An Experimental Study of Reduced-Voltage Operation in Modern FPGAs for Neural Network Acceleration

Behzad Salami[1] Erhan Baturay Onural[2] Ismail Emir Yuksel[2] Fahrettin Koc[2] Oguz Ergin[2]
Adrián Cristal Kestelman[1,3] Osman S. Unsal[1] Hamid Sarbazi-Azad[4] Onur Mutlu[5]

[1]*BSC* [2]*TOBB ETÜ* [3]*UPC and CSIC-IIIA* [4]*SUT and IPM* [5]*ETH Zürich*

Abstract

We empirically evaluate an undervolting technique, i.e., underscaling the circuit supply voltage below the nominal level, to improve the power-efficiency of Convolutional Neural Network (CNN) accelerators mapped to Field Programmable Gate Arrays (FPGAs). Undervolting below a safe voltage level can lead to timing faults due to excessive circuit latency increase. We evaluate the reliability-power trade-off for such accelerators. Specifically, we experimentally study the reduced-voltage operation of multiple components of real FPGAs, characterize the corresponding reliability behavior of CNN accelerators, propose techniques to minimize the drawbacks of reduced-voltage operation, and combine undervolting with architectural CNN optimization techniques, i.e., quantization and pruning. We investigate the effect of environmental temperature on the reliability-power trade-off of such accelerators.

We perform experiments on three identical samples of modern Xilinx ZCU102 FPGA platforms with five state-of-the-art image classification CNN benchmarks. This approach allows us to study the effects of our undervolting technique for both software and hardware variability. We achieve more than 3X power-efficiency (GOPs/W) gain via undervolting. 2.6X of this gain is the result of eliminating the voltage guardband region, i.e., the safe voltage region below the nominal level that is set by FPGA vendor to ensure correct functionality in worst-case environmental and circuit conditions. 43% of the power-efficiency gain is due to further undervolting below the guardband, which comes at the cost of accuracy loss in the CNN accelerator. We evaluate an effective frequency underscaling technique that prevents this accuracy loss, and find that it reduces the power-efficiency gain from 43% to 25%.

1. Introduction

Deep Neural Networks (DNNs) and specifically Convolutional Neural Networks (CNNs) have recently attained significant success in image and video classification tasks. They are fundamental for state-of-the-art real-world applications running on embedded systems as well as data centers. These neural networks learn a model from a dataset in their training phase and make predictions on new, previously-unseen data in their classification phase. However, their power-efficiency is inherently the primary concern due to the massive amount of data movement and computational power required. Thus, the scalability of CNNs for enterprise applications and their deployment in battery-limited scenarios, such as in drones and mobile devices, are crucial concerns.

Typically, hardware acceleration using Graphics Processing Units (GPUs) [135], Field Programmable Gate Arrays (FPGAs) [101, 85], or Application-Specific Integrated Circuits (ASICs) [102, 42, 22] leads to a significant reduction in CNN power consumption [109]. Among these, FPGAs are rapidly becoming popular and are expected to be used in 33% of modern data centers by 2020 [28]. This increase in the popularity of FPGAs is attributed to their power-efficiency compared to GPUs, their flexibility compared to ASICs, and recent advances in High-Level Synthesis (HLS) tools that significantly facilitate easier mapping of applications on FPGAs [84, 114, 92, 93, 94, 82, 6]. Hence, major companies, such as Amazon [44] (with EC2 F1 cloud) and Microsoft [29] (with Brainwave project), have made large investments in FPGA-based CNN accelerators. However, recent studies show that FPGA-based accelerators are at least 10X less power-efficient compared to ASIC-based ones [12, 73, 74]. In this paper, we aim to bridge this power-efficiency gap by empirically understanding and leveraging an effective undervolting technique for FPGA-based CNN accelerators.

Power-efficiency of state-of-the-art CNNs generally improves via architectural-level techniques, such as quantization [137] and pruning [67]. These techniques do not significantly compromise CNN accuracy as they exploit the sparse nature of CNN applications [80, 3, 134]. To further improve the power-efficiency of FPGA-based CNN accelerators, we propose to employ an orthogonal hardware-level approach: undervolting (*i.e.,* circuit supply voltage underscaling) below the nominal/default level (V_{nom}), combined with the aforementioned architectural-level techniques. FPGA vendors usually add a voltage guardband to ensure the correct operation of FPGAs under the worst-case circuit and environmental conditions. However, these guardbands can be very conservative and unnecessary for state-of-the-art applications. Supply voltage underscaling below the nominal level was already shown to provide significant efficiency improvements in CPUs [78], GPUs [138, 66], ASICs [17], and DRAMs [18, 50]. This paper extends such studies to FPGAs. Specifically, we study the classification phase of FPGA-based CNN accelerators, as this phase can be repeatedly used in power-limited edge devices (unlike the training phase, which is invoked much less frequently). Unlike simulation-based approaches that may

 DOI 10.1109/DSN48063.2020.00032

not be accurate enough [132, 90], our study is based on real off-the-shelf FPGA devices.

The extra voltage guardband can range between 12-35% of the nominal supply voltage of modern CPUs [78], GPUs [138], and DRAM chips [18]. Reducing the supply voltage in this guardband region does *not* lead to reliability issues under normal operating conditions, and thus, eliminating this guardband can result in a significant power reduction for a wide variety of real-world applications. We experimentally demonstrate a large voltage guardband for modern FPGAs: an average of 33% with a slight variation across hardware platforms and software benchmarks. Eliminating this guardband leads to significant power-efficiency ($GOPs/W$) improvement, on average, 2.6X, without any performance or reliability overheads. With further undervolting, the power-efficiency improves by an extra 43%, leading to a total improvement of more than 3X. This additional gain does not come for free, as we observe exponentially-increasing CNN accuracy loss below the guardband region. With further undervolting below this guardband, our experiments indicate that the minimum supply voltage at which the internal FPGA components could be functional (V_{crash}) is equal to, on average, 63% of V_{nom}. Further reducing the supply voltage results in system crash.

We evaluate our undervolting technique on three identical samples of the Zynq-based ZCU102 platform [125], a representative modern FPGA from Xilinx. However, we believe that our experimental observations are applicable to other FPGA platforms as well, perhaps with some minor differences. We previously showed benefits of reduced-voltage operation for on-chip memories on different, older FPGA platforms [96]. Other works observed similar behavior for different types of CPUs [78], GPUs [138], and DRAM chips [18]. In this paper, we characterize the power dissipation of FPGA-based CNN accelerators under reduced-voltage levels and apply undervolting to improve the power-efficiency of such accelerators.[1]

We experimentally evaluate the effects of reduced-voltage operation in on-chip components of the FPGA platform, including Block RAMs (BRAMs) and internal FPGA components, containing Look-Up Tables (LUTs), Digital Signal Processors (DSPs), buffers, and routing resources.[2] We perform our experiments on five state-of-the-art CNN image classification benchmarks, including VGGNet [106], GoogleNet [110], AlexNet [51], ResNet [35], and Inception [110]. This enables us to experimentally study the workload-to-workload variation on the power-reliability trade-offs of FPGA-based CNN accelerators. Specifically, we extensively characterize the reliability behavior of the studied benchmarks below the guardband level and evaluate a frequency underscaling technique to prevent the accuracy loss in this voltage region. Our study also examines the effects of architectural quantization and pruning techniques with reduced-voltage FPGA operation. Finally, we experimentally evaluate the effect of environmental temperature variation on the power-reliability behavior of FPGA-based CNN accelerators.

[1] Our exploration of the FPGA voltage behavior and the subsequent power-efficiency gain is applicable to any application.

[2] These internal FPGA components share a single voltage rail in the studied FPGA platform. To our knowledge, such voltage rail sharing is a typical case for most modern FPGA platforms.

