Journal of
Low Power Electronics and Applications

Article

A Bottom-Up Methodology for the Fast Assessment of CNN Mappings on Energy-Efficient Accelerators

Guillaume Devic, Gilles Sassatelli and Abdoulaye Gamatié *

LIRMM, Univ. Montpellier, CNRS, 34095 Montpellier, France
* Correspondence: abdoulaye.gamatie@lirmm.fr

Abstract: The execution of machine learning (ML) algorithms on resource-constrained embedded systems is very challenging in edge computing. To address this issue, ML accelerators are among the most efficient solutions. They are the result of aggressive architecture customization. Finding energy-efficient mappings of ML workloads on accelerators, however, is a very challenging task. In this paper, we propose a design methodology by combining different abstraction levels to quickly address the mapping of convolutional neural networks on ML accelerators. Starting from an open-source core adopting the RISC-V instruction set architecture, we define in RTL a more flexible and powerful multiply-and-accumulate (MAC) unit, compared to the native MAC unit. Our proposal contributes to improving the energy efficiency of the RISC-V cores of PULPino. To effectively evaluate its benefits at system level, while considering CNN execution, we build a corresponding analytical model in the Timeloop/Accelergy simulation and evaluation environment. This enables us to quickly explore CNN mappings on a typical RISC-V system-on-chip model, manufactured under the name of GAP8. The modeling flexibility offered by Timeloop makes it possible to easily evaluate our novel MAC unit in further CNN accelerator architectures such as Eyeriss and DianNao. Overall, the resulting bottom-up methodology assists designers in the efficient implementation of CNNs on ML accelerators by leveraging the accuracy and speed of the combined abstraction levels.

Keywords: design methodology; energy-efficient embedded systems; machine learning accelerators; simulation; multiply and accumulate; RISC-V; GAP8; Eyeriss; DianNao

check for updates

Citation: Devic, G.; Sassatelli, G.; Gamatié, A. A Bottom-Up Methodology for the Fast Assessment of CNN Mappings on Energy-Efficient Accelerators. *J. Low Power Electron. Appl.* **2023**, *13*, 5. https://doi.org/10.3390/jlpea13010005

Academic Editors: Teresa Cervero, Kevin Martin, Mario Kovač and Maurizio Martina

Received: 28 October 2022
Revised: 16 December 2022
Accepted: 23 December 2022
Published: 5 January 2023

1. Introduction

With edge computing, embedded systems and servers interact with sensor data to carry out a variety of tasks. A typical task is analyzing and extracting information from environment data by using machine learning (ML) algorithms. In order to reduce the response time and energy consumption of these tasks as much as possible, embedded systems have been deployed closer to the sensors [1,2].

The convolutional neural network (CNN) is a popular machine learning model that can be used for both training and inference. In order to execute it at the edge, it needs energy-efficient embedded systems. For such systems to be designed successfully, it is imperative to understand the computational complexity and size of the main CNN components. A breakdown of the major parameters for five typical CNN models can be found in Table 1. The neural networks are composed of convolutional layers and fully connected layers for classifying images. Convolution layers achieve a high number of multiply-and-accumulate (MAC) operations, between 564 million and 15.5 billion. To successfully execute such CNNs, powerful MAC execution units, along with adequate memory management, are required.

A central focus of this paper is the definition of a methodology for evaluating the development and incorporation of a powerful MAC unit into CNN accelerator architectures for energy-efficient model execution. In recent years, quantization [3] has become extremely popular for reducing the power consumption of CNN workloads by, for example, reducing the bit-width representation of weight parameters. Existing research on deep CNNs has demonstrated the importance of applying quantization at the layer level: layers can have different quantization thresholds. As a result, CNN models with mixed precision are produced.

The following questions are therefore of interest to us.

- How can a flexible and energy-efficient MAC unit be designed that can handle wide bit-width data representations, ranging from 2 to 32 bits?
- How can we easily integrate candidate MAC units into typical CNN accelerator architectures to quickly evaluate their impact on their energy efficiency, when mapping and executing CNNs?

There are several abstraction levels that can be applied to design methodologies, depending on the tradeoff expected between speed and accuracy in the design evaluation process (see Figure 1). The design space of systems has been studied for decades [4].

Table 1. Main parameters in selected CNN models (M = million, B = billion).

	MobileNet 2017 [5]	**AlexNet 2012 [6]**	**GoogleNet 2014 [7]**	**ResNet-50 2015 [8]**	**VGGNet-16 2014 [9]**
Input size	224 × 224	227 × 227	224 × 224	224 × 224	224 × 224
Num. of convolution layers	22	5	57	53	13
Num. weights	3.17 M	2.3 M	6 M	23.5 M	14,7 M
Num. of MAC operations	564 M	666 M	1,43 B	3,86 B	15.3 B
Num. of fully-connected layers	1	3	1	1	3
Num. of weights	1 M	58.6 M	1 M	2 M	124 M
Num. of MAC operations	1 M	58.6 M	1 M	2 M	124 M
Total weights	4.2 M	61 M	7 M	25.5 M	138 M
Total MAC operations	564 M	724 M	1.43 B	3.9 B	15.5 B

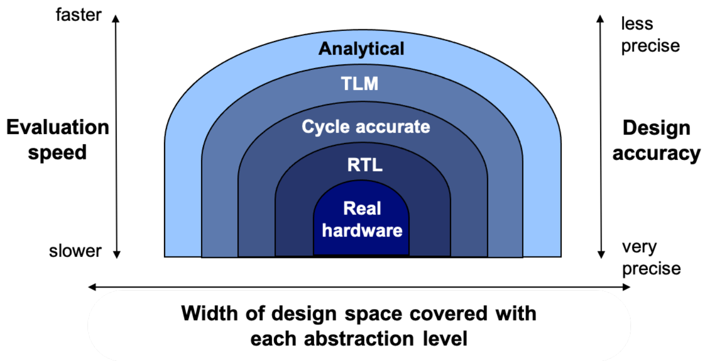


Figure 1. Abstraction levels for design space exploration.

