An FPGA Architecture and CAD Flow Supporting Dynamically Controlled Power Gating

Assem A. M. Bsoul, *Student Member, IEEE*, Steven J. E. Wilton, *Senior Member, IEEE*, Kuen Hung Tsoi, and Wayne Luk, *Fellow, IEEE*

***Abstract*— Leakage power is an important component of the total power consumption in field-programmable gate arrays (FPGAs) built using 90-nm and smaller technology nodes. Power gating was shown to be effective at reducing the leakage power. Previous techniques focus on turning OFF unused FPGA resources at configuration time; the benefit of this approach depends on resource utilization. In this paper, we present an FPGA architecture that enables dynamically controlled power gating, in which FPGA resources can be selectively powered down at run-time. This could lead to significant overall energy savings for applications having modules with long idle times. We also present a CAD flow that can be used to map applications to the proposed architecture. We study the area and power tradeoffs by varying the different FPGA architecture parameters and power gating granularity. The proposed CAD flow is used to map a set of benchmark circuits that have multiple power-gated modules to the proposed architecture. Power savings of up to 83% are achievable for these circuits. Finally, we study a control system of a robot that is used in endoscopy. Using the proposed architecture combined with clock gating results in up to 19% energy savings in this application.**

***Index Terms*— Computer aided design, field-programmable gate arrays, power gating, static/leakage power.**

I. Introduction

FIELD-PROGRAMMABLE gate arrays (FPGAs) have become ubiquitous in applications, such as telecommunications, digital signal processing, and scientific computing. In the mobile devices market, however, FPGAs have had limited penetration, partially due to their high power consumption. Compared with application-specific integrated circuit (ASIC) implementations, FPGA implementations consume 12× more power on average [1]. To bring reconfigurable technology to these hand-held applications, new programmable devices that consume significantly less power are required.

Many researchers have proposed techniques for reducing the power dissipation of FPGAs based on the methods that have originally been applied to ASICs, including guarded evaluation, clock gating, power gating, dual supply voltages, and power-aware CAD optimization [2]–[5]. Even after applying all these techniques, the power consumption of FPGAs remains prohibitive for some applications.

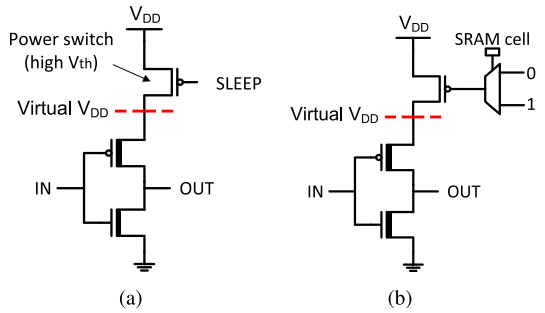

Fig. 1. Basic idea of power gating. (a) Basic power gating. (b) Configurable power gating.

Previous techniques to reduce the power dissipation of FPGAs have focused on reducing both the dynamic and the static (leakage) power of these devices. Dynamic power is dissipated due to charging and discharging of the circuit's capacitance, while leakage power is dissipated when the circuit is idle. Static power dissipation is a major component of the total power consumption in reconfigurable devices based on the sub-90-nm CMOS technology nodes. Recent reports from FPGA vendors indicate that FPGAs built on a 28-nm technology have roughly equal amounts of dynamic and static power [6], [7]. In handheld devices, 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 [8]. As shown in Fig. 1(a), by connecting the supply voltage or the ground of a circuit component 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 is limited by that of the power switch. A performance loss may result because of the extra resistance in the current path. By sizing the power switch appropriately, an acceptable tradeoff between the performance, power savings, and area can be found.

Previous proposals for power gating in FPGAs use configuration bits to control the power switches [as in Fig. 1(b)] [4], [9]–[11]. We refer to them as statically

Manuscript received June 11, 2014; revised September 27, 2014; accepted December 28, 2014. Date of publication February 12, 2015; date of current version December 24, 2015. The work of A. A. M. Bsoul and S. J. E. Wilton was supported in part by Altera Toronto Technology Center, Toronto, ON, Canada, and in part by the Natural Science and Research Council of Canada. The work of K. H. Tsoi and W. Luk was supported in part by the U.K. Engineering and Physical Sciences Research Council and in part by the European Union Seventh Framework Programme under Grant 257906, Grant 287804, and Grant 318521.

A. A. M. Bsoul and S. J. E. Wilton are with the Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, BC V6T 1Z4, Canada (e-mail: absoul@ece.ubc.ca; stevew@ece.ubc.ca).

K. H. Tsoi and W. Luk are with the Department of Computing, Imperial College London, London SW7 2AZ, U.K.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2015.2393914

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 a small number of resources in an FPGA are not used, the savings from this technique may be limited.

In this paper, we propose dynamically controlled power gating (DCPG) in an FPGA. In our architecture, the power switches can be turned ON and OFF at run-time under the 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.

This paper is based on [12] and [13]. The work in [12] focuses on power gating for logic resources in FPGAs, and the work in [13] focuses on power gating for coarse-grained routing resources. Our main additional contributions in this paper are as follows.

1) We propose fine-grained power gating for routing resources. This allows powering down a larger number of routing resources at configuration time, and enables dynamic power state control for a larger number of routing resources at run-time.
2) We present a CAD flow that can be used to map the application circuits that contain power-gated modules to the proposed architecture. In this flow, power control signals are connected to the different power-gated resources to control their power state at run-time using the existing general purpose routing fabric of an FPGA.
3) We propose enhancements to an FPGA routing algorithm that try to minimize the number of routing resources that cannot be powered down at run-time.
4) The presented CAD flow is used to evaluate the best granularity of routing resources power gating.
5) We evaluate a robot control system used in medical applications using the power gating architecture proposed in this paper, and we study its power savings for different operation activities.

We evaluate the 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 architecture granularity levels. We also use the proposed CAD flow to evaluate the potential power savings in a set of synthetic benchmark circuits, in addition to the robot control system mentioned above.

This paper is organized as follows. Section II provides overview of related works, and describes the FPGA architecture model adopted in this paper. Section III describes the proposed DCPG FPGA architecture. Section IV describes the proposed CAD flow and the enhancements to the routing algorithm to maximize the number of resources that can be turned OFF. In Section V, we describe the different benchmark circuits used to evaluate the proposed architecture. Finally, in Section VI, we experimentally evaluate the proposed architecture.

II. Background

A. Related Work

Lin *et al.* [9] studied fine-grained power gating for FPGAs to turn OFF unused resources at configuration time; their study showed that the area overhead could be >100%, which is undesirable because of the associated degradation in power and timing, and the increase in cost.

Gayasen *et al.* [10] proposed coarse-grained power gating using a power switch for a region of logic blocks. The use of dynamic reconfiguration was suggested to change the power state for the different regions in an FPGA based on their activity. However, this incurs power overhead and can only be applied at a very coarse granularity.

Tuan *et al.* [4] proposed power gating for an architecture similar to the Xilinx Spartan-3. Their architecture supports sleep mode using a sleep signal from an off-chip controller that is connected to all power switches in the FPGA; this scheme allows creating one controllable power domain only.

Bharadwaj *et al.* [11] proposed synthesizing a power state controller from the data flow graph of an application; this controller could exploit the idleness periods of the application to reduce the dissipated leakage energy in an FPGA. They used the same architecture in [10].

Li *et al.* [14] proposed using a power control hard macro that is associated with each tile in an FPGA to control its power state (clock and power gating). They assume a power gating architecture similar to that in [12].

Hoo *et al.* [15] proposed fine-grained power gating for switch blocks (SBs) and a routing algorithm to optimize the power savings. The proposed architecture, however, only supports powering down unused switches at configuration time.

Dynamic partial reconfiguration is also reported to reduce the static power at run-time by enabling time sharing of FPGA resources [6]. However, swapping reconfigurable modules happens at the scale of milliseconds, which may result high power overhead. In contrast, the proposed architecture enables changing the power state at the scale of nanoseconds.

B. Architecture Framework

In this paper, we assume a tile-based FPGA architecture [16]. An FPGA is composed of an array of tiles; each tile is composed of a logic cluster (LC) and the associated routing resources [two routing channels (RCs) and a SB], as shown in Fig. 2. An LC is composed of a number of basic logic elements (BLEs); each BLE is composed of a lookup table (LUT), a flip-flop (FF), and a multiplexer to select between the combinational or the registered output. A local switch matrix in the LC is typically included to support routing intracluster connections. Fig. 2 shows an LC composed of N BLEs.

Each LC is surrounded by RCs from its four sides. The intersection of two RCs forms a SB that can be configured to route the signals to the different directions. Fig. 3 shows examples of the connections for switches in an SB. A connection from a RC that borders an LC to one of its input pins can be made through configurable switches,

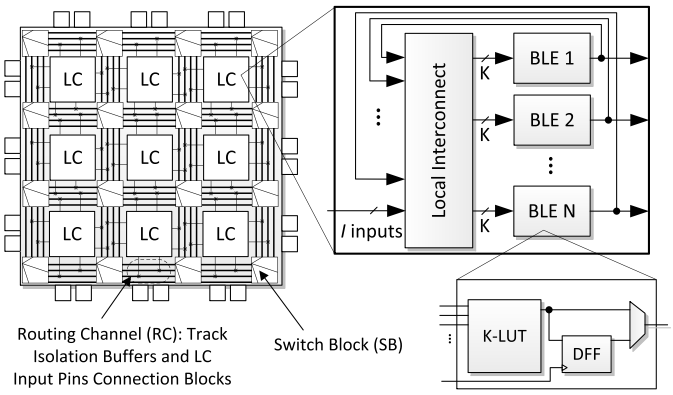

Fig. 2. Tile-based FPGA architecture.

