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 FPGAs built using 90 nm and smaller technology nodes. Power gating was shown to be effective at reducing leakage power. Previous techniques focus on turning off unused FPGA resources at configuration time; the benefit of this approach depends on resources 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 trade-offs 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.**

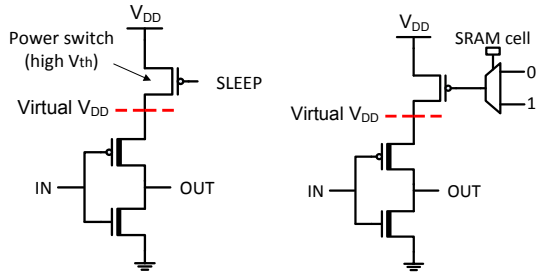
I. Introduction

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 to application-specific integrated circuit (ASIC) implementations, FPGA implementations consume 12 times 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 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.

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 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 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 hand-held 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 Figure 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 trade-off between performance, power savings, and area can be found.



(a) Basic power gating (b) Configurable power gating

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)) [4], [9]–[11]. We refer to them 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 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*

Manuscript received Month day, year.

Assem A. M. Bsoul and Steven J. E. Wilton are with the Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, BC V6T 1Z4, Canada (e-mail: {absoul, stevew}@ece.ubc.ca). Kuen Hung Tsoi and Wayne Luk are with the Department of Computing, Imperial College London, London SW7 2AZ, UK.

Bsoul and Wilton are supported in part by Altera Corp., and Luke and Tsoi are supported in part by the UK EPSRC and by the European Union Seventh Framework Programme under grant agreement 257906, 287804, and 318521.

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.

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

- 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.
- We present a CAD flow that can be used to map 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.
- We propose enhancements to an FPGA routing algorithm that try to minimize the the number of routing resources that cannot be powered down at run-time.
- The presented CAD flow is used to evaluate the best granularity of routing resources power gating.
- 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 the paper. Section III describes the proposed dynamically-controlled power gating (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. studied fine-grained power gating for FPGAs to turn off unused resources at configuration time [9]; their study showed that the area overhead could be more than 100%, which is undesirable because of the associated degradation in power and timing, and the increase in cost.

Gayasen et al. proposed coarse-grained power gating by using a power switch for a region of logic blocks [10]. 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.

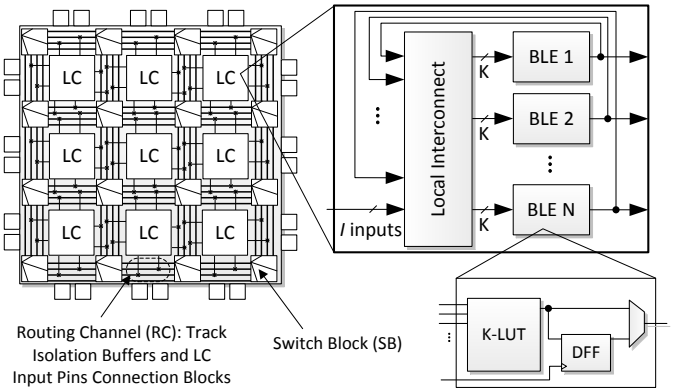


Fig. 2. Tile-based FPGA architecture

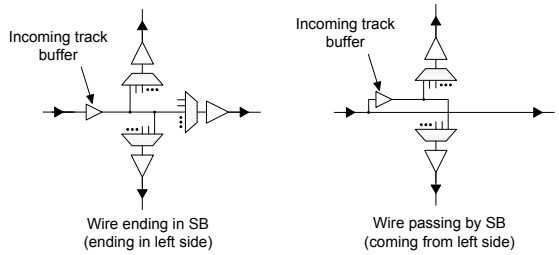


Fig. 3. Example incoming wire tracks and switches in an SB

Tuan et al. proposed power gating for an architecture similar to the Xilinx Spartan-3 [4]. Their architecture supports sleep mode by 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. proposed synthesizing a power state controller (PSC) from the data flow graph (DFG) of an application; this controller could exploit the idleness periods of the application to reduce the dissipated leakage energy in an FPGA [11]. They used the same architecture in [10].

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

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

B. Architectrure 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 and a switch block) as shown in Figure 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, 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 intra-cluster connections. Figure 2 shows an 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 block (SB) that can be configured to route the signals to the different directions. Figure 3 shows examples of the connections for switches in an SB. Connections can be made from a routing channel that borders an LC from one of its sides to the input 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. Finally, the outputs of an LC are connected directly to multiplexers in the switch blocks through isolation buffers. This is similar to the architectural assumptions made in the VPR 5.0 tool [17].

