2015 IEEE 9th International Symposium on Embedded Multicore/Many-core Systems-on-Chip

Hierarchical Library Based Power Estimator for Versatile FPGAs

Hao Liang*, Yi-Chung Chen†, Tao Luo‡, Wei Zhang*, Hai Li†, Bingsheng He‡
*Department of Electronic and Computer Engineering, The Hong Kong University of Science and technology (HKUST)
Clear Water Bay, Kowloon, Hong Kong
{hliangac, wei.zhang}@ust.hk
†Department of Electrical and Computer Engineering, University of Pittsburgh, Pennsylvania, USA
{yic63, hal66}@pitt.edu
‡School of Computer Engineering, Nanyang Technological University, Singapore
tluo001@e.ntu.edu.sg, bshe@ntu.edu.sg

***Abstract*—FPGA is a promising hardware accelerator in modern high-performance computing systems, *e.g.* cloud computing, big-data processing, *etc*. In such a system, power is a key factor of the design requiring thermal and energy-saving considerations. Modern power estimators for FPGA either support specific hardware provided by vendors or contain power models for certain types of conventional FPGA architectures. However, with technology advancement, novel FPGA of versatile architectures are introduced to further augment current FPGA architecture at various aspects, such as emerging FPGA with non-volatile memory, nanowire interconnection of reconfigurable array, *etc*. To evaluate the power consumption of various FPGA designs, the power estimator has to be made more flexible and extendable for supporting new devices and architectures. We introduce in this paper a novel power estimator with hierarchical library supporting power models at different levels, *e.g.* novel circuit of components, emerging memory devices, architecture of time-multiplexing fashion, *etc*. The power estimator also supports coarse-grain or fine-grain power estimation defined by users for achieving complexity-accuracy trade-off. Simulation results of benchmarks of our power estimator against commercial one demonstrate accuracy of our tool. Furthermore, we present an example of RRAM FPGA power estimation, which has novel memory devices and potential of power gating. Our tool demonstrates flexibility to well support, but not limited to, the power estimation of such state-of-the-art FPGAs.**

I. Introduction

Field-Programmable Gate Array (FPGA) has advantages of post-fabrication reconfiguration, low development risk, fast implementation to market and low cost for low volume products, *etc* [1]. Under high throughput and/or low latency requirement of modern applications, system designers adopts FPGA as auxiliary processors to boost performance. FPGA chips are found in Microsoft's Catapult system accelerating large-scale data center applications [2] and in Convey system for real-time signal processing applications [3].

Given that power consumption is an important metric to evaluate electronic systems and the fact FPGAs are extensively utilized nowadays, FPGA power analysis attracts a decent amount of research efforts. For modern FPGAs, vendors provide power estimation tools for specific commercial products themselves, such as *Xilinx Power Estimator* (XPE) [4] and *PowerPlay Early Power Estimators* (EPE) [5]. There are also academic FPGA CAD tools proposed to evaluate FPGA power consumption under certain architecture assumptions [6], [7]. K. Poon *et al.* propose a power model for FPGAs which includes very detailed model for each components [8]. The power consumption models are validated with HSPICE simulation. However, the existing power models are not sufficient to estimate power of new circuit designs with new technology nodes and advanced features. L. Shang *et al.* propose detailed switching activity model for dynamic power consumption [9]. The work is based on a commercial product and the analysis is based on measurement, so it is hard to be used to predict power of other commercial products or academic architectures. F. Li, *et al.* propose a flexible power estimation tool with a mix-level power model for FPGAs [6]. It pre-sets delay information and provide circuit models for SRAM FPGA. However, the tool is based on a conventional FPGA architecture and lack of support for advanced technology nodes and techniques. Versapower proposed by J. Goeders *et al.* can support power estimation for all architectures supported by VPR [7]. The work takes switching activity information from switching activity estimation tool ACE2.0 for dynamic power estimation [10]. Though it provides multiple technology nodes to estimate power, options are still limited for exploration of emerging FPGA architectures with new devices and circuits.

In this paper, we propose a full-customized power estimation tool based on hierarchical library targeting emerging FPGAs. The customized hierarchical library contains power models for various circuit components ranging from high-level blocks to low-level devices, which enables the tool to perform power estimation at both coarse-grain and fine-grain level; meanwhile complexity-accuracy tradeoff is allowed. Our tool can be easily extended to support emerging technologies. We validate our tool against commercial tool for its estimation accuracy and provide discussions of how to expand the tool for support of emerging technologies, such as non-volatile memory based FPGA and features like power gating.

The remaining paper is organized as follows. Section II introduces the framework of our power estimation tool. Section II covers more specific details about power library structure and estimation methodology. In Section III, we present the validation of the proposed tool and power estimation results of an example emerging FPGA to demonstrate the great flexibility of the proposed tool. Section IV concludes the paper and gives future direction.

978-1-4799-8670-5/15 $31.00 © 2015 IEEE
DOI 10.1109/MCSoC.2015.44

IEEE computer society

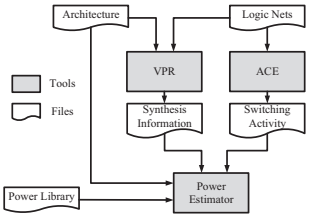


Fig. 1. Flow of the proposed power estimation tool.