Design evaluation can be enhanced by considering typical hardware implementations [10], such as field-programmable gate array (FPGA) or application-specific integrated circuit (ASICs). This is the most accurate method. The tedious implementation phases, however, limit its flexibility for early exploration of design options. To partially overcome this required design effort, some work proposed systematic engineering methods to facilitate hardware development from higher-level modeling [11–14]. In other approaches, designs are defined at register–transfer level (RTL). VHDL and Verilog are commonly used for creating and simulating target system models. Although less accurate than hardware-based design, they provide a high level of precision for reliable design assessment. Even so, exploring the design space at the RTL level is known to be time consuming [15]. Further approaches focus on system design assessment at the cycle-accurate or cycle-approximate abstraction level. They often consider software simulators, which provide good design flexibility [16–18]. These simulators are quite accurate, but they are often slow and cannot accommodate large design areas. By using transaction-level modeling (TLM) [19–22], simulation and modeling of complex systems are possible at a faster rate than at a cycle-accurate level. The transmission delay of communications is not taken into account when simulating communication transactions. By doing so, the modeled system is executed more rapidly. Last but not least, analytical modeling allows designers to quickly assess systems [23–26]. Thus, such approaches can be used to explore early design space. This advantage, however, comes at the cost of a lower level of modeling accuracy. Most analytical approaches rely on mathematical descriptions of the system's behavior in order to analyze it.

Every method outlined above has its own advantages and disadvantages when it comes to exploring the design space. It is possible to adopt different abstraction levels at different stages of the exploration process. By using a top-down approach, for example, one could begin with a high abstraction level and then move to a lower abstraction level in order to refine the design evaluation over a smaller design space. A bottom-up approach, on the other hand, can first accurately define some basic system components in order to extract precise behavioral properties, such as latency and power consumption, from them. By utilizing this information, high-level models can be calibrated and assessed quickly.

1.1. Our Contribution

The purpose of this paper is to propose a bottom-up design methodology for the efficient execution of CNNs on improved accelerator architectures. We consider two abstraction levels among the five aforementioned ones: the RTL and analytical modeling levels. By using a two-step bottom-up design methodology, we first study an energy-efficient MAC unit dedicated to ML accelerator architectures. In the first step, we develop a flexible MAC unit design at RTL level by using the open-source RI5CY core, which is based on the RISC-V instruction set architecture (ISA) [27]. The GAP8 processor is also based on the RISC-V core architecture [28]. Data bit-width representations between 32 and 2 bits are supported in our proposed architecture, as well as asymmetric bit-width operations. A comparison between our novel MAC unit and the native MAC unit of the RI5CY core shows savings of 25% and 10% in power and area, respectively.

In a second step, we create a more abstract model of our MAC in the Timeloop/Accelergy environment [29,30]. This allows us to simulate and evaluate the execution of CNNs mapped to specific accelerator architectures. The simulation is realized by Timeloop, whereas the energy estimation is achieved by Accelergy. It should be noted that combining complementary tools such as Timeloop and Accelergy for system simulation and evaluation is not a novel concept. At the cycle-approximate level, similar approaches exist, as illustrated in [31] in which gem5 is combined with McPAT and NVSim to assess the design. As part of this study, we leverage the advantages of an analytical Timeloop/Accelergy approach to assess CNN mappings on target architectures in order to explore a broader design space. Specifically, we are able to evaluate the impact of our proposed MAC unit on ML accelerators, such as Eyeriss and DianNao, beyond GAP8. It is demonstrated here that energy efficiency can be improved, by evaluating both homogeneous and heterogeneous (i.e., mixed precision) bit precision with regard to CNN layers. Recently, Timeloop/Accelergy has been applied to analyzing approximate computing in hardware accelerators [23].

In general, this paper illustrates how a typical bottom-up design methodology can be used by designers to explore the design space of CNN executions on accelerator architecture, whether it is RISC-V or another instruction set architecture. It is worth noting that the first step of our methodology is completely based on our prior work [32] on the design of MAC units for smart, low-power edge computing. Consequently, the present paper is an expanded version of this seminal work. The goal of this paper is to demonstrate how such a MAC component may also be abstracted at a higher level and incorporated into ML accelerator architectures in order to evaluate energy-efficiency gains at system level. The focus in [32] was mainly on the MAC design at RTL level.

1.2. Outline

The remainder of this paper is organized as follows. Section 2 discusses some related work on machine learning accelerator architectures. Section 3 introduces the general methodology proposed in the current study. Section 4 describes a flexible MAC unit designed at RTL level that improves the efficiency of an existing RISC-V core. To explore a broader range of designs at system level, the resulting MAC design is modeled at a higher abstraction level in Section 5. Finally, some concluding remarks are given in Section 6.

2. Related Work

Matrix multiplications are commonly processed by convolutional neural networks, which can then be decomposed into vector multiplications in the context of linear algebra. This is achieved by performing parallel data processing at the processing element (PE) level and/or within each PE. In the following paragraphs, we provide a brief overview of CNN accelerators [33,34]. We discuss three mainstream design approaches: CPU or GPU-based, FPGA-based, and specialized ASIC-based. Last but not least, we discuss some design and evaluation approaches for the RISC-V instruction set architecture.

CPU or GPU-based designs. Tightly-coupled multicore clusters provide highly parallel processing capabilities, which are suited to CNN models with high data parallelism. The GAP8 multicore accelerator proposed by Gautschi et al. [28] relies on this paradigm by using RISC-V cores. In its cluster, CNN weights and input data are stored in a shared scratchpad memory. Another CPU-based accelerator has been proposed in [35] to provide higher flexibility thanks to its CPUs capability to execute a wide range of operations supported by their ISA. The von Neumann fetch–decode–execute model is used in both [28,35], which is potentially power-consuming [36].

