PEAF: A Power-Efficient Architecture for SRAM-Based FPGAs Using Reconfigurable Hard Logic Design in Dark Silicon Era

Zahra Ebrahimi, Behnam Khaleghi, Hossein Asadi, senior Member, IEEE

***Abstract*—Significant increase of static power in nano-CMOS era and, subsequently, the end of Dennard scaling has put a *Power Wall* to further integration of CMOS technology in *Field-Programmable Gate Arrays* (FPGAs). An efficient solution to cope with this obstacle is power gating inactive fractions of a single die, resulting in *Dark Silicon*. Previous studies employing power gating on SRAM-based FPGAs have primarily focused on using large-input *Look-up Tables* (LUTs). The architectures proposed in such studies inherently suffer from poor logic utilization which limits the benefits of power gating techniques. This paper proposes a *Power-Efficient Architecture for FPGAs* (PEAF) based on combination of Reconfigurable Hard Logics (RHLs) and a small-input LUT. In the proposed architecture, we selectively turn off unused RHLs and/or LUTs within each logic block by employing a reconfigurable controller. By mapping a majority of logic functions to simple-design RHLs, PEAF is able to significantly improve power efficiency without deteriorating the performance. Experimental results over a comprehensive set of benchmarks (MCNC, IWLS'05, and VTR) demonstrate that compared with baseline 4-LUT architecture, PEAF reduces the total static power and *Power-Delay-Product* (PDP), on average, by 24.5% and 21.7%, respectively. This is while the overall system performance is also improved by 1.8%. PEAF increases total area by 18.9%, however, it still occupies 22.1% less area footprint than the 6-LUT architecture with 31.5% improvement in PDP.**

***Index Terms*—Field-Programmable Gate Arrays, Static Power, SRAM, Dark Silicon, Hard Logic.**

I. Introduction

Field-Programmable Gate Arrays (FPGAs) are ubiquitously used in a wide range of applications from embedded systems to parallel high-performance computing. This widespread usage has been motivated by their inherent privileges such as inexpensive design update, shorter time-to-market, and flexibility to implement diverse range of applications, as compared to *Application-Specific Integrated Circuits* (ASICs). This flexibility, however, comes at the expense of 20x larger area, 4x lower performance, and 12x higher power consumption as compared to ASIC counterparts [1]. These challenges make FPGAs less attractive for designers in applications where power, performance, and die cost are of major concerns [2]. In particular, with tremendous growth in transistor density (e.g., more than six billion transistor on a single chip [3]), the role of power has become more challenging for designers, especially in low-power embedded applications. Exponential increase in power consumption of FPGAs can degrade the overall *Power Per Operation* which hinders the use of these devices, particularly in power-constrained applications [4].

With significant downscaling of transistor feature size and threshold voltage in recent years, ever-increasing static power dissipation, which is mainly attributed to idle status of the circuit components, has become a major contributor to the FPGA total power (5–87x of ASIC equivalent [1]). In embedded mobile applications, static power is even more pronounced since these devices are mostly in idle status. In conventional FPGAs, static power is mainly originated by leakage of SRAM-based configuration bits. The increasing static power creates a power wall which restricts the number of active transistors on a single die and hinders further logic integration in current and upcoming technologies.

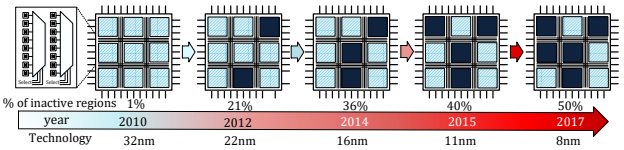

Fig. 1: Growth of dark silicon with technology downscaling due to increasing static power

An effective solution to overcome power wall along with preserving the performance efficiency is to apply power gating to inactive fractions of the silicon die. This phenomenon, illustrated in Fig. 1, has been referred to as *dark silicon*. As shown in this figure, the inactive fractions of the silicon die (shaded in black) contribute up to half of the total chip resources as technology ceaselessly shrinks. When it comes to FPGAs, the logic resources dissipate significant portion (about 50%) of the static power in SRAM-based FPGAs fabricated in 28nm and beyond [5]. Smaller footprint of logic resources besides their relative high power consumption is another testimony that implies reducing the logic power is vital due to its substantial *power density*. The higher power density in logic resources may cause hot spots or thermal challenges such as leakage-temperature positive feedback, performance degradation, and intensified aging [6]. Hence, power gating in logic resources can effectively help to improve power density, accompanied with the reliability of FPGAs.

One major source of high power dissipation of conventional FPGA logic resources is originated by power-*inefficient* structure of SRAM-based *Look-Up Tables* (LUTs) which implement logic functions within FPGAs. A typical structure of N-input LUT (N-LUT) is composed of a) 2^N SRAM cells which contain the truth-table of any N-input function and b) a 2^N-to-1 multiplexer which routes the appropriate cell value to the output. Among various configurations of LUTs, six-input LUTs (i.e., 6-LUT) are well-suited for performance

efficiency while 4-LUTs afford designs with smallest area in conventional FPGA architectures [7]. Despite the generous flexibility of a LUT, it has poor logic utilization efficiency due to non-uniform distribution of functions used in applications. Such flexibility comes with the cost of higher power consumption, inefficient structure and thereby performance, and more area overhead as compared to ASIC counterpart. Confronting the power wall due to downscaling of transistor feature size, power dissipation of FPGAs has become more pronounced for upcoming technologies.

To tackle the power wall in SRAM-based FPGAs, various approaches have been proposed which can be classified into three major categories. The first category exploits low-leakage manufacturing process such as variable transistor gate length, triple oxide, and multiple-Vth in recently fabricated devices [8]. Although these elaborate fabrication processes are not cost-efficient, yet they can be applied along with other power-efficient approaches. The second category has targeted mainly to reduce static power by static or dynamic power gating of unused logic and routing resources [5], [9], [10]. Static power gating is applied offline (i.e., during configuration time) only to unused resources of the design, while dynamic power gating is applied online (i.e., during runtime). The most important challenge in dynamic power gating architectures is the so-called *Inrush Current*, i.e., a large wake-up (power-on) current. This is drawn from the power rails which may lead to register content instability, functional error, and more wake-up time and power overhead [11]. Notice that the problem of inrush current does not exist for static power gating techniques wherein a power-gated module remains off permanently, and thus, no abrupt current is drawn. In addition, relying on inefficient soft logic (LUTs) building blocks is a major obstacle in these architectures to improve power, area, and performance. Additional resources for power control switch which incurs area and performance overhead to the circuit is another negative point attributed to these works. The third category has focused on using *Reconfigurable Hard Logic* (RHL) in conjunction with soft logic in the FPGA architecture in order to achieve area and performance efficiency [12]–[19]. In such studies, however, power aspect of the suggested architectures has not been neither considered nor evaluated.

In this paper, we present a fine-grained heterogeneous architecture based on static power gating, called PEAF, that aims to improve both power and performance efficiency of SRAM-based FPGAs. In PEAF, an efficient set of RHLs is developed to replace the conventional LUT. The applicability of the proposed RHLs is motivated by comprehensive function characterization in industrial and standard benchmark circuits, i.e., MCNC, IWLS'05, and VTR. Based on this characterization, it is revealed that substantial fraction of functions (up to 97%) is homomorphic (i.e., have the same homomorphic structure or uniformity) and can be implemented with power/performance-aware RHLs, making them a promising alternative for 4-LUT. Our analysis also shows that three-input functions contribute the most to those functions that could not be mapped into any RHL. Such functions can be implemented by 3-LUTs. Based on this investigation, each 4-LUT is replaced with a *Reconfigurable Logic Unit* (RLU) which comprises three 4-input RHLs with eight or less number of SRAMs and one 3-LUT.

To alleviate the area overhead of the proposed architecture, we propose a novel technique referred to as *Logic SRAM Sharing* (LSS) scheme. In this technique, the SRAMs of each RHL/LUT can be shared among all RHLs and 3-LUT, since at most one of RHLs or 3-LUT is activated at a time (i.e., only the RHL/LUT that can implement the mapped function to the corresponding logic block will be activated). In addition to the aforementioned power saving through exploiting power gating technique, another advantage of the LSS technique is to enhance the reliability of the FPGAs due to using RHL/LUTs with eight SRAMs as contrary to the conventional 4-LUTs by reducing number of SRAMs of a logic block (which are significantly susceptible to energetic particles and strikes [20]). A *Reconfigurable Power-Controller* (RPC) is also designed to power on either one of RHLs or the 3-LUT or power off the entire logic block (if it was entirely unused) at the (re)configuration time.

We have evaluated the proposed and baseline architectures in terms of their power, performance, and area footprint. In this regard, we used Berkeley ABC [21] to synthesize the circuits and Boolean Matcher [22] for characterization of the functions. Fully automated transistor sizing models (COFFE) [23] is exploited for efficient transistor sizing. HSPICE along with ACE 2.0 [24] activity estimator is used for subsequent delay, and power measurements. Finally, we used VTR 7.0 toolset [25] to place and route both the baseline and proposed architectures. Experimental results over a set of MCNC, IWLS, and VTR benchmarks demonstrate an average 24.5% and 16.9% reduction in total (i.e., considering both logic and routing power) static and dynamic power, respectively, compared as to the conventional 4-LUT based architecture. In addition, our proposed architecture enhances the performance by 1.8%, which leads to 21.7% improvement in *Power-Delay-Product* (PDP). **Our novel contributions with respect to the state-of-the-art work [26] are as follows:**

- We perform analytic characterization of most frequent functions (with a full examination in the proposed architecture) in comprehensive standard and industrial benchmarks such as MCNC, IWLS'05, and VTR.
- Novel power and performance efficient RHLs to be replaced with conventional 4-LUT are proposed.
- Logic SRAM sharing scheme to share SRAMs among RHLs and 3-LUT within logic block is proposed.
- A novel mapping algorithm based on the proposed RHLs which aims to minimize power against conventional 4-LUT based architectures is proposed.
- A novel RPC able to power gate unused logic resources based on the proposed mapping algorithm is suggested.
- We examine the efficiency of the proposed architecture with parameters (e.g., LUT and multiplexer structure and sizing and routing topology) that closely matches with commercial FPGAs which guarantees the effectiveness of the proposed architecture in industrial use-cases.

The rest of this paper is organized as follows. In Section II, related works are reviewed. In Section III, the methodology

of designing the proposed RHLs, RLU, and RPC is explained, and afterwards, the proposed mapping algorithm is presented. Implementation and elaboration of the proposed RLU along with experimental setup and results are detailed in Section IV. Finally, Section V concludes the paper.