II. Power Estimation Tool

A. Overview of the tool flow

The flow of the proposed power estimation tool is shown in Fig. 1. The tool takes in four major inputs: FPGA architecture description, placement and routing result, switching activity information and a power library. The four inputs provide information of four necessary steps of FPGA power estimation:

- What resources are available and how are they organized on FPGA?
- What resources are utilized to implement a circuit?
- How are these resources been utilized?
- How to estimate power of the resources based on their utilization condition?

FPGA architecture description file is in the XML format defined by VPR which only supports island-style FPGAs for now. Hierarchy is inherited in the nature of XML format. The format is easily comprehensible processable by human and programs. We extract information from the file and build the architecture correspondingly with FPGA organization and connection details. The architecture file along with a netlist of the circuit to be implemented on FPGA serve as input to VPR. The netlist is technology-mapped and optimized in the format of *Berkeley Logic Interchange Format* (blif) by ABC [11]. In this case, VPR will pack, place and route it and give the physical synthesis results to our tool. For the record, VPR is not the only choice here, any results generated by any other packing, placement or routing algorithm or tools are acceptable as long as the results are in the VPR format and does not go against architecture assumptions. We decouple our power estimator from physical synthesis tool like VPR to allow flexibility of supporting different FPGA architectures and CAD flows. Users can use this tool to evaluate their packing, routing and placement algorithms' performance with respect to power consumption. The netlist is fed to the switching activity estimation tool ACE2.0 as well. By assigning signal probabilities and switching activities to primary inputs of the circuit, it will calculate signal probability and switching activity information of every net in the netlist. Such information is then input to our power estimator to provide guidance in evaluating dynamic power consumption of the circuit. Power library, as the most important concept in this paper, will be elaborated in detail in the next section. Basically it contains information about how much power a typical component consumes under certain conditions, and how we scale it to adapt to various scenarios in order to get reliable power results.

B. Custom-defined Library

As shown in Fig. 2, our power estimation scheme is a hierarchical approach. All the high-level components in the architecture are composed of lower-level components, and the bottom level components are basic elements in our estimation framework. At the top level of abstraction, FPGAs can be divided into 3 major parts: logic resources, routing resources and clock tree. Different FPGAs have different designs and organizations of the three constituents, varying from typical island-style design [12], hierarchical design [13] to 3D design [14].

A general island style FPGA architecture is shown in Fig. 3. Logic resources in island-style architectures are usually grouped as *complex logic blocks* (CLBs). They can be programmed to implement various logic functions. CLBs are surrounded by a "sea" of routing resources. Routing elements are normally composed of routing tracks, *connection blocks* (C blocks) and *switching blocks* (S blocks). Connection blocks provide programmability to connections between CLB pins and routing tracks, while switching blocks provide programmability for track-to-track connections. Clock trees are used to propagate clock signal and synchronize operations of circuits. Again the design of these components are not identical in all island-style FPGA designs. There can be different amount of logics in one CLB, typically different number of look-up tables; the programmable switches in connection blocks and switching blocks can be implemented differently by multiplexers, pass-transistors or tri-state buffers. CMOS designs of look-up tables, multiplexers, pass-transistors or buffers can be further decomposed into physical wires and CMOS transistors.

Performing power estimation at which level is sometimes hard to determine. At higher levels, power estimation is fast but usually suffers from inaccuracy, while at lower levels, it is time-consuming and leaves little flexibility on design details. We understand the need of making tradeoff at different abstraction levels and provide the best possible power estimation according to available design details or user specified abstraction level.

Fig. 2 shows three levels we can look at an FPGA architecture, from high abstraction to high resolution. In the top level, *e.g.*, defined as Abstract level, the FPGA consists of logic, routing resources and clock tree. If we zoom into these high-level components and decompose them in the Basic level, we

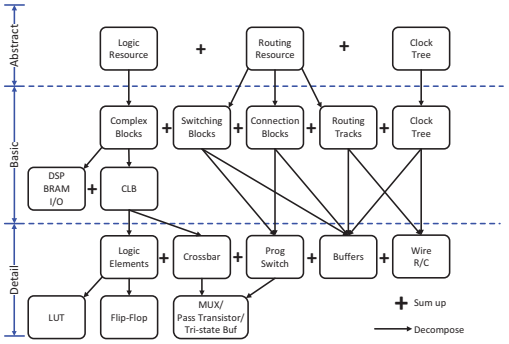


Fig. 2. Hierarchy of custom-defined library

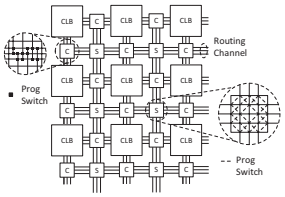


Fig. 3. A general island-style FPGA architecture[15]

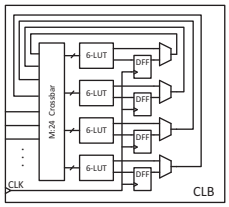


Fig. 4. A general CLB design

can divide them into basic components. The Logic Resources include complex blocks like CLBs, DSPs and IO blocks, etc. The Routing Resources are composed of connection blocks, switching blocks and routing tracks. The elements at this level can be further decomposed into primary elements in Detail level, such as look-up tables (LUT), D-flip-flops (DFF), multiplexers (MUX), Buffers, etc. Our estimator usually should perform power estimation on Detail and Basic level, and the power library is supposed to contain power information for required components in the two levels. In previous academic tools, the primary elements at our bottom level are usually further decomposed into wires and transistors, but we do not wish to go that far since transistor level estimation usually takes significantly longer time and is not necessarily more accurate then our framework when estimating power of an entire FPGA. Another important reason for us to choose them as the primary components is that the connectivity between them are usually simple, wires and transistors usually have very complicated connections to form Detail level components and not suitable to serve as libraries.

