# Stagioni: Temperature Management to Enable Near-sensor Processing for Performance, Fidelity, and Energy-efficiency of Vision and Imaging workloads

by

Venkatesh Kodukula

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved September 2018 by the
Graduate Supervisory Committee:

Robert LiKamWa, Chair
Chaitali Chakrabarti
John Brunhaver

ARIZONA STATE UNIVERSITY

December 2018

# ABSTRACT

Vision processing on traditional architectures is inefficient due to energy-expensive off-chip data movements. Many researchers advocate pushing processing close to the sensor to substantially reduce data movements. However, continuous near-sensor processing raises the sensor temperature, impairing the fidelity of imaging/vision tasks.

The work characterizes the thermal implications of using 3D stacked image sensors with near-sensor vision processing units. The characterization reveals that near-sensor processing reduces system power but degrades image quality. For reasonable image fidelity, the sensor temperature needs to stay below a threshold, situationally determined by application needs. Fortunately, the characterization also identifies opportunities – unique to the needs of near-sensor processing – to regulate temperature based on dynamic visual task requirements and rapidly increase capture quality on demand.

Based on the characterization, the work proposes and investigate two thermal management strategies – stop-capture-go and seasonal migration – for imaging-aware thermal management. The work present parameters that govern the policy decisions and explore the trade-offs between system power and policy overhead. The work's evaluation shows that the novel dynamic thermal management strategies can unlock the energy-efficiency potential of near-sensor processing with minimal performance impact, without compromising image fidelity.

ACKNOWLEDGMENTS

First of all, I acknowledge Dr. Robert LiKamWa for his great mentorship over the past two years. He really helped shape my thinking as a researcher. I am very fortunate to be a part of Meteor studio group. I would like to thank all the group members, especially Bharadwaj Medapuram, Saad Katrawala, and Britton Jones, with whom I collaborated for this project. Special thanks to Siddhanth Prakash, Sridhar Gunnam, Ali Bahremand, Jinhan Hu, Paul Nathan, Shahabedin Sagheb, and Alexander Shearer for great discussions.

I would like to thank Dr. Chaitali Chakrabarti and Dr. John Brunhaver for their insights and feedback on the early ideas of this project. I thank Dr. Carole-Jean Wu and Dr. Patrick Phelan for their guidance on thermal modeling.

I thank my friends, especially Hidayatullah Chowdhary, Nagendar Reddy Ponagandla, Ravi Teja Chikkam, and Sameer Nekkalapu for the many lively conversations and a great social life outside the lab. I always look forward to those moments of camaraderie.

Most importantly, I deeply thank my parents Nageswararao and Latha Kodukula for their unconditional support and for providing me freedom to pursue my dreams from my childhood. I am grateful to my sister Vandana Kodukula who always stood my side helping me through while making important decisions in my life.

I also acknowledge the Arizona State University's School of Electrical, Energy, and Computer Engineering, School of Arts and Media Engineering, and the National Science Foundation for academically and financially supporting my graduate career.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

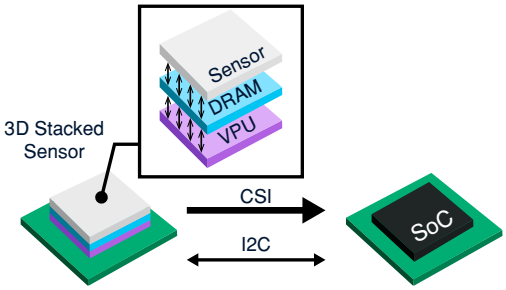
# Chapter 1

# INTRODUCTION

Imaging and vision systems allow computing systems to sense and react to real-world situations and to capture images for human consumption. This affords a range of utilities on many devices spanning a wide variety of power profiles, including smartphones and tablets, wearable headsets, personal computers, security cameras, drones, automobiles, and security and monitoring systems. Unfortunately, imaging requires high data rates to transfer pixel data from the image sensor to computational units. In traditional systems (Fig. 1.1a), where the computational units are separated from the sensor via long interconnects, e.g., ribbon cables, data rates create bottlenecks to energy efficiency and processing. Thus, current vision systems result in power profiles on the order of multiple watts; it has been shown that state-of-the-art convolutional neural network efficiency need at least 1 W of processing power to process low resolution QVGA frames at 30 fps Azarkhish *et al.* (2018); Pena *et al.* (2017). For high performance processing at high resolutions and framerates, the power requirements rapidly rise, easily going up to over 10 W of processing power on mobile-based implementations Cavigelli *et al.* (2015).

This has motivated a trend towards three-dimensional "stacked" integrated circuit architectures (Fig. 1.1b) for sensor capture and processing, i.e., *near sensor processing*. A 3D stacked sensor stacks the sensor, vision processor unit (VPU), and memory on top of each other in the same package. By processing data near the sensor, various proposed and implemented systems can achieve energy-efficient vision processing LiKamWa *et al.* (2016); Du *et al.* (2015), and bursts of high-speed capture Nose *et al.* (2018). With advances in fabrication, stacked image sensors have


(a) Traditional vision pipeline.




(b) Near-sensor vision pipeline.

Figure 1.1: Due to Energy-expensive Interface Data Movements, Traditional Pipelines Are Inefficient. Near-sensor Processing Helps Greatly Reduce Data Traffic Promoting Energy-efficiency.

been commercially released since 2012 3D-IC-blog (2016), and are still under active development for high performance Lee *et al.* (2012) and efficiency Amir *et al.* (2018).

Unfortunately, sensor sensitivity to temperature prevents a full adoption of near-sensor processing, creating noise in captured images. Furthermore, low light environments force the sensor to operate at high exposure and ISO [1] to capture the scene, which increases a sensor's vulnerability to noise. Despite a plethora of CPU dynamic thermal management (DTM) mechanisms, current techniques do not suffice imaging requirements; traditional DTM reduces package cooling costs and maintains maximum temperature limits, i.e., TDP, turning a blind eye to the transient imaging needs of near-sensor processing. Thus, despite performance and energy benefits of near-sensor processing, the temperature profile of visual computing limits stacked architectures in many situations.

[1] ISO controls the sensitivity of an image sensor to light

To assess thermal issues, in §3 we characterize the temperature implications of stacked near-sensor processing for visual workloads. In addition to confirming and modeling relationships between near-sensor processing power and sensor temperature, our characterization reveals a consequential insight: despite the long time constants for the sensor to settle to steady-state temperatures, *removing near-sensor power results in an immediate and dramatic reduction in transient junction temperature of the sensor*. For example, for a 2.5 W system, the sensor temperature drops by 13° $C$ in 20 ms, when we turn off the processing. This stems from the high thermal capacitance of chip packaging and low thermal capacitance of the die. This immediate temperature drop is neglected by existing dynamic thermal management works, whose primary aim is to confine chip temperature below an emergency limit. However, as reducing transient temperature raises sensing fidelity, this observation allows on-demand high-fidelity capture.

In §4, we build on characterized challenges and opportunities to provision for imaging-specific temperature management. We design two mechanisms – stop-capture-go and seasonal migration – for effective near-sensor vision processing that minimizes system energy consumption and affords performance computation and high fidelity capture. Stop-capture-go suspends the processing briefly to allow for a high fidelity capture and resumes the processing after the capture. On the other hand, seasonal migration occasionally shifts processing to a thermally isolated far-sensor processing unit for high fidelity capture. We design *Stagioni*, a runtime that orchestrates the temperature management for near-sensor processing.

In §6, we evaluate the effectiveness of our mechanisms in managing sensor temperature to suit imaging needs. We also demonstrate Stagioni's robustness in smoothly handling the dynamic fidelity needs. In §7, we contextualize our work by discussing

future avenues of research towards temperature-aware stacked sensor architectures for near-sensor processing.

**Vision case study - Continuous life-logger:** Enabling high performance and high efficiency near-sensor processing would unlock the potential for several vision/imaging applications, including sophisticated dashboard cameras, continuous augmented reality tracking, and other futuristic use cases. Throughout this work, we study the implications of near-sensor processing and evaluate the policies around a life-logger case study. A wearable life-logger device chronicles important events and objects in a person's life. The device runs object detection and tracking algorithms to continuously figure out the objects in the scene and track them. Meanwhile, the device performs occasional captures upon detecting any important event, e.g., a person entering the scene. This can form the basis for personalized real-world search engines, and assist those with memory impairments or visual impairments.

Chapter 2

# BACKGROUND AND RELATED WORK

**Near-sensor processing paradigm:** The paradigm of near-sensor processing emerged in the early 1990s to reduce the communication and storage overhead of off-sensor processing. Early works Forchheimer and Astrom (1994) leveraged the physical properties of the sensor to perform low-level image processing tasks, e.g., median filtering. Later, researchers integrated image processing units Shi and Lichman (2005) after the read-out circuits in the imaging plane, outputting extracted image feature. With advancement in 3D circuit integration technology, recent works Amir *et al.* (2018) design 3D stacked image sensors, which include a system on a chip (SoC). Inside the SoC, sensor, processor, and memory are stacked into the same package. This architecture performs high-level image processing tasks, such as ConvNet-based classification.

3D stacked architectures have also seen commercial advances. For slow-motion capture, Sony Haruta *et al.* (2017) stacked a DRAM beneath the sensor layers. With local memory, the sensor captures and buffers frames at 1000 fps, later sending them across the slower camera interface to the host. Samsung TechInsights (2018) uses a similar sensor for their recent Galaxy smartphone. For surveillance, Sony Kumagai *et al.* (2018) integrated a motion estimation block, microcontroller, and DRAM in the 3D stacked sensor.

