GAP-8: A RISC-V SoC for AI at the Edge of the IoT

[1]Eric Flamand, [1,2]Davide Rossi, [2,3]Francesco Conti, [1]Igor Loi, [1,3]Antonio Pullini, [1]Florent Rotenberg, [2,3]Luca Benini
[1]GreenWaves Technologies, Villard-Bonnot, France; [2]DEI, University of Bologna; [3]Integrated System Laboratory, ETH Zurich;

***Abstract*—Current ultra-low power smart sensing edge devices, operating for years on small batteries, are limited to low-bandwidth sensors, such as temperature or pressure. Enabling the next generation of edge devices to process data from richer sensors such as image, video, audio, or multi-axial motion/vibration has huge application potential. However, edge processing of data-rich sensors poses the extreme challenge of squeezing the computational requirements of advanced, machine-learning-based near-sensor data analysis algorithms (such as Convolutional Neural Networks) within the mW-range power envelope of always-ON battery-powered IoT end-nodes. To address this challenge, we propose GAP-8: a multi-GOPS fully programmable RISC-V IoT-edge computing engine, featuring a 8-core cluster with CNN accelerator, coupled with an ultra-low power MCU with 30 µW state-retentive sleep power. GAP-8 delivers up to 10 GMAC/s for CNN inference (90 MHz, 1.0V) at the energy efficiency of 600 GMAC/s/W within a worst-case power envelope of 75 mW.**

Keywords—Edge Processing, Internet of Things, Convolutional Neural Networks, Near Sensor Processing, Parallel Architectures.

I. INTRODUCTION

During the last few years we have seen rapid progress in the field of data analytics, thanks to a large variety of robust learning techniques, combined with the availability of large training sets. Common sources of data are those produced from sensors probing the environment for data such as images, sounds, vibrations [1]. It is possible to directly connect sensors to cloud servers carrying out analysis. However if the device needs wireless operation, state-of-the-art low-power streaming data links require too much energy for battery operation as soon as the data volume becomes significant [2]. Performing all or part of data analytics on the edge device can dramatically reduce the amount of data transmitted over the air, since what needs to be carried is a compact signature extracted from the raw data, for example the presence of a target object, that is at least 5 orders of magnitude smaller than the original image. The challenge becomes how to deliver processing capabilities well above 1 Giga Operations Per Second (GOPS) on a small battery power and energy budget, achieving reasonable life time expectation (one year or more).

To address this challenge, we developed GAP8, a multi-core processor derived from the PULP [1] open source computing platform [1], fabricated in TSMC 55nm LP CMOS technology. It leverages the flexibility and openness the RISC-V ISA to integrated a state-of-the-art microcontroller, a rich set of peripherals, secure execution, associated with a powerful programmable parallel processing engine for flexible multi-sensor (image, audio, inertial) data analysis and fusion, which includes a dedicated convolutional engine for deep neural network inference. These two key building blocks are supported by aggressive on-chip power conversion and management to reduce component count while maximizing battery power down-conversion efficiency, enabling an "energy-proportional" heterogeneous architecture for always-ON IoT end-nodes. GAP-8 can deliver up to 10 GMAC/s for CNN inference (90 MHz, 1.0V) at the energy efficiency of 600 GMAC/s/W within a worst-case power envelope of 75 mW.

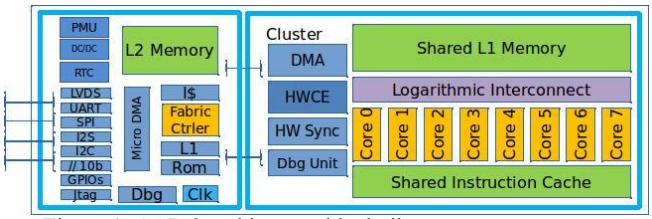


Figure 1. GAP-8 architecture block diagram.

II. ARCHITECTURE

Figure 1 provides a top-level view of the GAP-8 SoC architecture, including two voltage and frequency domains controlled by embedded DC/DC and Frequency-Locked Loops (FLLs): the Fabric Controller (FC) and the Cluster.

