Optimizing energy efficiency of CNN-based object detection with dynamic voltage and frequency scaling

Jiang Weixiong, Yu Heng, Zhang Jiale, Wu Jiaxuan, Luo Shaobo, Ha Yajun

University of Nottingham
UK | CHINA | MALAYSIA

Faculty of Science and Engineering, University of Nottingham Ningbo China, 199 Taikang East Road, Ningbo, 315100, Zhejiang, China.

First published 2020

University of Nottingham
UK | CHINA | MALAYSIA

Optimizing Energy Efficiency of CNN-Based Object Detection with Dynamic Voltage and Frequency Scaling

Jiang Weixiong[1,2,3], Yu Heng[4], Zhang Jiale[1,2,3], Wu Jiaxuan[1,2,3], Luo Shaobo[5], Ha Yajun[1,2,3]

(1 School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China)
(2 Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai 200050, China)
(3 University of Chinese Academy of Sciences, Beijing 100049, China)
(4 University of Nottingham Ningbo China, Ningbo 315100, China)
(5 Universite Paris-Est, Paris 93162, France)

Abstract: On the one hand, accelerating convolution neural networks (CNNs) on FPGAs requires ever increasing high energy efficiency in the edge computing paradigm. On the other hand, unlike normal digital algorithms, CNNs maintain their high robustness even with limited timing errors. By taking the advantages of this unique feature, we propose to use dynamic voltage and frequency scaling (DVFS) to further optimize the energy efficiency for CNNs. First, we develop a DVFS framework on FPGAs. Second, we apply the DVFS to SkyNet, a state-of-the-art neural network targeting on object detection. Third, we analyze the impact of DVFS on CNNs in terms of performance, power, energy efficiency and accuracy. Compared to the state-of-the-art, experimental results show that we achieved 38% improvement in energy efficiency without any loss in accuracy. Results also show that we can achieve 47% improvement in energy efficiency if we allow 0.11% relaxation in accuracy.

Key words: CNN; Object Detection; FPGA; DVFS
DOI: 10.1088/1674-4926/30/1/014003 PACC: 7220J;7340Q

1. Introduction

FPGA has become a promising platform for edge computing in recent years, given its energy efficiency compared to GPU and flexibility compared to ASICs [1]. There are many previous works such as [2–13] that present various methods on optimizing architecture or design flow. However, none of them specifically optimize energy efficiency. Dynamic Voltage and Frequency Scaling (DVFS) has been extensively applied as a system-level methodology to optimize the system execution. Through judiciously scaling up or down the execution voltage and speed of the processing units, DVFS effectively achieves throughput maximization, temperature management, application quality maximization, and energy minimization. Although the delay of the combinational logic will increase as the voltage decreases, which may bring a certain chance of error during operation, the CNN itself is robust to errors [14–17]. Even if a small number of neurons get wrong values in the process of inference, it does not affect the final accuracy. In this context, combining CNN and DVFS is an ideal method to extrude the potential of FPGAs to optimize either performance or energy efficiency.

To add the DVFS support to CNN accelerators, a flexible DVFS platform is needed. Previous works have been focusing on adding the DVFS support to commercial FPGAs, but their solutions for DVFS have various limitations. For example, the DVFS module consumes too much logic resources and thus affects timing and power [18]. The scaling resolution is not high enough, or the scaling time is too long [19]. To solve these issues, we develop a novel DVFS framework with high resolution and flexibility as well as low area overhead

and scaling time to implement a DVFS system quickly. Our main innovative technical contributions are as follows:

- We propose a framework for fine-grained DVFS on state-of-the-art FPGA devices.
- We combine the framework with SDSoC and then apply it to SkyNet, a lightweight CNN for object detection.
- We analyze the impact of DVFS on performance, power, energy efficiency, and accuracy.
- We achieve 54% improvement in performance, 38% improvement in energy efficiency, and 106% improvement in unified energy efficiency (UEE) without any loss in accuracy compared to the original SkyNet. If we relax the requirement on the accuracy, we can achieve 56% improvement in performance, 47% improvement in energy efficiency, and 121% improvement in UEE at the cost of 0.11 decrease in IoU.
- We develop a DVFS policy basing on the measured metrics targeting on real-time applications in realistic scenarios. With this DVFS policy, the average power has been reduced by 30% compared to the original design.

Section II presents the related works on CNN and DVFS. Section III defines a power optimization problem in the scenario of edge computing. Section IV introduces the DVFS framework as well as the DVFS policy for energy efficiency optimization. Section V describes how to combine the DVFS framework with SDSoC and the architecture of the system after applying DVFS to CNN accelerators. Section VI gives the experimental results and analyzes the impact of DVFS on the original CNN accelerator. It also presents the experimental results of the DVFS policy. Finally, Section VII concludes the paper.

2. Related Work

2.1. CNN

For edge applications, there are many previous works on optimizing the architecture of the FPGA-based CNN accelerator. [4] presents an RTL-level CNN compiler that automatically generates customized FPGA hardware for the inference tasks of various CNNs, in order to enable high-level fast prototyping of CNNs from software to FPGA. A programmable and flexible CNN accelerator architecture together with a data quantization strategy and compilation tool is introduced in [8]. Authors in [6] present an efficient hardware accelerator design of deep residual learning algorithms. In [5], the authors quantitatively analyze and optimize the design objectives of the CNN accelerator based on multiple design variables. In [7], an architecture named Tile-Grained Pipeline Architecture (TGPA) for low latency CNN inference is proposed. [20] proposes a layer conscious memory management framework for FPGA-based CNN hardware accelerators.

Recently, software-hardware co-design gains more and more attention. [21] proposes REQ-YOLO, a resource-aware, systematic weight quantization framework for object detection, considering both algorithm and hardware resource aspects in object detection. [22] and [23] raise a novel and practical bi-directional co-design approach, including a bottom-up DNN model design strategy together with a top-down flow for DNN accelerator design. It enables a joint optimization of both DNN models and their deployment configurations on FPGAs. Also, [23] builds an automatic co-design flow, including an Auto-DNN engine to perform a hardware-oriented DNN model search, as well as an Auto-HLS engine to generate synthesizable C code of the FPGA accelerator for explored DNNs. However, all of the works mentioned above ignore specific energy efficiency optimization.

[24] proposes an energy proportional framework with adaptive voltage and frequency scaling and applied it to binary neural networks (BNN) for classification. However, BNN has very limited precision and is almost unavailable in practical applications. What's more, [24] lacks exploration of actual scenes in an edge

computing context. In this paper, we combine our fine-grained DVFS framework with CNN based object detection accelerators to perform specific optimization on energy efficiency. Besides, we define two kinds of actual scenarios in edge computing and perform specific optimization. To demonstrate the effectiveness of DVFS, we choose SkyNet [25] as a case study. SkyNet won the championship in Design and Automation Conference-System Design Contest 2019(DAC-SDC2019), a low power object detection challenge in images captured by unmanned aerial vehicles (UAVs). SkyNet delivers 0.716 IoU and 25.05 FPS on an Ultra96 FPGA. The reason why we choose SkyNet as a case study is that SkyNet represents the highest level of CNN implementation on FPGAs considering performance and energy efficiency. We want to develop a DVFS based method to further tap its potential and achieve better improvement in both performance and energy efficiency.