II. Related Work

Several research studies investigating static power of logic and/or routing resources have been proposed in the literature. The focus of this work is to conserve logic static power, however, we also briefly review studies that attempted to reduce routing static power, as well.

Routing Static Power: The main motivation behind the method presented in [27] is originated from placement algorithm characteristic that clusters with high logic correlation being placed near together to minimize the target cost function (e.g., routing resources and/or delay). Accordingly, the authors propose to dedicate a shared power gating switch to a coarse set of clusters (including both routing and logic resources), and exploit a region-constrained placement that attempts to minimize the used cluster sets. Effectiveness of this method is limited by the ratio of unused regions.

In [28], the multiplexer and its associated buffer of routing *Switch Boxes* (SBs) are modified in such a way that they can selectively operate in three distinct operation modes, i.e., high speed (normal operation), low power for the non-critical pass SBs, and sleep mode for unused SBs. A variant sleep mode is also proposed to avoid floating nodes which could lead to high-leakage scenarios.

In [29], a multiplexer is added to each *Switch Matrix* (SM) that can select between an always-on, always-off, or a power-controlled state using dedicated configuration cells. The power-controller signal is also used for power gating the neighbouring *Configurable Logic Blocks* (CLBs) and *Connection Blocks* (CBs) during their idleness period. Efficiency of this method depends on the number of unused switch matrices and CAD tool ability to minimize the number of switch matrices that route either sporadic nets of non-adjacent CLBs or the power-controller signals themselves.

An architecture is proposed in [30] to power gate unused/under-utilized SRAMs in routing (including SBs and CBs) and logic resources. This architecture dedicates an extra configuration cell for each SB and turns off the whole SB if it was unused. In addition, this method groups the configuration cells of LUTs into smaller sets and assigns a power gating signal (and SRAM cell) to turn off the cells if the LUT was unused or the cells were *don't-care*. The latter happens when not all LUT inputs are used. An architectural assessment to choose the optimum number of grouping LUT configuration cells was also carried out in the same study.

Lastly, it is shown in [31] that different routing architectures exhibit different ratio and pattern of resource utilization, and thereby, a certain power-gating scheme with specific granularity may not be efficient in varied routing architectures. Accordingly, by investigating the utilization ratio and pattern of multiplexers within SBs, the efficient power-gating architecture corresponding to each routing architecture has been proposed.

Logic Static Power: The previous studies that aim to reduce static power in logic resources can be classified into three major categories. The first category takes advantages of using low-leakage manufacturing processes such as variable transistor gate length, triple gate oxide, and multiple-Vth in recent devices fabricated by Xilinx [8] which is employed in interconnect pass transistors and configuration memory cells. Despite of considerable static power reduction (nearly 40%), such techniques cannot be employed in entire chip resources, due to the performance loss as inherent attribute of high-threshold transistors. In addition, these techniques may not be cost-conscious due to the fabrication complexity. Nevertheless, these techniques are orthogonal to our proposed architecture (which will be discussed later) and can be used along with our proposed architecture to provide further power saving.

The second category aims to reduce static power by static or dynamic power gating of unused logic and routing resources [5], [10], [32], [33]. Static power gating is applied offline only to unused resources of the chip, while dynamic power gating is applied online during device runtime when a module is (temporarily) idle. In this regard, in [5] and [32], a reconfigurable architecture supporting static and dynamic power gating has been proposed. Frequently used modules and unused resources are configured to always-on and always-off states, respectively. Conversely, functional modules that encounter long idle periods are configured to dynamically controlled power state which allows their power supply to be controlled during run-time via signals which are generated and routed from a power controlling module. Dynamical power gating of a module is advantageous if the energy saving conquers the associated overheads, i.e., power controller energy, wake-up energy during power state transitions, and overhead of routing the controlling signal. Identifying idleness periods of modules (through either dataflow graph or application netlist) is the main issue in dynamic power gating methods. The idleness period should be large enough to throttle the mentioned overheads. Nonetheless, the application behaviour is unpredictable in interactive or input-dependent usages.

Accordingly, [33] proposes an approach for dynamic power gating of designs that are generated using high-level synthesis (HLS) process. Using the HLS scheduling information, this approach automatically detects the promising idleness periods of designs and generates the power-state controller, as well. The authors examine their approach using CHStone benchmarks which are proposed for C-based high-level synthesis. Even in applications following HLS methodology, only five (out of 12) benchmarks exhibited long idleness periods to profit from this approach. Further details such as power-on energy and time overhead and inrush current issue are neglected in this work.

In [10], an asynchronous FPGA with autonomous fine-grained power gating has been proposed. Due to using 2-input LUTs and elaborate power gating structure, area and dynamic power of the proposed FPGA significantly are increased compared with conventional (i.e., synchronous) FPGAs.

The third category has focused on using RHL that replaced or used alongside soft logic to provide area and performance efficiency [12]–[19]. These architectures, however, may impose power overhead if power is not considered as a main

concern. The FPGA architecture proposed in [18] replaces conventional LUTs with PLA-like macro-cells which are inherently power-inefficient due to their large standard cell-based structure. The authors in [19] propose a hybrid RHL-LUT FPGA architecture composed of so-called *Universal Logic Gate* (ULG) and LUT-based logic blocks in order to achieve performance and area benefits provided by efficient ULG structure. Nonetheless, choosing an appropriate ratio of LUT and ULG is critical in such architectures. Low LUT ratio can increase the FPGA array size (and thereby area) since ULG cannot cover all types of the functions. On the other hand, high LUT ratio circumscribes the architecture efficiency. In addition, effectiveness of such architectures is benchmark-dependent, and therefore, unpredictable. Toward improving power and performance efficiency, a heterogeneous architecture was proposed in [26]. In this architecture, each logic block contains three cells, including two power-efficient RHLs which can implement a considerable fraction of 4-input functions, and a 4-LUT. Both inputs and outputs of the cells are shared. Thus, no additional intra-cluster input/output multiplexer is required and also smaller power gating transistor is demanded. For each function mapped to a logic block, the priority is given first to the RHLs, and then to the 4-LUT. The other two unused cells (or alternatively the entire logic block when no function is mapped) are power gated to offset their static power. This work has neglected the inputs SRAMs overhead (needed for input negating) and suffers from exploiting inefficient 4-LUTs. In addition, the efficiency of the suggested RHLs in implementing functions in larger benchmarks was not investigated.

III. Proposed Architecture

In this section, we first present the motivation behind the proposed architecture by analyzing the inefficiency of LUT-based architectures. Then, we detail our approach in proposing the architecture by characterizing the benchmarks functions and accordingly, suggesting efficient RHLs that aim to cover the high repetition functions. Afterwards, the algorithm to map designs in the proposed architecture is elaborated. Lastly, our novel SRAM-sharing scheme to further enhance the RLU, along with the power controller circuitry is presented.

A. Motivation

The main building blocks of logic resources are LUTs that contribute to significant fraction of logic area and power consumption. A typical structure of 4-LUT is depicted in Fig. 2. Besides the configuration cells and their associated isolating buffers, tree-based structure, large input buffers (to drive large number of transistors) and demand for level-restorers indicate the power and area inefficiency of LUTs. In addition, the number of transistors grows exponentially as the number of inputs increases. Among different LUT structures, FPGAs based on 4-LUT have been reported to have minimum area, and consequently minimum configuration cell count by making a trade-off between total number of LUTs and area of each LUT [7]. However, our investigations reveal that SRAM configuration cells of 4-LUT is also often remained

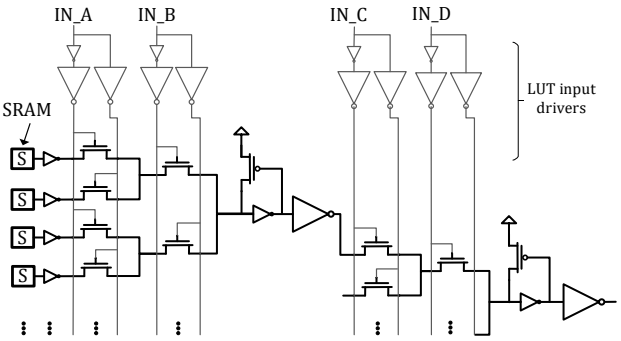

Fig. 2: 4-LUT structure [23]

underutilized for majority of functions (see Section III-B). In other words, it is not necessary to employ 16 SRAM cells to implement the majority of four (or less) input functions; instead, smaller logic structure converged with fewer number of SRAMs can implement a considerable fraction of functions [12], [16], [26].

This mismatch, besides intrinsic abundance of resources, causes additional power and area overhead in FPGAs and exacerbates the FPGA-ASIC power gap. In recent works, fine-grained heterogeneous architectures have been proposed to close this gap by using reconfigurable hard logic cells leveraging few number of SRAM cells and an efficient structure using simple logic gates instead of large pass-gates, to replace or be used in conjunction with LUTs [12], [16], [26]. However, the main target of these works is improving the performance while power efficiency has been neglected. Relying on inefficient LUT-based building blocks, and more importantly, poor assessment of larger benchmarks obscures the effectiveness of these approaches in industrial-scale applications.

B. Function Characterization

The first step in designing RHLs is to identify the most frequent functions. In this regard, we investigate a comprehensive set of standard and industrial benchmarks, i.e., MCNC, IWLS'05 and VTR. Classifying the functions is carried out based on the concept of NPN classes. Two functions belong to the same NPN-class if each function can be obtained from the other by negating and/or permuting the inputs and/or negating the output of the other function. For example, two functions $F = AB + C\bar{D}$ and $G = AC + BD$ can be obtained from each other by permuting C and B and negating D. As reported in [16], all 4-input functions (i.e., $2^{2^4} = 65536$ function) can be classified in 222 NPN-classes. Utilization rate of these functions, however, is not uniform across different circuits.