1.1. Contributions

To our knowledge, for the first time, this paper experimentally studies the power-performance-accuracy characteristics of CNN accelerators with greatly reduced supply voltage capability implemented in real FPGAs. In summary, we achieve a total of more than 3X power-efficiency improvement for FPGA-based CNN accelerators. We gain insights into the reduced-voltage operation of such accelerators and, in turn, the effect of FPGA supply voltage on the power-reliability trade-off. We make the following major contributions:

- We characterize the power consumption of FPGA-based CNN accelerators across different FPGA components. We identify that the internal on-chip components, including processing elements, contribute to a vast majority of the total power consumption. We reduce this source of power consumption via our undervolting technique.
- We improve the power-efficiency of FPGA-based CNN accelerators by more than 3X, measured across five state-of-the-art image classification benchmarks. 2.6X of the power-efficiency gain is due to eliminating the voltage guardband, which we measure to be on average 33%. An additional 43% gain is due to further undervolting below the guardband, which comes at the cost of CNN accuracy loss.
- We characterize the reliability behavior of FPGA-based CNN accelerators when executed below the voltage guardband level and observe an exponential reduction in CNN accuracy as voltage reduces. We observe that workloads with more parameters, *e.g.,* ResNet and Inception, are relatively more vulnerable to undervolting-related faults.
- To prevent CNN accuracy loss below the voltage guardband level, we combine voltage underscaling with frequency underscaling. We experiment with a supply voltage lower than V_{nom} and with operating frequency $F_{op} < F_{max}$. Our experiments show that the most *energy-efficient* operating point is the one with the maximum frequency and minimum safe voltage, namely, V_{min}. However, lower voltage and lower frequency lead to better *power-efficiency*.
- We combine voltage underscaling with the existing CNN quantization and pruning techniques and study the power-reliability trade-off of such optimized FPGA-based CNN accelerators. We observe that these bit/parameter-size reduction techniques (quantization and pruning) slightly increase the vulnerability of a CNN to undervolting-related faults; but, they deliver significantly higher power-efficiency when integrated with our undervolting technique.
- We study the effect of environmental temperature on the power-reliability trade-off of reduced-voltage FPGA-based CNN accelerators. We observe that temperature has a direct

effect on the power consumption of such accelerators. However, at very low voltage levels, this effect is not noticeable.
- We evaluate the effect of hardware platform variability by repeating our experiments on three identical samples of the Xilinx ZCU102 FPGA platform. We find large voltage guardbands in all platforms (an average of 33%), *i.e.,* $V_{min} = 0.67 * V_{nom} = 570mV$. However, across three FPGAs, we observe a variation on V_{min}, *i.e.,* $\Delta V_{min} = 31mV$. This variation can be due to process variation. Our results show that the variation of guardband regions across different CNN workloads is insignificant.

2. Background

In this section, we briefly introduce the most important concepts used in this paper, including the architecture of CNNs as well as the undervolting technique.

2.1. Convolutional Neural Networks (CNNs)

DNNs are a class of Machine Learning (ML) methods that are designed to classify unseen objects or entities using non-linear transformations applied to input data [53]. DNNs are composed of biologically inspired neurons, interconnected to each other. Among different DNN models, multi-layer CNNs are a common type, which has recently shown acceptable success in classification tasks for real-world applications.

2.1.1. Phases of a CNN: Training and Classification. A CNN model encompasses two stages: training and classification (inference). Training learns a model from a set of training data. It is an iterative, usually a single-time (or relatively infrequently-executed) step, including backward and forward phases. It adjusts the CNNs parameters, *i.e.,* weights and biases, which determine the strength of the connections between different neurons across CNN layers. The training phase minimizes a loss function, which directly relates to the accuracy of the neural network in the classification phase. In contrast, inference is a post-training phase that aims to classify unknown data, using the trained network model. The inference phase is more frequently executed in edge devices with power-constrained environments. The target of this paper is the inference stage, similar to many existing efforts on the acceleration of CNNs [32, 109, 50].

2.1.2. Internal Architecture of a CNN. A CNN is composed of multiple processing layers such as Convolution, Pooling, Fully-Connected, and SoftMax for feature extraction with various abstractions. Other customized layers can be used case by case for more optimized feature extraction, such as Batch Normalization [71]. The functionality of each type of layer depends on the way in which the neurons are interconnected. Convolution layers generate a more profound abstraction of the input data, called a feature map. Following each Convolution layer, there is usually a Max/Avg Pooling layer to reduce the dimensionality of the feature map. Successive multiple Convolution and Pooling layers generate in-depth information from the input data. Afterward, Fully-Connected layers are typically applied for classification purposes. Finally, the SoftMax layer generates the class probabilities from the class scores in the output layer. Between layers, there are activation functions, such as Relu or Sigmoid, to add non-linear properties to the network. The required computations of different layers are translated to matrix multiplication computations. Thus, matrix multiplication optimization techniques, such as FFT or Strassen [52], can be applied to accelerate the inference implementation. Matrix multiplication is an ideal application to take advantage of parallel and data flow execution model used in FPGA-based hardware accelerators.

2.1.3. Architectural Optimizations. To improve the power-efficiency of CNNs, two most commonly-used architectural-level techniques are quantization [136] and pruning [67].[3] These two techniques rely on the sparse nature of CNNs, *i.e.,* a vast majority of CNN computations are unnecessary. Quantization aims to reduce the complexity of high-precision CNN computation units by substituting selected floating-point parameters with low-precision fixed-point. Pruning aims to reduce the model size by eliminating unnecessary weight/neurons/connections of a CNN. These architectural techniques are applicable to any underlying hardware. There are numerous extensions of quantization [136, 137] and pruning [129, 36] techniques. In our experiments, we integrate typical quantization [34] and pruning [33] techniques with our proposed hardware-level undervolting technique to further improve the power-efficiency of FPGA-based CNN accelerators.

2.2. Undervolting: Supply Voltage Underscaling Below the Nominal Voltage Level

The total power consumption of any hardware substrate is directly related to its supply voltage: quadratically and linearly with dynamic and static power, respectively. Thus, supply voltage underscaling toward the threshold voltage significantly reduces power consumption. Voltage underscaling is a common power-saving approach as manufacturing technology node size reduces. For instance, the V_{nom} of Xilinx FPGAs is $1V$, $0.9V$, and $0.85V$ for implementations in 28nm, 20nm, and 16nm technology nodes, respectively. The aim of our undervolting technique is to reduce the supply voltage below the default V_{nom}. However, circuit latency can increase substantially when supply voltage is reduced below the guardband level, and in turn, timing faults can appear. These timing faults are manifested as bit-flips in memories or logic timing violations in data paths. They can potentially cause the application to produce wrong results, leading to reduced accuracy in CNNs, or, in the worst-case, they may cause system crashes. There are several approaches to deal with undervolting faults, such as preventing these faults by: *i)* simultaneously decreasing the frequency [111], which has an associated performance degradation cost, *ii)* fixing the faults by using fault mitigation techniques, such as Error Correction Codes (ECCs) for mem-

[3] There are also other techniques, such as batching [104], loop unrolling [130], and memory compression [49].

ories [9, 99] and Razor shadow latches for data paths [27], which comes at the cost of extra hardware, or *iii)* architectural improvements, such as additional iterations in CNN training [133] that may incur hardware and/or software adaptation costs.

There are two approaches to undervolting studies: *i)* simulation-based studies [89, 127, 132, 108], or *ii)* direct implementation or testing on real hardware fabrics, mainly performed on CPUs, GPUs, ASICs, and DRAMs [138, 9, 78, 18, 81, 50]. The simulation-based approach requires less engineering effort. However, validation of simulation results on real hardware is the primary concern with such an approach. In contrast, the real hardware evaluation approach requires substantial engineering effort, and it is device- and vendor-dependent. Such a real hardware approach leads to exact experimental results and it provides an opportunity to study device-dependent parameters, such as voltage guardbands and real power and reliability behavior of underlying hardware. In this paper, we follow the real hardware approach by evaluating undervolting on real modern off-the-shelf FPGA devices for state-of-the-art CNN workloads and benchmarks.

3. Experimental Methodology

Figure 1 depicts the overall methodological flow of our experiments. In this section, we elaborate on its different aspects, including our implementation methodology, benchmarks, and undervolting methodology of our FPGA platform.

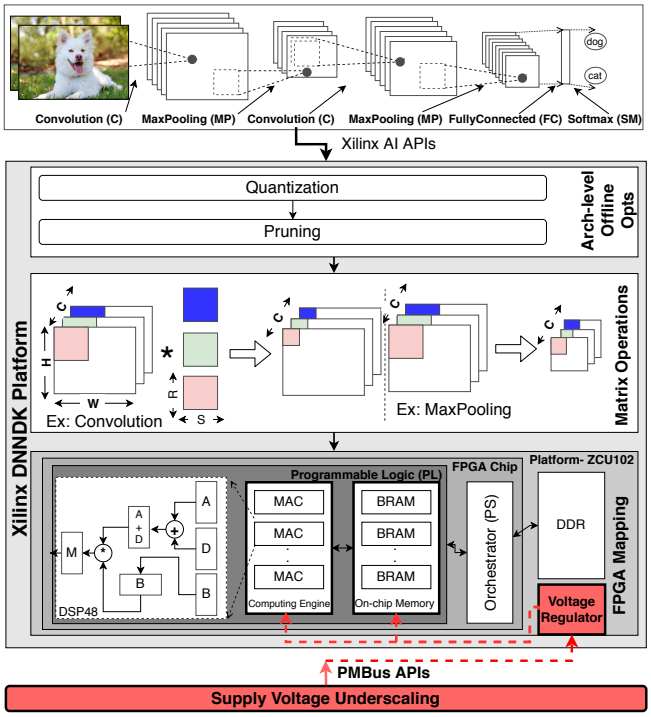


Figure 1: Our overall methodology (for simplicity, we show a simplified block diagram of Xilinx DNNDK

3.1. CNN Model Development Platform

For our implementation, we leverage the Deep Neural Network Development Kit (DNNDK) [122], a CNN framework from Xilinx. DNNDK is an integrated framework to facilitate CNN development and deployment on Deep learning Processing Units (DPUs). In this paper, we use DNNDK as it is a freely-available framework instead of a specialized custom design, to ensure that the results reported in this paper are reproducible and general-enough for state-of-the-art CNN implementations. Although we do not expect a significant difference by experimenting on DNNDK versus other DNN platforms, our future plan is to verify this by repeating the experiments on other platforms, such as DNNWeaver [101]. DNNDK provides a complete set of toolchains with compression, compilation, deployment, and profiling, for the mapping of CNN classification phases onto FPGAs integrated with hard CPU cores via a comprehensive and easy-to-use C/C++ programming interface.

