FPGA-SPICE: A Simulation-Based Architecture Evaluation Framework for FPGAs

Xifan Tang, *Member, IEEE*, Edouard Giacomin, *Student Member, IEEE*, Giovanni De Micheli, *Fellow, IEEE*, and Pierre-Emmanuel Gaillardon, *Senior Member, IEEE*

***Abstract*—In this paper, we developed a simulation-based architecture evaluation framework for field-programmable gate arrays (FPGAs), called FPGA-SPICE, which enables automatic layout-level estimation and electrical simulations of FPGA architectures. FPGA-SPICE can automatically generate Verilog and SPICE netlists based on realistic FPGA configurations and a high-level eTtensible Markup Language-based FPGA architectural description language. The outputted Verilog netlists can be used to generate layouts of full FPGA fabrics through a semicustom design flow. SPICE simulation decks can be generated at three levels of complexity, namely, full-chip-level, grid-level, and component-level, providing different tradeoff between accuracy and simulation time. In order to enable such level of analysis, we presented two SPICE netlist partitioning techniques: loads extraction and parasitic net activity estimation. Electrical simulations showed that averaged over the selected benchmarks, the grid-/component-level approach can achieve 6.1×/7.5× execution speed-up with 9.9%/8.3% accuracy loss, respectively, compared to the full-chip level simulation. FPGA-SPICE was showcased through three different case studies: 1) an area breakdown analysis for static random access memory-based FPGAs, showing that configuration memories are a dominant factor; 2) a power breakdown comparison to analytical models, analyzing the source of accuracy loss; and 3) a robustness evaluation against process corners, studying their impact on energy consumption of full FPGA fabrics.**

***Index Terms*—Circuit simulation, design automation, design tools, field programmable gate arrays, rapid prototyping.**

I. Introduction

MAINSTREAM field-programmable gate array (FPGA) architecture evaluation tools rely on analytical models to predict area, delay, and power [1]–[9]. Even though these analytical models have been intensively refined for modern FPGAs, they share some common limitations.

Manuscript received May 10, 2018; revised August 17, 2018 and October 24, 2018; accepted November 24, 2018. This work was supported in part by the Swiss National Science Foundation under Project 200021-146600, in part by the United States–Israel Binational Science Foundation under Grant 2016016, and in part by the Air Force Research Laboratory and Defense Advanced Research Projects Agency under Grant FA8650-18-2-7855. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) or the U.S. Government. *(Corresponding author: Xifan Tang.)*

X. Tang, E. Giacomin, and P.-E. Gaillardon are with the Laboratory for NanoIntegrated Systems, Electrical and Computer Engineering Department, University of Utah, Salt Lake City, UT 84112 USA (e-mail: xifan.tang@utah.edu; pierre-emmanuel.gaillardon@utah.edu).

G. De Micheli is with the Integrated Systems Laboratory, School of Computer and Communication Sciences, École Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland.

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.2018.2883923

1) The area evaluation considers only the core logic and not the configuration peripheral circuits, whose design topology may lead to a difference in the total area.
2) The area models focus on standalone modules but neglect full-chip-level interconnecting wires. Consequently, the area of multiplexers, which are the most frequent components in FPGA architectures, can be underestimated by up to 139% when compared to actual layouts [11].
3) The power consumption is highly dependent on the transistor technology and circuit topology while the analytical models support only a limited subset of them [5]–[11].
4) The power consumption is sensitive to the actual FPGA configurations that differ from a design to another, while their impacts are not considered in the analytical models. For example, the power of lookup tables (LUTs) under high signal switching activity can be overestimated by 20% when compared to SPICE simulations [8].

Consequently, the architecture-level conclusions could be misleading. For instance, by under/overestimating the area/power contribution of FPGA components, the area/power breakdown results could be inaccurate, leading to wrong directions in optimizing FPGA architectures [10]–[12].

In this paper, we develop a simulation-based architecture evaluation framework for FPGAs, called FPGA-SPICE, which is released to the public [13]. FPGA-SPICE enables full-chip-level layout estimation and electrical simulations of FPGA architectures, which eases area and power studies. Being tightly integrated within the popular academic architecture exploration tool suite Verilog-to-Routing (VTR) [2], FPGA-SPICE supports an extended architecture description language [3] to consider transistor-level parameters related to each module inside the FPGA architecture under evaluation. FPGA-SPICE can automatically generate the following.

1) Transistor-level SPICE netlists of FPGA fabrics and associated simulation decks at three levels of complexity: full-chip-level, grid-level, and component-level, which tradeoff between accuracy and simulation time. We develop netlist splitting strategies to slice full-chip-level netlists into grid-level netlists each of which consists of either a complete transistor-level logic block netlist, or component-level netlists which consider individual circuit elements, such as LUTs, flip-flops (FFs), and multiplexers. Electrical simulations show that averaged over the selected benchmarks, the grid-/component-level approach can achieve 6.1×/7.5× execution speed-up with 9.9%/8.3% accuracy loss, respectively, when compared to the full-chip level.
2) Verilog netlists of full FPGA fabric and associated testbenches supporting two types of configuration

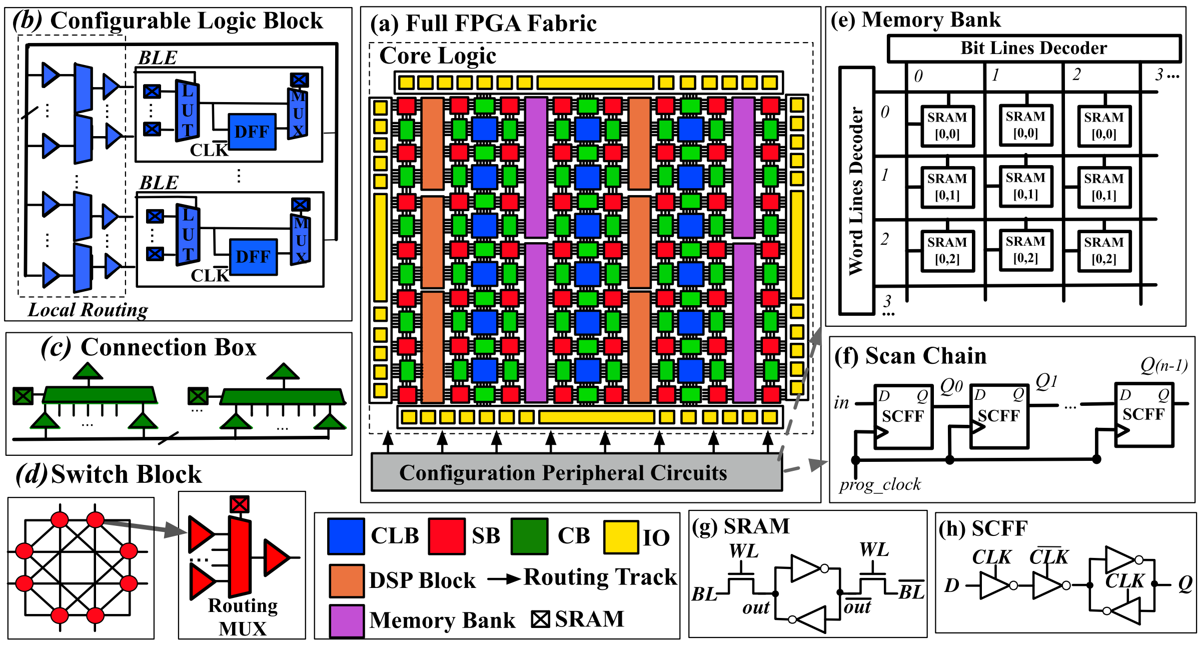

Fig. 1. (a) Full FPGA fabric consisting of a core logic and a configuration peripheral circuit. (b) Detailed architecture of a CLB. (c) Detailed architecture of a CB. (d) Detailed architecture of an SB. (e) Memory-bank-style configuration peripheral circuits. (f) Scan-chain-based configuration peripheral circuits. (g) Compact transistor-level design of an SCFF. (h) Transistor-level design of a SRAM.

peripherals: scan-chains and memory address decoders. The Verilog netlists built at gate-level can be used to implement layouts of full FPGA fabrics.

To capture the effect of different FPGA configurations, the outputted SPICE and Verilog netlists include mapping, placement, and routing results as well as technological information.

To showcase the power of FPGA-SPICE, we present three case studies.

1) We perform a layout-level area breakdown analysis for full FPGA fabrics where we show that configuration memories are a dominant factor in total area.
2) We do a power breakdown comparison to analytical power models, i.e., VersaPower [8], where we analyze the source of accuracy loss.
3) We evaluate the robustness against process corners, and we study their impact on the energy consumption of full FPGAs.

Note that FPGA-SPICE is not limited to the presented case studies. It can also be used for prototyping FPGAs, verifying functionality, analyzing hotspot management, and evaluating novel FPGAs based on advanced technologies [14]–[19].

This paper is organized as follows. In Section II, we report a brief background on FPGA architectures and academic architecture exploration tools. In Section III, we show the electrical design automation (EDA) flows exploiting FPGA-SPICE. In Section IV, we introduce the architecture description language supported by FPGA-SPICE. In Section V, we present the core engine of FPGA-SPICE. In Section VI, we report experimental results. In Sections VII and VIII, we conclude the paper and inspire the future work, respectively.

II. Background

In this section, we first introduce the principles of modern FPGA architectures. Then, we comment on the state-of-the-art academic architecture exploration tools that are compatible with modern FPGA architectures.

A. Island-Style FPGA Fabric

Modern FPGA architectures are based on island-style fabrics that are interconnected to rich programmable routing resources. Fig. 1 describes the full fabric of modern FPGA architectures, which consists of two parts.

