

# FPGA Energy Efficiency by Leveraging Thermal Margin

Behnam Khaleghi, Sahand Salamat, Mohsen Imani, Tajana Rosing  
CSE Department, UC San Diego, La Jolla, CA 92093, USA  
{bkhaleghi, sasalama, moimani, tajana}@ucsd.edu

**Abstract**—FPGA devices are continuously evolving to meet high computation and performance demand for emerging applications. As a result, cutting edge FPGAs are not energy efficient as conventionally presumed to be, and therefore, aggressive power-saving techniques have become imperative. The clock rate of an FPGA-mapped design is set based on worst-case conditions to ensure reliable operation under all circumstances. This usually leaves a considerable timing margin that can be exploited to reduce power consumption by scaling voltage without lowering clock frequency. There are hurdles for such opportunistic voltage scaling in FPGAs because (a) critical paths change with designs, making timing evaluation difficult as voltage changes, (b) each FPGA resource has particular power-delay trade-off with voltage, (c) data corruption of configuration cells and memory blocks further hampers voltage scaling. In this paper, we propose a systematic approach to leverage the available thermal headroom of FPGA-mapped designs for power and energy improvement. By comprehensively analyzing the timing and power consumption of FPGA building blocks under varying temperatures and voltages, we propose a thermal-aware voltage scaling flow that effectively utilizes the thermal margin to reduce power consumption without degrading performance. We show the proposed flow can be employed for energy optimization as well, whereby power consumption and delay are compromised to accomplish the tasks with minimum energy. Lastly, we propose a simulation framework to be able to examine the efficiency of the proposed method for other applications that are inherently tolerant to a certain amount of error, granting further power saving opportunity. Experimental results over a set of industrial benchmarks indicate up to 36% power reduction with the same performance, and 66% total energy saving when energy is the optimization target.

## I. INTRODUCTION

The prevalence of computation-intensive workloads with high-performance requirements such as machine learning (ML) and data center applications [1], [2] accompanied with the advance of technology node have persuaded the FPGA vendors to integrate more resources with boosted clock rate in state-of-the-art FPGAs [3]. This, together with slowed shrinking of FPGAs supply voltage [4], have pushed these devices to a point they consume power comparable to CPUs [5]. Besides, there are prevailing energy-constrained applications in the Internet of Things (IoT) era, implemented in FPGAs, with the need for extreme energy efficiency [6], [7]. All in all, more efficient power reduction approaches for FPGAs are now indispensable.

As FPGAs already employ a manifold of device-level optimization to throttle power consumption [8], more *aggressive* power reduction techniques are gaining traction. These techniques generally build upon the conservative timing margin ( $d_g$ ) that is considered to compensate reliability threats in

deep-nano technologies; while an FPGA-mapped design is able to deliver an *actual* clock period of  $d_{V_{nom}}$  at nominal voltage  $V_{nom}$ , in practice, STA (static timing analysis) tools report an operating clock delay of  $d_{V_{nom}} + d_g$  to make up for uncertainties such as voltage fluctuations, degradation, temperature, etc. [5]. The aforementioned aggressive techniques exploit this available timing headroom to reduce the supply voltage of FPGAs down to  $V_{low}$  for which the *actual* clock period  $d_{V_{low}}$  becomes equal or close to  $d_{V_{nom}} + d_g$  (leaving no margin for guardbands). Thus, the device still delivers the original performance at a lower voltage. Note that *aggressive* voltage scaling techniques are different from conventional DVFS (dynamic voltage and frequency scaling) that concurrently tunes the frequency and voltage based on per-task performance demand of applications.

Although lowering the supply voltage of processors and ASIC devices has been known to be an effective power saving technique [9]–[11], there are limited studies presenting voltage scaling in FPGAs. Voltage scaling (in particular, aggressive and performance-aware one) in FPGAs is challenging mainly because: (a) *Critical Path* (CP) in an FPGA is design-dependent, which makes timing probing difficult under voltage scaling, especially considering the impact of temperature. Therefore, in contrast to ASIC designs, a set of precalibrated stand-alone sensory circuits, e.g., ring oscillators and CP monitors [10] cannot accurately correlate the sensor frequency with all varying CPs. (b) FPGA architectures are heterogeneous, comprising soft-fabric (i.e., programmable logic and routing resources), DSP cores, memory blocks (BRAM), etc. These building blocks are tightly coupled, and each has a particular power/delay relation with supply voltage. Considering the separate voltage rails provided for certain components, i.e.,  $V_{bram}$ ,  $V_{core}$ , and  $V_{io}$  that can be regulated separately, finding efficient voltage points becomes design-dependent and challenging. In other words, in multi-supply devices, multiple voltage combinations can lead to the target  $d_{V_{nom}} + d_g$  boundary, while only one tuple yields minimum power. This makes speculative voltage decrement no more efficient. (c) Scaling the voltage of FPGAs is also constrained by the data corruption of configuration and memory SRAM cells. In addition, as we will discuss in this paper, reducing the voltage of configuration cells unexpectedly increases FPGA power consumption in certain cases, calling for cautious analysis.

In this paper, we leverage the pessimistic thermal-induced timing slack to scale down FPGA operating voltage for power saving while tackling the aforementioned challenges as follows.

(1) We propose to incorporate thermal-aware voltage scaling in the FPGA design flow. We first obtain the temperature-delay-voltage correlation of FPGA resources within the supported temperature range. Then, we statically estimate the thermal distribution of applications to obtain the available timing headroom, for which voltages of different power rails can be efficiently determined based on the characterized library. For further effectiveness, we also suggest online (i.e., dynamic) voltage adaptation based on the response of thermal sensors. The proposed methods consider the voltage-temperature feedback loop and the separate power rails of specific resource types to yield maximum power efficiency.

(2) We leverage the proposed flow of (1) to explore the *energy* consumption, whereby we trade-off the performance and power consumption to achieve the minimum energy point. This is desirable for a majority of edge and IoT applications for which total energy consumption is the utmost concern.

(3) Our approaches mentioned in (1) and (2) are deterministic, as they guarantee timing closure. Nonetheless, many use-cases such as image processing and machine learning applications can tolerate a certain level of computation errors, which gives an opportunity for voltage *over-scaling*. However, it needs an examination of applications under these non-ideal conditions to get a glimpse of produced error in the output. We propose a primary FPGA simulation framework to be able to evaluate an FPGA-mapped design under voltage-scaling, i.e., when the delay of resources varies. Using the proposed framework, we map demonstrative machine learning applications into FPGA fabric and examine the impact of voltage *over-scaling* on extra power gain versus accuracy drop.

## II. BACKGROUND AND RELATED WORK

### A. FPGA Architecture

Fig. 1 illustrates the architecture of conventional tile-based FPGAs. The architecture comprises tiles of logic clusters (a.k.a CLBs or slices) that bind together using the configurable switch boxes (SBs) and connection blocks (CBs) to implement larger functions. Each logic CLB consists of  $N$  (e.g.,  $N = 10$ )  $K$ -input look-up tables (LUTs) each of which is capable to implement Boolean expressions up to  $K$  variables. SB multiplexers are located in the intersection of horizontal and vertical channels (wire tracks) to enable connectivity and bending of nets. These multiplexers are also responsible for connecting outputs of logic resources (LUTs, FFs, carry chain, etc.) to global routing, i.e., to horizontal and vertical channels. Analogously, CB multiplexers pass the selected global wires into the logic clusters. Specific FPGA columns are repetitively dedicated to Block RAMs and DSP cores. The majority of logic and routing resources have a multiplexer-like structure, mainly implemented as two-stage multiplexers that have been shown to provide optimal area-delay efficiency [12]. These resources often drive large loads, especially the global routing resources that have a high number of long fanout wires, so are augmented with large output buffers to improve performance.

