Theme Article

Inside Project Brainwave's Cloud-Scale, Real-Time AI Processor

Jeremy Fowers, Kalin Ovtcharov, Michael K. Papamichael, Todd Massengill, Ming Liu, Daniel Lo, Shlomi Alkalay, Michael Haselman, Logan Adams, Mahdi Ghandi, Stephen Heil, Prerak Patel, Adam Sapek, Gabriel Weisz, Lisa Woods, Sitaram Lanka, Steven K. Reinhardt, Adrian M. Caulfield, Eric S. Chung, and Doug Burger
Microsoft

***Abstract*—Growing computational demands from deep neural networks (DNNs), coupled with diminishing returns from general-purpose architectures, have led to a proliferation of Neural Processing Units (NPUs). This paper describes the Project Brainwave NPU (BW-NPU), a parameterized microarchitecture specialized at synthesis time for convolutional and recurrent DNN workloads. The BW-NPU deployed on an Intel Stratix 10 280 FPGA achieves sustained performance of 35 teraflops at a batch size of 1 on a large recurrent neural network (RNN).**

The diminishing returns of general-purpose architectures coinciding with the escalating computational requirements for state-of-the-art deep neural networks (DNNs) has led to a Cambrian explosion of neural processing units (NPUs).[1–4] NPUs that target interactive services must satisfy two criteria: 1) ability to execute DNN models at low latency, high throughput, and high efficiency with no input batching; and 2) flexibility to accommodate evolving state-of-the-art DNN models such as recurrent neural networks (RNNs), convolutional neural networks (CNNs), MLPs, and attention models. These requirements are not well served by existing throughput-oriented processors such as GPUs that batch inputs to achieve high throughput at the expense of high latency.

Digital Object Identifier 10.1109/MM.2019.2910506
Date of publication 11 April 2019; date of current version 8 May 2019.

Published by the IEEE Computer Society

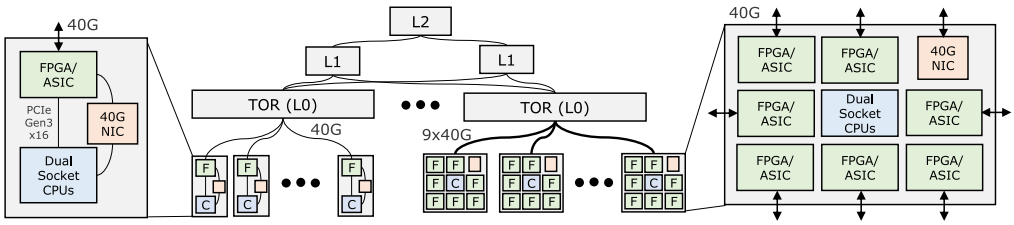

Figure 1. Brainwave system at cloud scale. From left to right, servers with bump-in-the-wire accelerators, accelerators connected directly to the hyperscale datacenter network, an accelerator appliance.

Project Brainwave is a datacenter-scale production system for hardware-accelerated DNN inference at Microsoft. The Brainwave Neural Processing Unit (BW-NPU) architecture performs at over 35 teraflops at a batch size of 1 on an Intel Stratix 10 280 FPGA device—more than an order of magnitude improvement in latency and throughput over modern GPUs on large recurrent neural network (RNN) models.[5] This paper covers the following inner workings of the Project Brainwave system.

- *Architecture:* A single-threaded, dynamically scheduled, mega-SIMD Instruction Set Architecture (ISA) capable of dispatching over seven million operations per instruction. The simple-to-target BW-NPU architecture stands in contrast to other accelerators with more complex programming models such as multi-threaded GPUs or fine-grained statically scheduled MIMD.[2]
- *Memory System:* To achieve low latency, DNN model weights are pinned into distributed on-chip SRAM memories capable of delivering tens of terabytes per seconds of memory bandwidth.
- *Microarchitecture:* Hierarchical decode and dispatch enables scaling of up to 96 000 parallel operations per cycle with a single matrix-vector multiply instruction on a Stratix 10 280 FPGA. The microarchitecture also exploits both intra- and inter-vector-level parallelism as well as instruction chaining to reduce pipeline bubbles, increasing hardware utilization, and reducing latency.
- *System:* The NPU is deployed on a hyperscale datacenter system in which accelerator cards have direct access to the datacenter network and are placed in-line between the server NIC and top-of-rack (TOR) switches. This infrastructure enables large, partitionable DNN models to be spatially distributed across multiple accelerators and communicate point-to-point at very low latencies.
- *Synthesis Specialization:* When targeting FPGAs, microarchitectural and datapath synthesis specialization enables the BW-NPU architecture to adapt quickly to new and emerging DNN models. A narrow precision data type is used to quantize DNN models with minimal impact to classification accuracy on production and state-of-the-art models. The exploitation of low precision data types on FPGA enables BW-NPU to achieve competitive performance and efficiency with hardened NPUs leveraging standard datatypes (e.g., 16- or 32-bit floating point).