**VPU architectures and power profiles:** Though vision can be done through handcrafted feature analysis Lowe (1999), the current trend uses Convolutional Neural Networks (ConvNets) for visual tasks on a wide range of architectures. High

programmability, performance, and energy-efficiency are desired to meet the rapid pace at which ConvNets are evolving.

General-purpose platforms built around GPUs provide highly programmable and high performance software libraries Jia *et al.* (2014); Abadi *et al.* (2016) to implement ConvNets at the expense of more power, e.g., 60 fps at 10s to 100s of watts BVLC (2018); Pham *et al.* (2012); Cavigelli *et al.* (2015). FPGAs also provide high performance and scalability, but at reduced power. The state-of-the-art FPGA implementations Zhang *et al.* (2015); Pham *et al.* (2012); Zhang *et al.* (2016) typically consume several watts of power. In recent years, we see the rise of domain specific processors such as Myriad2 Pena *et al.* (2017) that provide programmable SIMD capabilities on a RISC processor. This brings down the power to a few watts Pena *et al.* (2017), but at the cost of performance, e.g., 3 fps. Meanwhile, academic ASICs Chen *et al.* (2016); Han *et al.* (2016); Du *et al.* (2015) provide energy-efficiency and performance for ConvNets. However, the benefits are heavily bottlenecked by DRAM accesses. For example, Eyeriss Chen *et al.* (2016) achieves 278 mW @ 35 fps for AlexNet. But when scaled for VGG16 Simonyan and Zisserman (2014), performance drops to about 10 fps within the same power budget.

For reasonable performance, scalability, and mobility, the system power profile ranges from 1 - 15 W. Placing these VPUs near the sensor and solving temperature challenges would unlock substantial improvements in performance and energy-efficiency through near-sensor processing.

**Thermal noise in image sensors:** Image sensors are susceptible to different types of noise due to imperfections in lighting, sensing elements, and the underlying imaging circuitry. Sources of noise can be grouped into fixed-pattern noise and temporal noise. Fixed-pattern noise arises due to non-uniform sensitivities of photodiodes to light. As it remains constant over time, conventional strategies read it once and subtract it later

to eliminate its effect. In contrast to fixed-pattern noise, temporal noise sources vary with every capture.

Temporal noise sources include read noise and dark current shot noise, which exhibit strong dependence on temperature. All electronic noise sources, e.g., readout elements, amplifiers, are grouped together as read noise, which has a variance of kT/C. This noise is due to random thermal activity of the electronic charge carriers. Dark current shot noise also stems from similar phenomenon happening in photodiodes; high temperatures trigger randomness in the photodiode charge carriers, thereby inducing more noise in images. Unfortunately, thermal noise cannot be fully corrected using signal processing techniques without generating imaging artifacts Levoy (2014). The only solution is to manage the sensor temperature.

**Dynamic thermal management in microprocessors:** For efficient thermal management, different techniques have been explored for multi-core processors. Stop-and-go Brooks and Martonosi (2001) suspends the execution of a thread, for a while, when a core on which it is running gets overheated and resumes its operation once the core cools down. Heat-and-run Gomaa *et al.* (2004) technique migrates the thread from a hotter core to a cooler one to allow the hotter core to cool down. Traditional DTM techniques are designed to keep the processor power within a thermal design power (TDP). We are inspired by the same core mechanisms – stop-go and seasonal migration – for power and temperature reduction. In contrast to the existing works, we redesign these mechanisms to fulfill the dynamic imaging needs.

**Thermal problems in 3D stacked image sensor:** Recent works report temperature issues in 3D stacked image sensors. One of them Amir *et al.* (2018) stack a DRAM and a deep neural network (DNN) processor beneath the sensor layer. They report that sensor temperature can increase due to DNN computation, resulting in higher noise and lower ConvNet accuracy. Another work Lie *et al.* (2014) report

similar issues for their 3-layer stack architecture with a image compression unit integrated inside the stack. Similar to earlier works, we report similar issues for our characterized 3D stacked image sensor. However, previous works provide *design time* solutions. e.g., statically partitioning computation to execute partial ConvNets on the sensor and the rest on the host. Our work is complementary to theirs by providing *runtime solutions* for thermal management.

Chapter 3

# MODELING THE ENERGY, TEMPERATURE, AND FIDELITY IMPLICATIONS OF NEAR SENSOR PROCESSING

In this section, we examine the implications of using 3D stacked integration to place a VPU layered underneath an image sensor for near-sensor processing. In particular, we study the relationship of near-sensor processing with system energy, sensor temperature, and image noise. Our studies confirm that near-sensor processing minimizes the off-chip data movements, thereby substantially reducing system power. With near-sensor processing in our case study, we can reduce the system power of ResNet-based classification by 36%.

We also relate near-sensor processing power to image fidelity through temperature simulation, confirming that image fidelity degrades over time with additional near-sensor processing power. However, we also observe that removal of near-sensor processing power favorably leads to rapid drops in sensor temperature, reducing sensor temperature by 13° C in 20ms. We can exploit this observation to allow the sensor to operate at higher temperatures and lower image fidelities for energy-efficient vision, e.g., continuous object detection, while immediately switching to low temperature operation for high-fidelity image capture when an application needs high quality imaging, e.g., photographing a particular object.

### *3.0.1 Energy-efficiency of near-sensor processing*

Near-sensor processing reduces energy-expensive data movement across the lengthy interconnects between different chips. Here, we examine energy profiles of vision pipelines, comparing traditional and near-sensor processing pipelines.

Table 3.1: Energy-per-pixel of Various Components in the Traditional Vision Pipeline. Communication Cost Is Atleast an Order of Magnitude More than Other Costs.

| **Component** | **Energy (pJ/pixel)** |
|---|---|
| Sensing | 595 |
| Communication (Sensor - SoC) | 900 |
| Communication (SoC - DRAM) | 2800 |
| Storage (Read) | 283 |
| Storage (Write) | 394 |

**Energy analysis of vision pipeline components**

Traditional pipelines operate across chips to connect a variety of subsystems: camera, processing unit, memory, as shown in Fig. 1.1a. The camera chip connects to processing units on the System-on-Chip (SoC) through a standard camera serial interface (CSI) for data transfer and an $I^2C$ interface for control and configuration. Meanwhile, the SoC uses DRAM through an external DDR interface to buffer image frames for processing.

Using regression models on measurements and reported values, we construct a coarse energy profile model to motivate the need for near-sensor processing. As shown in Table 3.1, we find that sensing, processing, and storage consume energy on the order of 100s of pJ per pixel. On the other hand, communication interfaces draw more than 3 nJ per pixel.

*Sensing* requires an energy of 595 pJ/pixel LiKamWa *et al.* (2013); Choi *et al.* (2015), mostly drawn from three components: pixel array, read-out circuits, and analog signal chain, which consume 25 pJ/pixel, 43 pJ/pixel, and 527 pJ/pixel, respectively. *DRAM storage* on standard mobile-class memory chips (8 Gb, 32-bit

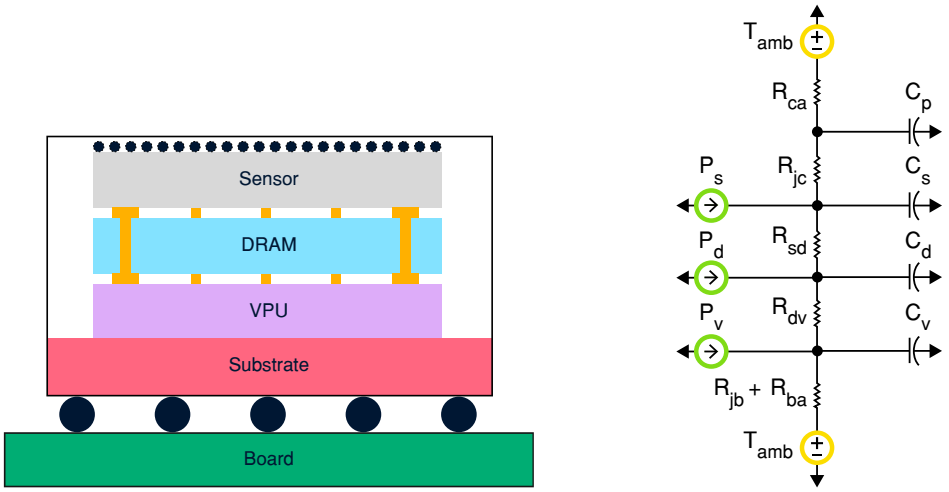
LPDDR4) draws 677 pJ/pixel for writing and reading a pixel value technologies (2018). This roughly divides into 283 pJ/pixel for reading and 394 pJ/pixel for writing. *Communication* over CSI and DDR interfaces incur 3.7 nJ/pixel, mostly due to operational amplifiers on both transmitter and receiver. We measure the interface power consumption Xilinx (2018b) on 4-lane CSI interfaces and LPDDR4 interfaces by inputting several data rates. From this information, we construct a linear-regression model to estimate the energy per pixel to be 0.9 nJ/pixel over CSI and 2.8 nJ/pixel over DDR. For *computation*, we gather reported power consumptions of various ConvNet architectures from the literature.

