

An FPGA Architecture Supporting Dynamically Controlled Power Gating

Assem A. M. Bsoul¹ and Steven J. E. Wilton²

Department of Electrical and Computer Engineering
University of British Columbia
2332 Main Mall, Vancouver, BC V6T 1Z4, Canada
{¹absoul, ²stevew}@ece.ubc.ca

Abstract—Leakage power is an important component of the total power consumption in FPGAs built using 90 nm and smaller technology nodes. Power gating, in which regions of the chip can be powered down, has been shown to be effective at reducing leakage power. However, previous techniques focus on statically-controlled power gating. In this paper, we propose a modification to the fabric of an FPGA that enables dynamically-controlled power gating, in which logic clusters can be selectively powered-down at run-time. For applications containing blocks with large idle times, this could lead to significant leakage power savings. Our architecture utilizes the existing routing fabric and unused input pins of logic clusters to route the power control signals. No modifications to the existing routing algorithms are required to support the new architecture. We study the area and power tradeoffs by varying the basic architecture parameters of an FPGA, and by varying the size of the power gating regions. We also study the leakage energy savings using a model that characterizes an application in terms of its structure and behavior. We show less than 1% of area overhead for a power gating region size of 3X3 logic tiles. Using the application model, we show that up to 40% leakage energy reduction can be achieved using the proposed architecture for different application parameters, not including power dissipated by the power state controller.

I. INTRODUCTION

Field-programmable gate arrays (FPGAs) have become ubiquitous in applications such as telecommunications, digital signal processing, and scientific computing. In the mobile device market, however, FPGAs have had limited penetration, partially due to their high power consumption. Compared to application-specific integrated circuit (ASIC) implementations, FPGA implementations suffer 7-14X power overhead [1].

To bring reconfigurable technology to these handheld applications, new programmable devices that consume significantly less power are required. Many researchers have proposed techniques for reducing the power dissipation of FPGAs using techniques that have originally been applied to ASICs, including guarded evaluation, clock gating, power gating, dual supply voltages, and power-aware CAD optimization [2], [3], [4], [5]. Even after applying all these techniques, however, the power consumption of FPGAs remains prohibitive for some applications.

Previous techniques to reduce the power dissipation of FPGAs have focused on reducing both the dynamic and static (leakage) power of these devices. Dynamic power is dissipated in a circuit when it performs activity, while leakage power is dissipated when the circuit is idle. Leakage power has become

a major component of the total FPGA power consumption in deep submicron technologies. In [6], leakage power was found to be about 22% of the total FPGA power consumption for a 90-nm-based FPGA; this percentage is increasing for smaller technology nodes. In handheld devices, such as mobile phones, it is conceivable that the leakage power will be even more significant, since these devices are often used in an “always on” state, remaining idle except for short bursts of activity. Thus, low-leakage FPGAs are essential if they are to be used for these kinds of applications.

An effective way to reduce leakage power is to employ *power gating* [7]. The basic idea of power gating is illustrated in Figure 1(a). By connecting the supply voltage or the ground of a circuit component (an inverter in the figure) through a power gating transistor, also called a *sleep transistor* or a *power switch*, the circuit component can be turned on or off by turning the corresponding power switch on or off. When the power switch is turned off, the leakage current of the whole circuit is limited by that of the power switch, which significantly reduces the leakage power dissipated. A performance loss may result in because of the extra resistance that present in the current path. By sizing the power switch appropriately, an acceptable tradeoff between the performance and the amount of leakage power saving could be found. Obviously, adding the power switch incurs some area overhead. However, this overhead is small if the amount of circuit components that are power-gated by the same switch is large.

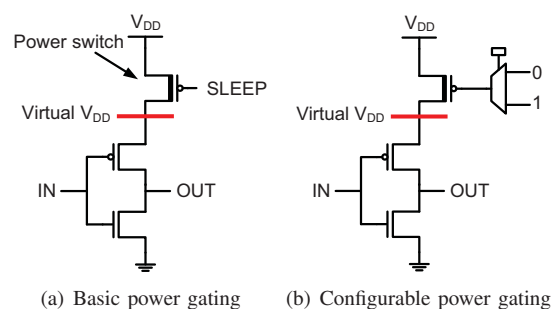


Fig. 1. Illustration of the basic idea of power gating

Previous proposals for power gating in FPGAs use configuration bits to control the power switches (as in Figure 1(b)) [8], [9], [4], [10]. We refer to this as *statically-controlled power*

gating, since once configured, the state of each part of the chip (on or off) does not change. Statically-controlled power gating is effective for FPGAs, since if the design does not fill an entire FPGA, the remainder of the FPGA can be safely turned off, saving leakage power. However, if only small amounts of the FPGA are not used, the savings from this technique may be limited.

In this paper, we propose *dynamically-controlled power gating* in an FPGA. In our architecture, the power switches can be turned on and off at *run-time* under control of other circuitry either running on the FPGA itself, or external to the FPGA. The signals to control the power switches are connected to the general-purpose routing fabric of the FPGA. For many mobile applications, dynamically-controlled power gating is very compelling. It is conceivable that such applications contain blocks that only need to operate occasionally. Turning off blocks when they are not needed can significantly reduce the leakage power of the device.