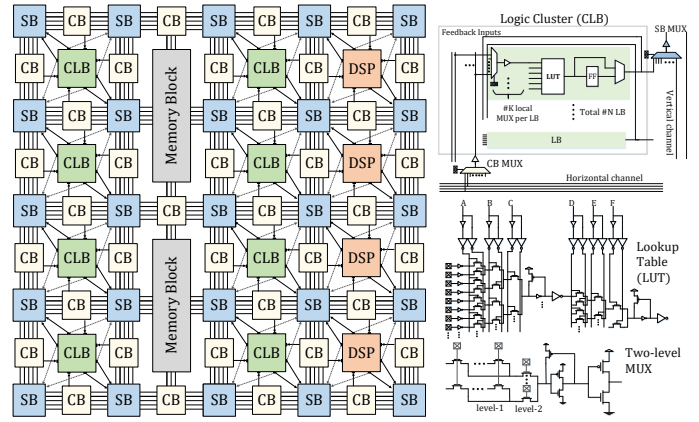


Fig. 1. Tile-based FPGA architecture (left), and constructing building blocks (right) [18].

### B. Related Work

While many previous studies have attempted to reduce FPGA power dissipation by power-gating, conventional DVFS, configurable dual supply voltage, control of signal levels, architectural innovations, etc. [13], there is a limited number of works that leverage the available timing margin to reduce voltage without sacrificing performance. Authors of [14] explore the timing headroom of FPGA-mapped designs by gradually reducing the voltage (keeping the frequency fixed) until observing the error. They detect the error by inserting a shadow register per each CP with a phase-shifted clock to detect the data mismatch caused by voltage reduction. As there are a huge number of CP and near-CP paths in a large synthesis-flattened design, the area and power overhead of this technique can be excessive. Also, measuring the slack in this technique depends on capturing the signal traversing the CP, while the CP may not be controllable at the runtime. Furthermore, the introduced capture registers cannot be used for paths that head to hard blocks such as BRAM. The study in [15] addresses the latter issue by replicating such paths and inserting an end-point proxy register (instead of the hard block) where an error-detector circuitry checks timing violations. However, the error-detector itself imposes delay, so a timing error raised in the original CP might propagate into the memory before being detected.

In [16], the authors propose a two-step self-calibrating voltage scaling scheme by exploiting the available timing slack of thermal margin. CPs of a design are extracted by using the STA tool and are then implemented on the FPGA fabric. Afterward, for different values of temperature and core voltage ( $T$  and  $V_{core}$ ), the maximum frequency is obtained by gradually increasing it until error is observed by the implemented error-detection circuit. This approach does not consider the thermal distribution within the chip [17] where a CP may experience varied temperature in reality. This either results in timing violation by ignoring the parts of CP that reside in hot (hence, slower) tiles or gives non-optimal results if a related thermal margin is considered. In addition, the STA tool reports the CPs according to worst-case condition while CPs

might change at lower temperatures. Thus, a larger number of near-CP paths need to evaluate which makes the entire process further cumbersome. BRAM and soft-logic voltage rails also are not separately considered, so the minimum employed voltage will be limited by the one that violates the timing first.

Finally, recent work in [19] examines the voltage scaling of FPGA BRAMs. The authors showed that up to 39% of BRAM voltage can be reduced without observing any error. Though it is promising in reducing the power of BRAMs by one order of magnitude, a trial-and-error based approach does not guarantee correct functionality as it is infeasible to examine all the inputs. Moreover, the overall efficiency is limited to the power of BRAMs.

Aforementioned techniques are all speculative as they decrease the voltage until observing an error. This overlooks errors that emerge gradually due to violating the guardbands (e.g., timing error as a result of degradation [20]) or may arise abruptly due to voltage transients [5]. Recently, work in [5] showed a margin of over 36% is needed for voltage transients as a result of load transients, and this margin is already considered in STA tools. Nevertheless, as voltage transients occur infrequently, the speculative voltage scaling methods do not take them into consideration, which will lead to timing violation at certain conditions. Our voltage scaling approach is different from previous studies as we incorporate it in the FPGA design flow by characterizing the resources during FPGA architecting. This eliminates the arduous task of voltage-timing speculation and guarantees timing. Our method precisely considers the correlation of temperature, delay, voltage, and power of resources and separate power rails of soft-fabric and memories, so it yields maximum efficiency by setting the optimal core and BRAM voltages as well as accurately estimating the timing according to the thermal distribution of the blocks. Finally, *over-scaling* of voltages needs timing simulation to observe the impact of timing violations. We enable it by our novel FPGA simulation flow. It is noteworthy that, similar to our work, the studies in [21] and [22] also propose *deterministic* frequency and/or voltage adaption through resource characterization. Nonetheless, [21] targets performance boosting by using non worst-case guardband in lower temperatures, and [22] adjusts both the frequency and voltage in non-peak workloads to save power (hence, has nothing to do with the temperature).

### III. PROPOSED METHOD

#### A. Preliminary

FPGA flow is different from conventional standard-cell based design of ASICs. Thus, we first elaborate the setup of experiments used in the rest of the paper before detailing the proposed method.

- **Power and Delay.** We use circuit-level simulations to obtain the delay and power of FPGA resources. For this end, we use the latest version of COFFE [23] that generates and characterizes an accurate netlist of FPGA resources according to the given architectural description using comprehensive

TABLE I  
FPGA ARCHITECTURE PARAMETERS USED IN COFFE

Parameter	Value	Parameter	Value
K	6	$SB_{mux}$ size	12
N	10	$CB_{mux}$ size	64
Channel tracks	240	$local_{mux}$ size	25
Wire segment length	4	$V_{core}, V_{bram}$	0.8 V, 0.95 V
Cluster global inputs	40	BRAM	$1024 \times 32$ bit

circuit-level HSPICE simulations. Besides the description of the target FPGA architecture, COFFE also requires technology process of transistors, for which we use 22 nm predictive technology model (PTM) [24]. COFFE also generates and evaluates the memory blocks of FPGA and has been shown to have a suitable delay and exact area match with commercial FPGAs [25]. Similar to configuration SRAM cells, the core of the memory blocks (i.e., eight-transistor dual-port SRAM cells) is implemented by 22 nm high-threshold low-power transistors [26], which throttles their leakage power by two orders of magnitude. As we will show in the rest of this section, our simulations show a similar power trend to commercial FPGAs.

COFFE does not model DSP blocks. We thus develop the DSP HDL code based on the Stratix IV description [27] and characterize it using Synopsys Design Compiler with NanGate 45 nm open cell library [28]. Then we scale the results to 22 nm based on measuring scaling factors of a selected set of cells at 45 nm and 22 nm technologies. Based on PrimeTime report, the developed DSP consumes 4.6 mW at 250 MHz, which is comparable to a 28 nm DSP that dissipates 5.6 mW at the same frequency [29]. To characterize the delay and power of programmable resources at different temperature and voltages, we sweep the parameters of COFFE-generated netlists in HSPICE simulations. For DSP, we create a set of standard-cell libraries using NanGate netlists by the means of Synopsys SiliconSmart so we could characterize the DSP at different operating conditions.

- **FPGA Flow.** We use VTR 7.0 (Verilog-to-Routing) [30] toolchain that enables defining a customized FPGA architecture and place and route the benchmarks. We select our benchmarks from VTR repository that belong to a wide variety of applications (vision, math, communication, etc.), contain single- and/or dual-port memory blocks as well as DSP blocks, with an average of over 23,800 6-input LUTs (maximum over 106 K). We use FPGA architecture parameters similar to Intel Stratix devices [18], [31] in COFFE and VPR<sup>1</sup> placement and routing, which is summarized in Table I. COFFE uses these parameters to generate SPICE netlist and area and delay report, which are then fed into VTR to place and route the benchmarks.  $K$  and  $N$  are the sizes of LUTs and number of logic blocks in a cluster (see Fig. 1), which are chosen to be 6 and 10 in accordance with Intel devices. Estimating the power consumption of applications (for both thermal simulation and power saving estimation) needs also their signal activity, for which we use ACE 2.0 [32]. We modified VPR to enable timing analysis at different scenarios using the characterized libraries.