For each component defined in the library, it should give power information under two scenarios: static power consumption and dynamic power consumption. Static power information is the average power consumption of the component when switching activity is zero for all the inputs of the component with all possible input signal combinations. Dynamic power consumption related information, however, can be presented in two ways: macro-based or model-based. In macro-based style, dynamic power information is the average dynamic power consumption of the component when all the inputs of the component are random bit streams. In model-based style, however, dynamic power information is a coefficient vector with the length of component input size, in which each coefficient indicates the contribution of switching activity of the corresponding input signal to dynamic power consumption of the component. A specific example of how to use the library power information is provided in Section II-C2.

What are the flexibilities that users are guaranteed under this framework? First of all, users are able to choose at which level they want to perform their power estimation. For example, black boxes like multipliers, Block RAMs, DSP blocks or I/O pads on FPGAs are hard to decompose into basic elements due to their complexity or confidentiality, or sometimes the accurate power consumption measurement is readily available. In this case, users may prefer to regard these components as atomic and perform power estimation at coarser grained level. In the library, users can estimate power of different types of components at different abstraction level. For example, for the DSP blocks, the estimation can be at Basic level, and for CLB power estimation, Detail level may be provided for higher accuracy. Another very important consideration is the continuous technology scaling, CMOS is facing its limit and emerging technologies are evolving fast, such as non-volatile memory, carbon nanotube transistors, etc. For emerging technologies and devices based on them, theoretical power models may not be available yet, and their power models usually comes from real measurement. Our library based power models sufficiently support the emerging technologies without any change in entire power estimation flow. Users with cutting edge FPGA techniques can simply replace the power information of original outdated components and perform power estimation directly.

Power library should always remain consistent with FPGA architecture assumptions in the architecture description file. For example, any component defined in the power library will be ignored and all its children will be discarded if the component is not defined in the architecture file, and any component in the power library should store power related information conforming timing and area constraints of the component specified in the architecture file.

C. Power Estimation Procedure

1) Top Down Approach: During power estimation, the tool will start with top-level components first in the architecture, the estimator checks if it is defined in the power library. If yes, the estimator extract corresponding power information from the library, calculate corresponding power and ignore the detail implementation of this component. If the component is expanded in further details, the estimator breaks the component into lower-level children components and query them in the library recursively. If not, the power of the undefined components will be ignored and a warning is reported.

In general, FPGA power is divided into three parts: logic, routing and clock power (Special circuits like I/O pads and clock generators on FPGA are usually not considered, but in this tool, they are welcome to be defined in the power library and contribute to the overall power consumption). Configuration bit SRAM cells consumes static power but the power is usually included in the corresponding logic or routing components. However, as shown in Equation (1), in this work we include power of writing configuration bit cells as well. This power, or often referred as reconfiguration power in FPGA, is the power consumed during writing operation of configuration cells to initialize or change the functionality of FPGA circuits. Modern FPGAs usually provide features like partial reconfiguration or dynamic reconfiguration, which require run-time writing of reconfiguration cells and bring in reconfiguration power overhead. But reconfiguration power is not considered in any previous FPGA power estimation tools before. We are the first to integrate reconfiguration power in an FPGA power estimation tool.

Equations (2) and (3) explain the break down of the logic and routing resources from Abstract level down to Basic level according to Fig. 2. Among them, clock network power is special that it was calculated separately with methods introduced by [7]. Configuration power is discussed later. If a Basic level component is not defined in the library, it will be further decomposed by our tool down to Detail level.

As shown in Fig. 4, a user-defined CLB could be a cluster of four 6-input LUTs with a fully populated M-to-24 crossbar from CLB inputs to LUT inputs which provides intra-CLB connections. M is the sum of CLB input pin count and FlipFlop feedback count, which is available from the architecture description. For instance, if a CLB contains 33 input pins and 4 FlipFlops, M would be equal to 37. In the library, users can decide whether they provide the power-related information for the entire CLB directly. If they do, the tool would use the library to generate atomic power consumption of the entire CLB. By default, though, the crossbar will be decomposed by out tool to 24 M-to-1 multi-level multiplexers, which are again recursively divided into multiple single level multiplexers provided by the library. LUTs and FlipFlops can be primary components and will not be decomposed further to wires and transistors.

$$P_{FPGA} = P_{routing} + P_{logic} + P_{clock} + P_{conf} \tag{1}$$

$$\begin{aligned} P_{routing} = \Sigma P_{track.D} + \Sigma P_{sb.D} + \Sigma P_{cb.D} \\ + \Sigma P_{track.S} + \Sigma P_{sb.S} + \Sigma P_{cb.S} \end{aligned} \tag{2}$$

$$\begin{aligned} P_{logic} = \Sigma P_{clb.D} + \Sigma P_{dsp.D} + \Sigma P_{io.D} \\ + \Sigma P_{clb.S} + \Sigma P_{dsp.S} + \Sigma P_{io.S} \end{aligned} \tag{3}$$

2) Power Estimation of Single Component: When the top down approach reaches the granularity at which users define their power library, power estimation of single components at that level will be triggered. In Equations (2) and (3), suffixes *.D* and *.S* present dynamic power and static power of a component respectively. In this section, we will mainly introduce our procedure in dynamic and static power estimation of one single component.