III. Proposed DCPG FPGA Architecture

Figure 4 shows an example system of three modules that are mapped to an FPGA that supports dynamically-controlled power gating (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. 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.

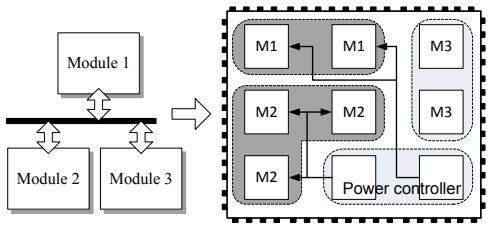


Fig. 4. Example application mapped to an FPGA supporting DCPG

In this example, two of the functional modules, M1 and M2, experience long idle periods, thus it is desired to power them down during these times to reduce the leakage power consumption. The power state of the PGRs of M1 and M2 is configured to "dynamically-controlled", 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 pre-existing FPGA routing fabric similar to any other user signal. The following subsections describe the details of this architecture.

A. Basic Power Gating Architecture

In this subsection, we describe a fine-grained version of the proposed power gating architecture. Figure 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 connection boxes, track isolation buffers, and switch blocks) dynamically at run-time. This makes the proposed power gating scheme suitable for various tile-based FPGA architectures.

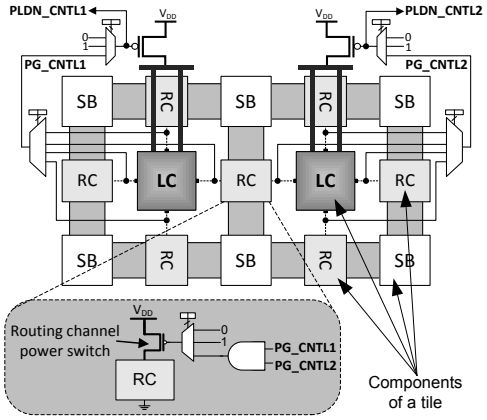


Fig. 5. Basic fine-grained DCPG for two neighboring tiles. Shared bordering routing channels have their own power gating circuit

The supported power modes are *always-on*, *always-off*, and *dynamically-controlled*. 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, low-leakage mode. This is useful for resources that are not utilized by the application that is mapped to the device. *Dynamically-controlled* 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 Figure 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

dynamically-controlled, 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 dynamically-controlled mode, the power state of the routing channel 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 routing channels of an LC. The lower part of Figure 5 shows the details. When configured to the dynamically-controlled mode, the AND gate ensures that the shared routing channel 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 routing channels 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 logic clusters 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, configuration memory cells and the flip-flops (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 flip-flops 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.

Figure 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 dynamically-controlled power gating mode during the idle times; this guarantees that the values stored in the FFs do not get corrupted during sleep mode. The pull-down 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. Figure 6 shows that an inverted version of PLDN_CNTL, which is the output of the related 3:1 multiplexer in Figure 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 chose the feedback signal ($\overline{\text{PLDN\_CNTL}}$) or

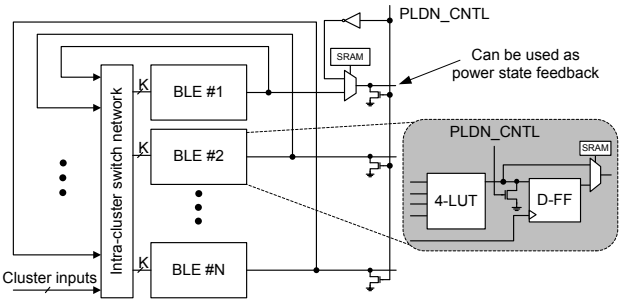


Fig. 6. Internals of an LC with DCPG showing pull-down NMOS devices at FF input and LC outputs. A status signal to indicate completion of power transition can be routed through one of the LC's outputs (the top)

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 which 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 the previous subsection are due to the sleep transistors of the LC and the routing channels, 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 proper power gating for the routing channels, 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 routing channels 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 the previous subsection 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 Subsection 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 the previous subsection, the SRAM configuration memory cells and flip-flops are powered on all the time.

Figure 7 shows an example dynamically-controlled power gating region (PGR) of size 2x2 tiles. Some details are omitted for clarity. The region's LCs and internal routing channels (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 dynamically-controlled.