<sup>1</sup>VPR (Versatile Place and Route) is the P&R tool in VTR toolchain.

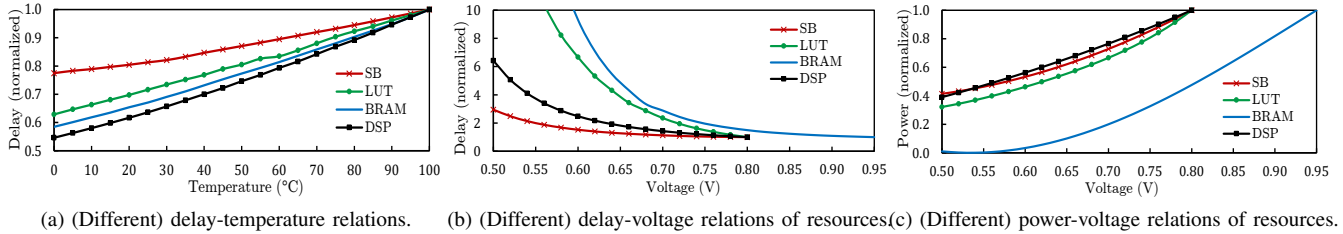


Fig. 2. Behavior of different FPGA resources under varying temperature and voltages.

• **Thermal Simulation.** We use HotSpot 6.0 [33] for thermal simulations. Inputs of HotSpot are device floorplan, power trace (or average power values), and device configuration parameters. For the floorplan file, we divide the device floorplan into a two-dimensional array of FPGA tiles with the areas reported by COFFE. We assume CLB tiles are square, and the heights of DSP and memory blocks are  $4\times$  and  $6\times$  of CLB tiles [30]. The number of tiles and location of each tile can be obtained from the placement and routing outputs reported by VPR. Leakage and dynamic power of each tile is obtained based on the current temperature and activities of resources of the tile. Finally, for the HotSpot configuration file, we change the parameters according to validated FPGA parameters in [34]. We adjust the convective resistance to concur with an effective thermal resistance ( $\theta_{JA}$ ) of contemporary FPGAs, i.e., we tune `r_conv` such that when the total power of given power trace is set to 1 Watt, the reported temperature by HotSpot equals  $\theta_{JA}$ . We examine the efficiency of the proposed technique by using a typical  $\theta_{JA}$  of  $2^\circ\text{C}/\text{W}$  as in today’s Intel and Xilinx devices (e.g., Virtex-7 and Stratix V) [29], [35], and a pessimistic thermal resistance of  $12^\circ\text{C}/\text{W}$ , corresponding to their mid-size devices (such as Spartan-7 or Artix-7) with still airflow.

### B. Proposed Thermal-Aware Voltage Scaling Flow

• **Motivation.** Fig. 2 gives perception on how temperature margin can be leveraged for power reduction. This figure is obtained using the experimental setup explained in the previous subsection. Numbers were not in the same range, so we normalized each one to its base value at  $100^\circ\text{C}$  and  $0.8\text{V}$  for the sake of clear illustration. According to Fig. 2(a), although FPGA timing analysis reports the worst-case to ensure timing meets in all scenarios [36], in practice, resources have a smaller delay at lower temperatures. For instance, at  $40^\circ\text{C}$ , delay of switch box ( $\times$  SB) is  $0.85\times$  of its delay at worst-case temperature<sup>2</sup>. This gap can be utilized for voltage reduction. Based on Fig. 2(b),  $0.68\text{V}$  is the point wherein this margin is fully utilized, i.e., delay of  $40^\circ\text{C}$  increases by  $\frac{1}{0.85\times}$  and becomes equal to delay at worst-case temperature. Eventually, Fig. 2(c) reveals this  $120\text{mV}$  reduction of voltage shrinks the switch box power down by 32%. As mentioned before, memory block comprises of low-threshold transistors, so uses a different power rail with a voltage higher than datapath (core) transistors. Other non-memory resources show

<sup>2</sup>We assume an upper-bound of  $100^\circ\text{C}$  for junction temperature [27].

a  $\sim V^2$  relation with voltage while BRAM observes a more dramatic power reduction as voltage scales. As is evident from the figure, different resources exhibit different delay behavior as temperature and voltage change. This stems from different sizing of transistors, input slope, output capacitance, etc., as detailed in previous works [11], [37].

We can get several insights from Fig. 2 and above discussion.

(a) Replica circuits such as ring oscillators used to correlate the temperature/voltage with frequency of ASICs are not a viable solution to track timing because, in FPGAs, CPs are design-dependent and made from different types and count of resources. Apparent from Fig. 2(a) and (b), designs bounded by routing (SB) have a totally different performance behavior compared to logic (LUT) bounded ones when temperature or voltage changes, so a set of representative paths fails to resemble all paths accurately.

(b) Previous studies rely on worst-case reported paths to check the timing (or to insert error detectors) while lowering the voltage. Nonetheless, from Fig. 2(a) and (b) we can infer a non-CP path may become CP at lower temperature or voltage. For instance, LUT delay severely increases at lower voltages, so the delay of LUT-bounded paths can exceed originally reported SB-bounded paths. This signifies cautious timing analysis in voltage scaling.

(c) More importantly, even if timing analysis of an FPGA-based design were possible under arbitrary (T, V) pairs, efficient voltage scaling would be still challenging because, as shown in Fig. 2(c), resources enjoy differently from voltage reduction. For instance, memory block shows better power saving as voltage scales, while, on the contrary, its delay also increases more under voltage scaling. Thus, for a certain timing headroom, there is a trade-off between power gain and increase of delay (i.e., use up of margin) when there are multiple power rails. This justifies why temperature-voltage-delay correlation and voltage-power libraries are indispensable for a reliable and efficient thermal-aware voltage scaling.

• **Thermal-Aware Voltage Scaling Flow.** Taking the above-mentioned insights into consideration, in the following we present our voltage scaling algorithm as the core of our energy efficiency technique, and then further elaborate it by exemplifying case-studies. Algorithm 1 can be either simply integrated into the current FPGA flow stack (i.e., in the original timing analysis step), or attached as an additional step. In either case, it relies on a pre-characterized library of delay

**Algorithm 1: Thermal-Aware Voltage Selection**


---

**Input:**  $netlist$ : Placed and routed design  
**Input:**  $T_{amb}$ : Ambient temperature  
**Input:**  $\vec{\alpha}$ : Input activities / sample inputs

- 1  $\vec{T}_{m \times n} = [T_{amb}, \dots, T_{amb}]$  //  $m, n$ : FPGA grid size
- 2  $\Delta \vec{T}_{m \times n} = [\infty, \dots, \infty]$
- 3  $d_{worst} = \mathcal{T}(netlist, T_{max}, V_{core_{max}}, V_{bram_{max}})$
- 4 **while**  $\|\Delta \vec{T}\|_{\infty} > \delta_T$  **do**
- 5      $\min_{V_{core}, V_{bram}} \vec{P}_{lkg}(\vec{T}, V_{core}, V_{bram}) +$
- 6          $\vec{P}_{dyn}(netlist, \vec{\alpha}, f_{worst}, V_{core}, V_{bram})$
- 7     **s.t.**  $\mathcal{T}(netlist, \vec{T}, V_{core}, V_{bram}) \leq d_{worst}$
- 8      $\vec{T}_{old} = \vec{T}$
- 9      $\vec{T} = HotSpot(\vec{P}_{lkg} + \vec{P}_{dyn})$
- 10     $\Delta \vec{T} = \vec{T} - \vec{T}_{old}$
- 11 **return**  $V_{core}, V_{bram}$

---

and power, as detailed in Section III-A. If Algorithm 1 is used as an additional step, then it needs to get the post place and route netlist. Other inputs of the algorithm are maximum temperature surrounding the FPGA board, and sample inputs or activities (we further discuss it later).