2.2. DVFS

DVFS has been proven to be a popular and efficient system-level methodology to optimize system execution metrics, such as throughput, power, energy, temperature, reliability, and QoS. For example, [26] proposes to adjust the frequency in response to application behavior changes, to optimize energy efficiency for general-purpose CPU systems. Authors in [27] empirically reveal that on a commercial smartphone platform, energy consumption strongly correlates its CPU frequency, and exhibits an optimal operating frequency for energy minimization. Huang et al. [28] propose to maximize throughput and prevent system overheating by carefully interleaving the hot and cool tasks and adjusting the task execution frequencies. QoS maximization through efficient DVFS can be found in [29,30]. By leveraging task adaptability, the output quality can be dynamically adjusted under temperature constraints, and judiciously applying DVFS can maximize the quality. To enhance system reliability, the authors propose to carefully tune the frequency of CPUs to control temperature overflow to minimize the thermal cycling that stresses chips [31]. While the evaluation of DVFS efficacy on those works is largely based on simulations, a realistic platform like ours that enables quick system synthesis and DVFS algorithm evaluation are highly desirable.

There have been existing works combining CNN and DVFS on ASICs, CPU, or GPU. [12] develops a principled approach and a data-driven analytical model with DVFS to optimize the granularity of threads during CNN software synthesis. [32] proposes a low-power CNN-based face recognition system for user authentication in smart devices with DVFS. [33] uses a performance-power analytical model fitted on a parameterized implementation of a CNN accelerator in a 28-nm FDSOI technology to explore large design space and to obtain the Pareto points that maximize the effectiveness of DVFS in the sub-space of throughput and energy efficiency. Our work is the first one to combine CNN-based object detection with DVFS on FPGA to our best knowledge.

On the hardware aspect, many DVFS supported logic and systems have been developed. Widely adopted frequency scaling approaches include phase-locked-loop [34] and delay-locked-loop [35]. Brynjolfson and Zilic propose a clock management scheme named DPCP on FPGA platforms [36]. To produce dynamically scaled frequency, state-of-the-art FPGA platforms (Xilinx 7 Series or later) adopt a hybrid Mixed-Mode Clock Manager (MMCM)+PLL architecture. Authors in [37] propose performance scaling on Virtex-7, where DFS is realized by employing an external programmable oscillator controlled by a Picoblaze processor. In addition to works on traditional FPGAs, there exist other works that implement system-level optimization on ZYNQ using DVFS. [38] raises a method for accurate power control and monitoring on ZYNQ device, [39] investigates the viability of physical power gating FPGA devices that incorporate a hardened processor in a different power domain, but they only implemented several computing units such as fp32mult, IIR, DCT, etc., rather than an accelerator. In order to solve these problems, we propose a DVFS framework with high scaling range and resolution as well as low scaling time in this paper. What's more, we combine the DVFS framework with high-level synthesis tool, SDSoC, making it possible to build a system with DVFS support in a complete software development flow.

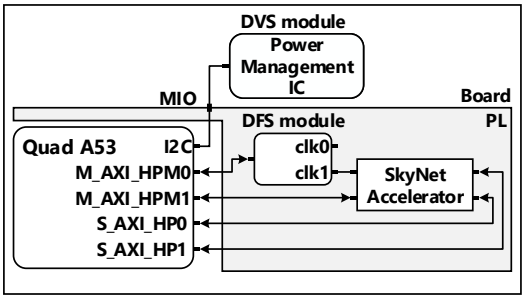


Figure 1: System architecture of CNN accelerator

3. Problem Definition

Edge computing refers to an open platform that integrates network, computing, storage, and application core capabilities on the side close to the data source, providing near-end services. The edge computing platform's power supply conditions are far less than the cloud computing platform, which put higher requirements on the energy efficiency ratio of edge computing devices. For the object detection task in edge computing scenario, there are generally two cases. The first application scenario pursues the least amount of energy required to process each frame of data. For example, in a home security scenario, only when the sensor detects a moving object, the camera will take a picture and wake up the edge computing platform to process the data. Otherwise, the edge computing platform can work in a sleep state. In the second application scenario, data is generated at a fixed rate, and the goal is to minimize the average power consumption throughout the work period. For instance, in autonomous driving, the camera produces images at a fixed rate, and the edge computing platform has to be running at all times. In this paper, we want to develop a DVFS method that significantly improve the energy efficiency of object detection on FPGA-based edge computing platform.

4. DVFS Framework

Fig.1 shows the system architecture of our DVFS framework, where DVS module is implemented by a power management IC (PMIC) that is responsible for controlling and monitoring the switching regulators. The switching regulators are connected to different components on the FPGA, such as PS, PL, IO ports, and BRAM. The DFS module is a clock generator that generates clock signals for the accelerators on the programmable logic. Our platform provides the capability of dynamic frequency and voltage scaling with high resolution, wide range, and low scaling time. Users can scale frequency from 20MHz to 400MHz at a step size of 1MHz in 3μ s and scale voltage from 650mV to 850mV at the step size of 10mV in 2ms. Besides, power monitoring APIs for various power rails on FPGA are also provided in our framework. The proposed DVS framework is coarse-grained because modern FPGAs do not support multiple voltages in different domains. The power supply of all the slices is connected together. Therefore it is impossible to implement the module-level DVS method on current FPGA devices up till now. In this section, we will introduce how the DVS and DFS are implemented in hardware and then discuss how to access them in Linux OS. The framework supports a series of Zynq 7000 and Zynq UltraScale+ series FPGA boards including ZC702, ZC706, ZCU102, etc at the moment. To be specific, we use ZCU104 as an example in this paper.