1) The core logic is the centerpiece of an FPGA that realizes logic functions. It consists of an array of configurable logic blocks (CLBs) which are surrounded by a global routing architecture. The global routing architecture is built with connection blocks (CBs) that connect CLB pins to routing tracks, and switch boxes (SBs) that interconnect routing tracks. A CLB typically comprises a number of basic logic elements (BLEs), each of which consists of an LUT, a D-FF, and an output selector (2:1 multiplexer). A local routing architecture consists of fully/sparsely populated crossbars that interconnect BLE pins and CLB pins. In order to improve the area and speed of arithmetic-intensive implementations, commercial FPGAs [20]–[24] apply specific block in the CLBs and architectural enhancements. Columns of CLBs are replaced by heterogeneous blocks such as memory banks and digital signal processing (DSP) blocks. BLEs comprise of fracturable LUTs [25] and hard carry chains [26]. Local routing architecture also interconnects adjacent CLBs to provide highways between neighbors [27], [28]. High-speed transceivers and hard-wired logic blocks are used to implement ultrahigh-speed I/Os in addition to the standard I/O blocks.
2) The configuration peripheral circuits aim at programing each static random access memory (SRAM) of the LUTs and routing multiplexers belonging to the core logic. Most SRAM-based FPGAs use scan chains [20]–[24], while nonvolatile FPGAs are more efficiently using memory address decoders [29]. When scan chains are employed, SRAMs are embedded into scan-chain FFs (SCFFs). SCFFs are connected in a chain, allowing

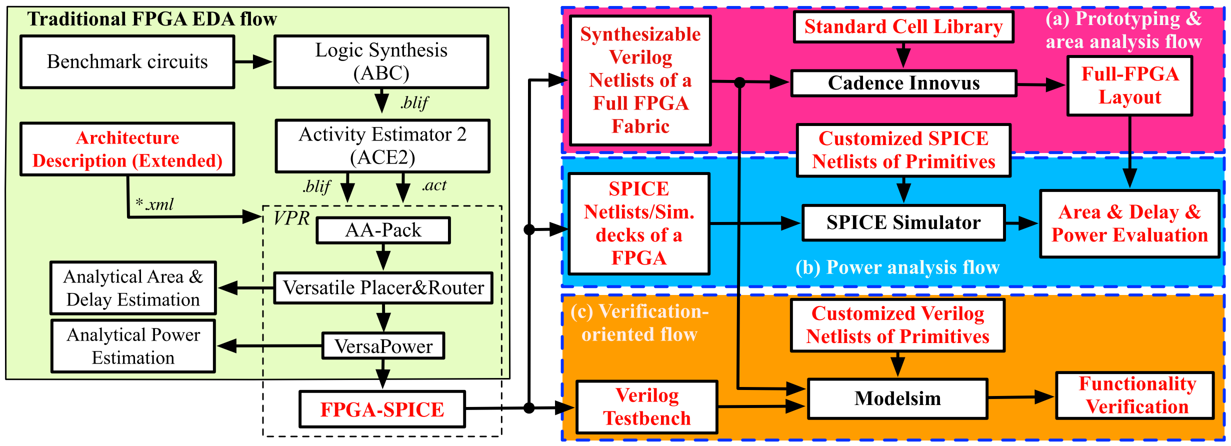
Traditional FPGA EDA flow
Benchmark circuits
Logic Synthesis (ABC)
.blif
Architecture Description (Extended)
Activity Estimator 2 (ACE2)
.blif
.act
*.xml
VPR
AA-Pack
Analytical Area & Delay Estimation
Versatile Placer&Router
Analytical Power Estimation
VersaPower
FPGA-SPICE
Synthesizable Verilog Netlists of a Full FPGA Fabric
Standard Cell Library
(a) Prototyping & area analysis flow
Cadence Innovus
Full-FPGA Layout
Customized SPICE Netlists of Primitives
SPICE Netlists/Sim. decks of a FPGA
SPICE Simulator
Area & Delay & Power Evaluation
(b) Power analysis flow
(c) Verification-oriented flow
Customized Verilog Netlists of Primitives
Verilog Testbench
Modelsim
Functionality Verification

Fig. 2. Detailed EDA flows based on FPGA-SPICE in the purpose of (a) prototyping and area analysis, (b) power analysis, and (c) functionality verification.

configuration bits to be serially loaded to each SCFF, as illustrated in Fig. 1(f) and (h). The output ports of SCFFs are directly connected to the programmable resources, i.e., LUTs and multiplexers. When memory address decoders are used, SRAMs are accessed through bit lines (BLs) and word lines (WLs). As illustrated in Fig. 1(g), SRAM cells belonging to the same row share a BL signal while each column is controlled by a WL signal. All the BL and WL signals are controlled by two decoders. Each SRAM cell can be individually programed when its associated BL and WL are enabled by manipulating the two decoders.

B. Academic FPGA Architecture Evaluation Tools

The performance of an FPGA relies on hardware/software cooptimizations. In order to study and optimize various possible architecture, academic architecture exploration tool suites have been developed. In this paper, we focus on the VTR tool suite whose flow is highlighted in green in Fig. 2. The logic synthesis tool ABC [30] optimizes the benchmark circuits and performs a technology mapping. The activity estimator activity estimation (ACE2) [1] computes the signal activities of all the internal nodes in the benchmark circuits. Finally, the tool versatile placement and routing (VPR) [2] packs, places, and routes the circuits into a hypothetic FPGA defined with an eTtensible Markup Language (XML)-based architecture description language. To model highly flexible FPGA architectures, the architecture description language can precisely describe the hierarchy and complex interconnects inside modern FPGA CLBs [3]. In the packing stage, LUTs and FFs are clustered into CLBs. Placement determines the physical positions of CLBs in the FPGA fabric. The routing stage maps the nets of CLBs into routing architectures. After routing, the built-in analytical models estimate the area and delay consumption of a benchmark circuit, while power estimation is performed by VersaPower [8], which is based on signal activities and analytical power models.

While classical architecture evaluation tools employ analytical models to benchmark FPGA architectures, recent works [8], [11], [14], [15] use layout-level estimation and electrical simulations to study the area, delay, and power characteristics. However, these works share common limitations.

1) Most architecture-level conclusions are drawn by summing the area, delay, and power of standalone FPGA components instead of using full FPGA fabrics. Consequently, the impacts of some global parasitics are ignored. For example, area and power of interconnecting wires between components can only be captured in full FPGA layouts.
2) A limited number of circuit design topologies and FPGA architectures were targeted, causing the evaluation results to be less meaningful. For instance, only fully-decoded multiplexer structures are supported in [15], while modern FPGAs typically employ two-level or one-level multiplexer structures at advanced technology nodes.
3) Most previous works [1]–[12] focused on studying the core logic of FPGAs, while very limited works [14], [15] considered the full FPGA fabrics. Moreover, as different configuration peripheral circuits are used in commercial products [20]–[24], [29], limited work has been done to study their impacts on FPGAs.

In particular, the current state-of-the-art power estimation techniques for FPGAs are based on probabilistic activity estimation [1] and analytical power models [5]–[8]. However, they face challenges from continuously evolving FPGA architectures and technologies.

1) *Infinite Number of Baselines to Consider:* SPICE simulations are regarded as the golden reference for validating the accuracy of power models [5]–[8]. Early works [5], [6] rely on SPICE results to derive the key modeling parameters, e.g., the capacitance in switch-level models. To achieve high accuracy, modern tools [7], [8] intensively use SPICE results to build power property libraries for FPGA architectures. However, the reconfigurable nature of FPGAs leads to enormous configurations being simulated and input signal patterns being considered, which may significantly differ from one design to another. For example, only a four-input LUT has 256 input patterns and 2^{16} different configurations, which is very timing-consuming to enumerate. As a result, most previous works investigated up to a few hundreds of input patterns in SPICE simulations and very few report how many configurations have been considered [5]–[8]. Despite being considered the most accurate baseline, very limited full-chip-level SPICE simulations (for < 5 benchmarks) have been reported [6] and no methods for SPICE decks autogeneration have been proposed to the best of our knowledge.

2) *Limited Accuracy Validation:* Due to the difficulties in achieving accurate baselines, limited verification can be performed to assess power models. Previous works typically focus on a small set of FPGA architectures and a specific technology node [5]–[8]. Modern academic FPGA architecture exploration tools [2], [4], [9] are capable of modeling highly flexible FPGA architectures [3]. The high parameterization in FPGA architecture opens a large design space to explore but also challenges the generality of power modeling. Even though previous works [5]–[8] have tested FPGA architectures under different LUT sizes and CLB capacities, it is extremely difficult to guarantee their accuracy for any FPGA architecture.
3) *Limited Technology and Circuit Designs:* Different from SPICE simulations, power models are typically refined for a specific technology node and support a limited number of circuit designs. For instance, FPGA-EVA-LP2 only consider an academic 100-nm technology and multiplexers based on pass transistor [6], while VersaPower only supports circuit designs with fixed transistor sizes [8]. To adopt another process technology, significant manual efforts are required since all the parameters have to be reworked based on SPICE simulations. Otherwise, as circuit designs may vary from a technology node to another, the power models may have to be reworked with a strong expertise being required.

In this paper, we present a novel architecture evaluation tool, FPGA-SPICE, to address these limitations and provide a more realistic study on area and power breakdown with full-chip-level layout estimation and electrical simulations. To the best of our knowledge, this is the first work that autogenerates full-chip-level SPICE netlists, supports any FPGA architecture and exploits them for power estimation purposes.

III. FPGA-SPICE-Based EDA Flow

To enable full-chip-level electrical simulations and automated layout generation, FPGA-SPICE provides an interface between the traditional EDA flow and various SPICE/Verilog-based EDA tools, as depicted in Fig. 2. FPGA-SPICE exploits the description of the architecture provided traditionally to VTR, the mapped netlists and the estimated signal activities to dump SPICE/Verilog netlists and the associated testbenches for the implemented benchmarks. As exemplified in Fig. 2(a)–(c), the tool can subsequently be followed by extra EDA steps to achieve different research purposes.

1) *Layout-Level Area Estimation:* FPGA-SPICE autogenerates the gate-level Verilog netlists modeling full FPGA fabrics, and then a semicustom design flow, e.g., Cadence Innovus [34], can be used to produce an actual layout.
2) *SPICE-Based Power Analysis:* FPGA-SPICE autogenerates the transistor-level SPICE netlists of configured FPGA fabrics, and then an electrical simulator, e.g., Synopsys version of SPICE. Expansion of SPICE is Simulation Program with Integrated Circuit Emphasis (HSPICE) [35], can be used to estimate the power consumption.
3) *Functionality Verification:* FPGA-SPICE autogenerates the gate-level Verilog netlists and the associated testbenches, and then a hardware description language (HDL) simulator, e.g., Modelsim [36], can be used to verify the functional correctness of the full FPGA fabrics.

To support a versatile modeling of circuit designs and FPGA architectures, we develop an extended architecture description language for FPGA-SPICE, which is detailed in Section IV. To be efficient in performing area and power analysis, we develop netlist partitioning algorithms, which are detailed in Section V.

IV. Architecture Description Language

To support the diverse functionality shown in Fig. 2, FPGA-SPICE uses the extended architecture description language with the following.

1) Enriched transistor-level design parameters for primitive blocks, such as LUTs, FFs, and multiplexers (see Section IV-A). Note that FPGA-SPICE can also use user-defined Verilog and SPICE netlists *in lieu* of automatically generated blocks. This is an interesting feature to model fine-grain FPGA components, such as SRAMs, whose performances are highly dependent on the technology and the circuit structure. This supports the capability to study the system-level impact of the circuit elementary blocks, thereby enabling interesting circuit/architecture cooptimization opportunities.
2) Architectural parameters modeling different configuration circuitry, such as memory access decoders and scan chains (see Section IV-B).
3) Additional parameters to model the physical structure of I/O blocks (see Section IV-C).

A. Transistor-Level Circuit Netlist Generation

FPGA-SPICE extends the architecture description language of [3]. This architecture description language can model highly flexible FPGA architectures at an abstract level. In the extension, we add transistor-level circuit design parameters for modeling the circuit components of the FPGA modules.