Among the components of DNNDK, the DEep ComprEssioN Tool (DECENT) is responsible for quantization and pruning tasks. The quantization utility of DECENT can convert a floating-point CNN model to a quantized model with the precision of at most INT8 [34]. The pruning utility aims to minimize the model size by removing unnecessary connections of the CNN [33]. We perform our baseline evaluation on a model with INT8 precision and without any pruning optimization. However, in Section 6, we evaluate different configurations to provide a more comprehensive analysis. There are different sizes of soft DPUs provided by DNNDK with various hardware utilization rates [123]. Among them, B4096 is the largest model that utilizes a maximum fraction of BRAMs and DSPs, *i.e.,* 24.3% and 25.6%, respectively, resulting in a peak performance of 4096 operations/cycle with a default DPU frequency of 333Mhz and DSP frequency of 666Mhz. In total, a maximum of three B4096 DPUs can be used in the hardware platform evaluated in this paper. Our experiments are based on the B4096 configuration to achieve peak performance.

3.2. CNN Benchmarks

We evaluate undervolting in FPGA-based CNN accelerators with five commonly-used image classification benchmarks, shown in Table 1: VGGNet [106], GoogleNet [110], AlexNet [51], ResNet [35], and Inception [110]. To perform a comprehensive analysis and study workload-to-workload variation better, we choose models whose parameter sizes vary from a few MBs, *e.g.,* GoogleNet, to hundreds of MBs, *e.g.,* AlexNet. Our benchmarks have different numbers and types of layers, as shown in Table 1. The default activation function used in benchmarks is Relu.

3.3. Undervolting

In this section, we briefly explain the prototype FPGA platform and the associated voltage control setup.

Table 1: Evaluated CNN Benchmarks.

CNN Model	Dataset			Parameters		Inference Accuracy (%)	
	Name	Inputs	Outputs	#Layers	Size	Literature	Our design @Vnom
VGGNet	Cifar-10	32*32	10	6	8.7MB	87% [106]	86%
GoogleNet	Cifar-10	32*32	10	21	6.6MB	91% [110]	91%
AlexNet	Kaggle Dogs vs. Cats	227*227	2	8	233.2MB	96% [51]	92.5%
ResNet50	ILSVRC2012	224*224	1000	50	102.5MB	76% [35]	68.8%
Inception	ILSVRC2012	224*224	1000	22	107.3MB	68.7% [110]	65.1%

3.3.1. Prototype FPGA Platform. Our prototype is based on the Xilinx ZCU102 FPGA platform fabricated at a 16nm technology node. We choose this platform because it is *i)* equipped with voltage underscaling capability, *ii)* supported by DNNDK. We repeat experiments on three identical samples of ZCU102 to study the effect of hardware platform variability. ZCU102 is populated with the Zynq UltraScale+ XCZU9EG-2FFVB1156E MPSoC that combines a Processing System (PS) and user-Programmable Logic (PL) in the same device. The PS part features a quad-core 64-bit ARM Cortex-A53 and is mainly used for the host communication in DNNDK. The PL part has 32.1Mbit of BRAMs, 600K LUTs, and 2520 DSPs. For the CNN implementation, DPUs are mapped into the PL side. As mentioned earlier, our baseline hardware configuration employs three B4096 DPUs, the maximum possible number, leading to a maximum utilization fraction of more than 75% for BRAMs and DSPs. ZCU102 is equipped with an 8GB 64-bit DDR-4 off-chip memory. In our implementation, this memory contains input images and CNN parameters. It is also used for interfacing purposes with the host.

3.3.2. Undervolting Methodology. Unfortunately, there is no voltage scaling standard for FPGAs. Different vendors have their unique voltage management methodologies. Moreover, there are some platforms without voltage scaling capability, such as the Xilinx Zedboard [7]. Even a single vendor's different devices do not necessarily have the same voltage distribution model. Although this non-standard approach of vendors adds some constraints to experimental studies, such as the one conducted in this paper, we believe that, with minor changes, the methodology we explain below for ZCU102 can be applicable to other platforms, as, for instance, we previously studied for on-chip memories of older FPGA generations [96].

Figure 2, adapted from [125], depicts the voltage distribution model of ZCU102. Here, the voltage scaling capability is provided using an on-board voltage regulator that can convert an input voltage level of $12V$ into different voltage levels. The voltage level of the output lines, usually called voltage rails, is fully configurable and also addressable using the Power Management Bus (PMBus) standard [83]. Each voltage rail feeds one or more components of the FPGA platform. ZCU102 is equipped with three voltage regulators, which in total provide 26 voltage rails accessible through the PMBus. In this paper, we focus on on-chip voltage rails: V_{CCINT} and V_{CCBRAM}, as shown in Figure 2. V_{CCINT} is accessible with PMBus address 0x13 and $V_{nom} = 850mV$; it supplies multiple PL components, including DSPs, LUTs, buffers, and routing resources. V_{CCBRAM} is accessible with PMBus address 0x14 and $V_{nom} = 850mV$; it supplies the BRAMs of the PL. To access these voltage rails for monitoring and regulation, we use a PMBus adapter and the provided API [65]. Using a similar approach and different PMBus commands, we monitor the power consumption of each voltage rail as well as the on-chip temperature.

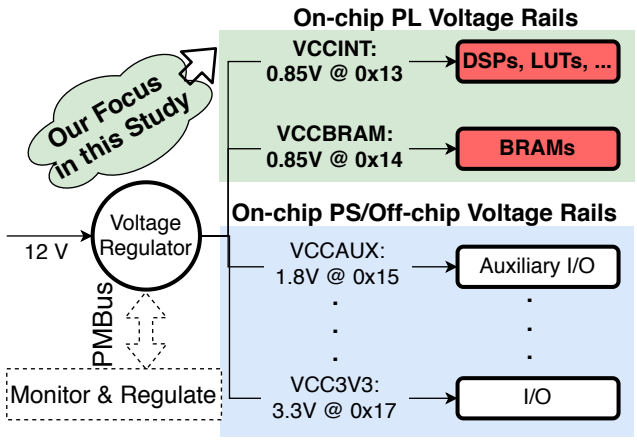


Figure 2: Voltage distribution on the Xilinx ZCU102 FPGA, adapted from [125].

4. Experimental Results

We present and analyze our experimental results from reduced-voltage operation on FPGA boards. These results are collected at ambient temperature. Section 7 presents further temperature analysis. Each result presented in this paper is the average of 10 experiments, in order to account for any variation between different experiments; although, the variation we observed was negligible.

4.1. Power Analysis of FPGA-based CNN Accelerators at the Nominal Voltage Level (V_{nom})

We measure the total on-chip power consumption of the baseline configuration to be an average of $12.59W$ for benchmarks, at the nominal voltage level (V_{nom}) and ambient temperature. This value includes the power consumption at on-chip voltage rails, including V_{CCBRAM} and V_{CCINT}. We observe that

internal FPGA components on the V_{CCINT} rail dissipate more than 99.9% of this on-chip power. We believe this observation is due to power-efficient BRAM designs, using techniques like dynamic power gating [124], in modern Ultrascale+ FPGA platforms, including in the studied ZCU102 FPGA. Older generations of Xilinx FPGAs like the 7-series are not equipped with this capability [121]. Thus, for such older devices, BRAM power consumption was the main source of FPGA power consumption, as shown in previous studies [96, 97, 99, 1]. For the rest of the paper, as we study the power-reliability trade-off, we concentrate on V_{CCINT} due to its dominance in FPGA power consumption.