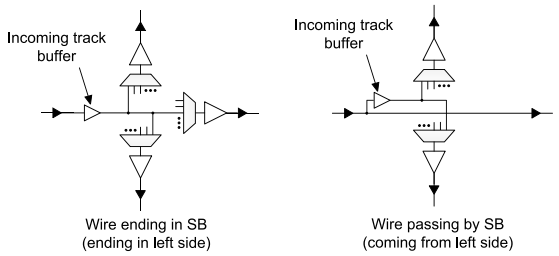

Fig. 3. Example incoming wire tracks and switches in an SB.

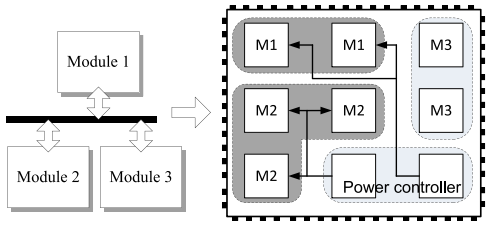

Fig. 4. Example application mapped to an FPGA supporting DCPG.

called connection boxes (CBs). Buffers are typically inserted to isolate the load capacitance of the wires in the RC from the inputs of the CBs for performance issues, and they are shared among all CBs bounded by that specific RC. Finally, the outputs of an LC are connected directly to multiplexers in the SBs through isolation buffers. This is similar to the architectural assumptions made in the VPR 5.0 tool [17].

III. Proposed DCPG FPGA Architecture

Fig. 4 shows an example system of three modules that are mapped to an FPGA that supports DCPG. Each module occupies a number of power gating regions (PGRs) in the target FPGA architecture. Each PGR is composed of a number of FPGA tiles; the number of tiles in a region dictates the architecture granularity. The power state of each PGR can be configured as always-ON, always-OFF, or dynamically-controlled (DC). As will be explained later in this section, the power state for some of the internal components of a PGR can be configured to a different power state than that for the encapsulating PGR.

In this example, two of the functional modules, M1 and M2, experience long idle periods, thus it is desired to power

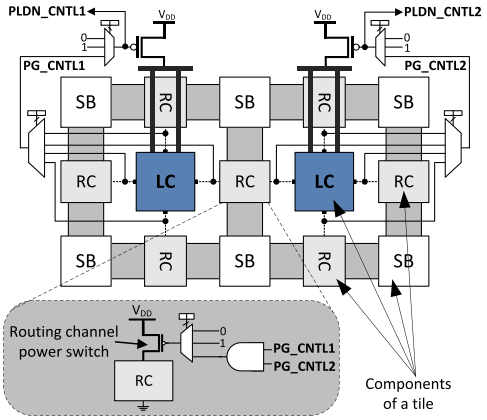

Fig. 5. Basic fine-grained DCPG for two neighboring tiles. Shared bordering RCs have their own power gating circuit.

them down during these times to reduce the leakage power consumption. The power state of the PGRs of M1 and M2 is configured to DC, which allows controlling their power state at run-time. Power control signals are routed from a power controller module to control the power states of modules M1 and M2. The third module, M3, does not experience idle periods, thus its power state is configured to be always-ON. Similarly, the power state for the power controller is configured to be always-ON. The power state for routing resources that are used to route the power control signals is configured to always-ON.

Power gating a module is beneficial if the energy consumed during its idle periods is larger than the overhead of applying power gating. This overhead results from the energy consumed by the power controller and during power state transitions.

The proposed architecture enables realizing power domains with different temporal (idle/active periods) and spatial characteristics (sizes and locations), thus it is suitable for a wide range of applications. There is no need to have fixed tracks in the FPGA fabric to work as power control signals; rather, power control signals can be routed on the preexisting FPGA routing fabric similar to any other user signal. The following sections describe the details of this architecture.

A. Basic Power Gating Architecture

In this section, we describe a fine-grained version of the proposed power gating architecture. Fig. 5 shows two tiles of an FPGA; some details are not shown for the sake of clarity. The basic power gating architecture supports power gating at the granularity of one tile, thus a PGR is one tile. The power state can be set by configuring the SRAM cells that control the select lines of the 3:1 multiplexers that drive the power switches. The novelty of the proposed architecture lies in its support for controlling the power state of individual LCs and the routing resources (input pin CBs, track isolation buffers, and SBs) dynamically at run-time. This makes the proposed power gating scheme suitable for various tile-based FPGA architectures.

The supported power modes are always-ON, always-OFF, and DC. The always-ON mode sets the resources in a powered state. This is useful for resources that need to be available all

the time, such as power control signals or application modules that do not experience idle times. The always-OFF power state puts the resources in sleep and in low-leakage mode. This is useful for resources that are not utilized by the application that is mapped to the device. DC means that the power state of a resource can be controlled at run-time, and can be changed by changing the value on the power control signal.

As shown in Fig. 5, one of the bordering input pins of each LC can be used to route the power control signal to the power switch (control signals are labeled PG_CNTL1 and PG_CNTL2). If an LC's power state is configured as DC, its power gating multiplexer (the 4:1 multiplexer in the figure) is used to route the power control signal by configuring the SRAM cells that control the select lines of the 4:1 multiplexer. If an LC's input pin is used to route the power control signal, then it cannot be used by the logic implemented in the LC. Variations to this organization where a subset of the input pins are used as inputs to the power gating multiplexer can be realized. However, this makes it harder to route the power control signals, since a smaller set of routing tracks can be used to route the control signals.

For correct operation of the DC mode, the power state of the RC that is used to route the power control signal must be configured as always-ON. To support this, separate power gating circuitry is used for the bordering RCs of an LC. The lower part of Fig. 5 shows the details. When configured to the DC mode, the AND gate ensures that the shared RC is turned OFF only when both neighboring LCs are turned OFF (when PG_CNTL1 = 1 and PG_CNTL2 = 1). This ensures that any of the bordering RCs of an LC can be used as the entry point for the power control signal. Therefore, such a signal could be routed from an on-chip power controller to the target LCs in the same way that any other user circuit signal is routed.

The same power control signal can be routed to any number of LCs that belong to the same power-gated module, which forms one power domain. The SBs' power state can be configured in the same manner discussed above. More details about SBs power gating will be discussed later in this section.

In the proposed architecture, the configuration memory cells and the FFs in an LC are not power gated. The configuration SRAM cells are typically implemented using a low leakage, high-V_{th} process, such as the medium oxide thickness transistors used in configuration SRAM cells in some commercial FPGAs [18]. The area of FFs within an LC is relatively small, and thus they consume only a small amount of leakage power. Therefore, these components are kept ON all the time instead of using other state-saving mechanisms that would increase the architecture complexity and power consumption.

Fig. 6 shows the details of an LC. Pull-down nMOS transistors are used to isolate the outputs of the LC when the LC is in sleep mode. This prevents large short-circuit current in SB buffers that are driven by the LC outputs. Similarly, the inputs of the FFs inside an LC are also isolated to prevent large short-circuit current in the FFs. Notice that we assume that clock gating is used in association with DCPG mode during the idle times; this guarantees that the values stored in the FFs do not get corrupted during sleep mode. The pull-down

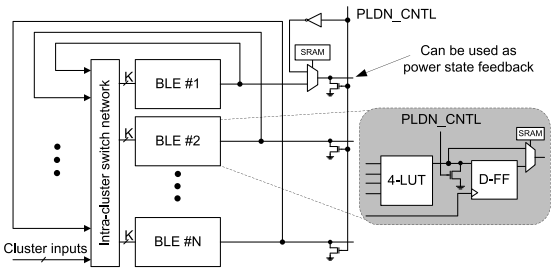

Fig. 6. Internals of an LC with DCPG showing pull-down nMOS devices at FF input and LC outputs. A status signal to indicate completion of power transition can be routed through one of the LC's outputs (the top).

nMOS transistors are controlled using the output of the 3:1 multiplexers that drive the power switch.

The architecture also provides a feedback signal to indicate that a power domain has completed a power transition. Fig. 6 shows that an inverted version of PLDN_CNTL, which is the output of the related 3:1 multiplexer shown in Fig. 5, can be routed through one of the LC's outputs. This is done using a 2:1 multiplexer at the output of BLE #1 inside the LC. The SRAM cell that controls the select line of the 2:1 multiplexer is configured to choose the feedback signal (PLDN_CNTL) or the normal output of BLE #1. If the feedback signal is selected to be routed to the output of the LC, the output of BLE #1 can only be used internally in the LC. Note that since only one feedback signal might be needed for a power-gated module, at most one BLE in a power-gated module might be unusable. Timing analysis can be used to determine that LC is the last one to be turned ON/OFF in a module, and it can be used to send the feedback signal.

B. Coarse-Grained Power Gating

The area and power overheads associated with the architecture in Section III-A are due to the sleep transistors of the LC and the RCs, the power gating multiplexer of the LC, the 3:1 multiplexers that drive the gate of the sleep transistors, the AND gates required to implement a proper power gating for the RCs, and the additional SRAM configuration memory cells.

Typically, when an application is mapped to an FPGA, blocks that are part of the same functional module are placed close to each other in order to minimize delay and wiring costs [19]. Thus, it is likely that a group of LCs and RCs that are spatially close to each other share the same power state. It is, therefore, feasible to support power gating at a coarser granularity level than what is described in Section III-A in order to reduce the area and power overheads of the power gating circuitry.