**Energy of trad. sensor processing architecture**

To present different architectures, Table 6.1 lists estimated system power numbers alongside the type of ConvNet and the performance of the processing. We combine reported computation values with modeled sensing, storage, and communication costs to estimate system power. When operating at FullHD (1920 X 1080) @ 30 fps, and using ResNet for inference on the SoC VPU at 30 fps, the modeled system power uses 4 W. On the other hand, increasing the framerate to 60 fps demands 10 W of power on an FPGA. Our energy models provide coarse estimation; actual numbers will depend on architectural decisions, patterns of execution, and several other factors.

**Energy of near-sensor processing architecture**

On-chip data movement is known to be significantly more efficient than off-chip data movement by six orders of magnitude Borkar (1999). Advances in near-sensor processing leverage this insight for energy-efficiency gains, as shown in Fig. 1.1b. Near-sensor processing moves the DRAM into the sensor to eliminate off-chip DDR movement, and moves the VPU into the sensor to reduce the CSI interface data rate. Thus, the




(a) Cross-section of 3D stack.

(b) RC model of stack.

Figure 3.1: Using the Well-known Duality Between Thermal and Electrical Phenomena, Thermal Modeling of Stacked Sensors Can Be Performed by Analyzing an Equivalent RC Circuit.

output of the sensor can be reduced from a few MB to a few bytes. This information can be sent across efficient low data rate interfaces, e.g., $I^2C$. Altogether, when applying our energy profile models to the near sensor processing pipeline, we find that the FullHD VPU near sensor system consumes 2.5 W, thereby yielding 36% savings over traditional architectures.

### *3.0.2 Thermal implications of near-sensor processing*

Though tight integration yields energy-efficiency and performance benefits, near-sensor processing generates heat at the sensor through thermal coupling between the tightly integrated components. While dynamic thermal management for CPU is only concerned with keeping the maximum temperature below a TDP, we pay close attention to temperature patterns, as the transient temperature affects image fidelity. Conduction is the dominant heat transfer mechanism in integrated circuits. To model

temperature dynamics, we use thermal resistance-capacitance (RC) modeling Skadron *et al.* (2002) techniques to determine the thermal characteristics of the stacked sensor.

### Deriving the component values in RC model

Fig. 3.1 shows a typical structure of a 3D stacked sensor package and its equivalent RC model. Inside the package, the sensor, DRAM, and VPU layers are stacked on top of each other. The layers can be connected to each other using through-silicon-vias (TSVs). The top of the stack opens to the surroundings through microlenses, while the bottom of the stack sits on a substrate that opens to the printed circuit board (PCB). Mobile-class image sensors omit heat sinks or cooling fans, due to their size, weight, and placement challenges. The layers consume power when active, which dissipates as heat. We primarily consider vertical heat transfer; vertical resistances are several orders of magnitude smaller than the lateral resistances of convective heat transfer. We obtain component values of the layers through a mixture of analytical and empirical approaches.

Table 3.2 shows different RC component values derived for our model. Previous works report layer dimension values of typical 3D stacked image sensors Amir *et al.* (2018). Table 3.2 shows different RC component values derived for our model. In these works, the layer thickness ranges in the order of a few microns to 10s of microns, while the layer's area ranges from 10s of $mm^2$ to 100s of $mm^2$. The ITRS roadmap provides layer dimensions and material property constants $\rho$ and $c$ to define the guidelines for semiconductor fabrication. From these, we derive the thermal resistance $R = \rho t/A$ and thermal capacitance as $C = ctA$ where A is the layer's cross sectional area and t the thickness.

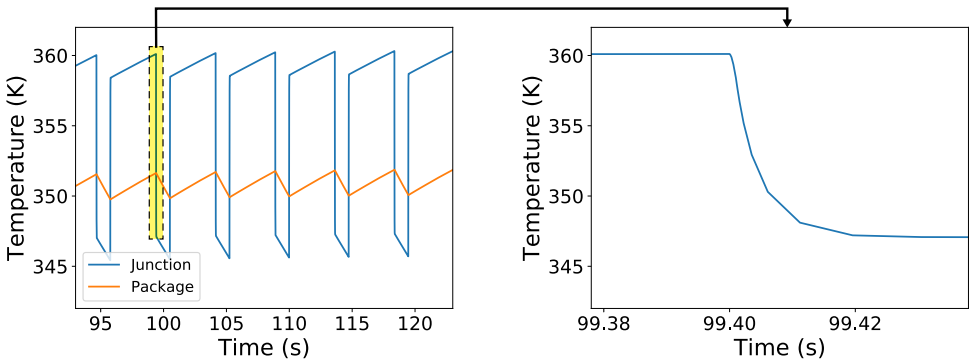
Package capacitance can be deduced empirically by observing the temperature trace of an image sensor chip while subjecting the sensor to thermal stress. We con-

Table 3.2: Thermal Resistance and Capacitance Values of Different Components in RC Model of Stack.

| **Component** | **R (K/W)** | **Layer** | **C (J/K)** |
|---|---|---|---|
| $R_{ca}$: Case-to-Ambient | 56 | $C_p$: Package | 1 |
| $R_{jc}$: Junction-to-Case | 6 | $C_s$: Sensor | 0.65m |
| $R_{sd}$: Sensor-to-DRAM | 0.6 | $C_d$: DRAM | 0.65m |
| $R_{dv}$: DRAM-to-VPU | 0.6 | $C_v$: VPU | 0.65m |
| $R_{jb}$: Junction-to-Board | 40 | | |
| $R_{ba}$: Board-to-Ambient | 14 | | |

struct regression models from the temperature trace of an OnSemi AR0330 smartphone-class image sensor to derive package capacitance. Finally, termination thermal resistance depends on the type of casing and board properties. Sensor companies make these values available through datasheets. We use such provided values for typical packages directly in our model.

**Sidenote: Off-sensor power does not affect sensor temperature.** While processing far from the sensor, the off-sensor VPU and SoC components do not influence the sensor temperature. Even in tightly integrated mobile systems, e.g., smartphones, the sensor and SoC reside on two different boards and communicate over a ribbon cable. As a result, the sensor and SoC are nearly in thermal isolation. That is, any increase in temperature of one component will not cause appreciable change in temperature of the other. We verify this effect by running a CPU-bound workload on SoC on a Google Nexus smartphone while keeping the camera idle. Our instruments do not report any rise in camera temperature with rise in SoC temperature. Thus, in




(a) Junction temperature trace with downward vertical lines indicating the immediate jump.

(b) Zoomed-in version of the jump: the junction temperature drops by 13°C within 20 ms

Figure 3.2: When Disabling Nsp, a Jump in Junction Temperature Occurs Within 20 Ms, Due to Ms-scale Junction Time Constants.

our study, we do not consider thermal coupling effects from off-sensor components.

### Simulation-based thermal analysis

Through LTSpice simulation on our RC models, we estimate the thermal behavior of near-sensor processing architectures. We evaluate temperature profiles as the sensor operates in two different modes: *NSP* mode, in which power consumptions are representative of capturing image frames and processing vision workloads near the sensor, and *CAP* mode, in which power consumptions are representative of capturing image frames and transmitting frames to the SoC. With various execution patterns, we can simulate the thermal behavior of the sensor as the system operates among different sensor modes.

Previous analysis has reported that we can safely ignore spatial variations in temperature if the chip power density is within 20 W/$cm^2$ Yu and Wu (2018), as is the case in NSP mode. Power density, which is the power dissipated over the chip

area, measures the degree of spatial non-uniformities in temperature. The physical dimensions of our 3D stacked image sensor combined with the power profile of our case study results in a power density of 16 W/$cm^2$. Therefore, we do not consider the spatial variations of temperature inside the stack for our modeling near-sensor processing architectures.

**Steady-state temperature:** Inter-layer resistances are at least two orders of magnitude smaller than termination resistances. This results in negligible drop across the resistor, leading to minuscule temperature gradients among a layer. For example, for 1 W of VPU power, the sensor, DRAM, and VPU will be at 60.7°C, 60.9°C, and 61.0°C, respectively. Thus, we can combine the layers and treat the sensor's temperature as a single junction. Consequently, termination resistance largely influences the sensor junction's steady-state temperature.

In addition to resistances, power consumption plays a crucial role in deciding steady-state. High power dissipates more heat in the physical structures resulting in a hotter junction. Conversely, low power consumptions relieves the heat generation, allowing for a drop in steady-state temperature. We find that reducing near-sensor power consumption from 1 W to 100 mW results in a temperature drop of 5°C. Finally, a higher ambient temperature leads to raised steady state temperatures.

**Transient temperature:** Thermal dynamic time constants govern the transient temperature of the stacked image sensor. As chip package capacitance is several orders of magnitude greater than die capacitance, the chip package time constant dominates the time constant of the overall approach to steady-state temperature, taking 10s of seconds to reach a steady state temperature. This allows dynamic temperature management policies ample time to form decisions, e.g., altering steady state temperature by changing near-sensor power draw.

Notably, near-sensor power consumption raises the transient temperature of the sensor die above the package temperature. This is because the heat source is on the sensor die itself, dissipating heat through the package into the ambient environment. Consequently, reducing power consumption rapidly reduces the gap between sensor die transient temperature and package temperature, as shown in Fig. 3.2. The speed of this drop is governed by the sensor junction die time constant, which is on the order of milliseconds. Because transient temperature affects image fidelity, these rapid temperature drops – such as the charted 13°C drop in 20 ms – provide unique opportunities for dynamic thermal management for on-demand image sensor fidelity. We discuss this in more detail in §4.