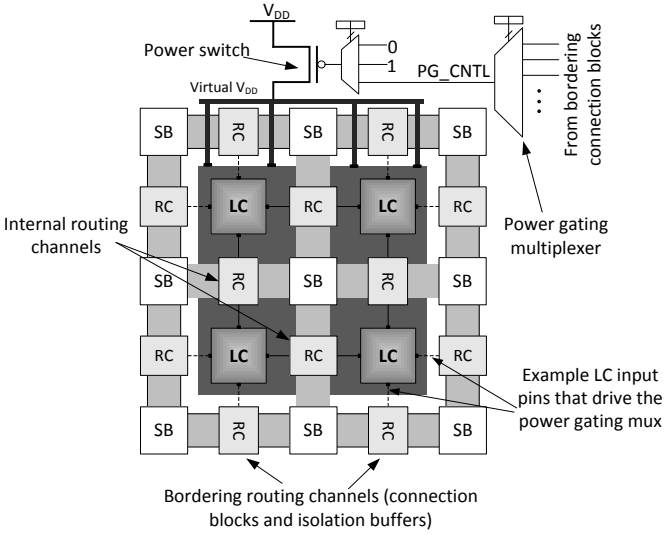


Fig. 7. Example PGR of 2x2 tiles. Internal region's LCs and CBs share the power switch. PGR's SBs and bordering RCs have individual power switches

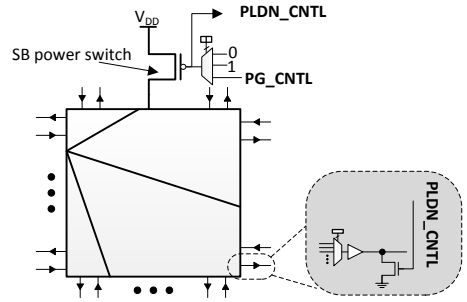


Fig. 8. Power gating circuit for a switch block. SB outputs are pulled down to GND when the SB's power is off (in sleep mode)

TABLE I
CONFIGURABLE POWER MODES SUPPORTED BY THE DIFFERENT COMPONENTS IN A POWER GATING REGION (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.

The bordering RCs in the coarse-grained PGRs have the same structure and functionality described for the RCs in Figure 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 3x3 PGR consists of 3x3 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 subsections, we described the proposed power gating architecture for LCs and RCs (track isolation buffers and connection boxes). This subsection focuses on describing the power gating circuitry for SBs in a PGR.

The example PGR in Figure 7 has a size of 2x2 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.

Figure 8 shows the power gating circuitry for an SB. This circuitry is similar to that for the other components as described in the previous subsections. Minimum-sized pull-down NMOS transistors are placed at the outputs of the SB to pull them to ground during 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 dynamically-controlled, 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 inter-module signals.

D. Fine-Grained Power-Gated Switch Blocks

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

Figure 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 Figure 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), whilst $PPS = N_{switch}$ indicates the finest power gating granularity where the power state for each switch can 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 (see Figure 3) must be always-on, i.e., not power gated.

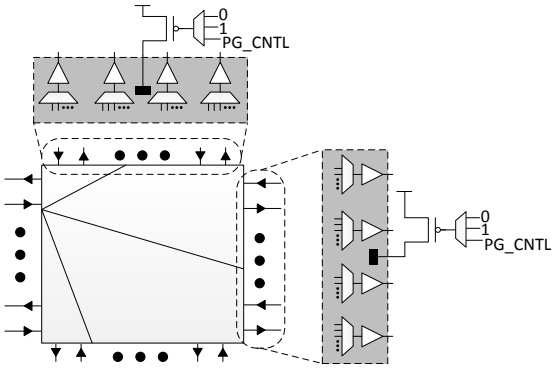


Fig. 9. One power gating partition per SB side (details for two sides are shown). The power state for each partition can be configured separately

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].

Refs. [21], [22] presented 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], [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 about 10 clock cycles on a 300 MHz clock frequency, assuming 25 PGRs can be turned on simultaneously and each PGR has a size of 4x4 tiles [22].

The inrush current handling architecture in [21], [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 de-activated *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 non-delayed 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.

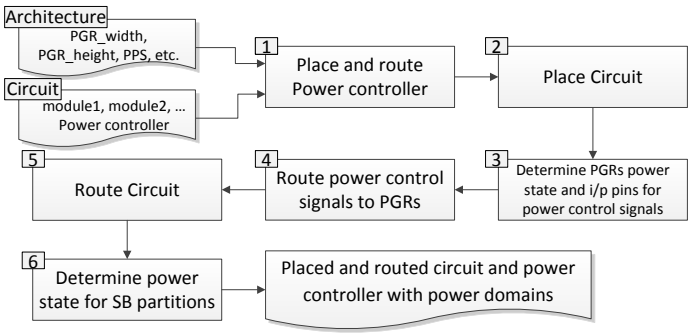


Fig. 10. CAD flow for the DCPG FPGA architecture

A. Placement and Routing

Figure 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:

- Dynamically-controlled: if only one power-gated module is mapped to the PGR's LCs.
- Always-off: if all of the PGR's LCs are empty.
- 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 dynamically-controlled, a free input pin from its bordering routing channels is selected (if one is available) to act as a sink for the control signal. Note that we cannot use predetrmined 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. 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 desribed in

Subsection V-A to the proposed architecture with a PGR size of 4x4 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:

- Dynamically-controlled: if PGR is dynamically-controlled and only one power-gated module is routed through the SB partition.
- Always-off: SB partition is not used to route signals.
- 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. Figure 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.