BACKGROUND

The Brainwave system integrates with an existing hyperscale cloud infrastructure that runs production services with real-time AI requirements.[6] This section gives background on the salient features of the target datacenter architecture and the main layers of the DNN serving stack.

Hyperscale Datacenter Acceleration Architecture

Figure 1 illustrates the components of a hyperscale datacenter. The acceleration architecture we describe is in large scale, world-wide production. Every standard dual-socket server hosts one or more PCIe-attached accelerator cards that contain FPGAs and/or ASICs. The accelerator cards have direct access to the datacenter network and are placed in-line between the server NIC and the TOR switches. Using an on-chip RDMA-like lossless protocol, the accelerators can communicate point-to-point directly at low latency to any of the hundreds of thousands

of servers located in the same datacenter. The datacenter architecture shown in Figure 1 enables accelerators to be logically disaggregated and pooled into instances of hardware microservices with no software in the loop. Once initialized and registered with a distributed resource manager, a given hardware microservice is published to subscribing CPUs in the system and accessed directly through an IP address.

The system architecture presented in Figure 1 fundamentally influences the way in which accelerators are managed and architected at cloud scale. The CPU and FPGA resources devoted to a particular acceleration scenario can be scaled independently, avoiding stranded capacity and improving overall datacenter utilization. Large partitionable problems can be spatially distributed across multiple accelerators. For example, we can split bidirectional RNNs across two independent FPGAs, with the server invoking the forward and backward RNN FPGAs separately and concatenating their outputs.

DNN Acceleration Platform

The DNN acceleration platform consists of three components: 1) a toolflow that transforms pretrained DNN model checkpoints into software and accelerator executables; 2) a federated runtime that orchestrates model execution between CPUs and distributed hardware microservices; and 3) the programmable BW-NPU instantiated on FPGAs.

In the initial phases of the toolflow, a pretrained DNN model is exported from a DNN framework (e.g., TensorFlow[7]) into Brainwave's graph intermediate representation (GIR). The GIR undergoes a series of optimizations and transformations based on target constraints of the backend system, including the target number of accelerators and the available on-chip memory per accelerator. In latency-sensitive real-time scenarios, the toolflow can partition large graphs that exceed the capacity of a single FPGA into subgraphs whose parameters can be pinned individually into accelerators' on-chip memory. This partitioning avoids the memory capacity/bandwidth tradeoff that often thwarts the deployment of large RNNs and MLPs. Operations that are not supported by or cannot be profitably accelerated on the BW-NPU are grouped into subgraphs for execution on CPU cores.

Once the partitioning is complete, the FPGA and CPU subgraphs are compiled to BW-NPU and CPU ISA binaries, respectively. The backend executables are then packaged and deployed to a federated CPU runtime in the cloud that executes both the CPU and accelerator subgraphs initiated through calls to a remote hardware microservice.

ARCHITECTURE

Many commercial NPUs target acceleration of matrix–matrix multiplication primitives, which enable high data reuse and reduced memory bandwidth requirements. However, effectively utilizing the hardware on models dominated by fully connected layers (e.g., LSTMs) requires batching of inputs for hardware efficiency, which can increase the effective end-to-end latency of individual requests. In contrast, NPUs that target matrix–vector multiplication routines are better suited for single-request (no batch), low-latency serving. Although matrix-vector multiplication is more challenging to design for than higher dimensional tensor operators, the reduced dependence on batching is instrumental to achieving lower latencies.