Static power of a single component is directly extracted from the power library as in Equation (4). As mentioned in Section II-B, the static power information stored in the power library is the power consumption of the component whose input signals at zero switching activity, and this static power value Lib_{static} is averaged among all possible input signal combinations.

$$P_{Component.S} = Lib_{static} \tag{4}$$

Dynamic power, on the other hand, is more complicated because it is related to run-time switching activity and frequency of the component. We use *SA* to present the *switching activity* of a component, which is the average number of both 0-to-1 and 1-to-0 transitions happened per clock cycle and divide by 2.

We introduced in Section II-B that we provide two styles of defining dynamic power information in the power library. In macro-based style, the power library will provide a measured/simulated average power number $Lib_{dynamic}$ for the component, representing an average dynamic power consumption value of the component. In this scenario, the component's all input signals are independent random bit streams at the same bit rate $1/f_{measure}$. The tool then apply $Lib_{dynamic}$ to our power estimation with Equation (5). In this equation, $SA_{measure}$ is the average switching activity of the component during measurement/simulation. In macro-based style,

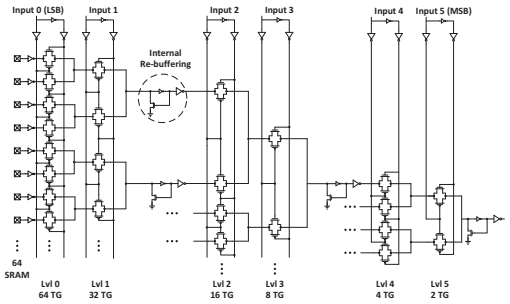


Fig. 5. A transmission gate based 6-input LUT design with internal re-buffering every 2 levels

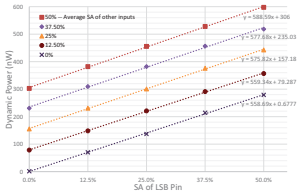


Fig. 6. Dynamic power of LUT6 scaling with signal SA of LSB

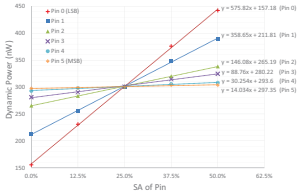


Fig. 7. Dynamic power of LUT6 scaling with signal SA of input pins

as all input signals are independent random bit streams, the occurrence probability of 0-to-1 and 1-to-0 transitions is 0.5. According to the previous definition of SA, we divide the probability of transitions by 2 and $SA_{measure}$ is numerically 0.25 in model-based dynamic power library definition. $SA_{current}$ is the average SA of all input signals of the components with the help of ACE in the FPGA synthesis flow. $f_{current}$ is the current frequency of the implemented circuit, reciprocal of the critical path delay decided by placement and routing results.

$$P_{Component.D} = \frac{SA_{current} * f_{current}}{SA_{measure} * f_{measure}} * Lib_{dynamic} \tag{5}$$

Model-based dynamic power estimation for single components, however, is based on the observation that for some components input signal SA would have asymmetric influences on dynamic power consumption. Taking a LUT6 component depicted in Fig. 5 as an example, input 0 (least significant bit) controls on/off states of 64 transmission gates, while input 5 (most significant bit) only controls on/off states of 2 transmission gates. Flipping the value of input 0 would most probably cost more dynamic power in the LUT6 than toggling input 5.

Fig. 6 elaborates how dynamic power scales with variance in SA of the LSB. Values stored in 64 SRAM cells are assigned randomly and signals of each input pin are generated by HSPICE *Pseudo Random-Bit Generator Source* (PRBS). By changing the bit rate parameter of PRBS, we are able to get signals of different SA at specific clock frequency. We keep other input pins' SA remain the same while varying SA in LSB. The figure shows a linear relationship between dynamic power consumption of a LUT6 and the SA of LSB. Fig. 7 further presents how dynamic power varies with SA of all LUT6 input pins with different slopes in the linear regression curve.

Such modeling methodology discards the obscurity in the details of a black box component and allows users to define dynamic power consumption of a single component through a vector of coefficients indicating the impact of input signal SA on overall dynamic power consumption. Hence in model-based dynamic power information definition, users need to provide a coefficient vector with the length of number of input pins of the component. Each coefficient in the vector provides the information about how dynamic power of this component scales with the switching activity of this pin. For now we assume linear relationships only and the values in the coefficient vector is acquired from the slope value of linear regression curve of different HSPICE simulation results. Dynamic power consumption is estimated with Equation (6), where SA_i is the switching activity of the *ith* input pin of the component and $coef_i$ is the aforementioned linear regression slope corresponding to the *ith* input pin SA.

$$P_{Component.D} = \Sigma(SA_i * coef_i) \tag{6}$$

Power consumption of tracks are considered differently, however. Tracks in FPGAs are designed not to be identical, and there can be length-1, length-2, length-4 or even longer track segments. But the capacitance of tracks are relatively easy to obtain. In our tool, track power is estimated by Equation (7). Capacitance of a length-1 track segments C_{wire} should be provided by the library (related to the design and feature node). $P_{track.S}$ is considered zero.