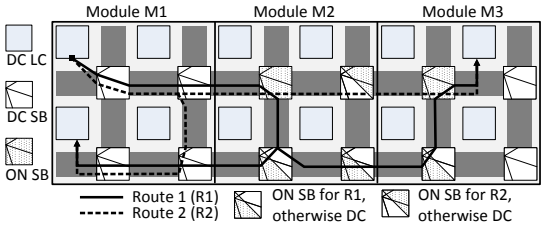


Fig. 11. Three power-gated modules placed on three PGRs with two ways shown to route the same net. The SBs used to route the net in M2 and M3 must be always-on (ON), other resources can be dynamically-controlled (DC)

In Figure 11, R2 has a smaller number of always-on SBs (larger number of always-off and dynamically-controlled SBs) compared to R1, which improves the power savings during idle periods. In this subsection, 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 Equation 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 $Congs(n)$ is the congestion cost of using node n.

$$Cost(n) = Crit_{ij} \times T(n) + (1 - Crit_{ij}) \times Congs(n) \quad (1)$$

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

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

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

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

where K and L are weighting parameters determined empirically, and $\delta_{PGR,net}$ 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 an 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.

$Cost_{Partition}$ 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 dynamically-controlled. 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 less than 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 (MCNC) benchmark circuits available with the VPR download [17] as sub-circuits (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 sub-circuits. 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. 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 dynamically controlled. Thus, a

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

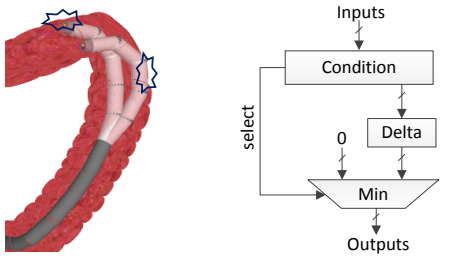


Fig. 12. Snake robot example application

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

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 our study 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 by using the DCPG FPGA architecture.

B. Robot Control System

The application presented in this subsection 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 Figure 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 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 Figure 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 about 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: LUT size $K = 4$, LC size $N = 6$, inputs per cluster $I = 16$, routing channel width $W = 90$, routing segment length $L = 4$, switch box flexibility $F_s = 3$, input pins connection box flexibility $F_{c,in} = 0.2$, and output pins connection box flexibility $F_{c,out} = 0.1$.

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 $V_{DD} = 1.0$ 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 to an architecture that does not support power gating. We assume 20% activity in doing this.

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]. Lookup tables 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 switch blocks (SBs) have been generated as follows. Unless otherwise indicated, we assume a routing channel 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 switch blocks 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 subsection, 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:

- Ungated: this is the baseline FPGA architecture that does not support power gating.
- 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.
- Dynamic-gating: this is the dynamically-controlled power gating architecture that we proposed in Section III-A.

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 routing channels (W). When varying N, we also vary the number of input pins (I) of the logic cluster. When varying N, we set $W = 90$, and when varying W we set $N = 6$ and $I = 16$.

Figures 13 and 14 show the effect of the cluster size (N) and routing channel width (W) on power gating. The results shown in the figures are for a tile that supports power gating, not including the switch block, compared to a tile that does not support power gating (ungated).

The area overhead decreases as the cluster size and the channel width increase (Figures 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 (Figures 13(c) and 14(c)). The results show that the leakage power reduction in the off state (compared to an ungated architecture) can be up to about 91% for a cluster size of 10 (channel width of 160). Figures 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: In this subsection, we study the granularity of the proposed architecture. Figure 15 shows the different results as we increase the region size.

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 4x4 tiles.