The BW-NPU architecture optimizes for matrix-vector operations and is designed around three goals: 1) provide a simple programming model that can be easily targeted by programmers and backend compilers while supporting dynamic dataflow (e.g., time-dependent LSTMs); 2) encode the necessary information for an underlying microarchitecture to effectively exploit the available parallelism of the DNN model; and 3) provide enough flexibility to execute a wide range of DNN models including LSTMs, GRUs, 1-D (text) CNNs, 2-D (image) CNNs, word/character embeddings, dense MLPs, and attention layers. To achieve these goals, the BW-NPU leverages a single-threaded mega-SIMD ISA composed of instructions that can operate on vectors and matrices of a fixed "native" size that can be statically optimized at compile time.

Specialized instructions, data types, and memory abstractions are also exposed directly in the ISA. Subgraphs of a large DNN model can be encoded through atomic instruction chains (without named storage between instructions) that efficiently capture explicit communication between graph edges, simplifying software

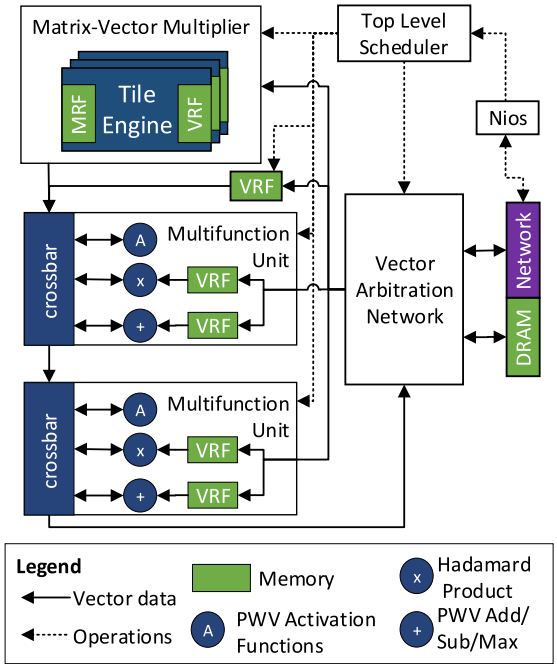

Figure 2. Microarchitecture overview.

development, and reducing complexity in hardware. Chaining of operations also enables the microarchitecture to 1) exploit substantial pipeline parallelism without complex hardware dependence checking; 2) improve instruction density; 3) reduce register allocation pressure; and 4) resolve critical serial dependencies at low latency. In addition, model weights are pinned to the on-chip memory banks that deliver up to tens of terabytes per second of bandwidth.

MICROARCHITECTURE

The BW-NPU microarchitecture, as shown in Figure 2, is designed to maximize vector-level parallelism (VLP) of a single DNN request at high hardware efficiency. The BW-NPU targets operations that begin with a matrix-vector multiplication followed by less compute-intensive pointwise vector–vector (PWV) operations. As a result, it is important to maximize hardware utilization of the matrix–vector multiplier (MVM), the primary workhorse of the BW-NPU, to obtain good performance. This is achieved by mapping the set of data-dependent instructions (i.e., instruction chains) onto a continuous spatial pipeline of functional units with the MVM at the head.

Matrix–Vector Multiplier

The MVM occupies the majority of the circuit and can be specialized to support both RNN and CNN workloads as shown in Figure 3. Both configurations instantiate multiple matrix-vector tile engines (TEs) that operate on submatrix blocks defined by the hardware native size. Each TE consists of an array of dot product engines (DPE) and each DPE is responsible for multiplying a vector by a single native row. In order to minimize stalls (i.e., maximize read data bandwidth), a DPE is directly connected to dedicated on-chip memories. Internally, a DPE consists of a number of parallel multipliers followed by an add-reduction tree and accumulator that perform an intravector multiply-add operation.