The concept of coarse-grained PGRs is presented here. Unlike the fine-grained architecture in Section III-A where each PGR is composed of only one tile, we propose a coarse-grained architecture, in which a PGR is composed of a number of tiles. Similar to the tile-level architecture in Section III-A, the SRAM configuration memory cells and FFs are powered on all the time.

Fig. 7 shows an example DCPG (PGR) of size 2×2 tiles. Some details are omitted for clarity. The region's LCs and

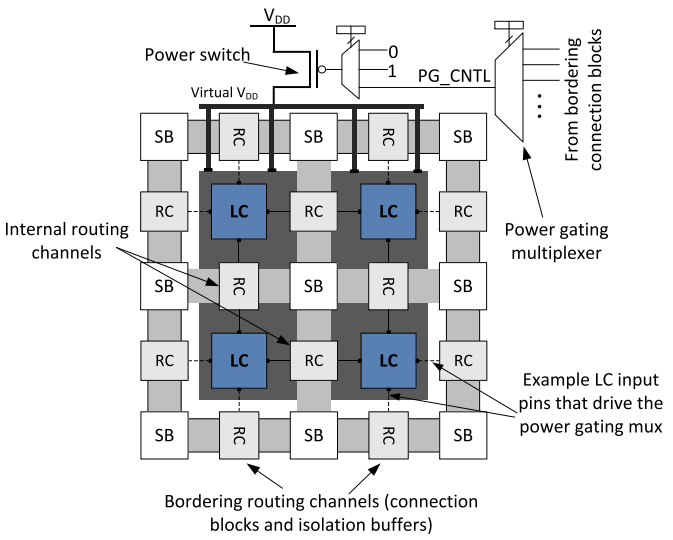

Fig. 7. Example PGR of 2×2 tiles. Internal region's LCs and CBs share the power switch. PGR's SBs and bordering RCs have individual power switches.

internal RCs, within the large, dark box in the figure, share the same power switch; thus, their power state can be configured as one unit. The region's SBs and bordering RCs have their own power switches and their power states can be configured separately; however, their power switches can still be controlled using the region's control signal (labeled PG_CNTL in the figure) when they are configured as DC. The bordering RCs in the coarse-grained PGRs have the same structure and functionality described for the RCs in Fig. 5; they can be used to route the power control signal to a PGR.

Different PGR sizes can be realized in the same manner. For example, a 3×3 PGR consists of 3×3 tiles (this PGR has 12 bordering RCs). Larger PGRs make it more challenging for the CAD tools to group related blocks in the same PGR, resulting in smaller power savings. On the other hand, the area and power overheads in smaller PGRs are larger. In terms of application mapping, a small PGR size means that an application occupies a larger number of PGRs; more routing resources would be needed to route the power control signals, which may negatively impact routability and requires more always-ON routing resources.

C. Coarse-Grained Power-Gated Switch Blocks

In the previous sections, we described the proposed power gating architecture for LCs and RCs (track isolation buffers and CBs). This section focuses on describing the power gating circuitry for SBs in a PGR.

The example PGR in Fig. 7 has a size of 2×2 tiles. The power control signal that is used to control the power state of the LCs region (PG_CNTL) is also used to selectively control the power state of the individual SBs that belong to the same region. For each LC, the SB that belongs to the same region as the LC lies in the right-bottom corner of that LC.

Fig. 8 shows the power gating circuitry for an SB. This circuitry is similar to that for the other components, as described in the previous sections. Minimum-sized pull-down nMOS transistors are placed at the outputs of the SB to pull

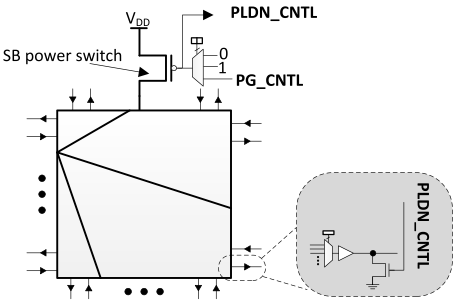

Fig. 8. Power gating circuit for a SB. SB outputs are pulled down to GND when the SB's power is OFF (in sleep mode).

TABLE I
CONFIGURABLE POWER MODES SUPPORTED BY THE DIFFERENT COMPONENTS IN A PGR

Internal Components [a]	Each Border RC	Each SB Partition
OFF	OFF / ON / DC[b]	OFF / ON / DC
ON	OFF / ON / DC	OFF / ON / DC
DC	OFF / ON / DC	OFF / ON / DC

[a] The PGR's internal LCs and RCs that share the same power switch.
[b] OFF = always-off, ON = always-on, DC = dynamically-controlled.

them to ground during the sleep mode to ensure proper output isolation. The gate input of the pull-down transistors is the same as the gate input to the power switch.

This scheme enables different power modes for different components in a PGR. Table I shows the supported power modes. For example, if the internal part of a PGR (LCs and internal RCs) is configured as DC, there is flexibility in configuring the power state for the individual SBs and bordering RCs. This flexibility allows some SBs to be always-ON to route important signals, such as power control signals or signals that connect between different modules.

D. Fine-Grained Power-Gated Switch Blocks

The power-gated SB architecture in Section III-C enables configuring an SB's power state as one unit. However, our experiments for many application circuits showed that >50% of the SBs' switches are not utilized. Supporting finer granularity power gating for SBs, therefore, may result in a larger number of switches that can be turned OFF either statically or dynamically at run-time compared with the coarse-grained SB power gating. This would result significant reduction of the total leakage power consumption, since an SB consumes ~70% of a tile's leakage power.

Fig. 9 shows how all switches in a specific SB side are grouped into one power gating partition to implement a finer granularity power gating. Partitions per side (PPS) is used as an architecture parameter to describe the power gating granularity of an SB. For example, PPS = 1 for the SB in Fig. 9. Increasing PPS results in finer granularity power gating. PPS = 0 represents an architecture where the power state for an SB is configured as one unit (coarse-grained power gating), while PPS = N_{switch} indicates the finest power gating granularity where the power state for each switch can

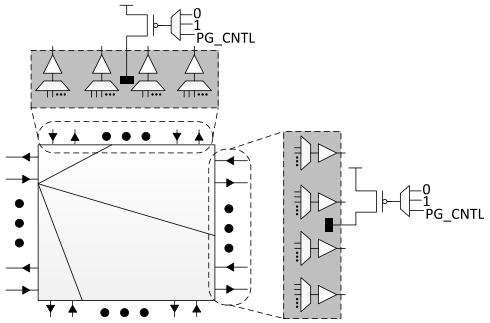

Fig. 9. One power gating partition per SB side (details for two sides are shown). The power state for each partition can be configured separately.

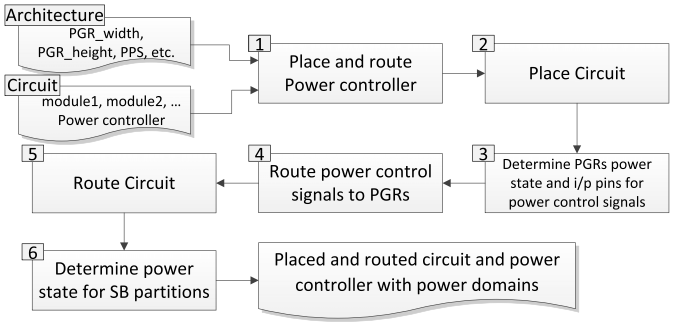

Fig. 10. CAD flow for the DCPG FPGA architecture.

be controlled individually. N_{switch} is the number of switches that exist in an SB's side. The cost of increasing PPS is the additional area and leakage power due to the additional power gating circuit components and the increase in the total effective sleep transistor size. In order to ensure correct operation for SBs when PPS > 0, the incoming tracks buffers (Fig. 3) must be always-ON, i.e., not power gated.

E. Inrush Current During Wakeup Phase

When a power-gated module is turned ON, a large current is drawn from the power grid lines in the chip to recharge the internal nodes of the FPGA circuitry. This current is known as inrush or wakeup current. If not handled appropriately, a large inrush current may cause malfunction of the design [20].

The work in [21], [22] describes a configurable architecture to solve the inrush current problem in FPGAs that support DCPG by staggering the turn ON phase of the PGRs in a power-gated module. The architecture in [21] and [22] can be used to solve the inrush current problem in the proposed architecture in this paper with small area and power overheads. The architecture provides short turn ON times. For example, turning ON a 1000 tiles module takes ∼10 clock cycles on a 300-MHz clock frequency, assuming 25 PGRs can be turned ON simultaneously and each PGR has a size of 4×4 tiles [22].

The inrush current handling architecture in [21] and [22] enables delaying the wakeup signal for each PGR using configurable and fixed delay elements. The timing for activating the isolation mechanism in our architecture (pull-down nMOS transistors) must be handled appropriately. When a PGR is turned OFF, isolation must be done before the rest of the PGR is powered down. On the other hand, when a PGR is turned ON, isolation must be deactivated after the PGR is powered up. To enable this, a 2:1 multiplexer can be used to drive the isolation activation signal (PLDN_CNTL). This multiplexer selects between the delayed or nondelayed power control signal. The select line can also be the power control signal.

IV. CAD Flow for DCPG FPGA Architecture

In this section, we present the CAD flow that is used to map applications to the proposed DCPG FPGA architecture. Our CAD flow is based on the VPR FPGA tool (version 5.0 [17]). The proposed flow focuses on low-level CAD steps, i.e., placement and routing. We assume that higher level tools will pass information in the netlist about the blocks that belong to a power-gated module and the power controller. We leave the discussion of higher level tools and their optimization to enable power gating for future work.