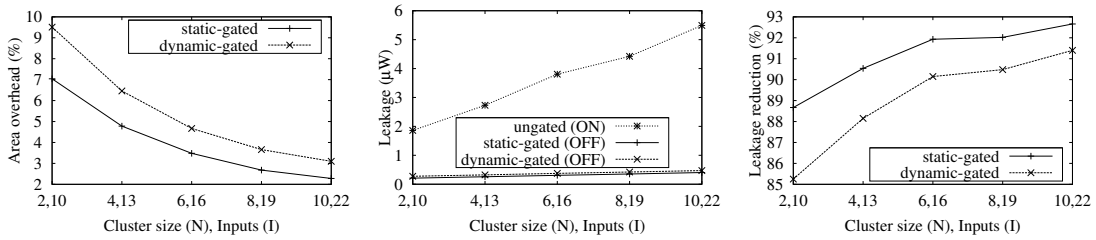
The off state leakage power of the power-gated architectures is much lower than that for the ungated architecture, leading to a leakage power reduction of more than 90% (about 95% for a PGR of 4x4 tiles). Increasing the region size by more than 4x4 tiles does not significantly increase the leakage power savings. As can be seen in Figure 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 subsection, we vary the architecture parameters that describe the switch blocks (SBs). Results are shown for SB architectures that have different segment lengths, $L = 2$ in Figure 16 and $L = 4$ in Figure 17, compared to an architecture that does not support power gating. In addition to varying the routing channel width (W), we also vary the power gating granularity of SBs by varying the number of partitions per side (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.

Figures 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]; as the number of power-gated circuit components increases, the area overhead and the leakage power increase.

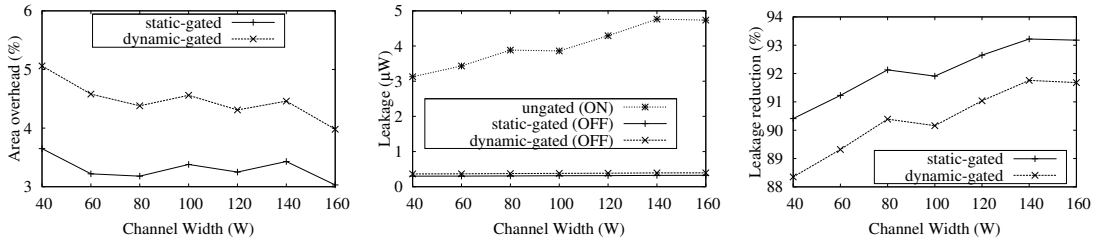
The area overhead (Figures 16(a) and 17(a)) of the power gating circuitry decreases as the channel width increases. This is because as we increase W the sleep transistor size increases at a lower rate than the increase in the number of circuit components that are powered through it.

The power gating granularity (PPS) also affects the area overhead. Fine-grained power gating results in prohibitive area overhead (more than 60% for $L = 4$ and about 100% for $L = 2$). For large values of W, the area overhead for granularities down to $PPS = 4$ ranges between between 10-16%. This area overhead, however, is only for SBs. The overall area overhead for the power gating architecture is lower. Recall from the previous subsection that the area overhead (without SBs) for a



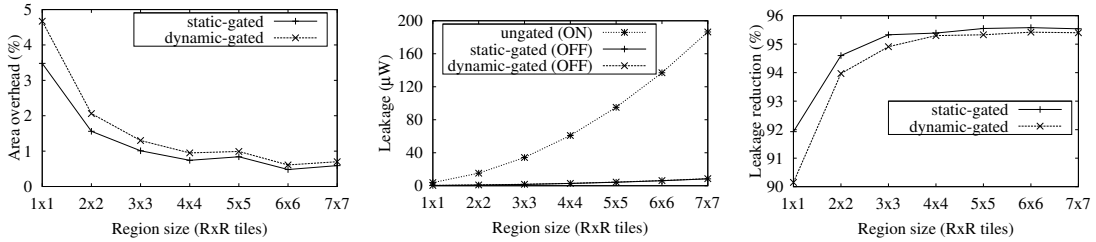
(a) Area Overhead (b) Leakage Power (c) Leakage Power Reduction

Fig. 13. Results for sweeping LC's cluster size (N). Switch blocks are not included in the results



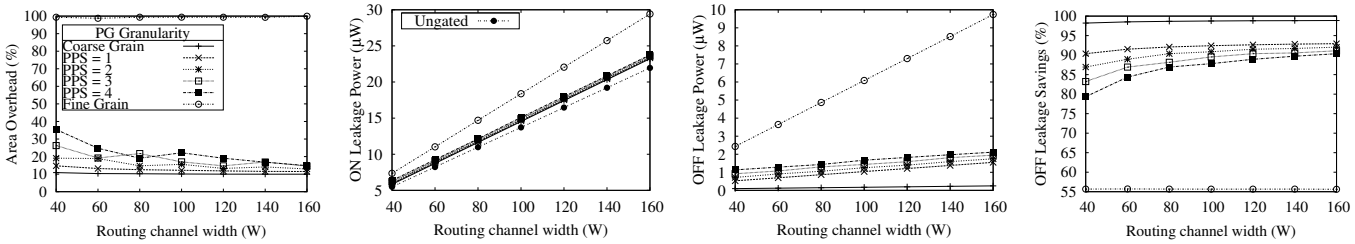
(a) Area Overhead (b) Leakage Power (c) Leakage Power Reduction

