Manticore: A 4096-Core RISC-V Chiplet Architecture for Ultraefficient Floating-Point Computing

Florian Zaruba and Fabian Schuiki, *Integrated Systems Laboratory, ETH Zürich, 8092, Zürich, Switzerland*

Luca Benini, *Department of Electrical, Electronic and Information Engineering, University of Bologna, 40126, Bologna, Italy, and Integrated Systems Laboratory, ETH Zürich, Zürich, 8092, Switzerland*

Data-parallel problems demand ever growing floating-point (FP) operations per second under tight area- and energy-efficiency constraints. In this work, we present Manticore, a general-purpose, ultraefficient chiplet-based architecture for data-parallel FP workloads. We have manufactured a prototype of the chiplet's computational core in Globalfoundries 22FDX process and demonstrate more than 5x improvement in energy efficiency on FP intensive workloads compared to CPUs and GPUs. The compute capability at high energy and area efficiency is provided in "Snitch: A tiny pseudo dual-issue processor for area and energy efficient execution of floating-point intensive workloads," IEEE Trans. Comput., containing eight small integer cores, each controlling a large floating-point unit (FPU). The core supports two custom ISA extensions: The SSRs extension elides explicit load and store instructions by encoding them as register reads and writes ("Stream semantic registers: A lightweight RISC-V ISA extension achieving full compute utilization in single-issue cores," IEEE Trans. Comput.). The floating-point repetition extension decouples the integer core from the FPU allowing floating-point instructions to be issued independently. These two extensions allow the single-issue core to minimize its instruction fetch bandwidth and saturate the instruction bandwidth of the FPU, achieving FPU utilization above 90%, with more than 40% of core area dedicated to the FPU.

Domains such as data analytics, machine learning, and scientific computing are dependent on increasing compute resources.[3] Increasing technology node densities result in systems that are mainly limited by thermal design power and the most feasible way to increase the amount of active compute units is to design more energy-efficient architectures. While many emerging architectures,[4] especially in the machine learning domain, tradeoff floating-point (FP) precision for higher throughput and efficiency, algorithms such as stencils and linear differential equations require higher precision arithmetic. Domain-specific accelerators are a prominent example for how to leverage specialization.[5] Unfortunately, they are hard to adjust to algorithmic changes and tied to a specific application domain.

The trend in leading-edge general-purpose computer architectures paints a similar picture on the importance of increasing energy efficiency. Two prominent examples of recent high-performance architectures are Fujitsu's A64FX[6] and NVIDIA's A100.[7] Both systems strive to control their 32 (A64FX) and 16 (A100) wide multilane SP (sp) data-path with as few instructions as possible.

With the proposed Manticore system, we pursue a similar goal. We achieve this goal by pairing a simple, in-order, 32-bit RISC-V integer core with a large floating-point unit (FPU). Two instruction set architecture (ISA) extensions: SSRs and floating-point repetition (FREP) make it possible for the single-issue integer

0272-1732 © 2020 IEEE
Digital Object Identifier 10.1109/MM.2020.3045564
Date of publication 17 December 2020; date of current version 26 March 2021.

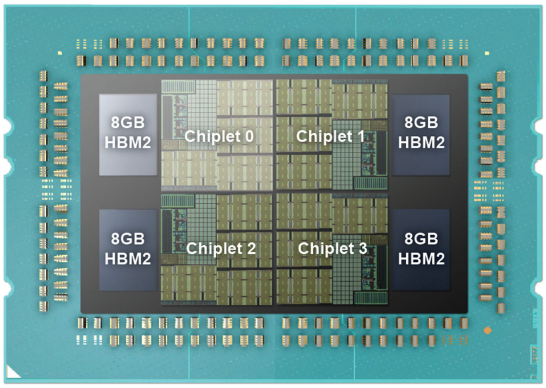

FIGURE 1. Conceptual floorplan of the package. Arrangement of the chiplets and HBM on the interposer. Each chiplet has its own, private, 8 GB HBM. Chiplets interconnect via die-to-die serial links[8].

core to saturate the bandwidth of its FPU, achieving utilization higher than 90% for compute-bound kernels.

