The Accelerator Wall: Limits of Chip Specialization

Adi Fuchs David Wentzlaff
Department of Electrical Engineering
Princeton University
{adif, wentzlaf}@princeton.edu

***Abstract*—Specializing chips using hardware accelerators has become the prime means to alleviate the gap between the growing computational demands and the stagnating transistor budgets caused by the slowdown of CMOS scaling. Much of the benefits of chip specialization stems from optimizing a computational problem within a given chip's transistor budget. Unfortunately, the stagnation of the number of transistors available on a chip will limit the accelerator design optimization space, leading to diminishing specialization returns, ultimately hitting an *accelerator wall*.**

In this work, we tackle the question of what are the limits of future accelerators and chip specialization? We do this by characterizing how current accelerators depend on CMOS scaling, based on a physical modeling tool that we constructed using datasheets of thousands of chips. We identify key concepts used in chip specialization, and explore case studies to understand how specialization has progressed over time in different applications and chip platforms (e.g., GPUs, FPGAs, ASICs)[1]. Utilizing these insights, we build a model which projects forward to see what future gains can and cannot be enabled from chip specialization. A quantitative analysis of specialization returns and technological boundaries is critical to help researchers understand the limits of accelerators and develop methods to surmount them.

***Keywords*-Accelerator Wall; Moore's Law; CMOS Scaling**

I. Introduction

The ubiquity of specialized chips in modern architectures is motivated by a trisection of trends: (i) Power budget limitations imposed by the nearing end of Moore's law and the demise of Dennard scaling. (ii) The growing computation demands of applications in battery-limited mobile devices and warehouse-scale datacenters. (iii) The emergence of compute-intensive applications that exhibit high reuse rates of computation-patterns such as matrix-vector multiplication in deep learning applications or cryptographic hashing in cryptocurrency miners. By sacrificing flexibility and targeting specific application domains, specialized chips deliver higher performance under the same power envelope. This trend has driven the increasing use of specialized IP blocks in mobile SoC platforms [1] and the deployment of accelerators in datacenters [2]–[7].

Chip Specialization Caveats: Knowns and Unknowns. Recent studies have explored ways to ameliorate shortcomings of specialized architectures. Frequent algorithm changes make targeted domains volatile and render specialized hardware obsolete [8], and narrowed applicability reduces the profitability of non-programmable ASIC production due to non-recurring engineering costs [9].

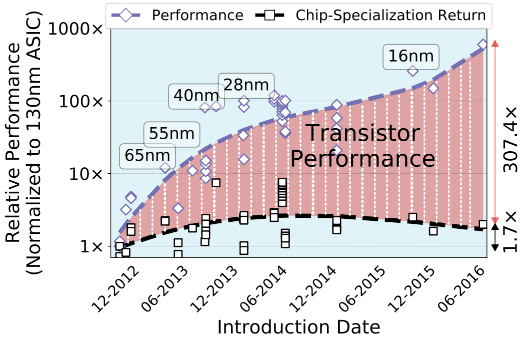


Figure 1: Evolution of Bitcoin Mining ASIC Chips. Performance metric is SHA256 Hashing Throughput per Chip Area (Hashes/Seconds/mm^2).

While the aforementioned shortcomings stem from the tradeoff between efficiency and applicability, in this work **we focus on the limitations imposed by applying chip specialization.** Specialization relies on the optimization of hardware to a computation domain, subject to a given budget of power, area, and cost. Specialization has become a prominent means to prolong the evolution of chip capabilities amid the dwindling increase in transistor budget and power limitations due to the demise of CMOS scaling [10,11].

Unfortunately, the optimization space for specializing a single application is limited; there are a finite number of ways to map computational problems to fixed chip resources, and therefore, limits to the attainable chip returns for a given domain and budget. As the nearing end of CMOS scaling will cap chip transistor budgets, optimization will eventually deliver diminishing returns, and accelerators will reach a limit of near-optimal compute and hardware co-optimization. We term this limit the *accelerator wall*.

In this work, we conduct a study on the theoretical and empirical limits of chip specialization. Understanding these limits is the first step needed for academia and industry to create novel solutions to surmount the accelerator wall. Our paper begins with developing a methodology that quantifies the benefits of chip specialization. We then build a model for a chip's physical potential using datasheets of thousands of manufactured chips. The potential model enables us to tease apart the end-to-end chip gains that come from advances in physical chip capabilities (e.g., CMOS technology advancements) versus the CMOS-independent benefits of optimizing a chip under a given budget. We name this CMOS-independent metric the **Chip Specialization Return (CSR)** of a design.

[1] Our modeling tool and evaluated application data are available at: https://github.com/PrincetonUniversity/accelerator-wall

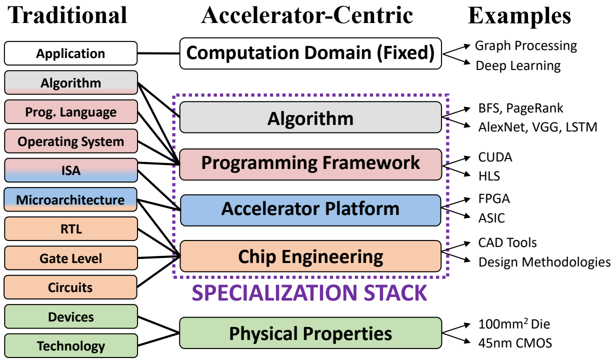


Figure 2: Abstraction Layers: Traditional and Accelerated Systems. Dashed Box Groups The Layers of Specialization.

As a motivating example of how chip specialization progresses as applications are better understood over time, we examine the performance and CSR (how good of a job the designer did to use the transistors given) for Bitcoin mining ASIC chips as a function of time in Figure 1. Transistor performance is derived from the CMOS potential of evaluated chips (e.g., transistor count, CMOS speed), and the CSR is the ratio of performance/transistor performance. While performance has improved by 510×, it was mainly due to a 307× improvement in transistor performance, i.e., better physical capabilities. Furthermore, the figure shows that CSR did not improve in the last two years.

We present multiple case studies to quantify the benefits of different specialization techniques and understand how specialization varies across different implementation platforms (e.g., GPUs, FPGAs, ASICs). We examine common concepts used for chip specialization, and their theoretical limits. Utilizing Aladdin [12] and our physical chip model, we conduct a design space exploration of specialization concepts to approximate CSR for a variety of applications. Finally, building upon the past knowledge of how CSR has progressed and how physical chip capabilities will progress until CMOS scaling ends, we build a model to project forward and explore possible future chips, quantitatively predicting the limits imposed by the accelerator wall.

Our contributions in detail include:

- We use CMOS scaling equations and datasheets from thousands of chips to construct an application-independent CMOS potential model of a chip. We use this model to decouple the contributions of CMOS technology and specialization to accelerator gains.
- We develop the metric of Chip Specialization Return (CSR) and examine popular applications and different accelerator platforms (GPU, FPGA, and ASIC) to quantify how CSR has changed in accordance to the improvement in accelerator gains.
- We identify commonly used chip specialization techniques and their theoretical limitations.
- We build a modeling framework based on Aladdin [12] and CMOS equations, to tease apart the gains from specialization techniques and CMOS scaling, on a range of accelerator benchmarks [13]–[15].
- We perform a Pareto-optimal projection study for the evaluated applications, and predict the limit of accelerator potential at the end of transistor scaling.

II. The Interplay of CMOS Scaling And Chip Specialization Return

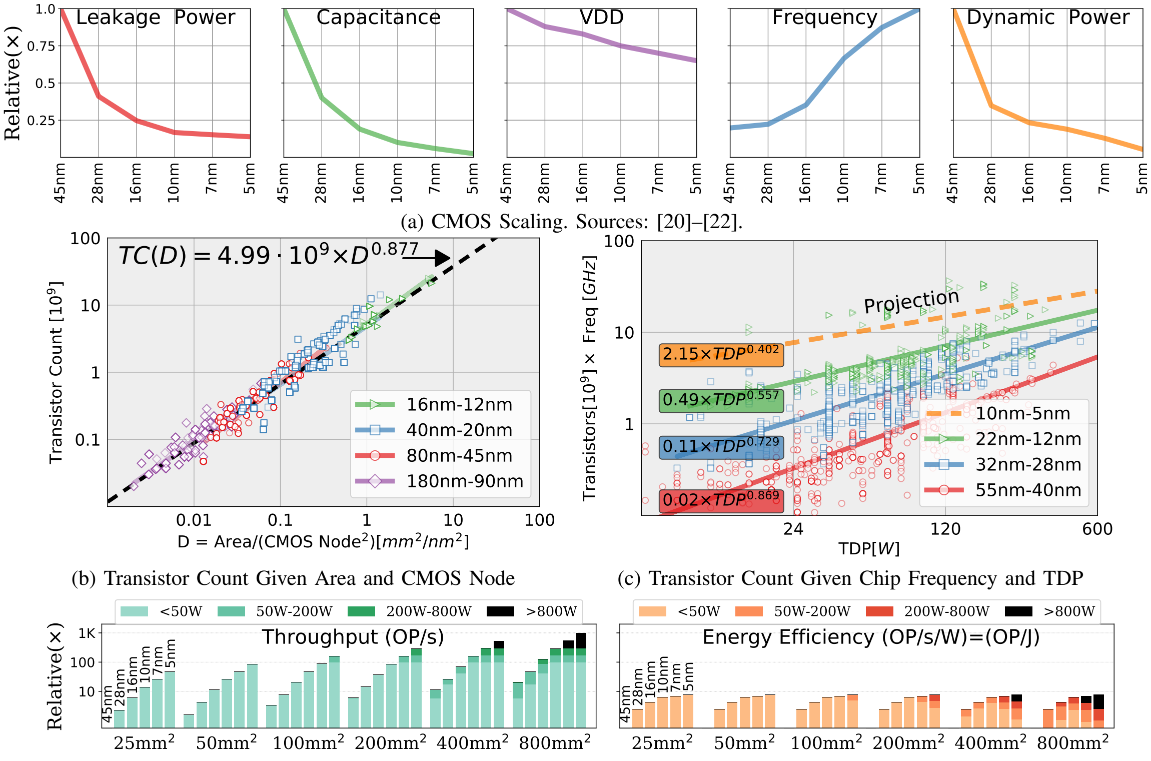
The Case For Specialization-Driven Roadmaps: While CMOS scaling has been the driving force of decades of exponential gains in processing capabilities, chip specialization improved the gains of specific applications or domains under a given budget. The focus on specific domains alleviates design restrictions and eliminates the structural support for unnecessary functionality [16]. We believe that future processing roadmaps and evaluation methods will become specialization-driven, focusing on specific domains [17,18] and shift away from traditional evaluation methods of benchmarking processors on a range of applications (e.g., SPEC) [19]. Similarly, new metrics will become a necessity to evaluate the quality of specialization-driven architectures.