Fig. 14. Results for sweeping routing channel width (W). Switch blocks are not included in the results



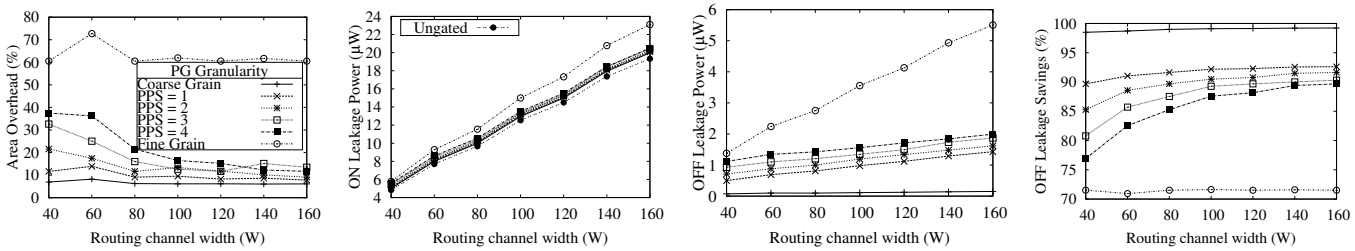
(a) Area Overhead (b) Leakage Power (c) Leakage Power Reduction

Fig. 15. Results for sweeping granularity of power gating regions (PGRs). Switch blocks are not included in the results



(a) Area Overhead (b) ON Leakage Power (c) OFF Leakage Power (d) Leakage Power Reduction

Fig. 16. Results for SBs power gating granularity by sweeping routing channel width (W). Segment length (L) = 2



(a) Area Overhead (b) ON Leakage Power (c) OFF Leakage Power (d) Leakage Power Reduction

Fig. 17. Results for SBs power gating granularity by sweeping routing channel width (W). Segment length (L) = 4

PGR of size 4x4 tiles is about 1%. The overall area overhead for the same PGR with SBs ranges between 4.7-10.3% for PPS from 0 to 4 and $W = 90$.

Figures 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 $0-4$, for example, is about 6-10% for $L = 2$ and 3.5-7.7% for $L = 4$. As we 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.

Figures 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 Figures 16(d) and 17(d) show the leakage power reduction compared to 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 (see Subsection III-D). The leakage power reduction for the proposed architecture is more than 95% for the coarse-grained power-gated SBs, and could reach more than 90% for PPS between $1-4$. For fine-grained power gating, the leakage power reduction is about 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% - 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% - 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 subsection, 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 Subsection 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 4x4 tiles, segment length of 4, and routing channel 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 Subsection IV-B.

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

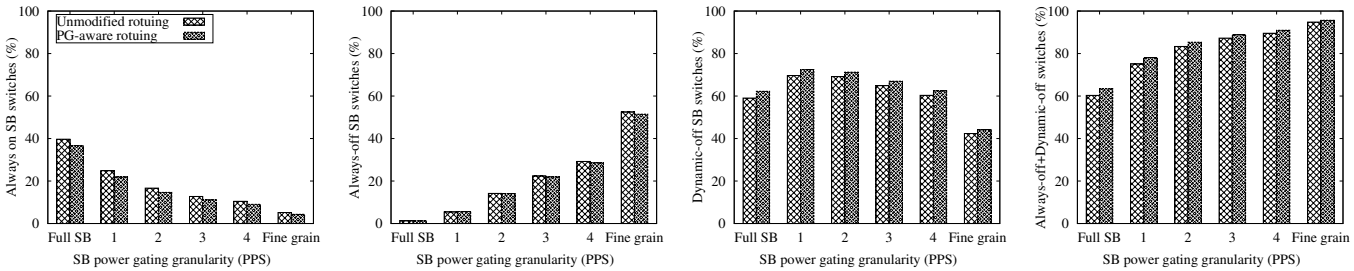
Figure 18(a) shows the percentage of always-on switches for different SB power gating granularities. For finer granularity power gating, the number of always-on SB switches decreases. This is expected since more SB switches can statically be turned off because the SB partitions they belong to are not 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 dynamically controlled 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 about 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.

Figure 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.

