FPGA-SPICE: A Simulation-based Power Estimation Framework for FPGAs

Xifan Tang, Pierre-Emmanuel Gaillardon and Giovanni De Micheli
Integrated Systems Laboratory (LSI), École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Vaud, Switzerland
Email: xifan.tang@epfl.ch

***Abstract*—Mainstream *Field Programmable Gate Array* (FPGA) power estimation tools are based on probabilistic activity estimation and analytical power models. The power consumption of the programmable resources of FPGAs is highly sensitive to their configurations. Due to their highly flexible nature, the configurations of FPGAs routing multiplexers or *Look Up Tables* (LUTs) are really different from a design to another but current analytical power models cannot accurately capture the associated power differences. In this paper, we introduce a simulation-based power estimation framework for FPGAs, called *FPGA-SPICE*, which supports any FPGA architecture that can be described with an architectural description language. Our power estimation engine automatically generates accurate SPICE netlists according to the FPGA configurations and enables precise power analysis of FPGA architectures. SPICE testbenches can be generated at different level of complexity, denoted as full-chip-level, grid-level and component-level testbenches. Full-chip-level testbenches dump the netlists associated with the complete FPGA fabric. To reduce simulation time, FPGA-SPICE can split the full-chip-level testbenches into grid-level testbenches, each of which consisting of a complete logic block netlist, or component-level testbenches, which consider individual circuit elements, i.e., multiplexers, LUTs, flip-flops, *etc.*, separately. We show that the grid/component-level approach can achieve 14× speed-up with a moderate 14% accuracy loss, compared to the full-chip level. We also use FPGA-SPICE to study the power characteristics of a commercial FPGA architecture at different technology nodes. Experimental results show that the global routing architecture consumes 50% of the total power, the local routing architecture claims for 40% of the total power, and the remaining 10% comes from the LUTs and flip-flops.**

I. Introduction

Very Large Scale Integration (VLSI) power estimation techniques can be classified into two categories: simulation-based and probability-based [1]. On the one hand, simulation-based methods are the most direct way to do accurate power analysis. They typically rely on SPICE-based simulators to analyze the power consumption of a given circuit netlist. However, in the 1990s, SPICE simulations were regarded to be only applicable for small-scale circuits due to the low simulation speed and high memory usage [1]. On the other hand, probability-based methods are based on signal activity estimation and analytical power models. Analytical power models estimate the switching power associated with input signal changes. Average power consumption is calculated by combining signal switch density and switching power. Compared to a simulation-based method, a probability-based method is faster but trades off accuracy due to the approximate errors in analytical power models and signal activity estimations.

In the specific context of *Field Programmable Gate Arrays* (FPGAs), the power estimation engines embedded in academic architecture exploration tools are typically based on probabilistic activity estimation [2] and analytical power models [3]–[5]. A probabilistic power estimation tool faces two challenges. First, the accuracy of these analytical power models is guaranteed to only few input signal patterns of the different circuit elements. Unfortunately, the input signal patterns of FPGAs may significantly differ from a design to another. For instance, the power differences of a 4-input *Look-Up Table* (LUT) can reach 69% under diverse input signal patterns [4]. Therefore, current power estimation tools guarantee accuracy on very restrictive conditions. Second, academic FPGA architecture exploration tools [7] employ architecture description language [8] to model highly flexible FPGA architectures. The hierarchy and complex interconnects inside modern FPGA logic block architectures can be precisely described with the architecture description language. The timing parameters of logic and routing elements are richly provided for accurate timing analysis. However, there are very limited transistor-level modeling parameters in architecture description language, that can be exploited for power estimations. These two challenges cause that current power estimation tools relying on analytical power models cannot capture well the power characteristics of a wide range of novel FPGA architectures. Fortunately, during the last decade, we saw significant advances in SPICE simulators and computing capabilities. Nowadays, SPICE simulators can handle millions of transistors in a few minutes [9], motivating us to explore simulation-based approaches to FPGA.

In this paper, we introduce FPGA-SPICE, a simulation-based power estimation framework for FPGAs, that is tightly integrated within the popular academic architecture exploration tool suite VTR [7]. We extend the generic architecture description language [8] to consider transistor-level parameters related to each module inside the FPGA architecture under evaluation. FPGA-SPICE can output SPICE testbenches at different level of complexity, namely full-chip-level, grid-level and component-level. The full-chip level dumps a netlist of a complete FPGA fabric, including mapping, placement and routing results as well as technological information. To reduce simulation time, FPGA-SPICE employs netlist splitting strategies to slice full-chip-level testbenches into grid-level testbenches, each of which consisting of a complete logic block netlist, or component-level testbenches, which consider individual circuit elements, such as LUTs, *flip-flops* (FFs) and multiplexers. We use FPGA-SPICE to study the power characteristics of a commercial FPGA architecture at 22nm, 45nm and 180nm technology nodes. Experimental results

978-1-4673-7166-7/15/$31.00 ©2015 IEEE

show that the grid-level and component-level testbenches can achieve 14× simulation time reduction with only 14% error accuracy on average, when compared to the most accurate power estimation approach, i.e., the full-chip-level testbench. FPGA-SPICE verifies the conclusion given in literatures [5], that only 10% of the total power comes from LUTs and FFs and the rest 90% is consumed by the routing architectures. FPGA-SPICE reports that the global routing architecture and local routing architecture consume 50% and 40% of the total power, respectively. Leveraging on SPICE simulations, FPGA-SPICE can produce more accurate power analysis than analytical power models, bringing new opportunities in architecture explorations.