Quantifying Chip Specialization Return: Chip specialization benefits from the co-optimization of different elements in the hardware-software stack such as the programming model and chip microarchitecture. Figure 2 outlines the abstraction layers of traditional architectures and our taxonomy of layers that comprise an accelerator-centric architecture. The ability to specialize a chip and improve returns, such as performance or energy efficiency, under a fixed budget is the main factor that makes accelerators relevant and appealing for future architectures. In an era of non-scaling CMOS, after transistor improvements stagnate, gains would become dependent on fixed-budget chip specialization.

We analyze chip *gains*. A gain is a target function the chip strives to maximize (e.g., energy efficiency) for the computation domain (the top layer). The gain is measured by executing the targeted computation on the analyzed chip, and it depends on the layers that are not fixed, i.e., Algorithm (Alg), Framework (Fwk), Platform (Plt), Engineering (Eng), and Physical(Phy). Since the complexity of chip architectures makes it challenging to measure the contribution of individual components to overall gain we define *Chip Specialization Return (CSR)* as the ratio between the gain and the gain from a chip's physical properties (e.g., due to more transistors):

$$CSR(Alg,Fwk,Plt,Eng) = \frac{Gain(Alg,Fwk,Plt,Eng,Phy)}{Gain(Phy)} \tag{1}$$

The CSR metric defined in Equation 1 is used as a metric to quantify the merit of chip specialization, i.e.: *"how much did the chip's compute capabilities improve under a fixed physical budget?"* The value of CSR improves in accordance to different components from the specialization stack, e.g.: better compilers or hardware-expressive libraries (programming framework), switching from FPGAs to more efficient ASICs (platform), advancement of chip design methodologies and RTL languages (chip engineering), and so on. By decoupling returns of chip specialization from transistor-driven benefits, the CSR can also be used to estimate the gains of future



(a) CMOS Scaling. Sources: [20]–[22].

(b) Transistor Count Given Area and CMOS Node

(c) Transistor Count Given Chip Frequency and TDP

(d) Chip Gains: Throughput and Energy Efficiency for Different CMOS Nodes, TDPs, and Die Sizes for: $f_{Chip} = 1GHz$

Figure 3: CMOS Potential Model: Device Scaling, Chip Transistor Budget, and Physical Chip Gains.

specialized chips. We use a comparative approach to estimate the impact of specialization as chip gains improve, and establish a roadmap of chip specialization trends. Given two chips, with their reported gains for the targeted computation domain, $Gain_A$ and $Gain_B$, under the accelerator taxonomy are as follows:

$$\underbrace{\frac{Gain_A}{Gain_B}}_{Reported\ \ Gains} = \underbrace{\frac{CSR(Alg_A, Fwk_A, Plt_A, Eng_A)}{CSR(Alg_B, Fwk_B, Plt_B, Eng_B)}}_{Specialization-Driven\ \ Gains} \times \underbrace{\frac{Gain(Phy_A)}{Gain(Phy_B)}}_{CMOS-Driven\ \ Gains} \qquad (2)$$

From the relations in Equation 2, we can examine trends of CSR for an application domain and a group of accelerators using the reported gains (e.g., energy efficiency or throughput), given a physical chip potential model (constructed in Section III). These will provide us with the ability to evaluate the hardware/software co-design space for popular application domains through the following questions: (i) to what extent do specialized chips depend on the nearly-ending improvement in CMOS transistors capabilities, and to what extent do they rely on an improved specialization? (ii) when CMOS transistors stop improving, what are the projected gains for a target application before specialization reaches its limits?

III. CMOS Potential Model

We construct an application-independent model to estimate the CMOS-driven capabilities of a chip, given its physical properties. This model allows us to decouple the chip's specialization gains from its transistor driven gains. We build the model using datasheets of 1612 CPUs and 1001 GPUs we gathered from online sources [19,23,24]. Our model receives as input the following: (i) CMOS node (N), (ii) the die size (A) or transistors count (TC), (iii) chip operation frequency, and (iv) the chip thermal design power (TDP). While our model is not confined to a specific target function, we focus on throughput and energy efficiency.

Device Scaling Model: To obtain device-level properties, we use contemporary scaling equations [20] and projections for 5nm CMOS from the recent IRDS report [22] and construct the model shown in Figure 3a which we use to model changes in transistor density, speed, energy, and power.

Transistor Budget Model: As transistor count is not always disclosed for specialized chips, we approximate the number of chip transistors. We use logarithmic regression with least mean square errors (MSE) to fit the exponential curve of transistor count as a function of chip transistor density factor (D), which is proportional to the die area, A, and inversely proportional to the square of the node size, i.e.: A/N^2. Figure 3b shows the model constructed based on the datasheets. Empirically, transistor count scales sub-linearly to the density factor, since for larger chips the design complexity makes it harder to fully-utilize the chip. While Figure 3b projects that for large 5nm CMOS chips ($D \leq 30$) the number of transistors can reach 100 billion, not all of them can be utilized. As classic device scaling rules no longer apply to modern CMOS nodes, chips are limited by the TDP budget amid the increasing power density. Power limitations restrict the fraction of active chip transistors to keep dissipation rates within a TDP envelope [10,11]. Figure 3c shows the exponential curves of the number transistor as a function of the chip TDP and operation frequency for different CMOS nodes. Given the TDP, CMOS node, and frequency, we use our model to derive the number of active chip transistors.

Chip Gains Model: We integrate the CMOS scaling model with the transistor budget models, and construct the physical chip gain model. Figure 3d depicts the relative scaling of the chip's throughput and energy efficiency (executed operations per energy spent). We treat chip throughput as the targeted performance since we explore applications that possess high degrees of parallelism, like most accelerated workloads studied in literature [8,16,25,26]. Gains were normalized to a 25mm^2 chip fabricated with 45nm CMOS. Using the transistor budget TDP model, we show the gains for different numbers of active transistors under various power envelope zones. As reflected by the figure, power constraints cap the gains of large chips. For example, the figure shows that, while the relative throughput of an 800mm^2 chip with 5nm transistors is $\sim 1,000\times$, under an 800W envelope, it drops by about 70% to $\sim 300\times$. As expected, small chips are favorable for energy efficiency. As chips get larger, the high transistor count and static power of new CMOS nodes make old nodes more appealing under a restricted TDP.

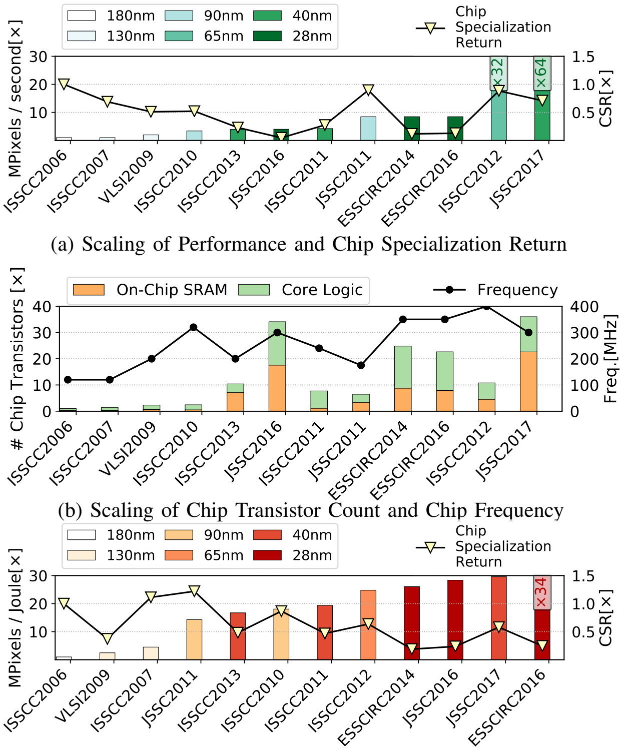
We use the CMOS potential model to measure how specialized chips scale with respect to their transistor-driven capabilities for a given domain. The model allows us to decouple transistor-driven gains from specialization-driven gains. Finally, in Section VII, we combine the potential model and estimated specialization-driven gains to project the attainable gains of future accelerators, ultimately reaching the accelerator wall imposed by the end of CMOS scaling.

IV. Empirical Specialization Returns

In this section, we characterize the trends of chip specialization return using popular accelerator domains: video decoding ASICs, graphics rendering GPUs, Convolutional neural networks on FPGAs, and Bitcoin mining using CPUs, GPUs, FPGAs, and ASICs. For each domain, we use the model from Section III to model the physical potential of the examined chips. In terms of the abstraction layers in Figure 2, by isolating the domain and physical layers, we examine the influence of the specialization stack layers: algorithm, programming framework, chip platform, and chip engineering.

A. ASIC Video Decoders

The footprint of digital videos has increased in recent years, due to the ubiquity of mobile devices, video sharing applications, and online services such as Netflix, whose streaming bandwidth peaks at 37% of overall internet traffic [39], and YouTube, which store hundreds of new video hours uploaded every minute [40]. Popular videos in online services can be decoded millions of times by HDTVs, smartphones, and tablets. These trends, and the spatial nature of videos, motivate the use of specialized video IP cores in commercial SoCs.