### *3.0.3 Image fidelity implications of sensor temperature*

While raised temperatures cause reliability and packaging issues for integrated circuits, they introduce another problem for image sensors: noise. The influence of noise on vision tasks has been widely reported. Recent work Dodge and Karam (2016) find that neural networks have difficulty predicting semantics of an image when challenged by different types of image noise. Similar findings from another work Amir *et al.* (2018): image classification accuracy generally degrades with increase in temperature/noise. Thus, reliable vision demands images of reasonable fidelity.

Images for human consumption further raise the fidelity bar for imaging needs; high fidelity is often needed in many real-life scenarios. For example, if a set of dashcam images is to be used in an auto insurance claim, the images need to have superior quality to obtain maximal information to make decisions on benefits. While denoising can help mitigate fidelity issues, denoising algorithms often create imaging artifacts which can also impair perceived image quality. Thus, as images are required

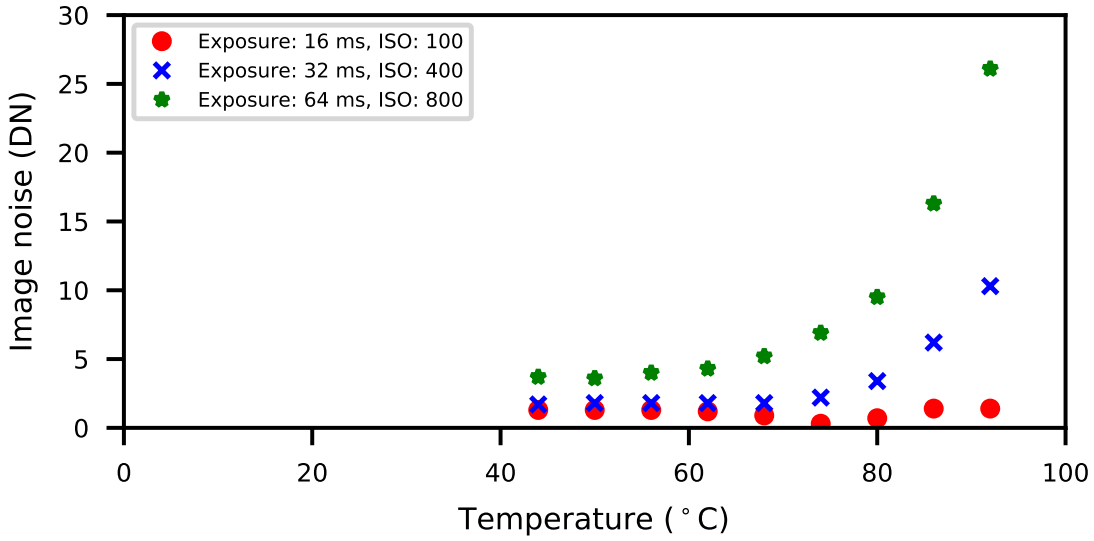



Figure 3.3: Variance of Image Noise, Expressed in Pixel Intensities, Showing Sensitivity to Temperature, Exposure, and ISO.

to fiducially represent the real physical world, imaging fidelity needs are even more stringent than vision-based needs.

The sources of image noise are theoretically well-understood (§2). However, to understand the practical relationship between temperature and image quality on commercial sensors, we perform thermal characterization on a 3 Mp OnSemi AR0330 sensor OnSemi (2016) connected to a Microsemi SmartFusion2 FPGA Microsemi (2016). The AR0330 sensor includes noise correction stages inside the sensor, as is common in commercial sensors. We use a heat gun to raise the sensor temperature and capture raw images in a dark room setting while monitoring sensor temperature with a FLIR One thermal camera.


(a) Image at 44°C (b) Image at 92°C (c) Image histograms

Figure 3.4: Two Images Captured at Different Temperatures and Their Histograms. Hotter Image Is Brighter and Grainier, Due to Influence of Thermal Noise. This Is Also Reflected in the Shift in Mean and Variance Width in the Histogram.

**Noise is more prevalent at high temperatures**

Fig. 3.3 charts a trend : sensors are particularly susceptible to noise above a particular temperature value. This is despite the presence of noise correction stages inside the sensor. The correction blocks could bring the noise under control but only for lower temperature settings. For high temperatures, the denoising appears to fail to exercise control on noise minimization. Notably, this knee shifts with exposure and analog gain settings, presumably due to noise amplification. For instance, at high exposure and high analog gain, which correspond to low light situations, sensors start to become thermally sensitive even at low temperatures, e.g., 52°C. To adapt to all experienced conditions, the sensor's thermal management should be adaptive to the varying needs of different lighting conditions.

**Noise visibly and substantially impairs quality**

Thermal noise is visibly apparent on images, whether in low light or bright light conditions. For example, Fig. 3.4 shows the images captured under daylight conditions at different sensor temperatures. We can observe the graininess in the hotter image

due to the strong influence of noise. Paired with the noisy images, the histograms represent the pixel intensity distribution of an image.The wider peaks in the distribution signify the variance of pixel intensity, while the mean of the peaks represent average pixel intensity. We can observe that the histogram of the hotter image shifts to the right, increasing pixel intensity due to dark current. We also observe that the variance of the pixel intensity increases, due to increased thermal noise.

### *3.0.4 Motivational observations*

To summarize, we have the following insights for NSP.

- Near-sensor processing architectures promote system energy-efficiency, but also increase sensor temperature
- Raised sensor temperatures aggravate thermal noise
- Transient junction temperatures crucially determine fidelity
- Smaller (ms) junction time constants facilitate immediate drop in temperature allowing on-demand high fidelity
- Fidelity needs are highly dynamic, depending on environmental factors such as lighting and ambient temperature
- Imaging demands more fidelity than vision

These observations motivate the need for novel dynamic thermal management strategies for near-sensor processing at sufficient vision and imaging fidelity.

# Chapter 4

# THERMAL MANAGEMENT MECHANISMS

Our characterization shows that near-sensor processing increases system energy efficiency, but sacrifices image fidelity due to raised sensor temperatures. This raises a natural question: *Can we leverage near-sensor processing to create efficiency benefits while maintaining sufficient image fidelity for our vision and imaging tasks?* Driven by this, we develop novel mechanisms that can efficiently regulate sensor temperature for continuous and on-demand image fidelity needs. In our design, these mechanisms are governed by a runtime controller, which we call *Stagioni*.

Dynamic temperature management for microprocessors is a mature research area, as we summarize in §2. However, traditional processor DTM mechanisms are not designed to suit imaging needs. Rather than simply being limited by TDP, sensor fidelity is impaired by the immediate transient sensor temperature while capturing. Furthermore, thermal management for near-sensor processing should adapt to the situational needs of the vision/imaging application, e.g., allowing higher temperatures when in brighter environments and rapidly dropping temperature when high fidelity is required.

To account for near-sensor processing temperature management, we modify traditional DTM techniques to introduce two potential mechanisms that quell image quality concerns, while striving to optimize for system power and performance. (1) Stop-capture-go: Temporarily *halt* near-sensor processing for temperature regulation and on-demand high fidelity capture. (2) Seasonal migration: Occasionally *migrate* the processing to a thermally isolated far-sensor VPU for temperature regulation and on-demand high fidelity captures.

### *4.0.1 Principles for managing sensor temperature*

To design thermal management mechanisms that are effective for near-sensor processing, we introduce three core principles: (1) Situational temperature regulation: The mechanism should confine sensor temperature within a threshold that suffices for imaging fidelity needs. (2) On-demand temperature drop: Upon application request, the mechanism should quickly drop the temperatures to desired capture temperature for high fidelity imaging. (3) Duty cycle governs system efficiency. Here, we discuss these in more detail.

#### Situational temperature regulation

As we discuss in §3, vision tasks have varying fidelity needs, which are sensitive to camera settings, e.g., ISO and exposure, and lighting situation, e.g., bright conditions. This translates directly to temperature requirements, resulting in a simple upper bound:

$$T_{sensor} < T_{vision} \tag{4.1}$$

Thus, temperature management must be cognizant and respectful of immediate vision task requirements in situational conditions to provision for effective vision accuracy.

#### On-demand fidelity

While vision processing can operate on low fidelity images, certain applications may require high fidelity images on demand, e.g., life logging capture after object detection. Such capture must be immediate, before the object leaves the view of the camera. Fortunately, as we characterized, sensor temperature rapidly drops with the removal of near-sensor power, i.e., by entering $CAP$ mode. For example, when the sensor drops its near-sensor power consumption from 2.5 W to 100 mW, the sensor drops

in temperature by 13.2°C. We define the time it takes the temperature to reduce by 98% of the drop as $time_{jump} = 4 \times RC_{die}$. In our simulation, this amounts to 20 ms. Temperature management can leverage this drop to provision for on-demand high fidelity.