This paper is organized as follows. In Section II, we report some brief background on FPGA architectures and academic architecture exploration tools. In Section III, we introduce the core engine of FPGA-SPICE. In Section IV, we report experimental results, while in Section V we conclude the paper.

II. Background

In this section, we first introduce the principles of modern FPGA architectures. Then, we comment on 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 by rich programmable routing resources. Fig. 1 describes the principles of modern FPGA architectures where *Configurable Logic Blocks* (CLBs), organized in array, are surrounded by a global routing architecture. The global routing architecture is built with *Connection Blocks* (CBs), which connect CLB pins to routing tracks, and *Switch Boxes* (SBs), that interconnects routing tracks. A CLB typically comprises a number of *Basic Logic Elements* (BLEs) each of which consists of a *Look Up Table* (LUT), a D *Flip-flop* (FF), and an output selector (2:1 Multiplexer). A local routing architecture consists of fully populated crossbars that interconnects BLE pins and CLB pins. Nowadays, commercial FPGAs [10]–[12] apply architectural enhancements to improve the area and speed of arithmetic-intensive implementations. Columns of CLBs are replaced with heterogenous blocks, such as memory banks and DSP blocks. BLEs comprise of fracturable LUTs [13] and hard carry chains [14]. Local routing architecture also interconnects adjacent CLBs to provide highways between neighbours [15], [16]. High speed transceivers and hard wired logic blocks are used to implement ultra-high speed I/Os in addition to the standard I/O blocks.

B. Academic FPGA Architecture Exploration Tools

The performance of an FPGA strongly depends on hardware/software co-optimizations. In order to study various possible architecture, academic architecture exploration tool suites are developed. In this paper, we focus on the VTR tool suite, whose flow is illustrated in Fig. 2. The Logic synthesis tool ABC [18] optimizes the benchmark circuits and performs a technology mapping. The activity estimator ACE2 [2] computes the signal activities of all the internal nodes in the benchmark circuits. Finally, the tool VPR [7] packs, places

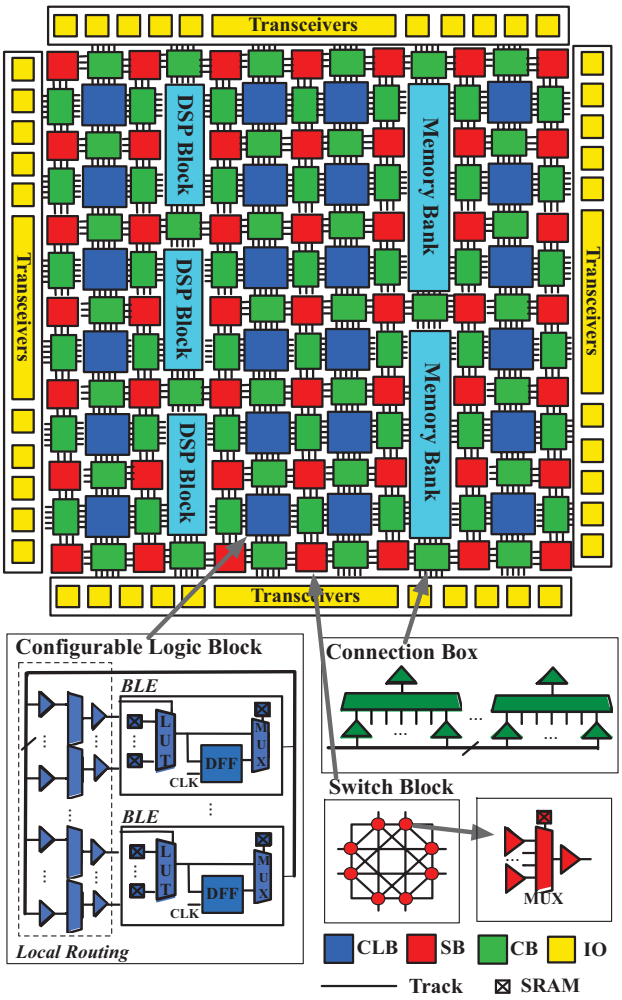

Fig. 1. Modern FPGA architecture

and routes the circuits into a hypothetic FPGA defined with the architecture description language. In the packing stage, LUTs and FFs are clustered into CLBs. Placement determines the physical positions of CLBs in the FPGA fabric. Routing stage maps the nets of CLBs into routing architectures. After routing, VersaPower [5] estimates the power consumption with signal activities and analytical power models. Previous works [3], [4] were designed exclusively for early VPR versions, whose focus were limited to restricted FPGA architectures. VersaPower [5], integrated in the latest VPR, supports the architecture description language and diverse architectures. In this paper, we will compare on our power estimation framework to the analytical power model used in VersaPower.