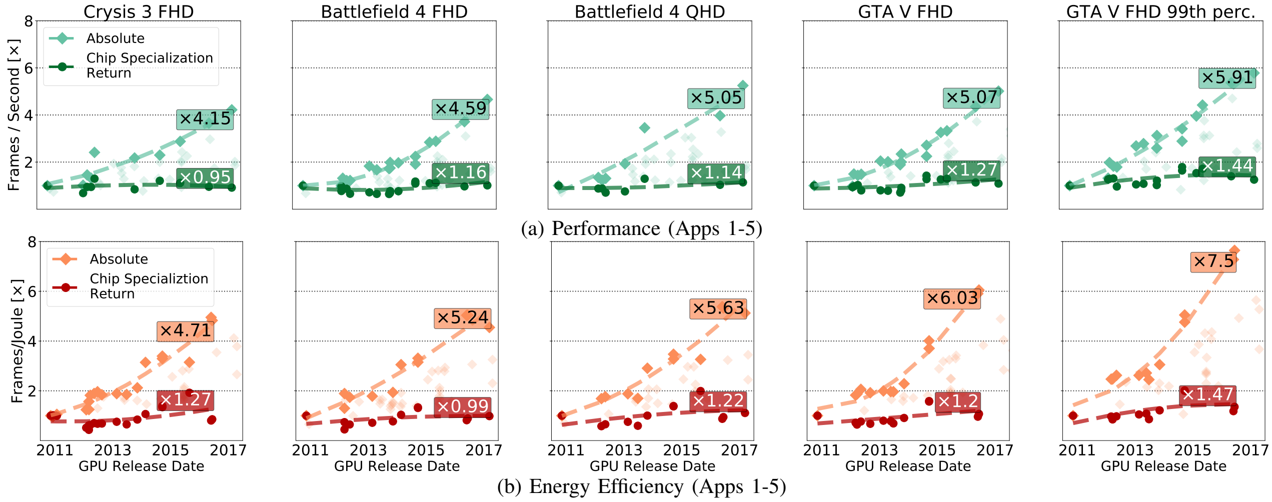
(a) Scaling of Performance and Chip Specialization Return

(b) Scaling of Chip Transistor Count and Chip Frequency

(c) Scaling of Energy Efficiency and Chip Specialization Return

Figure 4: Video Decoder ASICs: Performance, Hardware Budget, and Energy Efficiency. Sources: [27]–[38]

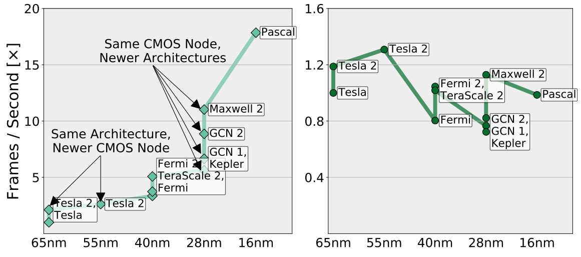
Impact of The Entire Specialization Stack: We evaluate fabricated video decoding ASIC chips and examine their gains with respect to throughput and energy efficiency. Figure 4 shows the scaling trends of performance, on-chip memory and core logic transistors, clock frequencies, and energy efficiency. Gains are presented in an ascending manner and are normalized to the least performing ASIC. Figure 4b shows estimations of the number of transistors given the number of NAND logic gates, and the number of SRAM bits for on-chip memory and auxiliary buffers for logical units. Not all works are presented in Figure 4b since some works did not specify the size of on-chip SRAMs. As Figures 4a and 4c show, compared to the baseline ASIC presented in ISSCC2006 [27], absolute decoding throughput and throughput per energy improved by rates of up to 64× and 34×, respectively. In contrast to throughput and throughput per energy, for the best performing ASICs, chip specialization did not improve, and even got worse since CSR was less than one. This is caused by the increase in chip transistor count (JSSC2017 has$\sim$36× more transistors) which enables more processing elements in parallel, and the improvement in energy efficiency for used CMOS nodes (180nm versus 40nm in JSSC2017, and 28nm in ESSCIRC2016) outpaced the improvement in overall performance and energy efficiency. Therefore, the physical layer had a higher impact than the layers in the specialization stack.



(a) Performance (Apps 1-5)

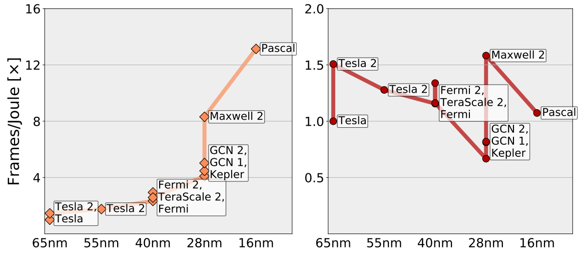
(b) Energy Efficiency (Apps 1-5)

Figure 5: GPU Frame Rates (Apps 1-5): Throughput and Energy Efficiency



(a) Absolute (b) Chip Specialization Return

Figure 6: Architecture + CMOS Scaling: Throughput



(a) Absolute (b) Chip Specialization Return

Figure 7: Architecture + CMOS Scaling: Energy Efficiency

B. GPU Graphics Rendering

GPUs are the most widespread specialized chips, due to the popularity of massively-parallel workloads and maturity of programming environments such as CUDA and OpenCL. As early GPU application scope focused on graphic applications, we would like to examine the behavior of this mature domain. We analyzed a database of reported graphics benchmark results [41] and combined it with the database of GPU datasheets we used to construct the CMOS potential model in Section III. Within these results, we have selected 24 popular game benchmarks including Battlefield 4, Crysis 3, GTA V, and Portal 2. Figure 5 shows the results of five applications (other applications show similar trends). Each of the presented applications was tested on over 20 different GPUs and normalized to the oldest GPU chip evaluated. Opaque markers show the gains of high-performance GPUs, and translucent markers show the gains of mid-end and low-end GPUs. We use quadratic curve fitting to construct curves for the reported frame-rates and CSR (frame-rate versus CMOS potential). We see that over a period of six years performance increased by $4-6\times$ (Figure 5a) and energy efficiency increased by $4.5-7.5\times$ (Figure 5b). However, the CSR trends reflect that performance and energy efficiency did not improve much, if at all, beyond the CMOS potential.

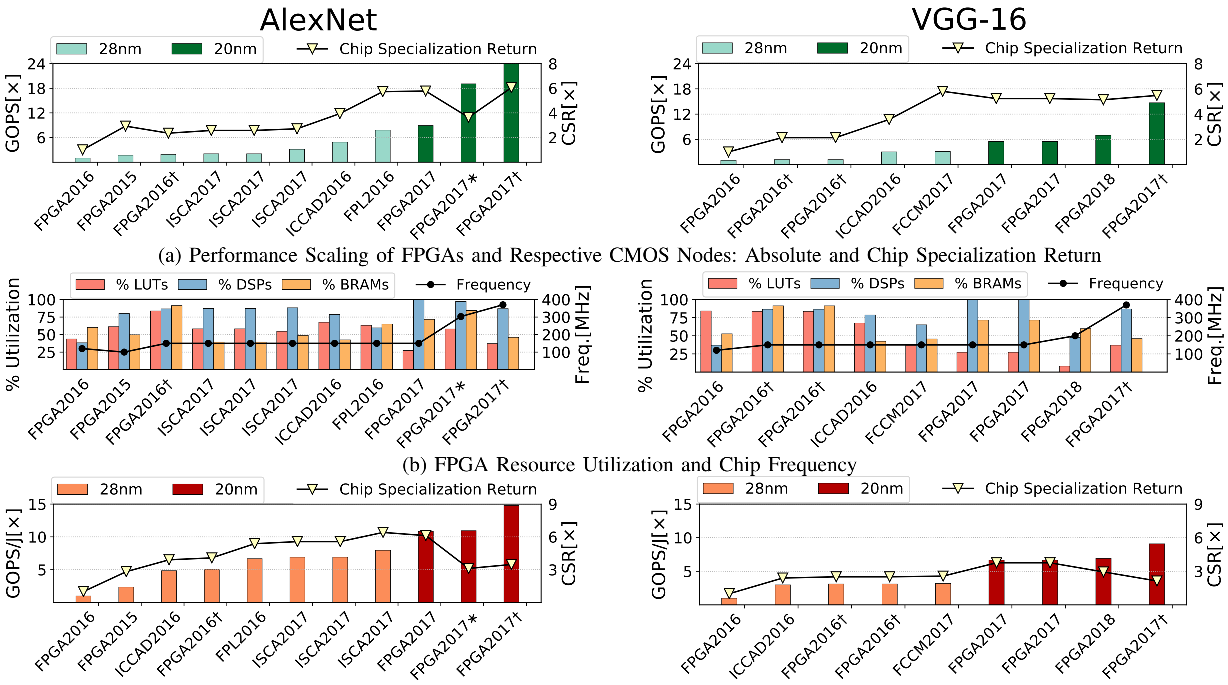
Impact of Programming Framework and Chip Engineering: We evaluate the effects of advancements in GPU architectures. In terms of the specialization stack, shown in Figure 2, this is a way of quantifying the benefits from: (i) Improved chip engineering, as new architectures are designed with newer tools and compilers, and using matured engineering disciplines. (ii) Better programming platforms, e.g., newer CUDA versions that support new computationally-driven libraries (e.g., video decoding, and sparse matrix computations), performance primitives, and ISA extensions [42]. We compare the gains between each pair of GPU architectures (and potentially CMOS nodes), $<\text{Arch}, X>$ and $<\text{Arch}, Y>$, using the relative geometric gains mean of all shared applications, meaning:

$$Gain(X \mapsto Y) = \sqrt[N]{\prod_{\ell=1}^{N} \frac{Gain_{<\text{Arch},X>}(App_\ell)}{Gain_{<\text{Arch},Y>}(App_\ell)}} \tag{3}$$

We set the relative gains of architecture pairs with at least five shared applications ($N \geq 5$), and use transitivity to compute the relative gains of architecture pairs with less than five shared applications. If relative gain between architectures X and Y was not set, we use the geometric means of all M intermediary architectures with relations to X and Y:

$$Gain(X \mapsto Y) = \sqrt[M]{\prod_{\Gamma=1}^{M} Gain(X \mapsto \Gamma) \cdot Gain(\Gamma \mapsto Y)} \tag{4}$$

We iteratively construct the relations matrix, until we do not add a new pair. Figures 6 and 7 show the trends of performance and energy efficiency of GPU architectures and their respective CMOS nodes. We see that as expected, for a given CMOS node, newer architectures deliver better absolute gains. In terms of CSR, the first architectures to be implemented on a new CMOS node always seem to perform worse than their predecessors on the old node (for example Fermi). As the given CMOS node stabilizes, architectures become more mature and yield better CSR. However, for both performance and energy efficiency, the overall CSR



(a) Performance Scaling of FPGAs and Respective CMOS Nodes: Absolute and Chip Specialization Return

(b) FPGA Resource Utilization and Chip Frequency

(c) Energy Efficiency Scaling of FPGAs and Respective CMOS Nodes: Absolute and Chip Specialization Return

Figure 8: FPGA Implementations of AlexNet and VGG-16. Sources: FPGA2016 [43], FPGA2016† [44], FPGA2016∗ [45], FPGA2017 [43], FPGA2017† [46], FPGA2017∗ [47], FPGA2018 [48], FPGA2018† [49].

does not improve over time, as the CSR for the 16nm Pascal is roughly the same as that of the 65nm Tesla.

While newer GPU architectures tend to deliver better performance and energy efficiency, when accounting for average specialization return rates, newer GPUs do not necessarily perform better and sometimes even perform worse than their expected potential. The reason is that newer, better-performing GPUs were implemented using newer CMOS nodes with higher density and energy efficiency, therefore on average, the impact of better CMOS potential is higher than the impact of average improvement over that potential. Furthermore, maintaining even a fixed CSR is challenging since it requires new programming models and architecture capable of delivering the same relative gains for the newer and more-complex chips. Figures 6 and 7 reflect that, while GPU frame-rate gains improved by $13-16\times$, average CSR rates were an order-of-magnitude less, i.e., $1.0-1.6\times$, implying that CMOS potential is the dominating factor in the GPU graphics roadmap.