1) Technology Declaration: First, the transistor model and basic geometrical properties are defined in XML nodes `tech_lib` and `transistor`, as follows:

```
  <tech_lib lib_path="40nm.lib" type="TT"
nominal_vdd="0.9"/>
  <transistors pn_ratio="2">
      <nmos chan_length="40e-9"
min_width="140e-9"/>
      <pmos chan_length="40e-9"
min_width="140e-9"/>
  </transistors>
```

The channel length, the transistor width, and the ratio between p-type and n-type transistors are defined in the XML properties `nmos` and `pmos`, respectively.

2) SPICE Model Declaration: The transistor-level circuit design parameters of an FPGA module are defined under an XML property called `spice_model`. The VTR architecture description language models all logic blocks with a hierarchy of XML properties, called `pb_type`. We create a property `spice_model_name` under `pb_type` to link the logic blocks to the defined `spice_model`. The following code shows an example, where a six-input LUT spice model, `lut6`, is defined and linked to a logic block, `n_lut6`:

```
  <spice_model type="lut" name="lut6"
sp_netlist="lut6.sp"
  verilog_netlist="lut6.v">
```

```
     <port type="input" prefix="in"
size="6" is_global="false"
is_clock="false"/>
     <port type="output" prefix="out"
size="1"/>
     <port type="sram" prefix="sram"
size="64" spice_model_name="sram6T"
     default_val="1"/>
  <spice_model>
  <pb_type name="n_lut6"
spice_model_name="lut6">
  </pb_type>
```

Under the XML property spice_model, the ports of an LUT should be defined by providing the size, port type, and port name. Since the circuit designs of some of the FPGA modules are highly dependent on the technology nodes, such as SRAMs, hard logic blocks, or FFs, FPGA-SPICE allows user-customized SPICE netlists for each defined spice model. In the previous example of lut6, user-customized SPICE and Verilog netlists are defined in the XML properties, sp_netlist and verilog_netlist, respectively.

In an FPGA, circuit topologies of different blocks, such as channel wires, multiplexers, and LUTs, can be really different in order to meet the architectural needs. To be customizable, FPGA-SPICE provides transistor-level design parameters, based on which their SPICE/Verilog netlists can be automatically generated. In the following parts, we will discuss the details of transistor-level modeling.

3) Multiplexers: The multiplexers used across an FPGA have diverse sizes and fan-outs depending on their locations, e.g., in local routing or global routing. In this context, different circuit-level optimizations, such as transistor sizing and the use of tapered buffers, may apply. The transistor sizes and buffer allocation can be specified in the Verilog and SPICE model definitions. The presence or absence of input–output inverters/buffers can be declared by setting the XML properties exist and type. In addition, the size and design topology can be customized by properly setting the XML properties tapered, tap_buf_level, and f_per_stage. The use of a pass-gate logic or a transmission-gate logic design style can be specified in the XML property pass_gate_logic. The sizes of the transistors used in the pass gate or transmission gate logic can be specified in the XML properties nmos_size and pmos_size.

Transistor-level circuit design examples of global routing multiplexers and local routing multiplexers are shown in Fig. 3(a) and (b), respectively. The treelike structure of multiplexers is depicted in Fig. 3(c). The transistor-level circuit design of a global routing multiplexer in Fig. 3(a) can be modeled by the following code:

```
  <spice_model type="mux" name="sb_mux"/>
     <design_technology type="cmos"
structure="one-level"/>
     <input_buffer exist="on"
spice_model_name="inv1"/>
     <output_buffer exist="on"
spice_model_name="tap_buf4"/>
     <pass_gate_logic
spice_model_name="tgate"/>
     <port type="input" prefix="in"
size="4"/>
     <port type="output" prefix="out"
size="1"/>
```

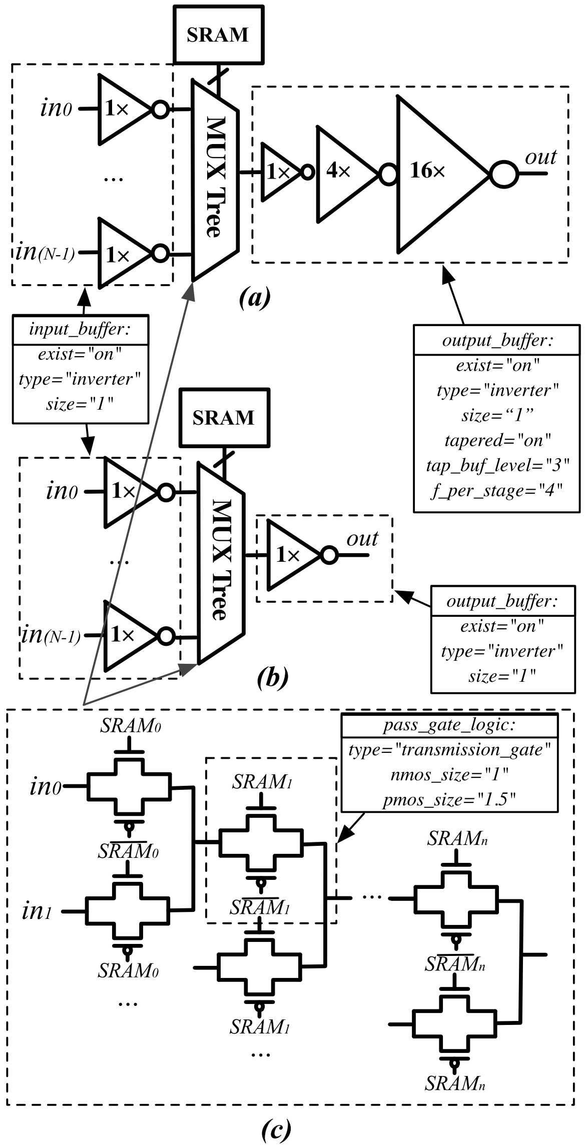

Fig. 3. Transistor-level circuit design of (a) global routing multiplexer, (b) local routing multiplexer, and (c) internal treelike structure.

```
     <port type="sram" prefix="sram"
size="4"/>
  </spice_model>
```

Global routing multiplexers require an output-tapered buffer [32], in order to drive the long routing metal wires as well as downstream loads due to the SB and CB multiplexers [31]. The output-tapered buffer in Fig. 3(a) consists of three stages and the logical effort between stages is four. Input buffers are added to restore the input signals and drive the treelike internal structure of the multiplexer. Fig. 3(b) depicts the circuit design of a local routing multiplexer which interconnects CLB input pins to BLE input pins. Because the fan-out of the multiplexer is typically small (one or two inverters), only a minimum-size output inverter is required.

FPGA-SPICE translates the architectural needs and design topologies into multiplexer SPICE netlists and initializes the SRAM configurations according to VPR routing results.

4) Lookup Tables: LUTs are crucial components in FPGAs as they serve as combinational function generators. Fig. 4 illustrates the transistor-level circuit design of an LUT considered

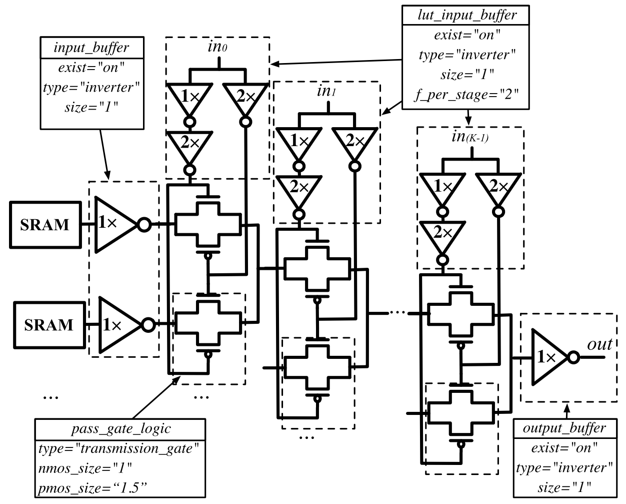

Fig. 4. Example of the transistor-level design of an LUT.

in this paper, which includes SRAMs, decoded multiplexers, and buffers [10].

The following XML properties are used to describe the circuit characteristics of the implementation in Fig. 4. The `input_buffer` properties model the buffers between the inputs of internal multiplexer and SRAM outputs. The `lut_input_buffer` properties describe the buffers at LUT inputs, where `f_stage` denotes the logic efforts of the input buffers. FPGA-SPICE decodes technology mapping results of LUTs to properly initialize the SRAM bits.

```
  <spice_model type="lut" name="lut6">
    <lut_input_buffer exist="on"
spice_model_name="buf_size2"/>
    <input_buffer exist="on"
spice_model_name="inv1">
    <output_buffer exist="on"
spice_model_name="inv1">
    <pass_gate_logic
    spice_model_name="tgate"/>
     <port type="input" prefix="in"
size="6" is_global="false"
is_clock="false"/>
     <port type="output" prefix="out"
size="1"/>
     <port type="sram" prefix="sram"
size="64" spice_model_name="sram6T"
     default_val="1"/>
  </spice_model>
```

5) Channel Wire: In modern FPGAs, channel wires are nonnegligible modules owing to the facts that the CLB area is significant and long interconnecting metal wires are required. A length-L channel wire is abstracted as L cascaded segments, each of which spans a unique CLB. Fig. 5(a) depicts a length-2 channel wire in unidirectional routing architecture [9]. The channel wire is divided into two segments, namely, $Segment_0$ and $Segment_1$.

We assume that the inputs of CBs are connected to the middle of segments, breaking segments into two parts. We model each part of the segments with distributed RC lines. The type of RC lines, i.e., either π-type or T-type, is specified in the XML property *model_type*. The number of levels of an RC line can be customized by setting the XML property *level*. The total resistances and capacitance of a segment can be defined in the XML properties *res_val* and *cap_val*, respectively. The following example describes the RC models of segments in Fig. 5(b), corresponding to the segments in Fig. 5(a).