The single instruction multiple data (SIMD) execution model has become widely used in modern CPUs to overcome the limitations of the von Neumann paradigm [28,37]. Date bit widths are usually predefined in SIMD. As a practical matter, hardware support and ISA limitations often restrict bit widths to a limited range. For example, SIMD units in ARM M-cortex cores can only handle data with 8 or 16 bit widths [37]. In the GAP8 architecture, the RISC-V RI5CY core implements a SIMD unit that supports 2, 4, 8, and 16 bit widths [27]. In this case, each data precision level has its own SIMD unit. This significantly increases the overhead in terms of area and consumption.

FPGA-based designs. Models and techniques associated with machine learning are rapidly evolving. The hardware designs implementing these models should therefore be flexible enough to postpone their obsolescence as much as possible. A reprogrammable hardware device such as an FPGA is an excellent candidate for solving this problem. In existing work [38–40], CNN-specific functions can be implemented on FPGAs. In spite of high performance levels, FPGA accelerators consume more energy than ASIC accelerators [41].

Specialized ASIC CNN accelerators. Recent edge computing requirements have increased the demand for lightweight, energy-efficient ASIC CNN accelerators. Energy efficiency is affected by several factors, including the memory hierarchy, the parallelism level implemented, and the processing unit. When designing the latter, optimization goals can be set in order to target these key factors. In order to meet this requirement, the Eyeriss accelerator optimizes the memory hierarchy, the on-chip communication interconnect, and the dataflow execution model [42,43]. A number of accelerators, including DianNao [44], EIE [45], and others [46–48] share similar optimization targets. By using flexible compute units, accelerators such as Bit Fusion [49], Loom [50], and others [51,52] aim to improve energy efficiency. Table 2 summarizes some popular CNN accelerators. A very recent and comprehensive survey of accelerators can be found in [34].

Table 2. Selected convolutional neural network accelerators.

Name	Technology (nm)	Frequency (MHz)	Energy Eff. (TOPS/W)	MAC Adaptivity	Supported Bit Widths
Eyeriss [42]	28	200	0.15–0.35	no	16
Eyeriss V2 [43]	65	200	0.96	no	8
Envision [53]	28	200	0.53–10	yes	2, 4, 8, 16
UNPU [51]	65	200	3.08–50	yes	1–16
DNPU [54]	65	200	2.1–8.1	yes	4, 8, 16
Origami [46]	65	500	0.44–0.80	no	12
BRein memory [48]	65	400	2.3–6.0	no	2, 3
QUEST [52]	40	330	0.88	yes	1, 4
Zhe Yuan et al. [55]	65	100	0.13–13.3	-	1, 2, 4, 8
DianNao [44]	65	980	0.93	no	16
EIE [45]	45	800	10.49	no	4
Bit Fusion [49]	45	500	-	yes	2, 4, 8, 16
Loom [50]	65	1000	-	yes	2, 4, 8, 16
Yin et al. [47]	65	200	1.27	yes	8–16
Wang et al. [56]	FPGA	200	10.3	yes	1–8

CNN processing relies heavily on MAC units to perform matrix multiplications. Their characteristics and performance directly influence the efficiency of processing. CNN model quantization improves computation efficiency at the edge by enabling CNN accelerators to execute with a variety of precision [57]. Therefore, mix-precision quantization has become prevalent [58–61]. Alternatively, techniques that duplicate MAC units with different bit widths [27] are inefficient in terms of area overhead. Due to this, flexible MAC unit designs have emerged [49–53,62], which are also advocated in our proposal.

Design and evaluation approaches for RISC-V ISA. Tools for developing RISC-V architectures are still in their infancy. However, some simulation and emulation environments have been extended to support RISC-V implementations. As an example, the gem5 cycle-approximate simulator [63] currently supports RISC-V as well as x86, ARM, Alpha, and MIPS. The Spike simulator [64] allows defining possible extensions of the RISC-V ISA and simulating their corresponding instructions. RISC-V ISA improvements can therefore be investigated by using Spike as a research simulator. The riscvOVPsim simulator [65] has been derived from the well-known OVP simulator for prototyping embedded software on an x86 computer by using cross-compilations. The RISC-V Assembler, Simulator, and Runtime (RARS) [66] promotes an easy way for beginners to run assembly code on RISC-V-compliant architectures. In contrast to previous tools, QEMU, which is also open source, is capable of emulating 32-bit and 64-bit RISC-V implementations [67]. Furthermore, RISC-V development boards such as HiFive Unleashed of SiFive and PolarFire of Microchip are also supported by QEMU.

It is possible to design and evaluate RISC-V systems quickly by using the approaches described above, but only a few platforms allow for a more accurate evaluation of design. PULPino [28], for instance, defines a family of platforms that support monocore and multicore processor architectures. The architectures are implemented at RTL level and can be synthesized on FPGAs. A commercial version of one of these prototype architectures can be found in the GAPuino development board [68]. In the corresponding system-on-chip, known as GAP8, eight cores are optimized for vectorized and parallel algorithms along with a CNN accelerator.