We evaluate the efficiency of our proposed architecture in terms of its area overhead and the amount of leakage power reduction that it can achieve by varying the basic FPGA architecture parameters, and by studying different granularity levels of the proposed architecture. We also present a simple model that represents an application mapped to our architecture in order to evaluate the effectiveness of our architecture for a range of application parameters. This paper does not address the tools required to map an application into our architecture. Instead, we focus on the tradeoffs related to the basic architecture and leave the issues related to application mapping as a future work.

The paper is organized as follows. Background on a tile-based FPGA architecture and related work on power gating for FPGAs is presented in Section II. In Section III, the proposed power gating architecture is described. In Section IV, we present the experimental procedure we use, and the area overhead and the leakage power results we obtain by varying the basic FPGA architecture parameters of the proposed architecture; we also present a simple application model, and we evaluate the advantages of our architecture using different parameters of the model. Finally, we conclude the paper and we discuss future work in Section V.

II. BACKGROUND

A. Related Work

Most of the related work in the literature focused on architectures to enable statically-controlled power gating for FPGAs. Lin et al. studied fine-grain power gating for FPGAs [8]; their study showed that the area overhead could be more than 100%; this is undesirable for FPGAs because the area overhead will eventually translate to longer wires, more capacitance, and therefore more power consumption, in addition to the corresponding increase in cost.

Gayasen et al. proposed coarse granularity power gating by using a power switch for a region of logic blocks [9], and they designed the required region-constrained placement algorithm. Although their technique is effective in reducing

leakage power for logic blocks, it does not consider power gating for the routing resources, which contribute to a large percentage of the total FPGA leakage power. In the same work, the authors suggested using dynamic reconfiguration to change the power state for the different parts in an FPGA to exploit the application behavior in order to reduce the dissipated leakage energy; however, the architecture to support this idea has not been provided.

Tuan et al. proposed power gating for an architecture similar to that of Xilinx Spartan-3 FPGAs [4]. Their architecture permits power gating for individual tiles; each tile is composed of a configurable logic block (CLB) and its associated switch matrix that connects it to neighboring CLBs. Their architecture supports a sleep mode by using a sleep signal from an off-chip controller that is connected to all power switches in the FPGA. This sleep signal could be supplied from an off-chip controller to control the power state for the whole chip or the parts of the chip that are configured to be controlled by that signal.

Bharadwaj et al. were the first to propose synthesizing a power state controller (PSC) from the data flow graph (DFG) of an application in order to exploit the idleness periods of the application to reduce the dissipated leakage energy [10]. They assumed a power gating architecture similar to that proposed in [9]. However, their work does not address power gating for routing resources, and more importantly, it does not address how the power switches will be connected to the routing resources in an FPGA to enable routing the power control signals generated from elsewhere on the device.

No previous work, to the best of our knowledge, has described an architecture to support dynamically-controlled power gating. Our work, however, is an extension to previous works because we assume that a power state controller (PSC) can be synthesized on-chip from a description of the application's behavior as in [10]. We also extend the statically-controlled power gating architecture proposed in [4] to enable flexible routing of power gating control signals to the different parts of an FPGA device.

B. Architectural Framework

In this paper, we assume a tile-based FPGA architecture [11]. In this architecture, an FPGA is composed of an array of tiles; each tile is composed of a logic cluster (LC) and the associated routing resources. Figure 2(a) shows a tile-based FPGA. An LC is composed of a number of basic logic elements (BLEs); each BLE is composed of a lookup table (LUT), a flip-flop, and a multiplexer to select between the combinational or the registered output of the BLE. A local switch matrix in the LC is typically included to support routing intra-cluster connections. Figure 2(b) shows an example LC composed of N BLEs.

Each LC is surrounded by routing channels (RCs) from its four sides. The intersection of two routing channels forms a switch box (SB) that can be configured to route the signals to different directions. Connections can be made from a routing channel that borders an LC from one of its sides to the input

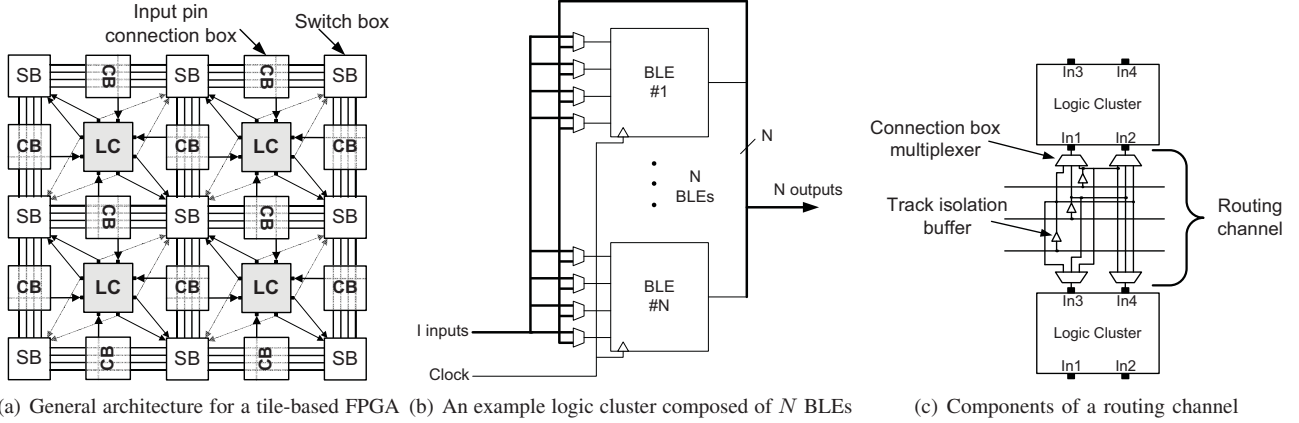


Fig. 2. Tile-based FPGA architecture

pins of the LC through configurable switches, called connection boxes (CBs). Buffers are typically inserted to isolate the load capacitance of the wires in the routing channel from the inputs of the connection boxes for performance issues, and they are shared among all connection boxes bounded by that specific routing channel. Figure 2(c) illustrates these components.