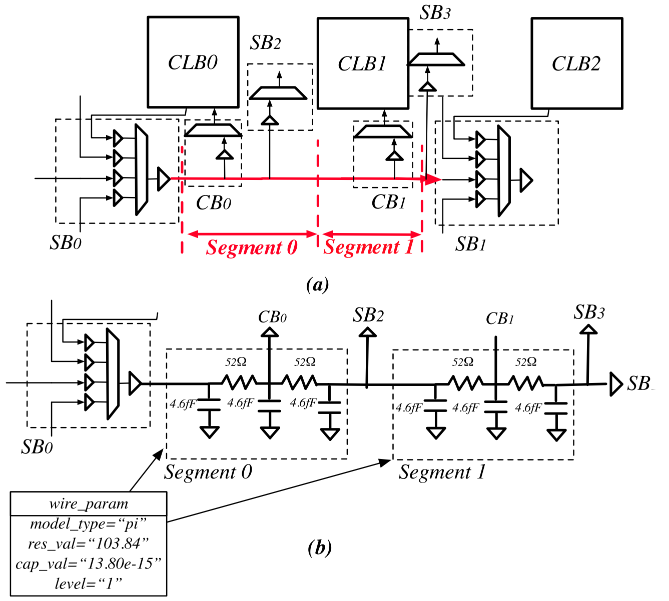

Fig. 5. (a) Length-2 unidirectional wire (highlighted in red) within FPGA routing architecture. (b) Corresponding RC modeling of segments.

```
  <spice_model type="chan_wire"
name="chan_segment">
     <wire_param model_type="pi"
res_val="103.84"
     cap_val="13.80e-15" level="1"/>
  </spice_model>
```

B. Configuration Circuitry

As introduced in Fig. 1, memory bits of FPGAs can be accessed by different types of configuration circuits, leading to difference in the full-chip area and also other merits. The area of scan chains is linear to the number of memory bits, while the area of memory address decoders has a square root relationship to the number of memory bits. However, since most FPGA researches only focus on the logic core, the exact impact of configuration circuits has not been carefully examined. As FPGA-SPICE aims at accurately modeling a full FPGA fabric with SPICE or Verilog netlists, the architecture description language is extended to model the configuration circuits. Under the XML node `sram`, the details of configuration circuits can be specified separately for SPICE and Verilog generator, as follows:

```
  <sram area="6">
     <verilog organization="memory_bank"
spice_model_name="sram6T_blwl"/>
     <spice organization="standalone"
spice_model_name="sram6T"/>
  </sram>
```

In the example of the XML node `verilog`, the type of configuration circuit can be specified by the XML

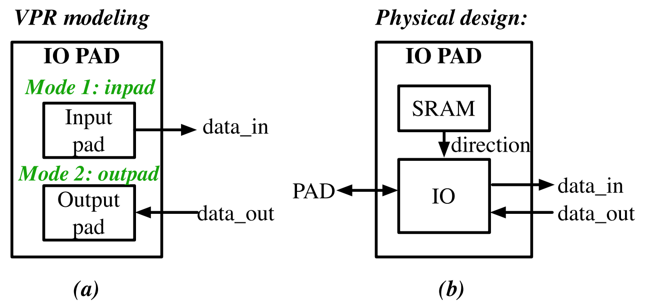

Fig. 6. I/O pad. (a) VPR abstract-level modeling. (b) Actual physical design.

property `organization`. The supported configuration circuits include memory-bank style (shown in Fig. 1) and scan chains (shown in Fig. 1). The memory model accessed by the configuration circuits can be declared in the XML property `spice_model_name`, which is linked to a defined `spice_model` devoted to the transistor-level designs of a SRAM and an SCFF (see details in Fig. 1).

As a result, the FPGA-SPICE can automatically generate the bitstream used to program the configuration circuits, according to the selected implementations.

C. Physical Structure Modeling

To map logic functions efficiently to circuit modules, VPR uses abstract-level modeling to bridge the technology mapping results and FPGA architecture resources. The VPR architecture description language focuses on describing the structure of circuit modules at a behavioral level rather than a structural level. For instance, an I/O pad is described with two operating modes: an input pad and an output pad, as illustrated in Fig. 6(a). However, with abstract modeling, the physical structure of I/O pads cannot be accurately described, causing difficulties in transistor-level modeling. Comparing to Fig. 6(b), an I/O pad modeled by VPR [in Fig. 6(a)] lacks two critical elements: 1) the SRAM controlling the directionality of the I/O module and 2) two ports, `direction` and `PAD`, of the I/O module. `PAD` is a bidirectional port that interfaces the FPGA to the outside world; `direction` determines whether the signal is propagated from `PAD` to `data_in` or from `data_out` to `PAD`. Hence, in the purpose of accurately modeling FPGAs with SPICE or Verilog netlists, the abstract-level modeling should be improved to exactly describe the physical design.

We extend the architecture description language to model the physical design of an I/O pad, as follows:

```
  <pb_type name="io" idle_mode_name="inpad"
physical_mode_name="io_phy">
      <mode name="io_phy">
        <pb_type name="iopad" num_pb="1"
spice_model_name="iopad"/>
      </mode>
      <mode name="inpad">
        <pb_type name="inpad" num_pb="1"
mode_bits="1"/>
      </mode>
      <mode name="outpad">
        <pb_type name="outpad" num_pb="1"
mode_bits="0"/>
      </mode>
  </pb_type>
```

In parallel to the original abstract-level modeling, an extra mode named by `io_phy` is added to the `pb_type`, under which the physical design of an I/O pad is described by the architecture description language. An XML property `physical_mode_name` is added to the `pb_type`, in order to identify which mode describes the physical design of the module. As a module depends on the configuration bits to switch between the operating modes, each operating mode, e.g., `inpad` and `outpad`, contains a new XML property `mode_bits`, used to define its unique configuration bits. For instance, `mode_bits="1"` under operating mode `inpad` specifies that it is enabled when the SRAM is configured to logic `1`. Note that the new mode `io_phy` is only used by FPGA-SPICE for SPICE and Verilog generation, while the two original modes `inpad` and `outpad` are used in VPR packing, placement, and routing. As such, the extended architecture description language does not influence any results of VPR packing, placement, and routing.

V. Netlist Partitioning Techniques

To provide a high-level of flexibility in the analysis of area and power, FPGA-SPICE can extract SPICE netlists and simulation decks at different levels of complexity (see Section V-A). This enables the user to tradeoff accuracy for simulation time. The netlist splitting is based on the following techniques.

1) A parasitic activity estimation technique, which considers the impact of both used and unused circuit modules at architecture-level in signal activity estimation (see Section V-B).
2) A voltage stimuli and loads extraction technique, which uses the results of parasitic activity estimation in generating voltage stimuli and adds downstream loads to each simulation deck by considering its architectural context (see Section V-C).

Last but not least, to ensure the correctness of simulations, verification strategies are applied and discussed in Section V-D.

A. Simulation Deck Organization

FPGA-SPICE can generate SPICE netlists with three different simulation strategies: full-chip-level, grid-level, and component-level, as illustrated in Fig. 7. The full-chip-level simulation decks include all the components, such as CLBs, SBs, and CBs, in a unique Verilog/SPICE netlist [see Fig. 7(a)], leading to very accurate simulations. Due to the exponential complexity of EDA algorithm [42], [43], full-chip-level SPICE simulations may require long runtimes and large memory usages. To better tradeoff the simulation time, memory usage, and accuracy, FPGA-SPICE can split the evaluation of a full-chip-level simulation deck into grid-level and component-level simulation decks. The grid-level simulation decks, as shown in Fig. 7(b), consider separately each individual CLBs, memory banks, DSP blocks, SB multiplexers, and CB multiplexers. In the component-level simulation decks, the CLBs are further sliced into individual modules, such as LUTs, FFs, and local routing multiplexers, for each of which an associated simulation deck is created, as depicted in Fig. 7(c).

To be accurate, voltage stimuli and loads are added to the grid-level and component-level simulation decks by considering their architectural context and the impact from parasitic nets (see details in Sections V-B and V-C).

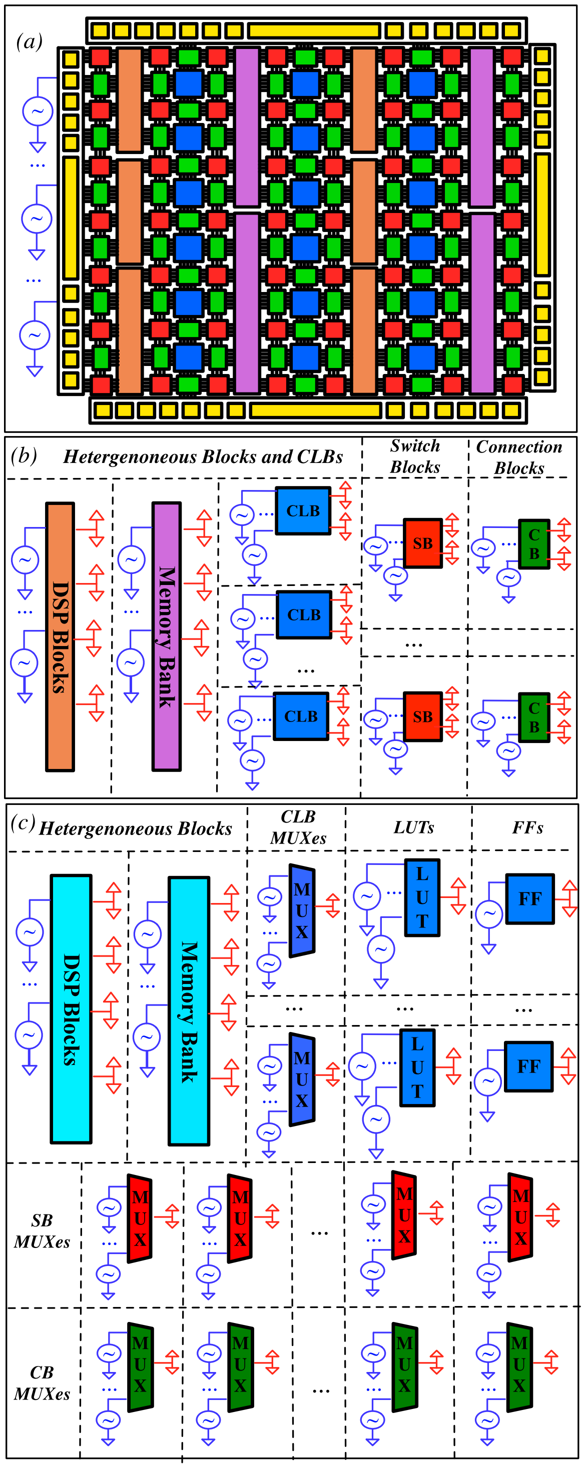

Fig. 7. Illustration of the SPICE netlists partitioning strategies. (a) Full-chip-level, (b) grid-level, and (c) component-level simulation decks.

B. Parasitic Activity Estimation

Input signals in grid-/component-level netlists should accurately model the internal signal activities of FPGA modules. In an FPGA, the signals of the used nets may be parasitically propagated to unused nets, depending on the topology of the routing architecture. ACE2 estimates the signal activities of the used nets but cannot foresee the parasitically propagated activities because they are only predictable after the routing pass finishes. Fig. 8 illustrates the parasitic net signals sourcing from a used net, called net_0. Assume net_0 is only used by the CLB through local routing (green path) and not routed to the global routing architecture. VPR assumes that all the downstream components driven by net_0 are idle and configures them to propagate their first inputs. However, for such condition, net_0 will be propagated through the routing structure (red path). These parasitic activities will cause extra power consumption and should be taken into account.

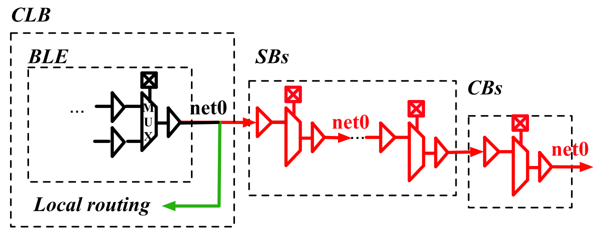

Fig. 8. Example of parasitic nets estimation.