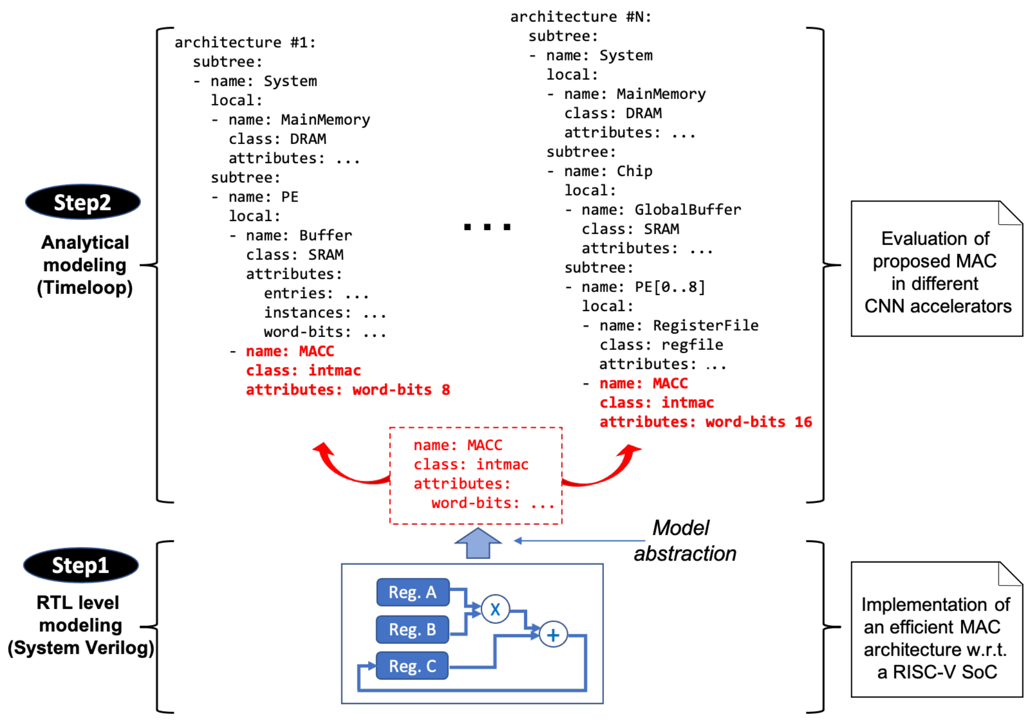
Our approach differs from the aforementioned approaches in that we design a design methodology on top of a RISC-V open-source system platform [28] and the analytical modeling Timeloop/Accelergy framework [29,30]. As a first step, we focus on the design improvement of a hardware component, the MAC unit, which is crucial for the efficient execution of CNNs. In order to accurately quantify the gains enabled by the novel MAC unit, this issue is addressed at the RTL level. Nevertheless, conducting a wide range of system-level evaluations that incorporate this unit at RTL level is a tedious task. As a result, we derive a consistent analytical model to make this possible. This analytical modeling also has the advantage of being easily integrated into other accelerator architectures beyond RISC-V compliant architectures within Timeloop/Accelergy. It is therefore possible to quickly analyze the execution of CNNs by exploring a large mapping space on the considered architectures. By using our bottom-up methodology, we demonstrate how to explore a nontrivial design space by taking advantage of different levels of abstraction.

In the following, we devise a flexible and energy-efficient MAC unit that can be integrated in typical CNN accelerators for a better energy efficiency. In particular, we will compare our solution against some of the aforementioned architectures, namely the GAP8 RISC-V SoC, the Eyeriss and DianNao ASIC CNN accelerators. More details on these three architectures will be provided later in Section 5, which focuses on their evaluation and comparison with our solution.

3. Overview of the Proposed Design Methodology

Our bottom-up design methodology is organized around a flexible and energy-efficient MAC unit. It consists of two main steps shown in Figure 2, as follows.

1. Accurate RTL design [32]. First, we present an efficient MAC unit and assess its energy gain in comparison to the current RISC-V GAP8 MAC unit.
2. Fast analytical design. Secondly, we explore the impact of the above MAC on energy efficiency at the chip or system level, and we develop an abstract model of the above MAC and integrate it into multiple ML accelerator architectures, besides GAP8.


Step2
Analytical modeling (Timeloop)
. . .
Evaluation of proposed MAC in different CNN accelerators
Model abstraction
Step1
RTL level modeling (System Verilog)
Reg. A
Reg. B
Reg. C
X
+
Implementation of an efficient MAC architecture w.r.t. a RISC-V SoC

```
architecture #1:
  subtree:
  - name: System
    local:
    - name: MainMemory
      class: DRAM
      attributes: ...
    subtree:
    - name: PE
      local:
      - name: Buffer
        class: SRAM
        attributes:
          entries: ...
          instances: ...
          word-bits: ...
      - name: MACC
        class: intmac
        attributes: word-bits 8
```

```
architecture #N:
  subtree:
  - name: System
    local:
    - name: MainMemory
      class: DRAM
      attributes: ...
    subtree:
    - name: Chip
      local:
      - name: GlobalBuffer
        class: SRAM
        attributes: ...
      subtree:
      - name: PE[0..8]
        local:
        - name: RegisterFile
          class: regfile
          attributes: ...
        - name: MACC
          class: intmac
          attributes: word-bits 16
```

```
name: MACC
class: intmac
attributes:
  word-bits: ...
```

Figure 2. Overview of the proposed design methodology.

Our first step is to revisit the general design of microprocessors with regard to data quantization requirements. A microprocessor's basic multiply unit allows for only one MAC iteration per clock cycle. With the addition of SIMD units, ML algorithms can perform more MAC iterations per clock cycle. Aside from the additional hardware requirements for SIMD, the operations are usually limited to bit widths of 16 bits and/or 8 bits [37]. From both a power consumption and space perspective, this is inefficient for embedded systems. Thus, we define and implement at the RTL level a flexible MAC unit architecture capable of supporting asymmetric bit-width operations and data bit-width representations ranging from 32 bits to 2 bits. We analyze its power and area gains in the context of RISC-V RI5CY.