In order to identify the most frequent functions, we map the target benchmarks to 4-input functions by means of Berkeley ABC tool [21], which optimizes the logic based on *And-Inverter-Graphs* (AIGs) and can map to LUTs using optimal DAG-based technology mapping. Afterwards, the output netlist is fed to Boolean Matcher [22] to extract NPN-class of each boolean function (that is mapped to a 4-LUT). Table I summarizes the most frequent NPN-classes in the target benchmarks. We reported the NPN classes that have coverage

TABLE I: Coverage ratio (in percentage) of most frequent 4-input and 3-input *NPN* classes in MCNC, IWLS'05, and VTR benchmarks

NPN	MCNC	IWLS'05	VTR	Average
$ABCD$	37.1	29.6	35.2	34.0
$AB(C+D)$	12.1	14.9	16.2	14.4
$AB+CD$	16.1	13.7	10.8	13.5
$A(!BC+BD)$	8.0	12.3	5.6	8.6
$A(B+CD)$	8.7	5.4	6.9	7.0
$A(B+C+D)$	9.6	3.2	2.5	5.1
$ABCD$+!(AB)!CD	1.3	10.7	1.2	4.4
$AB+AC+BC$	0.2	0.4	4.1	1.6
$A(B\oplus C)$	0.1	1.7	1.8	1.2
$AB(C\oplus D)$	1.5	1.0	0.6	1.0
$A\oplus B\oplus C$	0.1	0.3	2.4	0.9
$A(B\oplus C+D)$	1.2	0.8	0.6	0.9
$AB(C+D)+CD$	0.2	0.2	1.3	0.6
$A(BC+BD+CD)$	0.2	0.1	1.2	0.5

ratio higher than 1% in at least one of the benchmark suites. Referring to Table I, minority of NPN-classes implement a significant fraction (by average, more than 93%) of 4-input functions in all benchmark suites. This observation motivates us to design RHLs which can implement a significant portion of functions, yet be an efficient alternative for conventional 4-LUT. The remaining NPN-classes have non-uniform utilization ratio in three benchmark sets. Thus, we do not consider them in designing RHLs. However, the proposed RHLs may incidentally support other NPN-classes that have not initially been targeted. The RHLs design methodology will be discussed in Section III-C.

C. RHLs Design

Designing RHLs necessitates a heuristic procedure by taking the coverage information of Table I into account. Nevertheless, the following prerequisites should be considered in proposing RHLs: (1) RHLs should be considerably area and delay efficient compared as to LUTs, (2) RHLs should be mutually exclusive as much as possible, i.e., should have minimum overlap in implementing the NPN-classes, (3) RHLs should afford input permutability and avoid input negation as much as possible (to avoid routing overhead in depopulated FPGA architectures and reduce the number of input configurable inverters), and (4) RHL with maximum number of SRAM cells determines the total LRU SRAM (due to employing LSS scheme).

In few previous works solely using RHLs [16], or in conjunction with LUT [12], [13], [17], [26] wherein if none of RHLs could implement the function, it would be mapped to the LUT. The former can incur delay and area overhead because every function which could not be implemented by RHLs will be mapped into two or more RHLs. The latter architectures can also impose area overhead due to employing large LUTs. As revealed by our experiments, majority of the functions that could not be implemented by 4-input RHLs, have three inputs and can be implemented using a 3-LUT. Thus, it is no longer necessary to use a LUT with the same number of inputs as for RHLs and instead an efficient small LUT can be exploited. Accordingly, in spite of previous studies, in our proposed architecture we exploit 3-LUTs.

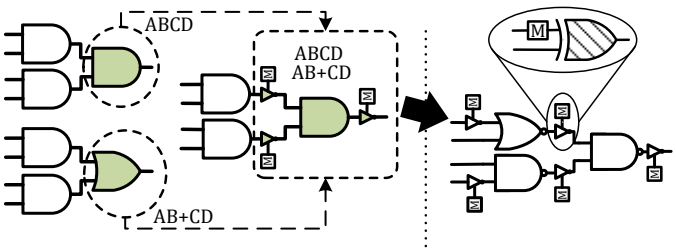

Fig. 3: The RHL1 design flow

Design steps of the first RHL, referred to as RHL1, is illustrated in Fig. 3. RHL1 is able to support three most frequent functions presented in Table I, i.e., ABCD, AB(C+D), and AB+CD, and have been primarily proposed in our original work [26], which covers 65.3% of the functions in MCNC benchmarks. Analyzing large industrial benchmarks of IWLS'05 and VTR confirms that RHL1 with five configuration bits covers majority of the functions in these benchmarks, as well (see Table I). Therefore, it is an efficient alternative for conventional 4-LUT by offering superior power, area, and performance characteristics while implementing significant fraction of functions.

To support the most frequent NPN-class (i.e., ABCD), the primitive design structure of RHL1 has been shown in Fig. 3 as three 2-input AND gates. To implement the next frequent NPN-class (i.e., AB+CD), the second structure is two 2-input AND gates, followed by a 2-input OR gate in its second level. These designs are interchangeable by merging the 2-input AND and OR gates (in the second level of both structures) into a reconfigurable AND gate which is convertible to both AND and OR. Notice that by providing negation into the inputs and output, AND and OR gates are interchangeable, i.e., $A+B=(A'\cdot B')'$. The reconfigurable inverters can act as either buffer or inverter and can be implemented by an XOR gate associated with a configuration cell.

To optimize RHL1, we simply replaced the AND gates with NAND gates (which is area/delay efficient than AND gate in static-CMOS implementation) without corrupting the functionality or supporting NPN-classes (due to the configurable inverter in the output of each gate). In addition, by replacing one of the NAND gates in the first level with a NOR gate and using only two configurable inverter in the RHL1 input, it can implement almost the same number of functions as using four configurable inverters in primary inputs and a cell with all NAND gates. In other words, NOR-NAND-NAND structure with only two configurable inverters in the inputs can implement almost the same functions as a cell with NAND-NAND-NAND structure owing four configurable inverters in the primary inputs. Analogously, we designed two other RHLs, called RHL2 and RHL3, to support the majority of remaining NPN classes. Indeed, there is a trade-off between the number of RHLs inside the RLU (i.e., the coverage ratio of the RLU) and area or power overhead. Our pre-characterizations revealed that choosing three RHLs (rather than two or four) achieves proper power efficiency while giving an efficient area.

Fig. 4 depicts the proposed RHLs designed based on the

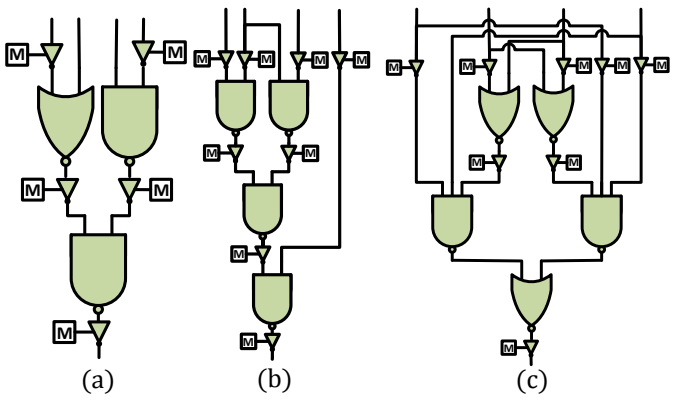

Fig. 4: Our proposed RHLs: (a) RHL1, (b) RHL2, (c) RHL3

most frequent NPN-classes represented in Table I. The coverage ratio of each RHL and 3-LUT can be derived from Table II. The proposed RHLs along with 3-LUT are able to implement all most-frequent NPNs presented in Table I[1]. Several number of other NPNs in addition to those presented in Table I is supported by RHLs and 3-LUT, however, their utilization rate is trivial; hence, they are not reported in Table I for the sake of brevity. According to Table II, there are a few NPNs with 3 or less inputs that are covered by both RHL3 and 3-LUT. In these cases, we choose RHL3 to implement such functions due to the power efficiency of RHL3 over 3-LUT which will be discussed in Section IV.

To summarize, RHL1, RHL2, and RHL3 are able to implement, on average, 61.9%, 21.9%, and 9.2% of functions, respectively. Input negation can be provided by configurable inverters in the output of the previous logic block as well. However, when the previous logic has multiple fan-outs, input negation cannot be guaranteed anymore since it might cause conflict between the other fan-out logic. It is noteworthy that the other RHL candidates have been examined to implement the remaining functions, however, considering their low utilization rate accompanied with complexity of the target RHL, it fades out the overall power gain and imposes further area overhead. Therefore, we concluded that three RHLs together with a 3-LUT leads to optimum trade-off between utilization rate and power/area efficiency.

D. Proposed Mapping Algorithm

As discussed in Section III-C, more than 95% of functions can be directly mapped to either one of the RHLs or 3-LUT. Although the proposed RHLs have small overlap in NPN coverage, for every function, the mapping priority is given to the RHL with less power dissipation. This scenario often occurs with three or less input functions that can be implemented by more than one cell. For the rest of the functions, i.e., unsupported functions, we leverage a novel mapping algorithm which eventuates to two scenarios. First, we examine whether an arbitrary 4-input function which could not be mapped to neither RHLs nor 3-LUT, can be implemented by cascading two RLUs. Otherwise, the target 4-input function will be decomposed to a pair of cofactors using Shannon expansion, i.e., $F = \bar{x_i} \cdot F(x_0, \ldots, x_{i-1}, 0, \ldots, , x_n) + x_i \cdot F(x_1, \ldots, x_{i-1}, 1, \ldots, , x_n)$. Based on this theorem, each 4-input function can be implemented using two 3-input functions (which can be implemented by 3-LUT and/or RHL) and a 2-to-1 multiplexer. To choose the most efficient function pair, the decomposition is performed on all of four variables and the pair resulting in better power efficiency is selected. For example, HL1-HL1 pair has higher priority over HL3-LUT pair. This is due to the different power, delay, and area characteristics of the proposed RHLs and 3-LUT, which is detailed in Section IV. If either of cofactors (which has three or less inputs) cannot be implemented by any RHLs, it will be mapped to 3-LUT. The required multiplexer can be implemented by all RHLs or 3-LUT; hence, we choose RHL1 due to its power efficiency. Our investigations show that 4.5% of functions can be implemented by cascading two RLUs (and choosing an appropriate cell within the RLU). The proposed mapping scheme is outlined in Algorithm 1.

[1]Except AB(C+D)+CD which has only 0.6% utilization ratio.