4.1. Dynamic Frequency Scaling

The Zynq 7000 and Zynq UltraScale+ series FPGA mainly has two kinds of clock sources, FCLK on the PS and MMCM on the PL. The FCLK has the advantage of low area overhead and existing driver in Linux. However, it can only provide limited frequency points. Therefore, we choose MMCM as the clock generator in our design since it is armed with a larger range and higher resolution. As shown in Fig.2, the

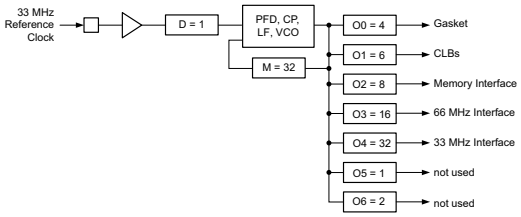


Figure 2: MMCM

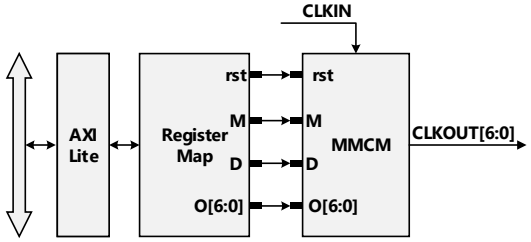


Figure 3: Dynamically reconfiguring MMCM using AXI4-Lite interface

```
#define clk_addr
void set_frequency(int F)
{
    clk = mmap(clk_addr);
    if ((F >= 20) && (F < 500))
    {
        D = 5;
        O = 5;
        M = F / 4.0;
    }
    clk.write(D);
    clk.write(O);
    clk.write(M);
}
```

Figure 4: Pseudocode of the driver for dynamic frequency scaling

input clock is firstly multiplied and then divided by VCO in MMCM. The VCO output is further divided to generate different frequencies for different components. The output frequency can be calculated by the following equation,

$$F_{OUT} = F_{CLKIN} \times \frac{M}{D \times O} \tag{1}$$

where M, D, and O can be configured through dynamic scaling port available in the MMCM blocks and new frequencies are generated at run-time. As shown in Fig.3, the configuration information and the reset signal are sent to the MMCM via an AXI4-Lite interface and then saved in the register map. MMCM retrieves the configuration from the register map and resets to desired frequency accordingly.

Fig.4 shows the driver for DFS. The clock generator is mapped to a device firstly so that it can be accessed in Linux userspace, then M, D, and O are calculated according to the target frequency. The clock input of the MMCM is connected to the FCLK on the PS, and the FCLK is set to 100MHz. As shown in Eq.1, the M can be the fractional number that is a multiple of 0.25. Thus we can divide the target frequency by 4.0. As shown in Eq.1, the output frequency is $F_{CLKIN}\times$ M $\div$ D $\div$ O $= 100 \times F \div 4.0 \div 5 \div 5 = F$(MHz) and the resolution is 1MHz. Benefit from high bandwidth and low latency on-chip bus between the PS and the PL, it takes only 3us for each frequency scaling operation.

4.2. Dynamic Voltage Scaling

The Power Management Bus (PMBus) is an open standard power-management protocol, which is compatible with the I2C protocol at the physical level. This flexible and highly versatile standard allows for

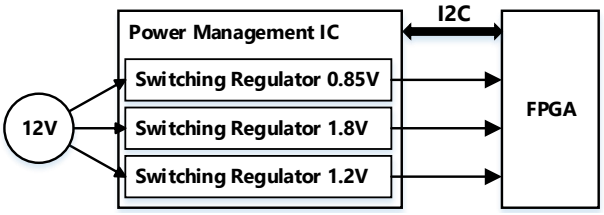


Figure 5: Power Management Framework on FPGA

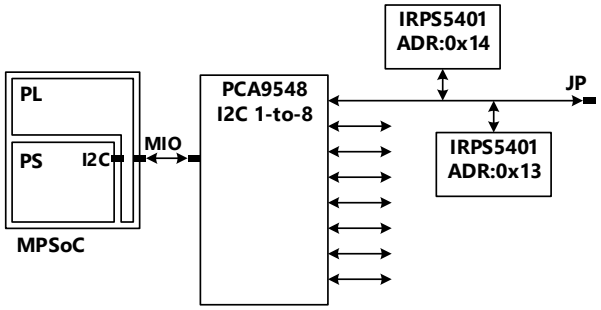


Figure 6: Power monitoring system topology on ZCU104

communication between devices based on both analog and digital technologies and provides true interoperability, which will reduce design complexity and shorten the time to market for power system designers. The-state-of-the-arts FPGA boards usually adopt power regulators and a PMBus-compliant system controller to supply core and auxiliary voltages as shown in Fig.5. For instance, ZC702 and ZC706 use the Texas Instrument (TI) UCD92x, ZCU100 and ZCU102 use the TI INA226, and ZCU104 uses the Infineon IRPS5401. All of the PMICs mentioned follow the same power management protocol, thus our DVS framework is compatible with a wide variety of FPGA boards with minimal changes. The power management IC(PMIC) gets configuration information from the FPGA through the I2C bus and controls the output voltage of each switching regulator. In the meantime, the PMIC monitors the voltage and the current of each switching regulator and reports the information back to the FPGA through the I2C bus.

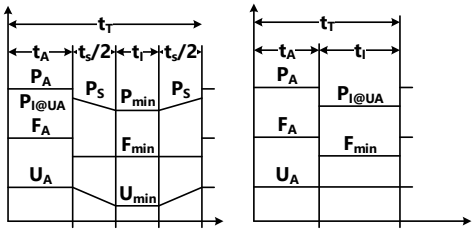
As shown in Fig.6, there are two IRPS5401s on the ZCU104 board, both of which are connected to PCA9548, which is an I2C 1-to-8 bus switch external to the device. Then PCA9548 is connected to the PS I2C controller via MIO. IRPS5401 can control and monitor up to ten voltage rails on board, including VCC12 and VCCINT. The former is the power input of the whole board and the latter is the PL side supply voltage. This gives us the possibility to scale the accelerator's voltage and frequency in real-time. Besides, IRPS5401 can also monitor the power consumption of each power rail and feedback the information through PMBus. There are two alternatives to communicate with the PMIC internally, as both the PL and the PS banks have access to the PMIC. The hardware method is to implement an AXI I2C controller in PL, while the software method connects the MIO pins directly to the PS. We choose the software method because it not only does not have any area overhead but also can take advantage of the mature I2C driver embedded in the Linux kernel.

Fig.7 demonstrates the driver for DVS. Since the PMBus protocol is compatible with the I2C protocol at the physical level, we can leverage the integrated I2C driver in Linux to communicate with the PMIC. The output voltage can be modified by writing PMBus commands to the PMIC in a fixed order according to the PMBus protocol. Because of the low speed of I2C and more commands to transfer, the voltage scaling takes about 2ms each time. On the other hand, the power consumption can also be accessed in Linux in a similar way in which the PMIC is configured.