FPGA thermal estimation usually relies on a single total power and ambient temperature value to estimate the junction temperature [29], however, we divide the target FPGA into a grid of  $m \times n$  tiles, for  $m$  and  $n$  being the number of FPGA rows and columns. It improves the accuracy of thermal estimation and helps to catch potential hotspot regions, where the blocks have higher delay than the rest of the board, hence need fine-grained timing analysis for both accuracy and efficiency (i.e., to avoid under- or over-estimation of timing).  $d_{worst}$  is the delay that conventional one-size-fits-all timing analysis  $\mathcal{T}$  of FPGA reports under nominal memory and core voltages and maximum temperature [36] while also considers some margin for reliability issues such as voltage transients [5]. Thus  $d_{worst}$  is the target delay that our algorithm attempts to deliver with lower voltages. One drawback of previous voltage scaling approaches [14], [16] is they invade this reliability margin when they speculatively reduce the voltage until observing an error in the output, as the error does not show up in regular conditions. The core of the algorithm is a loop where, based on previously obtained temperature for each tile (set to  $T_{amb}$  initially), it finds the  $(V_{core}, V_{bram})$  pair that minimizes the power while watches over the delay of candidate pair to not exceed  $d_{worst}$ . In the first iteration of the algorithm, it explores all  $|V_{core}| \times |V_{bram}|$  pairs. In the next iterations, execution time can be significantly reduced by limiting the search to the boundaries of the previous solution, making subsequent iterations  $O(1)$ . Note that in both timing analysis and power calculation (lines 5–7), each tile has its own activity and potentially different temperature, so we use vectors of length  $m \times n$  to store the values associated with each tile. Temperature affects the leakage power and delay of the tile resources, while activity affects the dynamic power. Hence,  $\vec{T}$  is passed to  $\vec{P}_{lkg}$  calculation and timing ( $\mathcal{T}$ ) analysis, while  $\vec{\alpha}$  is passed to  $\vec{P}_{dyn}$  to estimate dynamic power at  $d_{worst}$

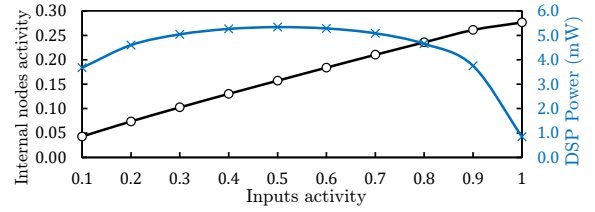


Fig. 3. Activity of benchmarks nodes for different activities of primary inputs (left/blue), and DSP power at different activities of its inputs (right/red).

(clock cycle will be always  $d_{worst}$ ). Finally, the power values are imported in a thermal simulator to update temperatures of tiles. This procedure repeats until reaching a steady-state temperature. For thermal simulation in our experiments we use HotSpot 6.0 [33], with setup already detailed in Section III-A.

• **Static and Dynamic Implementations.** Static implementation of the proposed technique is straightforward as both core and memory voltages are determined during the configuration of design, whether by incorporating Algorithm 1 in original timing analysis of FPGA, or using it as a post-routing addendum. As the voltages remain fixed in the field operation, the algorithm needs to consider the corner case of the programmed design, i.e., the highest temperature it might reach. A noteworthy point here is that activities of internal nodes of a design do not linearly correspond to activities of primary inputs. In Fig. 3, when signal activity factor ( $\alpha$ ) of benchmarks inputs increases from 0.1 to 1, activity of internal nodes (averaged over all 10 benchmarks) increases from 0.05 to  $\sim 0.27$ , which is significantly less than  $\alpha = 1$  considered for primary inputs. In addition, in some blocks such as DSP, the increase in activity of primary inputs does not necessarily translate to increase of power. As can be seen from Fig. 3, DSP power increases by only around 37% when its inputs activities raise from 0.1 to 0.3, then its power saturates until  $\alpha \in [0.3, 0.7]$ , and declines thereafter. This behavior of power is because the frequently changing inputs offset each other more often (e.g., when both inputs of an XOR function change in a clock, its output remains the same). All in all, by caring for the worst-case input activity, the proposed static scheme guarantees reliable operation at corners without overly pessimistic activity estimation, though the ambient temperature needs to consider maximum possible.

The proposed thermal-aware voltage scaling scheme can be implemented online (dynamic) to avoid pessimistic assumption on the temperature bound. Instead of thermal simulation, online scheme reads the junction temperature using on-board sensors available on all contemporary FPGAs. For instance, Intel devices already contain temperature sensing diodes (TSD) with instantiatable IP cores having their internal clock source that can output the junction temperature with a resolution of 10 bits in 1,024 clock cycles (i.e., 1 ms) [38]. Therefore, during the configuration of each design, we create a look-up table with temperature  $T$  as its keys and  $(V_{core}, V_{bram})$  as the values that minimize the power for that  $T$ . Optimal  $(V_{core}, V_{bram})$  of each temperature can be obtained in the same way explained above for the static approach. Reading

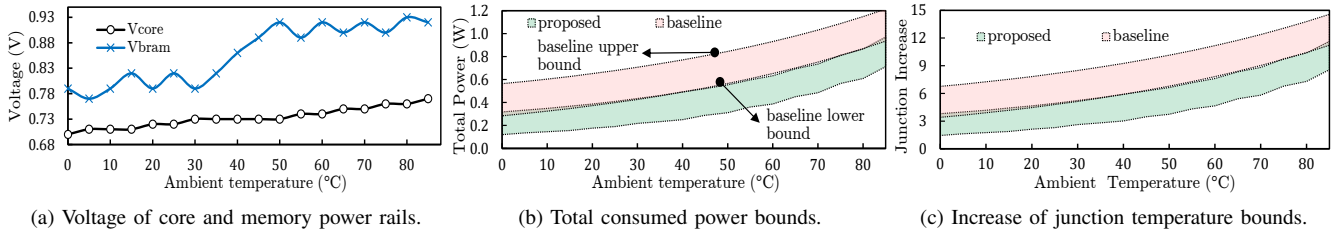


Fig. 4. Outputs of Algorithm 1 for `mkDelayWorker` benchmark under different ambient temperatures.

the temperature with steps of few milliseconds is large-enough to allow on-chip voltage regulators (such as Intel on-the-fly regulators) to adjust the voltage [39], and yet is small enough to avoid temporal heat-up mismatch that takes orders of seconds [40]. The sensed junction temperature can be directly used as VID (voltage identification) for the programmable integrated voltage regulator to adjust the voltage with the pre-loaded values [39]. A thermal margin (e.g., 5 °C) might be considered to account for the error of TSDs and potential spatial thermal gradients [41].

- **Case-study.** To elaborate our method, we use `mkDelayWorker` benchmark with 6,128 LUTs and 164 memory blocks that VPR mapped it to a  $92 \times 92$  grid device due to its high BRAM demand, with a frequency of 71.6 MHz. Our simulations show the device consumes a leakage power of 0.367 W at 25 °C (considering all used and unused resources), while the closest Intel device (Stratix V 5SGSD3) is  $1.5 \times$  in size with a power of 0.646 W. This  $1.76 \times$  power ratio is acceptable considering the size difference and more advanced technology we use in our simulations (22 nm versus 28 nm).