4.2. Overall Voltage Behavior

Our experiments reveal that a large voltage guardband below V_{nom} exists for V_{CCINT}, as shown in Figure 3 for three hardware platforms and five CNN benchmarks. In the voltage guardband region, as we reduce supply voltage there is no performance or reliability degradation, and thus, under normal conditions, eliminating this voltage guardband can lead to significant power savings without any overhead. As Figure 3 shows, we measure the average guardband amount to be $850mV - 570mV = 280mV$, with a slight variation across different benchmarks. In other words, we observe that $V_{min} = 570mV$ (on average) is the minimum safe voltage level of the accelerator, where there is no accuracy loss. As we further undervolt below V_{min}, we enter a region called the *critical region* in which the reliability of the hardware and, in turn, the accuracy of the CNN starts to decrease significantly. As Figure 3 depicts, we measure the average critical voltage region size, to be $570mV - 540mv = 30mV$, with a slight variation across different benchmarks. As we further undervolt below V_{min}, we reach a point at which the FPGA does not respond to requests and it is not functional. This point is called V_{crash}. We find that $V_{crash} = 540mV$ on average, with a slight variation across different hardware platforms.

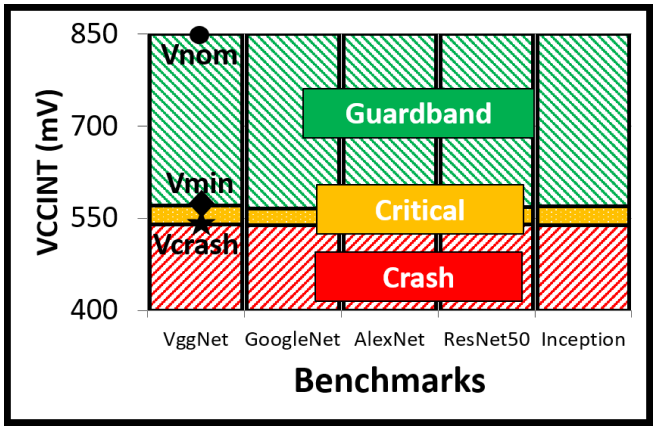


Figure 3: Voltage regions with a slight workload-to-workload variation (averaged across three hardware platforms).

Figure 4 illustrates the overall behavior we observe for the power-efficiency and CNN accuracy trade-off on our FPGA-based CNN accelerator. As we perform undervolting, the FPGA enters the guardband region, where we observe no reliability degradation (*i.e.,* CNN accuracy loss), and therefore, the power-efficiency comes with no cost. We observe this behavior until we reach the point V_{min}, *i.e.,* minimum safe voltage level. With further undervolting, the FPGA enters the critical region, where power-efficiency constantly increases, but we start to observe fast-increasing CNN accuracy loss. When we undervolt down to a specific point, called V_{crash}, the FPGA becomes non-functional and starts to hang. Sections 4.3 and 4.4 provide more details on the power-reliability trade-off. Our demonstration is on three identical samples of Xilinx ZCU102. However, we believe that the overall voltage behavior, illustrated in Figure 4, is reproducible for other FPGA platforms as well.

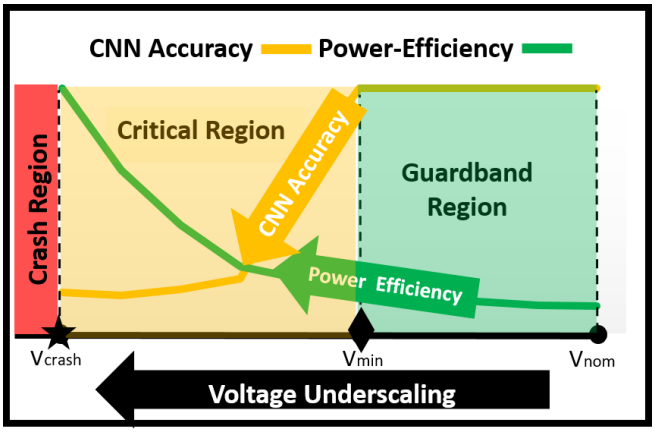


Figure 4: Overall voltage behavior observed for V_{CCINT}.

4.3. Detailed Power-Efficiency Analysis

Figure 5 presents the power-efficiency experimental results ($GOPs/W$) for five CNN workloads, averaged across three FPGA hardware platforms. The power-efficiency gain at V_{crash} is more than 3X of that at nominal voltage level, *i.e.,* V_{nom}, for the same design of the given CNN accelerator. 2.6X of the gain in power-efficiency is the result of eliminating the voltage guardband without any CNN accuracy loss. 43% further power-efficiency gain is due to further undervolting in the critical region, which has an associated CNN accuracy loss cost.

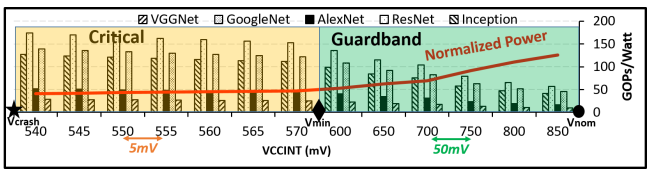


Figure 5: Power-efficiency ($GOPs/W$) improvement via undervolting (averaged across three hardware platforms).

The power-efficiency gain via undervolting until V_{min} is not application-dependent, so it is useful for any application mapped onto the same FPGA. However, the reliability overhead in the critical region below V_{min} is application-dependent due to different vulnerability levels of different applications/workloads.

4.4. Detailed Reliability Analysis

As we undervolt until V_{min}, there is no reliability overhead. However, as we further undervolt below V_{min}, the reliability of the hardware is significantly affected due to the further increase in datapath delay. The effect of the reliability loss is fully application-dependent due to different inherent resilience levels of different applications. In this paper, we study this effect on several CNN workloads. Figure 6 depicts our experimental results. As shown before, as we reduce the supply voltage, power-efficiency improves. When we reduce the supply voltage below V_{min}, we observe that the accuracy of all benchmarks gradually reduces. With further undervolting, when the supply voltage reaches an average of $V_{crash} = 540mV$ across different platforms and benchmarks, the accuracy of the benchmarks drops greatly, and the classifier behaves randomly. Our experiments show that benchmarks with more parameters, *e.g.,* ResNet and Inception are relatively more vulnerable to undervolting faults below V_{min}. Also, as seen, there is a variation of $\Delta V_{min} = 31mV$ and $\Delta V_{crash} = 18mV$ across different FPGAs. This variation can be due to the process variation across different FPGAs.

5. Frequency Underscaling

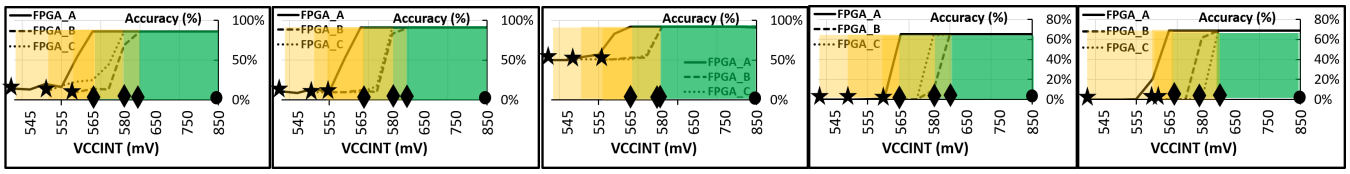
As shown earlier, in the critical voltage region below the guardband, CNN classification accuracy dramatically decreases. In this section, we aim to overcome this accuracy loss by exploiting frequency underscaling. To be more precise, we aim to find a more energy-efficient voltage setting than the undervolted V_{min}, which also provides accurate results. To this end, for each supply voltage setting below V_{min}, we aim to identify the maximum frequency value F_{max} with which the system does not experience any accuracy loss. When we find this frequency point, we evaluate the energy efficiency of the system. As we underscale the frequency of the system, the performance of the application reduces. Therefore, we use the $GOPs/J$ metric as it accommodates for both performance and energy consumption.

Table 2 summarizes the results of the frequency underscaling in the critical region. These experiments are based on frequency and voltage steps of $25Mhz$ and $5mV$, respectively. The column V_{CCINT} corresponds to the supply voltage of a given setting. The column F_{max} corresponds to the maximum frequency at which there is no accuracy loss. The remaining columns: $GOPs$, $Power$, $GOPS/W$, $GOPS/J$ are normalized to the respective values of executing the system in the default setting $V_{CCINT} = V_{min} = 570mV, F_{max} = 333Mhz$ which are the baseline settings of our accelerator. Table 2 indicates that multiple voltage settings V_{CCINT} map to the same operating Frequency F_{max}: supply voltages between $560mV$ to $545mV$ require the same frequency of $F_{max} = 250Mhz$. This is because the frequency step we use is $25Mhz$. Using smaller steps of frequency can lead to more spread-out F_{max} values.

Table 2: Evaluation of frequency underscaling to prevent CNN accuracy loss in the critical voltage region (averaged across three hardware platforms). Best result with frequency underscaling in terms of each metric is marked in blue.