$$P_{track.D} = SA * {V_{dd}}^2 * C_{wire} * length * f \tag{7}$$

D. Extensions to the Framework

1) Run-Time Reconfiguration: New participants in FPGA power consumption can be easily estimated by enriching the power library. In Equation (1), P_{conf} is the power of configuration bit cells. Normally P_{conf} is zero that if an FPGA is only configured once and users do not care too much about the power consumption of the initial configuration. But power consumption of run-time reconfigurable FPGA becomes assessable if we support one more parameters for configuration memory cells, write energy, denoted $E_{conf.write}$. We take NATURE [16] as an run-time reconfiguration example. NATURE folds a circuit into multiple reconfiguration stages, and generate physical synthesis information for partial circuit in every stage. For every stage, NATURE will have the critical path information, and generate the reconfiguration frequency f_{reconf}, which is the reciprocal of the longest critical path delay of all stages. NATURE assumes dynamic reconfiguration at each reconfiguration stage, so f_{reconf} is the frequency at which configuration memory cells are overwritten during run-time reconfiguration process. Hence P_{conf} is obtained by adding up the energy of every overwritten configuration bit at all stages and multiply by f_{reconf} as shown in Equation (8):

$$P_{conf} = f_{reconf} * \sum_{stages} E_{conf.write} \tag{8}$$

The total power consumption of run-time reconfiguration is in Equation (9):

$$P_{run-time} = P_{conf} + (\sum_{stages} P_{FPGA})/num\_stages \tag{9}$$

2) Power Gating: Emerging memory technologies are now very popular in FPGA designs, and among them non-volatility is one of the most important consideration when choosing a memory design. In order to better support non-volatile memory designs, we provide power gating for users who wants to shut down a part of FPGA to reduce static power consumption.

In our power gating strategy based on [17], we first decide which are the complex logic blocks are not utilized (from placement information), and those blocks are marked as power gated. All the lower level components inside those blocks will not consume static power. The next step we find connection blocks that can be power gated. Only if the two complex logic blocks beside one connection block are both power gated, we consider this connection block can be power gated and all the programmable switches will consume zero static power. Lastly, from routing information, we decide those switching blocks does not control active track-to-track connections, and mark them as power gated. Fig. 8 shows how power gating works in our three-step procedure.

3) Upcoming Features: Power gating is not the only feature this framework is capable of. It is yet very easy to be extended to other features like multiple clock domain, partial reconfiguration and so on. For example, we provide a choice for users to define the clock frequency of the component despite of the critical path delay information from physical synthesis in order to support coarse-grain FPGA and multiple clock domains in one FPGA. User can define a fixed clock frequency 300MHz of customized component such as DSP in the library, to make them always run at 300MHz, regardless of the critical path delay given by the synthesis tool.

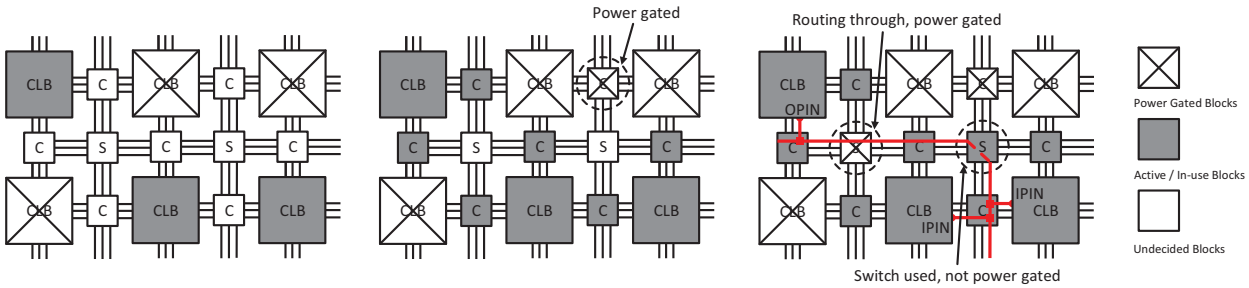
III. Experimental Results

The accuracy of this tool highly depends on the power information given by the hierarchical power library. However users can use our primary power library integrated with the tool if they do not have detailed information about low-level FPGA architecture.

In the experiments, we use VPR flagship architecture [18] to present our power estimation results. The architecture uses 6-input LUT as its logic resources. Such LUT6 and another D-flip-flop compose a *Logic Element* (LE) and 10 LEs are grouped into a CLB. Programmable Switches in the routing resources of the architecture adopt multiplexer design.

We designed our primary power library accordingly to provide power information of the following basic components: 6-input LUT (LUT6), 4-to-1 multiplexer (MUX4), D-flip-flop (DFF), Buffer (BUFFER), 6-transistor SRAM configuration memory cells (CONF) and routing track (TRACK). All the components are implemented with 32nm/28nm *Synopsys Interoperable Process Design Kit* (iPDK) Library via Synopsys University Program [19] in Typical-Typical corner. We use HSPICE simulation to obtain dynamic and static power macros at 200MHz clock frequency, operating at 1.05V supply voltage and 85°C temperature. The power information of the primary library are presented in Table I.