CHIPLET ARCHITECTURE

The proposed *Manticore* architecture consists of four 222-mm^2 (14.9 x 14.9 mm^2) 22-nm chiplet dies on an interposer. Using chiplets improves yield and reduces cost. Each die has three short-range, multichannel, in-package chip-to-chip links,[8] one to each sibling. They are used for interdie synchronization and chiplet-to-chiplet nonuniform memory access. Furthermore, each chiplet has access to a private 8-GB high-bandwidth memory (HBM). The conceptual floorplan is depicted in Figure 1.

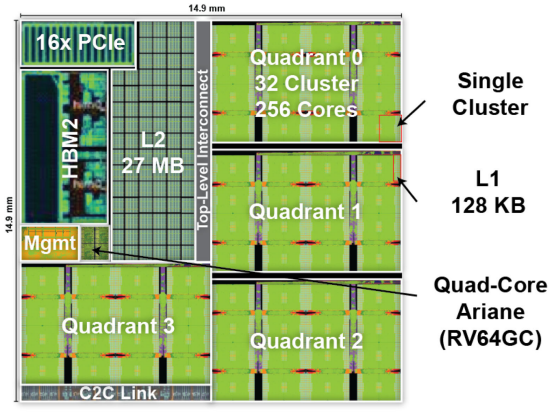

FIGURE 2. Conceptual floorplan of an individual chiplet. Arrangement of individual cluster quadrants, interconnects, L2 memory, HBM2 controller, PCIe controller, and quad-core Ariane RV64GC system.

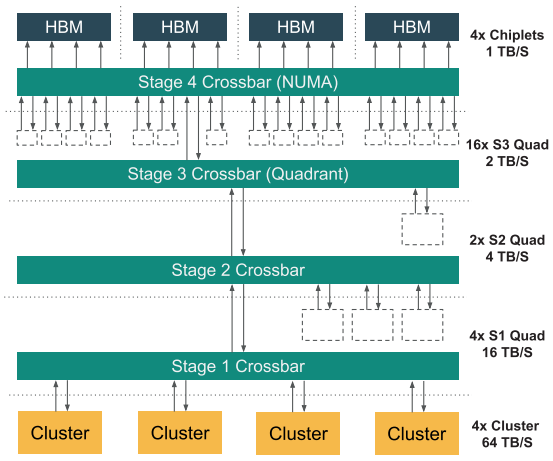

FIGURE 3. Memory hierachy of the Manticore concept. Four cluster form a quadrant and share an uplink into the next stage. Four S1 quadrants form an S2 quadrant which share an uplink to the next stage. Two S2 quadrant form an S3 quadrant. Four S3 quadrants per chiplet share access to the HBM memory.

The chiplet (see Figure 2) contains four quadrants, consisting of 32 clusters with eight cores each, which results in 1024 cores for all four quadrants on a chiplet. Furthermore, each chiplet contains four Ariane RV64GC management cores[9] an HBM (256 GB/s) controller, a 27 MB of L2 memory, and a 16x PCIe endpoint (31.5 GB/s) for host communication, as shown in Figure 2.

The four Ariane management cores run a general-purpose operating system such as Linux and manage the Snitch clusters and program off-loading. The Manticore chiplet has enough silicon area to support 27-MB on-chip shared L2 memory for critical data storage such as neural network weights or stencil kernels.