Convolutional Neural Networks

As shown in Figure 3(b), CNNs are supported efficiently by lowering convolutions into MVMs and modifying the BW-NPU datapaths and memory structures to better exploit reuse of weights and activations. Small per-TE vector register files (VRFs) are replaced with a single, large

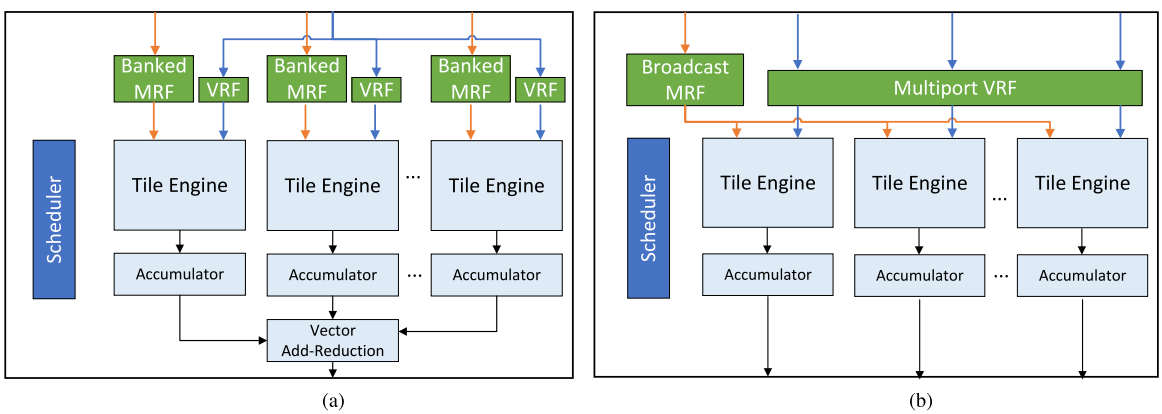

Figure 3. MVM microarchitecture targeting RNN and CNN workloads. (a) RNN-specialized (b) CNN-specialized.

multiported VRF shared across TEs. The per-TE matrix register file (MRF) banks are replaced with a single shared MRF, enabling weights to be shared and broadcasted to all TEs. Lowering of CNNs into MVMs produces many small, independent MVMs, as opposed to RNNs which tend to be comprised of a few large MVMs. The CNN BW-NPU computes one output activation per TE in parallel by removing the post-TE add-reduction tree and providing additional data lanes in the multifunction unit (MFU) to match the increased bandwidth. Finally, the first layer of many CNNs (e.g., ResNet-50, DenseNet-121, VGG-16, etc.) have as few as three channels, which underutilizes the high native-dimension MVM datapath. BW-NPU addresses these first layers with a specialized circuit that unrolls each input window into a single vector to improve the MVM efficiency.

Multifunction Units

The output from the MVM is followed by a series of vector MFUs. The MFUs support vector–vector operations such as Hadamard product and pointwise addition as well as unary vector activation functions like ReLU, sigmoid, and tanh. Dedicated VRFs associated with the add/subtract and multiply function units provide the secondary operands needed for those operations.

In addition, a vector-based network-on-chip (NoC) manages data movement across various memory components such as DRAM, I/O (network and PCIe), VRFs, and MRFs. Control is managed by a scalar processor that issues instructions to a global scheduler that dispatches thousands of individual commands to control many spatially distributed compute resources.

SYNTHESIS SPECIALIZATION

The BW-NPU microarchitecture can be viewed as a fully parameterizable processor family that can be customized to specific models and underlying implementation substrates for efficiency. While hardened NPUs can operate at high clock frequencies, DNNs must be adapted to match the fixed parameters of the microarchitecture to run efficiently. In contrast, "soft" NPUs targeting FPGAs can be synthesis-specialized to state-of-the-art DNN models.

Datapath Specialization: The BW-NPU architecture exposes several parameters that can be used for specializing a microarchitecture instance to specific models: 1) aligning the native vector dimension to parameters of the model to minimize padding and waste during model evaluation; 2) increasing lane widths to drive up intra-row-level parallelism; and 3) increasing matrix multiply tiles to exploit submatrix parallelism for large models.

Narrow Precision Data Types: Numerical quantizations improve the memory bandwidth and storage requirements, and increase the number of multiply accumulate units. A narrow precision block floating point format (BFP) is used to quantize data yet still retain sufficient degree of dynamic range.[8] Using variations of BFP, mantissas are trimmed to as low as 2–5 bits with negligible impact on accuracy (within 1%–2% of baseline) with just a few epochs of fine-tuning on production state-of-the-art DNN models and large ImageNet models (e.g., ResNet-50).

EVALUATION

We measure the performance of BW-NPU using DeepBench,[5] a set of microbenchmarks containing layers from popular DNN models such as DeepSpeech[9] on a Stratix 10 280 FPGA device. Table 1 summarizes the raw effective TFLOPS and execution latency for each benchmark. The BW-NPU can run *all* DeepBench layers in under 4 ms at batch 1, reaching up to *35.9 effective TFLOPS* for a large GRU over hundreds of timesteps. A batch size of 1 is used to achieve the lowest possible latency.