VersaPower [7] is the most recent FPGA power estimation tool which has been integrated in the current version of VPR, which adopts VPR CAD flow as we do. It builds a



(a) CLB power gating (b) C Blocks power gating (c) S Blocks power gating

Fig. 8. 3 steps to recognize and mark components can be power gated.

TABLE I. POWER LIBRARY

Component	Clock-activity Power (W)	Zero-activity Power (W)
LUT6	1.68e-6	1.56e-7
MUX4	4.96e-8	4.90e-9
DFF	7.4e-7	8e-9
BUFFER	7.03e-7	7.13e-9
	Write (J)	Static Power (mW)
CONF	2.59e-9	2.41e-9
	Segment-1 Capacity (pF)	
TRACK	2.21	

detailed low-level power model for components in the VPR architecture. Hence it should be the perfect verification tool for our estimation. However, we found they significantly over-estimated dynamic power consumption when primary inputs of a circuit has 0 switching activity, compared to commercial power estimators.

Hence we mainly compare with commercial Altera Early Power Estimator to help verify our tool. The architecture we used is the flag ship architecture VPR introduced, and it is similar to Stratix family developed by Altera in many aspects like logic equivalence in one CLB and tuned delay numbers. More information about the architecture is available on the Verilog-to-Routing Project [20]. We fixed the routing channel width to 300 to further approximate the real case in a Stratix family FPGA according to the description of the flagship architecture in [18].

The FPGA device we choose to present power estimation results of commercial product is model 5SEE9H40 from Stratix V family. It is fabricated in TSMC 28nm technology node and support up to 1GHz clock frequency. We choose this model because of two reasons: first it is big enough to support most of our benchmarks, and secondly it has the simplest structure without features like high-speed transceivers and PCI-E hard IPs in other models in Stratix V family. We try to use the simplest one because the architecture we use from VPR does not have specifications on those features, and hence we can get a closer estimation.

However, we need to mention that some of the features in the selected model is still not available in the VPR architecture like dedicated carry chains and DSP blocks. They will introduce gaps into power estimation. We will try to give a comprehensive analysis about them in the future. Note that Altera provides SPICE models of the transceivers and IPs on their website [21], and that makes our framework able to include these features easily.

We use a set of 19 benchmarks consisting of real application circuits provided by VPR to be implemented on both architectures. We assume that the primary inputs of every circuit have switching activity of 0.5 and signal probability of 0.5 (equal chance to have 0s and 1s). The primary input is fed to ACE2.0 to obtain detailed switching activity analysis for our estimation.

A. Static Power

Stratix V 5SEE9H40 FPGAs have 317,000 *adaptive logic modules* ALMs, and according to Altera's Device Handbook [22], each ALM is composed of two combinational adaptive LUTs. Hence this device totally has 634,000 adaptive LUTs. According to EPE, a 5SEE9H40 device consumes static power of $1.737W$, no matter what circuit it is implementing. We fixed our VPR architecture to a 120 by 120 grid to be able to fit in our largest benchmark, and this architecture will have 108,000 equivalent LUTs.

If we scale the size of 5SEE9H40 FPGA from 634,000 LUTs to 108,000 LUTs, we find out that this model will consume about $296mW$ at a similar size of our VPR architecture. Our power estimation tool estimates that the static power consumption of the academic FPGA architecture is $172mW$. This is an acceptable estimation because static power consumption of I/O pads, DSP blocks and block memories in VPR architecture are ignored in our estimation while they are estimated in EPE.

B. Dynamic Power

In this subsection, we will mainly discuss the dynamic power consumption estimated by our tool. Without loss of generality, we are going to ignore some benchmarks that has very small resource utilization or switching activity, as those circuits will generate very small amount of dynamic power compared to total power. Typically, we will analyze the dynamic power consumption of 8 benchmarks: bgm, ch_intrinsics, diffeq1, mcml, raygentop, stereovision0, sterovision1, stereovision2. In this subsection, I/O pad, Block Memory and DSP block dynamic power consumption will be ignored in both estimations in EPE or in our tool.

To begin with, how is dynamic power evaluated in EPE? There is no publicly available data on how EPE calculates power, but our observations in Table II give us an insight that EPE most likely assumes linear relationship between dynamic power and two parameters: average switching activity of the

circuit and the clock frequency the circuit runs on. Dynamic power is also linearly related to the number of used LUTs and registers in the circuit. Average fanout of nets is set to 4 in our observation. This is a very simple and rough estimation, without considering detailed switching activity information of each resource in use and how placement and routing will have influence on the dynamic power consumption. These factors are carefully considered in our power estimation framework.

In Table III, dynamic power consumption comparison is presented. Quartus is the synthesis tool for Altera FPGA synthesis flow. Frequency values in the table presents the maximum clock frequency allowed by Quartus or VPR for each benchmark circuit. Switching activity information is analyzed in our power estimation flow, and the average number of switching activity is provided to EPE for its power estimation. Due to optimization techniques like carry chains used in Altera FPGAs, critical path delay is significantly reduced comparing to the architecture provided by VPR, hence maximum frequency value obtained for each benchmark by Quartus is higher then frequency acquired in VPR flow. To make a fair comparison, we allow circuits to run at the same low frequency given by VPR flow and present the scaled dynamic power in the parentheses in EPE dynamic power column.