A. Placement and Routing

Fig. 10 shows the proposed CAD flow. The inputs to VPR include the circuit netlist, the names of the power-gated modules in the circuit, the power controller netlist, and the architecture parameters (PGR's width and height, and PPS).

The input netlists to the flow are generated using a CAD flow that is typically used with VPR. This includes Odin II for Verilog synthesis [23], ABC for technology mapping [24], and T-VPACK [25] for packing LUTs and FFs into LCs.

In Step 1, the power controller is placed and its internal connections are routed. The FPGA resources that are used by the power controller are locked, and their power state is set to always-ON. In Step 2, placement is performed for the application circuit. In Step 3, the power state for each PGR is determined based on the blocks that occupy the different LCs in the PGR. The power state for a PGR is set as follows.

1) *DC:* If only one power-gated module is mapped to the PGR's LCs.
2) *Always-OFF:* If all of the PGR's LCs are empty.
3) *Always-ON:* All other cases.

1) Routing Power Control Signals: The net for each power control signal is built in this step. The net's source is one of the outputs of the power controller that has already been placed. The sinks are found as follows. For each PGR that belongs to the power-gated module under consideration and is set to DC, a free input pin from its bordering RCs is selected (if one is available) to act as a sink for the control signal. Note that we cannot use predetermined sinks since the placement phase determines the number and locations of the PGRs in a power-gated module. In Step 4, the nets of the power control signals are routed. The SB partitions that are used to route these signals are set as always-ON to ensure that the power control signals are available all the time. Note that when selecting the sinks of the power control signals, we try to build a trunk-branch routing topology that minimizes the number of always-ON SB partitions as in [12].

2) Routing Circuit's Signals: In Step 5, the connections in the circuit netlist are routed on the available FPGA resources.

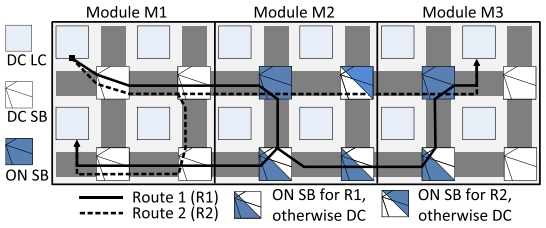

Fig. 11. Three power-gated modules placed on three PGRs with two ways shown to route the same net. The SBs used to route the net in M2 and M3 must be always-on (ON), other resources can be DC.

Although the power control signals are routed before the circuit's nets, this has negligible effect on routability and performance of the circuit because only a small fraction of the routing resources are used to route the control signals. In order to verify this, we mapped the circuits described in Section V-A to the proposed architecture with a PGR size of 4 × 4 tiles and PPS = 0. We found that the minimum channel width has increased by 2.3% on average, with a maximum increase of 16% for one circuit.

Finally, in Step 6, we determine the power state for each of the SB partitions as follows.

1) *DC:* If PGR is DC and only one power-gated module is routed through the SB partition.
2) *Always-ON:* SB partition is not used to route signals.
3) *Always-ON:* All other cases.

B. PG-Aware Routing

Due to the complexity of the routing topology of multiple-module power-gated circuits, some SB partitions are required to be always-ON. Fig. 11 shows an example of three power-gated modules mapped to three PGRs. Two possible ways to route the net from M1 to its sinks in M1 and M3 are shown (Route 1 and Route 2—R1 and R2). In both ways, SBs in M2 and M3 are required to be always-ON to ensure proper operation. For example, when M2 is powered down, the SBs in M2's PGR that route the net need to be powered.

In Fig. 11, R2 has a smaller number of always-ON SBs (larger number of always-OFF and DC SBs) compared with R1, which improves the power savings during the idle periods. In this section, we present the enhancements made to the router in order to increase the number of SBs that can be powered down. We modified the timing-driven router in VPR to implement these enhancements.

VPR uses the pathfinder negotiated congestion-delay router [16]. The routing resources are represented by a routing-resource graph. In this graph, nodes represent wire segments and logic block pins, and edges represent switches. In the inner loop of the algorithm, when searching for a route from a source node to a sink node, nodes of the graph are visited and added with a cost value (path cost) to a priority queue. These nodes are used later to iteratively investigate other nodes connected to them until the sink is reached. The path cost to reach a node from the source of the net is the sum of the costs of nodes in that path. The function that is used to measure the cost of using a node has timing and congestion terms as in (1), where Crit_{ij} is the timing criticality of the connection (i, j), $T(n)$ is the timing cost of the route to reach node n from the source, and $\text{Congs}(n)$ is the congestion cost of using node n

$$\text{Cost}(n) = \text{Crit}_{ij} \times T(n) + (1 - \text{Crit}_{ij}) \times \text{Congs}(n). \quad (1)$$

For the PG-aware router, we modified the congestion term of the cost function as

$$\text{Congs}(n) = \text{Congs}(n)_{\text{old}} \times (1 + \text{Cost}_{\text{Partition}}) \quad (2)$$

where $\text{Congs}(n)_{\text{old}}$ is the original congestion cost, and $\text{Cost}_{\text{partition}}$ is used to modify the cost of using a specific power-gated SB partition. The following function is used to calculate $\text{Cost}_{\text{partition}}$:

$$\text{Cost}_{\text{partition}} = \begin{cases} K, & \text{if } \delta_{\text{PGR,net}} = 0 \\ -L, & \text{if } \delta_{\text{PGR,net}} = 1 \end{cases} \quad (3)$$

where K and L are weighting parameters determined empirically, and $\delta_{\text{PGR,net}}$ is a binary function that has the value of 1 if the connection being routed belongs to the same module as that of the node's PGR, and 0 otherwise. Each wire segment (node) in FPGA architecture is driven by an SB switch; we consider a node to belong to a PGR if the switch driving it belongs to an SB in that PGR.

$\text{Cost}_{\text{Partition}}$ is used to change the weight given to the congestion cost of the node being investigated. If the module of the node's PGR is the same as that of the net being routed, then the congestion cost is decreased (by a factor of L) to encourage routing through the node. Routing nets that belong to the same module as that of the PGR through an SB partition in the PGR enables configuring the partition as DC. On the other hand, if the net does not belong to the same module as that of the node's PGR, then the cost is increased (by a factor of K) to discourage routing the net through that node; if the net is routed through that node, then the SB must be set as always-ON.

We found that $L = 0.2$ and $K = 3$ give good results with <1% increase in the critical path delay on average (up to 4% for a circuit). Larger L may result in circuits that cannot be routed because the router will not be able to resolve congestion. Larger K may result in a large congestion cost, which may negatively impact the critical path delay.

V. Benchmark Circuits

In this section, we describe the benchmark circuits used to evaluate the proposed architecture.

A. Synthetic Benchmarks Generation

We used the largest 20 Microelectronics Center of North Carolina benchmark circuits available with the VPR download [17] as subcircuits (or modules) in the generated synthetic circuits. Each of the generated circuits is composed of two or more modules (up to nine), connected to each other using the primary I/Os of the subcircuits. Table II shows the details of the generated circuits.

The modules in each circuit are connected together after performing the packing phase using T-VPack [25], i.e., after each circuit's LUTs and FFs are grouped in LCs.

TABLE II
GENERATED SYNTHETIC CIRCUITS

Circuit	Modules	# LCs
c2_1	elliptic, s38417	1498
c2_2	clma, s298	1625
c3_1	s298, elliptic, s38417	1833
c3_2	diffeq, frisc, s38584.1	1705
c4_1	ex5p, s298, apex2, seq	1104
c4_2	spla, diffeq, misex3, s38417	2106
c5_1	apex4, tseng, seq, pdc, clma	2709
c5_2	s38417, ex5p, diffeq, ex1010, s298	2536
c6_1	seq, tseng, apex4, pdc, spla, misex3	2275
c6_2	spla, ex5p, s298, seq, apex2, ex1010	2504
c7_1	misex3, spla, s38417, pdc, seq, tseng, apex4	3312
c8_1	clma, ex1010, s298, spla, misex3, spla, apex2, seq	4473
c9_1	s298, apex2, seq, alu4, elliptic, tseng, s38417, apex4, ex5p	3219

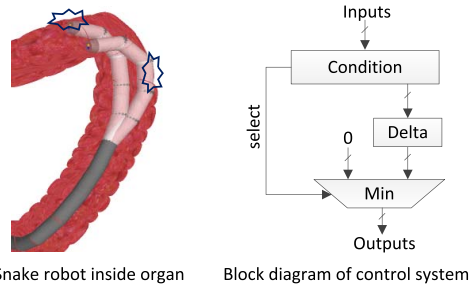

Fig. 12. Snake robot example application.

TABLE III
INFORMATION ABOUT FPGA-BASED iSnake ROBOT CONTROL SYSTEM

Module Name	# LCs	LCs %	Potential Power State
Condition	25342	75.16	always-on
Delta	8352	24.77	dynamically-controlled
Min	22	0.07	always-on

This guarantees that the subcircuits used in stitching closely represent independent functional modules in an application.

We assume that the power state for each module in the circuits of Table II can be DC. Thus, a power controller that has an output power control signal for each module is generated for each circuit. Timers are used to generate the power control signals in a power controller. A timer is sized assuming the sleep signal of a module is asserted after 50 ms. This amount has been chosen arbitrarily. Longer timer periods may increase the number of resources required to implement the controller circuit on an FPGA. Notice that more sophisticated power controllers can be implemented. However, the goal of this paper is to use the power controller circuits to evaluate the number of resources (especially routing resources) that are occupied by power control signals. This provides an estimate of the FPGA resources that will be in the different power states, and hence the potential power savings using the DCPG FPGA architecture.