Figure 4 shows the utilization, which is the percentage of the peak TFLOPS reached for each layer. At batch size of 1, the BW-NPU reaches 23% to 75% of peak FLOPS for medium to large LSTM/GRUs (>1500 dimension). The results show a 4–23× improvement over the utilization of a Titan Xp, a high-end GPU of comparable process geometry. The BW-NPU is able to fully expose on-chip RAM bandwidth, pipeline dependent RNN vectors, and exploit all degrees of matrix-vector parallelism to keep its compute units busy.

In all measured instances of the BW-NPU, peak card-level power consumption never exceeded 125 W. Given this measurement, the power efficiency ceiling of the BW-NPU is calculated to be *326 GFLOPS/W* for large models at high device utilization given a pessimistic estimate based on TDP.

Table 1. DeepBench RNN inference performance results.

Benchmark	Device	Latency (ms)	TFLOPS	% Utilization
GRU $h = 2816$ $t = 750$	BW-NPU	1.987	35.92	74.8
	Titan Xp	178.60	0.40	3.3
GRU $h = 1536$ $t = 375$	BW-NPU	0.951	11.17	23.3
	Titan Xp	31.73	0.33	2.8
GRU $h = 512$ $t = 1$	BW-NPU	0.013	0.25	0.5
	Titan Xp	0.06	0.05	0.4
LSTM $h = 2048$ $t = 25$	BW-NPU	0.074	22.62	47.1
	Titan Xp	5.27	0.32	2.7
LSTM $h = 1536$ $t = 50$	BW-NPU	0.145	13.01	27.1
	Titan Xp	6.20	0.30	2.5
LSTM $h = 256$ $t = 150$	BW-NPU	0.425	0.37	0.8
	Titan Xp	1.99	0.08	0.7

CNN Performance Analysis

The BW-NPU architecture can also accelerate and serve large CNN models at low latency. In this section, we report on results of a BW-NPU variant specialized for CNNs running a production image-based featurizer based on ResNet-50.[10] The topology and computational requirements are nearly identical to the originally reported model except for the final dense layer, which is replaced by scenario-specific classifiers (e.g., decision trees in a Bing ranking pipeline) that run on CPU.

Table 2 compares the latency and throughput to run the ResNet-50-based featurizer standalone on a BW-NPU hosted on an Arria 10 1150 (TSMC 20-nm FPGA) against a high-end inference-optimized Nvidia P40 GPU (TSMC 16 nm). Our measurements on real hardware include the latency to compute a single request as well as the transfer time over PCI express between host CPU and accelerator.

At batch size 1, the high-end P40 using INT8 precision achieves 461 inferences/s (IPS), while the BW-NPU on Arria 10 achieves 676 IPS. On an