Fig. 4 shows the results of our static voltage scaling scheme on `mkDelayWorker` benchmark. We assume ambient (near-board) temperature range from 0 °C up to 85 °C as previous studies have shown that board temperature of datacenter FPGAs can reach up to  $\sim 70$  °C [1]. Fig. 4(a) shows that, moving from 0 °C to 85 °C, generally both  $V_{core}$  and  $V_{bram}$  increase towards their nominal 0.8 V and 0.95 V values to meet timing in worst-case junction temperature. Small fluctuations of BRAM voltage at certain points is to yield maximum power saving. For instance, at 30 °C,  $V_{core}, V_{bram} = (0.73, \mathbf{0.79})$  while at 25 °C,  $V_{core}, V_{bram} = (0.72, \mathbf{0.82})$ , however, we expected BRAM voltage to be lower for 25 °C. This is because actually the 10 mV reduction of  $V_{core}$  at 25 °C is worth the 30 mV increase of  $V_{bram}$ ; as we examined, it resulted in a power of 410 mW while experiments showed that the other combination (0.73, 0.79) would consume 420 mW ( $> 410$  mW). It indicates preciseness of our technique in determining most efficient voltage pairs for a given  $T_{amb}$ .

Fig. 4(b) compares the total power consumption of the proposed technique and baseline. Each curve shows the lower and upper bound of the power, where lower bound corresponds to  $\alpha = 0.1$  and upper bound corresponds to maximum dynamic power consumption, i.e.,  $\alpha = 1.0$ . As explained above and showed in Fig. 3, power does not increase linearly with activity (i.e., upper bound of power is not  $10 \times$  of lower

TABLE II  
ITERATIONS OF ALGORITHM 1 ON `mkDelayWorker` AT  $T_{amb} = 60$  °C.

Iter.	$V_{core}$ (mV)	$V_{bram}$ (mV)	Power (mW)	$T_{junct}$ (°C)	Time (s)
1	740	920	485	65.82	10.9
2	750	900	558	66.69	3.1
3	750	910	564	66.76	3.1
4	750	910	564	66.77	3.1
5	750	910	564	66.77	3.1

bound) because leakage power is independent of activity and also activities of internal nodes are not linearly correlated with primary inputs activity. As expected, lower temperatures have more power saving as there is more margin to reduce voltages. Also, although the proposed method optimizes the voltages according to worst-case activity, our method still significantly improves power when activity is low. Note that the baseline has fixed voltages; however, it also consumes less leakage power in lower temperature so its total power also reduces in lower temperatures. It is noteworthy that in our experiments we observed the leakage power has an exponential relation of  $e^{0.015T}$  with temperature, which is comparative to  $e^{0.017T}$  we derived for Intel devices [35]. The junction temperature of baseline exceeded 100 °C when ambient temperature reaches 85 °C. Thus, Fig. 4(b) and (c) are limited to 85 °C.

Finally, Fig. 4(c) shows the upper ( $\alpha = 1.0$ ) and lower ( $\alpha = 0.1$ ) bounds of increase in junction temperature of device tiles for different ambient temperatures (and corresponding voltages). Higher activity consumes more dynamic power hence has a higher impact on temperature. There is a close correlation between Fig. 4(b) and (c) as steady-state junction temperature is correlated to total power, especially when design activity is uniform. Please note that overlapping (almost) of lower bound of the proposed method with upper bound of the baseline is haphazard and specific to this benchmark.

- **Algorithm Runtime.** For all of our benchmarks, the flow converges in less than 6 iterations. At low  $T_{amb}$  values, due to weak temperature-leakage feedback, the algorithm converges in 2–3 iterations. On a typical desktop system, the first iteration takes less than 12 seconds, and in subsequent iterations the algorithm limits the search space to the boundary of the current solution, making each iteration less than 4 seconds. Thermal simulation takes  $\sim 2.5$  seconds of each iteration. Table II shows the details of the static voltage scaling algorithm. At first round, voltages are set to (0.74, 0.92). The resultant power increases the temperature by 5.82 °C, which increases the delay (tightens the margin) and leakage. Thus, the second iteration changes the voltages to (0.75, 0.90) for timing

closure. The increase of temperature in the first iteration also considerably increases the (leakage) power, from 485 mW to 558 mW. The temperature then starts converging; hence the subsequent voltage and power changes are insignificant.

- **Discussion.** We do not change the voltages of other power rails such as auxiliary supply voltage ( $V_{aux}$ ) and I/O voltage ( $V_{io}$ ) as they enable interfacing with other devices and have relatively low power consumption. We also do not touch the voltage of configuration SRAM cells as they use high-threshold mid-oxide transistors with two orders of magnitude less leakage [26]. In addition, we observed that reducing the voltage of SRAM cells causes voltage drop in pass-gate based multiplexer structure of resources, which increases the leakage power of buffers due to non-ideal voltage at their input.

### C. Proposed Thermal-Aware Energy Optimization Flow

While performance is the major concern of high-end FPGA designs, total energy usage is a primary concern of battery-backed and IoT applications. The goal of optimal energy exploration is to find the voltage(s)  $V_{opt}$  and clock period  $d_{opt}$ , for which  $E(V_{opt}, d_{opt}) = P(V_{opt}, d_{opt}) \times d_{opt}$  is minimum, where  $E(V_{opt}, d_{opt})$  is the design energy consumption rate operating with  $V_{opt}$  and clock  $d_{opt}$ . To obtain the  $V_{opt}$  and clock period  $d_{opt}$ , clearly, the operating voltage  $V_{opt}$  must be able to deliver the clock period of  $d_{opt}$  to avoid timing violations. Second, the design must operate with maximum possible frequency for the given voltage. Otherwise, if the clock period is set to  $\alpha \cdot d_{opt}$  ( $\alpha > 1$ ), total energy becomes:

$$E(V_{opt}, \alpha \cdot d_{opt}) = (P_{lkg}(V_{opt}) + \frac{P_{dyn}(V_{opt})}{\alpha}) \times \alpha \cdot d_{opt} \quad (1)$$

$$= (\alpha P_{lkg}(V_{opt}) + P_{dyn}(V_{opt}))d_{opt} > (P_{lkg}(V_{opt}) + P_{dyn}(V_{opt}))d_{opt}$$

That is, scaling the clock by  $\alpha$  scales the dynamic power by  $\frac{1}{\alpha}$ , but since the execution time also scales by  $\alpha$ , the total consumed dynamic energy remains the same. Nonetheless, the leakage power is independent of the clock period. Thus, the leakage energy scales by  $\alpha$ . Therefore, for a given voltage (which we aim to find the best one), the clock period needs to be minimum possible to minimize total energy.

That being said, we derive the new Algorithm 2 that looks for the  $(V_{core}, V_{bram})$  pair that achieves minimum power-delay product as the energy metric whilst also exploits the temperature headroom for further efficiency. Clearly, having the temperature-delay-voltage and voltage-power characterization is vital for the reliability and efficiency of the proposed flow. Having Algorithm 1 already explained, understanding the Algorithm 2 is straightforward. Essentially, it looks for all  $(V_{core}, V_{bram})$  pairs, and as reasoned above, finds the maximum frequency considering thermal margin. In contrast with the voltage scaling flow, Algorithm 2 exploits the available thermal margin to maximize the frequency for a candidate voltage, rather than lowering the voltage for a fixed frequency. Thermal simulation (line 10) is again crucial as changing the frequency (line 6) changes the power and hence temperature.