V_{CCINT} (mV)	F_{max} (Mhz)	$GOPs$ (Norm)	$Power(W)$ (Norm)	$GOPs/W$ (Norm)	$GOPs/J$ (Norm)
570	333	**1.00**	1.00	1.00	**1.00**
565	300	0.94	0.97	0.97	0.87
560	250	0.83	0.84	0.99	0.75
555	250	0.83	0.78	1.06	0.80
550	250	0.83	0.75	1.10	0.83
545	250	0.83	0.74	1.12	0.84
540	200	0.70	**0.56**	**1.25**	0.75

For all the combinations of (V_i, F_i) that provide error-free results presented in Table 2 in the critical region, power decreases with decreasing $V_i < V_{min}$ and $F_i < F_{max}$. This is because we decrease both the supply voltage and the operating frequency. However, at the same time, this leads to decreasing the system performance. Consequently, the best voltage-frequency combination in terms of *energy-efficiency* ($GOPs/J$) is the one with the highest frequency of $F_{max} = 333Mhz$, which also is our baseline. In other words, it is not worth to underscale the frequency and voltage to find a more energy-efficient optimal point. However, as a trade-off, the design is more *power-efficient* (*i.e.,* has higher $GOPs/W$) at lower voltage-frequency levels, up to 25% at $V_{crash} = 540mV$.



(a) VGGNet. (b) GoogleNet. (c) AlexNet. (d) ResNet. (e) Inception.

Figure 6: Effect of reduced supply voltage on the accuracy of CNN workloads (separately for three hardware platforms). (V_{nom} : ●, V_{min} : ♦, V_{crash} : ★)

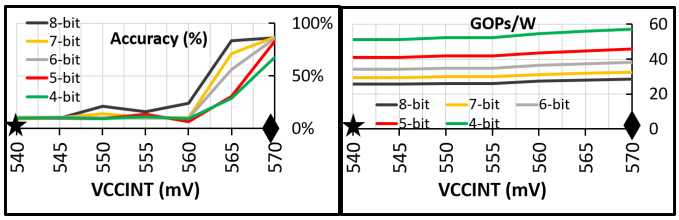
6. Combining Undervolting with Architectural CNN Optimization Techniques

In this section, we experimentally evaluate undervolting for employing the CNN's quantization and pruning techniques. Via experiments, we observe that these bit reduction techniques can deliver additional power-efficiency gains proportional to the quantization/pruning level. However, applying these techniques can slightly increase the vulnerability of CNNs to undervolting-related faults. This section reports results for VGGNet as we observe similar results for other workloads.

6.1. Quantization

Our baseline is optimized with INT8 precision. As shown in Table 1, this precision does not incur any significant accuracy loss in comparison to baseline models that use floating-point precision. For further analysis of the effect of undervolting with lower precision models, we evaluate INT7, INT6, INT5, and INT4 precisions. Using DNNDK, we observe significant accuracy loss for INT3, INT2, and INT1 when executed at V_{nom}. Thus, we do not present them in this paper.

Figure 7 shows results of different precisions (INT8 to INT4). We find that *i)* when operating at reduced-voltage levels, accuracy loss is relatively high due to lower precision; *ii)* power-efficiency is proportional to voltage as well as quantization levels. In conclusion, combining low-precision and low-voltage operation can significantly deliver higher power-efficiency. However, it comes at the cost of accuracy loss.

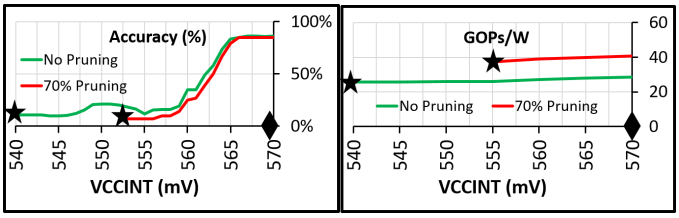


(a) CNN Accuracy. (b) Power-efficiency ($GOPs/W$).

Figure 7: Effect of undervolting at different quantization levels for VGGNet (averaged across three hardware platforms).

6.2. Pruning

Figure 8 shows results of pruned and baseline (without any pruning) models. We find that undervolting-related faults have a relatively more significant effect on the pruned model. However, this comes with higher power-efficiency of the pruned model, as shown in Figure 8b, due to fewer operations in the pruned model. With undervolting, power consumption reduces for both pruned and baseline models, at a similar rate. V_{crash} is different for the pruned model. Specifically, the pruned version demonstrates a higher V_{crash} voltage equal to $555mV$ in contrast to the baseline V_{crash} of $540mV$.



(a) CNN Accuracy. (b) Power-efficiency ($GOPs/W$).

Figure 8: Effect of undervolting on pruned CNN models for VGGNet (averaged across three hardware platforms).

7. Effect of Environmental Temperature

The power consumption of a modern chip, including FPGAs, also depends on temperature. Temperature affects static power consumption. As the external temperature increases, the leakage current and, in turn, the leakage-induced static power increases [11, 38, 47, 39]. As technology node size reduces, a large fraction of power consumption comes from the static power. Therefore, temperature has a larger effect on the power consumption of denser chips [69]. On the other hand, temperature can have a considerable effect on circuit latency [72, 70], *i.e.,*, circuit latency *decreases* as the temperature increases in contemporary technology nodes. Therefore, there are fewer undervolting-related faults at higher temperatures.

To understand the combination of multiple effects mentioned above, we study the effect of the environmental temperature on the power-reliability trade-off of our FPGA-based CNN accelerator under reduced-voltage operation. To this end, we use GoogleNet as a benchmark and undervolt V_{CCINT}. We discuss the voltage behavior in both critical and guardband regions at different temperatures ranging from 34°C to 52°C degrees. To regulate the FPGA temperature, we control the fan speed using the PMBus interface. We also use the same PMBus interface to monitor the on-board live temperature. By doing so, we can test different ambient temperatures ranging from 34°C to 52°C degrees.[4]

7.1. Temperature Effect on Power Consumption

Figure 9 depicts the power consumption of our CNN accelerator when executing GoogleNet with different V_{CCINT} values at different temperatures. Clearly, temperature has a direct effect on power consumption. As temperature increases, power consumption proportionally increases. This is due to increase in static power when the chip heats up. Dynamic power consumption is also affected by temperature, but this effect is almost negligible. Importantly, we observe that the effect of temperature on power consumption reduces for lower voltages. For example power change from 34°C to 52°C are 0.46% and 0.15%, respectively at $V_{CCINT} = 850mV$ and $V_{CCINT} = 650mV$.

[4][34°C, 52°C] is the temperature range that we could generate using the fan speed. Experimenting with wider temperature ranges requires more facilities, which were not available to us.

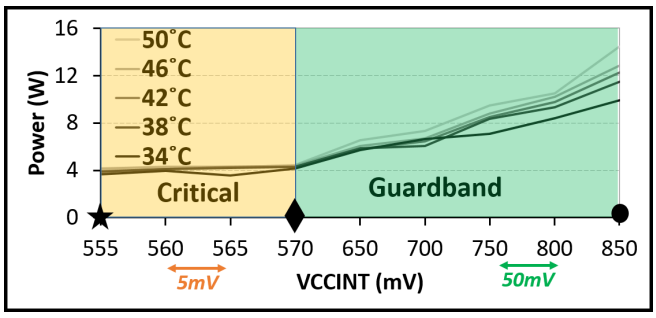


Figure 9: Power consumption of our reduced-voltage CNN accelerator at temperature range of [34°C, 52°C], shown for GoogleNet (averaged across three hardware platforms).

7.2. Temperature Effect on Reliability

Figure 10 shows the effect of temperature on the accuracy of our reduced-voltage CNN accelerator. Our experiment demonstrates that *i)* there is no noticeable change in the size of the guardband and critical regions, and *ii)* higher temperature at a particular voltage level leads to higher CNN accuracy. This is because at higher temperatures, there are fewer undervolting related errors due to decreased circuit latency, an artifact due to the Inverse Thermal Dependence (ITD) property of contemporary technology nodes [72, 113].

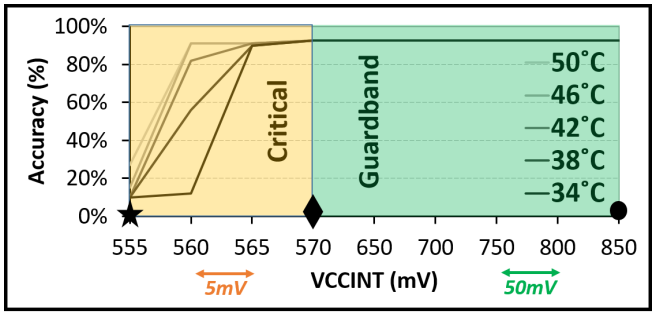


Figure 10: Accuracy of our reduced-voltage CNN accelerator at temperature range of [34°C, 52°C], shown for GoogleNet (averaged across three hardware platforms).