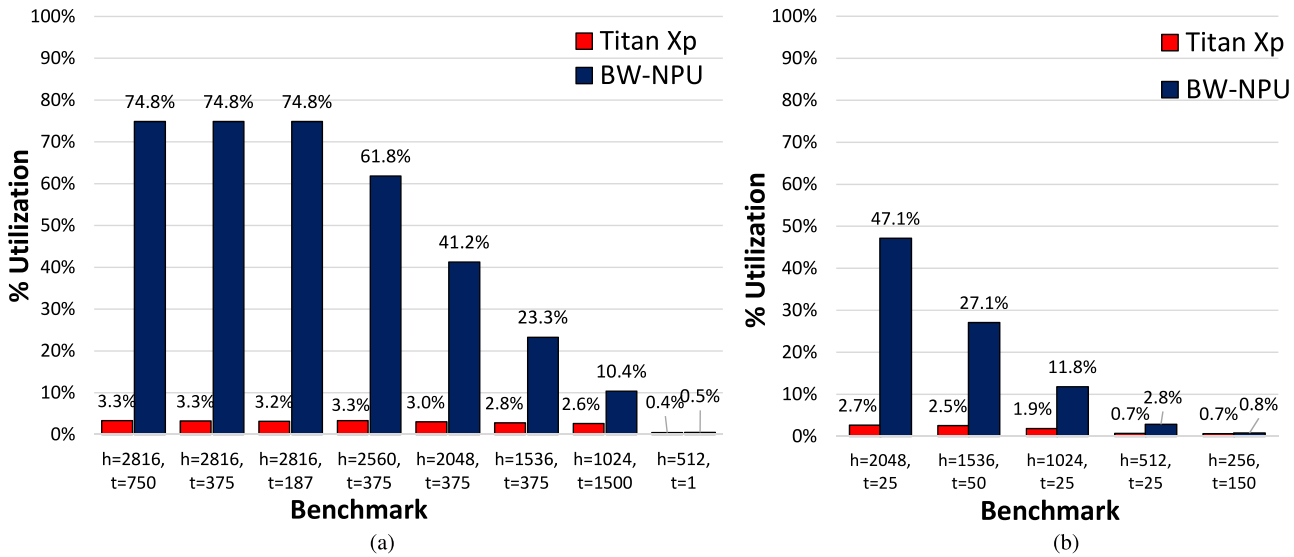

Figure 4. Hardware utilization across DeepBench RNN inference experiments. h = hidden dimension, t = time steps. (a) GRU. (b) LSTM.

Table 2. BW-NPU on Arria 10 achieves competitive throughput and latency to an Nvidia P40 GPU at batch size 1 on a resnet-50-based image featurizer.

	Nvidia P40	BW_CNN_A10
Technology node	16-nm TSMC	20-nm TSMC
Framework	TF 1.5 + TensorRT 4	TF + BW
Precision	INT8	BFP (1s.5e.5m)
IPS (batch 1)	461	676
Latency (batch 1)	2.17 ms	1.48 ms

unloaded system, the BW-NPU serves a single instance of the model in 1.48 ms, while the P40 achieves 2.17 ms. The P40 achieves higher throughput than the Arria 10 at large batch sizes; for example, at a batch size of 16, the P40 attains 2270 IPS; however, latency increases to 7 ms per batch, which does not include the time needed to form inputs from a batching queue. These results show that the BW-NPU is an effective architecture for single-batch, low-latency serving—competitive with high-end, modern GPUs on compute-intensive CNNs and orders of magnitude faster on RNNs.

CONCLUSION

Interactive services require NPUs that can efficiently serve a diverse set of models at batch size of 1, while preserving the flexibility to accommodate a rapidly changing DNN landscape during the lifetime of the hardware device.

By exploiting mega-SIMD parallelism and leveraging the aggregate bandwidth of thousands of on-chip memories, the BW-NPU serves models at low latencies with no batching. As a result, the BW-NPU removes the adversarial tradeoff between throughput and latency that often occurs with throughput-oriented architectures such as GPUs. This benefit is preserved when scaling to larger DNN models by spatially distributing models across multiple devices connected point-to-point by a hyperscale datacenter network.

The machine learning industry is relatively nascent and as a result is constantly evolving, with new models and applications being deployed at a very fast pace. Therefore, the speed of deployment of hardware accelerators with new capabilities to match the requirements of a growing number of applications must be matched. When targeting FPGAs, the BW-NPU leverages synthesis specialization to enable a greater level of agility when deploying new capabilities. This "software-defined" approach to deployment of hardware accelerators will become even more important as the landscape of machine learning continues to evolve. Synthesis specialization exploits the configurable routing, logic, memory, and DSPs of FPGAs and allows scaling of the BW-NPU onto larger devices. Although FPGAs tend to run at lower clock speeds, specialized architectures, and codesign of algorithms and hardware (e.g., narrow precision) can make up for that difference.

The limits of exploitable VLP in a single thread of execution are still largely unknown, and it is unclear whether this model will continue to scale to the increased area afforded by the few remaining silicon process nodes. However, we have shown that the BW-NPU architecture does scale well across three generations of FPGAs of vastly different sizes—from Stratix V (172 kALMs), through Arria 10 (427 kALMs), and later to Stratix 10 (933 kALMs). The Project Brainwave system can be publicly accessed and currently supports the following models as featurizers: ResNet-50, ResNet-152, DenseNet-121, and VGG-16, with many more models in the pipeline to launch.[11]