In summary, at the hardware level, the proposed DVFS framework takes advantage of the MMCM module on modern FPGAs, as well as the PMBus compliant power supply system. In the software level, we

```c
#include <i2c-dev.h>
void set_voltage(int voltage)
{
    i2c_write(POWER_GOOD_OFF);
    i2c_write(POWER_GOOD_ON);
    i2c_write(VOUT_UV_FAULT_LIMIT);
    i2c_write(VOUT_UV_WARN_LIMIT);
    i2c_write(VOUT_MARGIN_LOW);
    i2c_write(VOUT_COMMAND);
    i2c_write(VOUT_MARGIN_HIGH);
    i2c_write(VOUT_OV_WARN_LIMIT);
    i2c_write(VOUT_MAX);
}
```

Figure 7: Pseudocode of the driver for dynamic voltage scaling



(a) (b)

Figure 8: (a) Timing diagram illustrating the DVFS policy with enough time to perform DVS. (b) Timing diagram illustrating the DVFS policy without enough time to perform DVS.

adapt the provided memory map and I2C drivers in Linux to control the DVFS module, making it easier to use. We take ZCU104 as an example in this paper, however, the DVFS framework can be easily ported to Zynq 7000 devices or even non-Zynq FPGA devices as long as the board is equipped with a configurable PMIC. For the non-Zynq FPGA devices, we can inherit the drivers for Zynq devices if a MicroBlaze, a RISC Harvard architecture soft processor provided by Xilinx, is adapted as the DVFS module controller and running Linux on it.

4.3. DVFS Policy

For practical applications like real-time object detection, there is a constant frame rate of images obtained from a camera. In this context, it is reasonable to let the processing frame rate of the accelerator being consistent with the frame rate of the camera to save power consumption. If the peak processing frame rate of the accelerator is higher than that of the camera, there are two strategies, namely "long active, short idle" and "short active, long idle," to meet the performance requirements. The former is to set the accelerator to low frequency thus it takes a long time for the accelerator to process one frame and thus the idle time is short, while the latter is to set the accelerator to a high frequency and thus has a shorter active time.

To maximize energy savings, we can scale both the voltage and the frequency to a minimum when the accelerator is idle. And this makes it impossible to analyze which solution is the optimal one in the qualitative method. To quantitatively illustrate the problem, we define E_T, t_T, t_S, t_I, T_A, P_S, P_I, P_A, and P_M, where E stands for energy, P stands for power, t stands for time, T stands for total, S stands for scaling, I stands for idle, A stands for Active and M stands for mean. Their relationship is shown in the following equations:

$$t_T = t_I + t_S + t_A \tag{2}$$

$$E_T = t_I \times P_I + t_S \times P_S + t_A \times P_A \tag{3}$$

$$P_M = E_T / t_T \tag{4}$$

Fig.8 shows the detailed steps of our DVFS policy. The two sub-figures show the voltage, frequency, and power consumption varies with time differently under different t_T. In each period, the accelerator first

runs at F_A and U_A and then scales to the minimum supported frequency F_{min} and the minimum supported voltage U_{min} to save static power once the computation is finished. According to our DVFS framework, frequency scaling takes 3us while voltage scaling takes 2ms. This means that when the t_T is long enough, both frequency scaling and voltage scaling are performed. In this occasion, the chip's power consumption first drops to power-idle-at-active-voltage ($P_{I@UA}$) when frequency scaling is performed and then gradually decreases to P_{min} as the voltage gradually decreases to U_{min}. The power when the voltage is scaling (P_S) is given by the average of P_{min} and $P_{I@UA}$. However, if the t_T is short, only the frequency can be adjusted as shown in the right sub-figure. Our goal is to find a voltage/frequency combination that has a minimum of P_M with our DVFS policy.

5. Design Flow

5.1. System Architecture

Fig.1 shows the topology details of the system. It is implemented on the latest Zynq UltraScale+ device that integrates processing system (PS) and programmable logic (PL). The PS is a Quad-Core A53 processor running at 1.2GHz. It provides three M_AXI buses and six S_AXI buses for high throughput data transfer between the PL and the PS. What's more, the PS is also embedded with common interfaces for various peripherals, such as I2C, USB, Ethernet, Display Port, etc. The I2C interface is connected to the power management IC on the FPGA board via MIO. Two high-performance M_AXIs are enabled in this design, one of them is connected to the DFS module, and the other one is connected to the SkyNet accelerator. The PS is running Linux OS and is responsible for loading images and sends control signals such as target frequency of the DFS module and the configuration parameters for the SkyNet accelerator. All the clock signals related to the SkyNet accelerator are under the DFS module/clk1 domain including the accelerator's clock input as well as that of M_AXI_HPM1, S_AXI_HP0, S_AXI_HP1. All the input feature maps and weights are transferred through S_AXI_HP0 and all the output feature maps are transferred through S_AXI_HP1. The frequency and the voltage can only be modified during the iteration interval of the hardware calls.

5.2. SDSoC Support

To build an ARM + FPGA system with DVFS support in the traditional design flow, designers should first add the DVFS module in PL, then prepare the drivers for controlling DVFS module in Linux. After the DVFS function is ready, the accelerators are added to the system and connected with the DVFS modules, which is quite exhausting. What's more, preparing testbench that is able to cover a sufficient number of test cases is also time-consuming. To solve this problem, we combine the DVFS framework with Xilinx's latest tool for developing ARM + FPGA system in C/C++/OpenCL, SDSoC, to further enhance the efficiency of system development.

The starting point of an SDSoC-based design is an SDSoC hardware platform, where all the hardware components such as DDR and DVFS modules are defined. Then the boot loaders and target operating systems are built to bring-up the hardware platform. After that, the drivers for the DVS and the DFS module are embedded in the operating system. The bootloaders, target operating system, and the drivers together are called SDSoC software platform, and the hardware platform and software platform together are called SDSoC platform. With this platform, users can develop an embedded system with hardware accelerators in a total software design flow. The user's C/C++/OpenCL code will be translated into Verilog/VHDL code and package the Verilog/VHDL into IPs by SDSoC compiler. After that, SDSoC will connect the accelerator to the PS and the DFS module automatically and then generate the bitstream. Using this platform, we can focus on the accelerator design rather than paying attention to the DVFS module.

Table 1: Resource utilization of the system

Resource	LUT	LUTRAM	FF	BRAM	DSP
SkyNet Total	54639	1984	65196	209	333
Accelerator	49934	921	57101	209	333
AXI Bus	4691	1062	8030	0	0
System Reset	14	1	65	0	0
SkyNet DVFS Total	56902	2084	66781	209	333
Accelerator	49910	921	56095	209	333
AXI Bus	5799	1161	9127	0	0
DFS Module	1164	0	1492	0	0
System Reset	29	2	67	0	0
Total	230400	101760	460800	312	1728

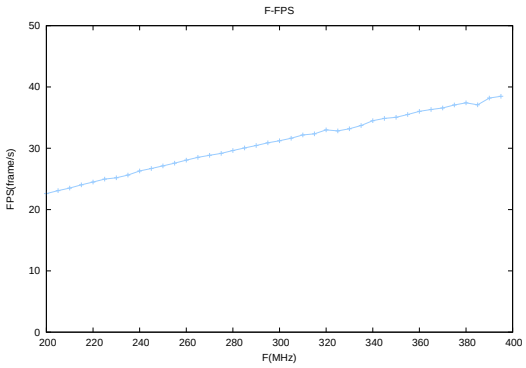


Figure 9: Total time change with with frequency respectively.