Memory Hierarchy

Each quadrant[†] (see Figure 3) is further subdivided into multiple stages, in a tree-structure using an interconnect tuned for burst-based direct memory transfer (DMA) accesses. Four clusters share an instruction cache and an uplink into the next stage. These four clusters have a high aggregate bandwidth of 64 TB/s among each other and can perform low-latency, high-bandwidth intracluster data transfers. As shown in Figure 3, clusters share

[†]The term quadrant is somewhat generic and does not necessarily imply four members (cores or lower stage quadrants), as the number of members can be adjusted to match the available bandwidth into the next stage, as for example, in stage three of our system

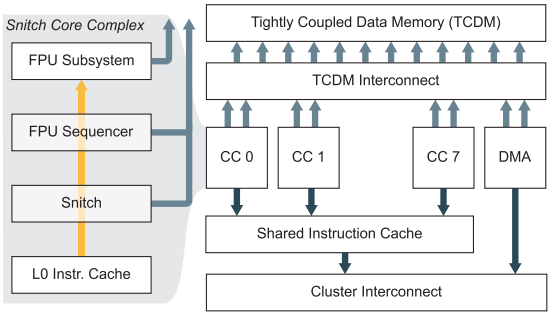

FIGURE 4. Simplified block diagram of a Snitch-based compute cluster. The core complex (CC) contains the integer core and the FPU as well as necessary hardware for the SSRs and FREP. The cluster contains eight core corplices, which share an instruction cache and a tightly coupled data memory. A DMA engines is used for efficient, bulk, data movement.

```
loop:                          scfg 0, %[a], ldA
fld f0, %[a]          ──→      scfg 1, %[b], ldB
fld f1, %[b]                   loop:
fmadd f2, f0, f1               fmadd f2, ssr0, ssr1
```
(a)

```
mv    r0, zero
loop:                          frep  r1, 1
addi  r0, 1           ──→      loop:
fmadd f2, ssr0, ssr1           fmadd f2, ssr0, ssr1
bne   f0, r1, loop
```
(b)

FIGURE 5. Effect of SSRs and frep on the hot loop of a dot product kernel. (a) *Left:* baseline simplified RISC-V implementation, with address calculation and pointer increment omitted for brevity. *Right:* SSRs implementation with memory loads encoded as reads from stream registers; additional stream configuration instructions required ahead of the loop. (b) *Left:* implementation with loop bookkeeping using baseline RISC-V instructions. *Right:* implementation with an frep hardware loop, with all bookkeeping to occur implicitly in the hardware.

the uplink into the next higher stage, the bandwidth to the other S1 quadrants becomes smaller. Bandwidth is subsequently thinned as four S1 quadrants share an instruction cache and an uplink into the S2 quadrant and two S2 quadrants share an uplink into the S3 quadrant. In the last stage of hierarchy 16× S3 quadrants, distributed over four chiplets (nonuniform memory access), share four HBMs with an aggregated peak bandwidth of 1 TB/s. This bandwidth thinning scheme allows us to have a very low diameter, low latency interconnect topology, which can sustainably saturate the HBM bandwidth while being benign to floorplanning and physical design. The interconnect also allows for a very high cluster-to-cluster internal bandwidth, through multiple stages, which by far exceeds the bandwidth into the memory. With this model, we efficiently support cluster-to-cluster traffic, while, at the same time, fully loading the memory system.

Compute Cluster

The compute cluster consists of eight small, 22 kGE, single-stage, 32-bit RISC-V processor cores[1] (see Figure 4). Each Snitch core contains a double-precision (DP) FPU, which can be used to compute one DP fused multiply–add (FMA) operation or two SP FMA per cycle. When running at 1 GHz, a cluster with eight Snitch cores is able to compute 16 DP or 32 SP flop, resulting in 4 TDPflop/s for the entire Manticore system. All eight cores have elementwise, low latency, access into 128-KiB tightly coupled and shared scratchpad memory. Moreover, a DMA engine is in charge of moving blocks of data into the scratchpad memory over a 512-bit data bus. The cores are clocked at 1 GHz, thus delivering more than 4 TDPflop/s peak compute per chiplet.

With this architecture, we achieve a very high compute/control ratio: 44% of the system consisting of compute units, another 44% spent on the L1 memory and just 12% of the area are spent on the control parts.

PROGRAMMING

We leverage two custom RISC-V ISA extensions to achieve extremely high fp utilization and efficiency: Xssr and Xfrep.