The temperature drop is directly proportional to the disparity between the near-sensor power before and after power reduction: $T_{jump} = \alpha(P_{NSP} - P_{CAP})$. We experimentally find that for our modeled sensor, every 1 W causes a 5.5°C temperature jump, i.e., $\alpha = 5.5\frac{^{\circ}\text{C}}{\text{W}}$. When constrained by a latency deadline, e.g., to immediately capture a moving object or to meet a synchronization deadline, the achievable jump within the latency deadline is a fraction of the time it takes to drop: $T_{jump}^{latency} = T_{jump} \times (e^{-t_{latency}/RC_{die}})$ Thus, to provision for predicted fidelity needs and latency needs of an application, the temperature management mechanism can set reduced bounds :

$$T_{sensor} < T_{imaging} + T_{jump}^{latency} \tag{4.2}$$

**System Power Minimization through Duty Cycle**

While removal of processing power from sensor can effectively regulate temperature and provide on-demand high fidelity captures, the scheduling of operation should also strive to optimize for average system power. We can characterize this through the duty cycle and frequency of switches between the $NSP$ and $CAP$ modes. For duty cycle $d$, switching frequency $f_{switch}$ and energy per switch $E_{switch}$, average system power can be modeled as:

$$P_{avg} = d \times P_{NSP}^{system} + (1 - d) \times P_{CAP}^{system} + f_{switch} \times E_{switch} \tag{4.3}$$

In minimizing average power, there is a notable tradeoff between the duty cycle and the frequency of switches. Spending more time in $CAP$ mode allows the sensor

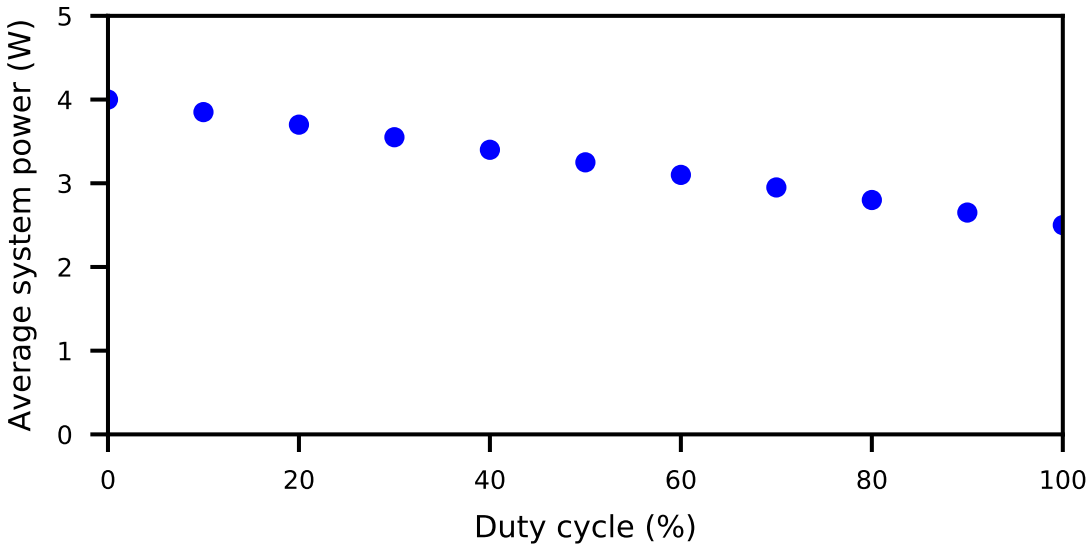



Figure 4.1: Larger Duty Cycles Maximize *NSP* Mode Operation, Thereby Optimizing System Power.

to cool down, increasing the length of time spent in $NSP$ mode as well. This reduces the number of switches. On the other hand, spending less time in $CAP$ mode allows the sensor to spend a greater proportion of time in $NSP$ mode, promoting energy savings through the duty cycle, at the expense of number of switches. Notably, the time spent in each mode must be a multiple of time spent capturing an image. It is not possible to switch to $CAP$ mode for a partial frame duration while an image is being captured. As shown in Fig. 4.1, for our implementation, which has minimal switching overhead, higher duty-cycles tend to provide favorable average system power profiles.

### *4.0.2 Stop-capture-go for near-sensor processing*

The traditional stop-go DTM technique regulates processor temperature by halting execution through clock gating. For near-sensor processing, we can similarly put the sensor in $CAP$ mode, gating near-sensor units for some time before resuming *NSP*

mode. The resulting "temporal slack" allows the sensor to regulate capture fidelity at the expense of task performance. Stop-go techniques are architecturally simple, requiring only the ability to gate the clock or power of various components.

Unlike traditional stop-go, our proposed stop-capture-go requires unique modifications to be sensitive to near-sensor processing tasks. First, frequently clock gating the entire sensor is not advisable; interruptions to the camera pipeline create substantial capture delays on the order of multiples of frames. Instead, the system will clock gate the near-sensor VPU and DRAM, putting the sensor into CAP mode. Second, rather than being governed by TDP, the temperature regulation will trigger as the sensor reaches a situational upper bound specified by the principles, such that $T_{sensor} < T_{vision}$ and $T_{sensor} < T_{imaging} + T_{jump}^{latency}$. Third, the execution halt can be triggered by the controller to achieve on-demand fidelity upon application request. For this, the sensor simply enters CAP mode to retrieve the requested frame.

**Parameterization of stop time**

The amount of "stop" time – the amount of time the processor is halted – is an important policy parameter. During the stop time, the system will "drop" frames, failing to process them. Elongated stop times allow a sensor to cool down further, which reduces the number of switches. For vision tasks, stop times can be detrimental, as contiguously dropped frames may contain important ephemeral visual information. Thus, if a system wishes to prioritize a continuity of visual information, stop time should be reduced. In our simulated study, we find that the minimal stop time of 33 ms (one frame time) is sufficient to cool down the sensor from 87 to 74°C, enabling sufficient continuous temperature regulation and on-demand fidelity.

**Usability of stop-capture-go**

Due to the architectural simplicity of stop-capture-go, the system overhead is minimal, promoting continuously low system power. However, frequent frame drops will impair the visual task performance. Thus, stop-capture-go is suitable for systems that demand low power but are not performance-critical and/or systems that require minimal architecture modifications.

### *4.0.3 Seasonal migration*

While stop-capture-go is a simple policy for temperature regulation and high-fidelity captures, it degrades application performance by halting execution. Towards minimizing performance loss, we investigate seasonal migration for near-sensor processing. Seasonal migration shifts the processing to a thermally isolated computational unit, allowing continuous computing. As we model in §3, spatial thermal isolation between the sensor and SoC allows thermal relief. Enabling seasonal migration comes at the expense of duplicated computational units near to and far from the sensor, but effectively regulates sensor temperature without sacrificing task performance.

As illustrated in Fig. 4.2, the process for seasonal migration is governed by two temperature limits: $T_{high}$ and $T_{low}$. In efficiency phase, triggered when the sensor reaches a temperature below $T_{low}$, it will enter NSP mode, performing near-sensor processing for system efficiency. In cooling phase, triggered when the sensor reaches a temperature above $T_{high}$, it will enter CAP mode, performing off-sensor processing on the SoC, allowing the sensor to cool down. The alternation between these phases allows the system to balance efficiency with sensor temperature. For on-demand

fidelity, the system simply enters the cooling phase regardless of current sensor temperature.

**Parameterization of temp. bounds**

$T_{high}$ and $T_{low}$ are important policy parameters, controlling the balance of efficiency and temperature. $T_{high}$ forces the sensor temperature regulation, and thus should be set to shift to situational needs:

$$T_{high} = min(T_{vision}, T_{imaging} + T_{jump}^{latency})$$

Meanwhile, the gap between $T_{high}$ and $T_{low}$ controls the system efficiency implications of the policy. Because it takes more time for the sensor temperature to bridge a larger gap, larger gaps decrease the frequency of switches, while smaller gaps increase the frequency of switches. The $T_{high} - T_{low}$ gap also controls the duty cycle of the system. When the desired sensor temperature range is closer to steady-state NSP temperature than steady-state CAP temperature, smaller gaps produce favorable duty cycles, spending more time in NSP mode. As shown in Eqn. 4.3, the average system power is a function of this duty cycle, balanced against the energy overhead and frequency of switches. Thus, $T_{low}$ should be chosen to create a gap that optimizes average system power.

As we defined earlier, the duty cycle is the proportion of time spent in $NSP$ mode. For seasonal migration, the relationships can be derived from standard charging models. After the rapid drop or rise in temperature $T_{jump}$, which takes approximately $time_{jump}$ amount of time, the sensor follows an RC charging curve towards the steady state temperature of the $NSP$ or $CAP$ mode. Altogether, this can be used to analytically model duty cycle $d$ and frequency of migration $f_{migration}$.

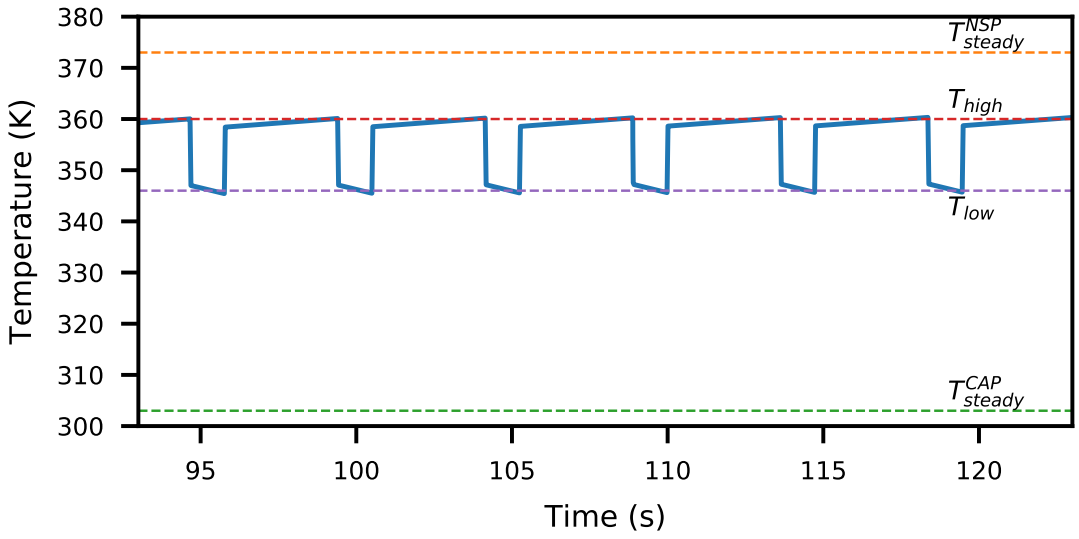



Figure 4.2: Transient Response of Seasonal Migration Mechanism with 77% Duty Cycle to Confine Sensor Temperature Within Thermal Boundaries ($T_{high}$ And $T_{low}$).