E. SRAM Sharing in Logic Block

The total number of configuration SRAM cells in the RLU is sum of the cells in RHLs and the 3-LUT, i.e.,

$$SRAM_{total} = \sum_{i=1}^{3} SRAM_{RHL_i} + SRAM_{LUT} = 29 \quad (1)$$

However, since in each RLU at most one of the logic cells (RHLs or 3-LUT) and its corresponding configuration SRAM cells are active in every design, a shared set of configuration SRAMs can be considered for them. We refer to this scheme as *Logic SRAM Sharing* (LSS). Therefore, applying LSS in the proposed architecture results in Equation 2.

$$SRAM_{total} = max\{SRAM_{RHL1,2,3}, SRAM_{LUT}\} = 8 \quad (2)$$

The significant reduction in the number of configuration cells (from 29 to 8) improves the power efficiency of the RLU by eliminating the leakage of power-gated SRAMs. In addition, it reduces the area of the RLU. It is also noteworthy that while up to eight SRAM cells can be used in RHL1, we just exploit five configuration cells within its structure and removed

TABLE II: Coverage ratio of the proposed RHLs

NPN	Avg. ratio	RHL1	RHL2	RHL3	3-LUT
$ABCD$	34.0	✓			
$AB(C+D)$	14.4	✓			
$AB+CD$	13.5	✓			
$A(!BC+BD)$	8.6		✓		
$A(B+CD)$	7.0		✓		
$A(B+C+D)$	5.1		✓		
$ABCD$+!(AB)!CD	4.4			✓	
$AB+AC+BC$	1.6				✓
$A(B \oplus C)$	1.2		✓		✓
$AB(C \oplus D)$	1.0			✓	
$A \oplus B \oplus C$	0.9				✓
$A(B \oplus C+D)$	0.9			✓	
$A(BC+BD+CD)$	0.6			✓	
$!A!BC+ABC$	0.5			✓	✓
$AB(C+D)$+$!A!B!C!D$	1.6 (shared)			✓	
ABC+$!B!C$				✓	✓
$(A \oplus B)+CD$				✓	
$A(B!CD$+$!BC!D)$				✓	
$ABCD$+$!A!B!C!D$				✓	
$!A!BCD+(A \oplus B)!C!D$				✓	
ABC+$!A!B!C+BC!D$+$!B!CD$				✓	
Other supported NPNs				✓	
Unsupported NPNs	4.5				

Algorithm 1: Proposed algorithm to map a design into RLUs

Input: $Bool_{list}$: Input design boolean functions
Input: NPN_{RHL1}: NPNs supported by RHL1
Input: NPN_{RHL2}: NPNs supported by RHL2
Input: NPN_{RHL3}: NPNs supported by RHL3
Output: RLU_{list}: Output list, determines corresponding RHL/LUT cell(s) for each function

for *each Boolean* $b \in Bool_{list}$ **do**
  $cell_b \leftarrow mapRLU(b)$;
  if $cell_b \neq false$ **then**
    $RLU_{list}(b) \leftarrow cell_b$;
  else
    $RHLset_b \leftarrow ABC(b, NPN_{RHL1} \cup NPN_{RHL2} \cup NPN_{RHL3})$;
    // Synthesizing b into RHLs by ABC using RHLs library
    if $RHLset_b.size() = 2$ **then**
      $RLU_{list}(b) \leftarrow RHLset_b$;
    else
      //Decompose b into cofactors
      $CFset_b \leftarrow ShannonExpansion(b)$;
      $MINpair_b \leftarrow min\{CFset_b\}$;
      $RLU_{list}(b) \leftarrow mapRLU(MINpair_b.get(0)) \cup mapRLU(MINpair_b.get(1)) \cup RHL1$;

Function *mapRLU (Boolean b)*
  $NPN_b \leftarrow BooleanMatcher(b)$;
  if $NPN_b \in NPN_{RHL1}$ **then**
    return RHL1;
  if $NPN_b \in NPN_{RHL2}$ **then**
    return RHL2;
  if $NPN_b \in NPN_{RHL3}$ **then**
    return RHL3;
  if $inputCount(b) < 4$ **then**
    return 3-LUT;
  return $false$;

two configurable inverters from its inputs. It is because they show negligible improvement in function coverage of RHL1 which will be offset by the area of the two configurable inverters (XOR gates). In addition, as it will be demonstrated in Section III-F, by removing two configurable inverters and using only five SRAMs within RHL1 (rather than eight), three of the shared configuration cells can be powered off if RLU is configured to RHL1.

F. Reconfigurable Power Controller

Fig. 5 illustrates the overall structure of the proposed RLU along with the proposed RPC. Power gating of each RHL or 3-LUT is performed by inserting a pair of pMOS and nMOS sleep transistors between the power supply and the cell supply node[2]. Four configurable SRAM cells, i.e., S1-S4 which we refer to as power gating SRAMs, are dedicated to determine the power state of both RHLs/LUT cells and

[2]In Fig. 5, only the pMOS sleep transistors are shown for the sake of simplicity.

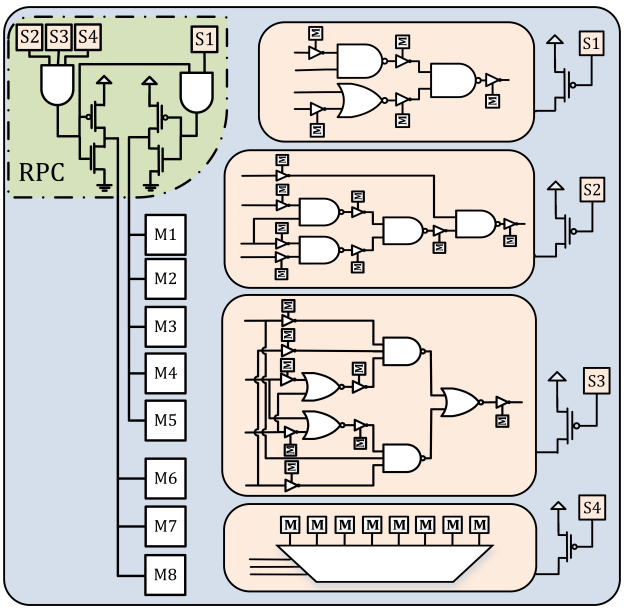

Fig. 5: Overall structure of the proposed RLU and RPC

eight shared configuration SRAMs. Logical 0 when power gating an SRAM indicates that the associated cell is ON. For example, $S3 = 0$ or $S4 = 0$ implies that RHL3 or 3-LUT is active inside an RLU, respectively. In such cases, the power gating pMOS transistor of a logic cell is ON and delivers strong Vdd to the corresponding logic cell. In addition, when RHL1 is used within an RLU, since it uses only five configuration cells (i.e., M1-M5), other three SRAMs can be power gated. Therefore, if none of RHL2, RHL3, or 3-LUT was used ($S2 = S3 = S4 = 1$), then M6-M8 can be power gated. Analogously, when another logic cell was used, e.g., RHL2, then $S2 = 0$ produces a logical 0 in the output of both AND gates (see Fig. 5), which subsequently turns on all the configuration cells (notice that RHL2 requires all of the eight cells). Finally, if the entire RLU was not used, then the output of both AND gates (see Fig. 5) is high, which turns off the power supply of all the configuration cells (while the logic cells are power gated as well).

IV. Experimental Results

In this section, first we explain the implementation and evaluation flow. Afterwards, experimental setup and parameters used for evaluating the baseline architectures (i.e., no power gating and fine-grained power gating based on typical pure LUT) and the proposed architecture are described. We elaborate a detailed comparison among baseline architectures including traditional 4-LUT, 6-LUT, a similar work in [26], and *Fine-Grained Power Gating* (FGPG). The FGPG scheme applies power gating with single logic block (considering pure LUT architecture) granularity that has been suggested to have better results over coarse-grained power gating in FPGAs [34]. Both [26] and [34] are most relevant studies to our proposed architecture which exploit the *static power gating* in *logic resources*.

Overall experimental flow is illustrated in Fig. 6. As shown in this figure, we first synthesize HDL description of the

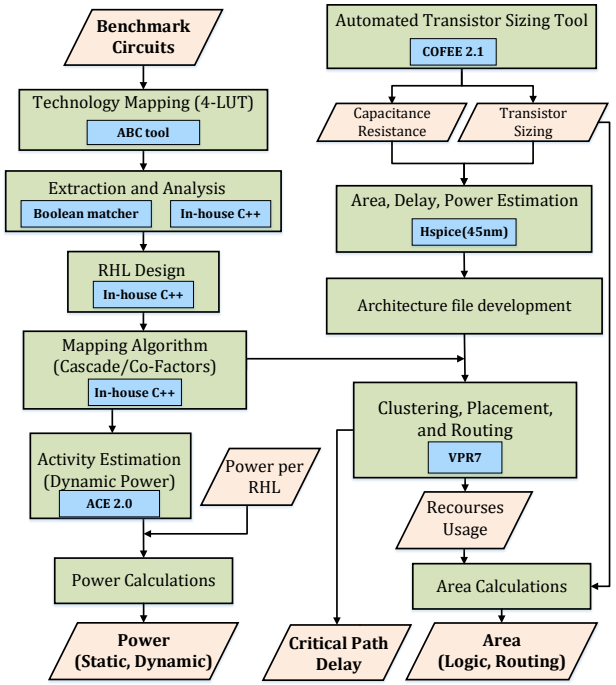

Fig. 6: Overall implementations flow

benchmark circuits into 4-LUTs using Berkeley ABC logic synthesis and optimization tool [21]. Using an in-house C script, boolean truth-table of each LUT is obtained and fed into Boolean Matcher [22] to generate the corresponding NPN class. According to generated NPNs, the proposed RHLs are designed, as detailed in Section III-C. Afterwards, by exploiting Algorithm 1, the initial BLIF netlist is modified to replace 4-LUTs with RHLs/3-LUT cells or a cascade of two or more cells (in the case of using Shannon expansion). To evaluate the architectures in terms of delay, area, and performance, we exploit COFFE [23] to automatically generate the efficient transistor sizing for both 4-LUT and 6-LUT architectures (including the transistor sizes for SB and CBs with different number of inputs). HSPICE simulations using 45nm high performance *Predictive Technology Model* (PTM) [35] is performed for delay and power estimation. Delay and area parameters are wrapped in VPR [25] architecture description files to obtain the corresponding parameters for each benchmark after place and routing steps. Finally, we post-process the VPR-generated placement and routing information to estimate the static and dynamic power for each benchmark. For dynamic power estimation, average activity factor of each benchmark is calculated using ACE activity estimator tool [24].

A. Experimental Setup and Parameters

1) Parameters: Table III summarizes the delay, power, and area characteristics of the proposed designs and typical 4-LUT and 6-LUT. In addition, characteristics of the multiplexers (denoted by MUX which includes both CB and intra-cluster multiplexers) and SB are reported in this table.