Stream Semantic Registers (Xssr)

SSRs[2] offer a means to elide explicit load/store instructions in a program. This is achieved by giving a subset of the processor core's registers *stream semantics*. When enabled, a read from such an SSRs is translated in hardware into a load from memory, and conversely, a register write becomes a store to memory. Since an in-order single-issue core can only execute a single instruction every cycle, the presence of loads and stores in a hot loop of the program diminishes FPU utilization significantly. For example, consider a dot product, which has to issue two loads from memory for every FMA operation, as shown in Figure 5(a). In this scenario, even if the loop is fully unrolled, we achieve at most 33% FPU utilization. In theory, this allows the FPU to be 100% utilized, and even a simple processor can achieve >90% utilization in many relevant kernels without resorting to complex and energy-inefficient wide issue superscalar or very long instruction word (VLIW) architectures.[2] SSRs offer a way to elide memory

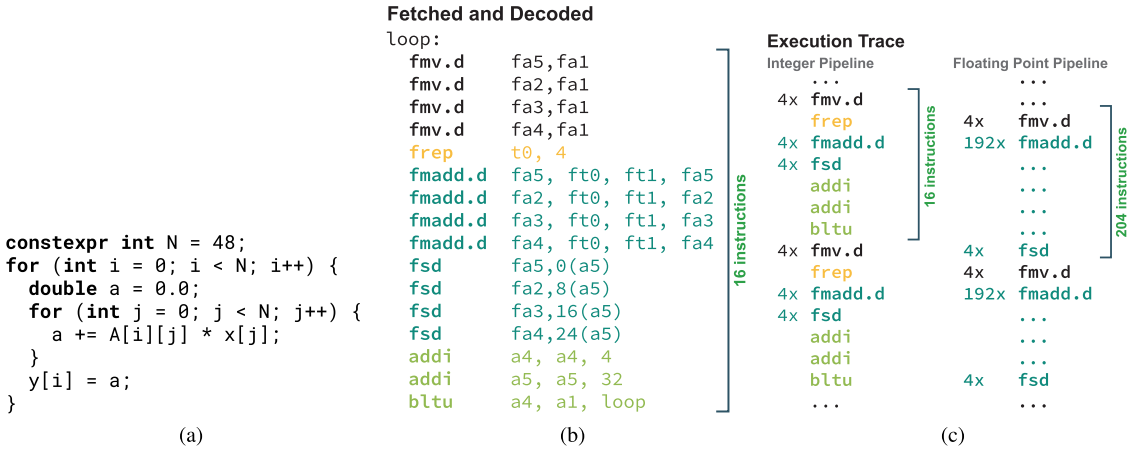

FIGURE 6. Typical execution of a matrix-vector multiplication implementation leveraging the SSRs and frep extensions. The 16 instructions are fetched and decoded once by the integer pipeline of the processor core (b), and expanded to 204 executed instructions in the fpu (c). (a) Reference C implementation with a square matrix A of fixed size 48. (b) Resulting assembly implementation as stored in the binary and fetched/decoded by the processor core. (c) Execution traces of the integer pipeline (left) and the fp pipeline (right).

accesses and address computation in hot loops, which in many cases leaves no integer instructions in the loop body.

Floating-Point Repetition (Xfrep)

The frep[1] extension implements a FPU-only hardware loop. Consider a dot product utilizing SSRs for example, as shown in Figure 5(b). Besides the essential FMA operation running on the FPU, the loop only consists of a trip count increment (addi) and a back branch (bne). This loop can be replaced by a FREP instruction, which loops a range of subsequent FP instructions (one in this case) for a configurable number of times. The RISC-V ISA makes the integration of such an extension very straightforward as most instructions either operate entirely on integer or entirely on FP registers. Only a handful, such as comparisons or moves between integer and FP domains, exchange information from one domain to the other. We leverage this separation and insert a microloop sequence buffer of 16 instructions between the Snitch core and the FPU. FREP instructions configure this buffer to emit a range of buffered instructions multiple times into the FPU, which essentially implements the hardware loop. Since this happens entirely in the FPU subsystem outside of the Snitch core, the core's integer pipeline can run in parallel, enabling a *pseudo-dual-issue* mode of operation that would not be achievable with a traditional hardware loop. This allows the core to perform nontrivial bookkeeping and address calculation while the FPU is running, without incurring a reduction of the FPU utilization.