The second step involves implementing a Timeloop abstraction [29] for the MAC unit implemented in RTL above. By abstraction, we mean customizing Timeloop architecture models in order to reflect both the structural and nonfunctional properties of the target RTL models. Timeloop is used with Accelergy [30] to evaluate various mappings of CNNs on a particular architecture. The topology of an architecture can be described by a combination of arithmetic units and memory components. Timeloop requires information about the CNN parameters, including its layers, inputs, and trained weights, in order to simulate CNN inference workloads on a given architecture model. Any CNN training framework, such as Keras or PyTorch, can provide such information. By simulating the described architectures, one can determine how well they perform for a given CNN workload and how energy efficient they are. The number of data transfers within the memory hierarchy and the number of arithmetic operations performed are included in this measurement. A general rule of thumb is to establish behavioral monotonicity between the RTL model and its abstract Timeloop counterpart.

Definition 1 (Behavioral monotonicity with regard to a performance metric)**.** *Given M_1 and M_2 two different models of the same system, they satisfy a behavioral monotonicity with regard to a metric μ if the evaluation of μ on their respective behaviors follows the same value tendency, for the same input sequence in both models.*

For instance, let μ denote execution time or power consumption and $i_1 \dots i_n$ represent an input sequence for two models M_1 and M_2 satisfying the behavioral monotonicity property; therefore,

$$\forall k \in 1 \dots n-1, \mu_{M_1}(i_k) \sim \mu_{M_1}(i_{k+1}) \Leftrightarrow \mu_{M_2}(i_k) \sim \mu_{M_2}(i_{k+1})$$

where $\mu_{M_j}(i_k)$ evaluates metric μ on model M_j for input i_k, and $\sim \in \{\geq, \leq\}$.

Our current study focuses on the behavioral monotonicity between RTL and Timeloop/ Accelergy models in terms of energy consumption.

4. A Flexible MAC Unit Design at RTL Level

The MAC unit is developed by decomposing multiplication into 2-bit operands. We use Figure 3a to illustrate this principle on a 4-bit data example.

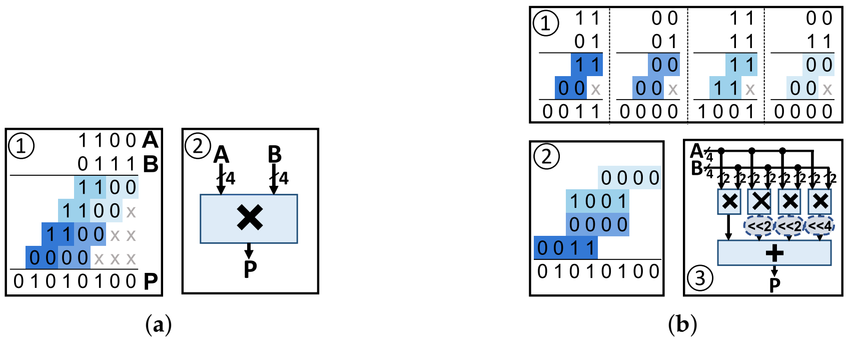


Figure 3. Illustration of 4-bit multiplication [32]: (**a**) General principle. (**b**) Operation decomposition.

In this example, A and B are operands defined by *1100* and *0111* respectively. Multiplications of two binary operands are classically performed as shown in Figure 3a ①, where the former is multiplied by each bit of the latter before the partial products are added. Therefore, *01010100* is obtained. Figure 3a ② shows how this operation can be schematically represented by a simple 4-bit multiplier. In hardware implementations, 16 AND gates are required for bit-by-bit products, and 12 adders consisting of two AND gates, one OR gate, and two XOR gates. It is, however, possible to design such an approach in a variety of ways. Rather than using a 4-bit × 4-bit multiplication, we can decompose it into four independent 2-bit × 2-bit multiplications, as illustrated in Figure 3b ①. In such a case, the final result can be obtained by adding the results of the four 2-bit × 2-bit multipliers with a proper shift, as shown in Figure 3b ②. Lastly, Figure 3b ③ presents the basic blocks necessary to carry out the decomposed multiplication.

4.1. Proposed MAC Architecture

There are three distinct components of the proposed MAC unit: multipliers, adders, and accumulators. This schematic is shown in Figure 4: a line of 2-bit multipliers on top, adders and shifters in the middle, the shift control on the left, the output multiplexer on the right, and the accumulator at the bottom.

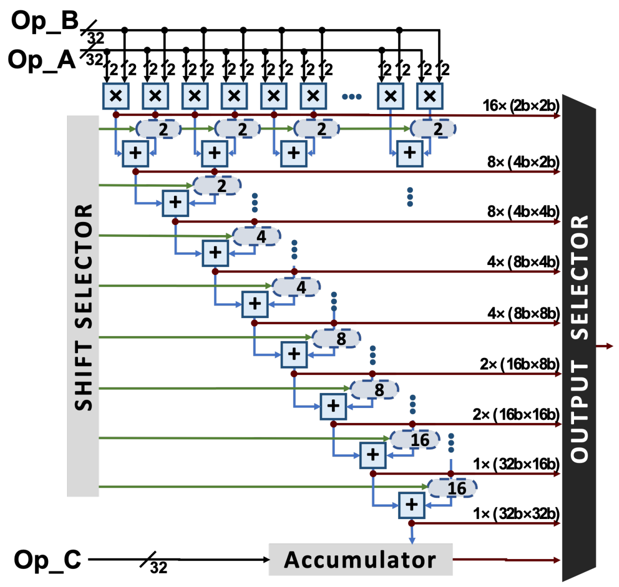


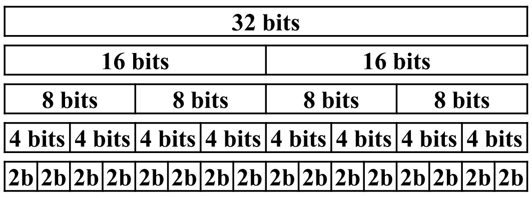
Figure 4. Schematic representation of our MAC unit architecture [32].