Considering the differences in the FPGA architecture assumption,power management and optimization techniques between two CAD flows, our tool gives power estimation within acceptable range of correctness, and shows the right trend of how power scales with resource utilization, frequency and switching activity. Residing at higher level of abstraction,we sacrifice certain accuracy to gain flexibility of supporting emerging technologies and advanced methodologies.

C. ReRAM FPGA power estimation and power gating

In order to demonstrate more on its flexibility, the tool is used to perform power estimation for an emerging *Resistive Random Access Memory* (RRAM) based FPGA [23]. Circuits in this FPGA can not be evaluated by previous tools because their tool flows only support CMOS based LUT, CB and SB designs. In [23], pass gate in CB and SB Switching are controlled by *Complementary Resistive Switches* (CRS) arrays composed by RRAM. And they use *3D High-Density Interleaved* (3D-HIM) structure to build RRAM based fast programming LUTs (refer to Fig. 9).

This design is currently not digestible by current FPGA power estimation tools as they are no longer using traditional CMOS transistor to build the circuits. However, we can deploy power related parameters of these customized circuits to our hierarchical library and generate corresponding FPGA power estimation.

Table IV showed the power numbers of these ReRAM circuits. In this customized power library, we add power numbers of a CB switch and SB switch at higher level, instead of implementing those programmable switches with multiplexers. We also replace the LUT6 power numbers by those from the ReRAM design. A CB switch here contains a pass transistor, a buffer and a ReRAM configuration cell. A SB switch, is a group of N pass transistors, buffers and ReRAM configuration cells, and N is decided by the architecture parameter Fs, defining the number of fanouts of a track going through a switching block. N equals to 3 in this design. The rest of the FPGA remains in CMOS SRAM-based design.

TABLE IV. RERAM CUSTOMIZED POWER LIBRARY

Component	Clock-activity Power (W)	Zero-activity Power (W)
CB Switch	5.02e-7	1.8e-7
SB Switch	6.52e-6	2.1e-6
LUT	1.97e-5	6.1e-6

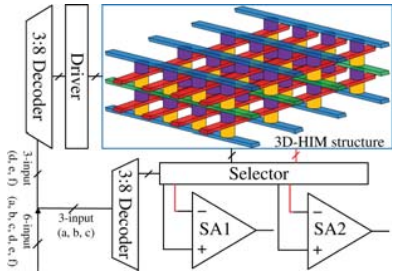


Fig. 9. 3D-HIM RRAM based 6-input LUT [24].

We found that the power numbers are significantly larger than SRAM counterparts, and the average static power consumption of the 19 benchmarks is $12.98W$. However, due to the non-volatility of ReRAM cells, we can shut down some of the unused circuits adopting the strategy introduced in Section II-D. We presented the ratio between static power consumptions in scenarios with and without power gating in 19 benchmarks, to give a hint about how much static power can be saved by this power management strategy. Fig. 10 shows this ratio along with LUT utilization numbers, and it shows that we can save up to 73% of static power with power gating, especially when implementing a small circuit on a large FPGA.

IV. CONCLUSION AND FUTURE WORK

In this paper, we propose a power estimation tool for emerging FPGA architectures with a full-customized hierarchical power library. Under this extremely flexible framework, we can evaluate power for a much broader range of FPGA architectures, including emerging memory techniques, run-time reconfiguration, power gating, *etc*. We evaluate our power estimation results with our primary library with Altera's Early Power Estimator power estimation, and prove that our power estimation can maintain good accuracy.

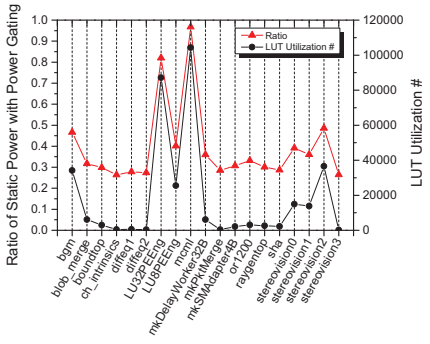


Fig. 10. Static power consumption ratio with power gating.

TABLE II. OBSERVATION OF EPE DYNAMIC POWER ESTIMATION

Scenario	# LUTs	# FFs	Clock Freq [MHz]	SA	Dynamic Power [mW]
Baseline	1000	1000	200	25%	22.12
2xSA	1000	1000	200	50%	44.24
0.5xSA	1000	1000	200	12.5%	11.06
2xFreq	1000	1000	400	25%	44.24
0.5xFreq	1000	1000	100	25%	11.06
+1000LUT	2000	1000	200	25%	34.23
-1000LUT	0	1000	200	25%	10.01
+1000FF	1000	2000	200	25%	32.13
-1000FF	1000	0	200	25%	12.11

TABLE III. DYNAMIC POWER ESTIMATION

Benchmark	# ALUTs (Quartus)	# LUTs (VPR)	Freq [MHz] (Quartus)	Freq [MHz] (VPR)	Dyn Power [mW] (EPE)	Dyn Power [mW] (Our tool)
bgm	25840	34121	82.19	35.09	31.77 (13.56)	41.37
ch_intrinsics	68	446	431.22	273.22	0.887 (0.56)	3.64
diffeq1	1598	529	95.68	50.25	5.30 (2.78)	3.65
mcml	132588	104296	32.66	9.26	39.03 (11.07)	8.76
raygentop	4222	2616	213.90	153.37	26.19 (18.78)	19.76
stereovision0	7594	14964	411.69	220.26	38.48 (20.58)	21.93
stereovision1	15012	13789	215.75	175.75	146.71 (119.51)	68.14
stereovision2	26440	36523	177.71	62.11	268.10 (93.70)	185.67

Due to its flexibility, a lot of research can be integrated into the framework. One important task is to enlarge our primary power library to support already existed circuits and designs. The tool can also be upgraded to support three dimensional FPGA power estimation as well. Other features like multiple clock domain, partial reconfiguration power are also possible extensions.