Finally, the outputs of an LC are connected directly to multiplexers in the switch boxes through isolation buffers. This is similar to the architectural assumptions made in the VPR 5.0 tool [12], which is more realistic than using output pin connection boxes.

III. DYNAMICALLY-CONTROLLED POWER GATING

Figure 3 shows an FPGA that has an application composed of two modules, M1 and M2. A power state controller (PSC) is synthesized from the information that describes the behavior of the application, such as the DFG of the application as has been discussed in [10]. This controller could exploit the idleness periods that the modules in the application may experience by turning off the logic in the idle modules, and turning them back on when they exit their idle periods. This requires routing *power control signals* from the controller to the logic blocks that support power gating.

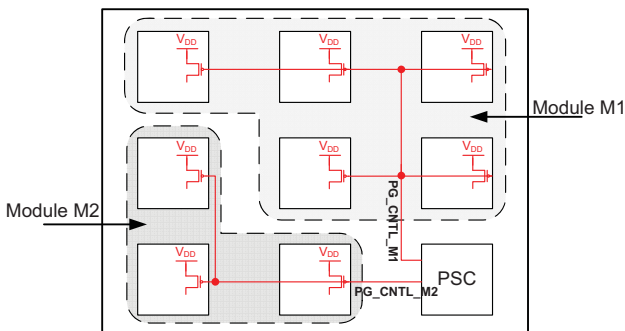


Fig. 3. Using a power controller to control the power state of the different modules in an application

Our architecture enables routing the power control signals

from elsewhere in the device to the power gating enabled components in order to realize the dynamic control of the power modes. This section presents the proposed dynamic power gating architecture for logic clusters and their bordering routing channels (connection boxes and track isolation buffers). We focus on these components because they dissipate about 50% of the total leakage power in an FPGA [9].

A. Fine-Grained Power Gating Architecture

Our basic power gating architecture enables dynamically controlling the power state of individual logic clusters and the routing resources in their bordering routing channels (input pin connection boxes and track isolation buffers) without modifying the routing architecture; therefore, no changes to the routing algorithms are required. The switch boxes in our architecture are always assumed to be powered on in order to enable flexible routing of connections, including the power control signals. Nevertheless, we believe that enabling dynamic control of the power state of switch boxes is crucial to enable complete dynamic control of the power state of a device; this feature will be added to our architecture in future work.

Figure 4 shows an example of the basic power gating architecture. In this figure, a logic cluster has four input pins, with the required four connection boxes, distributed uniformly on its four sides. Each of the connection boxes can be used either to route an endpoint of a connection to the corresponding input pin, or to route a power control signal to the cluster. If a power control signal is to be routed, then the corresponding input pin of the cluster is not used. The outputs of the connection boxes are fed as inputs to the *power gating multiplexer* of the LC. This multiplexer selects the input pin that will be used as the power control signal for the cluster and the bounding routing channels; this signal is labeled PG_CNTL1 in the figure. As shown in Figure 4(b), PG_CNTL1 could drive the gate of the sleep transistor to turn it off for low-leakage mode, or to turn it on for normal circuit activity.

The first advantage of this architecture is that there is a

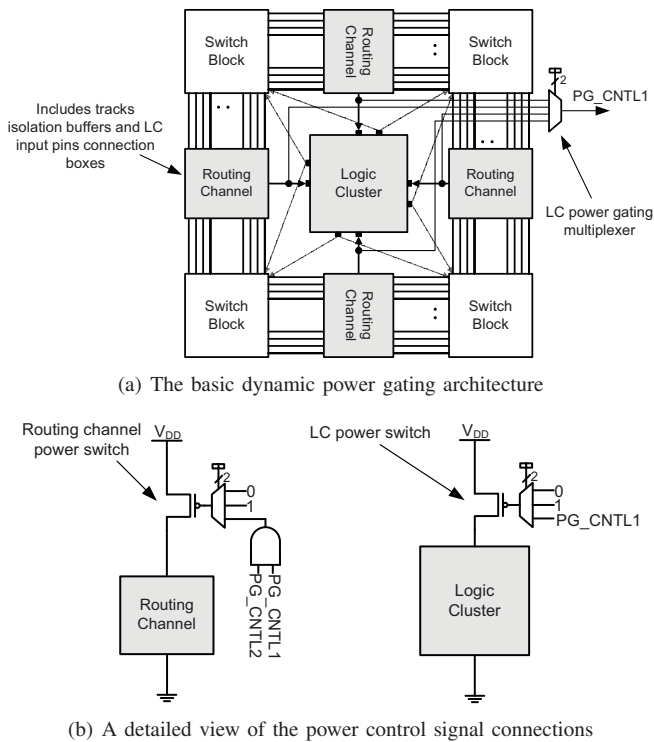


Fig. 4. Dynamic power gating architecture for a logic cluster and its routing channels

flexibility in using any of the connection boxes as the entry point for the power control signal. Therefore, such signal could be routed from, say, an on-chip power controller to the target logic clusters in the same way that conventional signals are routed in the original architecture. The second advantage is that we can utilize unused connection blocks to route the power control signal. Obviously, other variations of this architecture are possible where a subset of the connection boxes could be used to provide inputs to the power gating multiplexer instead of using all connection boxes.