The MAC unit of our system consists of 256 independent 2-bit multiplications in order to support 32-bit multiplication. As explained in the previous section, multiplying 4 bits requires four separate 2-bit multiplications $((4/2)^2)$. It follows that, for a 32-bit multiplication, 256 independent 2-bit multiplications are required $((32/2)^2)$.

Using adders with two inputs, the partial products are summed, facilitating multiplier configuration and adaptivity. The addition of partial products requires bit shifting. One of the inputs of each adder is shifted beforehand in order to reduce the number of shifts. Multiplication requires a maximum of eight adder levels, as shown in Figure 4. Multiplication results are retrieved at the outputs of the 2-bit × 2-bit multipliers and at each adder level.

The accumulator performs the MAC operation as a final step. A dedicated register stores the intermediate value of the previous accumulation. The data value is returned to the accumulator via Op_C as depicted in Figure 4.

As shown in Figure 4, the 32-bit op_A and op_B operands are suitable for performing SIMD operations. It is possible to load them with 2-bit × 16-bit, or 4-bit × 8-bit, or 8-bit × 4-bit, or 16-bit × 2-bit operands, as described in Figure 5a. In SIMD mode, the proposed MAC unit is capable of performing parallel operations in one clock cycle, as illustrated in Figure 5b.



(a)

Data bitwidth	MAC operation
32 & 32×16	1
16×16 & 16×8	2
8×8 & 8×4	4
4×4 & 4×2	8
2×2	16

(b)

Figure 5. Details on supported operands and operations: (**a**) Representation of data contained in a 32-bit register for each data bit width. (**b**) Number of available operations for each data bit width.

4.2. Design Evaluation

Our MAC unit is implemented in SystemVerilog, a hardware description language. Simulation and synthesis were performed by using ModelSim and Synopsys Design Compiler, respectively. The proposed MAC architecture is verified through simulation. It is also possible to gather switching activity in order to obtain an accurate estimate of power. This synthesis provides the area and power costs for the selected technology, which is 28 nm FD-SOI at 200 MHz. As evaluation workload, we define a simple program consisting of a few thousand multiplications and additions to be executed for assessing the energy efficiency of MAC units. The evaluated metric is expressed as the number of MAC operations per power consumption unit, referred to as Op/mW. For emulating mixed-precision quantized operands, bit widths of different sizes are used, as illustrated in Figure 5b. Scripts written in Python generate random values for the operands. A SystemVerilog *benchmark* is used to load these data into the 32-bit input registers of the MAC unit. Each bit width is tested 1000 times with different values loaded into the MAC architecture's input registers.

We compare our proposed MAC unit with that of the RI5CY core [28]. This core is selected for two reasons: it is well-known within the RI5CY ISA community and its operations are similar to those targeted by our MAC unit. It has also been developed in SystemVerilog and is freely available on the web at [69]. In its associated MAC unit, there are five types of multipliers, each representing a specific bit precision, as illustrated in Figure 6. There is a similar parallelization capability to that of Figure 5b. For simulation and synthesis with FD-SOI technology from 28 nm to 200 MHz, ModelSim and Synopsys Design Compiler are also used. In the RI5CY core, the native MAC unit does not support the 4-bit:4-bit and 2-bit:2-bit configurations. To include them, we rely on the MAC unit in [27]. A key difference between this unit and our adaptive MAC design is that it employs hardware redundancy, i.e., each precision level requires a dedicated multiplier.

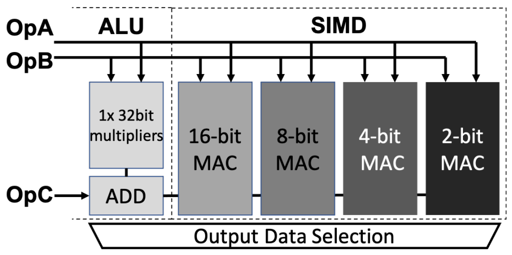


Figure 6. Redundant MAC unit design in the native RI5CY core.

Area estimation. Initially, the architecture synthesis provides the area surface value of 9930 μm^2. Figure 7 shows the distribution of space among the various parts of the MAC unit. Adders and shifters make up 75% of the circuit, while the 256 multipliers make up 20%. In the circuit, the accumulator occupies approximately 1% of the area, while the connections and output multiplexers occupy the remaining 4%. During synthesis, weak optimization capabilities result in the adders and offsets taking up 75% of the space. Figure 4 illustrates the data capture at each level of the adder towards the output. As a result of this capture, it is possible to retrieve the results of multiplication operations involving less than 32 bits. Due to this limitation, synthesis tools cannot perform optimizations consisting of reducing the number of logic gates.

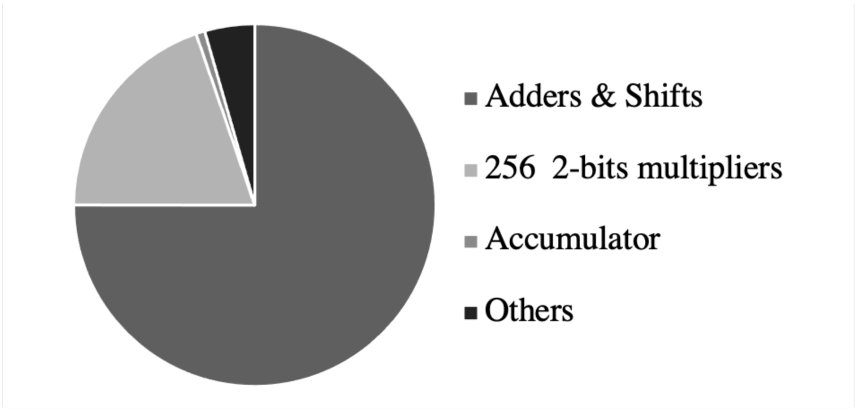


Figure 7. The distribution of the surface occupied by the main functions of our MAC unit.