Delay: The delay of each logic cell (RHLs and LUTs) is the average delay of its different inputs. Since VPR estimates the CB delay internally (because its size is not determined before routing stage), it is not included in the table. However, the delay of the intra-cluster multiplexers (which have the same size of 16 in all architectures) is $40ps$. In addition, the size of an SB is independent from channel width and can be obtained from Equation 3 (the parameters are defined in Table V).

$$X_{SB} = N \cdot F_{cout} \cdot \frac{L}{2} + L \cdot (F_s - 1) + 1 \tag{3}$$

Taking into account the fact that F_{cout} has minor impact in efficiency of an FPGA architecture, it is usually considered as $\frac{1}{N}$ [36]. Thus, regardless of value of the N, SB size can be rewritten as Equation 4:

$$X_{SB} = \frac{L}{2} + L \cdot (F_s - 1) + 1 \tag{4}$$

Using this equation, $X_{SB} = 11$ for constant $L = 4$ and $F_s = 3$ in all architectures. Therefore, similar SB parameters have been used for different architectures. It is noteworthy that two-stage multiplexers (that has been shown to afford best area-delay efficiency [23]) have been used for all multiplexer structures except for LUTs (i.e., CB, SB, and intra-cluster multiplexers) for which the same structure shown in Fig. 2 has been used.

Static Power: The second column in Table III reports the static power of different designs, obtained by transistor-level HSPICE simulations under the temperature of $65°C$. The static power of a CB or intra-cluster multiplexer owing X inputs can be obtained by $110 \times X^{0.42}$ (in nW) which is obtained by interpolating all multiplexers from 8 to 64 inputs. However, SB multiplexer has larger sizing and consumes higher power, so it is reported separately. Notice that, in the proposed RLU, when a cell is active, static power of the other power gated cells has been also considered.
Power of the cells is also reported in their power gated state. When the proposed RLU is entirely power gated, it consumes $192nW$, however, the 4-LUT dissipates $418nW$ in power gated state due to its large input drivers and configuration cells.

Dynamic Power: The dynamic power of the designs is calculated assuming a switching input frequency of 100MHz with switching probability $\alpha = 1$. Later, we scale the dynamic powers according to $P \propto \alpha \cdot f$ using actual activity and frequency of each benchmark. Load capacitance of each design has been set equal to input capacitance of subsequent module. Thus, power of SB is considerably higher than equivalent 11-input multiplexer due to the larger transistors of the SB

TABLE III: Proposed and Baseline Cells Characteristics

Design	Delay (ps)	Static Power ON state (nW)	Static Power OFF state (nW)	Dynamic Power (nW)	Config. Cell	Area (min width transistors)
RHL1	78	406	192	242	5	72
RHL2	105	601		308	8	114
RHL3	98	801		386	8	120
3-LUT	123	746		1098	8	126
4-LUT	172	2388	418	1639	16	264
6-LUT	207	5769	845	3212	64	940
MUX	-	$110X^{0.42}$	-	$174X^{0.22}$	$\lceil\sqrt{X}\rceil + \lceil\frac{X}{\lceil\sqrt{X}\rceil}\rceil$	$15X^{0.6}$
SB	49	429	-	767	7	71.5

TABLE IV: The proposed and baseline architectures parameters

Parameter	PEAF	4-LUT (Normal, PG)	6-LUT (Normal, PG)	[26]
N	10	10	10	10
K	4	4	6	4
I	25	25	36	25
F_{cin}	0.2	0.3	0.3	0.2
F_{cout}	0.1	0.2	0.2	0.1
Intra-cluter multiplexer	16	16	16	16
F_{fb}	$5/16$	$5/16$	$5/16$	$5/16$
F_{blb}	$0, 1/5, 1/9$	$0, 1/5, 1/9$	$0, 1/5, 1/9$	$0, 1/5, 1/9$

TABLE V: Explanation of the architectural parameters

Parameter	Definition
N	Cluster (CLB) size
K	LUT size
I	Cluster inputs (from adjacent routing channels)
F_{cin}	CB connectivity factor (determines number of channel tracks connected to each CB)
F_{cout}	Cluster output connectivity factor (determines number of SBs a logic block output is connected to)
Intra-cluter multiplexer (X)	LUT input multiplexer size
F_{fb}	Cluster feedback factor (determines no. of other logic blocks output in LUT input multiplexer)
F_{blb}	Repetition factor of black-box modules
L	Routing wire segment length
F_s	Number of output branches of a routing wire, when it intersects an SB

(thereby larger parasitics capacitance) and the large wire capacitance that it drives.

Area Model: The area of all designs are reported as *minimum width transistor* model [23], formulated by Equation 5:

$$Area(x) = 0.447 + 0.128x + 0.391\sqrt{x} \qquad (5)$$

in which x is the size (strength) of a single transistor. The area of an SRAM cell is considered as 6.0.

2) General Setup: Similar to commercial FPGAs such as Xilinx Virtex IV [37] and Altera Stratix II [38], we leveraged conventional island-style architecture supposed in VPR tool. Architectural parameters of the baseline and proposed architecture are summarized in Table IV. It should be noted that heterogeneous black-box modules such as single/dual port RAMs and multipliers in VTR benchmarks needs specifying the repeat frequency of black-box columns (i.e., F_{blb}) in the FPGA architecture. This ratio is different among varied commercial FPGAs. By trying comprehensive examinations over the VTR benchmarks, we realized that $F_{blb} = 5$ results in the best area efficiency when there are large number of such modules[3]. On the other hand, $F_{blb} = 12$ is optimum when repetition frequency of black-boxes is low. Our results is in concordance with [39] which suggests $F_{blb} = 5$ for area efficiency. Since the contribution of this paper does not involve hard-wired modules such as multipliers and RAMs, we do not consider these modules in our area and power estimations. However, since these modules are placed and routed in our experiments, they will affect the placement and routing the corresponding benchmarks, which has been considered in the experiments.

[3]It means that, per every five CLB columns there is one black-box module column.

B. Critical Path Delay

In order to compare the delay of PEAF and the baseline, we implemented all the benchmarks using VPR with the delay and architectural parameters reported in Table III and Table IV. Fig. 7 compares the logic and routing delay of different architectures in MCNC, IWLS, and VTR benchmarks[4]. Logic and routing delays in this figure are separately normalized to 4-LUT (with no power gate scheme). Compared with the baseline 4-LUT architecture, PEAF reduces the logic delay by 25.6%, 30.3%, and 4.6% in MCNC, IWLS, and VTR benchmarks, respectively, which eventuates to average logic delay reduction by 21.3%. Notice that, according to Table III, the average delay of RLU (i.e., average delay of comprising cells) is reduced by 41.3% with respect to 4-LUT delay. This improvement, however, is diminished to 21.3% in the benchmarks due to the impact of the intra-cluster multiplexer delay (which is equal in all architectures). In addition, shrinking the logic that cannot be implemented with a single RLU (detailed in Section III-D) could increase the number of logic blocks within circuit critical path.

Since our proposed RHLs do not support full input permutation (only inputs of gates in the first-stage can be swapped), PEAF imposes routing overhead. In addition, shrinking some of the functions (as mentioned in the above paragraph) results in more SBs for routing the nets, thereby higher routing delay. Thus, routing delay is increased, as expected, by 8.3%, 14.8%, and 27.6% in MCNC, IWLS, and VTR benchmarks, respectively. Altogether, taking into account both the logic and routing delays, the total delay is improved by 1.8%, on average, considering all benchmarks suites (8.2% improvement is achieved if only MCNC and IWLS benchmarks are considered). As expected, [26] exhibits better performance with respect to PEAF since it does not shrinks functions due to employing 4-LUT within its so-called Mega Cell. Conversely, FGPG 4-LUT architecture[5] increases the logic and total delay with respect to the baseline 4-LUT by 7.5% and 2.5%,

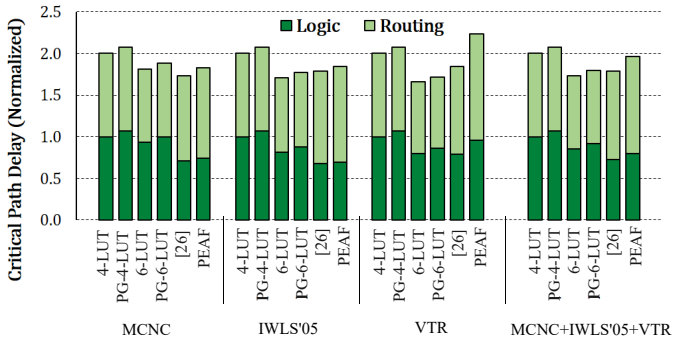

Fig. 7: Delay comparison of different architectures over MCNC, IWLS'05, and VTR benchmark suites

[4]All experiments (delay, area, and power) have been performed for all circuits of MCNC, IWLS'05, and VTR benchmark suites but for the sake of brevity, we only report the detailed results for MCNC and the average results of these three benchmark suites.

[5]In all of the figures, FGPG architectures are denoted by PG-4-LUT and PG-6-LUT.

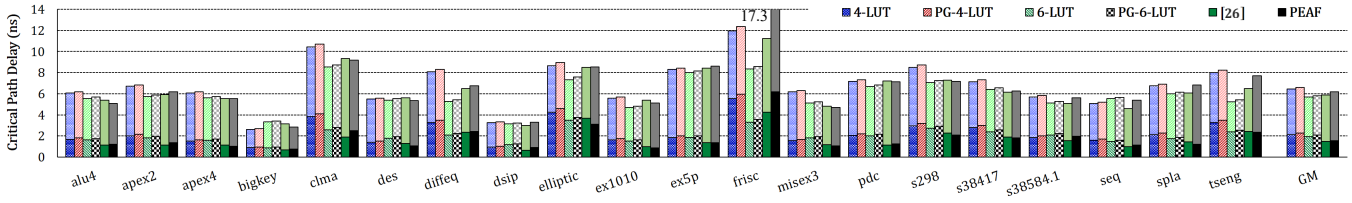

Fig. 8: Comparing the delay of different architectures in MCNC benchmarks (bottom and top half of each bar corresponds to logic and routing delay, respectively.)

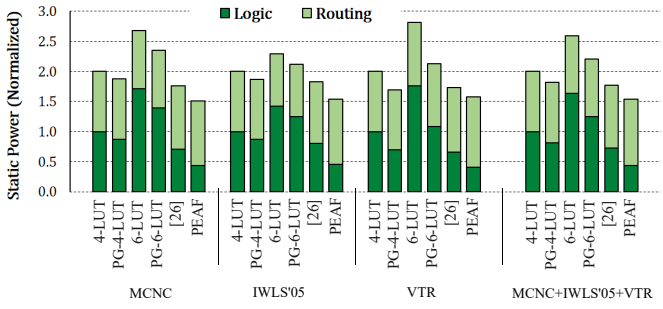

Fig. 9: Comparing the static power of different architectures over MCNC, IWLS'05, and VTR benchmark suites

respectively. This arises from delay overhead of employing power gating[6]. It should be noted that 6-LUT inherently is the most delay-efficient architecture with worst area [7]. Hence, direct comparison of 6-LUT with less input architectures is not fair. Therefore, we compare the holistic PDP parameter of this architecture with the counterparts. Detail results of delays for different architectures implementing MCNC benchmarks are represented in Fig. 8.