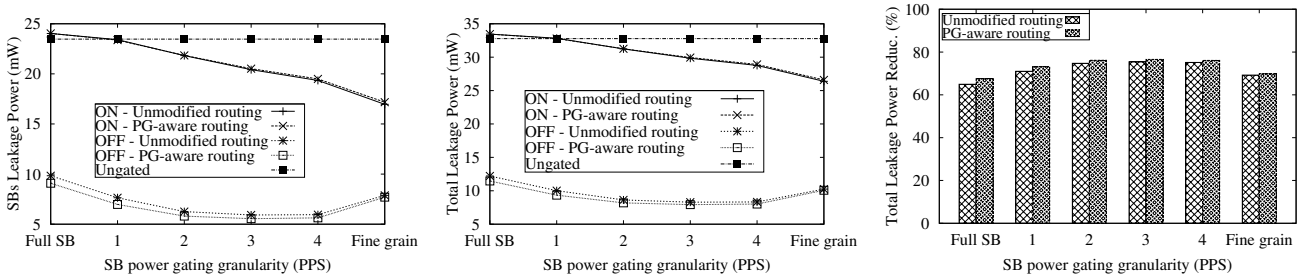
Figure 18(c) shows the percentage of dynamically controlled switches for different SB power gating granularities. These are switches that can be powered off at run time when the module they belong to becomes idle. The largest number of dynamically-controlled switches is when $PPS = 1$. Finer SB power gating granularity reduces the number of dynamically-controlled 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 dynamically controlled switches. This improvement is roughly the same as the amount of reduction in the always-on switches shown in Figure 18(a).

Finally, Figure 18(d) shows the sum of the always-off and dynamically-controlled 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 to the original VPR router; these improvements diminish as the granularity of power gating



(a) Always on (b) Always off (c) Dynamically controlled (d) Always off + dynamically controlled

Fig. 18. SBs switches power state averaged over all synthetic benchmarks using the unmodified and power gating-aware routing



(a) SBs Leakage Power (b) Total Leakage Power (c) Total Leakage Power Reduction

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

decreases as explained above.

2) Leakage Power: In this subsection, 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 dynamically-controlled components are turned off. In both cases, the always-off components are assumed to be turned off at configuration time.

Figure 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 dynamically-controlled 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.

Figure 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 dynamically-controlled switches that can be turned off when an application circuit is idle, as has been shown in Figure 18(c).

Figure 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 Figure 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 Figures 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.

Figure 19(c) shows the upper limit of reduction in the OFF leakage power compared to 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 about 68%. For $PPS = 3$, the reduction is about 77%.

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

aware routing.

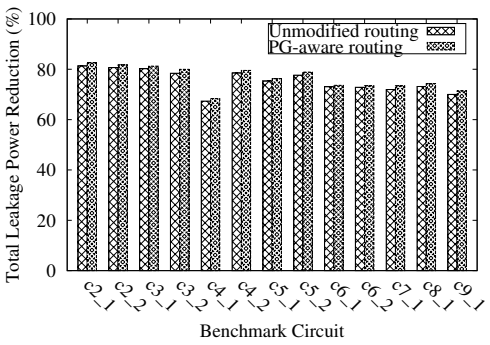


Fig. 20. Total leakage power reduction for individual synthetic circuits

D. Example Application Results

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

Figure 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 about 19% energy savings. Compared with the second baseline (includes clock gating), the proposed architecture could achieve about 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.

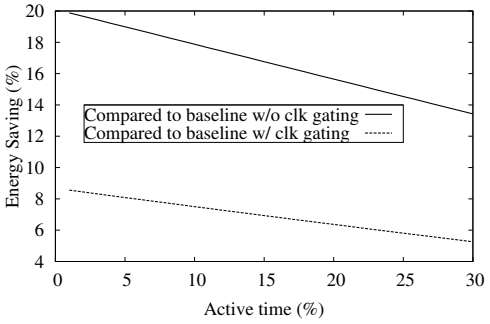


Fig. 21. Energy reduction for iSnake's control system in DCPG FPGA

VII. Conclusion and Future Work

In this paper, we present an FPGA architecture that supports dynamic power gating. This architecture enables powering down modules in an FPGA when they are idle, and powering them up again when active, thus reducing the overall energy consumption. 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 about 10.3% when the power gating region size is 4x4 tiles and the number of power-gated SB partitions per side 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 about 8% energy saving can be achieved by the proposed architecture when compared to 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 *Proceedings of the 14th ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 21–30, 2006.

[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*, pp. 624–633, 2000.

[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*, pp. 13–22, 2009.

[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 International Symposium on Field-Programmable Gate Arrays*, pp. 3–11, 2006.