6. Experimental Results and Analysis

6.1. Experimental Setup

The SkyNet is written in High-Level Synthesis (HLS) C++ and then synthesized in SDSoC 2019.1. The Quad Cortex-A53 cores in ZCU104 is running Ubuntu18.04 modified by Xilinx. We test the system using the sample dataset of DAC-SDC2019 that is composed of 1000 images and corresponding ground truth. We record the total time and energy consumption for processing all the images and calculated the average IoU under different voltage & frequency combinations. We sweep the VCCINT from 680mV to 840mV at the step size of 20mV and sweep the frequency from 200MHz to 400MHz at the step size of 1MHz. Table.1 gives the resource consumption of the original SkyNet and SkyNet with DVFS.

6.2. Performance Analysis

Fig.9 shows the relationship between the frequency and the achieved frames per second (FPS). The figure shows that the highest performance of SkyNet in Zynq Ultrascale+ is at 371MHz with 38.3FPS and that the achieved FPS performs a linear relation with frequency as expected.

6.3. Power Analysis

To better illustrate the impact of DVFS on power consumption, in this section, we focus on the power of the PL that is supplied by the VCCINT power rail. Other power rails include VCCAUX which charges the clock managers, IOs among the other blocks and VCCBRAM used by BRAMs. The power drawn from

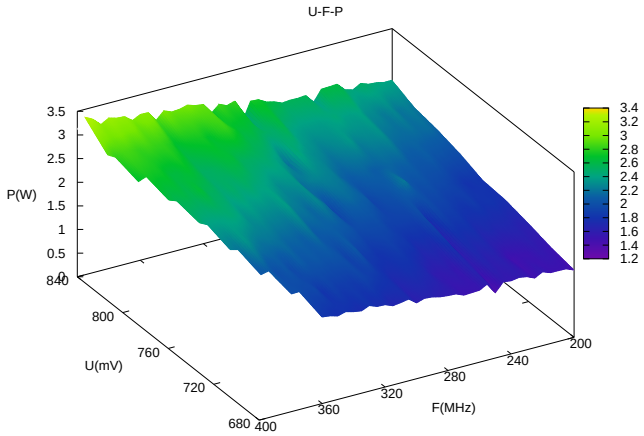


Figure 10: PL side power change with voltage and frequency respectively.

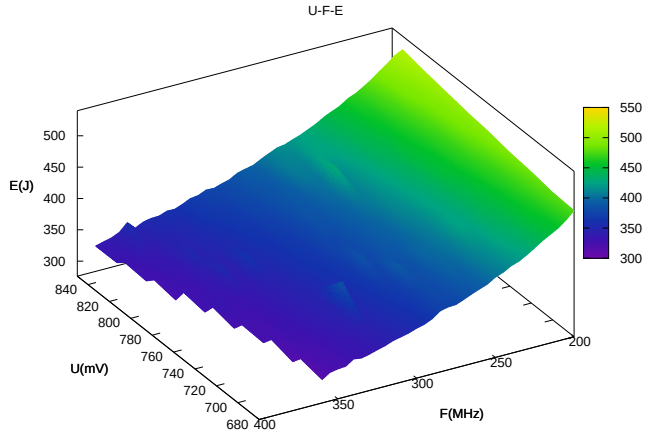


Figure 11: Total energy changes with voltage and frequency respectively.

these additional power rails is considerably lower than VCCINT. Also, the PS of the device where the ARM processor resides is not included in the calculations in that scaling VCCINT does not affect the power of the ARM processor.

Fig.10 shows the measured power in function of the clock frequency and the voltage when the SkyNet accelerator operates on ZCU104. The highest frequency at which the accelerator can operate increases with the voltage of the PL. For instance, if VCCINT is set to 680mV, the highest frequency of the accelerator is 360MHz, above which the system is unstable. Whether the calculation result is correct or not is not considered here, and the impact of DVFS on accuracy will be discussed later. As expected, power has a linear relation with frequency and that the configurations of voltage scaled reduce power significantly since voltage affects both dynamic and static power. The minimum power measured is 1.42 Watts at 200MHz/680mV for the Ultrascale+ device. Therefore, these experiments confirm that significant performance and power margins are available that can be exploited by the DVFS framework.

6.4. Energy Analysis

In the previous section, we only considered the power of the PL part to make the result look more obvious, but considering the actual edge computing scenario, we should also take the power consumption of CPU into account. For example, when the data arrives, the CPU wakes up and processes the images, and then the CPU immediately enters the sleep state. Although the power of the PL side goes up as the

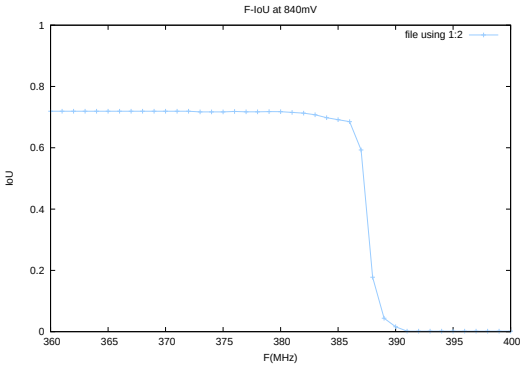


Figure 12: IoU changes with frequency at 840mV.

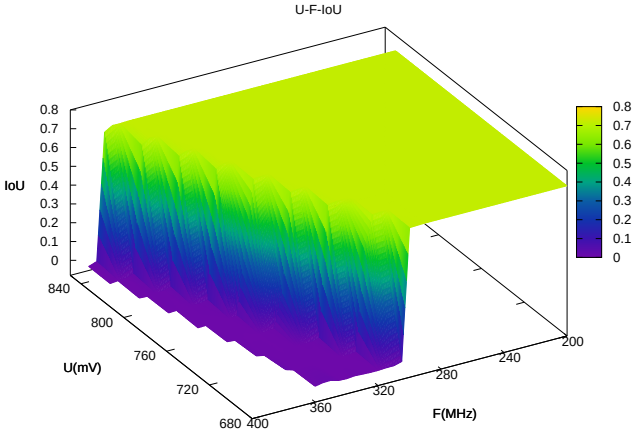


Figure 13: Average of IoU changes with voltage and frequency respectively.

frequency increases, it is also considered that the time for processing each picture as the frequency increases is also reduced. In such a scenario, the computational time reduction caused by the frequency increase may bring energy savings in the CPU part.