Algorithm 1 was performing thermal simulation for the best  $(V_{core}, V_{bram})$  at each iteration and we observed that, in the

---

### Algorithm 2: Thermal-Aware Energy Optimization

---

**Input:** *netlist*: Placed and routed design  
**Input:**  $T_{amb}$ : Ambient temperature  
**Input:**  $\vec{\alpha}$ : Input activities / sample inputs

```

1  $E_{min} = \infty$ 
2 for  $\forall V_{core}, \forall V_{bram}$  do
3    $\vec{T}_{n \times n} = [T_{amb}, \dots, T_{amb}]$ 
4    $\vec{\Delta T}_{n \times n} = [\infty, \dots, \infty]$ 
5   while  $\|\vec{\Delta T}\|_{\infty} > \delta_T$  do
6      $d_{max} = \mathcal{T}(\text{netlist}, \vec{T}, V_{core}, V_{bram})$ 
7      $\vec{P}_{total} = \vec{P}_{lkg}(\vec{T}, V_{core}, V_{bram}) +$ 
8        $\vec{P}_{dyn}(\text{netlist}, \vec{\alpha}, d_{max}, V_{core}, V_{bram})$ 
9      $\vec{T}_{old} = \vec{T}$ 
10     $\vec{T} = HotSpot(\vec{P}_{lkg} + \vec{P}_{dyn})$ 
11     $\vec{\Delta T} = \vec{T} - \vec{T}_{old}$ 
12    if  $d_{max} \times \sum_i \vec{P}_i < E_{min}$  then
13       $E_{min} = d_{max} \times \sum_i \vec{P}_i$ 
14       $V_{core_{min}} = V_{core}$ 
15       $V_{bram_{min}} = V_{bram}$ 
16 return  $V_{core_{min}}, V_{bram_{min}}$ 

```

---

worst case, it converges in less than eight iterations. However, Algorithm 2 needs to explore all  $|V_{core}| \times |V_{bram}|$  combinations and perform several thermal simulations under each. It could take up to several hours for large benchmarks. We enhanced it by, first, skipping a  $(V_{core}, V_{bram})$  combination if its energy in the initial loop (i.e., before involving temperature-delay feedback of line 10) was larger than the already found optimum. In addition, we also considered a small temperature margin of 0.1 °C, so we could avoid the thermal simulation of cases with power within  $\frac{0.1}{\theta_{JA}}$  range of a previously obtained case. These optimizations reduced the average runtime on benchmarks by two-order of magnitude (from 72 minutes to 49 seconds) with virtually no impact on the solution.

### D. Timing-Speculative Voltage Over-Scaling

Timing speculation has been shown to provide opportunistic power reduction in applications that can inherently tolerate a certain amount of error. Examples are: (a) image processing circuits such as DCT/IDCT where small drop of PSNR (Peak Signal to Noise Ratio) might not be perceived by human [42]. (b) Deep Neural Networks (DNNs) because of their pooling layers that filter out a significant portion of intermediate results and also the stochastic nature of the gradient descent [43] (c) light-weight alternatives of DNNs such as the brain-inspired computing that performs the main machine learning applications by using inexpensive operations on hypervectors that can bear a certain amount of inaccuracy [44], etc. Timing-speculative voltage over-scaling is orthogonal to the thermal-aware voltage reduction with the opportunity of violating the timing for more significant power saving.

Timing-speculative voltage scaling requires post P&R simulation of the targeted design with the timing data at the scaled voltage to estimate the incurred inaccuracy. In standard-cell design approaches, timing speculation can be realized

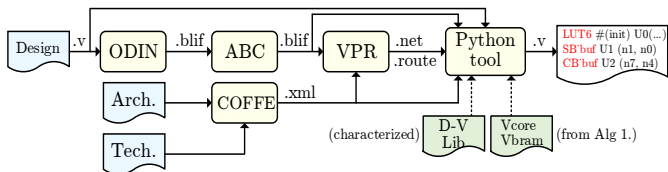


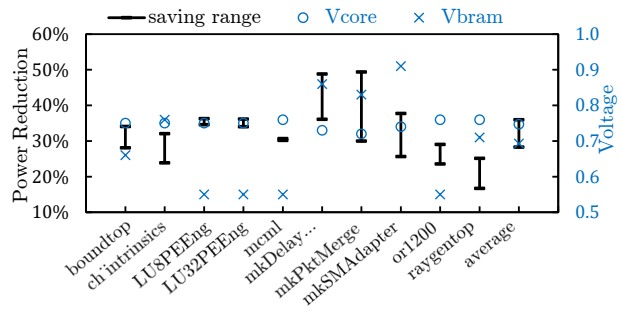
Fig. 5. The proposed simulation flow for FPGA-mapped applications, enabling speculative voltage scaling.

by generating the same cell library under scaled voltages or providing multiple operating condition modes at the same library. Thus, the synthesized design can be simulated using the new timing data. Nonetheless, to the best of our knowledge, there is no FPGA framework for post place and route (timing) simulation, particularly under varying voltages.

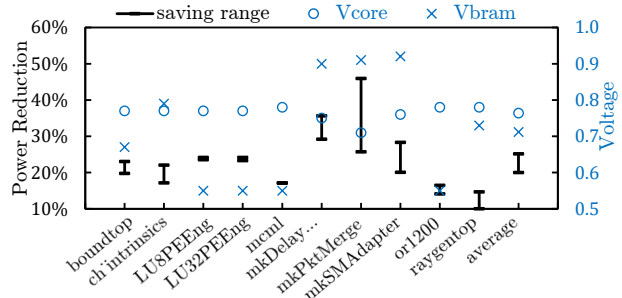
By taking advantage of our characterization libraries, we build our simulation framework upon the Verilog-to-Routing (VTR) [30] project explained in Section III-A. Fig. 5 demonstrates the proposed post P&R simulation framework. The flow begins with synthesis (using ODIN [45]) and technology mapping (using Berkeley ABC [46]) of the HDL description to BLIF format that can be imported to the VPR P&R tool [30]. Architectural description of the target FPGA needed by VPR can be hand-written or auto-generated by COFFE [23]. VPR generates a .net file that describes the placement information of resources, and a .route file that provides the routing information of design nets. Finally, we developed a Python-based tool to analyze VPR outputs and instantiate the FPGA’s utilized components using primitives similar to Xilinx syntax in Verilog HDL. The generated file is augmented with the delay of the resources obtained by parsing the VPR outputs. It is noteworthy that all configurable multiplexers (SBs, CBs, etc.) are programmed in a place and routed design, so we could simplify their functionality as a buffer just to incorporate their delay information. VPR’s placement output (.net) does not include a functional description. Thus, we retrieve the configuration of LUTs from the technology-mapped BLIF. Similarly, BLIF files do not contain BRAMs initialization, so we read them back from the original HDL file. To make the voltage over-scaling efficient, we utilize the proposed thermal-aware voltage scaling as follows. For a given timing rate (e.g.,  $1.1\times$  of original clock), we change the timing condition of Algorithm 1 (line 7) to meet the new constraint ( $1.1\times$  of  $d_{worst}$ ). Hence, the obtained over-scaled voltages are optimal for that allowed amount of violation. We repeat it for different timing violation rates. In Section IV we report the additional power saving granted by speculative voltage-scaling.

#### IV. EXPERIMENTAL RESULTS

• **Power Reduction.** Fig. 6 demonstrates the power reduction using the proposed flow of Algorithm 1. The flow finds the optimum core and memory voltage pair ( $V_{core}$  and  $V_{bram}$ ) assuming highest inputs activity ( $\alpha$ ) to guarantee it satisfies temperature corners. In practice, however, the input activity range might be lower. Therefore, for the obtained optimal voltages, we assumed a varying activity  $\alpha \in [0.1,$



(a) Range of power reduction at 40°C (left axis) and corresponding core and memory voltages of each benchmark (right axis).



(b) Range of power reduction at 65°C (left axis) and corresponding core and memory voltages of each benchmark (right axis).

Fig. 6. Power reduction and voltages for 40°C and 65°C board temperatures.