C. FPGA Convolutional Neural Networks

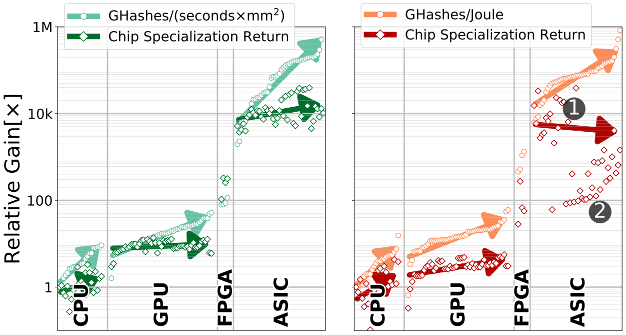
Machine learning algorithms became popular due to the increase in demands for user-personalized experience, the ubiquity of user-owned devices [50], and the increase in generated data which needs to be processed to provide a better experience on user-owned devices. The recent success of machine learning algorithms based on Convolutional Neural Networks (CNN) in image recognition algorithms has sparked growing efforts to implement CNN image-recognition algorithms in hardware. We examine FPGA implementations of two popular models, that were notable landmarks following their performance in the ImageNet Large Scale Visual Recognition Competition [51]: (i) “AlexNet” which reduced the top-5 error from 26% to 15.3% in 2012 [52], and (ii) “VGG-16” which further reduced the top-5 error to 7.3% in 2014 [53].

Impact of Algorithm: Figure 8 shows the performance, resource utilization, and energy efficiency for FPGAs implementing both CNN models. All studies use FPGAs implemented in 28nm or 20nm CMOS. While most evaluated works explored ways to optimize the data layout and resource partitioning, some works leveraged algorithmic optimizations. FPGA17† used built-in GEMM optimizations in OpenCL and BRAM access locality to reduce the memory bandwidth requirement of CNNs. In FPGA1027∗ [47] the authors applied the Winograd transform [54] to exploit the locality in small 3×3 filters (used in AlexNet) and improve throughput by minimizing the complexity of Convolutional operations [55]. As Figure 8 shows, FPGA CMOS technology had a high impact on gains, as most 20nm FPGAs were superior to the 28nm FPGAs in terms of performance and energy efficiency. While AlexNet performance and energy efficiency improved by about 24× and 14×, respectively, VGG-16 improved by about 9× and 7×, in terms of performance and energy efficiency. A source for these disparities lies in the model size. The amount of data needed to represent VGG-16 is three times the amount of data for AlexNet [56], and the amount of operations per image is about 20× [46]. The size of the model stresses FPGA resources, making it harder to optimize FPGAs for computation pattern reuse, and to run at the same clock frequencies as an FPGA implementing AlexNet. While CSR improved by up to 6× in both models, for the best performing FPGAs in each model CSR did not improve while absolute performance increased. For these designs performance improved due to better physical budget (higher utilization of FPGA resources). As CNNs are a relatively new domain, there is hope for new algorithms to emerge and achieve better CSR, i.e., better gains per fixed FPGA budget.

D. CPU/GPU/FPGA/ASIC Bitcoin Miners

Bitcoin and other cryptocurrencies made an impact on the global economy, with an equally important aspect being the energy spent to produce new currency blocks. A block stores a portion of the network transactions in exchange for fees paid to the block generator. In “proof-of-work” cryptocurrencies such as Bitcoin, blocks are added to the blockchain in a process called “mining”, which requires solving a computationally intensive problem, whose difficulty increases with the number of blocks. In the period of August-October 2018, the aggregated energy consumed by Bitcoin miners has peaked, and it was estimated at 73.12 Tera Watt-Hours annually, which is the more than the energy spent by Austria [57]. While first generation miners relied on CPUs, the growing energy costs and the fact that mining computation relies on a fixed SHA-256 hash function [58] incentivized hardware specialization, and mining hardware shifted to GPUs and later FPGAs, which were quickly overtaken by ASIC miners [59]. We constructed a mining database using datasheets and data collected from online forums to compare Bitcoin mining CPUs, GPUs, FPGAs, and ASIC chips [60]–[63]. As ASIC miners significantly differ in the number of integrated chips and their sizes, we treat performance per chip area as the performance metric, as it is a better indicator of chip performance than absolute throughput.

Impact of Chip Platforms: Figure 9 shows the mining gains of all tested chips. As reflected by the figure, ASIC chip gains beat CPUs by several orders-of-magnitude, as their initial chip specialization return rate is high. Following the initial ASIC improvement, specialization return does not improve (and even declines for energy efficiency). Figure 9a shows how performance per area has improved by almost 600× across different ASICs (and about 600,000× compared to the baseline CPU miner), but since physical capabilities improved by 300×, specialization returns improve by about 2× across ASICs. The reason for this disparity stems from the rapid transitions of one CMOS node to the next. This is demonstrated in the two distinctive energy efficiency regions, ❶ and ❷ , in Figure 9b. In ❶ specialization return rates improve for the early Bitcoin miners (130nm and 110nm) and in ❷ energy efficiency specialization return rates improve for the modern Bitcoin miners (28nm and 16nm), but the sharp decline in specialization returns between the two regions is the result of the transition from 110nm to 28nm in a period of 15 months. This advancement outpaced Moore’s law roadmap, introducing better silicon products at a faster rate than algorithmic innovations, and therefore most benefits of ASIC miners in that period are attributed to better chip physics. This is the result of the nature of Bitcoin mining eco-system. Initially, inexpensive platforms were used, but following the increase in difficulty, miners moved to expensive ASICs with new energy efficiency CMOS nodes, since the energy spent became the dominating factor for mining revenues.



(a) Performance

(b) Energy Efficiency

Figure 9: Bitcoin Mining Capabilities of CPU, GPU, FPGA and ASIC chips (vs. AMD Athlon 64 CPU Miner).

E. Observations and Insights

We summarize our insights from the conducted studies.

Specialization Returns and Computation Maturity: From our experiment it was apparent that for mature computation domains like video encoding or gaming framerates, specialization returns either plateau or drop for high performing chips. This is expected, because while physical capabilities exponentially scaled in the last decade, algorithmic innovation did not scale in a similar fashion for well-studied problems. For emerging applications such as Convolutional neural networks, shown in Section IV-C, the counter phenomena can be seen. As this domain is gaining traction, with multiple academic papers exploring a variety of algorithms on similar chips (20nm and 28nm FPGAs), we see improved CSR, by it will likely plateau as domain becomes more mature.

Introduction Of a New Specialization Platform Delivered a Non-Recurring Boost in Gains: As seen in Figure 9, most CSR gains were obtained by the transition to a new platform (e.g., from FPGA to ASIC). However, following that transition, CSR did not significantly improve, and gains were mainly attained via better CMOS capabilities.

Confined Computations: The stagnation of specialization returns for all platforms (i.e., GPUs, ASICs, etc.) in Figure 9 emphasizes the challenge of algorithmically innovating a confined domain such as Bitcoin mining. Aside from ASICBoost [64] that delivered a one-time 20% improvement by parallelizing the inner and outer loops in the algorithm, most miners operate in a brute-force and parallelized manner. Following the end of CMOS scaling, confined domains such as Bitcoin mining will become bound by the limited number of ways to represent the core algorithm in hardware.

While Motivated by Transistor Limitations, Specialized Chips Still Highly Depend on Transistors: In all experiments and for all chip specialization types, physical capabilities had a high impact on gains. When CMOS scaling ends, specialization improvements will slow down, and gains

		Simplification		Partitioning		Heterogeneity
Memory	❶	Simple DDR3 chips, interfaces, and physical memory space	❷	Memory module banking storing NN layer weights	❸	Hybrid memory for input and intermediary results
Communication	❹	Simple FIFO communication	❺	Concurrent FIFOs for weights and systolic array data	❻	Software-defined DMA Interface for chip I/O
Computation	❼	Multiply+add computation units with small precision (8-bit integers)	❽	Parallel multiply+add paths	❾	Non-linear activation unit (e.g., ReLU) and systolic array data reuse

Table I: Chip Specialization Concepts. Examples From a TPU ASIC Chip.

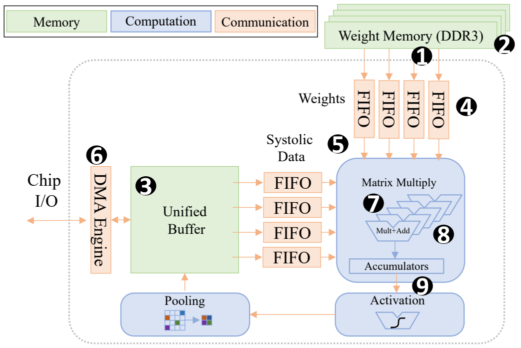


Figure 10: TPU: DNN Inference + Specialization Concepts

will remain solely dependent on improving specialization returns, that empirically scale more modestly. Therefore, the trends of CMOS and specialization returns across accelerators serve as a good indicator to assess the limits of specialization for a target domain before reaching the accelerator wall.

V. Chip Specialization: Concepts

The idea of chip specialization suggests that the knowledge of a computation domain makes it possible to couple computations from the domain layer with underlying structures implemented using the physical layer. By employing only compute-essential hardware structures, chip specialization circumvents the inefficiencies of general-purpose hardware [16,25,26]. In this section, we discuss the techniques to achieve this goal, and conduct an exploration of the chip specialization design space, focus on its theoretical limits.

A. *Chip Specialization Concepts*

We formalize the process of chip specialization and its relations to the three processing components: **memory, communication, and computation**. We observe three concepts of chip specialization, each can be applied independently to a component, or across multiple components.

Simplification: A narrow problem domain space provides architects with the ability to reduce the complexity of hierarchies and datapaths and attain simpler structures with similar functionality for reduced costs. Simplification can be structural (e.g., narrowing underutilized buses and datapaths for problems with limited variable representation), functional (e.g., removing floating point units for integer problems), or control (e.g., removing energy costly OoO pipe control).

Partitioning: Typical accelerated applications were shown to possess high degrees of parallelism [8,16]. This property motivated the development of concurrent hardware designs with replicated paths that operate independently on sub-portions of the application data. Common forms of partitioning are SIMD/VLIW vectorization, threaded parallelism, and bisection of on-chip mesh NoCs.

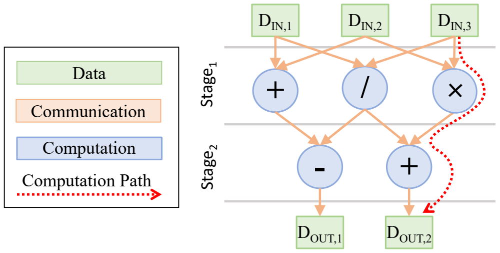


Figure 11: DFG Example: 3 Inputs, 2 Computation Stages, 2 Outputs (An Example Computation Path Is Marked In Red).