A. Fabric Controller

The Fabric Controller is an advanced Micro Controller Unit (MCU) built around a RISC-V core extended for energy efficient digital signal processing [2] equipped with an instruction cache and a fast access-time data memory. The SoC includes a full set of peripherals (i.e. QSPI, I2C, I2S, CAM) enabling parallel capture of images, sounds and vibrations, and a 4-channel PWM interface for motor control, for use in applications such as drones and domestic robotics. Data transfers from/to the peripherals are managed by a multi-channel I/O DMA to minimize the amount of interactions and the workload of the controlling core when performing I/O. 512 kB of L2 memory are available on the SoC, together with a ROM, storing the primary boot-code including secured boot support through eFUSE stored keys. Memory is extendable via a DDR HyperBus interface. GAP-8's memory hierarchy is organized as a single name space: every single core in the chip can access all memory locations, unless they are protected by the Memory Protection Unit (MPU), with progressively increased access latency for L2 memory or external DDR memory (L3). To hide the access cost of L3 and L2 memory the cluster contains a multi-channel DMA capable of 1D and 2D bulk memory transfers.

[1] PULP: The Open-Source Parallel Ultra-Low-Power Processing Platform – www.pulp-platform.org

978-1-5386-7479-6/18/$31.00 ©2018 IEEE

TABLE I. GAP-8 PEAK PERFORMANCE @ 100 MHZ.

	32x32-bit GMAC/s	16x16-bit GMAC/s	8x8-bit GMAC/s	16x4-bit GMAC/s
SoC	0.1	0.2	0.4	-
Cluster	0.8	1.6	3.2	-
HWCE	-	2.8	5.6*	11.2*
Total	0.9	4.6	9.2	11.2

*HWCE supports also 16x8 and 8x4, and 4x4 configurations.

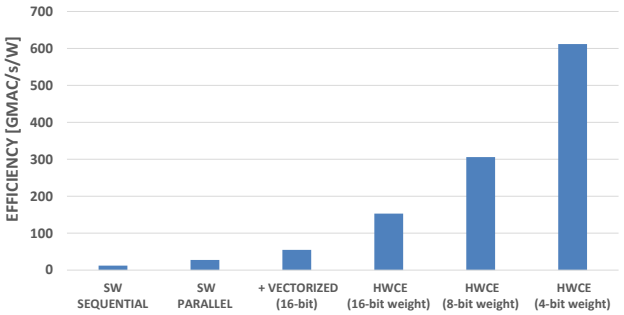


Figure 2. GAP-8 CNN Peak Efficiency @ 100 MHz, 1V.

B. Parallel Computing Cluster

The cluster, residing on a dedicated voltage and frequency domain, is turned on when applications running on the fabric controller offload highly computation-intensive kernels. It contains 8 RISC-V cores identical to the one used in the FC, allowing the SoC to run the same binary code on either the fabric controller or the cluster. This 8-core cluster is served by a shared L1 data memory, enabling shared-memory parallel programming models such as OpenMP. Access to the L1 memory is supported by a highly optimized interconnect located between the cores' load/store units and the memory banks. The shared L1 can serve all memory requests in parallel with single-cycle access latency and low average contention rate (<10% on data intensive kernels). The cluster program cache is also shared to maximize efficiency in fetching data-parallel code.

Fast event management, parallel thread dispatching, and synchronization are supported by a dedicated hardware block (HW Sync), enabling very fine-grained parallelism and hence high energy efficiency. The HW Sync block also controls the top-level clock gating of every single core in the cluster. A core waiting for an event (attached to a synchronization barrier or general event) is instantly brought into a fully clock gated state, zeroing its dynamic power consumption.

The cores in GAP-8 feature an in-order, 4-stage pipeline, compliant with the RISC-V ISA subsets I, M, C. Since the RISC-V ISA is architected to be extendable, we have extended it to boost performance for DSP.centric kernels, manipulating real or complex numbers represented as vectors of short integers. The fetch stage has been extended with zero-overhead hardware loops. Efficient array manipulation is supported by post-modified load and store instructions, single-cycle multiply and accumulate, single-cycle complex multiplication, as well as dedicated instructions for efficient rounding, normalization and clipping [4]. To further increase the instruction-level parallelism (ILP), single instruction multiple data (SIMD) instructions have been added, enabling vectors of 4-byte elements or 2 short elements. Finally, bit-manipulation oriented instructions, like bit insertion or extraction, are also added to make control-oriented code more compact and faster. The core also complies with the RISC-V privileged instructions specification, to enable the execution of secured code, assisted by a built-in programmable memory protection unit (MPU).

TABLE II. GAP-8 POWER MODES AND CONSUMPTION.