$$time_{warming} = RC \times \ln\left(\frac{T^{NSP}_{steady} - (T_{low} + T_{jump})}{T^{NSP}_{steady} - T_{high}}\right) + time_{jump}$$

$$time_{cooling} = RC \times \ln\left(\frac{(T_{high} - T_{jump}) - T^{CAP}_{steady}}{T_{low} - T^{CAP}_{steady} - T_{jump}}\right) + time_{jump}$$

$$d = time_{warming}/(time_{warming} + time_{cooling})$$

$$f_{migration} = 2/(time_{warming} + time_{cooling})$$

**Usability of seasonal migration**

Depending on implementation, seasonal migration could suffer from the switching latency and energy overhead resulting from state transfer and synchronization in shifting processing from one computational unit to another. However, reducing this migration overhead is a well-studied problem in distributed systems Milojičić *et al.* (2000). Several reported techniques mitigate migration latency, e.g., pre-copy-based

migration Richmond and Hitchens (1997), which promote smooth execution performance while incurring energy overhead by keeping both computational units on while preparing for migration. Similarly, in our implementation, prior to migration, we prepare the system by pre-emptively starting up the target computational unit and initiating its context so it is prepared for execution.

### *4.0.4 Stagioni Runtime Controller*

We propose the Stagioni Runtime Controller to execute the mechanisms at runtime. Stagioni's responsibility is to guarantee the fidelity demands of the application, coordinating state transfer between the operating modes to ensure smooth transition. Stagioni could be designed in a multitude of ways, e.g., a dynamically linked library, a runtime OS service, or dedicated hardware. In our implementation and evaluation, Stagioni is a runtime OS service that sits on the near-sensor processor, allowing the SoC to sleep. (In our implementation, the near-sensor processor also hosts the application context.) Many existing migration controller designs would sufficiently and equivalently serve the purposes of decision-making. Here we describe one set of modules that would achieve the goals. We discuss different aspects, including how Stagioni receives application inputs to meet fidelity demands.

**API for application-specific fidelity needs:** A vision application only needs to provide three pieces of information to the controller: (1) continuous image fidelity requirement for vision (2) on-demand image fidelity requirement for Imaging (3) when to trigger on-demand fidelity. A simple API can enable developers to specify requirements from their applications. A class with the following methods would suffice:

- `setVisionSNR(float)`: specify continuous fidelity
- `setImagingSNR(float)`: specify on-demand fidelity

- `triggerOnDemandFidelity()`: request high fidelity

Stagioni translates expectations into effective thermal management, sidestepping any form of developer burden. To do this, the controller applies application-specific requirements into appropriate policy parameters through characterized device models. Stagioni also continuously adapts the policy parameters to situational settings, i.e., ambient temperature and ambient lighting, to meet ongoing quality requirements.

Stagioni orchestrates the execution pattern in runtime, which consists of several system-level events. For stop-capture-go, Stagioni would use simple power gating mechanisms such as clock gating. For seasonal migration, Stagioni would handle the communication between two chips.

To this end, Stagioni can use simple message passing schemes to synchronize states between the sensor and the host. One such scheme, implemented in our evaluation, could operate as follows: (*i*) The sensor temperature monitor detects a thermal trigger and raises an interrupt. (*ii*) Stagioni sends a signal to the SoC controller to prepare for migration. (*iii*) In return, the SoC controller starts the application and sends an acknowledgement to the source conveying that it is ready to accept the tasks. (*iv*) Stagioni then transfers application context data transfer from source's memory to the host's memory. (*v*) Once the data transfer is done, both migration handlers notify their corresponding applications. The offloaded tasks can now run in the new context loading the state from the memory. This sequence of steps can be scheduled prior to the migration event, such that immediate migration is possible.

# Chapter 5

# IMPLEMENTATION

Since there are no readily available off-the-shelf readily programmable 3D stacked image sensors, we use a combination of simulation and emulation techniques to implement and study Stagioni's mechanisms. We build our simulation framework around our characterized energy, noise, and thermal models. The simulation tool operates on these models and reports system metrics such as average system power, performance for different policy schedules. To practically realize the policies, we build an emulation platform around FPGA. We design and implement Stagioni as a runtime controller and integrate it into the system to study execution patterns of different policies.

### *5.0.1 Simulation framework*

We build our simulation framework as a tool. Our tool can be used to evaluate the thermal, energy, and noise of a given 3D stacked sensor-based systems on our proposed policies across a range of workloads. The tool takes device models and policy details as inputs and provides different system metrics as outputs while running sensor-driven applications. The users may wish to override default characterization models to suit their needs. In this case, users can provide vision task, camera settings, thermal policies to apply, and the desired capture temperature for images. The tool solves for the policy parameters such as rise and fall times that govern the mechanisms. Finally, the tool generates the temperature and fidelity traces and also reports the power and performance of the system. We plan to make the tool open-source at publication time.

### *5.0.2 Emulation framework*

In addition to the simulation tool, we build an FPGA-based emulation platform on two ZCU102 boards. One of them emulates the sensor, while the other emulates the SoC. We use 1 Gbps Ethernet for communication, simulating a standard CSI interface that has similar bandwidth characteristics.

We design Stagioni around the CHaiDNN library Xilinx (2018a). The Stagioni controller takes the type of policy and its associated parameters as inputs. The parameters then generate a mode schedule that governs the task execution in runtime. The controller also handles high fidelity capture requests and services them to deliver high quality images through appropriate mechanisms. For stop-capture-go, we gate the execution of the neural network invocation. For seasonal migration, we perform message passing over Ethernet for state transfer and implement producer-consumer queues for synchronization.

Chapter 6

# EVALUATION

We investigate the effectiveness of our proposed policies in meeting the fidelity demands, while performing vision tasks of our case study. We find that our policies can deliver up to 36% savings in average system power compared to traditional far-sensor processing, for our case study. The savings primarily stem from maximizing near-sensor task operation. Furthermore, we find that the savings varies with the fidelity requirements of the application. The policies achieve the savings by incurring an latency overhead of only 100 µs, which is negligible in comparison to ms-scale image capture times.

### *6.0.1 Evaluation workloads*

To evaluate different system metrics, we use the simulation and emulation frameworks introduced in §5.

**Vision tasks:** For our vision task, we study image classification, identifying objects in a scene. We evaluate our policies on the GoogLeNet ConvNet Szegedy *et al.* (2015), modified to use 16-bit quantized weights for efficiency. We also evaluate our policies on other vision tasks, such as YOLO-based object detection Redmon and Farhadi (2017) with identical findings, omitted for brevity.

**Metrics and policies:** The major objective for evaluating a policy is: effectiveness in regulating sensor temperature for capture fidelity, while optimizing system's power with minimal performance overhead. We use SNR to gauge image quality and frame drops for performance overhead. In addition to stop-capture-go and seasonal

migration, we consider full-far sensor processing (the status quo) for policy comparison.

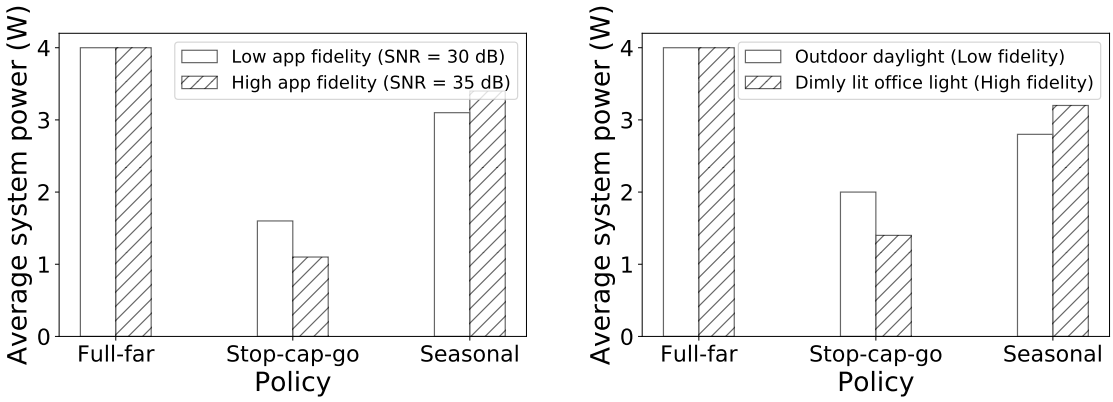
**Environment conditions:** We evaluate for a wide range of lighting conditions from bright outdoor to dark indoor environments. Such lighting translates into different camera settings, i.e., exposure and ISO. We use the flexible CapraRawCamera Su (2018) camera app to automatically determine appropriate camera settings based on the scene lighting. We notice and use the following camera settings for three sensor illuminations.