For a logic cluster, the sleep transistor could be turned on or off statically, or controlled by the power control signal PG_CNTL1 . This is determined by the configuration bits for the 3:1 multiplexer that drives the logic cluster sleep transistor as shown in Figure 4(b). Similarly, routing channel sleep transistor could be turned on or off statically, or controlled by the power control signal.

The track isolation buffers in a routing channel are shared between the connection boxes of the two neighboring logic clusters (Figure 2(c)); therefore, it is important to not turn the routing channel off if either of the neighboring logic clusters is on. This is ensured in our architecture by ANDing the power control signals of the two logic clusters, namely PG_CNTL1 and PG_CNTL2 , as shown in Figure 4(b).

Enabling static control, as well as dynamic control, of the power state allows us to keep clusters and routing channels on or off all the time if the application requires them to be on all the time, or if they are not used by the application. Furthermore, a specific routing channel is required to be on

all the time if it is used as an access point for the power control signal; our architecture handles this as well.

Our architecture does not perform power gating for the configuration memory cells or for the flip-flops in a logic cluster; the configuration memory cells are typically implemented using a low leakage, high- V_{th} process, such as the medium oxide transistors used to implement configuration memory in some of the recent Xilinx devices [13]. Therefore, they consume a small amount of leakage power. Flip-flops within logic clusters are relatively small, and thus consume only a small amount of leakage power. Therefore, we keep them on all the time instead of using other state-saving mechanisms that would increase the area overhead.

B. Region Power Gating

The area overhead associated with our proposed architecture in Section III-A comes from the area of the LC and routing channels sleep transistors, the power gating multiplexer of an LC, the 3:1 multiplexers that drive the gate of the sleep transistors, the AND gates required to implement proper power gating for the routing channels, and the extra required SRAM configuration memory.

Typically in an application that is mapped to an FPGA, blocks that are part of the same functional module are placed close to each other in order to minimize a target cost function, such as the combined timing and wiring cost function used in the VPR placement and routing tool [14]. This means that a group of logic clusters and routing channels, that are spatially close to each other, could be power-gated by the same power control signal. Therefore, it is feasible to include the support for power gating at a coarser granularity level than at the level of individual logic clusters and routing channels.

Similar to [9], we propose using a region of logic clusters and their associated routing channels as the basic power gating enabled block in our architecture. This would reduce the area overhead and increase the amount of leakage power reduction compared to the cluster-level architecture presented in the previous section.

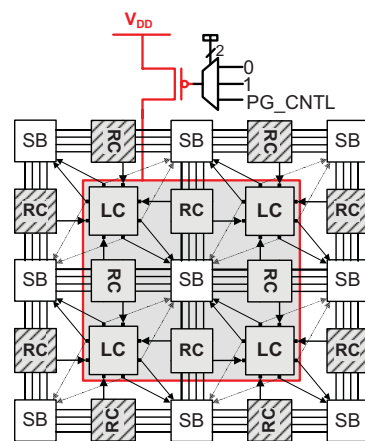


Fig. 5. An example region of the dynamic power gating architecture

Similar to the cluster-level architecture, the SRAM con-

figuration memory, flip-flops, and switch boxes are powered on all the time. Figure 5 shows an example dynamically-controlled power gating region. Controlling power for all of the gray-colored blocks is performed by one power control signal that could be routed through any of the routing channels that borders the region (hashed in the figure). The internal routing channels along with the logic clusters (un-hashed gray-colored blocks) are all supplied power through the same sleep transistor. The architecture for the bordering routing channels is exactly the same as the ones in the basic architecture shown in Figure 4(b). Hence, the case where neighboring regions have different power gating control signals is handled in a similar way as the case in the cluster-level architecture where two neighboring logic cluster could share the same routing channel.

C. Rush Current

When a module that is in sleep mode is turned on, the amount of current drawn from the power rail to charge the floating nodes could be large; this results in a voltage drop on the power rail. There are two consequences: the first is related to the resulting instability in the state of registers and SRAM cells and the performance of combinational paths in neighboring regions, and the second is related to the time the voltage bounce requires to settle down.

There are several solutions to this problem. Our current solution is to expose this to the application designer. The power state controller (PSC) which controls the power switches must be intelligent enough not to turn on large parts of the chip at once. Instead, the power state controller will follow a “staggered approach” in which small amounts of the chip are turned on sequentially.

This approach is somewhat unsatisfactory, as it requires the application designer to understand the intricacies of the power distribution capabilities, or requires these capabilities to be abstracted and guard-banded enough to be palatable to the designer. Two other approaches will be considered in future work: (1) the CAD tool will be informed of the capabilities of the power distribution network, and uses this information while synthesizing the logic making up the power state machine, or (2) a hardwired wake-up controller will be designed that turns on regions gradually, and provides a handshaking *completion* signal to the general-purpose logic; this signal could then be used by the PSC to start the operation of the newly turn-on block.