[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*, pp. 42–50, 2004.

[6] J. Hussein, M. Klein, and M. Hart, "Lowering Power at 28 nm with Xilinx 7 Series FPGAs." Xilinx, Inc. white paper WP389 (v1.1), Jun. 2011.

[7] "Meeting the Low Power Imperative at 28 nm." Altera Corporation, white paper WP-01158-2.0, Nov. 2011.

[8] 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.

[9] 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*, pp. 645–650, 2005.

[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 *Proceedings of the 12th International Symposium on Field-Programmable Gate Arrays*, pp. 51–58, 2004.

[11] 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*, pp. 651–656, 2005.

[12] A. A. M. Bsoul and S. J. E. Wilton, "An FPGA Architecture Supporting Dynamically Controlled Power Gating," in *Proceedings of The IEEE International Conference on Field-Programmable Technology (FPT)*, pp. 1–8, Dec. 2010.
[13] A. A. M. Bsoul and S. J. E. Wilton, "An FPGA with Power-Gated Switch Blocks," in *Proceedings of The IEEE International Conference on Field-Programmable Technology (FPT)*, pp. 87–94, Dec. 2012.
[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 *Proceedings of the 17th IEEE/ACM International Symposium on Low-Power Electronics and Design*, ISLPED '11, pp. 223–228, IEEE Press, 2011.
[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 *Field Programmable Logic and Applications (FPL), 2013 23rd International Conference on*, pp. 1–4, Sept 2013.
[16] V. Betz, J. Rose, and A. Marquardt, eds., *Architecture and CAD for Deep-Submicron FPGAs*. Norwell, MA, USA: Kluwer Academic Publishers, 1999.
[17] 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-DriverRouting, Heterogeneity and Process Scaling," in *Proceeding of the 17th ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 133–142, 2009.
[18] M. Klein, "Power Consumption at 40 and 45 nm." Xilinx, Inc. white paper WP298 (v1.0), April 2009.
[19] 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*, pp. 203–213, 2000.
[20] M. Keating, D. Flynn, R. Aitken, A. Gibbons, and K. Shi, *Low Power Methodology Manual: For System-on-Chip Design*. Springer Publishing Company, Incorporated, 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 *Proceedings of the International Symposium on Field-Programmable Gate Arrays*, FPGA '12, pp. 245–254, 2012.
[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 Electronics*, 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 *FCCM*, pp. 149–156, 2010.
[24] "ABC: A System for Sequential Synthesis and Verification."
[25] A. R. Marquardt, "Cluster-Based Architecture, Timing-Driven Packing and Timing-Driven Placement for FPGAs," Master's thesis, University of Toronto, 1999.
[26] K.-W. Kwok, K. H. Tsoi, V. Vitiello, J. Clark, G. Chow, W. Luk, and G.-Z. Yang, "Dimensionality Reduction in Controlling Articulated Snake Robot for Endoscopy Under Dynamic Active Constraints," *IEEE Transactions on Robotics*, vol. 29, no. 1, pp. 15–31, 2013.
[27] J. Detrey and F. de Dinechin, "FPLibrary, A VHDL Library of Parametrisable Floating-Point and LNS Operators for FPGA." http://www.ens-lyon.fr/LIP/Arenaire/Ware/FPLibrary/, 2004.
[28] "Quartus II University Interface Program." http://www.altera.com/education/univ/research/quip/unv-quip.html.
[29] "Predictive Technology Model (PTM)." http://www.eas.asu.edu/~ptm/.
[30] E. Hung, S. Wilton, H. Yu, T. Chau, and P. H. W. Leong, "A Detailed Delay Path Model for FPGAs," in *Field-Programmable Technology, 2009. FPT 2009. International Conference on*, pp. 96–103, IEEE, 2009.
[31] N. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*. USA: Addison-Wesley Publishing Company, 4th ed., 2011.
[32] T. Pi and P. J. Crotty, "FPGA Lookup Table with Transmission Gate Structure for Reliable Low-Voltage Operation," Dec. 2003. US Patent 6,667,635.
[33] 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*, pp. 41–48, 2004.
[34] S.-C. Wong, G.-Y. Lee, and D.-J. Ma, "Modeling of Interconnect Capacitance, Delay, and Crosstalk in VLSI," *Semiconductor Manufacturing, IEEE Transactions on*, vol. 13, pp. 108 –111, Feb. 2000.
[35] S. Huda, J. Anderson, and H. Tamura, "Charge Recycling for Power Reduction in FPGA Interconnect," in *Field Programmable Logic and Applications (FPL), 2013 23rd International Conference on*, pp. 1–8, Sept 2013.