Algorithm 1 Parasitic Activity Estimation Algorithm (Pseudocode)

```
Function
 NodeParasiticActivity(current_node):
    foreach next_node ∈ fanout_nodes do
        if (TRUE ==
         IsSignalPropagated(next_node)) then
            SetParasiticActivity(next_node);
            NodeParasiticActivity(next_node);
        end
    end
end
Function ParasiticActivityEsti():
    while TRUE ==
     NewParasiticNetAssigned() do
        foreach node ∈ global_routing_arch do
            NodeParasiticActivity(node);
        end
        foreach node ∈ local_routing_arch do
            NodeParasiticActivity(node);
        end
    end
end
```

FPGA-SPICE performs parasitic activity estimation for all the unused nets after the routing stage, whose detailed algorithm is shown in Algorithm 1. To accurately capture the signal activity of a full FPGA fabric, the parasitic activity estimation is applied to both global routing architecture and local routing architecture. The algorithm iteratively updates the signal activity of all parasitic nets in the routing architectures until there are no new parasitic nets detected during the last iteration.

C. Voltage Stimuli and Load Extraction

To accurately estimate the power consumption of individual components, FPGA-SPICE generates its individual simulation decks by including voltage stimuli and downstream loads. To illustrate the technique, Fig. 9 shows a BLE multiplexer (in blue) that is driven by signals A and B and that fan-outs to both local and global routing architectures.

First, voltage stimuli are added to model the signal activities of A and B. Their frequencies and pulse widths are derived from the signal switching density and activities, by considering the impacts of parasitic nets as explained in Section V-B. The signal switching density defines the number of switching

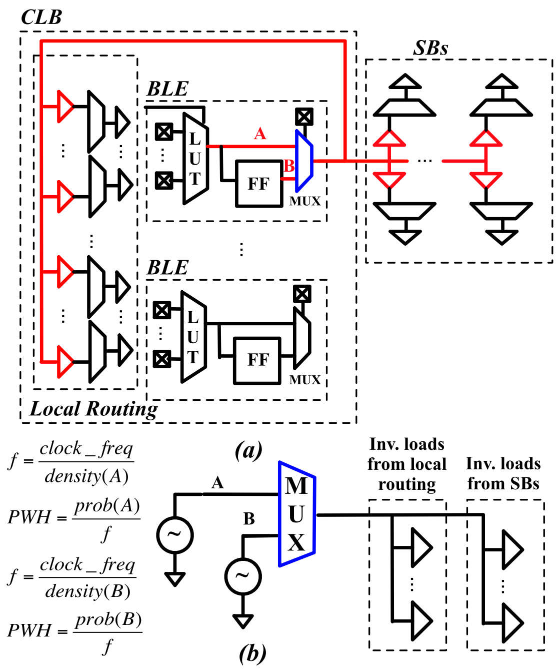
CLB
SBs
BLE
LUT
FF
MUX
A
B
Local Routing
(a)
$f = \frac{clock\_freq}{density(A)}$
$PWH = \frac{prob(A)}{f}$
$f = \frac{clock\_freq}{density(B)}$
$PWH = \frac{prob(B)}{f}$
MUX
Inv. loads from local routing
Inv. loads from SBs
(b)

Fig. 9. Illustration of the voltage stimuli generation and load extraction techniques. (a) BLE multiplexer with its architectural context. (b) Extracted simulation deck.

events of a signal in one clock cycle while the probability represents the proportion of the signal to be in logic 1 during one system clock cycle. To translate this activity information into waveforms, we set the frequency of the voltage stimuli to

$$\text{freq} = \frac{\text{clock\_period}}{\text{density(Signal)}}. \quad (1)$$

The pulsewidth (PWH) of a voltage stimuli is set to

$$\text{PWH} = \frac{\text{probability(Signal)}}{\text{freq}}. \quad (2)$$

Then, FPGA-SPICE loads the block by extracting the downstream elements in the architecture [highlighted in red in Fig. 9(a)]. The downstream loads of a grid/component should be included in the simulation deck for two reasons: 1) these loads are charged/discharged by the element and 2) the power consumption is sensitive to voltage slews, which are highly dependent on the downstream loads [5]. To guarantee a full load extraction, FPGA-SPICE searches the downstream network of a circuit element recursively. The detailed algorithm is shown in Algorithm 2. FPGA-SPICE checks the buffering condition for each fan-out of the grid/component. For buffered fan-outs, the function `AddBufLoadToSimDeck` will be executed, where their input buffers are added to the simulation deck. Regarding unbuffered fan-outs, the function `AddFullLoadToSimDeck` where the full circuit element will be added to the simulation deck is called, because it is charged/discharged by the upstream elements. Afterward, the function `DownstreamLoadExtract` is recursively called to the unbuffered fan-out element until a buffered downstream element is found and included in the simulation deck.

D. Functionality Verification

To examine the functionality of both configuration peripheral and core logic, as illustrated in Fig. 10(a), a top-level Verilog testbench includes two phases.

Algorithm 2 Downstream Loads Extraction Algorithm (Pseudocode)

```
fanout_node: Fanout circuit elements.
Function
 DownstreamLoadExtract(current_node):
    foreach next_node ∈ fanout_node: do
        if (TRUE == IsBuffered(next_node)) then
            AddBufLoadToSimDeck(next_node);
        else
            AddFullLoadToSimDeck(next_node);
            DownstreamLoadExtract(next_node);
        end
    end
end
```

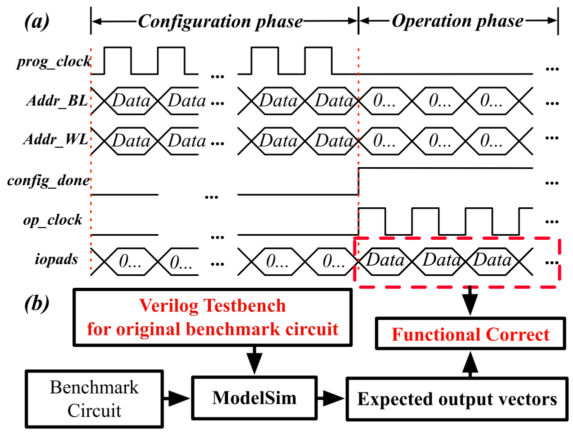

Fig. 10. Illustration of (a) waveforms and (b) EDA flow for functional verification purpose.

1) *Configuration Phase:* Each memory cell, e.g., SRAM, is programed serially according to the bitstream. During each programing cycle, a memory cell is configured individually. During this period, all the I/Os of FPGA stable at logic 0.
2) *Operating phase:* Configuration circuits are powered OFF and testing input patterns are fed to all the I/Os of FPGA. Details of the generation of the testing input patterns are described in Section V-C.

Fig. 10(b) illustrates the proposed verification flow. The top-level Verilog netlists and testbenches are simulated with an HDL simulator, e.g., Modelsim, or an electrical simulation, e.g., HSPICE. The output waveforms are then compared to the simulation results of postlogic-synthesis netlists to ensure they are consistent. As FPGAs contain a large number of memory bits, the electrical simulations for the programing phase are time-consuming. To accelerate the simulation and focus on the core logic, top-level SPICE simulation decks only include the operating phase.

VI. Experimental Results

This section is organized in three parts. In the first part, we introduce the general methodology (see Section VI-A). In the second part, we evaluate the quality of different simulation strategies offered by FPGA-SPICE (see Section VI-B). In the third part, we use FPGA-SPICE to present three case studies:

1) an area breakdown analysis for SRAM-based FPGAs, studying which components are the dominant factors (see Section VI-D);

TABLE I

COMPARISON OF RUNTIME, MEMORY USAGE OF FULL-CHIP/GRID/COMPONENT-LEVEL SIMULATION DECKS

Metrics	**Runtime (minutes)**			**Peak Used Memory (Gb.)**		
Simulation Deck	**Max.**	**Min.**	**Avg.**	**Max.**	**Min.**	**Avg.**
Full-chip-level	570.73	0.62	90.97	224.11	1.35	66.63
Grid-level	65.24	0.39	14.82	0.86	0.86	0.86
Component-level	51.41	0.34	12.18	0.67	0.68	0.67

2) a power breakdown comparison to analytical models, analyzing the source of accuracy loss (see Section VI-D);
3) a robustness evaluation against process corners with a focus on the energy consumption (see Section VI-E).

A. Methodology

We use the FPGA-SPICE EDA flow shown in Fig. 2. Benchmark circuits are first synthesized, packed, placed, and routed by the traditional FPGA EDA flow. Afterward, the FPGA-SPICE generates the full-chip-/grid-/component-level simulation decks with the architecture XML files. With the purpose of area analysis, the semicustom design tool, i.e., Cadence Innovus [34], is invoked to achieve the layouts of full FPGA fabrics. To enable power analysis for large full-chip-level netlists, SPICE simulations are performed with the 64-bit HSPICE in eight-processor multithreading mode [35]. The experiments are run on a 64-bit RedHat Linux server with 28 Intel Xeon Processors and 256-GB memory.

In this paper, all the circuit components and FPGA architectures are optimized by considering a commercial 40-nm technology. The transistor-level circuit designs of SRAMs, FFs, and multiplexers are derived from [10]. We model routing wire segments with a one-level π-type RC models and the wire parameters are derived from International Technology Roadmap for Semiconductors [33]. We consider a CLB architecture with 40-input pins ($I = 40$), consisting of 10 BLEs ($N = 10$), each of which contain a 6-input LUT ($K = 6$) [10]. Similar to commercial FPGAs [23], [24], we consider unidirectional routing architectures [37] with three types of wire lengths. In each routing channel, 30% of routing tracks are built with length-1 wires ($L = 1$), another 30% of routing tracks are built with length-2 wires ($L = 2$), and the rest 40% of routing tracks are built with length-4 wires ($L = 4$). Each routing track can be connected to other three routing tracks from adjacent channels ($Fs = 3$). Each CLB input pin can be connected to 15% of the routing tracks in a channel ($F_{c,\text{in}} = 0.15$), while each CLB output pin can reach 10% of the routing tracks ($F_{c,\text{out}} = 0.10$). The channel width, W, is set 300, being similar to commercial FPGAs [23], [24]. All the architecture description files used in this paper are available in [13].

In both HDL and SPICE simulations of this paper, we determine the simulation clock period by adding an empirical 20% slack to the VPR critical path delay, in order to consider maximum errors between the VPR timing analysis engine and SPICE simulations [9]. The time period of simulations should be a full operating cycle by considering the least active signal, as follows:

$$\text{sim\_time\_period} = \frac{\text{clock\_period}}{\min\{\text{density}(\text{Signal})\}}. \tag{3}$$

However, the density of the least active signal is typically very low, which leads to a long time period and large simulation time. Instead, we replace the min{density(Signal)} with the average density of signals to reduce the simulation time. The time step of the SPICE simulator is set to 0.1 ps and fast simulation algorithm is turned on.