III. FPGA-SPICE Engine

FPGA-SPICE aims at interfacing a SPICE-based electrical simulator with the VTR tool suite in order to perform accurate power analysis. As illustrated in Fig. 3, FPGA-SPICE exploits the description of the architecture provided by the architect to VTR, the mapped netlists and the estimated signal activities to dump circuit netlists and the associated testbenches for the implemented benchmarks. The tool subsequently invokes a

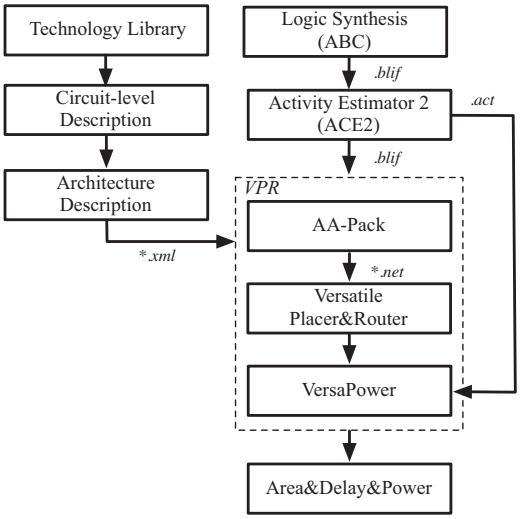

Fig. 2. Academic FPGA architecture exploration EDA flow.

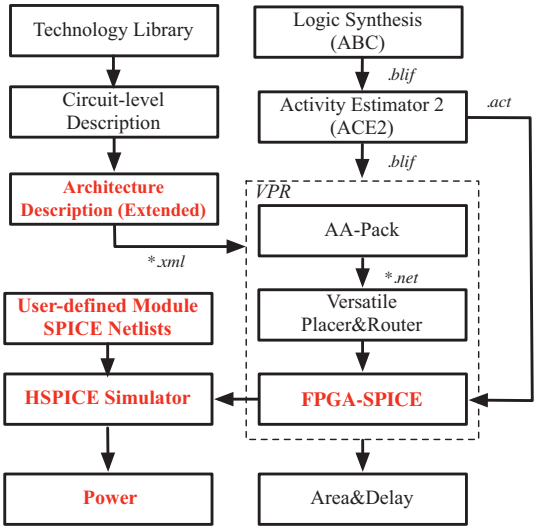

Fig. 3. FPGA-SPICE EDA flow.

SPICE simulator to conduct power analysis.

FPGA-SPICE reads transistor-level design parameters from an extended architecture description XML file and use them to automatically generate detailed SPICE netlists of the circuit elements of the FPGA architecture. Section III-A is devoted to introducing the proposed extension of the VTR architecture description language.

Alternatively, FPGA-SPICE can use a user-defined SPICE netlists rather than automatically generating them. 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 brings the capability to study the system-level impact of the circuit elementary blocks, thereby enabling interesting circuit/architecture co-optimization opportunities. Details about transistor-level SPICE netlists generation are introduced in Section III-B.

FPGA-SPICE can generate its netlists at three levels of complexity, which are full-chip-level, grid-level and component-level. Fig. 4 illustrates the granularity of each level. In a full-chip-level testbench, all the components, such as CLBs, SBs and CBs, are simulated within a unique SPICE netlist, leading to very accurate simulation. Nevertheless, a full-chip-level testbench simulation may require long runtime and large memory usage because of the exponential complexity of *Electrical Design Automation (EDA)* algorithms. To reduce both runtime and memory usage, FPGA-SPICE can split the evaluation of a full-chip-level testbench into grid-level and component-level testbenches. The grid-level testbenches consider separately each individual CLBs, memory banks, DSP blocks, SB multiplexers and CB multiplexers. In the component-level testbenches, the CLBs are further sliced into individual modules, such as LUTs, FFs and local routing multiplexers, for each of which an associated testbench is created. Section III-D focus on the splitting strategies in grid/component-level testbenches.

FPGA-SPICE is available at [17].

A. Extended Architecture Description Language

FPGA-SPICE extends the architecture description language of [8]. 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 modelling the circuit components of the FPGA modules.

First, transistor model and basic geometrical properties are defined in XML nodes *tech_lib* and *transistor*, as follows:

```
<tech_lib lib_path="45nmHP.pm" nominal_vdd="1.0"/>
<transistors pn_ratio="1.5">
  <nmos chan_length="45e-9" min_width="140e-9"/>
  <pmos chan_length="45e-9" min_width="140e-9"/>
</transistors>
```