7.3. Discussion

In our setup, considering the power-reliability trade-off discussed, the optimal setting is at Temp=50° and $V_{CCINT} = 565mV$, *i.e.,* the minimum voltage level at which there is almost no accuracy loss due to the healing effect of high temperature. However, the disadvantage of operating at higher temperatures is the overall decrease in lifetime reliability. Below, we summarize our findings on temperature effects.

- There is a negligible change in the value of $V_{min} = 570mV$ across temperatures, and thus, there is no significant change in the guardband region. However, the system crashes relatively earlier over temperature variation. We expect, though, that when the system undergoes a wider temperature range, there will be a more noticeable change in the V_{min} and V_{crash}.
- At any specific voltage point in either region, power consumption directly increases as temperature increases, mainly due to the direct relation of static power consumption and temperature.
- The effect of temperature on power consumption is significantly less at lower voltage levels, due to the relatively lower contribution of static power to total power consumption.
- In the critical voltage region and at any specific voltage level, higher temperature leads to higher CNN accuracy. The power cost of the higher temperature in the critical voltage region is relatively low.

Consequently, a lower voltage can be applied at higher temperatures without causing significant accuracy loss at a small power cost.

8. Related Work

To our knowledge, this paper provides the first study evaluating the effect of reduced-voltage operation in FPGA-based CNN accelerators. In this section, we review related works on *i)* undervolting, *ii)* power-efficient CNNs, and *iii)* reliability of CNNs.

8.1. Undervolting

Supply voltage underscaling below the nominal level is an effective approach to improve the power-efficiency of digital circuits. There are two different approaches to studying undervolting: simulation or real experiments.

8.1.1. Simulation Studies. This approach simulates hardware to study undervolting. It is convenient for early-stage studies as it does not require large engineering effort. However, this approach lacks the information of real hardware, and thus, validation of results is the main concern. Most of the existing simulation-based studies are for CPUs [89, 127, 108, 81] and specifically for CPU components such as caches [2, 118, 119, 126, 23] and branch predictors [20]. There are also studies for ASIC CNN accelerators [86, 132, 5]. Following this approach, studies on FPGA-based designs are either fully in simulation [70] or emulation of FPGA netlists on simulation frameworks [45, 90].

8.1.2. Experimental Studies on Real Hardware. Evaluating undervolting on real hardware is another approach that has recently been considered for multiple devices [31, 79]. Doing so requires relatively more engineering effort as well as considering physical constraints, such as non-standard device- and vendor-dependent voltage distribution models. Yet, the results produced are accurate and can be directly used in real-world applications.

Undervolting of real hardware is studied for various system components, such as CPUs [8, 76, 77, 43, 10], GPUs [138, 55, 56], ASICs [17, 75, 48], DRAMs [18, 19, 50], and Flash disks [14, 15, 16]. These studies focus on voltage guardband analysis, fault characterization, and fault mitigation. Undervolting on real FPGAs is not thoroughly investigated. Very recent works on FPGA undervolting are either accompanied

with frequency underscaling [1, 103] that can diminish performance, or are limited to BRAMs [95, 96, 97, 99, 91]. This paper, for the first time, extends real FPGA undervolting studies to multiple on-chip components of modern FPGA fabrics and evaluates it in-detail on the power-accuracy trade-off of CNN applications.

8.2. Power-efficient CNNs

Many works aim to improve CNN power-efficiency by optimizing the CNN architecture as well as the underlying hardware. In this paper, to achieve significant power-efficiency, we combine our hardware-level FPGA undervolting technique with architectural CNN optimization techniques, including quantization and pruning.

8.2.1. Architectural Techniques. This approach aims to reduce the parameter size of a CNN. The methods of this approach are independent of the underlying hardware, and in theory, they can be applied to any hardware, including hardware accelerators. The most common techniques are quantization [136, 34, 137], pruning [67, 129, 33], batching [104], loop unrolling [130], and memory compression [26, 49]. Among these, quantization and pruning have shown significant efficiency without significantly compromising the CNN accuracy; hence, we focus on them in our experiments.

8.2.2. Hardware-level Techniques. An orthogonal approach to reducing CNN power is to optimize the underlying hardware. To this end, since traditional processor-based architectures are inherently power-hungry and not suitable for CNNs, exploiting a dedicated hardware accelerator is the first approach. Further power savings are possible with low-level techniques, such as undervolting.

- Hardware Accelerators: Data-flow execution models using GPUs [46, 37], FPGAs [101, 107, 120, 64, 60] and ASICs [42, 4, 21, 115] are more efficient choices for CNNs than traditional CPUs. Among these, FPGAs are more flexible compared to ASICs and more efficient than GPUs. Efficient exploitation of the underlying hardware is fundamental for power-efficiency, using techniques like resource partitioning [105] and reuse [131, 88]. Our work uses an industrial tool [122] that inherently exploits these techniques.
- Undervolting: Undervolting has been shown to provide significant power-efficiency benefit for CNNs when applied to SRAMs [17], DRAMs [50], ASICs [68, 132, 128, 17, 48], and heterogeneous systems [24, 25, 100].

8.3. Reliability of CNNs

Although CNNs are inherently resilient to some error rate in data or underlying hardware, high enough error rates can cause significant accuracy loss. Error sources can be harsh environments, process manufacturing defects, undervolting, ionizing particles, noise in data, among others. Hence, CNN reliability is an active research area. Existing studies are based on fault injection or real errors.

8.3.1. Simulation-based Fault Injection. These studies inject randomly-generated faults into CNNs, but they do not consider undervolting [41, 98, 87, 58, 57, 30, 40, 63]. This approach provides an opportunity for comprehensive fault characterization of CNNs, such as the sensitivity of different layers, different location of faults, among others. However, these works do not consider faults in real hardware, which potentially can lead to inaccurate analysis.

8.3.2. Faults in Real Hardware. In real-world applications, such as IoT, airspace, and driver-less cars, CNNs can potentially experience different types of faults. Various works evaluate CNN reliability on faulty real hardware, *e.g.,* soft errors [61, 62, 112, 13] and undervolting in ASICs [59, 17, 116, 117, 54]. This approach requires significant engineering effort but can result in relatively more accurate results. None of these works study CNN reliability on undervolted FPGAs.

9. Summary and Future Work

In this paper, we experimentally evaluated the effects of supply voltage underscaling below the nominal level on real FPGA-based CNN accelerators. We showed that we could improve the power-efficiency of such accelerators by more than 3X via undervolting. 2.6X of the power-efficiency improvement comes from eliminating the voltage guardband (without compromising CNN accuracy), while the remaining 43% improvement comes from undervolting further below the guardband (which comes with CNN accuracy loss). We conclude that undervolting can significantly improve the power-efficiency of FPGA-based neural network accelerators.

As future work, we aim to develop *i)* fault mitigation techniques for very low-voltage regions even when the design operates at the maximum frequency (F_{max}), *ii)* dynamic voltage adjustment techniques considering temperature, accuracy, power consumption, and performance trade-off. We also aim to expand our experiments in hardware, by evaluating more FPGAs, as well as in software, by repeating experiments on other CNN platforms like DNNWeaver [101]. Finally, we believe it is promising to study potential security issues of FPGA-based CNN accelerators under reduced supply voltage levels.

Acknowledgments

We thank the anonymous DSN2020 reviewers for their feedback and comments, as well as Dr. Long Wang, who helped us with shepherding. Also, we thank Dr. Konstantinos Parasyris for his in-depth review of the first version of this paper. The work done for this paper was partially supported by a HiPEAC Collaboration Grant funded by the H2020 HiPEAC Project under grant agreement No. 779656. The research leading to these results has received funding from the European Union's Horizon 2020 Programme under the LEGaTO Project (www.legato-project.eu), grant agreement No. 780681.

References

[1] I. Ahmed et al. Automatic Application-Specific Calibration to Enable Dynamic Voltage Scaling in FPGAs. *TCAD*, 2018.

[2] A. Alameldeen et al. Adaptive cache design to enable reliable low-voltage operation. *TC*, 2010.

[3] J. Albericio et al. CNVLutin: Ineffectual-neuron-free deep neural network computing. In *ISCA*, 2016.

[4] R. Andri et al. YodaNN: An ultra-low power convolutional neural network accelerator based on binary weights. In *ISVLSI*, 2016.

[5] R. Andri et al. YodaNN: An architecture for ultra low-power binary-weight CNN acceleration. *TCAD*, 2017.

[6] O. Arcas-Abella et al. Hardware acceleration for query processing: leveraging FPGAs, CPUs, and memory. *CISE*, 2016.

[7] Avnet. Zedboard Developement Kit http://zedboard.org/product/zedboard. 2020.