- Outdoor daylight (32000 lux): Exp. = 16 ms, ISO = 100
- Indoor office light (320 lux): Exp. = 32 ms, ISO = 400
- Dimly lit office light (3.2 lux): Exp. = 64 ms, ISO = 800

For evaluating ambient temperature effects, we use a 20 °C to 40 °C range, representing cool indoor to hot outdoor situations.

### *6.0.2 Power*

We find that stop-capture-go and seasonal migration substantially reduce system power compared to status quo. Average system power largely depends on duty cycle. Naturally, we can get maximal power savings by operating at a maximum duty cycle. However, the achievable duty cycle is limited by the placement of thermal boundaries. The thermal boundary placement determines the steepness or gradualness of warming and cooling phases. Thermal boundaries closer to the steady-state temperature of the warming phase results in higher duty cycles. The fidelity requirements, dictated by application and ambient situation, decide the placement of thermal boundaries. High fidelity expectations result in lower thermal boundaries, and therefore, lower duty




(a) Variation With App Fidelity.

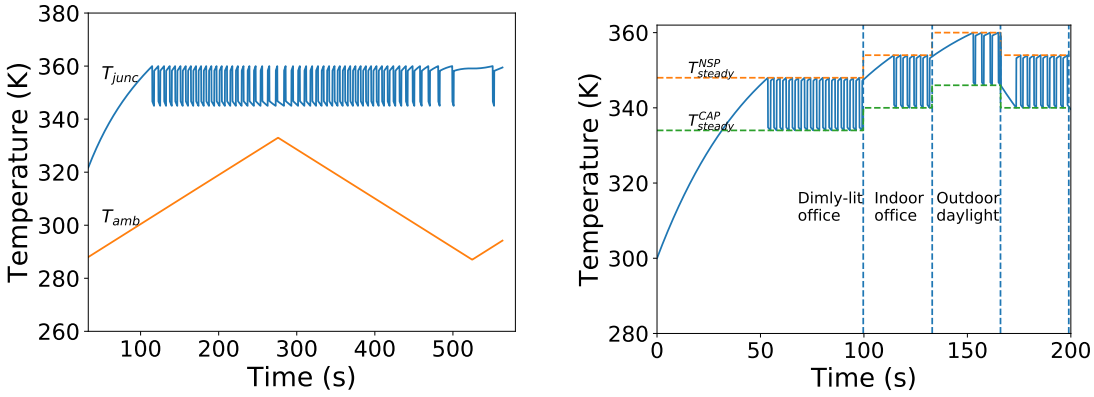
(b) Variation With Lighting.

Figure 6.1: Average System Power Varies with Fidelity Needs. For Stop-capture-go, Lower Duty Cycle Decreases the System Power Due to More Vpu Sleep Time. In Contrast, for Seasonal Migration, Lower Duty Cycle Increases System Power Due to More Far-sensor Operation. For Full-far Policy, There Is No Change in Power as It Does Not Create Any Fidelity Issues.

cycles. Here we evaluate the implication of different fidelity requirements on system power.

**Variation with app fidelity needs:** Fig. 6.1a shows the system power for different policies for different application fidelity needs. We see that stop-capture-go consumes the lowest amount of power among all the policies. This is because stop-capture-go operates entirely on near-sensor VPU for whole program execution in both *NSP* and *CAP* modes. In contrast, seasonal migration operates on far-sensor VPU during *CAP* mode and on near-sensor VPU during *NSP* mode. So, it consumes more power than stop-capture-go but less than full-far policy, which always operates on far-sensor VPU.

System power changes with fidelity demands, due to change in duty cycle; high fidelity pulls down the duty cycle, reducing efficiency. This is evident in our seasonal




(a) Adaptive To Temperature.

(b) Adaptive To Lighting.

Figure 6.2: Increasing Ambient Temperature (Left) And/Or Decreasing Ambient Illumination (Right) Pulls $T^{NSP}_{steady}$ Away From $T_{low}$ And Pushes $T^{CAP}_{steady}$ Close To $T_{high}$. Stagioni Shifts Thermal Boundaries to Smoothly Adapt to Different Ambient Conditions.

migration simulations; we see higher power for high app fidelity in comparison to the power with low app fidelity. Meanwhile, for stop-capture-go, a lower duty cycle increases VPU sleep time. Therefore, we see power *decrease* as we go from low to high app fidelity. Finally, for full-far policy, there is no change in system power as it doesn't create fidelity issues.

**Variation with situational fidelity needs:** We see similar trends for various policies with fidelity changes forced by lighting, as illustrated in Fig. 6.1b. Here outdoor daylight behaves similarly to a low fidelity case, while indoor dimly lit office light behave similar to a high fidelity case.

### *6.0.3 Overhead*

We discuss the policy execution overhead for seasonal migration and stop-capture-go policies. While the system executes seasonal migration, it switches between near-

Table 6.1: This Table Shows the Rough Estimates of Near-sensor System Power Profiles of Different VPUs and Savings Compared to Status Quo. For Power Profiles Without Temperature Issues, We Can Perform Near-sensor Processing for Entire Program Execution, I.E., 100% Duty Cycle, to Achieve Maximum System Savings. For the Rest, Stagioni Optimizes Duty Cycle Based on Fidelity Needs.

| **Architecture** | **Vision ConvNet** | **Frame Rate(fps)** | **Comp. Power (W)** | **Trad. Sys. Power (W)** | **NSP Sys. Power (W)** | **Duty Cycle (%)** | **Avg. Sys. Power (W)** | **Savings (%)** |
|---|---|---|---|---|---|---|---|---|
| Eyeriss + EIE Chen *et al.* (2016) | AlexNet | 35 | 0.9 | 2.71 | 1.0 | 100 | 1 | 63 |
| Myriad 2 Pena *et al.* (2017) | GoogLeNet | 3 | 1.3 | 1.45 | 1.31 | 77 | 1.34 | 8 |
| Neurostream Azarkhish *et al.* (2018) | ResNet50 | 34 | 2.5 | 4.25 | 2.59 | 66 | 3.16 | 26 |
| NeuFlow Pham *et al.* (2012) | N/A | 30 | 6 | 7.55 | 6.08 | 55 | 6.74 | 11 |
| TK1 Cavigelli *et al.* (2015) | N/A | 10 | 6.6 | 7.12 | 6.63 | 80 | 6.72 | 5 |

sensor and far-sensor VPUs, incurring an overhead. Switching overhead strongly relates to the number of frame drops. From our emulation setup, we find that the switching overhead is 100 $\mu$s, which is much less than frame capture/inference times (33 ms). Therefore, seasonal migration has negligible overhead, therefore, no impact on application performance.

For stop-capture-go, stop time determines the number of frame drops. At the same time, lower stop times also promote higher efficiency through higher duty cycles.Furthermore, the sufficient temperature drop can be achieved in less than a frame period. Therefore, we can operate at the minimum stop time (one frame time) for efficiency reasons.

### *6.0.4 Situational awareness*

One feature of Stagioni that differs from traditional DTM techniques is situational awareness to dynamic ambient settings. We find that Stagioni smoothly adapts thermal boundaries to match ambient temperature and lighting situations.

**Ambient temp. awareness:** Ambient temperature determines steady-state temperatures, which determine the warming and cooling times. Higher ambient temperatures push $T^{NSP}_{steady}$ far from $T_{low}$ and push $T^{CAP}_{steady}$ close to $T_{high}$. This forces the warming phase to take a steeper rise and the cooling phase to take a gradual fall in the exponential curve. Thus, increasing ambient temperature decreases duty cycle and vice-versa. We simulate the change in ambient temperature in our emulation platform, shown in Fig. 6.2a. Decreasing ambient temperature increases rise times and reduces fall times in the simulated temperature trace. We also notice that Stagioni smoothly adjusts to the changes in ambient temperature.

**Ambient light awareness:** Lighting dictates fidelity requirements, changing $T_{high}$ and $T_{low}$. Again, Stagioni adapts to these changes with light variation. We simulate change in illumination to generate a trace with random juggling between lighting scenarios. We provide this trace as input to our runtime and collect the temperature trace. Fig. 6.2b shows the temperature trace overlaid with $T_{high}$ and $T_{low}$. We can clearly see the smooth variation of temperature with light intensity.

### *6.0.5 Choice of VPU*

Table 6.1 lists the power profile of several VPU choices. VPU power profile determines the extent to which the system can leverage near-sensor processing. For the low power profiles that do not degrade fidelity, e.g., Eyeriss + EIE, we can fully execute tasks on near-sensor, i.e., at 100% duty cycle. For VPUs that cause fidelity issues, e.g., Neurostream, Stagioni enables near-sensor processing to leverage energy-efficiency benefits, determining duty cycles to maximize power savings.

# Chapter 7

# FUTURE WORK

Stagioni is an early exploration of thermal management for near-sensor processing. We envision a collection of significant extensions to unlock the benefits of 3D stacked integration.

**Fine-grained temperature management:** Our seasonal migration policy executes at coarse granularity, migrating the entire workload between near- and far-sensor VPUs. Migration at a fine granularity, e.g., OpenVX task graph nodes Itseez (2017), can help achieve fine-grained task migration towards precise temperature management and associated optimization.

**Enhancing near-sensor burst performance:** Temperature management for near sensor processing unlocks the ability to leverage near-sensor VPUs for efficiency, but could also provide burst performance benefits under a temperature "budget". Adapting temperature management for burst performance would need a deeper semantic awareness of application workload requirements. For non-trivial workloads, this may require reactive programming or other sophisticated techniques to, for example, reconfigure on-demand sensor operations and expectations when the visual task requires it.