C. Static Power

The static power of benchmarks is obtained by post-processing the reports generated by VPR, e.g., array size and channel width, and obtaining the number of CLBs, CBs (with different sizes) and SBs[7]. Afterwards, the numbers reported in Table III are used to estimate the static power of each architecture. Fig. 9 represents the static power of different architectures implementing various circuits. Detailed information of MCNC benchmark suite is shown in Fig. 10. According to Fig. 9, PEAF reduces logic power in MCNC, IWLS, and VTR benchmarks by 56.3%, 54.4%, and 59.1%, respectively, with respect to 4-LUT architectures (56.6% on average considering all benchmarks). This is mainly due to using small number of SRAMs by exploiting LSS scheme and replacing the 4-LUT with smaller 3-LUT. On the other hand, FGPG 4-LUT and the previous work in [26] result in only 18.8% and 27.7% average logic power saving. Therefore, PEAF increases the logic power gating efficiency by approximately 3x and 2x compared as to FGPG 4-LUT architecture and the work presented in [26].

[6]We set the power gating transistor size such that it yields the best PDP. A 67.5X power gating transistor for the entire logic block imposes 7.5% delay overhead but achieves minimum PDP.

[7]CB multiplexer size can be obtained by $F_{cin} \times W$.

As discussed in Section IV-B, PEAF may impose routing overhead due to using more logic blocks. Our experiments revealed that routing static power is increased by 7.6%, 7.8%, and 17.2% in MCNC, IWLS, and VTR benchmarks, respectively. As a result, taking both logic power reduction and routing power increase into account, PEAF improves the total static power by 19.0%, 19.2%, and 20.0% in MCNC, IWLS, and VTR benchmarks, respectively. This results in average 19.4% improvement over all benchmark suites. FGPG 4-LUT architecture keeps the routing power intact, therefore, the total average power improvement of this architecture is 10.1%, taking both logic and routing static power into consideration. Therefore, PEAF still increases the power gating efficiency as 2.4x with respect to FGPG.

D. Dynamic Power

As for the static power, dynamic power of different architecture is estimated by post-processing the VPR reports including number of CLBs and SBs, and number and size of CBs. Afterwards, parameters in Table III are used to obtain the dynamic power for each benchmark. Power numbers in Table III have obtained for frequency of 100MHz with $\alpha = 1$. Thus, actual dynamic power of each benchmark is obtained by Equation 6.

$$P_{dyn} = P_{dyn,base} \cdot \frac{10^{-8}}{T_{total}} \cdot \alpha \tag{6}$$

In this equation, $P_{dyn,base}$ is dynamic power of the circuit with base parameters, i.e., 100MHz frequency (10ns delay) and activity factor equal to 1, T_{total} is circuit total delay, and α is actual activity factor obtained by exploiting ACE tool. Fig. 13 reports the dynamic power of different architectures over all benchmark suites. In addition to the activity factor obtained by ACE which depends on activity of the primary inputs, we report the dynamic power considering constant $\alpha = 0.125$ for all benchmarks. Detailed results of the dynamic power for the MCNC benchmarks are reported in Fig. 11 and Fig. 12 for $\alpha = 0.125$ and for α reported by ACE tool, respectively.

Using the accurate activity factor reported by ACE, our proposed architecture reduces dynamic power by 6.3%, 14.1%, and 11.2% in MCNC, IWLS, and VTR benchmarks, respectively (10.4%, on average). On the other hand, assuming a constant $\alpha = 0.125$ results in 7.3% average improvement in dynamic power. Compared with 6-LUT architecture, PEAF reduces the dynamic power by 39.7%. The substantially higher dynamic power of 6-LUT architecture is mainly attributed to

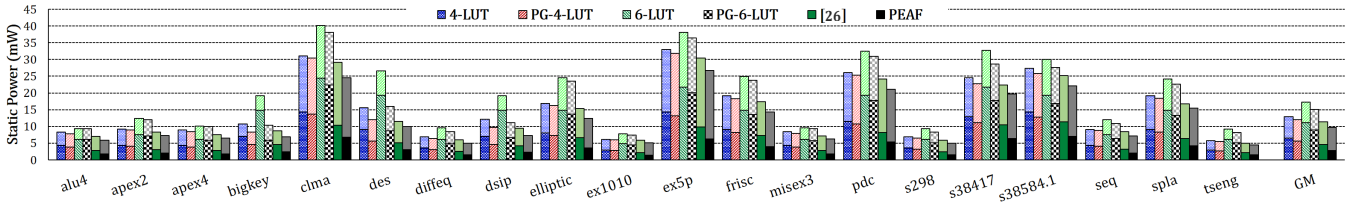

Fig. 10: Static power comparison of different architectures in MCNC benchmarks (bottom and top half of each bar corresponds to logic and routing power, respectively.)

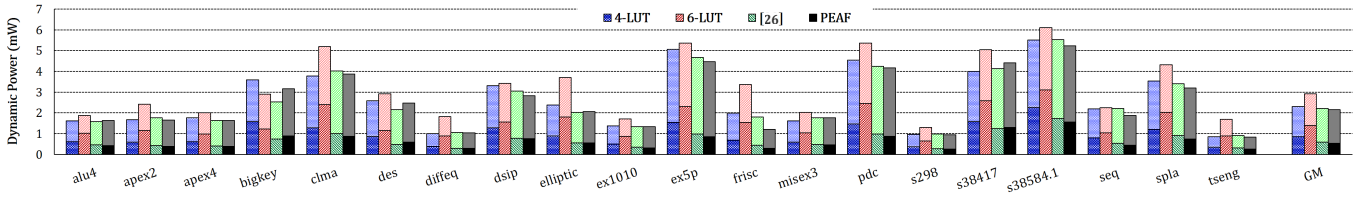

Fig. 11: Comparing the dynamic power of different architectures in MCNC benchmarks assuming constant $\alpha = 0.125$ (bottom and top half of each bar corresponds to logic and routing power, respectively.)

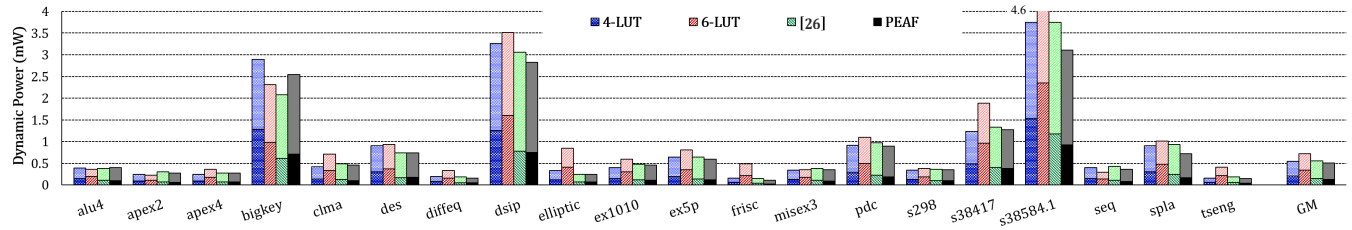

Fig. 12: Comparing the dynamic power of different architectures in MCNC benchmarks, using the α reported by ACE (bottom and top half of each bar corresponds to logic and routing power, respectively.)

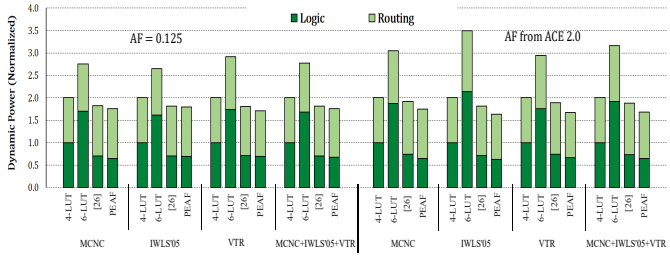

Fig. 13: Dynamic power comparison of different architectures over MCNC, IWLS'05, and VTR benchmark suites

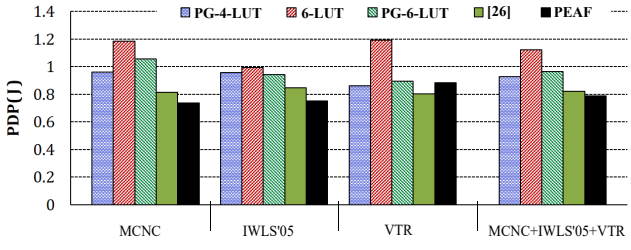

Fig. 14: PDP comparison of different architectures over MCNC, IWLS'05, and VTR benchmark suites

the tree-structure of 6-LUT multiplexer with large buffers and pass-gates having large parasitics capacitances. In addition, PEAF improves the dynamic power by 9.9% with respect to [26]. This improvement is due to exploiting 4-LUT in the logic block of [26] that consumes large dynamic power compared with our proposed RHLs.

E. Power-Delay Product

Fig. 14 compares the PDP of all architectures implementing various benchmark suites. PDP is obtained by multiplying the total delay in total power (static and dynamic) of each benchmark. As compared to the baseline 4-LUT, PEAF improves the PDP by 27.6%, 25.6%, and 12.0% for MCNC, IWLS, and VTR benchmarks, respectively, resulting in average of 21.7% over all benchmark circuits. Compared with [26], our proposed architecture enhances the PDP by 10.6% and 12.4% in MCNC and IWLS benchmarks, respectively. In the VTR benchmarks, however, [26] surpasses our proposed architecture. Finally, while 6-LUT architecture improves both the logic and routing delay (due to the intrinsic characteristics of 6-LUT that uses less logic resources, thereby less routing resources, as well), due to its substantial static/dynamic power consumption, PEAF improves the PDP by 31.5% over 6-LUT structure. Therefore, PEAF is a promising candidate for state-of-the-art 6-LUT architectures wherein overall energy consumption is an important factor.