B. Robot Control System

The application presented in this section is used to evaluate the proposed architecture. This application represents a control system for a snake-shaped robot, called iSnake, that is used in endoscopy [26]. The left side of Fig. 12 shows the robot inside an organ, in two different states.

The robot's control system provides haptic feedback to the surgeon to prevent harming the patient's organs during an operation. A proximity query (PQ) algorithm is used to approximate the distance between the iSnake and the surface of the patient's organ [26], which is computationally intensive and requires a high-performance implementation.

We developed an FPGA-based implementation of the control system. The right side of Fig. 12 shows the main modules in the system. The datapath performs stream processing for input data. The Delta module is only activated when the robot touches the organ's surface. This module can be put in sleep mode when its output is not required. The select output from the Condition module can be used as a power control signal for the Delta module.

The FPlibrary [27] was used to implement the required floating point operators in the PQ algorithm. Quartus II was used to generate a technology-mapped netlist of the circuit [28]. The netlist was then annotated with information about the modules of each circuit component. A modified version of T-VPack was then used to pack the circuit; this version ensures that each module's LUTs and FFs are clustered in the same LCs, thus generating a netlist that contains three interconnected modules.

Table III shows information about the robot control system. The size of the Delta module is ~25% of the system, indicating that properly managing its power state may result in large energy reduction. This is investigated in Section VI.

VI. EXPERIMENTAL RESULTS AND DISCUSSION

A. Experimental Setup

Unless otherwise indicated, the following FPGA architecture parameters are used: 1) LUT size $K = 4$; 2) LC size $N = 6$; 3) inputs per cluster $I = 16$; 4) RC width $W = 90$; 5) routing segment length $L = 4$; 6) switch box flexibility $F_s = 3$; 7) input pins CB flexibility $F_{c,\text{in}} = 0.2$; and 8) output pins CB flexibility $F_{c,\text{out}} = 0.1$.

We used HSPICE simulations to obtain the leakage power of the PGRs. We used the number of minimum-width transistors as in [16] to estimate the area. We used the 45-nm HP technology from the predictive technology models website [29], with supply voltage $V_{\text{DD}} = 1$ V and temperature $T = 85$ °C to measure the worst case power and timing.

For the power-gated architectures, the threshold voltage of sleep transistors has been increased by 100 mV by changing the V_{th0} parameter in the technology files. Sleep transistors have been iteratively sized to constrain the performance degradation to 10% compared with an architecture that does not support power gating. We assume 20% activity in doing this.

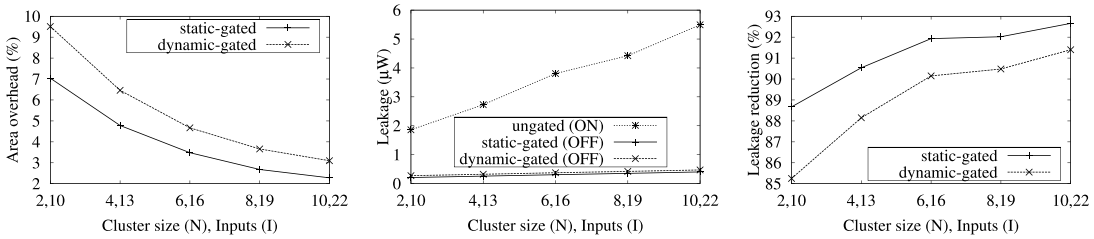

Fig. 13. Results for sweeping LC's cluster size (N). Switch blocks are not included in the results. (a) Area overhead. (b) Leakage power. (c) Leakage power reduction.

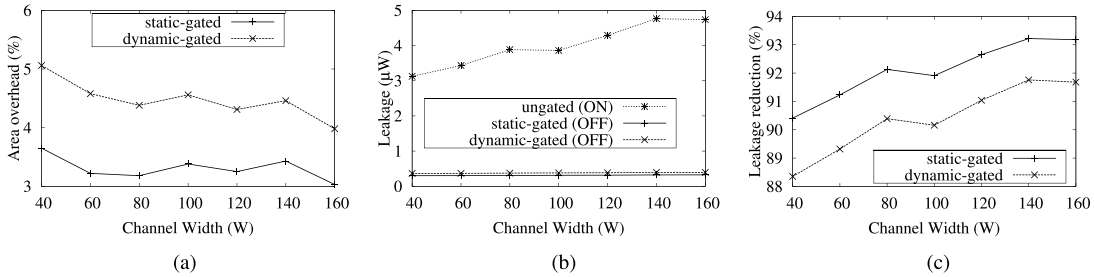

Fig. 14. Results for sweeping RC width (W). Switch blocks are not included in the results. (a) Area overhead. (b) Leakage power. (c) Leakage power reduction.

We assume that SRAM cells are built using six minimum-sized transistors. All multiplexers used in our architecture are based on pass transistors; each multiplexer is followed by a level restorer [30] and a buffer.

The SPICE netlists for LCs have been generated as follows. The size of the last inverter in a buffer is found by dividing the number of equivalent min-sized load inverters by four, and internal stages are sized by a stage ratio of four. Roughly, this sizing results in minimum delay [31]. LUTs are built using transmission gates as in [32], with inverters inserted after the second and last stages to reduce the delay of series-connected transmission gates.

The SPICE netlists for SBs have been generated as follows. Unless otherwise indicated, we assume a RC width (W) that is 20% larger than the minimum channel width required to route a circuit [16]. We used unidirectional, single driver routing architecture [33]. The output buffers of SBs are built using multiple stages of inverters. The stages are sized using a stage ratio of four. The capacitance of wire segments was obtained using the model in [34]. The outputs of the LCs connect directly to the SBs through isolation buffers without the need for output pins connection blocks; this is similar to the architecture assumptions made in VPR 5.0 [17].

B. Architecture Parameters Sweep

In this section, we study the area and power of the proposed architecture for different architecture parameters. Note that this is done without mapping applications to the architecture. The following list defines three architectures that are evaluated in our experiments.

1) *Ungated:* This is the baseline FPGA architecture that does not support power gating.
2) *Static Gating:* This is an architecture that supports Statically controlled power gating, such as the one presented in [4]. The power state for this architecture can only be set at configuration time.
3) *Dynamic Gating:* This is the DCPG architecture that we proposed in Section III.

1) Power-Gated Tiles: We first study a basic architecture that has only one tile (without the SB); we vary two parameters, the cluster size (N) and the width of the RCs (W). When varying N, we also vary the number of input pins (I) of the LC. When varying N, we set $W = 90$, and when varying W we set $N = 6$ and $I = 16$.

Figs. 13 and 14 show the effect of the cluster size (N) and RC width (W) on power gating. The results shown in the figures are for a tile that supports power gating, not including the SB, compared with a tile that does not support power gating (ungated).

The area overhead decreases as the cluster size and the channel width increase [Figs. 13(a) and 14(a)]; this is because a larger number of circuit components are powered through a single sleep transistor. However, there is no high correlation between the area overhead and the channel width. The area overhead for the static-gated architecture is lower than that for the proposed dynamic-gated architecture. This is because of the additional circuit components that are required to support dynamic power state control.

The leakage power reduction increases as the cluster size and the channel width increase [Figs. 13(c) and 14(c)]. This is similar to the area overhead trend. Smaller area overhead results in lower leakage power overhead due to the power gating circuitry. The results show that the leakage power reduction in the OFF-state (compared with an ungated architecture)

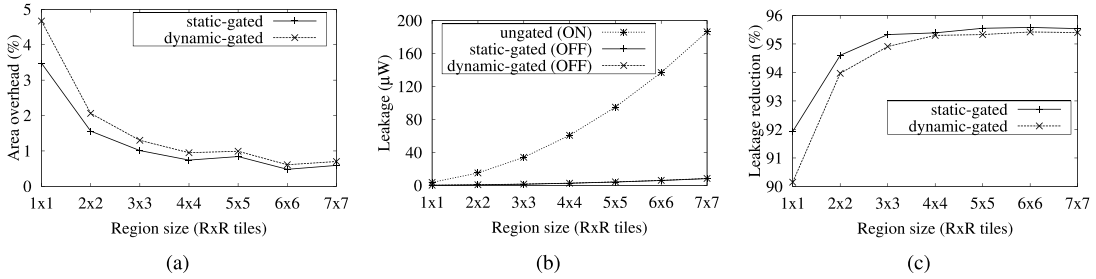

Fig. 15. Results for sweeping granularity of PGRs. Switch blocks are not included in the results. (a) Area overhead. (b) Leakage power. (c) Leakage power reduction.

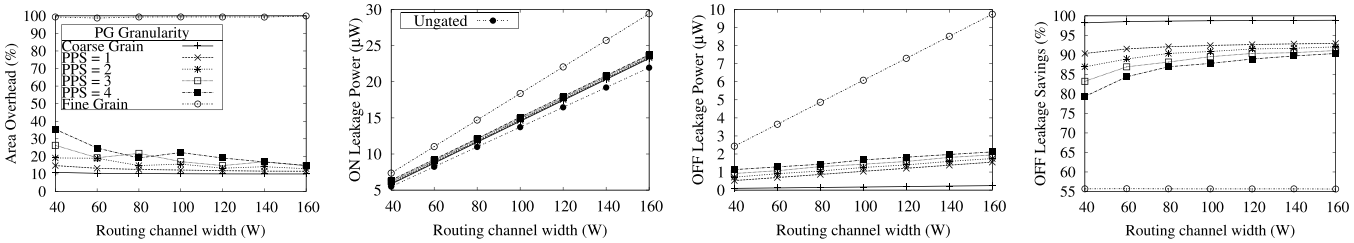

Fig. 16. Results for SBs power gating granularity by sweeping RC width (W). Segment length (L) = 2. (a) Area overhead. (b) ON-leakage power. (c) OFF-leakage power. (d) Leakage power reduction.

can be up to ∼91% for a cluster size of 10 (channel width of 160). Figs. 13(b) and 14(b) show that the proposed architecture has slightly larger leakage power in the OFF-state than the static-gated architecture. This is because the dynamic-gated architecture requires more circuit components to support controlling its power state dynamically.