IV. EXPERIMENTAL PROCEDURE AND RESULTS

In this section we present the experimental procedure that we used to evaluate the proposed architecture and the results of our experiments. For experiments that involve varying architecture parameters, we report results for leakage power and area overhead with and without including switch boxes. Including the area and leakage power of switch boxes would give us a better estimate of what we would expect for a complete chip. All of our experiments evaluate the following three architectures:

- Ungated: this is the standard architecture that does not have support for any type of power gating.
- Static-gating: this is an architecture that supports statically-controlled power gating, such as the one presented in [4]. This architecture is similar to our architecture that we presented in Section III-A, but does not have the LC power gating multiplexer and the AND gates in the power gating circuit of the routing channels.
- Dynamic-gating: this is the dynamically-controlled power gating architecture that we proposed in Section III.

For the area overhead computations, we use an area model similar to the one used in the VPR placement and routing tool [11], [15], which is based on transistor counts. We assume that SRAM cells are built using six minimum-sized transistors. All multiplexers used in our architecture are NMOS-based, two-stage multiplexers that use a combination of decoded and encoded stages, similar to the ones assumed in the VPR placement and routing tool, followed by a level restorer and buffer. For the routing architecture, we assume routing channels with wire segments of length one; this assumption is made to reduce the complexity of generating the SPICE netlists for switch boxes using our automatic tools. All routing channels are unidirectional [16]. The outputs of logic clusters connect directly to the switch boxes through isolation buffers without the need for output pins connection boxes, similar to the architecture assumptions made in the VPR 5.0 tool [12].

For the FPGA architecture generation, we assume a switch box flexibility $F_s = 3$, input pin connection box flexibility $F_{c,in} = 0.2$, and output pin connection box flexibility $F_{c,out} = 0.1$. For HSPICE simulations, we use the 45 nm technology model from the predictive technology model (PTM) website [17], and we assume a temperature $T = 85$ °C.

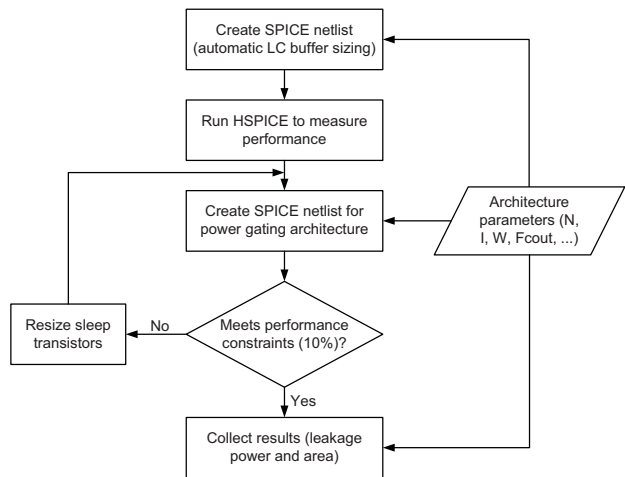


Fig. 6. Experimentation procedure used in the paper

We use the experimentation procedure shown in Figure 6. All of the SPICE netlists are generated automatically using scripts that we have developed for this purpose. When generating the power gating architecture, the sleep transistors are sized such that the timing performance is always constrained

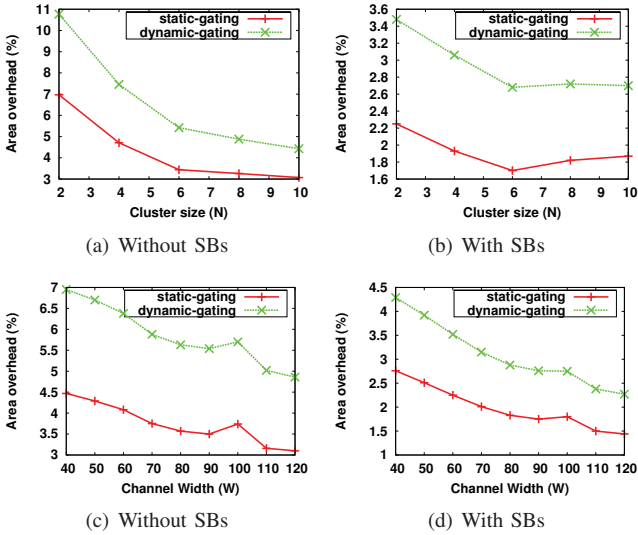


Fig. 7. Area overhead for cluster-level power gating

to be at most 10% slower than the ungated architecture. All inputs are assumed to be switching concurrently in order to account for the worst case activity.

A. Logic Cluster Power Gating

For the fine-grained power gating architecture, we vary two architecture parameters, namely the cluster size (N) and the width of the routing channels (W). In both cases, we also vary the number of input pins (I) of the logic cluster as a function of N . When varying N , we set $W = 90$; likewise, we set $N = 6$ when varying W .