Typical SSR/FREP Execution

As a concrete example, let us consider the matrix-vector multiplication operation shown in Figure 6(a). A typical implementation leveraging Manticore's SSRs and FREP extensions is shown in Figure 6(b). The address computation and memory accesses of A and x are entirely performed by the SSRs ft0 and ft1. The inner loop is implemented using an FREP instruction and unrolled to compute four results in parallel in order to avoid pipeline stalls due to FPU latency. The outer loop is executed by the integer core. It stores the results (fsd), implements loop bookkeeping (addi, bltu), and initializes a (fmv.d).

As shown in Figure 6(c), the 16 instructions of the assembly implementation are fetched and decoded once by the integer pipeline of the processor core and expand to 204 executed instructions in the FPU through the use of FREP. This leaves 188 cycles for the integer pipeline for other tasks, such as preparing the next loop iteration or coordination of data movement. In case no other work is required, the 16 instructions fetched over 204 cycles of execution amounts to roughly one instruction every 13 cycles, mitigating the von Neumann bottleneck by reducing instruction fetch bandwidth by more than one order of magnitude. Since the FPU can execute the loop iterations back-to-back and of the 204 instructions, 192 perform actual computation, this kernel can achieve up to 94% FPU utilization.

Compilers can leverage these new instructions through scalar evolution and loop analysis to detect loops with the appropriate structure, and matching