Fig.11 gives the relationship between the total energy consumption for processing all the 1000 images in the sample dataset and voltage/frequency obtained by our DVFS framework. As shown in Fig.11, at each voltage point, the total energy consumption goes down as the frequency increases. The reason is that when the accelerator's frequency runs at a low frequency, the dynamic power decreases but the static power remains. What's more, although the PS side's power consumption decreases as the operating frequency of accelerator decreases because the data throughput is reduced, the difference can be neglected compared to the total power of the PS. At the meantime, at each frequency point, the energy consumption performs a linear relationship with voltage. The highest energy consumption is 518.8J achieved at 200MHz/840mV while the least energy consumption is 354.1J achieved at 371MHz/840mV. About 32% of energy can be saved with our DVFS framework. It should be noted that all the data mentioned above is given under strict constraint that the result should be 100% correct. In the next section, we will relax the constrain and discuss the impact of DVFS on accuracy.

6.5. Accuracy Analysis

All the results presented so far have used the neural network at full accuracy, that is 71.9% Intersection over Union (IoU) on the whole sample dataset. As previously mentioned, it is possible to relax this constraint and let errors affect the user logic. As shown in Fig.12, there is a threshold frequency at 371MHz, above which there may be errors in the final results and the accuracy will change. However, the IoU does not go

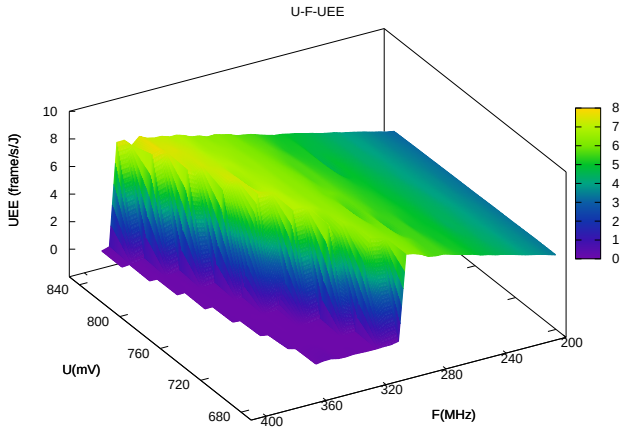


Figure 14: UEE changes with voltage and frequency respectively.

from 71.9% sharply to zero, but goes through four different stages, namely jitter period, slow decent period, quick decent period and zero period. In the first period, the IoU jitter around 71.9% within +1/-1% until 382MHz. In the second period, the IoU goes down gradually from 383MHz to 386MHz (70.8% to 68.5%). Only until the quick decent period (387MHz to 388MHz), we can see a sharp decrease in IoU. We regard the region where the IoU is below 10% as the zero period. This means that a further increase in performance is possible if we relax the IoU requirement to the jitter period and quick decent period.

It is foreseeable that the threshold frequency is related to voltage and Fig.13 shows the IoU varies concerning both voltage and frequency. The first error comes at 285MHz when VCCINT is set to 680mV and the threshold frequency comes later in an approximately linear relationship with the voltage. What's more, the jitter period, slow decent period and quick decent period narrow as the voltage goes up (285MHz-301MHz-304MHz at 680mV vs. 371MHz-382MHz-386MHz at 840mV).

Through the above analysis, we can draw a conclusion that, CNNs are robust to errors. Even if errors are generated by the accelerator during operation, it have a very limited effect on the final result when the number of errors is small. Through this feature, we can sacrifice accuracy and performance to some extent.

6.6. Comprehensive Evaluation Metric

Previous works usually compare performance, energy efficiency and the achieved accuracy respectively. However, these three metrics are not completely independent but can be transformed into each other through various means. For instance, network compression and quantization can be the method to balance between accuracy and performance. Our DVFS framework is a method to trade-off between energy efficiency and performance as well as accuracy. Therefore, a comprehensive evaluation metric that takes performance, energy efficiency and accuracy into account is necessary to evaluate a system design. In this paper, the performance is given by the achieved FPS, energy efficiency is given by the energy consumption for each frame and the accuracy is given by IoU. Under this context, we proposed a rough metric called unified energy efficiency (UEE):

$$UEE = \frac{Performance \times Accuracy}{Energy} \tag{5}$$

The UEE is given by:

$$UEE = \frac{FPS \times AverageIoU}{TotalEnergy} \tag{6}$$

There are previous works using frames per second per watt (FPS/W) as a metric. However, the watt is given by Joule per second and thus FPS/W equals to frames per Joule (FPJ), it does not take the performance into account. As a result, in this paper, we adapt energy rather than power as the numerator.

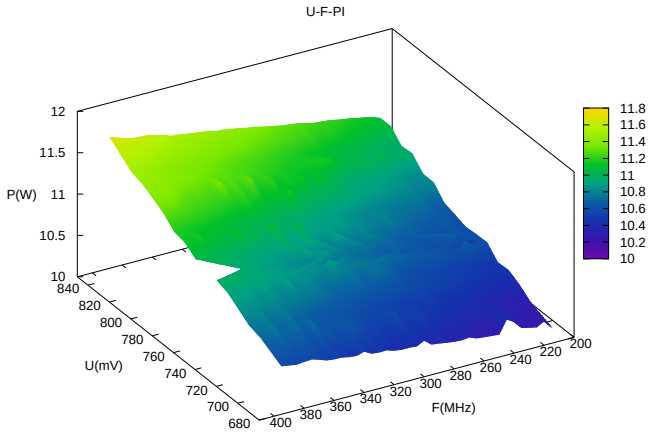


Figure 15: Idle power changes with voltage and frequency respectively.

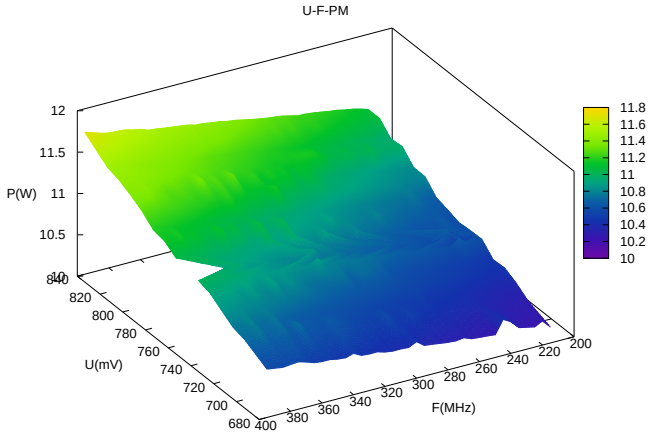


Figure 16: Average power changes with voltage and frequency respectively.

Fig.14 shows UEE changes with voltage and frequency respectively. As shown in the figure, UEE first goes up with frequency at each voltage since the accuracy remains and the performance and energy efficiency increase and then goes down sharply as IoU drops to zero. For each frequency point under 290MHz, the threshold frequency of 680mV, UEE decreases as voltage goes up since lower voltage has higher energy efficiency, However, higher voltage provides higher threshold frequency and thus can get higher performance and energy efficiency. The highest UEE, 7.71, is achieved at 840mV/379MHz and is beyond 7.22 achieved at 840mV/370MHz which is the threshold voltage. It also shows that the DVFS can further improve UEE.