1) *Area Overhead*: Figure 7 shows the area overhead as a function of the basic FPGA architecture parameters (N and W), for both the static and the dynamic power gating architectures. As the cluster size (N) increases ((a) and (b)), the area overhead decreases for both the static and the dynamic architectures. This is because the number of circuit components in a logic cluster that are power-gated by the same sleep transistor increases with the increase in N . We can see that when the switch box area is included in the area overhead calculations (as in (b)), the area overhead for both architectures is very small. Note that for $N \geq 6$ the area overhead is almost constant. This is due to the increase in the size of the sleep transistors in order to meet the performance constraints.

Similar observations can be made for varying W . As W is increased, the size of input pin connection boxes increase, which reduces the area overhead due to the power gating circuits.

In all cases, the area overhead for the dynamic architecture is slightly more than that for the static architecture. This is due to the additional circuit components that are required to enable dynamically controlling the power state of a logic cluster.

2) *Leakage Power*: Figure 8 shows the leakage power for a single logic cluster as a function of N and W . We can see that as N and W increase, the leakage power for the ungated architecture increases. Power gating becomes more effective in

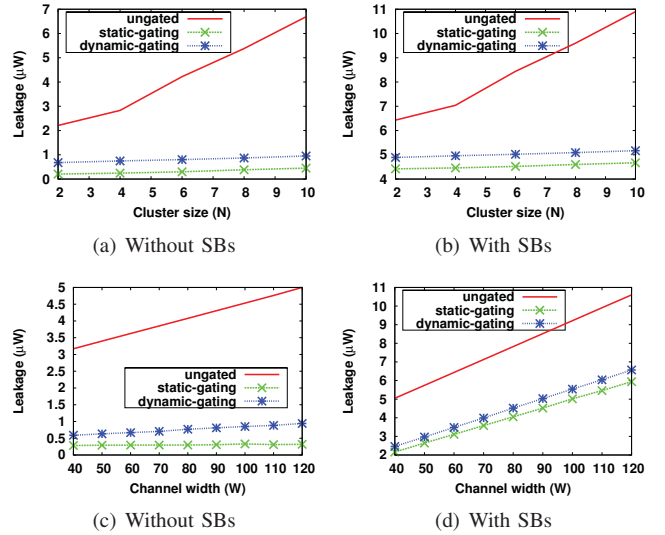


Fig. 8. Leakage power for a single logic tile

reducing leakage power as N and W increase. For example, for $N = 10$, leakage power reduction (without SBs leakage) is about 94% for the statically-controlled power gating architecture, and about 87% for the dynamically-controlled power gating architecture.

The leakage power for the dynamic architecture is slightly larger than that for the static architecture; this is due to the additional circuit components that are required to enable dynamically controlling the power state for the logic cluster. However, the difference is small compared to the savings that are possible if blocks can be dynamically powered down.

B. Region Power Gating

In this section we show how the area overhead and the leakage power are affected by the size of the region in the architecture described in Section III-B. We fix the basic architecture parameters to $N = 6$, $W = 90$, and $I = 16$, and we vary the region size (R). A region size R indicates that there are R LCs in the x - and y -dimensions of the region (meaning R^2 LCs in each region).

1) *Area Overhead*: As shown in Figure 9, the area overhead decreases as the region size increase. For example, for $R \geq 3$, the area overhead is less than 1% for both the static and the dynamic architectures when SBs are included in the area calculations.

This small area overhead for the region-level power gating is a result of using one sleep transistor for the whole region instead of using individual sleep transistors for each logic cluster; the size of the region sleep transistor is smaller than the sizes of the cluster-level sleep transistors. Another source for the small area overhead is that the internal routing channels of a region are power-gated using the sleep transistor of the region; therefore, they do not require their own sleep transistors.

Note that the area overhead for the dynamic architecture is slightly larger than that for the static architecture; this is

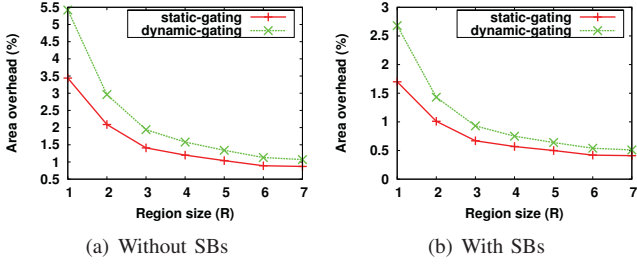


Fig. 9. Area overhead for region-level power gating

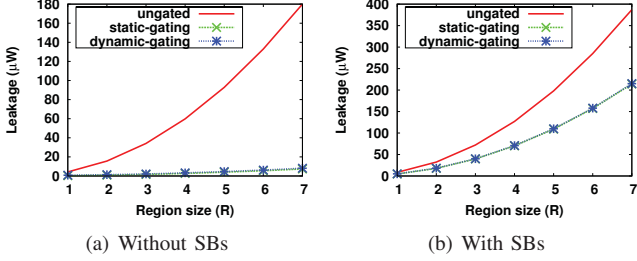


Fig. 10. Leakage power for a region of logic tiles

due to the additional circuit components required to enable dynamically controlling the power state for the region.

2) *Leakage Power*: Figure 10 shows the results for the leakage power of the three architectures (ungated, static-gating, and dynamic-gating). The gated architectures have a small leakage power compared to the ungated architecture. The reduction in leakage power ranges from 80% to 95% using the dynamic architecture, and about 93% to 95% using the static architecture.