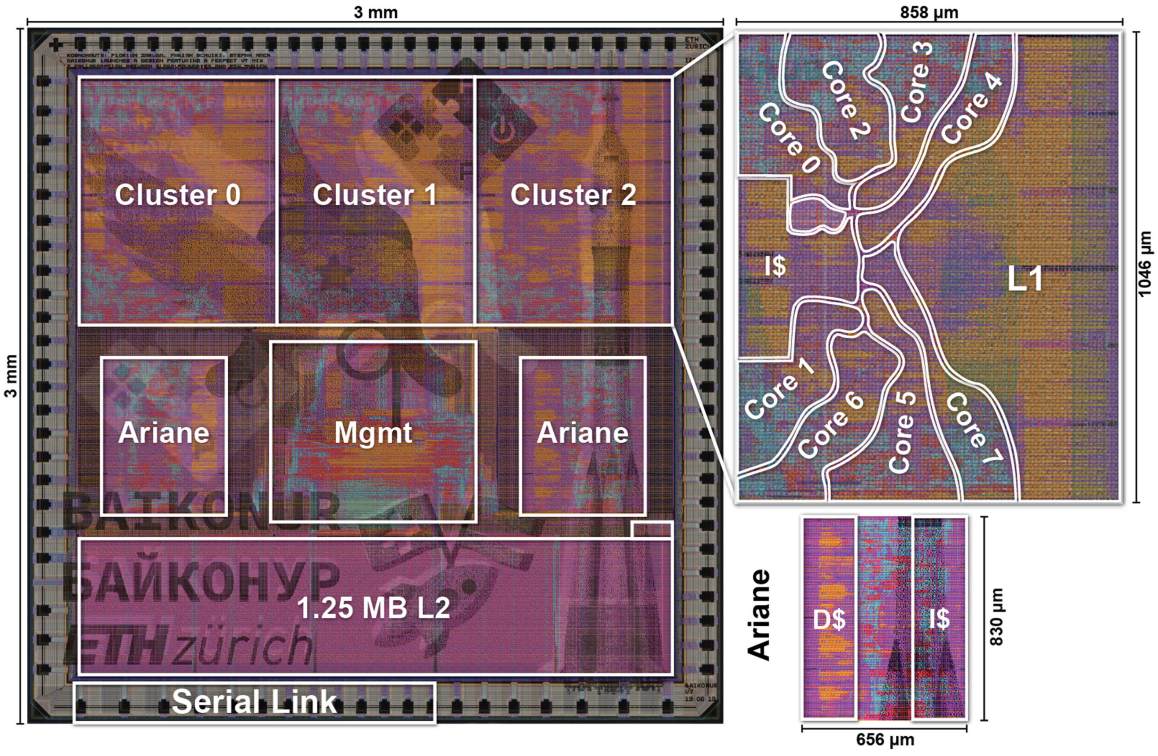

FIGURE 7. Floorplan of the prototype silicon. The two Ariane cores as well as the Snitch cluster have been designed hierarchically. The core's follow a star-shaped layout around the shared instruction cache.

against amenable address calculation, load and use patterns as a peephole optimization. These facilities are commonly provided as part of the existing compiler infrastructure in both GCC and LLVM.

PROTOTYPE

A 3 x 3-mm^2 prototype containing the logic core of the chiplet architecture was manufactured and characterized using the Globalfoundries 22FDX technology. The prototype in Figure 7 contains three Snitch clusters with eight cores (each configured with 8-KiB L1 instruction cache and 128-KiB L1 data memory organized in 32 banks), a dual-core Ariane (with 16-KiB L1 instruction cache and 32-KB data cache), 1.25-MiB L2 memory, and a 400-MHz, double data-rate, 2.56-GB/s, digital-only chip-to-chip link.

SILICON PERFORMANCE

Efficiency

We measured the speed and power consumption of the prototype silicon under representative workloads and operating conditions. Figure 8 shows the DP per-

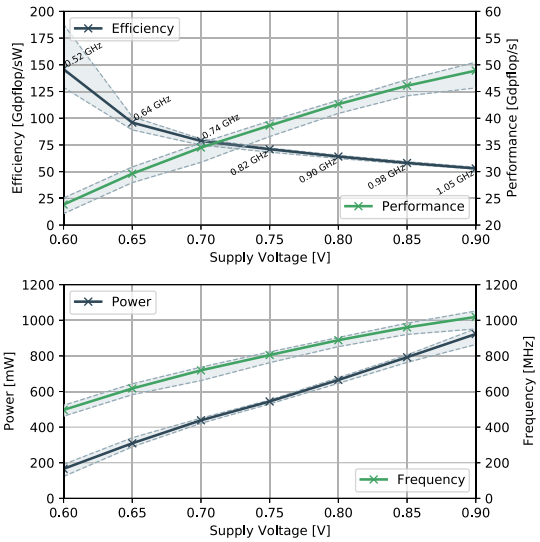

FIGURE 8. Compute performance, energy efficiency, speed, and power consumption for different operating voltages. Measured on the prototype silicon across eight sample dies. Cores performing matrix multiplications, at 90% FPU utilization. Performance and efficiency doubles across range.

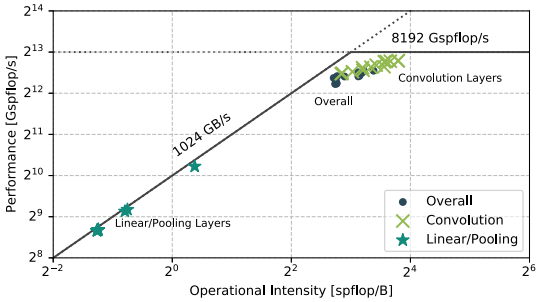

FIGURE 9. Performance roofline plot of DNN training workloads. We group convolutions and linear/pooling layers to indicate performance in the compute- and memory-bound regions, respectively. The Manticore architecture is very efficient at tracking the performance and bandwidth roofline, with a detachment down to 5% for low-intensity and 14% for high-intensity optimized kernels.

formance and energy efficiency achieved when executing parallel workloads on the 24 cores of our prototype 22-nm silicon, for different operating voltages. The chips offer a wide range of operating points and choices for performance/efficiency tradeoffs, which we leverage through dynamic voltage and frequency scaling based on the current workload's operational intensity. This allows us to essentially adjust the roofline of the system to match the current workload. In *high-performance mode* running over 1 GHz at 0.9V VDD, our architecture achieves a peak performance of 54 GDPflop/s across 24 cores and a compute density of up to 20 GDPflop/s mm^2, which translates to 9.2 TDPflop/s across a full 4096 cores. In *max-efficiency mode* running at 0.5 GHz at 0.6 V VDD, our architecture achieves an industry-leading efficiency of 188 GDPflop/sW, while still delivering a respectable 25 GDPflop/s across 24 cores, which translates to 4.3 TDPflop/s across a full 4096 cores.