**Stacked sensor architecture design/validation:** While we model and simulate implications of stacked sensor architectures, our future efforts will design stacked hardware to validate our claims. We plan to implement tunable components and interfaces and evaluate with different scenarios.

# Chapter 8

# CONCLUSION

Near-sensor processing has a great potential towards energy-efficient imaging and vision, as demonstrated by recent academic and industrial efforts on stacked image sensors. However, we show that by doing so hampers sensor fidelity due to thermal noise, thereby limiting the adoption of near-sensor processing. Our characterization reveals that immediate drop in temperatures can be realized within short duration. We use this observation to design principles for managing sensor temperature for efficient temperature regulation and high fidelity temperatures, while optimizing for system power. To implement the policies, we design and implement the Stagioni runtime to manage sensor temperature, while fulfilling imaging needs. Our work is the first runtime solution for stacked sensor thermal management. We foresee our work forming the foundation for future imaging-aware DTM techniques.

REFERENCES


3D-IC-blog, "History of 3d stacked image sensors", `http://www.3dic.org/3D_stacked_image_sensor` (2016).

Abadi, M., P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: a system for large-scale machine learning.", in "OSDI", (2016).

Amir, M. F., J. H. Ko, T. Na, D. Kim and S. Mukhopadhyay, "3-d stacked image sensor with deep neural network computation", IEEE Sensors Journal (2018).

Azarkhish, E., D. Rossi, I. Loi and L. Benini, "Neurostream: Scalable and energy efficient deep learning with smart memory cubes", IEEE Tran. on Parallel & Distributed Systems (2018).

Borkar, S., "Design challenges of technology scaling", IEEE micro (1999).

Brooks, D. and M. Martonosi, "Dynamic thermal management for high-performance microprocessors", in "The Seventh IEEE Int. Symp. on High-Performance Computer Architecture, HPCA", (2001).

BVLC, "Caffe performance measurements on nvidia gpus", `http://tutorial.caffe.berkeleyvision.org/performance_hardware.html` (2018).

Cavigelli, L., M. Magno and L. Benini, "Accelerating real-time embedded scene labeling with convolutional networks", in "Proc. of the ACM 52nd Annual Design Automation Conference", (2015).

Chen, Y.-H., J. Emer and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks", in "ACM SIGARCH Computer Architecture News", (2016).

Choi, J., S. Park, J. Cho and E. Yoon, "An energy/illumination-adaptive cmos image sensor with reconfigurable modes of operations", IEEE Journal of Solid-State Circuits (2015).

Dodge, S. and L. Karam, "Understanding how image quality affects deep neural networks", in "IEEE Int. Conf. on Quality of Multimedia Experience (QoMEX)", (2016).

Du, Z., R. Fasthuber, T. Chen, P. Ienne, L. Li, T. Luo, X. Feng, Y. Chen and O. Temam, "Shidiannao: Shifting vision processing closer to the sensor", in "ACM SIGARCH Computer Architecture News", (2015).

Forchheimer, R. and A. Astrom, "Near-sensor image processing: A new paradigm", IEEE Transactions on Image Processing (1994).

Gomaa, M., M. D. Powell and T. Vijaykumar, "Heat-and-run: leveraging smt and cmp to manage power density through the operating system", in "ACM SIGARCH Computer Architecture News", (2004).

Han, S., X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz and W. J. Dally, "Eie: efficient inference engine on compressed deep neural network", in "ACM/IEEE 43rd Annual Int. Symp. on Computer Architecture (ISCA)", (2016).

Haruta, T., T. Nakajima, J. Hashizume, T. Umebayashi, H. Takahashi, K. Taniguchi, M. Kuroda, H. Sumihiro, K. Enoki, T. Yamasaki *et al.*, "4.6 a 1/2.3 inch 20mpixel 3-layer stacked cmos image sensor with dram", in "IEEE Int. Solid-State Circuits Conference (ISSCC)", (2017).

Itseez, "Openvx computer vision acceleration standard", `https://www.khronos.org/openvx/` (2017).

Jia, Y., E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding", in "Proc. of the 22nd ACM Int. Conf. on Multimedia", (2014).

Kumagai, O., A. Niwa, K. Hanzawa, H. Kato, S. Futami, T. Ohyama, T. Imoto, M. Nakamizo, H. Murakami, T. Nishino *et al.*, "A 1/4-inch 3.9 mpixel low-power event-driven back-illuminated stacked cmos image sensor", in "IEEE Int. Solid-State Circuits Conference-(ISSCC)", (2018).

Lee, K.-W., Y. Ohara, K. Kiyoyama, S. Konno, Y. Sato, S. Watanabe, A. Yabata, T. Kamada, J. Bea, H. Hashimoto *et al.*, "Characterization of chip-level hetero-integration technology for high-speed, highly parallel 3d-stacked image processing system", in "IEEE Electron Devices Meeting (IEDM)", (2012).

Levoy, M., "Noise and iso", `https://graphics.stanford.edu/courses/cs178/lectures/noise-29apr14.pdf` (2014).

Lie, D., K. Chae and S. Mukhopadhyay, "Analysis of the performance, power, and noise characteristics of a cmos image sensor with 3-d integrated image compression unit", IEEE Tran. on Components, Packaging and Manufacturing Technology (2014).

LiKamWa, R., Y. Hou, J. Gao, M. Polansky and L. Zhong, "Redeye: analog convnet image sensor architecture for continuous mobile vision", in "ACM SIGARCH Computer Architecture News", (2016).

LiKamWa, R., B. Priyantha, M. Philipose, L. Zhong and P. Bahl, "Energy characterization and optimization of image sensing toward continuous mobile vision", in "Proc. of the ACM 11th annual int. conf. on Mobile systems, applications, and services", (2013).

Lowe, D. G., "Object recognition from local scale-invariant features", in "The Proc. of the seventh IEEE Int. conference on Computer vision", (1999).

Microsemi, "Imaging and video solution", `https://www.microsemi.com/products/fpga-soc/imaging` (2016).

Milojičić, D. S., F. Douglis, Y. Paindaveine, R. Wheeler and S. Zhou, "Process migration", ACM Computing Surveys (CSUR) (2000).

Nose, A., T. Yamazaki, H. Katayama, S. Uehara, M. Kobayashi, S. Shida, M. Odahara, K. Takamiya, S. Matsumoto, L. Miyashita *et al.*, "Design and performance of a 1 ms high-speed vision chip with 3d-stacked 140 gops column-parallel pes", MDPI Sensors (Basel, Switzerland) (2018).

OnSemi, "Ar0330 image sensor datasheet", `https://www.onsemi.com/pub/Collateral/AR0330CM-D.PDF` (2016).

Pena, D., A. Forembski, X. Xu and D. Moloney, "Benchmarking of cnns for low-cost, low-power robotics applications", in "RSS 2017 Workshop: New Frontier for Deep Learning in Robotics", (2017).

Pham, P.-H., D. Jelaca, C. Farabet, B. Martini, Y. LeCun and E. Culurciello, "Neuflow: Dataflow vision processing system-on-a-chip", in "IEEE 55th Int. Midwest Symp. on Circuits and Systems (MWSCAS)", (IEEE, 2012).

Redmon, J. and A. Farhadi, "Yolo9000: better, faster, stronger", arXiv preprint (2017).

Richmond, M. and M. Hitchens, "A new process migration algorithm", ACM SIGOPS Operating Systems Review (1997).

Shi, Y. and S. Lichman, "Smart cameras: a review", in "Citeseer survey", (2005).

Simonyan, K. and A. Zisserman, "Very deep convolutional networks for large-scale image recognition", arXiv preprint arXiv:1409.1556 (2014).

Skadron, K., T. Abdelzaher and M. R. Stan, "Control-theoretic techniques and thermal-rc modeling for accurate and localized dynamic thermal management", in "Proc. Eighth IEEE Int. Symp. on High-Performance Computer Architecture", (2002).

Su, P.-H., "Caprarawcamera: Android app for collecting raw photos for computer vision", `https://github.com/cucapra/CapraRawCamera` (2018).

Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, "Going deeper with convolutions", in "Proc. of the IEEE conf. on computer vision and pattern recognition", (2015).

TechInsights, "Samsung galaxy s9 camera teardown", `http://techinsights.com/about-techinsights/overview/blog/samsung-galaxy-s9-camera-teardown` (2018).

technologies, M., "Micron system power calculators", `https://www.micron.com/support/tools-and-utilities/power-calc` (2018).

Xilinx, "Hls based deep neural network accelerator library for xilinx ultrascale+ mpsocs", `https://github.com/Xilinx/CHaiDNN` (2018a).

Xilinx, "Xilinx power estimator", `https://www.xilinx.com/products/technology/power/xpe.html` (2018b).

Yu, Y.-J. and C.-J. Wu, "Designing a temperature model to understand the thermal challenges of portable computing platforms", in "Proc. of the IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITHERM)", (2018).

Zhang, C., Z. Fang, P. Zhou, P. Pan and J. Cong, "Caffeine: towards uniformed representation and acceleration for deep convolutional neural networks", in "Proc. of the ACM 35th Int. Conf. on Computer-Aided Design", (2016).

Zhang, C., P. Li, G. Sun, Y. Guan, B. Xiao and J. Cong, "Optimizing fpga-based accelerator design for deep convolutional neural networks", in "Proc. of the 2015 ACM/SIGDA Int. Symp. on Field-Programmable Gate Arrays", (2015).