The figure shows that as the region size increases, the amount of leakage power reduction due to power gating also increases. Note that the dynamic architecture has a slightly larger leakage power than the static architecture; this is a result of the additional circuit components required to support dynamically controlling the power state.

When switch boxes leakage is included in the leakage power of a region (Figure 10(b)), the savings in leakage power using the power gating architectures are smaller than the case in which SBs leakage is not included. This is because SBs contribute to a significant portion of the leakage power (about 50%). Nevertheless, the leakage reduction using power gating is very promising (about 45%) even when SBs are not included in the components that can be powered off.

C. Application Behavior

The preceding section showed that the area and leakage power overhead of our technique is small. The overall power savings that our technique provides depends very much on the application. Applications with blocks with long idle times would result in significant power savings, since large portions of the chip could be powered down for long periods. In this section, we present a simple model to characterize an application and we use it to evaluate the leakage energy that could be saved by using our architecture.

An application can be characterized by the percentage of logic clusters that could be power gated because they exhibit idleness periods, and the percentage of time they stay in the sleep mode. These two parameters could be derived from a more complex model that involves more details about the application structure and the behavior of the different modules during the different time periods.

The following definitions are used in our model.

- R_{device} : the number of regions in an FPGA device
- R_{psc} : the number of regions the power state controller occupies
- R_{app} : the number of regions that the application occupies
- F_{ir} : idle regions, the fraction of R_{app} that can be power gated because they go through some idle periods
- F_{it} : idle time, fraction of the application execution time in which $F_{ir} \times R_{app}$ regions are idle
- L_{ug} : leakage power of the ungated architecture
- UL_{sg} : ungating, i.e., *not* in sleep mode, leakage power of a region of the static gating architecture
- GL_{sg} : gating, i.e., in sleep mode, leakage power of a region of the static gating architecture
- UL_{dg} : ungating, i.e., *not* in sleep mode, leakage power of a region of the dynamic gating architecture
- GL_{dg} : gating, i.e., in sleep mode, leakage power of a region of the dynamic gating architecture

Let R_{sg} denote the number of regions that could be put into sleep mode at configuration time because they are not used by the application, and R_{dg} denote the number of regions that could be turned off during application execution for F_{it} of the application running time. We can calculate these two values as follows:

$$R_{sg} = R_{device} - (R_{psc} + R_{app}) \quad (1)$$

$$R_{dg} = \lfloor F_{ir} \times R_{app} \rfloor \quad (2)$$

Let T_{app} be the application execution time, and T_{app}^* be the application execution time when a gated architecture is used, i.e., accounts for performance degradation due to using sleep transistors. Then, we can calculate the dissipated leakage energy for an application for the three architectures using the following equations:

$$E_{ungated} = (R_{sg} \times L_{ug} + R_{dg} \times L_{ug} \times F_{it}) \times T_{app} \quad (3)$$

$$E_{static} = (R_{sg} \times GL_{sg} + R_{dg} \times UL_{sg} \times F_{it}) \times T_{app}^* \quad (4)$$

$$E_{dynamic} = (R_{sg} \times GL_{dg} + R_{dg} \times GL_{dg} \times F_{it}) \times T_{app}^* \quad (5)$$

The above model uses the simplification that the number of regions that could be put in sleep mode dynamically can be determined from the application structure (R_{app} and R_{psc}) and behavior (F_{it} and F_{ir}). To simplify the equations, no parameters account for the power dissipation of the PSC in the case of the dynamic gating architecture; we believe that the power dissipation of the PSC will be very small and does not affect the results significantly. Note that we do not model the required wakeup time for the idle blocks during their state transition; this time is largely dependent on how rush current is handled, and will be part of our future work.

We applied the model to the *clma* circuit from the MCNC benchmark suite. We assumed a device size such that the number of utilized logic blocks is 70%. We assumed a PSC size of 5% of the number of logic blocks in the application. Furthermore, we use the following parameters for the architecture generation and HSPICE simulations: $R = 3$, $W = 90$, $N = 6$, and $I = 16$; the other parameters are similar to the ones used in previous sections.

Figure 11 shows the leakage energy reduction in percent (z -axis) for both the statically- and the dynamically-controlled power gating architectures compared to the ungated architecture. The X -axis represents F_{ir} and the Y -axis represents F_{it} . We can see that the static architecture gives good results for a small range of values when F_{ir} and F_{it} are small. This range corresponds to the case where the regions that would be occupied by the PSC in the dynamic architecture are not used in the static architecture; hence, they are turned off. On the other hand, the dynamic architecture shows a consistent leakage energy reduction for a large range of values for F_{ir} and F_{it} . The leakage energy reduction obtained is about 40%. Note that these numbers takes into consideration the leakage of switch boxes, which are not put into sleep mode in our architecture. If we ignore the switch blocks, our architecture is capable of saving more than 90% of the leakage energy.