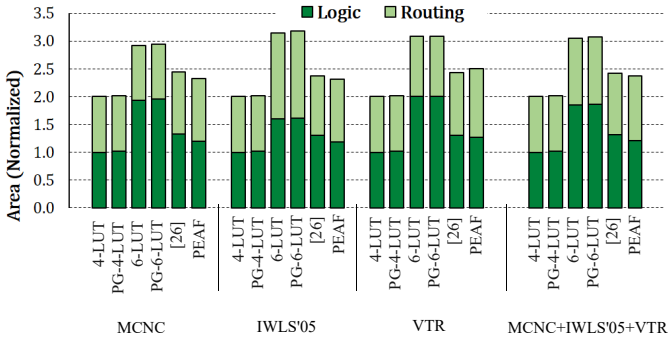

Fig. 15: Area comparison of different architectures over MCNC, IWLS'05, and VTR benchmark suites

F. Area

Unlike Section IV-B in which direct results of the VPR tool is reported as logic and routing delay of benchmarks, for estimating the area we post-processed the logic and routing information of the benchmarks (after place and route steps) including number of SBs and CLBs, and number and size of CBs using an in-house C script. It is because VPR follows an older area modelling that does not well match for newer technology sizes. It should be noted that VPR maps each benchmark to minimum-size FPGA, and then iteratively finds the minimum channel width necessary to route the design. In addition, we have used COFFE for transistor sizing which gives optimum sizing for given input technology file and architectural parameters (such as L, W, etc.) which results in different SB and CB transistor sizing with VPR defaults. Fig. 15 compares the area of different FPGA architectures for all the benchmark suites. Detailed information of MCNC benchmark suite is shown in Fig. 16.

PEAF increases the logic area by 21.3% compared with the baseline 4-LUT. Based on the parameters of Table III and taking into account the area of flip-flop and output multiplexer inside every logic block, the area of a logic block that consists of the proposed RLU is only 11% larger than the baseline 4-LUT based logic block. However, increasing the number of logic blocks due to shrinking of some functions into two or three logic blocks, the overall logic area overhead increases to 21.3%. Nonetheless, the previous work [26] increases the logic area by 31.3%. The area saving of the proposed architecture is achieved due to employing LSS scheme and exploiting small 3-LUT. Taking into account routing area increase of PEAF (by 16.5%) due to lacking input full permutation[8] and shrinking logic which demands further routing resources, the total area of the proposed architecture has been increased by 18.9% compared with 4-LUT. However, the area of PEAF is still 18.1% less than 6-LUT architecture. It is noteworthy that due to using large power gating transistors, FGPG imposes 3% area overhead per logic block, hence, its overall logic area also increases by 3%.

[8]It should be noted that routing area of [26] is also increased due to the same reason.

G. Commercial FPGAs

As mentioned in Section IV-F, VPR maps each benchmark to minimum-size logic array and routing channel width. However, commercial FPGA devices have pre-determined (fixed) number of logic and routing resources. To evaluate the efficiency of PEAF, we modified the VPR such that it maps each benchmark to smallest Virtex II device. These devices range from 8x8 to 104x112 array of CLBs with $N = 8$. The other parameters, I, W, intra-cluster multiplexer size, F_{cin}, and F_{cout}, are set to 20, 196, 14, 0.2, and 0.125, respectively, by exploring the Virtex II architecture using Xilinx FPGA-Editor and also from [40].

Fig. 17 compares the static power of PEAF with other architectures. Logic static power has been significantly reduced by 72.1%, 68.1%, and 74.8% in MCNC, IWLS, and VTR benchmarks, respectively, compared to 4-LUT architecture. This means an average of 72.6% static power reduction considering all benchmarks, which is higher than the former 56.6% when VPR maps each benchmark to minimum possible array. In overall, as demonstrated in Fig. 18, the PDP of PEAF is reduced by 37.9%, 37.1%, and 35.4% in MCNC, IWLS, and VTR benchmarks, respectively (36.9% on average, while it was 21.7% for VPR default mapping).

H. Limitations and Summary

We encountered some limitations with VPR tool which we believe resolving them could further enhance the delay and area efficiency of the proposed architecture. First, VPR does not support function-based permutation. For example, in a function such as $A(B + C + D)$, variables B, C, and D can be permuted, which reclaims the pressure on routing network. Notice that, based on our investigation, a significant portion of PEAF's routing area overhead (includes 10.5% of total 16.5% routing overhead) arises from input non-permutablity (the remaining 6% is due to routing cascaded logic). In addition, in some of the benchmarks, VPR is unable to detect the RHL-latch pairs (although we used its pack-pattern attribute to resolve this issue). Thus, when a latch follows an RHL, VPR separates them into two different logic blocks (one for RHL and the other for the latch), which imposes both area and delay overhead. This problem, however, does not occur

TABLE VI: Summary of the results across all benchmark suites. All parameters are normalized to baseline 4-LUT architecture.

Parameter/Architecture	PG 4-LUT	6-LUT	PG 6-LUT	[26]	PEAF
Logic Delay	1.07	0.85	0.91	0.73	0.80
Routing Delay	1.00	0.88	0.88	1.06	1.17
Total Delay	1.03	0.86	0.87	0.89	0.98
Logic Static Power	0.81	1.64	1.25	0.72	0.43
Routing Static Power	1.00	0.96	0.96	1.05	1.11
Total Static Power	0.90	1.31	1.09	0.88	0.81
Logic Dynamic Power	1.00	1.92	1.92	0.73	0.64
Routing Dynamic Power	1.00	1.23	1.23	1.15	1.04
Total Dynamic Power	1.00	1.49	1.49	0.99	0.90
Logic Total Power	0.82	1.64	1.26	0.72	0.44
Routing Total Power	1.00	0.97	0.97	1.05	1.10
Total Power	0.90	1.31	1.11	0.88	0.81
Logic Area	1.02	1.85	1.87	1.31	1.21
Routing Area	1.00	1.20	1.20	1.11	1.17
Total Area	1.01	1.40	1.41	1.21	1.19
PDP	0.92	1.13	0.96	0.82	0.78

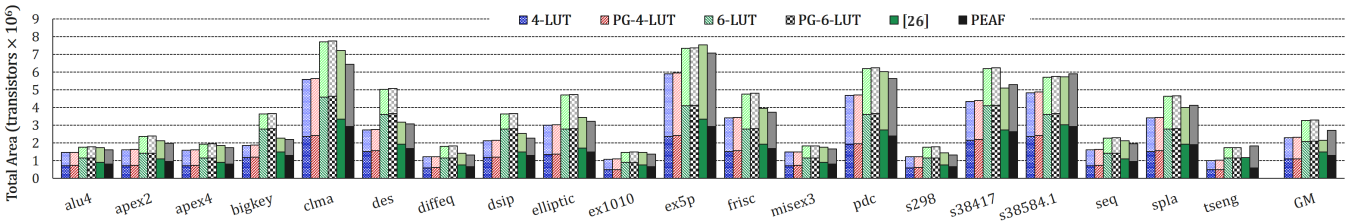

Fig. 16: Area comparison between the proposed and baseline architectures (bottom and top portions are logic and routing area, respectively.)

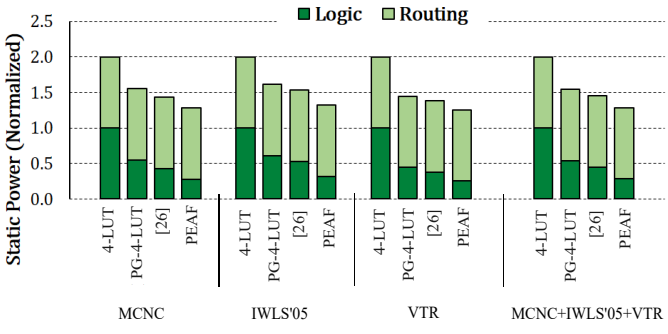

Fig. 17: Static power comparison of different architectures with logic/routing resources similar to commercial FPGAs over MCNC, IWLS'05, and VTR benchmark suites

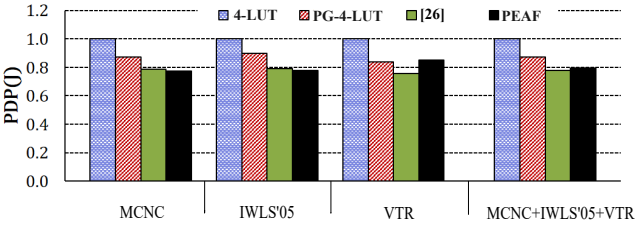

Fig. 18: PDP comparison of different architectures with logic/routing resources similar to commercial FPGAs over MCNC, IWLS'05, and VTR benchmark suites

for LUT-based architecture since VPR perfectly identifies the LUT-latch pairs. It should also be noted that VPR could not route *LU32PEEng* in VTR benchmarks after 20 days[9]. Thus, we omitted this application from our evaluations. Summary of the experiments results is presented in Table VI.

V. Conclusion

In this paper, we proposed a power-efficient architecture, called PEAF, to reduce the power efficiency of SRAM-based FPGAs while preserving the performance efficiency in dark silicon era. PEAF comprises a set of RHLs, a 3-LUT, and a reconfigurable power controller, called RPC. Three RHLs are designed in such a way that together with a 3-LUT can implement more than 95% of the functions. Furthermore, we proposed a novel scheme to share intra logic block SRAMs to alleviate the area overhead of the proposed architecture where we share eight SRAMs among RHLs/3-LUT within each logic block. PEAF reduces both the static power of unused logic as well as the used logic. The comparison of the proposed architecture (PEAF) and the equivalent 4-LUT based FPGAs demonstrates that total performance, static power, and the PDP are improved by 1.8%, 24.5%, and 21.7% respectively with the cost of 18.9% overhead in the total area.

[9]Using Intel Xeon CPU E5-2690 v2 @ 3.00GHz workstation with 132GB RAM.

References

[1] I. Kuon and J. Rose, "Measuring the gap between fpgas and asics," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 26, no. 2, pp. 203–215, 2007.

[2] B. Zahiri, "Structured asics: opportunities and challenges," in *Computer Design, 2003. Proceedings. 21st International Conference on*. IEEE, 2003, pp. 404–409.

[3] (2016 (accessed May 3, 2016)) Xilinx 7 series fpgas. [Online]. Available: http://www.xilinx.com/publications/prod_mktg/7-Series-Product-Brief.pdf

[4] A. Amara, F. Amiel, and T. Ea, "Fpga vs. asic for low power applications," *Microelectronics Journal*, vol. 37, no. 8, pp. 669–677, 2006.