[8] A. Bacha et al. Dynamic reduction of voltage margins by leveraging on-chip ECC in Itanium II processors. In *ISCA*, 2013.

[9] A. Bacha et al. Using ECC feedback to guide voltage speculation in low-voltage processors. In *MICRO*, 2014.

[10] R. Bertran et al. Voltage noise in multi-core processors: Empirical characterization and optimization opportunities. In *MICRO*, 2014.

[11] S. Borkar et al. Design challenges of technology scaling. *IEEE Micro*, 1999.

[12] A. Boutros et al. You can not improve what you do not measure: FPGA vs. ASIC efficiency gaps for convolutional neural network inference. *ACM TRETS*, 2018.

[13] R. Brewer et al. The Impact of Proton-Induced Single Events on Image Classification in a Neuromorphic Computing Architecture. *TNS*, 2019.

[14] Y. Cai et al. Threshold voltage distribution in MLC NAND flash memory: Characterization, analysis, and modeling. In *DATE*, 2013.

[15] Y. Cai et al. Read disturb errors in MLC NAND flash memory: Characterization, mitigation, and recovery. In *DSN*, 2015.

[16] Y. Cai et al. Error characterization, mitigation, and recovery in flash-memory-based solid-state drives. *Proceedings of the IEEE*, 2017.

[17] N. Chandramoorthy et al. Resilient Low Voltage Accelerators for High Energy Efficiency. In *HPCA*, 2019.

[18] K. Chang et al. Understanding reduced-voltage operation in modern DRAM devices: Experimental characterization, analysis, and mechanisms. *SIGMETRICS*, 2017.

[19] K. Chang et al. Voltron: Understanding and Exploiting the Voltage-Latency-Reliability Trade-Offs in Modern DRAM Chips to Improve Energy Efficiency. *arXiv:1805.03175*, 2018.

[20] A. Chatzidimitriou et al. Assessing the Effects of Low Voltage in Branch Prediction Units. In *ISPASS*, 2019.

[21] T. Chen et al. DianNao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. In *ISCA*, 2014.

[22] Y. Chen et al. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. In *ISCA*, 2016.

[23] Z. Chishti et al. Improving cache lifetime reliability at ultra-low voltages. In *MICRO*, 2009.

[24] A. Cristal et al. LEGaTO: first steps towards energy-efficient toolset for heterogeneous computing. In *SAMOS*, 2018.

[25] A. Cristal et al. LEGaTO: towards energy-efficient, secure, fault-tolerant toolset for heterogeneous computing. In *CF*, 2018.

[26] C. Deng et al. PermDNN: Efficient compressed DNN architecture with permuted diagonal matrices. In *MICRO*, 2018.

[27] D. Ernst et al. Razor: A low-power pipeline based on circuit-level timing speculation. In *MICRO*, 2003.

[28] M. Feldman. https://www.top500.org/news/good-times-for-fpga-enthusiasts/. 2019.

[29] J. Fowers et al. A configurable cloud-scale DNN processor for real-time AI. In *ISCA*, 2018.

[30] K. Givaki et al. On the Resilience of Deep Learning for Reduced-voltage FPGAs. In *PDP*, 2020.

[31] D. Gizopoulos et al. Modern Hardware Margins: CPUs, GPUs, FPGAs Recent System-Level Studies. In *IOLTS*, 2019.

[32] K. Guo et al. A survey of FPGA-based neural network accelerator. *arXiv:1712.08934*, 2017.

[33] S. Han et al. Learning both weights and connections for efficient neural network. In *NIPS*, 2015.

[34] S. Han et al. EIE: efficient inference engine on compressed deep neural network. In *ISCA*, 2016.

[35] K. He et al. Deep residual learning for image recognition. In *CVPR*, 2016.

[36] Y. He et al. Channel pruning for accelerating very deep neural networks. In *ICCV*, 2017.

[37] P. Hill et al. DeftNN: Addressing bottlenecks for dnn execution on gpus via synapse vector elimination and near-compute data fission. In *MICRO*, 2017.

[38] K. Himanshu et al. A 320 mV 56 μW 411 GOPs/W Ultra-Low Voltage Motion Estimation Accelerator in 65 nm CMOS. *JSSC*, 2009.

[39] W. Huang et al. Temperature-Aware Architecture: Lessons and Opportunities. *IEEE Micro*, 2011.

[40] S. Jha et al. Kayotee: A fault injection-based system to assess the safety and reliability of autonomous vehicles to faults and errors. *arXiv:1907.01024*, 2019.

[41] S. Jhaet et al. ML-based fault injection for autonomous vehicles: a case for Bayesian fault injection. In *DSN*, 2019.

[42] N. Jouppi et al. In-datacenter performance analysis of a tensor processing unit. In *ISCA*, 2017.

[43] M. Kaliorakis et al. Statistical analysis of multicore CPUs operation in scaled voltage conditions. *CAL*, 2018.

[44] S. Karandikar et al. FireSim: FPGA-accelerated cycle-exact scale-out system simulation in the public cloud. In *ISCA*, 2018.

[45] B. Khaleghi et al. FPGA Energy Efficiency by Leveraging Thermal Margin. *arXiv:1911.07187*, 2019.

[46] F. Khorasani et al. In-register parameter caching for dynamic neural nets with virtual persistent processor specialization. In *MICRO*, 2018.

[47] N. Sung Kim et al. Leakage current: Moore's law meets static power. *Computer*, 2003.

[48] S. Kim et al. MATIC: Learning around errors for efficient low-voltage neural network accelerators. In *DATE*, 2018.

[49] YD. Kim et al. Compression of deep convolutional neural networks for fast and low power mobile applications. *arXiv:1511.06530*, 2015.

[50] S. Koppula et al. EDEN: Enabling Energy-Efficient, High-Performance Deep Neural Network Inference Using Approximate DRAM. In *MICRO*, 2019.

[51] A. Krizhevsky et al. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012.

[52] A. Lavin et al. Fast algorithms for convolutional neural networks. In *CVPR*, 2016.

[53] Y. LeCun et al. Deep learning. *Nature*, 2015.

[54] S. Kyu Lee et al. A 16-nm Always-On DNN Processor With Adaptive Clocking and Multi-Cycle Banked SRAMs. *JSSC*, 2019.

[55] J. Leng et al. GPU voltage noise: Characterization and hierarchical smoothing of spatial and temporal voltage noise interference in GPU architectures. In *HPCA*, 2015.

[56] J. Leng et al. Safe limits on voltage reduction efficiency in GPUs: a direct measurement approach. In *MICRO*, 2015.

[57] J. Leng et al. Asymmetric Resilience: Exploiting Task-Level Idempotency for Transient Error Recovery in Accelerator-Based Systems. In *HPCA*, 2020.

[58] G. Li et al. Understanding error propagation in deep learning neural network (DNN) accelerators and applications. In *SC*, 2017.

[59] H. Li et al. On-Chip Memory Technology Design Space Explorations for Mobile Deep Neural Network Accelerators. In *DAC*, 2019.

[60] Z. Li et al. E-RNN: Design optimization for efficient recurrent neural networks in FPGAs. In *HPCA*, 2019.

[61] F. Libano et al. Selective Hardening for Neural Networks in FPGAs. *TNS*, 2018.

[62] F. Libano et al. Understanding the Impact of Quantization, Accuracy, and Radiation on the Reliability of Convolutional Neural Networks on FPGAs. *TNS*, 2020.

[63] Y. Liu et al. Fault injection attack on deep neural network. In *ICCAD*, 2017.

[64] Y. Ma et al. Optimizing the convolution operation to accelerate deep neural networks on FPGA. *TVLSI*, 2018.

[65] MaxIntegrated. https://www.maximintegrated.com. 2019.

[66] T. Miller et al. VRSync: Characterizing and eliminating synchronization-induced voltage emergencies in many-core processors. In *ISCA*, 2012.

[67] P. Molchanov et al. Pruning convolutional neural networks for resource efficient inference. *arXiv:1611.06440*, 2016.

[68] B. Moons et al. A 0.26-to-10 TOPs/W subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28nm FDSOI. In *ISSCC*, 2017.

[69] A. Moradi et al. Side-channel leakage through static power. In *CHES*, 2014.

[70] M. Hadi Mottaghi et al. Aging Mitigation in FPGAs Considering Delay, Power, and Temperature. *TR*, 2019.

[71] H. Nakahara et al. A batch normalization free binarized convolutional deep neural network on an FPGA. In *FPGA*, 2017.

[72] K. Neshatpour et al. Enhancing Power, Performance, and Energy Efficiency in Chip Multiprocessors Exploiting Inverse Thermal Dependence. *TVLSI*, 2018.

[73] E. Nurvitadhi et al. Accelerating binarized neural networks: Comparison of FPGA, CPU, GPU, and ASIC. In *FPT*, 2016.