Power Mode	VDD [V]	Nom. Freq. [MHz]	Power @ Nom. Freq.
Deep Sleep	0.8	0.32	3.6 µW
Retentive Deep Sleep	0.8	0.32	30 µW
FC ON	1.0	150	27 mW
FC ON, Cluster ON	1.0	150, 90	37 mW
FC ON	1.2	250	39 mW
FC ON, Cluster ON	1.2	250, 170	75 mW

C. Accelerating CNN Workloads

When the cluster runs CNN-based applications, it can offload convolutional layers computation to a dedicated accelerator, the Hardware Convolution Engine (HWCE) [5]. This block can evaluate a 5×5 convolution or three 3×3 convolutions on 4-, 8- or 16-bit operands in a single cycle (Table I). It is directly connected to the cluster's shared L1 memory through four load/store ports similar to the ones used in the cluster's programmable cores; a HW-accelerated convolution can be freely mixed with activities running on the cores. Besides boosting performance, the HWCE plays an essential role in improving the energy efficiency of the overall system when running CNN-based applications. Internal data reuse leads to a 4× to 5× energy efficiency boost compared with a parallel and vectorized SW implementation running on the cluster cores, and up to 10× energy efficiency improvement when using quantized 8-bit or 4-bit weights, as shown in Figure 2.

D. Power Management

To maximize power efficiency, and to minimize the number of external components, the SoC contains an internal DC/DC converter that can be directly connected to an external battery. It can deliver voltages in the range of 1.0V to 1.2V when the circuit is active. When the circuit is in sleep mode, this regulator is turned off and a linear drop-out (LDO) regulator is used to power the real-time clock, which controls programmed wake-up and, optionally, part of the L2 memory allowing retention of application state for fast wakeup. When in deep sleep, the current consumption is reduced to 3.6 µW (assuming the real time clock is active and no data retention), and 30 µW assuming full L2 retention. The two main domains have their own separate clocks. The typical turn-around time is between 100µs and 150µs allowing for agile power state transitions. Table II shows the power modes of GAP-8, together with nominal frequency of FC and Cluster, and the related power consumption at maximum computing workload.

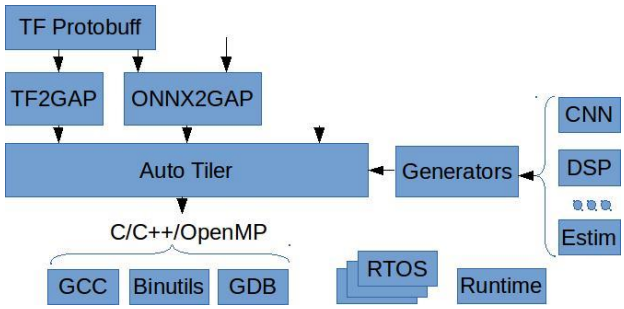


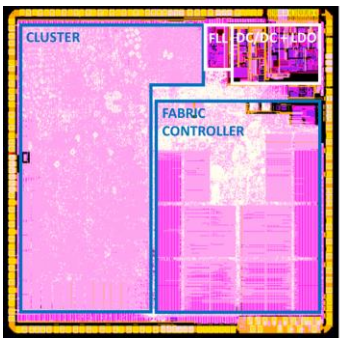
Figure 3. Overview of the GAP-8 Software Development Flow.

III. Programming GAP-8

GAP-8 supports GCC 7.1 for code generation, including all the core DSP extensions. Applications are written in C/C++, and extended with OpenMP as a parallel programming API, enabling access to hardware task dispatch and synchronization resources available on the cluster. Platform resource management (peripherals, time, memory, power) is supported either by the PULP-OS (event based) or through an ARM Mbed port (thread or event based). Programs are always fetched from L2 memory through the previously described instruction caches and can be statically or dynamically linked. In the latter case dynamic relocation is performed through a lightweight model to limit code expansion in L2. Combining dynamic relocation with the user/machine mode support in the core and memory protection (MPU) eases the deployment of secured kernels and applications.

Data is not cached to avoid the significant power penalty associated with data caches. To use the architecture at its best the preferred approach is to always try to keep data as close as possible to the cores using it (in L1). Data is efficiently moved from and to L2 and L1 by the DMA or uDMA engines, but code restructuring and organization can prove to be error prone and time consuming. To ease development, a tool has been developed to automate this process. Basic kernels are first written without taking into consideration where data is located. These kernels can be optimized and parallelized without the programmer being concerned with data placement. The basic kernels are then combined into what we call user-kernels. A user-kernel is described as a multi-dimensional iteration space. It contains a collection of connected basic kernels that can be inserted into the user-kernel iteration space at pre-defined locations (depth, prologue, body, epilogue). The user-kernel definition contains a series of argument definitions for the basic kernel arguments. Each argument defines which sub space of the kernel iteration space it is concerned with, a tiling direction, a home location of the data (L2 memory, external memory) and a set of properties.