6.7. DVFS for Real-Time Application

As mentioned in the previous section, in realistic scenarios, the needed frame rate is bounded by the camera. In this paper, we assume that the frame rate of the camera is 24FPS, the standard movie frame rate, and try to find the lowest average power solution that meets this performance requirement with our proposed DVFS policy. Fig.15 gives the relationship between idle power and voltage/frequency. As can be seen from the figure, the idle power performs a linear relationship between both the voltage and the frequency. Thus slowing down the clock, which can be regarded as a variant of clock gating technology, can be an effective method to save idle power. Furthermore, scaling down the voltage combined with clock gating can save idle power to the utmost. P_{Imin} is 9.81W, achieved when the frequency is scaled to 20MHz and the voltage is scaled to 680mV.

Table 2: Comparison with other work

	Performance	Energy Efficiency	IoU	UEE
SkyNet [25]	23.93FPS	2.02FPJ	71.91%	3.49
Candidate1	37.04FPS, 1.54×	2.79FPJ, 1.38×	71.91%	7.22, 2.06×
Candidate2	37.42FPS, 1.56×	2.97FPJ, 1.47×	71.80%	7.71, 2.21×

Fig.16 shows the achieved P_M at different voltage/frequency combinations. As shown in the figure, the minimum average power is 10.90W and is achieved at 680mV/285MHz, we get a 15% reduce in power consumption compared to 12.86W of the original design.

6.8. Comparison with Other Work

We compare the performance, energy efficiency, accuracy as well as UEE with the original SkyNet, where the energy efficiency is given by frames per Joule (FPJ). The original SkyNet is implemented on Ultra96, we transfer the design to ZCU104 without any modification. We choose the data achieved at the threshold frequency of 840mV/370MHz as the first candidate and that achieved at 840mV/379MHz as the second candidate. As shown in Table.2, our work achieves 54% improvement in performance, 38% improvement in energy efficiency and 106% improvement in UEE without any degradation in accuracy compared to the original SkyNet. If we relax the requirement on accuracy, we can achieve 56% improvement in performance, 47% improvement in energy efficiency and 121% improvement in UEE at the cost of 0.11 decrease in IoU.

7. Conclusion

FPGA has been proven as a favorable platform to accelerate convolution neural network (CNN) on the edge-computing paradigm given its high flexibility and energy efficiency. However, the energy efficiency of the FPGA platform can be further improved with dynamic voltage and frequency scaling (DVFS). Although an overly aggressive voltage and frequency combination can lead to errors, the high robustness of the CNN to errors makes it possible to combine DVFS with it. In this paper, we first introduce a framework for fine-grained DVFS on the state-of-the-art FPGA device and then apply the framework to SkyNet, a state-of-the-art neural network targeting on object detection. Third, we analyze the impact of DVFS on performance, power, energy efficiency and accuracy. We verify the possibility of sacrificing accuracy for performance or energy efficiency. In order to evaluate the entire system comprehensively, we propose a new metric, called unified energy efficiency (UEE), that takes performance, energy efficiency as well as accuracy into account. Finally, we achieve 54% improvement in performance, 38% improvement in energy efficiency and 106% improvement in UEE without any loss in accuracy compared to the original SkyNet. If we relax the requirement on the accuracy, we can achieve 56% improvement in performance, 47% improvement in energy efficiency and 121% improvement in UEE at the cost of 0.11 decrease in accuracy.

References

[1] Eriko Nurvitadhi, Ganesh Venkatesh, Jaewoong Sim, Debbie Marr, Randy Huang, Jason Ong Gee Hock, Yeong Tat Liew, Krishnan Srivatsan, Duncan Moss, Suchit Subhaschandra, et al. Can fpgas beat gpus in accelerating next-generation deep neural networks? In Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, pages 5–14. ACM, 2017.

[2] Paolo Mantovani, Emilio G Cota, and Kevin et al. Tien. An fpga-based infrastructure for fine-grained dvfs analysis in high-performance embedded systems. In Proceedings of the 53rd Annual Design Automation Conference, page 157. ACM, 2016.

[3] Lin Bai, Yiming Zhao, and Xinming Huang. A cnn accelerator on fpga using depthwise separable convolution. IEEE Transactions on Circuits and Systems II: Express Briefs, 65(10):1415–1419, 2018.

[4] Yufei Ma, Yu Cao, Sarma Vrudhula, and Jae-sun Seo. An automatic rtl compiler for high-throughput fpga implementation of diverse deep convolutional neural networks. In 2017 27th International Conference on Field Programmable Logic and Applications (FPL), pages 1–8. IEEE, 2017.

[5] Yufei Ma, Yu Cao, Sarma Vrudhula, and Jae-sun Seo. Optimizing loop operation and dataflow in fpga acceleration of deep convolutional neural networks. In Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, pages 45–54. ACM, 2017.

[6] Yufei Ma, Minkyu Kim, Yu Cao, Sarma Vrudhula, and Jae-sun Seo. End-to-end scalable fpga accelerator for deep residual networks. In 2017 IEEE International Symposium on Circuits and Systems (ISCAS), pages 1–4. IEEE, 2017.

[7] Xuechao Wei, Yun Liang, Xiuhong Li, Cody Hao Yu, Peng Zhang, and Jason Cong. Tgpa: tile-grained pipeline architecture for low latency cnn inference. In 2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pages 1–8. IEEE, 2018.

[8] Kaiyuan Guo, Lingzhi Sui, Jiantao Qiu, Jincheng Yu, Junbin Wang, Song Yao, Song Han, Yu Wang, and Huazhong Yang. Angel-eye: A complete design flow for mapping cnn onto embedded fpga. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 37(1):35–47, 2018.

[9] Yufei Ma, Yu Cao, Sarma Vrudhula, and Jae-sun Seo. Performance modeling for cnn inference accelerators on fpga. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2019.

[10] Jiantao Qiu, Jie Wang, Song Yao, Kaiyuan Guo, Boxun Li, Erjin Zhou, Jincheng Yu, Tianqi Tang, Ningyi Xu, Sen Song, et al. Going deeper with embedded fpga platform for convolutional neural network. In Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, pages 26–35. ACM, 2016.

[11] Xiaofan Zhang, Junsong Wang, Chao Zhu, Yonghua Lin, Jinjun Xiong, Wen-mei Hwu, and Deming Chen. Dnnbuilder: an automated tool for building high-performance dnn hardware accelerators for fpgas. In Proceedings of the International Conference on Computer-Aided Design, page 56. ACM, 2018.

[12] Mohammad Motamedi, Daniel Fong, and Soheil Ghiasi. Machine intelligence on resource-constrained iot devices: The case of thread granularity optimization for cnn inference. ACM Transactions on Embedded Computing Systems (TECS), 16(5s):151, 2017.