Heterogeneity: Even a single workload has diverse needs: certain computations require non-trivial functionality, a workload might possess phases with different computation intensities or reuse numerous distinct patterns. While simplified and partitioned hardware improves efficiency for these cases, further reduction of energy per computation is possible by diversifying the computation paths and tailoring each path to support specific functionality. Common forms of heterogeneity are fusing of functional units (e.g., clustering of instructions [65] or dataflow graph phases [66]), algorithm-specific functional units (e.g., tangent activation units for neural networks), and asymmetric memory banks and network topologies to accommodate irregular data patterns.

Nowatzki et al. [8] suggested a taxonomy of principles: concurrency, communication, data-reuse, computation, and coordination, that can be mapped to the concepts we define. We found our classification a better fit for this limit study.

Case Study: Tensor Processing Unit: The increase of user-generated sensory data necessitates high processing power. Google engineers designed a 28nm ASIC chip called a Tensor Processing Unit (TPU) to avoid from having to double their CPU-based datacenters to meet the demands of speech recognition alone [4]. They demonstrated how TPUs improve the energy-efficiency of deep neural network (DNN) workloads by 80× compared to CPUs. Figure 10 shows a simplified block diagram of a TPU, with annotated examples of all specialization concepts, summarized in Table I.

B. *The Limits of Chip Specialization Concepts*

We discuss the limits of chip specialization concepts. We present the target computation problem as a dataflow graph (DFG). A DFG is a concise representation of computation problems, limited solely by inherent computation restrictions (e.g., data dependencies), and not by implementation mediums (e.g., timing, area, or power restrictions). This property makes DFG optimization a useful way to model the design space visible to the specialization stack layers in Figure 2.

The DFG is a directed-acyclic graph (DAG), $G(V,E)$, in which V is the set of vertices: $V = \{v_1, v_2.., v_{|V|}\}$ and E is the set of edges: $E = \{e_{v_i,v_j} | v_i, v_j \in V\}$. The DFG succinctly

		Simplification	Heterogeneity	Partitioning
MEM.	Time	$\Theta(\lvert V\rvert \cdot log(max\lvert WS_s\rvert))$	$\Theta(D)$	$\Theta(D \cdot log(max\lvert WS_s\rvert)$
	Space	$\Theta(max\lvert WS_s\rvert)$	$\Theta(\lvert E\rvert)$	$\Theta(max\lvert WS_s\rvert)$
COMM.	Time	$\Theta(\lvert E\rvert)$	$\Theta(D)$	$\Theta(D)$
	Space	$\Theta(\lvert V\rvert)$	$\Theta(\lvert E\rvert)$	$\Theta(max\lvert WS_s\rvert)$
COMP.	Time	$\Theta(\lvert E\rvert)$	$\Theta(\lvert V_{IN}\rvert)$	$\Theta(D)$
	Space	$\Theta(1)$	$\Theta(2^{\lvert V_{IN}\rvert} \cdot \lvert V_{OUT}\rvert)$	$\Theta(max\lvert WS_s\rvert)$

Table II: Summary of Time and Space Complexity Limits for Chip Specialization Concepts, in Terms of DFG Definitions.

describes the high-level relations of data and computation, and how data flow across vertices. Definitions:

- *Input variables set* is a set of vertices with no incoming edges: $V_{IN} = \{v_{in_1},..,v_{in_N}\} \subseteq V$
- *Output variables set* is a set of vertices with no outgoing edges: $V_{OUT} = \{v_{out_1},..,v_{out_M}\} \subseteq V$
- *Computation nodes set* vertices with incoming edges and outgoing edges, and represent computation operands: $V_{CMP} = \{v_{cmp_1},..,v_{cmp_M} | v_{cmp_i} \notin V_{IN}, v_{cmp_i} \notin V_{OUT}\} \subseteq V.$
- *Computation paths set* is a set of all graph routes, i.e., vectors of edge-connected vertices, starting with an input variable and ending with an output variable: $P = \{\left(v_{p_1},..,v_{p_K}\right) | e_{v_{p_i},v_{p_{i+1}}} \in E, v_{p_1} \in V_{IN}, v_{p_K} \in V_{OUT}\}$
- *The DFG depth* is length of the longest computation path, i.e.: $D = max\{K | \left(v_{p_1},..,v_{p_K}\right) \in P\}$
- *Computation stage working set* is the set of variables computed in computation stage s, i.e.,: $WS_s = \{v_{p_{1,s}},..,v_{p_{n,s}} | \left(v_{p_{1,1}},..,v_{p_{1,s}},..\right),..,\left(v_{p_{n,1}},..,v_{p_{n,s}},..,\right) \in P\}$

Figure 11 shows a DFG with three input variables, two computation stages, and two output variables. We use DFG definitions to explore hardware optimization bounds, and consequently, chip specialization limits. For completeness, we do not account for hardware limits (e.g., unlimited area).

Memory Simplification: Memory optimizations target storage and memory access costs. Memory simplification reduces storage at the expense of access performance. The simplest hierarchy consists of a single module that stores only needed variables. The minimal state that needs to be stored in the memory hierarchy is the number of variables processed at a given time. Therefore it is bound by the largest working set size: $\Theta(max|WS_s|)$. For simplicity, and without loss of generality, we assume fixed variable sizes, hence reading or writing has $\Theta(1)$ cost, and access costs depend on the cost of lookups. Lookup is bounded by the variable naming space. Therefore, lookup costs are at least $\Theta(log(max|WS_s|))$ for a single variable, and for each stage of the stage's computation nodes access memory sequentially. Since all nodes have to access the memory, timing complexity is $\Theta(|V| \cdot log(max|WS_s|))$.

Memory Heterogeneity: We express memory heterogeneity as a hierarchy formed by a layout of modules and/or interfaces to support problem-specific access patterns. Maximal performance is achieved by a banked hierarchy that reflects all computation relations across nodes. Since relations are described as DFG edges, storage costs are on the order of $\Theta(|E|)$. The hierarchy performs fast $\Theta(1)$ accesses, done in parallel in each stage. Total timing complexity is on the order of stages, or DFG depth, i.e., $\Theta(D)$.

Communication Simplification: A communication fabric is simplified by reducing the number of wires at the expense of increased latency. The limit is a minimal spanning tree connecting all nodes. Further reduction of wires would leave unconnected nodes, therefore the minimal wire complexity is $\Theta(|V|)$. Since data has to traverse across all nodes to satisfy all dependencies defined by the edges, the timing complexity, in terms of network hops, is $\Theta(|E|)$.

Parameter	Explored Values
Partitioning Factor	1, 2, 4, ... 524288
Simplification Degree	1, 2, 3, .., 13
CMOS Process (nm)	45, 32, 22, 14, 10, 7, 5

Table III: CMOS-Specialization Sweep Parameters.

Application	Abbrev.	Domain
Advanced Encryption Standard [13]	AES	Cryptography
Breadth-First Search [13]	BFS	Graph Processing
Fast Fourier Transform [13]	FFT	Signal Processing
General Matrix Multiplication [13]	GMM	Linear Algebra
Molecular Dynamics [14]	MDY	Molecular Dynamics
K-Nearest Neighbors [13]	KNN	Data Mining
Needleman-Wunsch [13]	NWN	Bioinformatics
Restricted Boltzmann machine [15]	RBM	Machine Learning
Reduction [14]	RED	Microbenchmarking
Sum of Absolute Differences [67]	SAD	Video Processing
Merge Sort [13]	SRT	Algorithms
Sparse Matrix-Vector Multiply [13]	SMV	Linear Algebra
Single Source, Shortest Path (Internal)	SSP	Graph Processing
2D Stencil [13]	S2D	Image Processing
3D Stencil [13]	S3D	Image Processing
Triad [14]	TRD	Microbenchmarking

Table IV: Evaluated Applications and Domains.

Communication Heterogeneity: Since the DFG topology is structured in a way that reflects the problem-specific communication patterns, communication heterogeneity is explicitly defined by the DFG edges. The wiring complexity is on the order of network edges, i.e., $\Theta(|E|)$. The timing complexity is the DFG delay, i.e., the number of stages in the longest computation path, or DFG graph depth $\Theta(D)$.

Computation Simplification: DFG nodes express problems using pre-defined operands (e.g., ADD, MUL). As computation completeness dictates that mathematical operations can be computed using logic gates, a node is reduced to a set of $\Theta(1)$ gates. For fixed-size variables, operands are reduced to a fixed number of gates, and computation is done in a serial bitwise manner for each input-output pair. Therefore, the total timing is the number of nodes × number of input variables × of variable bits, which is on the order of all edges that express the relations between the problem nodes, or $\Theta(|E|)$.

Computation Heterogeneity: Computation heterogeneity is done by fusing nodes to form problem-specific "super nodes". The extreme case for fusing is a single node which acts as a lookup table that stores the computation results of all computation inputs. The table would have $\Theta(2^{|V_{IN}|})$ entries (total input bits), each entry contains the computation result, which the number of output variable bits, i.e., $\Theta(|V_{OUT}|)$. The total space complexity is, therefore: $\Theta(2^{|V_{IN}|} \cdot |V_{OUT}|)$. The time complexity consists of the time for lookup $\Theta(log(2^{|V_{IN}|})) = |V_{IN}|$ and reading the computation output, i.e., $\Theta(|V_{OUT}|)$ and in total $\Theta(|V_{IN}| + |V_{OUT}|)$. Partitioning the table to $|V_{OUT}|$ tables, each storing an output variable, would reduce output variable read time to $\Theta(|1|)$, resulting in a time complexity of $\Theta(|V_{IN}|)$, the same time as reading all inputs ($\Theta(|V_{IN}|)$), solving computation in $\Theta(1)$ runtime, and writing all outputs in parallel in ($\Theta(1)$) runtime.

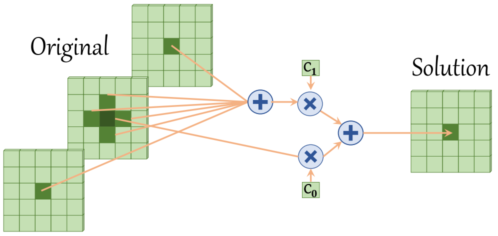


Figure 12: Visualization of a 3D Stencil Computation

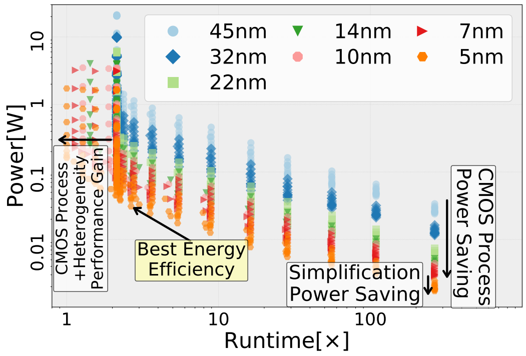
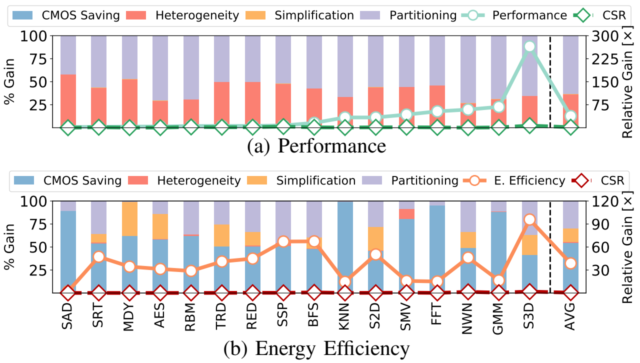


Figure 13: 3D Stencil Power, Timing, and CMOS Sweep. Arrows Highlight Optimal Point and Gain Sources.