2) Power-Gated Regions: The results of sweeping the granularity of the proposed architecture are shown in Fig. 15. As the region size increases, the area overhead decreases. The area overhead includes that of the sleep transistors and the circuit components required to support configuring the different power states of a PGR. The area overhead decreases as the region size increases because more circuit components are powered through a single sleep transistor, and the circuit components required to support the different power states of a region are shared among larger number of circuit components. The area overhead is as small as 1% for a PGR of 4×4 tiles.

The leakage power reduction increases as the region size increases [Fig. 15(c)]. This is because larger regions have smaller area overhead, which results smaller leakage power overhead due to the power gating circuitry. The OFF-state leakage power of the power-gated architectures is much lower than that for the ungated architecture, leading to a leakage power reduction of >90% (∼95% for a PGR of 4×4 tiles). Increasing the region size by more than 4×4 tiles does not significantly increase the leakage power savings. As can be observed in Fig. 15(c), the leakage power reduction in a static-gated architecture is slightly larger than that in the proposed dynamic-gated architecture. This is because of the additional circuit components that are required in the proposed architecture to support changing its power state at runtime.

3) Power-Gated Switch Blocks: In this section, we vary the architecture parameters that describe the SBs. Results are shown for SB architectures that have different segment lengths, $L = 2$ in Fig. 16 and $L = 4$ in Fig. 17, compared with an architecture that does not support power gating. In addition to varying the RC width (W), we also vary the power gating granularity of SBs by varying the number of PPS (larger PPS means finer granularity). In a fine-grained SB power gating, each output buffer in an SB has a power gating circuit, whereas in a coarse-grained power gating a single power gating circuit is used for all circuit components in an SB.

Figs. 16 and 17 show that the area overhead and leakage power when $L = 4$ is lower than that for $L = 2$. This is because an FPGA routing architecture that has shorter segments contains more circuit components in SBs [33].

The area overhead [Figs. 16(a) and 17(a)] of the power gating circuitry decreases as the channel width increases. This is because as we increase W the sleep transistor size increases at a lower rate than the increase in the number of circuit components that are powered through it.

The power gating granularity (PPS) also affects the area overhead. Fine-grained power gating results in large area overhead (>60% for $L = 4$ and ∼100% for $L = 2$). For large values of W, the area overhead for granularities down to PPS = 4 ranges between 10% and 16%. This area overhead, however, is only for SBs. The overall area overhead for the power gating architecture is lower. Recall from Section VI-A that the area overhead (without SBs) for a PGR of size 4×4 tiles is ∼1%. The overall area overhead for the same PGR with SBs ranges between 4.7% and 10.3% for PPS from 0 to 4 and $W = 90$.

Figs. 16(b) and 17(b) show the ON-leakage power for the gated architecture (for different PPS) and the ungated architecture. The ON-leakage increases as PPS increases; this is because finer granularity power gating requires more circuit components and larger sleep transistors. The ON-leakage overhead for $W = 100$ and PPS between 0 and 4, for example, is ∼6%–10% for $L = 2$ and 3.5%–7.7% for $L = 4$. As we

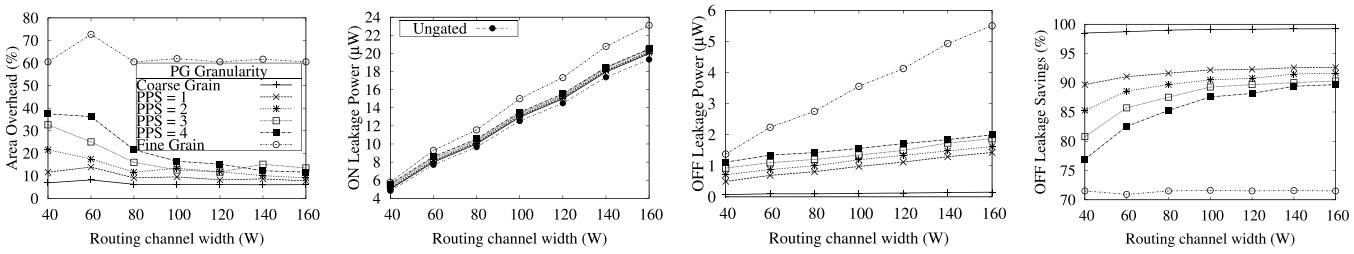

Fig. 17. Results for SBs power gating granularity by sweeping RC width (W). Segment length (L) = 4. (a) Area overhead. (b) ON-leakage power. (c) OFF-leakage power. (d) Leakage power reduction.

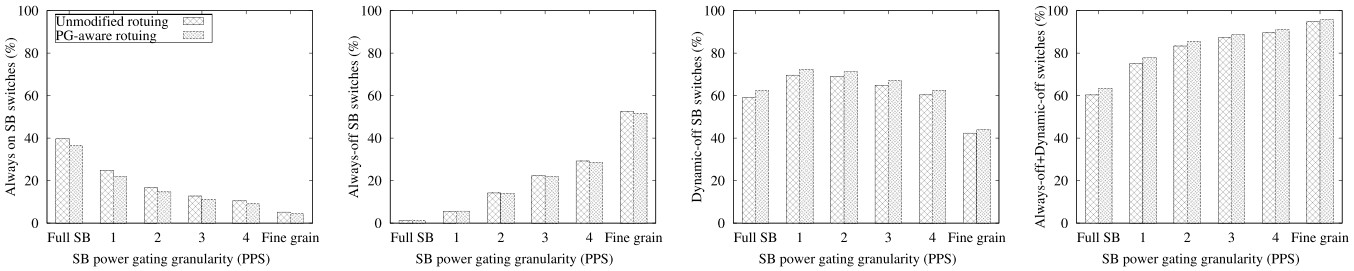

Fig. 18. SBs switches power state averaged over all synthetic benchmarks using the unmodified and power gating-aware routing. (a) Always-ON. (b) Always-OFF. (c) DC. (d) Always-OFF + DC.

will see later, finer granularity SBs power gating increases the number of always-OFF resources, resulting in lower total ON-leakage power for an application circuit.

Figs. 16(c) and 17(c) show the leakage power for the power-gated SBs in the OFF-state, i.e., static power when SBs are powered down, and Figs. 16(d) and 17(d) show the leakage power reduction compared with SBs with no power gating. The OFF-leakage power for SBs with PPS = 0 is the smallest because all of the SB circuit components are included in the power-gated circuit. SBs with larger PPS, incur larger leakage power overhead in the OFF-state because of the additional power gating circuit components, and because all the buffers for incoming wire tracks are designed to be powered during the OFF-state (Section III-D). The leakage power reduction for the proposed architecture is >95% for the coarse-grained power-gated SBs, and could reach >90% for PPS between 1 and 4. For fine-grained power gating, the leakage power reduction is ∼70%.

4) Total Area Overhead: Table IV shows the total area overhead for different architecture granularities of the power gating architecture. The area overhead ranges between 3.9% and 34.8%. The area overhead results in longer routing wire segments. This leads to larger interconnect capacitance, and potentially larger dynamic power. For example, using PPS = 3 would result in area overhead between 9.2% and 10.7%. This translates to 4.5%–5.2% increase in each dimension of a tile, assuming square tiles [35]. This represents a loose upper bound on the increase in interconnect capacitance [35]. In this paper, we do not evaluate the effect of the area overhead on the dynamic power and how this could affect the savings achieved by the proposed architecture; we leave this for future work.

TABLE IV
TOTAL AREA OVERHEAD FOR THE POWER GATING ARCHITECTURE FOR DIFFERENT ARCHITECTURE GRANULARITIES ($W = 80$ AND $L = 4$)

SB/PGR	1x1	2x2	3x3	4x4
Coarse Grained	5.4	4.3	4.1	3.9
$PPS = 1$	6.9	5.8	5.7	5.5
$PPS = 2$	8.3	7.2	7.1	6.9
$PPS = 3$	10.7	9.6	9.4	9.2
$PPS = 4$	13.7	12.6	12.4	12.2
Fine Grained	34.8	33.7	33.5	33.3

C. Benchmark Circuits Results

In this section, we use the CAD flow in Section IV to place and route the synthetic benchmark circuits presented in Section V. This is done to study the granularity of SBs power gating in the proposed architecture. For each circuit, a power controller has been generated as described in Section V-A to provide a power control signal for each module in the circuit. Each circuit has been placed and routed on an architecture with a PGR's size of 4 × 4 tiles, segment length of 4, and RC width that is 20% larger than the minimum channel width required to route the circuit. Multiple architectures with different SB power gating granularities (different PPS) have been used. The results are shown for the original VPR's routing algorithm, and the enhanced power gating-aware (PG-aware) algorithm that is described in Section IV-B.

1) Breakdown of SBs' Switches Power States: In this section, we report the percentages of SB switches that can be configured in the different power states.