In the RI5CY core, the surface area of the native MAC unit is 10,830 μm^2. Based on our proposal, we are able to reduce the size of this unit by 10%. As the MAC unit of RI5CY occupies approximately 40% of a core's surface, this surface reduction is beneficial.

Power estimation. The static and dynamic power consumption of the two MAC architectures can be seen in Figure 8. The dynamic power of our solution increases as the data bit width increases. As a matter of fact, our solution activates only the hardware required for a particular bit width. By using the divide-and-conquer principle, we activate only the logic required to perform the multiplication proportional to the operand bit width.

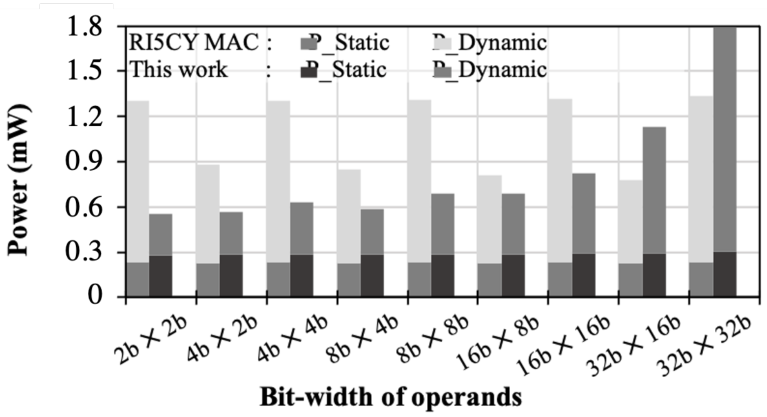


Figure 8. Power estimation with synopsys design compiler, based on our considered testbench.

The amount of power dissipated is directly related to the number of logic gates activated. As a consequence, 32-bit operations that utilize the multiplier's logic consume more energy. When compared with the native RI5CY design, 16 bits and 32 bits operations consume more power. We believe that the less optimal placement of logic components during synthesis is the cause of this. Due to the fact that our MAC unit consists of several connections distributed across the circuit, there is a limited amount of space for optimization. It should be noted that despite a smaller area, static power is slightly higher than native RI5CY. In the MAC units, we observe two levels of dynamic power. Consequently, the higher level corresponds to operands of the same bit width, while the lower level corresponds to operands with asymmetric bit widths. By adding zeros, asymmetric bit widths can be accommodated. The dynamic power dissipated by our newly designed MAC unit is 25% less than that of RI5CY. The static power increases by about 21% with our proposal. However, it represents a smaller portion of the global power consumption dissipated by the design, as shown in Figure 8.

Energy-efficiency evaluation. A comparison of the energy efficiency of the two solutions is shown in Figure 9. Both MAC architectures benefit from the lowest bit widths in terms of energy efficiency. Energy efficiency decreases when the data bit width exceeds 16 bits. There is a 50% improvement in the energy efficiency of the proposed MAC unit over the native design of the RI5CY core.

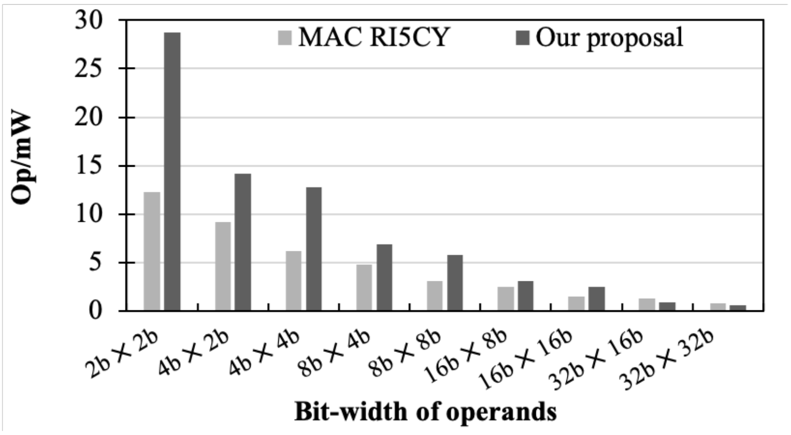


Figure 9. Energy efficiency with regard to dynamic power for different MAC operand bit widths.

Compared with the native unit, the novel MAC unit presented above shows notable energy efficiency for the RI5CY core. The results confirm the benefits of the adaptive nature of the proposed RI5CY MAC unit by reducing the area induced by the redundant design used in the RI5CY MAC unit (see Figure 6). We note that another MAC unit with a similar design principle has been developed for RI5CY [70].

5. Abstraction toward Analytical Modeling of Our MAC Unit

The GAPuino board uses the GAP8 chip to execute ML algorithms efficiently. In order to design efficient ML-dedicated architectures, it is critical to take into account the memory and arithmetic units that perform MAC operations. Our Timeloop modeling therefore heavily relies on these components. Figure 10a illustrates a model of the GAP8 architecture.

Each component of the model can be assigned the desired technology node. Currently, Timeloop supports 40-nm and 65-nm technologies. There are several characteristics that are taken into account when modeling memories, such as the geometry of the memory blocks (width and height), the data size, such as the "`word-bits`" attribute, and throughput. Arithmetic units are characterized by their type of operation, size, and type of components. The main modifications applied to devise the abstract model of the GAP8 in Timeloop concern the overall memory architecture as well as the two MAC unit variants, namely the native one and our proposal. As illustrated in Figure 10a, two intermediate memory levels are modeled between the PEs and the main memory. Memory parameters include `word-bits`, which are configured according to the precision level in data representation, e.g., from 2 bits to 32 bits, including mixed precision representations. We assume different layers of a CNN can have distinct data representations, so that a good compromise in terms of overall CNN energy efficiency becomes possible (see later in Section 5.2). Then, the remaining memory attributes are adjusted accordingly so that the overall memory geometry is unchanged for the different data precision levels (otherwise, the memory would be larger and therefore more energy consuming for 32 bits than for 2 bits). On the other hand, parallel MAC units are created within each PE according to the supported bit precision level (see Figure 5b). In the case of data bit widths including 32 bits, each PE will contain one MAC unit, while for all other bit widths, from 2 to 16 parallel MAC units may be used.