1.0] and calculated the power reduction for the entire range. Therefore, Fig. 6 demonstrates a range of power saving. The right axis of this figure also shows the optimal  $V_{core}$  and  $V_{bram}$  voltages for each benchmark. In several benchmarks, the paths containing memories were significantly shorter than critical paths. For instance, in LU8PEEng, the critical path is  $21\times$  longer than the longest BRAM path. For these paths,  $V_{bram}$  is reduced down to 0.55 V, which we set as the lowest voltage level before device crashes [19]. In Fig. 6(a) we considered a device operating at  $T_{amb} = 40^\circ\text{C}$  with  $\theta_{JA} = 12^\circ\text{C}/\text{W}$ , and in Fig. 6(b) as considered more high-end device operating at  $65^\circ\text{C}$  with  $\theta_{JA} = 2^\circ\text{C}/\text{W}$  (see Section III-A for details of experiments). The opportunity of power saving reduces in higher temperatures. At 40°C, the average power saving of 10 benchmarks is 28.3%–36.0% (depends on activity), while at 65°C it becomes 20.0%–25.0%. We observed up to 9.2°C increase in the junction temperature of the baseline, which reduced to 5.9°C in the proposed method due to consuming less power. Benchmarks have different power reduction and optimal voltages based on the resources on critical paths (that determine the voltage scaling limit), used resources, the activity of nodes, etc. Comparing Fig. 6(a) and (b) also reveals how differently the voltages of benchmarks need to be adjusted moving from 40°C to 65°C: raygentop needs boosting both voltages by 20 mV, or1200 needs only +20 mV of core rail, and mkPktMerge needs 80 mV increase of memory rail with 10 mV reduction of core voltage, suggesting the necessity of dynamic implementation (see Section III-B) for best efficacy.

• **Energy Reduction.** Fig. 7 shows the range of energy reduction (left axis) of each benchmark using the proposed



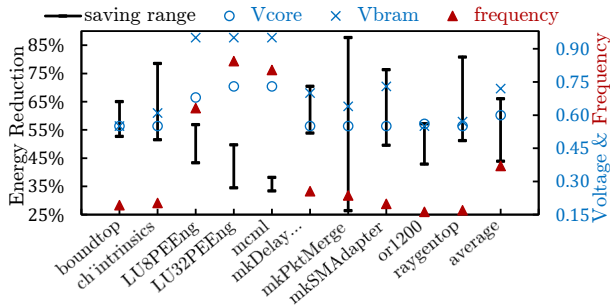


Fig. 7. Range of energy savings at 65 °C (left axis), and corresponding optimal voltage values and frequency ratio (right axis).

energy optimization flow at 65 °C. The points ( $\times$ ,  $\circ$ ,  $\blacktriangle$ ) correspond to the right axis and show the optimal voltage values and frequency ratio. Remember that the goal of our energy minimization flow was to find out the minimum energy consumption point (power-delay product) by compromising the delay and power, so the delay has been increased by  $\frac{1}{0.37} = 2.7\times$  while the overall consumed energy is improved by 44%–66% depending on input activities. There are obvious differences with optimal points of power and energy reduction flows. Unlike the power flow, here,  $V_{bram}$  of above-mentioned benchmarks (e.g., LU8PEEng) have not been shrunk down to 0.55 V because the delay of their critical paths is also increased, hence the memory voltage cannot be freely reduced. The ranges of energy savings are also more stretched (e.g., in mkPktMerge) because in higher activities, memory dynamic energy becomes the dominant contributor to total energy consumption, hence, throttling its energy becomes worthwhile even considering the increased delay. As it can be seen in the figure, its  $V_{core}$  is reduced to 0.64 V while in power reduction flow (Fig. 6(b)) it could be reduced to 0.91 V because the clock delay must have been remained fixed.

• **Speculative Voltage Over-Scaling.** We chose LeNet [47] as a classic CNN for handwritten digit recognition and implemented as a systolic array architecture [48]. Its relatively small size makes the timing simulation computationally tractable. We also selected another machine learning algorithm based on computing with hyperdimensional (HD) [49] vectors to detect two face/non-face classes among 10,000 web faces of a face detection dataset (FACE) from Caltech [50]. Fig. 8 shows the result of thermal-aware voltage over-scaling. Initially, CP delay is  $1\times$  of original clock, meaning that no timing violation is allowed and the  $\sim 34\%$  power reduction is because of thermal-aware voltage scaling. Thereafter, we allow up to 40% violation of CP delay, where accuracy drop becomes noticeable at  $1.2\times$  of the original clock. This tolerance is because DNNs are intrinsically error-tolerant (e.g., allow quantization of weights down to three bits in the LeNet [51]). Similarly, previous studies of HD have shown an accuracy drop of merely 4% when up to 30% of the vectors bits are flipped (i.e., noisy) [44] mainly because orthogonality of vectors, making them discernible under error. Based on Fig. 8, when voltage over-scaling increases the CP delay to  $1.35\times$  of the clock period, errors start spiking. At this point, by respectively 3%

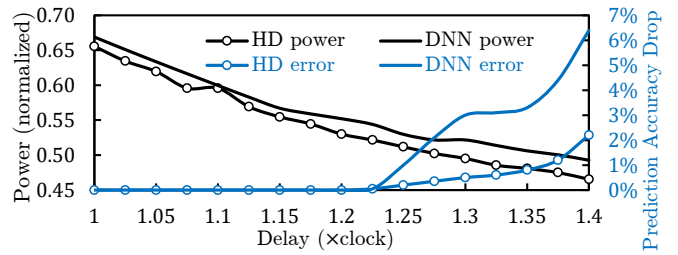


Fig. 8. Power reduction (left axis) and error increase (right axis) under voltage over-scaling. X-axis shows violation of critical path delay.  $T_{amb}$  is 40 °C.

and 0.5% accuracy drop, LeNet and HD powers are reduced by 48% and 50%, which means additional 15% and 16% improvement compared to our original voltage-scaling (with  $\sim 34\%$  improvement for both) in which CP delay does not exceed clock period.

## V. CONCLUSION

In this paper, we proposed power and energy optimization techniques for FPGAs by characterizing FPGA resources and utilizing thermal headroom. Keeping the performance intact, the voltage scaling flow determines the optimal voltages of designs based on their thermal distribution and can be implemented statically with fixed voltage(s), or dynamically using programmable voltage regulators at highly varying ambient temperatures. Our proposed energy optimization flow compromises the delay and power to seek optimal point that minimizes total consumed energy. Finally, we proposed a timing simulation framework that provides further power saving opportunity by making the impact of voltage over-scaling observable.

## ACKNOWLEDGEMENTS

This work was partially supported by CRISP, one of six centers in JUMP, an SRC program sponsored by DARPA, and also NSF grants #1911095, #1826967, #1730158 and #1527034.

## REFERENCES

- [1] A. Putnam, A. M. Caulfield, E. S. Chung, D. Chiou, K. Constantinides, J. Demme *et al.*, “A reconfigurable fabric for accelerating large-scale datacenter services,” *ACM SIGARCH Computer Architecture News*, vol. 42, no. 3, pp. 13–24, 2014.
- [2] G. Lacey, G. W. Taylor, and S. Areibi, “Deep learning on fpgas: Past, present, and future,” *arXiv preprint arXiv:1602.04283*, 2016.
- [3] “Intel stratix 10 logic array blocks and adaptive logic modules user guide,” User Guide, Intel, September 2017.
- [4] I. Ahmed, S. Zhao, J. Meijers, O. Trescases, and V. Betz, “Automatic bram testing for robust dynamic voltage scaling for fpgas,” in *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2018, pp. 68–687.
- [5] L. L. Shen, I. Ahmed, and V. Betz, “Fast voltage transients on fpgas: Impact and mitigation strategies,” in *2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 2019, pp. 271–279.
- [6] D. Blaauw, D. Sylvester, P. Dutta, Y. Lee, I. Lee, S. Bang *et al.*, “Iot design space challenges: Circuits and systems,” in *Symposium on VLSI Technology (VLSI-Technology): Digest of Technical Papers*. IEEE, 2014, pp. 1–2.
- [7] S. Venkataramani, K. Roy, and A. Raghunathan, “Efficient embedded learning for iot devices,” in *Design Automation Conference (ASP-DAC), 21st Asia and South Pacific*. IEEE, 2016, pp. 308–311.