Fig. 18(a) shows the percentage of always-ON switches for different SB power gating granularities. For finer granularity power gating, the number of always-ON SB

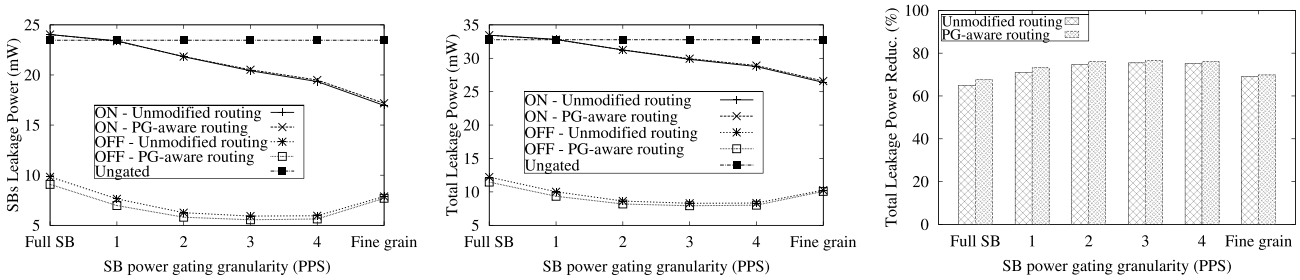

Fig. 19. Leakage power results for (a) SBs only and (b) and (c) SBs and PGRs averaged over all synthetic benchmark circuits.

switches decreases. This is expected since more SB switches can statically be turned OFF because the SB partitions they belong to are not used to route signals. Furthermore, with finer granularity, there is a better chance that an SB partition is only used to route signals that belong to a single module, which increases the number of DC SB switches.

Using the PG-aware routing algorithm results in slightly fewer always-ON switches, but this improvement diminishes as PPS increases; finer granularity power gating (larger PPS) results in larger number of components that can be statically turned OFF. Therefore, the improvement space available for the PG-aware router becomes tighter. The results show that the PG-aware router reduces the number of always-ON switches by 3% for coarse-grained SB power gating (PPS $= 0$), and $\sim$1% for PPS $= 4$ of the total number of switches. This corresponds to reduction of 8% and 14% of the always-ON switches for PPS $= 0$ and PPS $= 4$, respectively.

Fig. 18(b) shows the percentage of always-OFF switches for different SB power gating granularities. As expected, finer granularity power gating increases the always-OFF switches. The PG-aware routing has a negligible effect on the number of always-OFF switches.

Fig. 18(c) shows the percentage of DC switches for different SB power gating granularities. These are switches that can be powered OFF at run time when the module they belong to becomes idle. The largest number of DC switches is when PPS $= 1$. Finer SB power gating granularity reduces the number of DC SB switches; this is because more switches can be set as always-OFF at configuration time, which reduces the total number of remaining switches. The PG-aware routing helps in slightly improving the percentage of DC switches. This improvement is roughly the same as the amount of reduction in the always-ON switches shown in Fig. 18(a).

Finally, Fig. 18(d) shows the sum of the always-OFF and DC switches. As expected, SBs with finer granularity power gating result in larger number of switches that can be powered down. The PG-aware routing shows slight improvements compared with the original VPR router; these improvements diminish as the granularity of power gating decreases as explained above.

2) Leakage Power: In this section, we report the leakage power averaged over all benchmark circuits. We report the ON-leakage power, which is the leakage power assuming all modules in a circuit are idle but not powered OFF, the OFF-leakage power, which is the leakage power for the circuits assuming that all modules in a circuit are idle and their DC components are turned OFF. In both cases, the always-OFF components are assumed to be turned OFF at configuration time.

Fig. 19(a) shows the leakage power for the SBs for different SB power gating granularities, and the leakage power for the SBs when an ungated architecture is used. As can be seen, the ON-leakage power for both the PG-aware routing and the original routing are roughly equal because the algorithm enhancements do not significantly improve the number of always-OFF switches as explained earlier. For finer granularity SBs power gating, both the ON and OFF-leakage powers decrease. The ON-leakage power decreases since finer granularity results in more unused SB partitions that can be turned OFF at configuration time, thus the overall leakage power in the ON-state for an application circuit goes down. The OFF-leakage power also decreases with finer granularity power gating because more SB switches can be placed in the DC state. The minimum OFF-leakage power is when PPS $= 3$. Larger PPS values result in more leakage power consumption in the OFF state because of the large overhead of the power gating circuitry.

Fig. 19(a) also shows that the PG-aware routing slightly improves the OFF-leakage power compared with the unmodified routing algorithm. This is because PG-aware routing results in more DC switches that can be turned OFF when an application circuit is idle, as shown in Fig. 18(c).

Fig. 19(a) shows that ON-leakage power with the gated architecture is larger than that with the ungated architecture for the coarse-grained SBs power gating. However, for finer granularity power gating, the ON-leakage power for the gated architecture is lower than that for the ungated architecture. Although a single SB with finer granularity power gating has larger ON-leakage power than an ungated SB, finer granularity power gating enables turning OFF unused SB switches, which results in lower overall ON-leakage power.

The total leakage power is shown in Fig. 19(b). This includes the leakage power for both SBs and PGRs. The same trends discussed above apply here because a significant portion of the leakage power comes from SBs. Comparing Fig. 19(a) and (b) shows that SBs contribute to roughly 72% of leakage power in the ungated architecture. However, in the gated architecture, SBs contribute to roughly 67%–72% of the total leakage power.

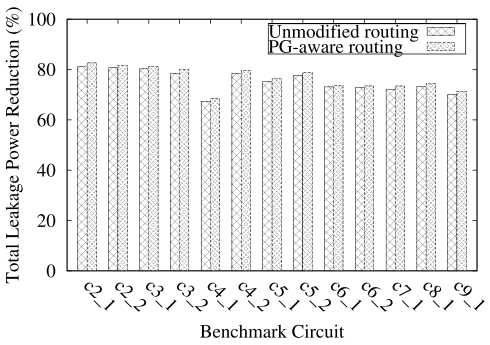

Fig. 20. Total leakage power reduction for individual synthetic circuits.

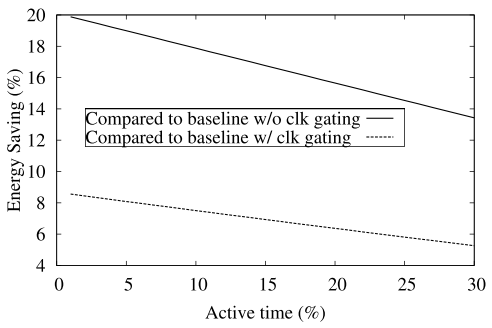

Fig. 21. Energy reduction for iSnake's control system in DCPG FPGA.

Fig. 19(c) shows the upper limit of reduction in the OFF-leakage power compared with the ungated architecture. As expected, the leakage power reduction increases with finer granularity SBs power gating, and the largest reduction is achieved when PPS = 3. For coarse-grained SB power gating, the OFF-leakage power reduction is ∼68%. For PPS = 3, the reduction is ∼77%.

Fig. 20 shows the individual circuits' potential power reduction when all modules are turned OFF (PPS = 3). The power reduction ranges between 67% and 81% using the original routing algorithm, and ranges between 68% and 83% using the PG-aware routing.

D. Example Application Results

In this section, we show the energy saving results of using the proposed power gating architecture for the iSnake robot control system described in Section V-B.

Fig. 21 shows the energy savings for the circuit for different active times of the Delta module. We compared mapping the application on the proposed power gating architecture with two baseline implementations. The first is a system that has no power optimizations at all. Normally, one would implement clock gating for modules that experience inactivity periods. We assume that the first baseline does not include this. The second baseline is an implementation that supports clock gating for the Delta module.

At 5% activity level, the results show that compared with the first baseline (no clock gating), the proposed architecture coupled with clock gating could achieve ∼19% energy savings. Compared with the second baseline (includes clock gating), the proposed architecture could achieve ∼8% additional energy saving. Given that only a small portion of the application benefited from the power gating architecture (25% of the circuit), the results are promising.

VII. Conclusion

We present an FPGA architecture that supports dynamic power gating. This architecture enables powering down modules in an FPGA when they are idle to reduce their static power dissipation. The architecture's flexibility enables the user to implement an arbitrary number and structure of power-gated modules, and enables routing power control signals on the general-purpose routing fabric of an FPGA. We also present a CAD flow that can be used to map applications to the proposed architecture, and enhancements to a routing algorithm in order to optimize the power savings of the architecture.

The area overhead of the architecture is ∼10.3% when the power gating region size is 4×4 tiles and the number of power-gated SB PPS is four ($W = 90$). The potential leakage power savings for the studied benchmark circuits are up to 83%. We also studied the energy savings in a control system for a robot that is used in medical applications. Assuming only 25% of the system can be powered down when idle, and it is idle for 95% of the time, we found that ∼8% energy saving can be achieved by the proposed architecture when compared with an implementation with only clock gating.

This research provides the basis for a new generation of FPGAs, which are capable of self-optimization. Future work includes automating the process of identifying application modules that can benefit from the proposed architecture. This is suitable for designs that use accelerators with components that operate for only a small fraction of time. Furthermore, enhancements to the CAD tools are required in order to better guide the different stages in the flow to increase idle times and increase the number of resources that can be powered down, while reducing the impact on performance and area.

References

[1] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," in *Proc. 14th ACM/SIGDA Int. Symp. Field-Program. 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 *Proc. Conf. Design, Autom. Test Eur.*, 2000, pp. 624–633.

[3] Q. Wang, S. Gupta, and J. H. Anderson, "Clock power reduction for virtex-5 FPGAs," in *Proc. 17th ACM/SIGDA Int. Symp. Field-Program. 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 *Proc. 14th Int. Symp. Field-Program. 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 *Proc. 12th ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, 2004, pp. 42–50.