REFERENCES

1. N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proc. 44th Annu. Int. Symp. Comput. Archit.*, 2017, pp. 1–12.
2. L. Gwennap, "Graphcore makes big AI splash," Microprocessor Rep., The Linley Group, Mountain View, CA, USA, Sep. 2018.
3. D. Moloney, "Embedded deep neural networks: The cost of everything and the value of nothing," in *Proc. IEEE Hot Chips 28 Symp.*, Aug. 2016, pp. 1–20.
4. K. Guo *et al.*, "From model to FPGA: Software-hardware co-design for efficient neural network acceleration," in *Proc. IEEE Hot Chips 28 Symp.*, Aug. 2016, pp. 1–27.
5. S. Narang and G. Diamos, "Baidu DeepBench," 2017. [Online]. Available: https://github.com/baidu-research/DeepBench

6. A. Putnam *et al.*, "A reconfigurable fabric for accelerating large-scale datacenter services," in *Proc. 41st Annu. Int. Symp. Comput. Archit.*, 2014, pp. 13–24.
7. M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Operating Syst. De. Implementation*, 2016, pp. 265–283.
8. J. H. Wilkinson, *Rounding Errors in Algebraic Processes*. 1st ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 1963.
9. A. Y. Hannun *et al.*, "Deep Speech: Scaling up end-to-end speech recognition," 2014, *arXiv:1412.5567*.
10. K. He *et al.*, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, Jun. 2016, pp. 770–778.
11. AzureML, "Microsoft azure machine learning hardware accelerated models powered by project brainwave," 2018. [Online]. Available: https://github.com/Azure/aml-real-time-ai

Jeremy Fowers is the lead architect of the Project Brainwave at Microsoft Research. His research is focused on applying FPGAs as best-in-class accelerators in a post Moore's Law World. He has a PhD from the University of Florida. Contact him at jfowers@microsoft.com.

Kalin Ovtcharov is a senior research hardware engineer at Microsoft, working in the AI and Advanced Architectures Group. His research focus is on the design of high-performance machine learning accelerators in the cloud. Prior to Microsoft, he was the Founder of a startup company applying image processing to the automation industry. He has a BS in computer engineering from McMaster University. Contact him at kaovt@microsoft.com.

Michael K. Papamichael is a researcher in the AI and Advanced Architectures Group, Microsoft. His research interests include the broader area of computer architecture with emphasis on hardware acceleration, reconfigurable computing, on-chip interconnects, and methodologies to facilitate hardware specialization. He has a PhD in computer science from Carnegie Mellon University. Contact him at papamix@microsoft.com.

Todd Massengill has been a senior developer on Project Brainwave in the AI and Advanced Architectures Group, Microsoft, since 2014. His research interests include delivering high quality, well-validated hardware acceleration services for cloud computing. He has an MSEE from the University of Washington. Contact him at toddma@microsoft.com.

Ming Liu is a senior research hardware engineer working on Project Brainwave/Catapult in the AI and Advanced Architectures Group, Microsoft, leveraging FPGAs to accelerate machine learning applications in the cloud. His primary research interests include hardware design, machine learning, application-specific accelerators, embedded systems, and computer architecture. Prior to joining Microsoft Research, he was a research assistant at MIT, where he was a core member of the BlueDBM Project. He has an MS in computer science from MIT and a BASc in computer engineering from the University of Toronto. Contact him at mliu@microsoft.com.

Daniel Lo is a senior hardware engineer at Microsoft, where he works on the intersection of machine learning and hardware accelerators. He has a PhD in electrical and computer engineering from Cornell University. Contact him at dlo@microsoft.com.

Shlomi Alkalay is a senior hardware engineer in the Project BrainWave Engineering Group, Microsoft AI + R. Working on Project Catapult since 2015, he has focused on delivering large-scale distributed hardware accelerators to the Microsoft cloud. Prior to joining Microsoft, he had worked in several companies, designing FPGA-based accelerators for the telecom, financial, and cyber-security industries. He has a BS in electrical and computer engineering from the Ben-Gurion University. Contact him at salkalay@microsoft.com.

Michael Haselman is a senior software engineer at AI&R (Bing). He started at Microsoft in 2013, joining the Catapult project. His research interests include FPGAs, computer architecture, and distributed computing. He has a PhD in electrical engineering from the University of Washington, Seattle, under the supervision of Scott Hauck. Contact him at mikehase@microsoft.com.