- [8] "Lowering power at 28 nm with xilinx 7 series devices," White Paper, Xilinx, January 2015.
- [9] M. Bao, A. Andrei, P. Eles, and Z. Peng, "On-line thermal aware dynamic voltage scaling for energy optimization with frequency/temperature dependency consideration," in *Design Automation Conference, 46th ACM/IEEE*. IEEE, 2009, pp. 490–495.
- [10] C. R. Lefurgy, A. J. Drake, M. S. Floyd, M. S. Allen-Ware, B. Brock, J. A. Tierno *et al.*, "Active management of timing guardband to save energy in power7," in *Microarchitecture (MICRO), 44th Annual IEEE/ACM International Symposium on*. IEEE, 2011, pp. 1–11.
- [11] H. Amrouch, B. Khaleghi, and J. Henkel, "Voltage adaptation under temperature variation," in *15th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*. IEEE, 2018, pp. 57–60.
- [12] D. Lewis, E. Ahmed, G. Baeckler, V. Betz, M. Bourgeault, D. Cashman *et al.*, "The stratix ii logic and routing architecture," in *Proceedings of the ACM/SIGDA 13th international symposium on Field-programmable gate arrays*. ACM, 2005, pp. 14–20.
- [13] Z. Seifoori, Z. Ebrahimi, B. Khaleghi, and H. Asadi, "Introduction to emerging sram-based fpga architectures in dark silicon era," *Advances in Computers*, 2018.
- [14] J. M. Levine, E. Stott, and P. Y. Cheung, "Dynamic voltage & frequency scaling with online slack measurement," in *Proceedings of the 2014 ACM/SIGDA international symposium on Field-programmable gate arrays*. ACM, 2014, pp. 65–74.
- [15] J. Nunez-Yanez, "Adaptive voltage scaling in a heterogeneous fpga device with memory and logic in-situ detectors," *Microprocessors and Microsystems*, vol. 51, pp. 227–238, 2017.
- [16] S. Zhao, I. Ahmed, C. Lamoureux, A. Lotfi, V. Betz, and O. Trescases, "Robust self-calibrated dynamic voltage scaling in fpgas with thermal and ir-drop compensation," *IEEE Transactions on Power Electronics*, vol. 33, no. 10, pp. 8500–8511, 2018.
- [17] A. Amouri, H. Amrouch, T. Ebi, J. Henkel, and M. Tahoori, "Accurate thermal-profile estimation and validation for fpga-mapped circuits," in *Field-Programmable Custom Computing Machines (FCCM), 2013 IEEE 21st Annual International Symposium on*. IEEE, 2013, pp. 57–60.
- [18] C. Chiasson and V. Betz, "Should fpgas abandon the pass-gate?" in *FPL*, vol. 13, 2013, pp. 1–8.
- [19] B. Salami, O. S. Unsal, and A. Cristal Kestelman, "Comprehensive evaluation of supply voltage underscaling in fpga on-chip memories," in *Proceedings of the 51th Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 2018.
- [20] B. Khaleghi, B. Omidi, H. Amrouch, J. Henkel, and H. Asadi, "Estimating and mitigating aging effects in routing network of fpgas," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 3, pp. 651–664, 2019.
- [21] B. Khaleghi and T. Š. Rosing, "Thermal-aware design and flow for fpga performance improvement," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 342–347.
- [22] S. Salamat, B. Khaleghi, M. Imani, and T. Rosing, "Workload-aware opportunistic energy efficiency in multi-fpga platforms," *arXiv preprint arXiv:1908.06519*, 2019.
- [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] Predictive technology model. [Online]. Available: <http://ptm.asu.edu/>
- [25] S. Yazdanshenas, K. Tatsumura, and V. Betz, "Don't forget the memory: Automatic block ram modelling, optimization, and architecture exploration," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 2017, pp. 115–124.
- [26] T. Tuan, S. Kao, A. Rahman, S. Das, and S. Trimberger, "A 90nm low-power fpga for battery-powered applications," in *Proceedings of the 2006 ACM/SIGDA 14th international symposium on Field programmable gate arrays*. ACM, 2006, pp. 3–11.
- [27] "Stratix iv device handbook." Datasheet, Intel, September 2014.
- [28] Nangate open cell library. [Online]. Available: <http://nangate.com/>
- [29] "Xilinx power estimator user guide," User Guide, Xilinx, 2018.
- [30] J. Luu, J. Goeders, M. Wainberg, A. Somerville, T. Yu, K. Nasartschuk *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.
- [31] D. Lewis, E. Ahmed, D. Cashman, T. Vanderhoek, C. Lane, A. Lee *et al.*, "Architectural enhancements in stratix-iii and stratix-iv," in *Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays*. ACM, 2009, pp. 33–42.
- [32] 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.
- [33] R. Zhang, M. R. Stan, and K. Skadron, "Hotspot 6.0: Validation, acceleration and extension," *University of Virginia, Tech. Rep*, 2015.
- [34] S. Velusamy, W. Huang, J. Lach, M. Stan, and K. Skadron, "Monitoring temperature in fpga based socs," in *2005 International Conference on Computer Design*. IEEE, 2005, pp. 634–637.
- [35] "Powerplay early power estimator user guide," User Guide, Intel, February 2017.
- [36] "Timing closure user guide," User Guide, Xilinx, January 2012.
- [37] H. Amrouch, B. Khaleghi, and J. Henkel, "Optimizing temperature guardbands," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2017, pp. 175–180.
- [38] "Intel fpga temperature sensor ip core user guide," User Guide, Intel, May 2018.
- [39] E. A. Burton, G. Schrom, F. Paillet, J. Douglas, W. J. Lambert, K. Radhakrishnan *et al.*, "Fivrfully integrated voltage regulators on 4th generation intel® core socs," in *2014 IEEE Applied Power Electronics Conference and Exposition-APEC 2014*. IEEE, 2014, pp. 432–439.
- [40] S. Tian and J. Szefer, "Temporal thermal covert channels in cloud fpgas," in *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 2019, pp. 298–303.
- [41] T. Ebi, D. Kramer, W. Karl, and J. Henkel, "Economic learning for thermal-aware power budgeting in many-core architectures," in *Proceedings of the seventh IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*. ACM, 2011, pp. 189–196.
- [42] H. Amrouch, B. Khaleghi, A. Gerstlauer, and J. Henkel, "Reliability-aware design to suppress aging," in *2016 53rd ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2016, pp. 1–6.
- [43] J. Zhang, K. Rangineni, Z. Ghodsi, and S. Garg, "Thundervolt: enabling aggressive voltage underscaling and timing error resilience for energy efficient deep learning accelerators," in *Proceedings of the 55th Annual Design Automation Conference*. ACM, 2018, p. 19.
- [44] M. Imani, A. Rahimi, D. Kong, T. Rosing, and J. M. Rabaey, "Exploring hyperdimensional associative memory," in *2017 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2017, pp. 445–456.
- [45] P. A. Jamieson and K. B. Kent, "Odin ii: an open-source verilog hdl synthesis tool for fpga cad flows," in *Proceedings of the 18th annual ACM/SIGDA international symposium on Field programmable gate arrays*. ACM, 2010, pp. 288–288.
- [46] R. Brayton and A. Mishchenko, "Abc: An academic industrial-strength verification tool," in *International Conference on Computer Aided Verification*. Springer, 2010, pp. 24–40.
- [47] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [48] J. J. Zhang and S. Garg, "Fate: fast and accurate timing error prediction framework for low power dnn accelerator design," in *Proceedings of the International Conference on Computer-Aided Design*. ACM, 2018, p. 24.
- [49] M. Schmuck, L. Benini, and A. Rahimi, "Hardware optimizations of dense binary hyperdimensional computing: Rematerialization of hyper-vectors, binarized bundling, and combinational associative memory," *arXiv preprint arXiv:1807.08583*, 2018.
- [50] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," 2007.
- [51] A. T. Elthakeb, P. Pilligundla, A. Yazdanbakhsh, F. Miresghallah, and H. Esmailzadeh, "Releg: A reinforcement learning approach for deep quantization of neural networks," *arXiv preprint arXiv:1811.01704*, 2018.