[13] Qingcheng Xiao, Yun Liang, Liqiang Lu, Shengen Yan, and Yu-Wing Tai. Exploring heterogeneous algorithms for accelerating deep convolutional neural networks on fpgas. In 2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC), pages 1–6. IEEE, 2017.

[14] Sanghamitra Dutta, Ziqian Bai, Tze Meng Low, and Pulkit Grover. Codenet: Training large scale neural networks in presence of soft-errors. arXiv preprint arXiv:1903.01042, 2019.

[15] Bin Nie, Devesh Tiwari, Saurabh Gupta, Evgenia Smirni, and James H Rogers. A large-scale study of soft-errors on gpus in the field. In 2016 IEEE International Symposium on High Performance Computer Architecture (HPCA), pages 519–530. IEEE, 2016.

[16] Yuanchang Chen, Yizhe Zhu, Fei Qiao, Jie Han, Yuansheng Liu, and Huazhong Yang. Evaluating data resilience in cnns from an approximate memory perspective. In Proceedings of the on Great Lakes Symposium on VLSI 2017, pages 89–94. ACM, 2017.

[17] Aurick Qiao, Bryon Aragam, Bingjing Zhang, and Eric P Xing. Fault tolerance in iterative-convergent machine learning. arXiv preprint arXiv:1810.07354, 2018.

[18] Jose Luis Nunez-Yanez. Adaptive voltage scaling with in-situ detectors in commercial fpgas. IEEE Transactions on Computers, 64(1):45–53, 2014.

[19] Atukem Nabina and Jose Luis Nunez-Yanez. Adaptive voltage scaling in a dynamically reconfigurable fpga-based platform. ACM Transactions on Reconfigurable Technology and Systems (TRETS), 5(4):20, 2012.

[20] Xuechao Wei, Yun Liang, and Jason Cong. Overcoming data transfer bottlenecks in fpga-based dnn accelerators via layer conscious memory management. In DAC, pages 125–1, 2019.

[21] Caiwen Ding, Shuo Wang, Ning Liu, Kaidi Xu, Yanzhi Wang, and Yun Liang. Req-yolo: A resource-aware, efficient quantization framework for object detection on fpgas. In Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, pages 33–42. ACM, 2019.

[22] Xiaofan Zhang, Cong Hao, Yuhong Li, Yao Chen, Jinjun Xiong, Wen-mei Hwu, and Deming Chen. A bi-directional co-design approach to enable deep learning on IoT devices. arXiv preprint arXiv:1905.08369, 2019.

[23] Cong Hao, Xiaofan Zhang, Yuhong Li, Sitao Huang, Jinjun Xiong, Kyle Rupnow, Wen-mei Hwu, and Deming Chen. FPGA/DNN co-design: An efficient design methodology for IoT intelligence on the edge. In Proceedings of the 56th Annual Design Automation Conference, page 206. ACM, 2019.

[24] Jose L. Nunez-Yanez. Energy proportional neural network inference with adaptive voltage and frequency scaling. IEEE Transactions on Computers, PP(99):1–1, 2018.

[25] Xiaofan Zhang, Cong Hao, Haoming Lu, Jiachen Li, Yuhong Li, Yuchen Fan, Kyle Rupnow, Jinjun Xiong, Thomas Huang, Honghui Shi, Wen-mei Hwu, and Deming Chen. Skynet: A champion design for DAC-SDC on low power object detection. arXiv preprint arXiv:1906.10327, 2019.

[26] Andreas Weissel and Frank Bellosa. Process cruise control: event-driven clock scaling for dynamic power management. In Proceedings of the 2002 international conference on Compilers, architecture, and synthesis for embedded systems, pages 238–246. ACM, 2002.

[27] Karel De Vogeleer, Gerard Memmi, Pierre Jouvelot, and Fabien Coelho. The energy/frequency convexity rule: Modeling and experimental validation on mobile devices. In International Conference on Parallel Processing and Applied Mathematics, pages 793–803. Springer, 2013.

[28] Huang Huang, Vivek Chaturvedi, Gang Quan, Jeffrey Fan, and Meikang Qiu. Throughput maximization for periodic real-time systems under the maximal temperature constraint. ACM Transactions on Embedded Computing Systems (TECS), 13(2s):70, 2014.

[29] Heng Yu, Rizwan Syed, and Yajun Ha. Thermal-aware frequency scaling for adaptive workloads on heterogeneous mpsocs. In Proceedings of the conference on Design, Automation & Test in Europe, page 291. European Design and Automation Association, 2014.

[30] Heng Yu, Yajun Ha, and Jing Wang. Quality optimization of resilient applications under temperature constraints. In Proceedings of the Computing Frontiers Conference, pages 9–16. ACM, 2017.

[31] Yue Ma, Thidapat Chantem, Robert P Dick, and Xiaobo Sharon Hu. Improving system-level lifetime reliability of multicore soft real-time systems. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 25(6):1895–1905, 2017.

[32] Kyeongryeol Bong, Sungpill Choi, Changhyeon Kim, and Hoi-Jun Yoo. Low-power convolutional neural network processor for a face-recognition system. IEEE Micro, 37(6):30–38, 2017.

[33] Giulia Santoro, Mario R Casu, Valentino Peluso, Andrea Calimera, and Massimo Alioto. Design-space exploration of pareto-optimal architectures for deep learning with dvfs. In 2018 IEEE International Symposium on Circuits and Systems (ISCAS), pages 1–5. IEEE, 2018.

[34] Guan-Chyun Hsieh and James C Hung. Phase-locked loop techniques. a survey. IEEE Transactions on industrial electronics, 43(6):609–615, 1996.

[35] J-H Kim, Y-H Kwak, Mooyoung Kim, S-W Kim, and Chulwoo Kim. A 120-mhz–1.8-ghz cmos dll-based clock generator for dynamic frequency scaling. IEEE Journal of Solid-State Circuits, 41(9):2077–2082, 2006.

[36] Ian Brynjolfson and Zeljko Zilic. Dynamic clock management for low power applications in fpgas. In Proceedings of the IEEE 2000 Custom Integrated Circuits Conference (Cat. No. 00CH37044), pages 139–142. IEEE, 2000.

[37] Arash Farhadi Beldachi and Jose L Nunez-Yanez. Run-time power and performance scaling in 28 nm fpgas. IET Computers & Digital Techniques, 8(4):178–186, 2014.

[38] Arash Farhadi Beldachi and Jose L Nunez-Yanez. Accurate power control and monitoring in zynq boards. In 2014 24th International Conference on Field Programmable Logic and Applications (FPL), pages 1–4. IEEE, 2014.

[39] Mohammad Hosseinabady and Jose Luis Nunez-Yanez. Run-time power gating in hybrid arm-fpga devices. In 2014 24th International Conference on Field Programmable Logic and Applications (FPL), pages 1–6. IEEE, 2014.