Using this model and a given L1 memory budget, the tool infers a tiling structure for each argument fitting within the L1 memory budget and satisfying the set of constraints put on all the arguments. Once the tiling structure has been computed, it generates a C program that takes care of providing tiles to the basic kernels in a pipelined manner to keep all the cores



Technology	CMOS 55nm LP
Die Area	10 mm²
Embedded Memory	4608 Kbit (4096kbit state-retentive)
Logic Gates	2 Mgates
Power Management	1 DC/DC, 1 LDO, 2 FLLs, embedded power switches
VDD range	0.8V – 1.2V
Frequency Range	32kHz – 250 MHz
Power Range	3.6 µW – 75mW

Figure 4. GAP-8 layout view and main chip features.

continuously working. The tool is delivered as a C library which exposes an API to create models. The models themselves are written in C and once linked with the library become executable. Running a compiled model produces C code wrappers containing calls to the basic kernels (either sequential or parallel) as well as DMA transactions. With this approach, it is possible to build generators for specific algorithms and then combine these together. We have developed generators that, for example, automatically generate different types of CNN layers. Adding a layer to a CNN graph is simplified to defining the nature of the layer (convolution, pooling, rectification, and so on), dimensionality and location of the coefficients (L2, external memory). These parameters are all captured in a single call to a generator. Since generators can be combined, the whole network can be easily built and executing the model will produce all the wrappers calling the optimized basic kernels and managing memory movements. This automatic tiling tool is used as a backend for higher level tools to automate support direct export from CNN frameworks such as TensorFlow into optimized C code ready to be compiled on the chip. Fig. 3 summarizes the key modules involved in the software development flow.

IV. GAP-8 Benchmarking

Figure 4 shows the GAP-8 chip layout and summarizes its main features. Tables III and Table IV show the performance measured on the silicon prototypes on several near-sensor data analytics applications, including CNN workloads exploiting the dedicated hardware of GAP-8 architecture.

In Table III cycles per produced elementary output are given as a function of the number of cores. It should be noted that the exact same binary is used when running on 1, 2, 4 or 8 cores. The code is automatically dispatched on the number of cores dynamically passed to the hardware dispatcher. All cycles counts are obtained using a hardware timer to capture time before and after the application. The cycles-counts capture all activities, not just CPU time but also DMA and distant level memory accesses. The applications have been implemented to optimally benefit from the parallelization and vectorization opportunities, showing the architecture capable of achieving nearly ideal speed-ups on a wide spectrum of near-sensor data analytics kernels.

Table IV shows the total number of cycles required when running a full inference on 3 CNNs. Cycle counts here include

TABLE III. APPLICATION PERFORMANCE [KCYCLES].

Application	Cores				Speed-up vs. 1-core
	1	2	4	8	
1D FFT1024 Radix4	28.2	14.3	7.8	4.7	6×
2D FFT 256 x 256 Radix4	78.9	41.9	22.6	13.3	5.9×
Byte 5x5 Conv	18.5	9.3	4.7	2.2	8.4×
Short 5x5 Conv	37.8	18.9	9.5	4.6	8.2×
Binary 5x5 Conv	20.8	10.5	5.3	2.8	7.4×
Short MaxPool2x2	8.2	4.2	2.1	1.1	7.5×
Short MatMult 32x32	41.9	20.9	14.0	5.2	8×
Short 2048 to 1 Fully Connected	3112.0	1616.0	847.0	495.0	6.3×
CannyEdge	99.5	50.9	26.2	12.7	7.8×
AES-CTR 128b	15.3	7.7	4.0	2.1	7.3×
64 Mel Coefficients	542.7	299.4	176.7	101.3	5.3×
HoG, 8x8 Cells, 2x2Blocks,9 Bins	65.0	35.0	18.0	9.0	7.2×

all operations. For example, for the TextReco network, access to coefficients stored in an external memory is included. Cycles are reported for 5 configurations, all using 16-bit of precision: 1, 2, 4, 8 cores and 8 cores with the HWCE accelerator running the convolutional layers, while all the other layers are running on the 8 cores.