Roofline

To assess the performance of the manufactured silicon, we analyzed workloads from training steps of a set of deep neural networks (DNNs). Figure 9 shows the roofline plot of our architecture across a full training step. We estimate full-system performance based on cycle-accurate simulation of a smaller instantiation of the hardware, combined with an architectural model of the full system and measured performance characteristics of the prototype silicon. The compute-bound convolution layers in the workload reach >80% of the system's peak performance, and the proximity to the point of inflection of the roofline indicates a balanced utilization of the hardware capabilities. The memory-bound linear and pooling layers reach >90%

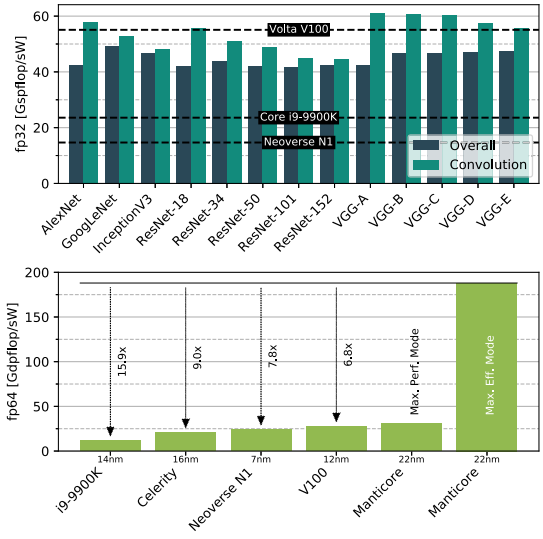

FIGURE 10. *Top:* Estimated sp energy efficiency of a full DNN training step, overall and specifically on the convolution layers for the conceptual Manticore chiplet architecture. *Bottom:* dp energy efficiency on linear algebra (assuming 90% of peak performance); Manticore shown for maximum performance and maximum efficiency operating points.

of the system's peak bandwidth. Since DNN workloads tend to be dominated by the convolution layers, the overall performance, which considers all layers, is almost identical to the convolution performance. Overall we observe that the Manticore architecture is very efficient at tracking the performance and bandwidth roofline, with a detachment down to 5% for low intensity and 14% for high-intensity optimized kernels. The worst-case detachment of 34% from the roofline is encountered in the intermediate region around the point of inflection, where intuitively, the aggregate bandwidth pressure on the L1 TCDM is highest due to the DMA and the compute units both operating at capacity and banking conflicts more frequently stall L1 memory accesses.

Applications

Figure 10 shows the SP energy efficiency achieved in a DNN training step overall, and on the compute-bound convolutions specifically, across a variety of networks, and the industry-leading dp efficiency on linear algebra. On sp DNN training workloads, Manticore's actual efficiency is competitive with the V100 GPU's peak efficiency and outperforms the Core i9-9900K CPU by 2× and the Neoverse N1[10] by 3×. On dp workloads, Manticore outperforms a V100 GPU's peak efficiency by 6×, the N1 by 7×, the Celerity RISC-V CPU by 9×, and the Core i9-9900K CPU by 15×. Our architecture

achieves this despite these chips having a substantial technology advantage due to their 7-, 12-, and 14-nm FinFET processes. Regarding the A100 GPU, our initial estimates based on data published by Nvidia[7] suggest that it achieves a 25% improvement on SP and DP over the V100 in terms of speed at similar power consumption. This indicates that Manticore has just 25% lower efficiency on SP than A100, but outperforms it on DP by 5×, despite the A100's significant 7-nm FinFET technology advantage. Manticore delivers significantly higher peak FP performance than comparable RISC-V architectures[11] in 16 nm.