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

Then, transistor-level circuit design parameters of a FPGA module are defined under a 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 defined spice models. The following code shows an example, where a 6-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">
  <port type="input" prefix="in" size="6"/>
  <port type="output" prefix="out" size="1"/>
  <port type="sram" prefix="sram" size="64"/>
</spice_model>
<pb_type name="n_lut6" spice_model_name="lut6">
</pb_type>
```

Under the XML property *spice_model*, the ports of a 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 above example of *lut6*, a user-customized SPICE netlist is defined in the XML properties, *sp_netlist*.

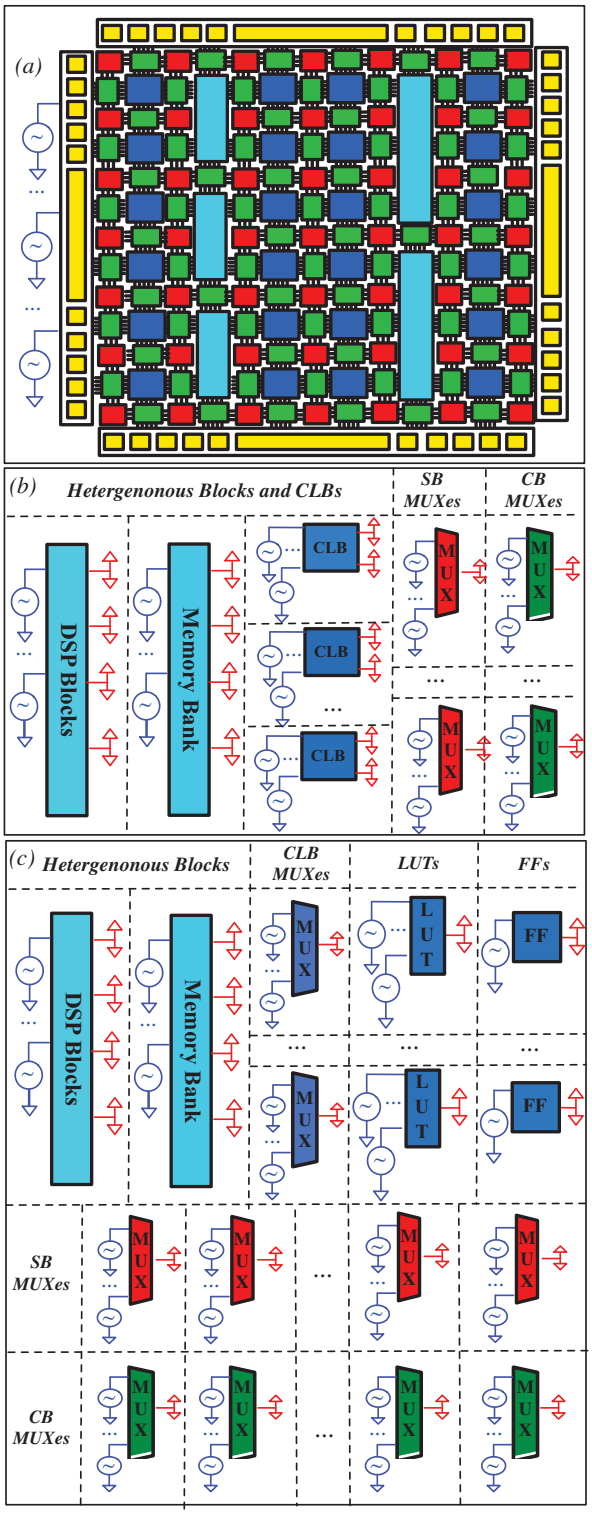

Fig. 4. Ilustration of the testbenches simularity: (a) Full-chip-level, (b)grid-level and (c) component-level testbenches.

B. Transistor-level Circuit Netlist Generation

In an FPGA, the circuit-level implementations for the different blocks, such as channel wires, multiplexers and LUTs, are highly dependent on the architectural choices. FPGA-SPICE can automatically determine their design parameters and generate their SPICE netlists. In this section, we will discuss the details of the circuit netlist generation engine.

1) Multiplexers: The multiplexers in FPGAs have diverse sizes and fan-outs, depending on their locations, i.e., in local routing or global routing.

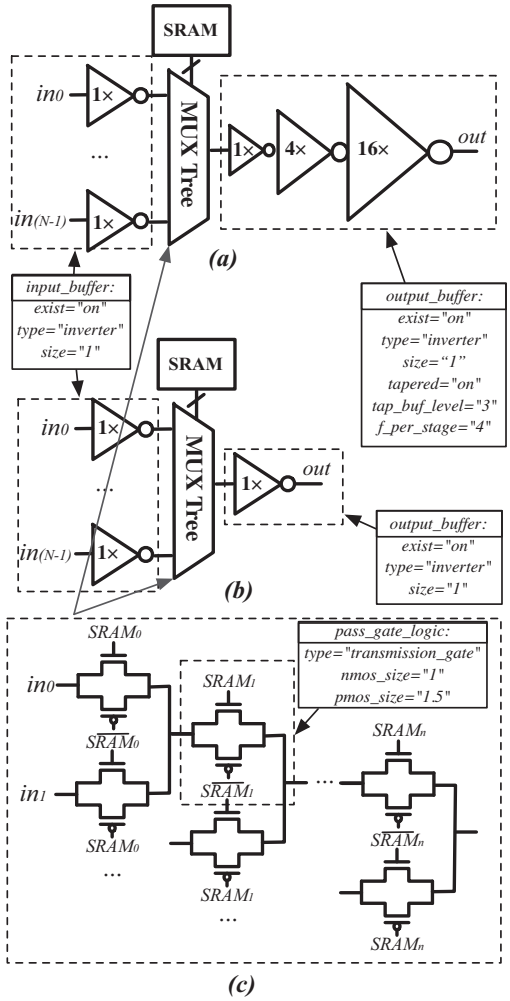

Fig. 5. Transistor-level circuit design of (a) a global routing multiplexer, (b) a local routing multiplexer, and (c) the internal tree-like structure.

In this context, different circuit-level optimization, such as transistor sizing and the use of tapered buffer, may apply. The transistor sizes and buffer allocation can be specified in the SPICE model definitions. The presence or absence of input/output inverters/buffers can be declared by setting the XML properties *exist* and *type*. Additionally, 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. 5(a) and Fig. 5(b), respectively. The tree-like structure of multiplexers is depicted in Fig. 5(c). The transistor-level circuit design of a global routing multiplexer in Fig. 5(a) can modelled by the following code:

```
<spice_model type="mux" name="sb_mux"/>
  <input_buffer exist="on" type="inverter" size="1"/>
  <output_buffer exist="on" type="inverter" tapered="on"
    tap_buf_level="3" f_per_stage="4"/>
  <pass_gate_logic type="transmission_gate"
  nmos_size="1" pmos_size="1.5"/>
</spice_model>
```

Global routing multiplexers require an output tapered buffer [21], in order to drive the long routing metal wires as well as downstream loads due to the SB and CB multiplexers [19]. The output tapered buffer in Fig. 5(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 tree-like internal structure of the multiplexer. Fig. 5(b) depicts the circuit design of a local routing multiplexer which interconnects CLB input pins to BLE input pins. Because the fanout of the multiplexer is typically small (one or two inverters), there is only a minimum-size output inverter.

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

2) Look-Up Tables: LUTs are crucial components in FPGAs as they serve as combinational function generators. Fig. 6 illustrates the transistor-level circuit design of a LUT considered in this paper, including SRAMs, decoded multiplexers, and buffers [20].

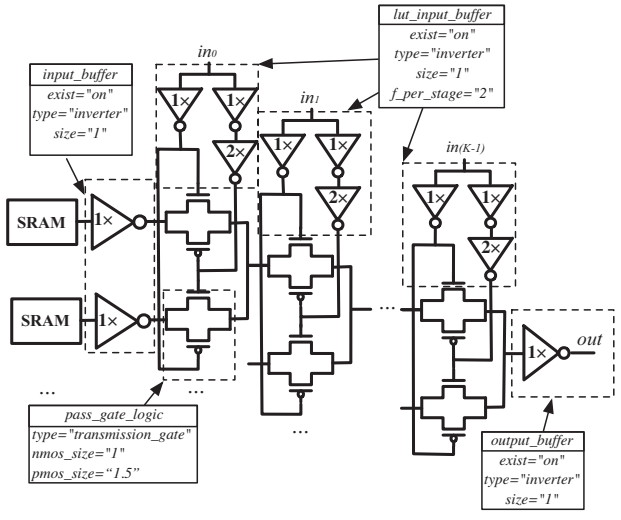

Fig. 6. An example of the transistor-level design of a LUT

The following XML properties are used to describe the circuit characteristics of the implementation in Fig. 6. 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" type="inverter"
    size="1" f_stage="2"/>
  <input_buffer exist="on" type="inverter" size="1"/>
  <output_buffer exist="on" type="inverter" size="1"/>
  <pass_gate_logic type="transmission_gate"
    nmos_size="1" pmos_size="1.5"/>
</spice_model>
```

C. Channel Wire

In modern FPGAs, channel wires are non-negligible modules owing to the facts that CLB area increases to contain heterogeneous blocks and difficulties in scaling down interconnecting metal wires. A length-L channel wire is abstracted as L cascaded segments, each of which spans a unique CLB. Fig. 7(a) depicts a length-2 channel wire in unidirectional routing architecture [6]. The channel wire is divided into two segments, namely $Segment_0$ and $Segment_1$.

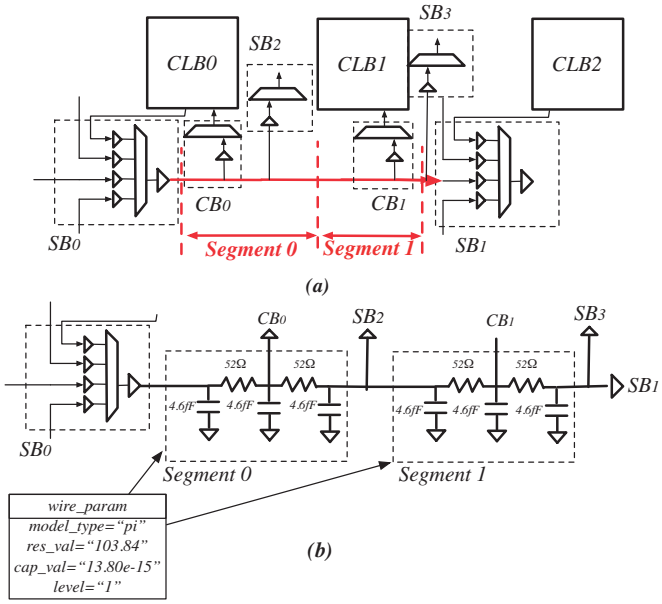

Fig. 7. (a) A length-2 unidirectional wire (highlighted in red) within FPGA routing architecture; (b) Corresponding RC modelling of segments

We assume that the inputs of CBs are connected to the middle of segments, breaking segments into two parts. We model each part of 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 a RC line can be customized by setting the XML property *level*. The total resistances and capacitance of a segment can be defined in XML properties *res_val* and *cap_val*, respectively. The following example describes the RC models of segments in Fig 7(b), corresponding to the segments in Fig 7(a).

```
<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>
```

D. Netlist Splitting Strategies

Full-chip-level netlists, that consider the full FPGA fabric in unique SPICE testbenches, produce accurate analysis but at the cost of large simulation time and memory usage. FPGA-SPICE can distribute the individual elements of a full-chip-level testbench into separate grid/component-level testbenches, significantly reducing the simulation time and memory usage at the cost of a lower accuracy. In this section, we introduce the two techniques used in FPGA-SPICE to split a full-chip netlist, namely voltage stimuli/load extraction and parasitic activity estimation.