(a) Performance

(b) Energy Efficiency

Figure 14: Specialization and CMOS Accelerator Gains.

Memory / Communication / Computation Partitioning: Partitioning is limited by the level of DFG parallelism, which is on the order of the maximum number of concurrently processed variables. This is the largest working set size, i.e., $\Theta[max|WS_s|]$, since further partitioning would produce diminishing returns. The total space complexity for a maximally partitioned system is, therefore $\Theta[max|WS_s|]$, and since the maximal path stages is: $\Theta(D)$, the lookup time for variables in the memory takes $\Theta(D) \cdot log(max|WS_s|)$, and communication and computation time complexity is $\Theta(D)$.

Table II summarizes the limits of the discussed chip specialization concepts. As chip specialization is conceptually limited, optimization space is finite and is further restricted when combined with realistic hardware limitations.

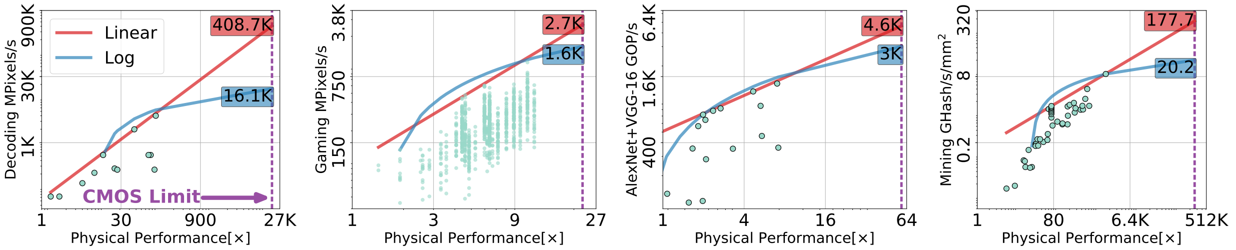
VI. Gains of Chip Specialization Concepts

While Section IV explores the empirical gains of real-world accelerators, the sources of specialization concepts that contribute to the obtained gains are not disclosed and should be quantitatively explored. We quantify the contributions of chip specialization and CMOS potential for a range of applications. We determine the optimal points for each application, attribute the sources of gains from CMOS savings (i.e., more energy-efficient CMOS nodes) and the specialization concepts described in Section V.

Methodology: Our framework is integrated with Aladdin, a modeling tool that enables fast exploration of accelerator design alternatives [12]. The original Aladdin flow supports partitioning of memory and datapaths using unrolling, pipelining and memory banking. It also models heterogeneity using DMA and scratchpads, register layout, and fusion of operands for dependent dataflow graph nodes that fit the same cycle. We extend the Aladdin flow to support CMOS scaling using our model described in Section III. We add support for simplification and pipelining of functional units and registers, using data from [68]. By applying these techniques, we approximate the hardware/software co-optimization of accelerator applications using the specialization concepts described in Section V. These concepts are also implemented in modern accelerators, e.g., Hyper-Pipelining in Stratix 10 [69], and computational heterogeneity in the Xilinx ACAP model [70]. Table III lists the explored parameters, and Table IV summarizes the evaluated applications.

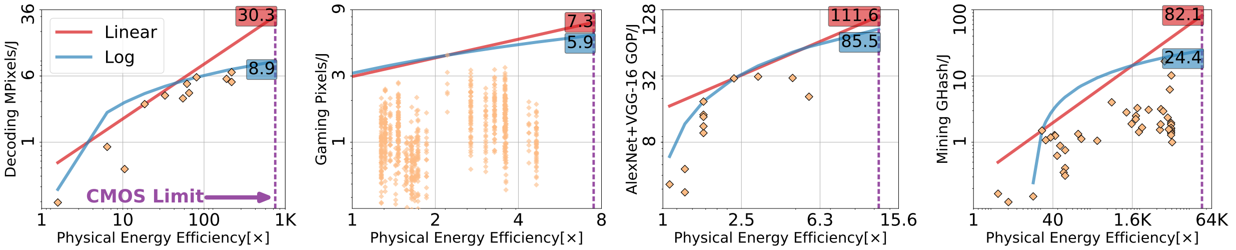
Case Study: Stencil Computation: 3D Stencil can be used to extract spatial image features for computer vision applications [71]. As Figure 12 demonstrates, the Stencil kernel is highly parallel, as filtering can be applied concurrently to different members of the "Orig" lattice. We explore different design alternatives for a 3D Stencil accelerator, shown in the Runtime-Power space in Figure 13. As expected, CMOS advancement reduces power. As the figure shows, partitioning improves performance gains for all technologies, until reaching a point where runtimes plateau, when reaching the maximal degree of kernel parallelism. Following that point, old nodes (e.g., 45nm) experience diminishing returns due to underutilized partitioned resources. However, performance still improves for newer CMOS nodes, since functional units are faster, and more computation units are fused and scheduled in a cycle. This is a combination of computation heterogeneity with advanced CMOS processes. Simplification of functional units, registers, and communication, also reduces power (as smaller structures generate less leakage power). The optimal points for energy efficiency are received for 5nm CMOS, for the highest degree of partitioning for which performance does not taper off, and the highest simplification degree that does not cause diminishing returns (i.e., increased latency due to deep pipelining).

We sweep the design space for the evaluated applications to determine the contribution of CMOS and specialization concepts. For each application, gains were normalized to a 45nm accelerator with no simplification or partitioning. Figure 14 shows the obtained gains. As simplification and CMOS power saving reduce energy and not runtime, they improved efficiency but not performance. While partitioning was the primary contributing source for performance, CMOS saving was the dominating factor for energy efficiency. For both performance and energy efficiency, CSR is low, since both CMOS saving and partitioning (i.e., using more transistors for parallelization) are inherently CMOS dependent.



(a) ASIC Video Decoding (b) GPU Gaming/Graphics (c) FPGA CNN (d) ASIC Bitcoin Mining

Figure 15: Evaluated Applications and Specialization Chips: Accelerator Performance Projections.



(a) ASIC Video Decoding (b) GPU Gaming/Graphics (c) FPGA CNN (d) ASIC Bitcoin Mining

Figure 16: Evaluated Applications and Specialization Chips: Accelerator Energy Efficiency Projections.

Computation Domain	Accelerator Platform	Min. / Max. Die Sizes [mm^2]		Thermal Power Budget [W]	Frequency [MHz]
Video Decoding	ASIC	1.68	16.0	7	400
Gaming/Graphics	GPU	40	815	345	1500
Convolutional NN	FPGA	100	572	150	400
Bitcoin Mining	ASIC	11.1	504	500	1400

Table V: Accelerator Wall: Physical Parameters.

VII. The Accelerator Wall

We explore the limitations of the applications evaluated in Section IV. The motivation for this limit study stems from the dominance of CMOS-driven capabilities on accelerator gains. As accelerators are CMOS dependent, we use the CMOS-driven potentials to estimate the attainable gains of each domain and project the accelerator wall, which is: the best performance and energy efficiency for accelerator chips manufactured after CMOS technology stops scaling.

Physical Chip Limits Parameters: We use the CMOS scaling equations and chip model in Section III to estimate the accelerator gains for chips implemented in the final CMOS node, currently projected to be 5nm [22]. Table V summarizes the explored parameters in each evaluated domain. We follow the insights from Section III, and use largest dies for performance, and smallest dies for energy efficiency. For FPGAs we used typical TDP and die size numbers reported for Xilinx FPGA boards [72], For video encoding ASICs we use a TDP budget of 7W, 10× higher than the highest power measure of 690mW in [38].

Projection Models: We examine two projection models for the evaluated accelerators. (i) The *Linear* model is a Pareto frontier projection that assumes a linear connection between physical capabilities and the chip gains:

$$\text{Projection}_{Linear}(Physical) = \alpha \cdot Physical + \beta \qquad (5)$$

(ii) The *Logarithmic* model is a Pareto frontier projection that assumes a sub-linear connection between physical capabilities and gains.

$$\text{Projection}_{Log}(Physical) = \alpha \cdot log(Physical) + \beta \qquad (6)$$

Our models account for the difference in application behavior, domain and silicon maturity, and specialization platforms. For example, GPUs rely on massive parallelism, and are likely to exhibit linear behavior with regards to physical performance (more transistors map to more cores). The sub-linear logarithmic projections reflect the difficulty in exploiting high complexity chips (e.g., due to peripheral overheads), or due to inherent (e.g., structural) algorithmic limitations in the ways the domain can be specialized using the added silicon budget.

Results: Figures 15+16 show the projections of performance and energy efficiency for the evaluated accelerators. Generally, the linear model fits the performance spaces, and the logarithmic model fits the energy efficiency spaces. Since accelerated applications posses high parallelism [16], performance scales linearly by adding more parallel processing elements. In contrast, energy efficiency requires simplifying datapaths and exploring other hardware tradeoffs. For example, while Figure 15b shows the performance Pareto frontiers for different GPUs and games behave linearly, that is not the case for Figure 16b. The high disparity of accelerator gains obtained under similar physical capabilities for GPUs is caused by the variety of different applications evaluated and targeted devices (e.g., server vs. smartphone GPUs).

Accelerator Limits: According to our limit study and given the available chip data, while ASIC video decoding performance and energy efficiency improved by 64× and 34×, respectively, we project further performance and energy efficiency improvements of $3-130\times$ and $1.2-14\times$. While GPU graphics frame rate improved by a rate of 16×, we project further performance and energy efficiency improvements of $1.4-2.5\times$ and $1.4-1.7\times$, respectively. While FPGA CNN performance improved by 22.5×, we project further performance improvements of $2.1-3.4\times$ and energy efficiency rates of $2.7-3.5\times$. While ASIC Bitcoin performance increased by 600× (and about $600,000\times$ over CPUs), we project further improvements of $2-20\times$ and

$1.4-5\times$ in performance and energy efficiency, respectively. All the limits were obtained using projections of 5nm CMOS technology, which is not commercially available yet.