B. Runtime, Memory Usage and Accuracy of Simulation Deck Levels

Performing full-chip-level simulations is the most accurate approach to power analysis, but it comes at the large cost of runtime and memory usage. To run full-chip-level simulations within the capability of our server, we select a set of benchmark circuits from Microelectronics Center of North Carolina [38], International Workshop on Logic & Synthesis'05 [39], and International Symposium on Circuits and Systems'85 [40] benchmarks.

1) Runtime and Memory Usage: As shown in Table I, the full-chip-level simulations have a wide range in runtime (from 0.62 min to 9.52 h with an average of 1.52 h) and memory usage (from 1.35 to 224 Gb with an average of 66.63 Gb). These variations are due to that the FPGA array size, number of clock cycles, and clock frequency applied in the full-chip-level netlists are different from one benchmark circuit to another. Compared to the full-chip-level simulation, the memory usage and runtime of the grid-level and component-level simulations are bounded by the largest component, such as CLB. Therefore, we see zero difference in the maximum, minimum, and average memory usage of these simulation decks. Thanks to the reduced complexity in the simulation decks, the grid-level approaches achieve 6.1× speed-up in runtime and 77.5× reduction in memory usage, while the component-level simulations accelerate 7.5× in runtime and reduce 99.4× in memory usage.

2) Accuracy on Power Estimation: In addition to a reduction in runtime and memory usage, the grid-level and component-level simulation decks can also guarantee the accuracy of power analysis. Fig. 11 compares the total power consumption obtained by the full-chip-level, grid-level and component-level simulation decks. On average, the accuracy loss of grid-level and component-level simulation decks are 9.9% and 8.3% respectively. Each approach has its own strength: the grid-level demonstrates better accuracy in benchmarks, *steppermotordrive*, *pci_conf_cyc_addr_dec*, *apex7*, and *s820*, while the component-level is more accurate in the case of *s298*, *elliptic*, *stereovision3*, and so on. The provided simulation decks allow users to select the most appropriate approaches for their benchmark set.

The observed accuracy loss in Fig. 11 for the grid-level and component-level simulations can be explained as follows. In this paper, for a fair comparison, we use the same activity estimation results in generating the voltage stimuli for grid-level and component-level netlists as well as analytical power models. Due to the errors between the estimated signal activities and the actual conditions [1], we see that the accuracy loss ranges from 3.2% to 16.2%. We believe that the accuracy can be further improved when our approach adopts scalable testing pattern generators [44].

3) Accuracy in Power Breakdown: For power estimation techniques, capturing the power characteristics of FPGA architectures is as important as providing accurate power values. Fig. 12 compares the power breakdown results between full-chip-level, grid-level, and component-level simulation decks, average over the benchmark circuits. For each FPGA module, the error of grid-level and component-level simulation decks

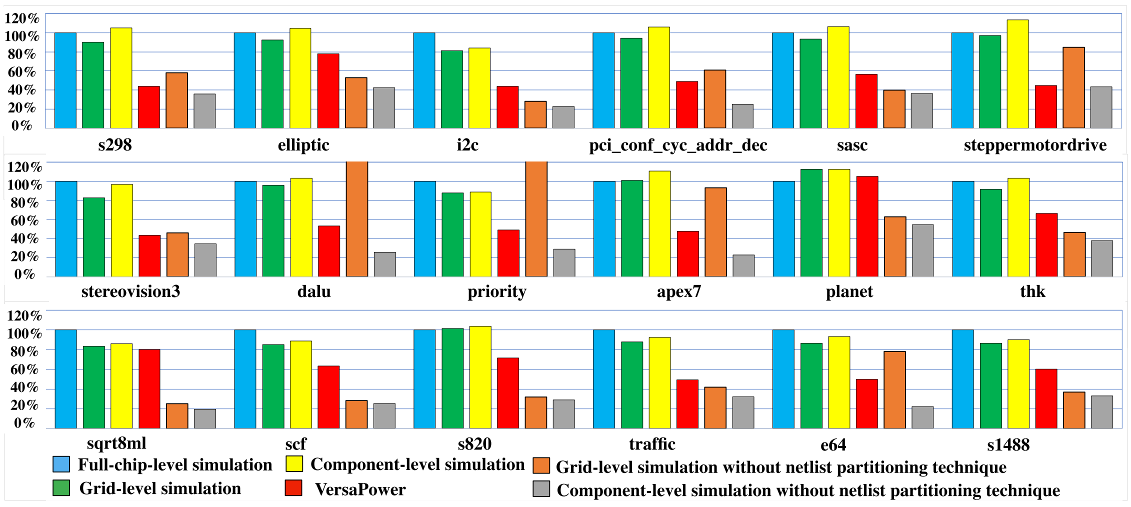

Fig. 11. Normalized power comparison between different levels of simulation decks of FPGA-SPICE and VersaPower.

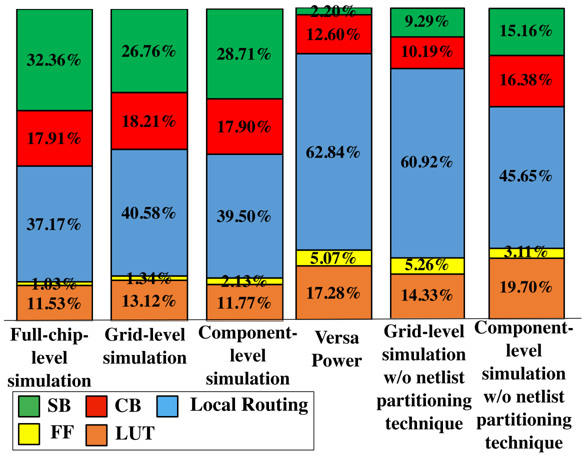

Fig. 12. Power breakdown comparison between different levels of simulation decks of FPGA-SPICE.

is below 7%. This shows that even in some cases, using grid-level and component-level simulation decks may not achieve accurate power consumption but they can at least provide an accurate analysis on power breakdown.

4) Impact of Netlist Partitioning Techniques: We also examine the effects of the netlist partitioning techniques introduced in Section V in terms of accuracy. As shown in Fig. 11, when the load extraction and parasitic activity estimation are disabled, on average, the accuracy of the grid-level and the component-level simulations degrades by 50% and 32%, respectively. In addition, ignoring the load extraction and parasitic activities causes misleading power breakdown results, as depicted in Fig. 12. In particular, the weight of SB power is underestimated as an SB typically drives a high fan-out with large parasitics, such as long metal wires, CBs, and other SBs. Our results show that the netlist partitioning techniques are critical to match the accuracy of the grid-level and component-level simulations with the full-chip level.

C. Case Study 1: Area Breakdown of SRAM-Based FPGAs

In this section, we use the FPGA-SPICE EDA flow shown in Fig. 2(a) to generate a full FPGA layout and study its area breakdown. In order to fit the capability of our Linux server without losing representativity, we consider the FPGA architecture described in Section VI-A but with a reduced CLB array size 5×5 and a channel width of 300. We perform

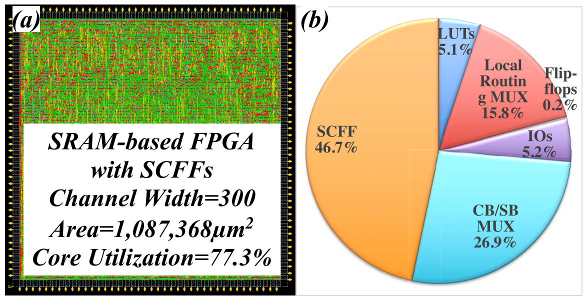

Fig. 13. SRAM-based FPGAs configured by SCFF. (a) Full-chip-level layout. (b) Area breakdown.

semicustom design flows for the scan-chain-based FPGAs as depicted in Fig. 1(f) and (g).

Fig. 13(a) depicts the full layout of SRAM-based FPGA chips configured by SCFFs. Note that the core utilization rate of the FPGAs is 77.3%, indicating that 22.7% of the total area is exclusively devoted to interconnecting wire. This shows that metal wires are a dominant factor in FPGA area. Fig. 13(b) depicts the area breakdown of the SRAM-based FPGA. The SCFFs can occupy 46.7% of the total area, being a major factor in the area. LUTs and FFs stand only up to 5.1% of the total area, while routing multiplexers (15.8%–26.9%) are another major contributor. Actually, the share of routing multiplexers may be even larger if we consider the area of SRAMs associated with the routing multiplexers. Note that the area weight of SRAMs and routing multiplexers would increase when the array size and channel width of the FPGA increase.

Even though our FPGA layout is generated by a semi-custom design flow rather than the conventional full-custom approach [45], it does not diminish the interest in area study. Indeed, our method enables: 1) rapid layout-level area evaluation on the FPGA architectural decisions, including full back-end parasitics, such as the area of interconnecting wires, which are difficult to capture in analytical models and 2) refining analytical models for accurate full-chip area prediction, where our layout samples can advise what fitting parameters are to be used. Not limited to SCFF-based FPGA architectures, FPGA-SPICE can be used to study the area breakdown of FPGAs configured by a variety of other means such as a traditional memory array organization. Beyond an area study, our design flow allows the prototyping of customizable FPGAs

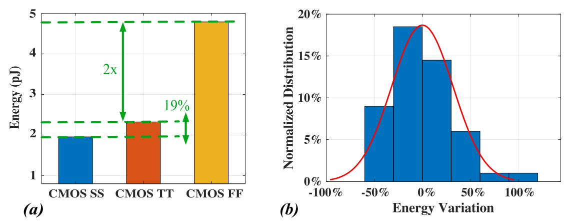

Fig. 14. Case study on benchmark *s298*. Energy consumption of the SRAM-based FPGAs (a) under different process corners and (b) under process variations (Monte Carlo simulation).

to automatic and feasible for those with limited knowledge of FPGA hardware, potentially reducing the product development cycle and cost.

D. Cased Study 2: Analytical Power Model Study

SPICE simulations are considered a golden reference as they show a good consistency with silicon measurements [47]–[50]. As such, SPICE simulation results are widely used in calibrating analytical power models [1]–[9]. In this section, we use FPGA-SPICE to evaluate the performance of the analytical power model VersaPower [8], which is the power estimation engine of most popular academic FPGA architecture evaluation tool suite VTR [2]–[4]. We employ the FPGA-SPICE EDA flow shown in Fig. 2(b) and consider the same experimental settings explained in Section VI-A.

In Figs. 11 and 12, we compare the power results between FPGA-SPICE and VersaPower. In terms of power consumption, VersaPower underestimates by 43% on average when compared to the full-chip levels simulation. Note that considering the benchmark "planet" in Fig. 11 which has few parasitic net activities and a high utilization rate of routing multiplexers, VersaPower can produce similar results as FPGA-SPICE. In terms of power breakdown, FPGA-SPICE predicts that the local routing architecture has a power share as large as the global routing architecture, which is different from VersaPower. It can be explained by the following reasons: 1) FPGA-SPICE takes the parasitic net activities into account which leads to additional power consumption in routing architectures; VersaPower assumes that unused resources in FPGAs can be regionally powered-off and, therefore, parasitic net activities can be neglected and 2) FPGA-SPICE uses electrical simulations and real configuration information from VTR, i.e., SRAM configurations in LUTs, used and unused routing multiplexer configurations, to accurately analyze the power of the architectures, while VersaPower only considers worst case scenario and basic scaling strategies [8].