1) Voltage Stimuli and Loads Extraction: FPGA-SPICE generates its individual testbenches by including voltage stimuli and downstream loads. To illustrate the technique, Fig. 8 shows a BLE multiplexer (in blue) that is driven by signals A and B, and that fanouts to local routing 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 signal density and activities. The signal density defines the number of switching events of a signal in one clock cycle while the probability represents the proportion that the signal is in logic `1` during one system clock cycle. To relate these activity information, we set the frequency of the voltage stimuli to:

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

The pulse width of a voltage stimuli is set to:

$$pulse\_width = freq \cdot probability(Signal). \quad (2)$$

Then, FPGA-SPICE adds the loads of the block by extracting the downstream elements in the architecture (highlighted in red in Fig. 8(a)). The downstream loads of a grid/component should be included in the testbench 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 [3]. Note that, if the downstream loads include channel wires, the channel wires should be extracted and included to the testbench.

2) 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. 9 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, in such condition, net_0 will be propagated through the routing structure (red path). These parasitic activities will cause

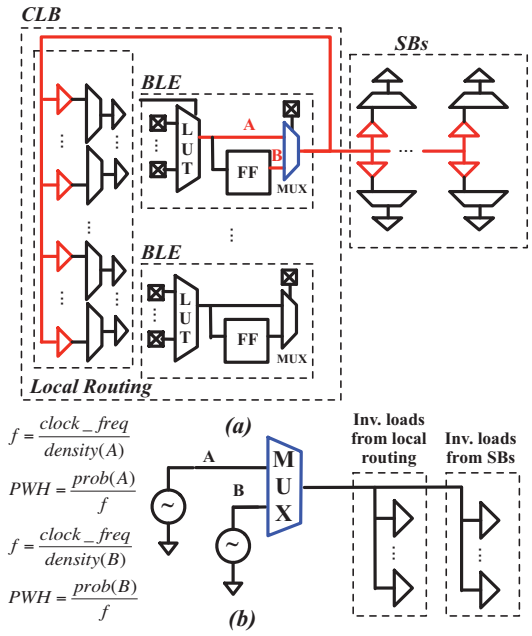

Fig. 8. Ilustration of the voltage stimuli generation and load extraction techniques. (a) BLE multiplexer with its architectural context; (b) extracted testbench.

extra power consumption and should be taken into account. FPGA-SPICE performs parasitic activity estimation for all the unused nets after routing stage.

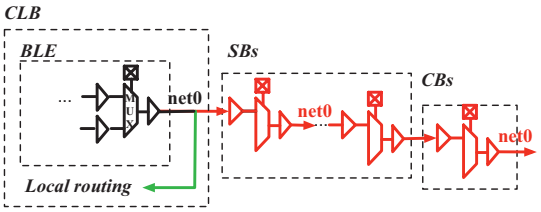

Fig. 9. An example for parasitic nets estimation.

IV. Experimental Results

In this section, we study the runtime, memory usage and accuracy of the different levels of FPGA-SPICE. Then, we use FPGA-SPICE to study the power breakdown of a modern FPGA architecture under different technology nodes and compare the results to standard analytical models, i.e., VersaPower.

A. Methodology

We use the FPGA-SPICE EDA flow in Fig. 3. MCNC big20 benchmarks [23] are selected as the EDA flow inputs. First, ABC synthesizes the benchmarks and ACE2 estimates the signal activities. Then, VPR packs, places and routes. Afterwards, the FPGA-SPICE generates the full-chip/grid/component-level testbenches with the architecture XML files. In the last step, we run SPICE simulators to analyze power. The experiments are run on a 64-bit RedHat Linux server with 32 AMD Opteron Processors and 128Gb memory.

In this paper, we resemble the architecture of an Altera Stratix IV FPGA, where each CLB contains $I = 33$ inputs pins and $N = 10$ fracturable 6-input LUTs ($K = 6$). Length-4 unidirectional routing architectures are employed to interconnects Wilton's Switch Blocks, where $F_s = 3$. We set $F_{c,in} = 0.15$ and $F_{c,out} = 0.10$. The channel width, W, is set to 120 by adding 20% margin to the minimum channel width that VPR can route the biggest tested benchmark. All the architecture description files used in this paper are available at [17]. We investigate three technology nodes, 22nm, 45nm and 180nm using the PTM model [24]. The transistor-level circuit designs of SRAMs, FFs and multiplexers are derived from [20]. We model routing wire segments with a one-level π-type RC models and the wire parameters are derived from ITRS [22]. We determine the simulation clock period by adding 20% slack to the VPR critical path delay, in order to consider errors between the timing analysis engine and SPICE simulations [6]. The time period of simulations should be a full operating cycle by considering the least active signal, as follows:

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

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

B. Studies on Runtime, Memory Usage and Accuracy

Simulating full-chip-level testbenches is the most accurate approach to power analysis at the cost of runtime and memory usage. Table I compares the runtime, memory usage and power results of full-chip/grid/component-level testbenches at different technology nodes, obtained for the MCNC big20 benchmark *s298*. Compared to the full-chip-level testbench, the grid-level testbenches achieve 12× speed-up in runtime with a moderate 14.5% error on average over the different technology nodes. Compared to the full-chip-level testbench, the component-level testbenches accelerate 14× in runtime with a 13.6% error on average over the different technology nodes. Component-level testbenches lead to the best trade-off in runtime and accuracy loss thanks to the efficient netlist splitting strategies discussed in Section III-D. Therefore, in the following, we use component-level power results to study power breakdowns. We also examine the effects of the parasitic activity estimation on the accuracy. Without the parasitic activity estimation, on average, the accuracy of the grid-level and the component-level testbenches degrades by 16%, while the runtime is reduced by 18%. Especially, the power of CBs and SBs gets under-estimated by 37%.