With respect to existing heterogeneous architectures specialized for CNN-based near sensor-data analytics such as *Mia Wallace* [6], *Fulmine* [7], *mote SoC* [8], *CEVA-XM4* [9] GAP-8 provides a significantly more power/performance scalable and flexible architecture, capable to run arbitrary algorithms. One peculiarity of GAP-8 lies in the capability to activate on-demand the controlling and data-acquisition resources of the SoC such as the Fabric Controller and the autonomous I/O DMA subsystem, while offloading compute intensive workloads to the parallel cluster only when required by the applications. This hierarchical and heterogeneous architecture, joint with strong power management enabled by the on-chip voltage regulators, power switches, and state-retentive L2 memory, couples the sleepwalking capabilities typical of low-power microcontrollers with several orders of magnitude higher peak performance, paving the way for always-on near-sensor data analytics at the edge of the IoT.

GAP-8 features a deep-sleep power of 3.6 µW (30 µW with 512kB of state retentive memory), more than one order of magnitude smaller than other works [6][7][8][9], with a peak performance of 10 GMAC/s/W at the energy efficiency of 600 GMAC/s/W, similar to the other works. The bulk of CNN-based data analytics applications, performed taking advantage of HWCE acceleration, enables CIFAR-10 classification in 650µs @ 100MHz, 15 mW - with a consumption of 10 µJ, which means that this operation could run for > 700 days on a single AAA battery if executed once every 100ms (10 fps).

TABLE IV. FULL CNN PERFORMANCE [CYCLES].

Topology	Cores					Speed-up vs. 1-core	Energy gain vs. 1-core
	1	2	4	8	+ HWCE		
CIFAR10	711 K	415 K	254 K	178 K	65 K	10.9×	4.9×
MNIST	7.6 M	4 M	2.4 M	15 M	0.8 M	9.3×	4.3×
TextReco	98 M	51 M	28 M	17 M	8.3 M	11.7×	5.3×

V. CONCLUSION

We have presented an ultra-low power programmable platform for near-sensor content understanding, in particular based on CNN. We have provided evidence that when this platform operates on real networks the combination of parallelism and a hardware accelerator leads to a 10x improvement versus a single core model while also improving energy efficiency. Through a set of kernels and real-life applications we have shown the capability of this platform to scale efficiently with the number of used cores. We have shown the importance of high-level tools to efficiently map complex applications onto a parallel architecture which by necessity is not equipped with hardware assistance to hide the complexity of explicit memory hierarchy management. The architecture has been taped out in GAP-8 using the TSMC55LP process. It shares power and size characteristics with state of the art ultra-low power MCUs, but at the same time, thanks to aggressive parallel and vector computing capabilities, is capable of delivering several GOPS within a very small power envelope. We show how this architecture enables new applications for battery operated edge devices with rich-data sensing capabilities.

REFERENCES

[1] F. Bonomi et al., “Fog computing and its role in the internet of things,” ACM MCC, pp. 13-16, 2012.

[2] H. De Groot, “IoT and the Cloud: a hacked personality and an empty battery head-ache or an intuitive environment to make our lives easier?,” S3S, pp. 1-5, 2015.

[3] D. Rossi *et al*., "PULP: A parallel ultra low power platform for next generation IoT applications," *2015 IEEE Hot Chips 27 Symposium (HCS)*, Cupertino, CA, 2015, pp. 1-39.

[4] M. Gautschi *et al*., "Near-Threshold RISC-V Core With DSP Extensions for Scalable IoT Endpoint Devices," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 10, pp. 2700-2713, Oct. 2017.

[5] F. Conti and L. Benini, "A ultra-low-energy convolution engine for fast brain-inspired vision in multicore clusters," 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015, pp. 683-688.

[6] A. Pullini, F. Conti, D. Rossi, I. Loi, M. Gautschi and L. Benini, "A Heterogeneous Multi-Core System-on-Chip for Energy Efficient Brain Inspired Computing," in *IEEE Transactions on Circuits and Systems II: Express Briefs*.

[7] F. Conti *et al*., "An IoT Endpoint System-on-Chip for Secure and Energy-Efficient Near-Sensor Analytics," in *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 9, pp. 2481-2494, Sept. 2017.

[8] T. Karnik *et al*., "A cm-scale self-powered intelligent and secure IoT edge mote featuring an ultra-low-power SoC in 14nm tri-gate CMOS," *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, San Francisco, CA, 2018, pp. 46-48.

[9] Y. Siegel, "The path to embedded vision & AI using a low power vision DSP," *2016 IEEE Hot Chips 28 Symposium (HCS)*, Cupertino, CA, 2016.