Note that SPICE results may underestimate the actual power consumption of FPGA devices, due to process variations and missing physical aspects, such as parasitics [47]. However, FPGA-SPICE remains an accurate power estimation method at the presilicon stage. It can accurately capture the power difference from one technology node to another, one FPGA architecture to another, from one FPGA implementation to another, and from one LUT/multiplexer to another. When the accuracy of analytical models is unknown in these scenarios, they do require SPICE validations. In addition, FPGA-SPICE can guarantee relative accuracy in capturing the different power distribution in FPGA architectures, being useful in validating the effectiveness of novel circuit designs and EDA algorithms, e.g., placement and routing algorithms, and FPGA architectural enhancements with regards to a baseline.

E. Case Study 3: Impact of Process Corners and Variations Energy Consumption

As shown in Section IV-A1, FPGA-SPICE supports different process corners in the architecture description language, enabling the study on power variation of FPGAs under process variation. In addition, FPGA-SPICE supports Monte Carlo simulations for full FPGA fabrics which enable us to study the impact of die-to-die and device-to-device variations on the energy consumption.

In this case study, we consider three process corners, namely, fast–fast (FF), typical–typical (TT), and slow–slow (SS). To be illustrative, we also performed a 100-run Monte Carlo simulation by considering the process variations on both transistors and R/C of interconnection wires [46]. Note that the process corners and variations are provided natively in the considered commercial 40-nm technology. We employ the FPGA-SPICE EDA flow shown in Fig. 2(b) and study the impact of process corners and variations on energy consumption. To provide maximum accuracy, we use the full-chip-level simulation deck and consider a representative benchmark *s298*. Fig. 14(a) compares the energy consumption of the SRAM-based FPGA under the three process corners. When compared to the TT baseline, using the FF corner results in a 2.1× increase in energy, while using the SS corner leads to a 19% reduction. Fig. 14(b) shows that the process variations could lead to a $< 50\%$ shift on the energy consumption in most cases (90%) with a maximum of 2× in the worst cases.

VII. Conclusion

This paper introduces a simulation-based architecture evaluation framework for FPGAs, called FPGA-SPICE. This tool extends the VTR architecture description language to include transistor-level modeling parameters of FPGA components. Tightly embedded in academic architecture exploration tool suites, FPGA-SPICE can automatically generate Verilog and SPICE netlists by considering FPGA configurations. To support flexible circuit designs and FPGA architectures, a high-level XML-based FPGA architectural description language is introduced. The generated Verilog netlists are exploited to achieve layouts of full FPGA fabrics through a semicustom design flow. The SPICE simulation decks can be generated at three levels of complexity, namely, full-chip-level, grid-level, and component-level, providing different tradeoffs between accuracy and simulation time. To guarantee accuracy for grid-level and component-level simulation decks, netlist partitioning techniques, such as parasitic activity estimation, are developed. Electrical simulations showed that averaged over the selected benchmarks, the grid-/component-level approach can achieve 6.1×/7.5× execution speed-up with 9.9%/8.3% accuracy loss, respectively, compared to the full-chip level. We presented three different case studies using FPGA-SPICE: 1) area breakdown analysis for SRAM-based FPGAs, showing that configuration memories are a dominant factor; 2) power breakdown comparison to analytical models, analyzing the source of accuracy loss; and 3) robustness evaluation against process corners, studying their impact on energy consumption of full FPGAs.

VIII. Discussion

FPGA-SPICE is more computing intensive than analytical model solutions. However, FPGA-SPICE can save the manual efforts and reduce the expertise required in developing analytical models for different technologies, thanks to its general-purpose simulation-based approach. More than the showcased examples in Section VI, FPGA-SPICE can go beyond the capability of current FPGA architecture evaluation tools by: 1) providing a baseline for examining the accuracy of analytical models; 2) prototyping FPGAs and verifying functionality [14], [15], [19]; 3) validating the effectiveness of EDA algorithms and novel FPGA architecture with post-P&R analysis [18], [19]; and 4) identifying physical design challenges in FPGAs, such as analyzing hotspot management [16], [17], and so on. The accuracy and runtime tradeoff of FPGA-SPICE can be further mitigated by exploiting hardware acceleration platforms such as multithreading, GPU, and FPGA [51]–[54]. As shown in Fig. 7, each grid-level/component-level netlist is fully independent of others, and multithreading parallelism can be easily applied. In particular, SPICE simulations can benefit $> 32\times$ runtime reduction from GPU acceleration, allowing a dynamic tradeoff between the accuracy and double-precision/single-precision floating-point formats [52].

Acknowledgment

The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory and Defense Advanced Research Projects Agency or the U.S. Government.

References

[1] J. Lamoureux and S. J. E. Wilton, "Activity estimation for field-programmable gate arrays," in *Proc. Int. Conf. Field Program. Log. Appl. (FPL)*, Aug. 2006, pp. 87–94.

[2] J. Rose *et al.*, "The VTR project: Architecture and CAD for FPGAs from verilog to routing," in *Proc. ACM/SIGDA Int. Symp. Field Program. Gate Arrays (FPGA)*, Monterey, CA, USA, 2012, pp. 77–86.

[3] J. Luu, J. H. Anderson, and J. S. Rose, "Architecture description and packing for logic blocks with hierarchy, modes and complex interconnect," in *Proc. 19th ACM/SIGDA Int. Symp. Field Program. Gate Arrays (FPGA)*, Monterey, CA, USA, 2011, pp. 227–236.

[4] J. Luu *et al.*, "VPR 5.0: FPGA cad and architecture exploration tools with single-driver routing, heterogeneity and process scaling," in *Proc. ACM/SIGDA Int. Symp. Field Program. Gate Arrays (FPGA)*, Monterey, CA, USA, 2009, pp. 133–142.

[5] K. K. W. Poon, S. J. E. Wilton, and A. Yan, "A detailed power model for field-programmable gate arrays," *Trans. Des. Automat. Electron. Syst.*, vol. 10, no. 2, pp. 279–302, 2005.

[6] F. Li, Y. Lin, L. He, D. Chen, and J. Cong, "Power modeling and characteristics of field programmable gate arrays," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 24, no. 11, pp. 1712–1724, Nov. 2005.

[7] H. Liang, Y. Chen, T. Luo, W. Zhang, H. Li, and B. He, "Hierarchical library based power estimator for versatile FPGAs," in *Proc. IEEE 9th Int. Symp. Embedded Multicore/Many-Core Syst.-Chip*, Turin, Italy, Sep. 2015, pp. 25–32.

[8] J. B. Goeders and S. J. E. Wilton, "VersaPower: Power estimation for diverse FPGA architectures," in *Proc. IEEE Int. Conf. Field Program. Technol. (ICFPT)*, Dec. 2012, pp. 229–234.

[9] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAS*. Norwell, MA, USA: Kluwer, 1998.

[10] C. Chiasson and V. Betz, "Should FPGAs abandon the pass-gate?" in *Proc. 23rd Int. Conf. Field Program. Log. Appl.*, Porto, Portugal, Sep. 2013, pp. 1–8.

[11] F. F. Khan and A. Ye, "An evaluation on the accuracy of the minimum width transistor area models in ranking the layout area of FPGA architectures," in *Proc. 26th Int. Conf. Field Program. Log. Appl. (FPL)*, Lausanne, Switzerland, Aug./Sep. 2016, pp. 1–11.

[12] X. Tang, P.-E. Gaillardon, and G. De Micheli, "FPGA-SPICE: A simulation-based power estimation framework for FPGAs," in *Proc. 33rd IEEE Int. Conf. Comput. Design (ICCD)*, New York, NY, USA, Oct. 2015, pp. 696–703.

[13] *The OpenFPGA Project*. Accessed: 2018. [Online]. Available: https://github.com/LNIS-Projects/OpenFPGA

[14] I. Kuon, A. Egier, and J. Rose, "Design, layout and verification of an FPGA using automated tools," in *Proc. ACM/SIGDA 13th Int. Symp. Field-Program. Gate Arrays (FPGA)*, Monterey, CA, USA, 2005, pp. 215–226.

[15] J. H. Kim and J. H. Anderson, "Synthesizable standard cell FPGA fabrics targetable by the verilog-to-routing CAD flow," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 10, no. 2, Apr. 2017, Art. no. 11.

[16] G. Deshpande, "Thermal management techniques for field programmable gate arrays," Ph.D. dissertation, Dept. Elect. Eng., Univ. Texas Dallas, Richardson, TX, USA, 2017.

[17] A. N. Nowroz and S. Reda, "Thermal and power characterization of field-programmable gate arrays," in *Proc. 19th ACM/SIGDA Int. Symp. Field Program. Gate Arrays (FPGA)*, Monterey, CA, USA, 2011, pp. 111–114.

[18] X. Tang, P.-E. Gaillardon, and G. De Micheli, "Accurate power analysis for near-Vt RRAM-based FPGA," in *Proc. 25th Int. Conf. Field Program. Log. Appl. (FPL)*, London, U.K., Sep. 2015, pp. 1–4.

[19] X. Tang, E. Giacomin, G. De Micheli, and P.-E. Gaillardon, "Post-P&R performance and power analysis for RRAM-based FPGAs," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 8, no. 3, pp. 639–650, Sep. 2018.

[20] D. Lewis *et al.*, "The Stratix II logic and routing architecture," in *Proc. ACM/SIGDA 13th Int. Symp. Field-Program. Gate Arrays (FPGA)*, Monterey, CA, USA, 2005, pp. 14–20.

[21] Altera Corporation. (Jul. 2008). *Stratix IV Device Handbook Version SIV5V1-1.1*. [Online]. Available: http://www.altera.com/literature/hb/stratix-iv/stratix4_handbook.pdf

[22] Xilinx. (Mar. 2008). *Virtex-5 User Guide UG190 (V4.0)*. [Online]. Available: http://www.xilinx.com/support/documentation/user_guides/ug190.pdf

[23] Altera Corporation. (Jul. 2015). *Stratix 10 Advance Information Brief*. [Online]. Available: https://www.altera.com/en_US/pdfs/literature/hb/stratix-10/S10_TX_FPGA_AIB.pdf

[24] Xilinx Corporation. (May 2015). *Virtex-7 User Guide DS180 (V1.17)*. [Online]. Available: https://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf

[25] M. Hutton *et al.*, "Improving FPGA performance and area using an adaptive logic module," in *Proc. Int. Conf. Field Program. Log. Appl. (FPL)*, 2004, pp. 135–144.