OVERALL WE OBSERVE THAT THE MANTICORE ARCHITECTURE IS VERY EFFICIENT AT TRACKING THE PERFORMANCE AND BANDWIDTH ROOFLINE, WITH A DETACHMENT DOWN TO 5% FOR LOW INTENSITY AND 14% FOR HIGH-INTENSITY OPTIMIZED KERNELS.

ACKNOWLEDGMENTS

This work was supported by the European Union's H2020 program under Grant 826647 (European Processor Initiative-EPI) and Grant 732631 (Open Transprecision Computing - "OPRECOMP").

REFERENCES

1. F. Zaruba, F. Schuiki, T. Hoefler, and L. Benini, "Snitch: A tiny pseudo dual-issue processor for area and energy efficient execution of floating-point intensive workloads," *IEEE Trans. Comput.*, to be published.
2. F. Schuiki, F. Zaruba, T. Hoefler, and L. Benini, "Stream semantic registers: A lightweight RISC-V ISA extension achieving full compute utilization in single-issue cores," *IEEE Trans. Comput.*, to be published.
3. "AI and compute," 2020. Accessed: Oct. 5, 2020. [Online]. Available: https://openai.com/blog/ai-and-compute/
4. N. P. Jouppi *et al.*, "A domain-specific supercomputer for training deep neural networks," *Commun. ACM*, 2020.
5. A. Yang, "Deep learning training at scale spring crest deep learning accelerator," in *Proc. Symp. High Performance Chips*, vol. 31, 2019.
6. T. Yoshida, "Fujitsu high performance CPU for the Post-K Computer," in *Proc. Symp. High Performance Chips*, vol. 30, 2018.
7. Nvidia, "NVIDIA Ampere GA102 GPU Architecture - The Ultimate Play," 2020.
8. P. Vivet *et al.*, "A 220GOPS 96-core processor with 6 chiplets 3D-stacked on an active interposer offering 0.6ns/mm latency, 3TBit/s/mm2 inter-chiplet interconnects and 156mW/mm2@82% Peak-Efficiency DC-DC Converters," in *Proc. IEEE Int. Conf. Solid-State Circuits*, 2020.
9. F. Zaruba and L. Benini, "The cost of application-class processing: Energy and performance analysis of a Linux-ready 1.7-GHz 64-bit RISC-V core in 22-nm FDSOI Technology," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 11, pp. 2629–2640, Nov. 2019.
10. R. Christy *et al.*, "A 3GHz Arm Neoverse N1 CPU in 7nm FinFet for infrastructure applications," in *Proc. IEEE Int. Conf. Solid-State Circuits*, 2020.
11. S. Davidson *et al.*, "The Celerity open-source 511-core RISC-V tiered accelerator fabric: Fast architectures and design methodologies for fast chips," *IEEE Micro*, vol. 38, no. 2, pp. 30–41, Mar./Apr. 2018.

FLORIAN ZARUBA received a B.Sc. from TU Wien, Vienna, Austria, in 2014 and an M.Sc. in 2017 from the Swiss Federal Institute of Technology Zürich, Zürich, Switzerland, where he is currently working toward a Ph.D. with the Digital Circuits and Systems group of Luca Benini. Contact him at zarubaf@iis.ee.ethz.ch.

FABIAN SCHUIKI received a B.Sc. and an M.Sc. in electrical engineering in 2014 and 2016, respectively, from ETH Zürich, Zürich, Switzerland, where he is currently working toward a Ph.D. with the Digital Circuits and Systems group of Luca Benini. Contact him at fschuiki@iis.ee.ethz.ch.

LUCA BENINI holds the Chair of Digital Circuits and Systems, ETHZ and is Full Professor with the Universita di Bologna. He is a Fellow of the ACM and a member of the Academia Europaea. Contact him at lbenini@iis.ee.ethz.ch.