[74] E. Nurvitadhi et al. Why Compete When You Can Work Together: FPGA-ASIC Integration for Persistent RNNs. In *FCCM*, 2019.

[75] P. Pandey et al. GreenTPU: Improving Timing Error Resilience of a Near-Threshold Tensor Processing Unit. In *DAC*, 2019.

[76] G. Papadimitriou et al. Harnessing voltage margins for energy efficiency in multicore CPUs. In *MICRO*, 2017.

[77] G. Papadimitriou et al. Voltage margins identification on commercial x86-64 multicore microprocessors. In *IOLTS*, 2017.

[78] G. Papadimitriou et al. Adaptive Voltage/Frequency Scaling and Core Allocation for Balanced Energy and Performance on Multicore CPUs. In *HPCA*, 2019.

[79] G. Papadimitriou et al. Exceeding Conservative Limits: A Consolidated Analysis on Modern Hardware Margins. *TDMR*, 2020.

[80] A. Parashar et al. SCNN: An accelerator for compressed-sparse convolutional neural networks. In *ISCA*, 2017.

[81] K. Parasyris et al. A Framework for Evaluating Software on Reduced Margins Hardware. In *DSN*, 2018.

[82] J. Park et al. Scale-out acceleration for machine learning. In *MICRO*, 2017.

[83] Power Management Bus (PMBus). https://pmbus.org/. 2020.

[84] A. Putnam et al. A reconfigurable fabric for accelerating large-scale datacenter services. In *ISCA*, 2014.

[85] J. Qiu et al. Going deeper with embedded FPGA platform for Convolutional Neural Network. In *FPGA*, 2016.

[86] B. Reagen et al. Minerva: Enabling low-power, highly-accurate deep neural network accelerators. In *ISCA*, 2016.

[87] B. Reagen et al. Ares: A framework for quantifying the resilience of deep neural networks. In *DAC*, 2018.

[88] M. Riera et al. Computation reuse in DNNs by exploiting input similarity. In *ISCA*, 2018.

[89] A. Roelke et al. Pre-RTL Voltage and Power Optimization for Low-Cost, Thermally Challenged Multicore Chips. In *ICCD*, 2017.

[90] S. Salamat et al. Workload-aware opportunistic energy efficiency in multi-FPGA platforms. *arXiv:1908.06519*, 2019.

[91] B. Salami. Aggressive undervolting of FPGAs: power & reliability trade-offs. *Ph.D. Dissertation, Universitat Politècnica de Catalunya (UPC)*, 2018.

[92] B. Salami et al. HATCH: hash table caching in hardware for efficient relational join on FPGA. In *FCCM*, 2015.

[93] B. Salami et al. Accelerating hash-based query processing operations on FPGAs by a hash table caching technique. In *CARLA*, 2016.

[94] B. Salami et al. AxleDB: A novel programmable query processing platform on FPGA. *MICPRO*, 2017.

[95] B. Salami et al. A Demo of FPGA Aggressive Voltage Downscaling: Power and Reliability Tradeoffs. In *FPL*, 2018.

[96] B. Salami et al. Comprehensive evaluation of supply voltage underscaling in fpga on-chip memories. In *MICRO*, 2018.

[97] B. Salami et al. Fault Characterization Through FPGA Undervolting. In *FPL*, 2018.

[98] B. Salami et al. On the resilience of RTL NN accelerators: Fault characterization and mitigation. In *SBAC-PAD*, 2018.

[99] B. Salami et al. Evaluating Built-in ECC of FPGA on-chip Memories for the Mitigation of Undervolting Faults. In *PDP*, 2019.

[100] B. Salami et al. LEGaTO: Low-Energy, Segure, and Resilient Toolset for Heterogeneous Computing. In *DATE*, 2020.

[101] H. Sharma et al. From high-level deep neural models to FPGAs. In *MICRO*, 2016.

[102] H. Sharma et al. Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network. In *ISCA*, 2018.

[103] L. Shen et al. Fast Voltage Transients on FPGAs: Impact and Mitigation Strategies. In *FCCM*, 2019.

[104] Y. Shen et al. Escher: A CNN accelerator with flexible buffering to minimize off-chip transfer. In *FCCM*, 2017.

[105] Y. Shen et al. Maximizing CNN accelerator efficiency through resource partitioning. In *ISCA*, 2017.

[106] K. Simonyan et al. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014.

[107] N. Suda et al. Throughput-optimized OpenCL-based FPGA accelerator for large-scale convolutional neural networks. In *FPGA*, 2016.

[108] K. Swaminathan et al. Bravo: Balanced reliability-aware voltage optimization. In *HPCA*, 2017.

[109] V. Sze et al. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 2017.

[110] C. Szegedy et al. Going deeper with convolutions. In *CVPR*, 2015.

[111] Z. Tang et al. The Impact of GPU DVFS on the Energy and Performance of Deep Learning: an Empirical Study. In *ACM e-Energy*, 2019.

[112] M. Garay Trindade et al. Assessment of a Hardware-Implemented Machine Learning Technique under Neutron Irradiation. *TNS*, 2019.

[113] A. Uht et al. Going beyond worst-case specs with TEAtime. *Computer*, 2004.

[114] A. Vaishnav et al. A survey on FPGA virtualization. In *FPL*, 2018.

[115] X. Wang et al. Bit Prudent In-Cache Acceleration of Deep Convolutional Neural Networks. In *HPCA*, 2019.

[116] P. Whatmough et al. 14.3 A 28nm SoC with a 1.2 GHz 568nJ/prediction sparse deep-neural-network engine with> 0.1 timing error rate tolerance for IoT applications. In *ISSCC*, 2017.

[117] P. Whatmough et al. DNN Engine: A 28-nm timing-error tolerant sparse deep neural network processor for IoT applications. *JSSC*, 2018.

[118] C. Wilkerson et al. Trading off cache capacity for reliability to enable low voltage operation. *ISCA*, 2008.

[119] C. Wilkerson et al. Reducing cache power with low-cost, multi-bit error-correcting codes. In *ISCA*, 2010.

[120] Q. Xiao et al. Exploring heterogeneous algorithms for accelerating deep convolutional neural networks on FPGAs. In *DAC*, 2017.

[121] Xilinx. 7 Series FPGAs Memory Resources https://www.xilinx.com/support/documentation/user_guides/ug473_7Series_Memory_Resources.pdf. 2019.

[122] Xilinx. DNNDK User Guide https://www.xilinx.com/support/documentation/user_guides/ug1327-dnndk-user-guide.pdf. 2019.

[123] Xilinx. DPU IP Product Guide https://www.xilinx.com/support/documentation/ip_documentation/dpu/v3_1/pg338-dpu.pdf. 2019.

[124] Xilinx. UltraScale Architecture Memory Resources https://www.xilinx.com/support/documentation/user_guides/ug573-ultrascale-memory-resources.pdf. 2019.

[125] Xilinx. Zynq UltraScale+ MPSoC ZCU102 Evaluation Kit https://www.xilinx.com/products/boards-and-kits/ek-u1-zcu102-g.html, 2019.

[126] G. Yalcin et al. Exploiting a fast and simple ECC for scaling supply voltage in Level-1 caches. In *IOLTS*, 2014.

[127] G. Yalcin et al. Exploring Energy Reduction in Future Technology Nodes via Voltage Scaling with Application to 10nm. In *PDP*, 2016.

[128] L. Yang et al. SRAM voltage scaling for energy-efficient convolutional neural networks. In *ISQED*, 2017.

[129] R. Yazdani et al. The dark side of DNN pruning. In *ISCA*, 2018.

[130] M. Yufei et al. Optimizing loop operation and dataflow in FPGA acceleration of deep convolutional neural networks. In *FPGA*, 2017.

[131] C. Zhang et al. Optimizing FPGA-based accelerator design for deep convolutional neural networks. In *FPGA*, 2015.

[132] J. Zhang et al. Thundervolt: enabling aggressive voltage underscaling and timing error resilience for energy efficient deep learning accelerators. In *DAC*, 2018.

[133] J. Jun Zhang et al. Analyzing and mitigating the impact of permanent faults on a systolic array based neural network accelerator. In *VTS*, 2018.

[134] S. Zhang et al. Cambricon-x: An accelerator for sparse neural networks. In *MICRO*, 2016.

[135] X. Zhang et al. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, 2018.

[136] A. Zhou et al. Incremental network quantization: Towards lossless CNNs with low-precision weights. *arXiv:1702.03044*, 2017.

[137] Z. Zhu et al. A Configurable Multi-Precision CNN Computing Framework Based on Single Bit RRAM. In *DAC*, 2019.

[138] A. Zou et al. Voltage-Stacked GPUs: A Control Theory Driven Cross-Layer Solution for Practical Voltage Stacking in GPUs. In *MICRO*, 2018.