[6] J. Hussein, M. Klein, and M. Hart, "Lowering power at 28 nm with Xilinx 7 series FPGAs," Xilinx, Inc., San Jose, CA, USA, Tech. Rep. WP389, Jun. 2011.

[7] *Meeting the Low Power Imperative at 28 nm*, Altera Corp., San Jose, CA, USA, Nov. 2011.

[8] S. Henzler, *Power Management of Digital Circuits in Deep Sub-Micron CMOS Technologies* (Advanced Microelectronics). Secaucus, NJ, USA: Springer-Verlag, 2007.

[9] Y. Lin, F. Li, and L. He, "Routing track duplication with fine-grained power-gating for FPGA interconnect power reduction," in *Proc. Asia South Pacific Design Autom. Conf.*, Jan. 2005, pp. 645–650.

[10] A. Gayasen, Y. Tsai, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, and T. Tuan, "Reducing leakage energy in FPGAs using region-constrained placement," in *Proc. 12th Int. Symp. Field-Program. Gate Arrays*, 2004, pp. 51–58.
[11] R. P. Bharadwaj, R. Konar, P. T. Balsara, and D. Bhatia, "Exploiting temporal idleness to reduce leakage power in programmable architectures," in *Proc. Asia South Pacific Design Autom. Conf.*, 2005, pp. 651–656.
[12] A. A. M. Bsoul and S. J. E. Wilton, "An FPGA architecture supporting dynamically controlled power gating," in *Proc. IEEE Int. Conf. Field-Program. Technol. (FPT)*, Dec. 2010, pp. 1–8.
[13] A. A. M. Bsoul and S. J. E. Wilton, "An FPGA with power-gated switch blocks," in *Proc. IEEE Int. Conf. Field-Program. Technol. (FPT)*, Dec. 2012, pp. 87–94.
[14] C. Li, Y. Dong, and T. Watanabe, "New power-aware placement for region-based FPGA architecture combined with dynamic power gating by PCHM," in *Proc. 17th IEEE/ACM Int. Symp. Low-Power Electron. Design (ISLPED)*, Aug. 2011, pp. 223–228.
[15] C. H. Hoo, Y. Ha, and A. Kumar, "A directional coarse-grained power gated FPGA switch box and power gating aware routing algorithm," in *Proc. 23rd Int. Conf. Field Program. Logic Appl. (FPL)*, Sep. 2013, pp. 1–4.
[16] V. Betz, J. Rose, and A. Marquardt, Eds., *Architecture and CAD for Deep-Submicron FPGAs*. Norwell, MA, USA: Kluwer, 1999.
[17] J. Luu *et al.*, "VPR 5.0: FPGA CAD and architecture exploration tools with single-driver routing, heterogeneity and process scaling," in *Proc. 17th ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, 2009, pp. 133–142.
[18] M. Klein, "Power consumption at 40 and 45 nm," Xilinx, Inc., San Jose, CA, USA, Tech. Rep. WP298, Apr. 2009.
[19] A. Marquardt, V. Betz, and J. Rose, "Timing-driven placement for FPGAs," in *Proc. 8th ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, 2000, pp. 203–213.
[20] M. Keating, D. Flynn, R. Aitken, A. Gibbons, and K. Shi, *Low Power Methodology Manual: For System-on-Chip Design*. New York, NY, USA: Springer-Verlag, 2007.
[21] A. A. M. Bsoul and S. J. E. Wilton, "A configurable architecture to limit wakeup current in dynamically-controlled power-gated FPGAs," in *Proc. Int. Symp. Field-Program. Gate Arrays (FPGA)*, 2012, pp. 245–254.
[22] A. A. M. Bsoul and S. J. E. Wilton, "A configurable architecture to limit inrush current in power-gated reconfigurable devices," *J. Low Power Electron.*, vol. 10, no. 1, pp. 1–15, 2014.
[23] P. Jamieson, K. B. Kent, F. Gharibian, and L. Shannon, "Odin II—An open-source Verilog HDL synthesis tool for CAD research," in *Proc. 18th IEEE FCCM*, May 2010, pp. 149–156.
[24] *ABC: A System for Sequential Synthesis and Verification*, Univ. California, Berkeley, CA, USA, 2012.
[25] A. R. Marquardt, "Cluster-based architecture, timing-driven packing and timing-driven placement for FPGAs," M.S. thesis, Dept. Elect. Comput. Eng., Univ. Toronto, Toronto, ON, Canada, 1999.
[26] K.-W. Kwok *et al.*, "Dimensionality reduction in controlling articulated snake robot for endoscopy under dynamic active constraints," *IEEE Trans. Robot.*, vol. 29, no. 1, pp. 15–31, Feb. 2013.
[27] J. Detrey and F. de Dinechin. (2004). *FPLibrary, a VHDL Library of Parametrisable Floating-Point and LNS Operators for FPGA*. [Online]. Available: http://www.ens-lyon.fr/LIP/Arenaire/Ware/FPLibrary/
[28] *Quartus II University Interface Program*. [Online]. Available: http://www.altera.com/education/univ/research/quip/unv-quip.html, accessed Jan. 28, 2015.
[29] *Predictive Technology Model (PTM)*. [Online]. Available: http://ptm.asu.edu/, accessed Jan. 28, 2015.
[30] E. Hung, S. J. E. Wilton, H. Yu, T. C. P. Chau, and P. H. W. Leong, "A detailed delay path model for FPGAs," in *Proc. Int. Conf. Field-Program. Technol. (FPT)*, 2009, pp. 96–103.
[31] N. H. E. Weste and D. M. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th ed. Reading, MA, USA: Addison-Wesley, 2011.
[32] T. Pi and P. J. Crotty, "FPGA lookup table with transmission gate structure for reliable low-voltage operation," U.S. Patent 6 667 635, Dec. 23, 2003.
[33] G. Lemieux, E. Lee, M. Tom, and A. Yu, "Directional and single-driver wires in FPGA interconnect," in *Proc. IEEE Int. Conf. Field-Program. Technol.*, Dec. 2004, pp. 41–48.
[34] S.-C. Wong, G.-Y. Lee, and D.-J. Ma, "Modeling of interconnect capacitance, delay, and crosstalk in VLSI," *IEEE Trans. Semicond. Manuf.*, vol. 13, no. 1, pp. 108–111, Feb. 2000.
[35] S. Huda, J. Anderson, and H. Tamura, "Charge recycling for power reduction in FPGA interconnect," in *Proc. 23rd Int. Conf. Field Program. Logic Appl. (FPL)*, Sep. 2013, pp. 1–8.

Assem A. M. Bsoul (S'07) received the B.Sc. degree in computer engineering from the Jordan University of Science and Technology, Irbid, Jordan, in 2006, the M.Sc. degree in electrical engineering from Queen's University, Kingston, ON, Canada, in 2009, and the Ph.D. degree in electrical and computer engineering from the University of British Columbia, Vancouver, BC, Canada, in 2014.

He is currently a Post-Doctoral Fellow with the University of British Columbia. His current research interests include low-power reconfigurable architectures and computer-aided design algorithms.

Steven J. E. Wilton (S'86–M'97–SM'03) received the M.A.Sc. and Ph.D. degrees in electrical and computer engineering from the University of Toronto, Toronto, ON, Canada, in 1992 and 1997, respectively.

He was a co-founder of Veridae Systems, Inc., Vancouver, BC, Canada, acquired by Tektronix, Beaverton, OR, USA, in 2011, which developed debug solutions for application-specific integrated circuits, field-programmable gate arrays (FPGAs), and FPGA-based systems. He joined the Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, in 1997, where he is currently a Professor and an Associate Head. His current research interests include the architectures of next-generation FPGAs and their associated computer-aided design tools.

Dr. Wilton served as the Program and General Chair of the ACM International Symposium on FPGAs, from 2005 to 2006, respectively, and the Program Co-Chair of the 2005 International Conference on Field Programmable Logic and Applications and the 2008 IEEE International Conference on Application-Specific Systems, Architectures and Processors. He was a recipient of best paper awards at the International Conference on Field-Programmable Technology in 2003, 2005, 2007, and 2013, respectively, and the International Conference on Field-Programmable Logic and Applications in 2001, 2004, 2007, and 2008, respectively. He is currently the Editor-in-Chief of the *ACM Transactions on Reconfigurable Technology and Systems*.

Kuen Hung Tsoi received the Ph.D. degree from the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong, in 2007.

He has been a Post-Doctoral Research Associate with the Custom Computing Group, Department of Computing, Imperial College London, London, U.K., since 2008. He is currently with Imagination Technologies Ltd., Kings Langley, U.K.

Wayne Luk (F'09) received the M.A., M.Sc., and D.Phil. degrees in engineering and computing science from the University of Oxford, Oxford, U.K.

He was a Visiting Professor with Stanford University, Stanford, CA, USA. He is currently a Professor of Computer Engineering with Imperial College London, London, U.K. His current research interests include reconfigurable computing, field programmable technology, and design automation.

Prof. Luk is a fellow of the Royal Academy of Engineering. He received the Research Excellence Award from Imperial College London for reconfigurable supercomputing, and over 15 awards for his work from international conferences, such as the Applied Reconfigurable Computing Conference, the Application-Specific Systems, Architectures and Processors Conference, the Field-Programmable Custom Computing Machines Conference, the Field Programmable Logic and Applications Conference, and the Field-Programmable Technology Conference. He was the Founding Editor-in-Chief of the *ACM Transactions on Reconfigurable Technology and Systems*.