ACKNOWLEDGMENT

This work is in part supported by Grant R9336 of the Hong Kong SAR and a MoE AcRF Tier 2 grant (MOE2012-T2-1-126) in Singapore.

REFERENCES

[1] I. Kuon, R. Tessier, and J. Rose, "FPGA Architecture: Survey and Challenges," *Foundations and Trends in Electronic Design Automation*, vol. 2, no. 2, pp. 135–253, 2008.

[2] A. Putnam, A. Caulfield, E. Chung, D. Chiou, K. Constantinides, J. Demme, H. Esmaeilzadeh, J. Fowers, G. P. Gopal, J. Gray, M. Haselman, S. Hauck, S. Heil, A. Hormati, J.-Y. Kim, S. Lanka, J. Larus, E. Peterson, S. Pope, A. Smith, J. Thong, P. Y. Xiao, and D. Burger, "A Reconfigurable Fabric for Accelerating Large-Scale Datacenter Services," in *ACM/IEEE ISCA*, June 2014.

[3] Convey Computer, "The Convey HC-2™ Computer Architectural Overview," 2012.

[4] Xilinx, "Xilinx Power Estimator (XPE)," 2012.

[5] Altera, "PowerPlay Early Power Estimators (EPE)," 2013.

[6] F. Li, D. Chen, L. He, and J. Cong, "Architecture Evaluation for Power-efficient FPGAs," in *ACM/SIGDA FPGA*, 2003, pp. 175–184.

[7] J. B. Goeders and S. J. Wilton, "VersaPower: Power Estimation for Diverse FPGA Architectures," in *IEEE FPT*, 2012, pp. 229–234.

[8] K. K. Poon, S. J. Wilton, and A. Yan, "A Detailed Power Model for Field-Programmable Gate Arrays," *ACM TODAES*, vol. 10, no. 2, pp. 279–302, 2005.

[9] L. Shang, A. S. Kaviani, and K. Bathala, "Dynamic Power Consumption in Virtex™-II FPGA Family," in *ACM/SIGDA FPGA*, 2002, pp. 157–164.

[10] J. Lamoureux and S. J. Wilton, "Activity Estimation for Field-Programmable Gate Arrays," in *IEEE FPL*, 2006, pp. 1–8.

[11] A. Mishchenko *et al.*, "ABC: A System for Sequential Synthesis and Verification," 2009, http://www.eecs.berkeley.edu/alanmi/abc.

[12] P. Chow, S. O. Seo, J. Rose, K. Chung, G. Paez-Monzon, and I. Rahardja, "The Design of an SRAM-based Field-programmable Gate Array-Part I: Architecture," *IEEE TVLSI*, vol. 7, no. 2, pp. 191–197, June 1999.

[13] A. Aggarwal and D. Lewis, "Routing Architectures for Hierarchical Field Programmable Gate Arrays," in *IEEE ICCD*, Oct 1994, pp. 475–478.

[14] Xilinx, "1st Generation All Programmable 3D ICs," 2013.

[15] 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 TVLSI*, vol. 7, no. 3, pp. 321–330, Sept 1999.

[16] W. Zhang, N. Jha, and L. Shang, "A Hybrid Nano/CMOS Dynamically Reconfigurable System–Part I: Architecture," *ACM JETC*, vol. 5, no. 4, p. 16, 2009.

[17] A. A. M. Bsoul and S. J. E. Wilton, "An FPGA architecture Supporting Dynamically Controlled Power Gating," in *IEEE FPT*, 2010, pp. 1–8.

[18] VTR, "VTR FPGA Flagship Architecture Description File," 2013, https://code.google.com/p/vtr-verilog-to-routing/source/browse/trunk/vpr/sample_arch.xml.

[19] Synopsys, "Synopsys University Program," 2014.

[20] J. Luu, J. Goeders, M. Wainberg, A. Somerville, T. Yu, M. N. Nasartschuk, S. Wang, T. Liu, N. Ahmed, K. B. Kent, J. Anderson, J. Rose, and V. Betz, "VTR 7.0: Next Generation Architecture and CAD System for FPGAs," in *ACM TRETS*, May 2014.

[21] Altera, "SPICE Models for Altera Devices," 2013, http://www.altera.com/download/board-layout-test/hspice/hsp-index.html.

[22] ——, "Stratix V Device Handbook Volume 1," 2013.

[23] Y.-C. Chen, W. Wang, H. Li, and W. Zhang, "Non-volatile 3D Stacking RRAM-based FPGA," in *IEEE FPL*, 2012, pp. 367–372.

[24] Y.-C. Chen, W. Zhang, and H. Li, "A Look Up Table Design with 3D Bipolar RRAMs," in *ACM/IEEE ASPDAC*, 2012, pp. 73–78.