C. Power Breakdowns

In this part, we use FPGA-SPICE to study the power breakdowns of the considered FPGA architecture. Fig. 10 shows the power repartition by components for the three considered technology nodes. These breakdowns are obtained by averaging the results over the complete MCNC big20 suite. In general, the routing architecture consumes 90% of the total power with the global routing architecture takeing 60% of the overall power. When the technology scales down from 180nm to 22nm, the power share of the global routing architecture increases, resulting from the fact that interconnect does not scale down as the same ratio as transistors do. Indeed, the parasitic transistor capacitance decreases by 90% from 180nm to 22nm technology node but the interconnect capacitance per length is reduced by only 70% [5]. Consequently, at 22nm and 45nm technology, the number of stages in the SB tapered buffers in typically larger in order to drive the interconnect wires. Therefore, the power share of SBs grows from 180nm to 22nm technology. The obtained results are in accordance with literature [5].

D. Accuracy Examination on VersaPower

We compare the power breakdown results between FPGA-SPICE and VersaPower, as shown in Fig. 10. FPGA-SPICE predicts that the local routing architecture occupies as large power share as the global routing architecture, which is different from the VersaPower. It can be explained in the following reasons. First, FPGA-SPICE takes the parasitic net activities in 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. Second, 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 [5]. Due to the large runtime cost, we only compare the power results of the *s298* benchmark between VersaPower and FPGA-SPICE. VersaPower over-estimates the total power by at least 24%, when compared to the full-chip-level results.

V. Conclusion

This paper introduces a simulation-based power estimation 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 generates SPICE netlists at different levels of complexity, considering precise technology mapping, placement and routing information as well as technological data. It subsequently uses SPICE simulators to perform accurate power analysis, leading to better accuracy compared to analytical power models. As a general-purpose power estimation framework, FPGA-SPICE can support more transistor-level circuit design topologies, such as one-level/two-level multiplexers as well as emerging technologies, such as *Resistive Random Access Memories* (RRAMs).

Acknowledgments

This work was supported by the Swiss National Science Foundation under the project number 200021-146600.

References

[1] F.N. Njam, *A Survey of Power Estimation Techniques in VLSI Circuits* , IEEE TVLSI, Vol. 2, No. 4, pp. 446-455, 1994.

[2] J. Lamoureux *et al.*, *Activity Estimation for Field-Programmable Gate Arrays*, IEEE FPL, pp. 87-94, 2006.

[3] K. K. Poon *et al.*, *A Detailed Power Model for Field-Programmable Gate Arrays*, ACM TODAES, Vol. 10, No. 2, pp. 279-302, 2005.

TABLE I. Comparison of runtime, memory usage and total power of full-chip/grid/component-level testbenches for 22nm, 45nm and 180nm technology nodes in the case of the MCNC big20 benchmark *s298*.

Benchmark: s298	Runtime (No. of minutes)			Peak Used Memory (Mb.)			Total Power (mW)		
Testbench/Tech.	**22nm**	**45nm**	**180nm**	**22nm**	**45nm**	**180nm**	**22nm**	**45nm**	**180nm**
Full-chip-level	129.48	106.15	102.56	4780	4827	4306	1.56	4.13	15.63
Grid-level	10.27(-92%[1])	9.82(-91%[1])	8.25(-92%[1])	768(-84%[1])	768(-84%[1])	825(-81%[1])	1.41(-9%[2])	3.37(-18%[2])	18.03(+15%[2])
Component-level	7.42(-94%[3])	6.97(-93%[3])	6.23(-94%[3])	589(-88%[3])	584(-88%[3])	621(-86%[3])	1.45(-7%[4])	3.21(-21%[4])	17.57(+12%[4])

[1]Gain(%) = (Grid-level/Full-chip-level-1)×100% [2]Error(%) = (Component-level/Full-chip-level-1)×100%
[3]Gain(%) = (Grid-level/Full-chip-level-1)×100% [4]Error(%) = (Component-level/Full-chip-level-1)×100%

TABLE II. Comparison of accuracy by modules in full-chip/grid/component-level testbenches for 22nm, 45nm and 180nm technology nodes in the case of the MCNC benchmark big20 *s298*.

Benchmark: s298	CLB Power (mW)			CBs Power (mW)			SBs Power (mW)		
Testbench/Tech.	**22nm**	**45nm**	**180nm**	**22nm**	**45nm**	**180nm**	**22nm**	**45nm**	**180nm**
Full-chip-level	0.42	1.06	7.85	0.12	0.23	2.53	1.02	2.82	5.26
Grid-level	0.44(+5%[1])	1.17(+10%[1])	10.00(+27%[1])	0.11(-8%[1])	0.22(-5%[1])	2.67(-5%[1])	0.86(-15%[1])	1.99(-29%[1])	5.37(+2%[1])
Component-level	0.47(+12%[2])	1.01(-5%[2])	9.54(+22%[2])	0.11(-8%[2])	0.22(-5%[2])	2.67(-5%[2])	0.86(-15%[2])	1.99(-29%[2])	5.37(+2%[2])

[1]Error(%) = (Grid-level/Full-chip-level-1)×100% [2]Error(%) = (Component-level/Full-chip-level-1)×100%