Logan Adams is a hardware engineer at Bing and working on Project Catapult and Project Brainwave. He has an MS in electrical engineering from the University of Washington, where he researched FPGA applications for the ATLAS experiment in Large Hadron Collider at CERN. He also has a BS in electrical engineering from the University of Portland. Contact him at loadams@microsoft.com.

Mahdi Ghandi is a member of Bing, focused on software and FPGA development for accelerated AI on networks of FPGAs. Prior to working at Microsoft, he was a senior staff engineer at Evertz

Microsystems, architecting video compression solutions for broadcast industry with extensive use of FPGAs. He has a PhD from the University of Essex, having done years of research on video compression and communications. Contact him at maghandi@microsoft.com.

Stephen Heil is a principal program manager in Azure Hardware Systems Group, Microsoft Corporation. In recent years, his focus has been on the development of custom programmable hardware accelerators for use in Microsoft datacenters. His research interests include field programmable gate arrays, application accelerators, and rack-scale system design. He has been involved in various aspects of computer system design and standards while working at Microsoft, Compaq, Panasonic and Unisys. He has a BS in electrical engineering technology and computer science from The College of New Jersey (formerly Trenton State College) and has been a longstanding member of the IEEE and the Association for Computing Machinery (ACM). Contact him at Stephen.Heil@microsoft.com.

Prerak Patel is a senior hardware engineer in Microsoft's AI & Research Group, working to accelerate new machine learning models on Project Brainwave. He has an undergraduate degree and an MS in computer engineering from Carnegie Mellon University. Contact him at prepatel@microsoft.com.

Adam Sapek is a principle software engineer at Microsoft working in AI and Advanced Architectures Group. He has an MS in computer science from Silesian University of Technology, Poland. Contact him at adamsap@microsoft.com.

Gabriel Weisz is a principal hardware engineer at Microsoft. His research interests include finding easier ways to design and use hardware accelerators without sacrificing performance. He has an MS and a PhD in computer science from Carnegie Mellon University and a BS in computer science and electrical engineering from Cornell University. Contact him at gabriel.weisz@microsoft.com.

Lisa Woods is a principal program manager for Project Catapult in the AI and Advanced Architectures Group, Microsoft. She has focused on aligning the many research and engineering groups within the enterprise-wide Project Catapult v-team. She is the team's PM for Project Brainwave, a deep learning platform that leverages FPGAs to accelerate inferencing of deep neural networks (DNNs). Contact her at Lisa.Woods@microsoft.com.

Sitaram Lanka is currently a partner group engineering manager. His research interests include deep learning, deep learning model execution, distributed systems, and search algorithms. He has a PhD in computer science from the University of Pennsylvania. Contact him at slanka@microsoft.com.

Steven K. Reinhardt is a partner hardware engineering manager in the Bing Engineering Team at Microsoft. He has a PhD from the University of Wisconsin-Madison, and is an IEEE Fellow and an ACM Distinguished Scientist. Contact him at stever@gmail.com.

Adrian M. Caulfield is a principal engineering manager at Microsoft. His research interests include systems architecture, data center acceleration, and storage technologies. He has a PhD in computer engineering from the University of California San Diego and a BA from the University of Washington, Seattle. He is a member of the Association for Computing Machinery (ACM). Contact him at acaulfie@microsoft.com.

Eric S. Chung is a principal researcher and manager in the Azure AI & Advanced Architectures Group, working at the intersection of hardware, applications, and machine learning. From 2013 to 2016, he served as a platform architect for Project Catapult, which pioneered the novel uses and widespread adoption of FPGAs for datacenter acceleration. Currently, he is the co-founder and lead for Project Brainwave, Microsoft's principal infrastructure for accelerated AI serving in real time using FPGAs. He has a PhD in electrical and computer engineering from Carnegie Mellon University. Contact him at erchung@microsoft.com.

Doug Burger is the technical fellow for hardware and computer architecture in Microsoft's Azure Division. His team works to develop and deploy new architectures for the intelligent cloud, intelligent edge, and artificial intelligence. He has a PhD from the University of Wisconsin, and was a professor of computer sciences at the University of Texas at Austin, before joining Microsoft in 2008. He is a Fellow of the IEEE and the Association for Computing Machinery (ACM). Contact him at dburger@microsoft.com.