Our study shows that while performance has a promising trajectory for most domains, energy efficiency is not projected to improve at the same rate. Although specialization is hard to predict, it is generally easier to predict domains that are mature, well-studied, and algorithmically-stable. For example, GPU frame rate is easier to predict than FPGA CNNs, since the analyzed CNN data spans only four years of specialization efforts. Interestingly, the properties that make specialization predictable also the same ones that make domains fit for specialization in the first place [8]–[10,16]; mature domains have a variety of adopted applications, and well-studied domains are stable and less prone to volatility and specialized hardware obsoletion. As domains mature and dominating algorithms as stabilized, so will chip specialization, and it will become harder to improve gains under a fixed chip budget. Following the end of CMOS scaling, accelerator gains will become bound by the reach of specialization, which can be quantified and projected using the metrics and methods presented in this work.

VIII. Related Work

Several works have explored the limiting factors of processing technology and innovation. One of the earliest works tried to project the practical limits of instruction-level parallelism [73]. Recent studies identified the dark silicon phenomenon that caps the number of active chip transistors [10,11]. These studies demonstrated how the slowdown of CMOS scaling and power budget limitations in multicore chips would necessitate a shift towards new architectures. In a later study, Taylor advocated for specialization as one of four avenues to mitigate dark silicon [74], and it was also motivated by Halpern et al. that examined trends in mobile and desktop processors [75]. While specialized chips are motivated by the slowdown of CMOS scaling, we demonstrate that they are prone to transistor-imposed limitations as well.

Recent studies on specialized hardware improve programmability to maintain hardware performance and relevance amid frequent algorithmic changes [8], optimize while accounting for non-recurring engineering costs in ASIC development [9], and use DRAMs to replace inefficient SRAMs in FPGAs [76]. Accelerator modeling expressed accelerated applications as transformable dependence graphs [77] and dynamic dependence graphs in Aladdin [12].

A study by Borkar and Chien decoupled chip and application performance to estimate the impacts of microarchitecture on general-purpose microprocessors [78]. In recent studies, the nearing-end of single monolithic chips was the motivation for devising multi-chip GPU architectures [79] and emerging memory based memoization architectures for accelerators [80]. To the best of our knowledge, our work is the first to quantify transistor dependence in accelerators, and explore the limitations of chips specialization foundations.

IX. Conclusions

While the end of transistor scaling has motivated a shift to specialized architectures, specialized chips still greatly depend on gaining more transistors. In this work, we conducted a careful limit study of chip specialization. In order to isolate and remove the benefits of device technology, we built a CMOS potential model using datasheets of thousands of real-world chips. We characterized how specialization returns have progressed in popular chip-specialized application domains. We demonstrated that while domain maturity and stability motivate hardware specialization, they also result in diminishing specialization returns. We identified common chip specialization concepts, presented their theoretical limits and evaluated their contribution to application gains. Finally, we conducted a limit study that projected the accelerator wall of future potential chips for each application. The limits imposed by the accelerator wall and the evaluation methodologies presented can be used as foundations for a better understanding of future accelerated applications. These will become a necessity for future designs in the era of accelerators and non-improving CMOS technology.

Acknowledgments

We thank Niraj Jha, Ruby Lee, Margaret Martonosi, Prateek Mittal, Mohammad Shahrad, and the anonymous reviewers for their useful feedback. This material is based on research sponsored by the NSF under Grants No. CNS-1823222 and CCF-1453112, Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) under agreement No. FA8650-18-2-7846, FA8650-18-2-7852, and FA8650-18-2-7862. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA), the NSF, or the U.S. Government.

References

[1] Y. S. Shao and D. Brooks, "Research infrastructures for hardware accelerators," *Synthesis Lectures on Computer Architecture*, vol. 10, no. 4, pp. 1–99, 2015.

[2] A. Putnam, A. M. Caulfield, E. S. Chung, D. Chiou, K. Constantinides, J. Demme, *et al.*, "A reconfigurable fabric for accelerating large-scale datacenter services," in *Intl. Symp. on Computer Architecture (ISCA)*, pp. 13–24, IEEE Press, 2014.

[3] K. Hazelwood, S. Bird, D. Brooks, S. Chintala, U. Diril, D. Dzhulgakov, *et al.*, "Applied machine learning at facebook: A datacenter infrastructure perspective," 2018.

[4] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Intl. Symp. on Computer Architecture (ISCA)*, pp. 1–12, ACM, 2017.

[5] NVIDIA, "NVIDIA Tesla P100." http://www.nvidia.com/object/tesla-p100.html.

[6] Amazon, “Amazon EC2 F1 instances.” https://aws.amazon.com/ec2/instance-types/f1/. [Online].

[7] I. Magaki, M. Khazraee, L. V. Gutierrez, and M. B. Taylor, “ASIC Clouds: Specializing the Datacenter,” in *Intl. Symp. on Computer Architecture (ISCA)*, pp. 178–190, IEEE Press, 2016.

[8] T. Nowatzki, V. Gangadhan, K. Sankaralingam, and G. Wright, “Pushing the limits of accelerator efficiency while retaining programmability,” in *Symp. on High-Performance Computer Architecture (HPCA)*, pp. 27–39, 2016.

[9] M. Khazraee, L. Zhang, L. Vega, and M. B. Taylor, “Moonwalk: NRE Optimization in ASIC Clouds,” in *Intl. Conf. on Arch. Support for Programming Languages & Operating Systems (ASPLOS)*, pp. 511–526, ACM, 2017.

[10] G. Venkatesh, J. Sampson, N. Goulding, S. Garcia, V. Bryksin, J. Lugo-Martinez, *et al.*, “Conservation cores: Reducing the energy of mature computations,” in *Intl. Conf. on Arch. Support for Programming Languages & Operating Systems (ASPLOS)*, pp. 205–218, ACM, 2010.

[11] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, “Dark silicon and the end of multicore scaling,” in *Intl. Symp. on Computer Architecture (ISCA)*, pp. 365–376, ACM, 2011.

[12] Y. S. Shao, B. Reagen, G.-Y. Wei, and D. Brooks, “Aladdin: A Pre-RTL, Power-performance Accelerator Simulator Enabling Large Design Space Exploration of Customized Architectures,” in *Intl. Symp. on Computer Architecture (ISCA)*, pp. 97–108, IEEE Press, 2014.

[13] B. Reagen, R. Adolf, Y. S. Shao, G. Y. Wei, and D. Brooks, “Machsuite: Benchmarks for accelerator design and customized architectures,” in *IEEE Intl. Symp. on Workload Characterization (IISWC)*, pp. 110–119, 2014.

[14] A. Danalis, G. Marin, C. McCurdy, J. S. Meredith, P. C. Roth, K. Spafford, *et al.*, “The Scalable Heterogeneous Computing (SHOC) Benchmark Suite,” in *Workshop on General-Purpose Computation on Graphics Processing Units (GPGPU)*, pp. 63–74, ACM, 2010.

[15] S. Thomas, C. Gohkale, E. Tanuwidjaja, T. Chong, D. Lau, S. Garcia, and M. B. Taylor, “Cortexsuite: A synthetic brain benchmark suite,” in *IEEE Intl. Symp. on Workload Characterization (IISWC)*, pp. 76–79, 2014.

[16] T. Nowatzki, V. Gangadhar, N. Ardalani, and K. Sankaralingam, “Stream-dataflow acceleration,” in *Intl. Symp. on Computer Architecture (ISCA)*, pp. 416–429, ACM, 2017.

[17] R. Adolf, S. Rama, B. Reagen, G.-Y. Wei, and D. Brooks, “Fathom: Reference workloads for modern deep learning methods,” in *IEEE Intl. Symp. on Workload Characterization (IISWC)*, pp. 1–10, IEEE, 2016.

[18] “MLPerf: A benchmark suite for machine learning.” https://mlperf.org.

[19] A. Danowitz, K. Kelley, J. Mao, J. P. Stevenson, and M. Horowitz, “CPU DB: Recording Microprocessor History,” *Queue*, vol. 10, no. 4, pp. 10:10–10:27, 2012.

[20] A. Stillmaker and B. Baas, “Scaling equations for the accurate prediction of CMOS device performance from 180 nm to 7 nm,” *Integration, the VLSI Journal*, vol. 58, pp. 74–81, 2017.

[21] “International Technology Roadmap for Semiconductors (ITRS).” http://public.itrs.net, 2015 executive summary, 2015.

[22] “International Roadmap for Devices and Systems (IRDS) 2017 Edition.” https://irds.ieee.org/roadmap-2017, 2017.

[23] TechPowerUp, “CPU Database.” https://www.techpowerup.com/cpudb. [Online; accessed 09-July-2018].

[24] TechPowerUp, “GPU Database.” https://www.techpowerup.com/gpudb. [Online; accessed 09-July-2018].

[25] R. Hameed, W. Qadeer, M. Wachs, O. Azizi, A. Solomatnikov, B. C. Lee, and C. Kozyrakis, “Understanding sources of inefficiency in general-purpose chips,” in *Intl. Symp. on Computer Architecture (ISCA)*, pp. 37–47, ACM, 2010.

[26] S. Bell, J. Pu, J. Hegarty, and M. Horowitz, “Compiling algorithms for heterogeneous systems,” *Synthesis Lectures on Computer Architecture*, vol. 13, no. 1, pp. 1–105, 2018.

[27] C. C. Lin, J. I. Guo, H. C. Chang, Y. C. Yang, J. W. Chen, M. C. Tsai, and J. S. Wang, “A 160kgate 4.5kb skram h.264 video decoder for hdtv applications,” in *Intl. Solid-State Circuits Conf. (ISSCC)*, pp. 1596–1605, 2006.

[28] C. D. Chien, C. C. Lin, Y. H. Shih, H. C. Chen, C. J. Huang, C. Y. Yu, C. L. Chen, *et al.*, “A 252kgate/71mw multi-standard multi-channel video decoder for high definition video applications,” in *Intl. Solid-State Circuits Conf. (ISSCC)*, pp. 282–603, 2007.

[29] D. Zhou, Z. You, J. Zhu, J. Kong, Y. Hong, X. Chen, X. He, *et al.*, “A 1080p@60fps multi-standard video decoder chip designed for power and cost efficiency in a system perspective,” in *2009 Symposium on VLSI Circuits*, pp. 262–263, 2009.

[30] T. D. Chuang, P. K. Tsung, P. C. Lin, L. M. Chang, T. C. Ma, Y. H. Chen, and L. G. Chen, “A 59.5mW scalable/multiview video decoder chip for Quad/3D Full HDTV and video streaming applications,” in *Intl. Solid-State Circuits Conf. (ISSCC)*, pp. 330–331, 2010.