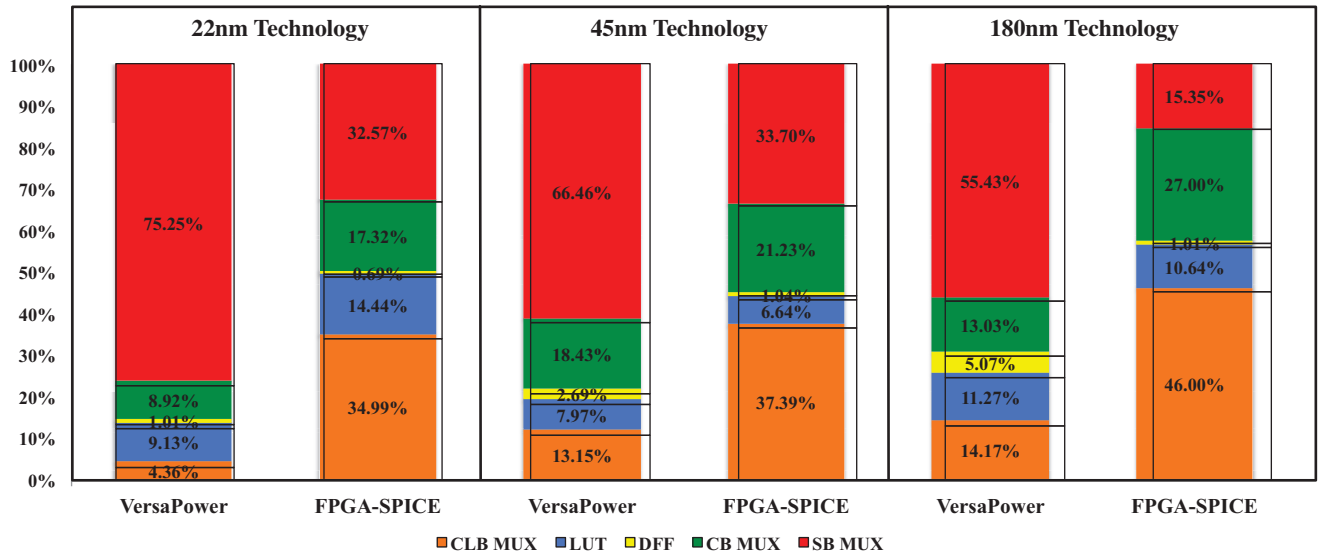

Fig. 10. Power breakdown results of the considered FPGA architecture between FPGA-SPICE and VersaPower averaged over the MCNC big20 benchmark suite for 22nm, 45nm and 180nm technology nodes.

[4] F. Li *et al.*, *Power Modeling and Characteristics of Field Programmable Gate Arrays*, IEEE TCAD, Vol. 24, No. 11, pp. 1712-1724, 2005.

[5] J. B. Goeders *et al.*, *VersaPower: Power Estimation for Diverse FPGA Architectures*, IEEE ICFPT, pp. 229 - 234 , 2012.

[6] V. Betz *et al.*, *Architecture and CAD for Deep-Submicron FPGAs*, Kluwer Academic Publishers, 1998.

[7] J. Rose *et al.*, *The VTR Project: Architecture and CAD for FPGAs from Verilog to Routing*, FPGA, 2012, pp. 77-86.

[8] J. Luu *et al.*, *Architecture Description and Packing for Logic Blocks with Hierarchy, Modes and Complex Interconnect*, FPGA, pp. 227-236 , 2011.

[9] A. Vladimirescu, *The Spice Book*, John Wiley & Sons Publishers, 2012.

[10] D. Lewis *et al.*, *The Stratix II Logic and Routing Architecture*, FPGA, 2005, pp.14-20.

[11] Altera Corporation, *Stratix IV device handbook version SIV5V1-1.1*, July 2008. http://www.altera.com/literature/hb/stratix-iv/stratix4_handbook.pdf

[12] Xilinx, *Virtex-5 User Guide UG190 (v4.0)*, March 2008. http://www.xilinx.com/support/documentation/user_guides/ug190.pdf

[13] M. Hutton *et al.*, *Improving FPGA Performance and Area Using an Adaptive Logic Module*, FPL, 2004, pp. 135-144.

[14] J. Luu *et al.*, *On Hard Adders and Carry Chains in FPGAs*, FCCM, 2014, pp. 52-59.

[15] G. Lemieux, *et al.*, *Generating Highly-Routable Sparse Crossbars for PLDs*, FPGA, 2000, pp. 155-164.

[16] G. Lemieux *et al.*, *Using Sparse Crossbars within LUT Clusters*, FPGA, 2001, pp. 59-68.

[17] *http://lsi.epfl.ch/downloads*

[18] University of California in Berkeley, *ABC: A System for Squential Synthesis and Verification*, Available online. http://www.eecs.berkeley.edu/~alanmi/abc/

[19] E. Lee *et al.*, *Interconnect Driver Design for Long Wires in Field-Programmable Gate Arrays*, ICFPT, 2006, pp. 86-96.

[20] C. Chiasson *et al.*, *Should FPGAs Abandon the Pass-gate?*, FPL, 2013, pp. 1-8.

[21] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits, Second Edition*, Prentice-Hall Publisher, 2002.

[22] ITRS, Interconnect Chapter, 2011.

[23] S. Yang, *Logic Synthesis and Optimization Benchmarks User Guide Version 3.0*, MCNC, Jan. 1991.

[24] Predictive Technology Model, Available on http://ptm.asu.edu/.