[26] J. Luu *et al.*, "On hard adders and carry chains in FPGAs," in *Proc. IEEE 22nd Annu. Int. Symp. Field-Program. Custom Comput. Mach. (FCCM)*, May 2014, pp. 52–59.

[27] G. Lemieuxm, P. Leventis, and D. Lewis, "Generating highly-routable sparse crossbars for PLDs," in *Proc. ACM/SIGDA 8th Int. Symp. Field Program. Gate Arrays (FPGA)*, Monterey, CA, USA, 2000, pp. 155–164.

[28] G. Lemieux and D. Lewis, "Using sparse crossbars within LUT," in *Proc. ACM/SIGDA 9th Int. Symp. Field Program. Gate Arrays (FPGA)*, Monterey, CA, USA, 2001, pp. 59–68.

[29] J. Greene *et al.*, "A 65 nm flash-based FPGA fabric optimized for low cost and power," in *Proc. 19th ACM/SIGDA Int. Symp. Field Program. Gate Arrays (FPGA)*, New York, NY, USA, 2001, pp. 87–96.

[30] University of California in Berkeley. *ABC: A System for Sequential Synthesis and Verification*. [Online]. Available: http://www.eecs.berkeley.edu/~alanmi/abc/

[31] E. Lee, G. Lemieux, and S. Mirabbasi, "Interconnect driver design for long wires in field-programmable gate arrays," in *Proc. IEEE Int. Conf. Field Program. Technol.*, Bangkok, Thailand, Dec. 2006, pp. 89–96.

[32] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2002.

[33] *Interconnect Chapter*, ITRS, 2011. [Online]. Available: http://www.itrs2.net/

[34] Cadence Design Systems Inc. (2016). *Innovus Implementation System*. [Online]. Available: https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/innovus-implementation-system-ds.pdf

[35] Synoposys Inc. (2017) *HSPICE: The Gold Standard for Accurate Circuit Simulation*. [Online]. Available: https://www.synopsys.com/content/dam/synopsys/verification/datasheets/hspice-ds.pdf

[36] Mentor Graphics. (2017). *ModelSim ASIC and FPGA Design*. [Online]. Available: https://www.mentor.com/products/fv/modelsim/

[37] G. Lemieux, E. Lee, M. Tom, and A. Yu, "Directional and single-driver wires in FPGA interconnect," in *Proc. IEEE ICFPT*, Dec. 2004, pp. 41–48.

[38] S. Yang, *Logic Synthesis and Optimization Benchmarks User Guide: Version 3.0*. Research Triangle Park, NC, USA: MCNC, Jan. 1991.

[39] C. Albrecht. (Jun. 2005). *IWLS Benchmarks 2005*. [Online]. Available: http://iwls.org/iwls2005/benchmarks.html

[40] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 1989, pp. 1929–1934.

[41] M. Lin, A. El Gamal, Y.-C. Lu, and S. Wong, "Performance benefits of monolithically stacked 3-D FPGA," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 2, pp. 216–229, Feb. 2007.

[42] F. N. Najm, "A survey of power estimation techniques in VLSI circuits," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 2, no. 4, pp. 446–455, Dec. 1994.

[43] A. Vladimirescu, *The SPICE Book*. Hoboken, NJ, USA: Wiley, 2012.

[44] D. B. Y. Yiunn, A. K. B. A'ain, Khor, and J. Ghee, "Scalable test pattern generation (STPG)," in *Proc. IEEE Symp. Ind. Electron. Appl. (ISIEA)*, Penang, Malaysia, Oct. 2010, pp. 433–435.

[45] P. Chow, S. O. Seo, J. Rose, K. Chung, G. Paez-Monzon, and I. Rahardja, "The design of a SRAM-based field-programmable gate array—Part II: Circuit design and layout," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 7, no. 3, pp. 321–330, Sep. 1999.

[46] K. Agarwal, D. Sylvester, D. Blaauw, F. Liu, S. Nassif, and S. Vrudhula, "Variational delay metrics for interconnect timing analysis," in *Proc. 41st Design Automat. Conf.*, San Diego, CA, USA, Jul. 2004, pp. 381–384.

[47] V. Degalahal and T. Tuan, "Methodology for high level estimation of FPGA power consumption," in *Proc. Asia South Pacific Design Automat. Conf. (ASP-DAC)*, Jan. 2005, pp. 657–660.

[48] F. Silveira, D. Flandre, and P. G. A. Jespers, "A g_m/I_D based methodology for the design of CMOS analog circuits and its application to the synthesis of a silicon-on-insulator micropower OTA," *IEEE J. Solid-State Circuits*, vol. 31, no. 9, pp. 1314–1319, Sep. 1996.

[49] B. Lee *et al.*, "A CPU on a glass substrate using CG-silicon TFTs," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2003, pp. 164–165.

[50] R. Wilson *et al.*, "A 460 MHz at 397mV, 2.6 GHz at 1.3 V, 32b VLIW DSP, embedding F_{MAX} tracking," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2014, pp. 452–453.

[51] A. M. Bayoumi and Y. Y. Hanafy, "Massive parallelization of SPICE device model evaluation on GPU-based SIMD architectures," in *Proc. 1st Int. Forum Next-Gener. Multicore/Manycore Technol. (IFMT)*, New York, NY, USA, 2008, Art. no. 12.

[52] K. Gulati, J. F. Croix, S. P. Khatri, and R. Shastry, "Fast circuit simulation on graphics processing units," in *Proc. Asia South Pacific Design Automat. Conf.*, Yokohama, Japan, Jan. 2009, pp. 403–408.

[53] N. Kapre and A. DeHon, "SPICE^2: Spatial processors interconnected for concurrent execution for accelerating the SPICE circuit simulator using an FPGA," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 1, pp. 9–22, Jan. 2012.

[54] L. Han and Z. Feng, "TinySPICE Plus: Scaling up statistical SPICE simulations on GPU leveraging shared-memory based sparse matrix solution techniques," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Austin, TX, USA, Nov. 2016, pp. 1–6.

Xifan Tang (S'13–M'17) received the B.Sc. degree in microelectronics from Fudan University, Shanghai, China, in 2011, and the M.Sc. degree in electrical engineering and the Ph.D. degree in computer science from the École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, in 2013 and 2017, respectively.

He is currently a Post-Doctoral Researcher at the University of Utah, Salt Lake City, UT, USA. His current research interests include computer-aided design for programmable architecture and emerging technologies.

Dr. Tang was a recipient of the 2015 Chinese Government Award for Outstanding Self-Financed Students Abroad.

Edouard Giacomin (S'17) received the M.Sc. degree in electrical and computer engineering from the École Supérieure de Chimie Physique Électronique de Lyon, Villeurbanne, France, in 2016. He is currently working toward the Ph.D. degree in electrical engineering at the University of Utah, Salt Lake City, UT, USA.

In 2018, he was a Visiting Research Intern at IMEC, Leuven, Belgium. His current research interests include emerging technologies such as resistive RAM, 3-D nanofabric architectures, and VLSI design.

Giovanni De Micheli (F'94) received the Dr.Eng. degree in nuclear engineering from the Polytechnic University of Milan, Milan, Italy, in 1979, and the M.S. and Ph.D. degrees in electrical engineering and computer science from the University of California at Berkeley, Berkeley, CA, USA, in 1980 and 1983, respectively.

He was a Professor of Electrical Engineering at Stanford University, Stanford, CA, USA. He is currently a Professor and the Director of the Institute of Electrical Engineering and the Integrated Systems Center, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland. He is also a Program Leader of the Nano-Tera.ch Switzerland. He has authored the book *Synthesis and Optimization of Digital Circuits* (McGraw-Hill, 1994), coauthor and/or coeditor of 8 other books and over 750 technical articles. His citation h-index is 93 (source: Google Scholar). His current research interests include several aspects of design technologies for integrated circuits and systems, such as synthesis for emerging technologies, networks on chips, and 3-D integration, heterogeneous platform design including electrical components and biosensors, and data processing of biomedical information.

Dr. De Micheli is a Fellow of ACM, a member of the Academia Europaea, an International Honorary Member of the American Academy of Arts and Sciences, and a member of the Scientific Advisory Board of IMEC (Leuven, B), CfAED (Dresden, D), and STMicroelectronics. He has been the Division 1 Director from 2008 to 2009, a Co-Founder and the President Elect of the IEEE Council on EDA from 2005 to 2007, the President of the IEEE CAS Society in 2003, and the Chair of several conferences, including Memocode in 2014, DATE in 2010, pHealth in 2006, VLSI SOC in 2006, DAC in 2000, and ICCD in 1989. He was a recipient of the 2016 IEEE/CS Harry Goode Award for seminal contributions to design and design tools of Networks on Chips, the 2016 EDAA Lifetime Achievement Award, the 2012 IEEE/CAS Mac Van Valkenburg Award for contributions to theory, practice, and experimentation in design methods and tools, the 2003 IEEE Emanuel Piore Award for contributions to computer-aided synthesis of digital systems, the Golden Jubilee Medal for outstanding contributions to the IEEE CAS Society in 2000, the D. Pederson Award for the best paper on the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS (TCAD/ICAS) in 1987, and several best paper awards, including DAC in 1983 and 1993, DATE in 2005, Nanoarch in 2010 and 2012, and Mobihealth in 2016. He served as the Editor-in-Chief for the IEEE TCAD/ICAS from 1997 to 2001.

Pierre-Emmanuel Gaillardon (S'10–M'11–SM'16) received the Degree in electrical engineering from the École Supérieure de Chimie Physique Électronique de Lyon, Villeurbanne, France, in 2008, the M.Sc. degree in electrical engineering from the Institut National des Sciences Appliquées de Lyon, Villeurbanne, in 2008, and the Ph.D. degree in electrical engineering from CEA-LETI, Grenoble, France and the University of Lyon, Lyon, France, in 2011.

He was a Research Associate at the Laboratory of Integrated Systems, Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland. He was a Visiting Research Associate at Stanford University, Palo Alto, CA, USA. He was a Research Assistant at CEA-LETI, Grenoble, France. He is currently an Assistant Professor at the Electrical and Computer Engineering Department and an Adjunct Assistant Professor at the School of Computing, University of Utah, Salt Lake City, UT, USA, where he leads the Laboratory for NanoIntegrated Systems. His current research interests include the development of novel computing systems exploiting emerging device technologies and novel EDA techniques.

Dr. Gaillardon was a recipient of the C-Innov 2011 Best Thesis Award, the Nanoarch 2012 Best Paper Award, the BSF 2017 Prof. Pazy Memorial Research Award, and the 2017 NSF CAREER Award. He serves as an Associate Editor for the IEEE TRANSACTIONS ON NANOTECHNOLOGY. He served as a TPC Member for many conferences, including DATE'15–19, DAC'16–18, Nanoarch'12–17. He served as Topic Co-Chair Emerging Technologies for Future Memories for DATE'17–19. He is currently a reviewer for several journals and funding agencies