[31] D. Zhou, J. Zhou, X. He, J. Zhu, J. Kong, P. Liu, and S. Goto, “A 530 Mpixels/s 4096x2160@60fps H.264/AVC High Profile Video Decoder Chip,” *IEEE Journal of Solid-State Circuits*, vol. 46, no. 4, pp. 777–788, 2011.

[32] P. K. Tsung, P. C. Lin, K. Y. Chen, T. D. Chuang, H. J. Yang, S. Y. Chien, *et al.*, “A 216fps 4096x2160p 3DTV set-top box SoC for free-viewpoint 3DTV applications,” in *Intl. Solid-State Circuits Conf. (ISSCC)*, pp. 124–126, 2011.

[33] D. Zhou, J. Zhou, J. Zhu, P. Liu, and S. Goto, “A 2Gpixel/s H.264/AVC HP/MVC video decoder chip for Super Hi-Vision and 3DTV/FTV applications,” in *Intl. Solid-State Circuits Conf. (ISSCC)*, pp. 224–226, 2012.

[34] M. Tikekar, C. T. Huang, C. Juvekar, V. Sze, and A. P. Chandrakasan, “A 249-Mpixel/s HEVC Video-Decoder Chip for 4K Ultra-HD Applications,” *Intl. Solid-State Circuits Conf. (ISSCC)*, vol. 49, no. 1, pp. 61–72, 2014.

[35] C. C. Ju, T. M. Liu, Y. C. Chang, C. M. Wang, H. M. Lin, C. Y. Cheng, *et al.*, “A 0.2nJ/pixel 4K 60fps Main-10 HEVC decoder with multi-format capabilities for UHD-TV applications,” in *European Solid State Circuits Conf. (ESSCIRC)*, pp. 195–198, 2014.

[36] C. C. Ju, T. M. Liu, K. B. Lee, Y. C. Chang, H. L. Chou, C. M. Wang, *et al.*, “A 0.5 nJ/Pixel 4 K H.265/HEVC Codec LSI for Multi-Format Smartphone Applications,” *IEEE Journal of Solid-State Circuits*, vol. 51, no. 1, pp. 56–67, 2016.

[37] C. C. Ju, T. M. Liu, Y. C. Chang, C. M. Wang, C. Y. Cheng, H. M. Lin, *et al.*, “A 2.6mm2 0.19nJ/pixel VP9 and multi-standard decoder LSI for Android 4K TV applications,” in *European Solid State Circuits Conf. (ESSCIRC)*, pp. 109–112, 2016.

[38] D. Zhou, S. Wang, H. Sun, J. Zhou, J. Zhu, Y. Zhao, *et al.*, “An 8K H.265/HEVC Video Decoder Chip With a New System Pipeline Design,” *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 113–126, 2017.

[39] T. Spangler, “Netflix bandwidth usage climbs to nearly 37% of internet traffic at peak hours,” 2015.

[40] B. Brouwer, “Youtube now gets over 400 hours of content uploaded every minute,” *URL: http://www.tubefilter.com/2015/07/26/youtube-400-hours-*

content-every-minute/, Abruf am, vol. 15, p. 2016, 2015.

[41] AnandTech, "Gpu benchmarks." https://www.anandtech.com/bench/GPU17. [Online; accessed 11-July-2018].

[42] NVIDIA, "NVIDIA CUDA Toolkit Release Notes." https://docs.nvidia.com/cuda/cuda-toolkit-release-notes/index.html.

[43] N. Suda, V. Chandra, G. Dasika, A. Mohanty, Y. Ma, S. Vrudhula, *et al.*, "Throughput-Optimized OpenCL-based FPGA Accelerator for Large-Scale Convolutional Neural Networks," in *Intl. Symp. on Field-Programmable Gate Arrays (FPGA)*, pp. 16–25, ACM, 2016.

[44] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, *et al.*, "Going Deeper with Embedded FPGA Platform for Convolutional Neural Network," in *Intl. Symp. on Field-Programmable Gate Arrays (FPGA)*, FPGA '16, pp. 26–35, ACM, 2016.

[45] Y. Ma, N. Suda, Y. Cao, J. s. Seo, and S. Vrudhula, "Scalable and modularized RTL compilation of Convolutional Neural Networks onto FPGA," in *Intl. Symp. on Field Programmable Logic and Applications (FPL)*, pp. 1–8, 2016.

[46] J. Zhang and J. Li, "Improving the Performance of OpenCL-based FPGA Accelerator for Convolutional Neural Network," in *Intl. Symp. on Field-Programmable Gate Arrays (FPGA)*, pp. 25–34, ACM, 2017.

[47] U. Aydonat, S. O'Connell, D. Capalija, A. C. Ling, and G. R. Chiu, "An OpenCL Deep Learning Accelerator on Arria 10," in *Intl. Symp. on Field-Programmable Gate Arrays (FPGA)*, pp. 55–64, ACM, 2017.

[48] J. Shen, Y. Huang, Z. Wang, Y. Qiao, M. Wen, and C. Zhang, "Towards a Uniform Template-based Architecture for Accelerating 2D and 3D CNNs on FPGA," in *Intl. Symp. on Field-Programmable Gate Arrays (FPGA)*, pp. 97–106, ACM, 2018.

[49] H. Zeng, R. Chen, C. Zhang, and V. Prasanna, "A Framework for Generating High Throughput CNN Implementations on FPGAs," in *Intl. Symp. on Field-Programmable Gate Arrays (FPGA)*, pp. 117–126, ACM, 2018.

[50] V. J. Reddi, H. Yoon, and A. Knies, "Two billion devices and counting," *IEEE Micro*, vol. 38, no. 1, pp. 6–21, 2018.

[51] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[52] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Intl. Conf. on Neural Information Processing Systems (NIPS)*, pp. 1097–1105, 2012.

[53] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014.

[54] S. Winograd, *Arithmetic Complexity of Computations*. Society for Industrial and Applied Mathematics, 1980.

[55] A. Lavin, "Fast algorithms for convolutional neural networks," *CoRR*, vol. abs/1509.09308, 2015.

[56] H. Sharma, J. Park, D. Mahajan, E. Amaro, J. K. Kim, C. Shao, A. Mishra, and H. Esmaeilzadeh, "From high-level deep neural models to FPGAs," in *Intl. Symp. on Microarchitecture (MICRO)*, pp. 1–12, 2016.

[57] Digiconomist, "Bitcoin energy consumption index." https://digiconomist.net/bitcoin-energy-consumption. [Online; accessed 06-March-2018].

[58] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[59] M. B. Taylor, "The evolution of bitcoin hardware," *Computer*, vol. 50, no. 9, pp. 58–66, 2017.

[60] Bitcoin Wiki, "List of Bitcoin mining ASICs." https://en.bitcoin.it/wiki/List_of_Bitcoin_mining_ASICs.

[61] Bitcoin Wiki, "Mining Hardware Comparison." https://en.bitcoin.it/wiki/Mining_hardware_comparison.

[62] Bitcoin Wiki, "Non-specialized Hardware Comparison." https://en.bitcoin.it/wiki/Non-specialized_hardware_comparison.

[63] Bitcointalk, "Bitcoin Forum." https://bitcointalk.org.

[64] T. Hanke, "Asicboost-a speedup for bitcoin mining," *arXiv preprint arXiv:1604.00575*, 2016.

[65] C. González-Álvarez, J. B. Sartor, C. Álvarez, D. Jiménez-González, and L. Eeckhout, "Accelerating an application domain with specialized functional units," *ACM Trans. Archit. Code Optim.*, vol. 10, pp. 47:1–47:25, Dec. 2013.

[66] V. Govindaraju, C.-H. Ho, and K. Sankaralingam, "Dynamically specialized datapaths for energy efficient computing," in *Symp. on High-Performance Computer Architecture (HPCA)*, pp. 503–514, IEEE Computer Society, 2011.

[67] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC benchmark suite: characterization and architectural implications," in *Intl. Conf. on Parallel Arch. and Compilation Techniques (PACT)*, 2008.

[68] S. Galal and M. Horowitz, "Energy-efficient floating-point unit design," *IEEE Transactions on Computers*, vol. 60, no. 7, pp. 913–922, 2011.

[69] Intel, "Intel stratix 10 high-performance design handbook." https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/stratix-10/s10_hp_hb.pdf, 2018.

[70] Xilinx, "Versal: The First Adaptive Compute Acceleration Platform (ACAP)." https://www.xilinx.com/support/documentation/white_papers/wp505-versal-acap.pdf, 2018.

[71] W. Qadeer, R. Hameed, O. Shacham, P. Venkatesan, C. Kozyrakis, and M. A. Horowitz, "Convolution engine: Balancing efficiency & flexibility in specialized computing," in *Intl. Symp. on Computer Architecture (ISCA)*, pp. 24–35, ACM, 2013.

[72] Xilinx, "Device Reliability Report." https://www.xilinx.com/support/documentation/user_guides/ug116.pdf.

[73] D. W. Wall, "Limits of instruction-level parallelism," in *Intl. Conf. on Arch. Support for Programming Languages & Operating Systems (ASPLOS)*, pp. 176–188, ACM, 1991.

[74] M. B. Taylor, "Is dark silicon useful?: Harnessing the four horsemen of the coming dark silicon apocalypse," in *Annual Design Automation Conference(DAC)*, pp. 1131–1136, ACM, 2012.

[75] M. Halpern, Y. Zhu, and V. J. Reddi, "Mobile CPU's rise to power: Quantifying the impact of generational mobile CPU design trends on performance, energy, and user satisfaction," in *Symp. on High-Performance Computer Architecture (HPCA)*, pp. 64–76, 2016.

[76] M. Gao, C. Delimitrou, D. Niu, K. T. Malladi, H. Zheng, B. Brennan, *et al.*, "DRAF: A Low-power DRAM-based Reconfigurable Acceleration Fabric," in *Intl. Symp. on Computer Architecture (ISCA)*, pp. 506–518, IEEE Press, 2016.

[77] T. Nowatzki and K. Sankaralingam, "Analyzing behavior specialized acceleration," in *Intl. Conf. on Arch. Support for Programming Languages & Operating Systems (ASPLOS)*, pp. 697–711, ACM, 2016.

[78] S. Borkar and A. A. Chien, "The future of microprocessors," *Commun. ACM*, vol. 54, no. 5, pp. 67–77, 2011.

[79] A. Arunkumar, E. Bolotin, B. Cho, U. Milic, E. Ebrahimi, O. Villa, *et al.*, "MCM-GPU: Multi-Chip-Module GPUs for Continued Performance Scalability," in *Intl. Symp. on Computer Architecture (ISCA)*, pp. 320–332, ACM, 2017.

[80] A. Fuchs and D. Wentzlaff, "Scaling datacenter accelerators with compute-reuse architectures," in *Intl. Symp. on Computer Architecture (ISCA)*, pp. 353–366, 2018.