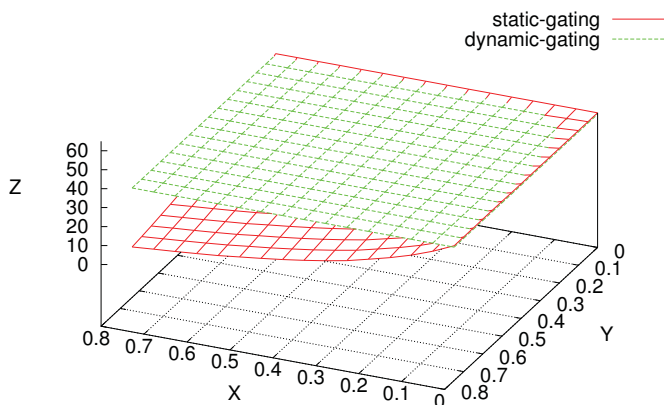


Fig. 11. Leakage energy reduction (z -axis) for *clma* as a function of F_{ir} (x -axis) and F_{it} (y -axis)

V. CONCLUSION AND FUTURE WORK

Leakage power is an important component of the total power consumption in FPGAs built using 90 nm and smaller technology nodes. Power gating has been proposed in the literature for FPGAs to enable turning off unused resources. However, the current available methods do not enable dynamic control of the power state for the different parts in an FPGA during the execution of the application. This capability is desirable in applications that show long idle times, such as mobile applications, in order to exploit idleness periods to turn off idle modules, thus reducing their leakage power.

In this paper, we presented an FPGA architecture that enables dynamic control for the power state of the logic clusters and the routing channels. Our architecture does not

require any changes to the routing algorithms and utilizes existing input pins of logic clusters to route power control signals. We showed that the leakage power reduction and the area overhead results are promising. Our future work will involve extending dynamic control of the power state to include switch boxes because they consume significant amount of the leakage power in an FPGA, and handling wakeup rush current.

REFERENCES

- [1] I. Kuon and J. Rose, "Measuring the Gap Between FPGAs And ASICs," in *Proceedings of the 14th ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2006, pp. 21–30.
- [2] M. Münch, B. Wurth, R. Mehra, J. Sproch, and N. Wehn, "Automating RT-level Operand Isolation to Minimize Power Consumption in Datapaths," in *Proceedings of the Conference on Design, Automation and Test in Europe*, 2000, pp. 624–633.
- [3] Q. Wang, S. Gupta, and J. H. Anderson, "Clock Power Reduction for Virtex-5 FPGAs," in *Proceeding of the 17th ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2009, pp. 13–22.
- [4] T. Tuan, S. Kao, A. Rahman, S. Das, and S. Trimberger, "A 90 nm Low-Power FPGA for Battery-Powered Applications," in *Proceedings of the 14th ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2006, pp. 3–11.
- [5] F. Li, Y. Lin, L. He, and J. Cong, "Low-Power FPGA using Pre-Defined Dual-Vdd/Dual-Vt Fabrics," in *Proceedings of the 12th ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2004, pp. 42–50.
- [6] (2005, August) Stratix II vs. Virtex-4 Power Comparison & Estimation Accuracy. Altera Corp. white paper WP-S20805-01 (v1.0). Altera Corp. [Online]. Available: www.altera.com/literature/wp/wp_s2v4_pwr_acc.pdf
- [7] S. Henzler, *Power Management of Digital Circuits in Deep Sub-Micron CMOS Technologies (Springer Series in Advanced Microelectronics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.
- [8] Y. Lin, F. Li, and L. He, "Routing Track Duplication with Fine-Grained Power-Gating for FPGA Interconnect Power Reduction," in *Proceedings of the 2005 Asia and South Pacific Design Automation Conference*, 2005, pp. 645–650.
- [9] A. Gayasen, Y. Tsai, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, and T. Tuan, "Reducing Leakage Energy in FPGAs Using Region-Constrained Placement," in *Proceedings of the 12th ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2004, pp. 51–58.
- [10] R. P. Bharadwaj, R. Konar, P. T. Balsara, and D. Bhatia, "Exploiting Temporal Idleness to Reduce Leakage Power in Programmable Architectures," in *Proceedings of the 2005 Asia and South Pacific Design Automation Conference*, 2005, pp. 651–656.
- [11] V. Betz, "Architecture and CAD for Speed and Area Optimization of FPGAs," Ph.D. dissertation, Department of Electrical and Computer Engineering, University of Toronto, 1998.
- [12] J. Luu, I. Kuon, P. Jamieson, T. Campbell, A. Ye, W. M. Fang, and J. Rose, "VPR 5.0: FPGA CAD and Architecture Exploration Tools with Single-Driver Routing, Heterogeneity and Process Scaling," in *Proceeding of the 17th ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2009, pp. 133–142.
- [13] M. Klein. (2009, April) Power Consumption at 40 and 45 nm. Xilinx, Inc. white paper WP298 (v1.0). Xilinx, Inc. [Online]. Available: http://www.xilinx.com/support/documentation/white_papers/wp298.pdf
- [14] A. Marquardt, V. Betz, and J. Rose, "Timing-Driven Placement for FPGAs," in *Proceedings of the Eighth ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2000, pp. 203–213.
- [15] V. Betz and J. Rose, "VPR: A New Packing, Placement and Routing Tool for FPGA Research," in *Proceedings of the 7th International Workshop on Field-Programmable Logic and Applications*, 1997, pp. 213–222.
- [16] G. Lemieux, E. Lee, M. Tom, and A. Yu, "Directional and Single-Driver Wires in FPGA Interconnect," in *Proceedings of the IEEE International Conference on Field-Programmable Technology*, 2004, pp. 41–48.
- [17] "Predictive Technology Model (PTM)," <http://www.eas.asu.edu/~ptm/>.