[5] A. A. Bsoul and S. J. Wilton, "An fpga architecture supporting dynamically controlled power gating," in *Field-Programmable Technology (FPT), 2010 International Conference on*. IEEE, 2010, pp. 1–8.

[6] F. Firouzi, S. Kiamehr, and M. B. Tahoori, "Statistical analysis of bti in the presence of process-induced voltage and temperature variations," in *Design Automation Conference (ASP-DAC), 2013 18th Asia and South Pacific*. IEEE, 2013, pp. 594–600.

[7] E. Ahmed and J. Rose, "The effect of lut and cluster size on deep-submicron fpga performance and density," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 12, no. 3, pp. 288–298, 2004.

[8] M. Klein, "Wp298: Power consumption at 40 and 45 nm," 2009.

[9] R. Ahmed, S. J. Wilton, P. Hallschmid, and R. Klukas, "Hierarchical dynamic power-gating in fpgas," in *Applied Reconfigurable Computing*. Springer, 2015, pp. 27–38.

[10] S. Ishihara, M. Hariyama, and M. Kameyama, "A low-power fpga based on autonomous fine-grain power gating," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 19, no. 8, pp. 1394–1406, 2011.

[11] A. A. Bsoul and S. J. Wilton, "A configurable architecture to limit wakeup current in dynamically-controlled power-gated fpgas," in *Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays*. ACM, 2012, pp. 245–254.

[12] I. Ahmadpour, B. Khaleghi, and H. Asadi, "An efficient reconfigurable architecture by characterizing most frequent logic functions," in *Field Programmable Logic and Applications (FPL), 2015 25th International Conference on*. IEEE, 2015, pp. 1–6.

[13] J. H. Anderson and Q. Wang, "Area-efficient fpga logic elements: architecture and synthesis," in *Proceedings of the 16th Asia and South Pacific Design Automation Conference*. IEEE Press, 2011, pp. 369–375.

[14] P. Jamieson and J. Rose, "Enhancing the area-efficiency of fpgas with hard circuits using shadow clusters," in *Field Programmable Technology, 2006. FPT 2006. IEEE International Conference on*. IEEE, 2006, pp. 1–8.

[15] H. Parandeh-Afshar, H. Benbihi, D. Novo, and P. Ienne, "Rethinking fpgas: elude the flexibility excess of luts with and-inverter cones," in *Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays*. ACM, 2012, pp. 119–128.

[16] Y. Okamoto, Y. Ichinomiya, M. Amagasaki, M. Iida, and T. Sueyoshi, "Cogre: A configuration memory reduced reconfigurable logic cell architecture for area minimization," in *Field Programmable Logic and Applications (FPL), 2010 20th International Conference on*. IEEE, 2010, pp. 304–309.

[17] Y. Hu, S. Das, S. Trimberger, and L. He, "Design and synthesis of programmable logic block with mixed lut and macrogate," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 28, no. 4, pp. 591–595, 2009.

[18] J. Cong, H. Huang, and X. Yuan, "Technology mapping and architecture evalution for k/m-macrocell-based fpgas," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 10, no. 1, pp. 3–23, 2005.

[19] T. Luo, H. Liang, W. Zhang, B. He, and D. Maskell, "A hybrid logic block architecture in fpga for holistic efficiency," 2016.

[20] G. Asadi and M. B. Tahoori, "Soft error rate estimation and mitigation for sram-based fpgas," in *Proceedings of the 2005 ACM/SIGDA 13th international symposium on Field-programmable gate arrays*. ACM, 2005, pp. 149–160.

[21] A. Mishchenko *et al.*, "ABC: A system for sequential synthesis and verification," *URL http://www. eecs. berkeley. edu/˜ alanmi/abc*, 2007.

[22] D. Chai and A. Kuehlmann, "Building a better boolean matcher and symmetry detector," in *Proceedings of the conference on Design, automation and test in Europe: Proceedings*. European Design and Automation Association, 2006, pp. 1079–1084.

[23] C. Chiasson and V. Betz, "Coffe: Fully-automated transistor sizing for fpgas," in *Field-Programmable Technology (FPT), 2013 International Conference on*. IEEE, 2013, pp. 34–41.

[24] J. Lamoureux and S. J. Wilton, "Activity estimation for field-programmable gate arrays," in *Field Programmable Logic and Applications, 2006. FPL'06. International Conference on*. IEEE, 2006, pp. 1–8.

[25] J. Luu, J. Goeders, M. Wainberg, A. Somerville, T. Yu, K. Nasartschuk, M. Nasr, S. Wang, T. Liu, N. Ahmed *et al.*, "Vtr 7.0: Next generation architecture and cad system for fpgas," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 7, no. 2, p. 6, 2014.

[26] A. Ahari, B. Khaleghi, Z. Ebrahimi, H. Asadi, and M. B. Tahoori, "Towards dark silicon era in fpgas using complementary hard logic design," in *Field Programmable Logic and Applications (FPL), 2014 24th International Conference on*. IEEE, 2014, pp. 1–6.

[27] A. Gayasen, Y. Tsai, N. Vijaykrishnan, M. T. Kandemir, M. J. Irwin, and T. Tua, "Reducing leakage energy in fpgas using region-constrained placement," in *Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays*. ACM, 2004, pp. 51–58.

[28] J. H. Anderson and F. N. Najm, "Low-power programmable fpga routing circuitry," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 17, no. 8, pp. 1048–1060, 2009.

[29] A. A. Bsoul and S. J. Wilton, "An fpga with power-gated switch blocks," in *Field-Programmable Technology (FPT), 2012 International Conference on*. IEEE, 2012, p. 8.

[30] S. Yazdanshenas and H. Asadi, "Fine-grained architecture in dark silicon era for sram-based reconfigurable devices," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 61, no. 10, pp. 798–802, 2014.

[31] Z. Seifoori, B. Khaleghi, and H. Asadi, "A Power Gating Switch Box Architecture in Routing Network of SRAM-Based FPGAs in Dark Silicon Era," in *Proceedings of the conference on Design, Automation & Test in Europe*. European Design and Automation Association, 2017.

[32] A. A. Bsoul, S. J. Wilton, K. H. Tsoi, and W. Luk, "An fpga architecture and cad flow supporting dynamically controlled power gating," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 24, no. 1, pp. 178–191, 2015.

[33] R. Ahmed, A. A. Bsoul, S. J. Wilton, P. Hallschmid, and R. Klukas, "High-level synthesis-based design methodology for dynamic power-gated fpgas," in *Field Programmable Logic and Applications (FPL), 2014 24th International Conference on*. IEEE, 2014, pp. 1–4.

[34] A. Rahman, S. Das, T. Tuan, and S. Trimberger, "Determination of power gating granularity for fpga fabric," in *Custom Integrated Circuits Conference, 2006. CICC'06. IEEE*. IEEE, 2006, pp. 9–12.

[35] (2013) Predictive Technology Model (PTM). [Online]. Available: http://ptm.asu.edu/

[36] G. Lemieux and D. Lewis, "Using sparse crossbars within lut," in *Proceedings of the 2001 ACM/SIGDA ninth international symposium on Field programmable gate arrays*. ACM, 2001, pp. 59–68.

[37] "Virtex-4 platform FPGA user guide," User Guide, Xilinx, December 2008.

[38] "Stratix-2 platform FPGA hand book," Hand Book, Altera, April 2011.

[39] E. Kadric, D. Lakata, and A. DeHon, "Impact of memory architecture on fpga energy consumption," in *Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays*. ACM, 2015, pp. 146–155.

[40] M. Lin, A. El Gamal, Y.-C. Lu, and S. Wong, "Performance benefits of monolithically stacked 3-d fpga," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 26, no. 2, pp. 216–229, 2007.

Zahra Ebrahimi received the B.Sc. degree in computer engineering from *Sharif University of Technology* (SUT), Tehran, Iran, in 2014, wherein she is currently pursuing the M.Sc. degree. She has been with the *Data Storage, Processing, and Networks* (DSN) Laboratory at the Department of Computer Engineering, SUT, as a researach assistant for three years. Her current research interests include reconfigurable computing and computer-aided design.

Behnam Khaleghi has received his B.Sc. and M.Sc. degrees in Computer Engineering from SUT, Tehran, Iran, in 2013 and 2016, respectively. He is currently working as a research assistant in the DSN Laboratory at the Department of Computer Engineering, SUT. He spent the summers 2014 and 2015 as a visiting researcher at the Chair for Embedded Systems in the Karlsruhe Institute of Technology. His research interests include reconfigurable computing, CAD, and reliable system design. He has two best paper nominations at the DAC'17 and DATE'17.

Hossein Asadi (M'08, SM'14) received the B.S. and M.S. degrees in computer engineering from the SUT, Tehran, Iran, in 2000 and 2002, respectively, and the Ph.D. degree in electrical and computer engineering from Northeastern University, Boston, MA, USA, in 2007. He was with EMC Corporation, Hopkinton, MA, USA, as a Research Scientist and Senior Hardware Engineer, from 2006 to 2009. From 2002 to 2003, he was a member of the Dependable Systems Laboratory, SUT, where he researched hardware verification techniques. From 2001 to 2002, he was a member of the Sharif Rescue Robots Group. He has been with the Department of Computer Engineering, SUT, since 2009, where he is currently a tenured Associate Professor. He is the Founder and Director of the DSN Laboratory at SUT. He spent three months in the summer 2015 as a Visiting Professor at the the School of Computer and Communication Sciences at the Ecole Poly-technique Federele de Lausanne (EPFL). He has also co-founded the first startup company in the Middle East, called HPDS, designing and fabricating midrange and high-end data storage systems. He has authored and co-authored more than sixty technical papers in reputed journals and conference proceedings. His current research interests include data storage systems and networks, solid-state drives, operating system support for I/O and memory management, and reconfigurable and dependable computing. Dr. Asadi was a recipient of the Technical Award for the Best Robot Design from the International RoboCup Rescue Competition, organized by AAAI and RoboCup, a recipient of Best Paper Award at the 15th CSI Internation Symposium on Computer Architecture and Digital Systems (CADS), and the Distinguished Lecturer Award from SUT in 2010, one of the most prestigious awards in the university. He is also recipient of Extraordinary Ability in Science visa from US Citizenship and Immigration Services in 2008. He has also served as the publication chair of several national and international conferences including CNDS2013, AISP2013, and CSSE2013 during the past three years. Most recently, he has served as a Guest Editor of IEEE Transactions on Computers and a Program Co-Chair of the 18th International Symposium on Computer Architecture & Digital Systems (CADS2015).