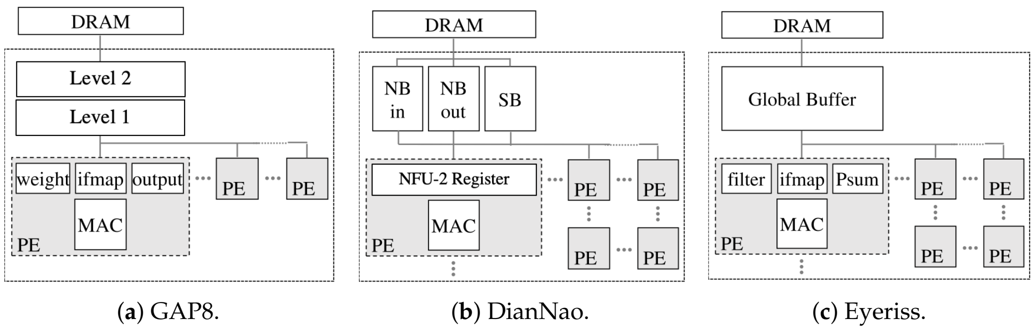


Figure 10. Considered accelerator architecture template models in Timeloop.

5.1. Timeloop-Based Modeling

In order for a high-level modeling tool such as Timeloop to be useful, it must be capable of producing relevant results based on the corresponding hardware implementation, i.e., behavioral monotonicity. We first measure the power consumption of GAP8 on the GAPuino compute board to meet this requirement. Through the use of a CNN workload (here the MNIST CNN), we examine whether the energy consumption trends obtained with the Timeloop GAP8 model and the GAPuino board are comparable.

Figure 11 compares the energy consumption of the Timeloop model and the GAPuino board. The considered workloads consist of convolution layers because they are the most energy-consuming parts of a CNN. We replicate such layers from 1 to 3 on the one hand, and we vary the number of filters inside each layer on the other hand. Hence, "NxConv F1-F2-...-FN" denotes a workload with N convolution layers, where the kth layer contains F_k filters.

The energy consumption of the GAPuino is higher, between 10 mJ and a few joules, while that of the Timeloop model is between 1.8 µJ and 50 µJ. Energy consumption trends are generally similar across all CNN execution configurations.

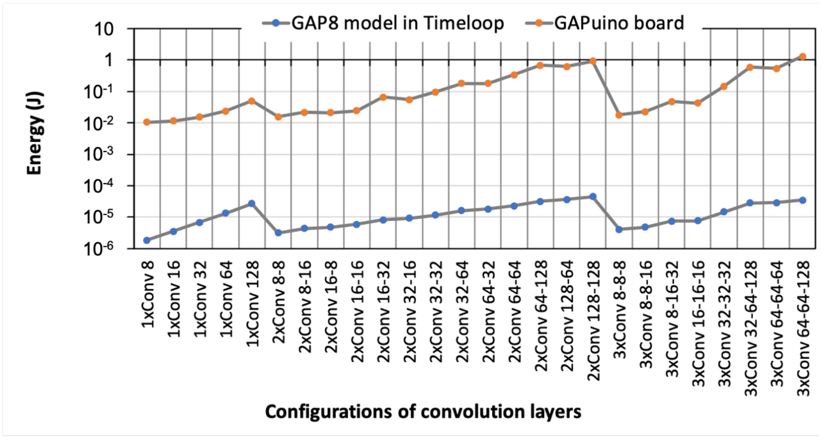


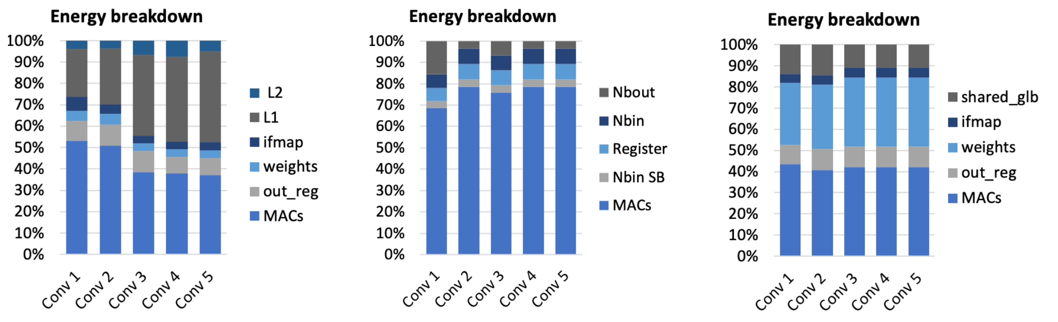
Figure 11. Energy-based behavioral monotonicity: GAP8 model in Timeloop vs actual GAP8 SoC on the GAPuino development board.

Based on the similarity in trends, the Timeloop model for GAP8 architecture appears to be valid for establishing behavioral monotonicity with regard to energy consumption.

Energy breakdown for on-chip components. Prior to investigating how our MAC unit might be integrated into the three architectures, it is critical to assess how efficient optimization can be achieved. We analyze the breakdown of energy consumption of the on-chip microarchitecture components based on the native MAC unit model in Timeloop for each accelerator. Figure 10 illustrates the architectures under consideration. We modified the templates provided in Timeloop only marginally for Eyeriss and DianNao. In contrast, we developed a model for GAP8.

The Eyeriss architecture (see Figure 10c) is composed of processing elements. Each PE contains three registers containing the weights (filters) of CNN filters, the input data (ifmaps) and the partial sums (psums) computed by the MAC operations. An intermediate SRAM buffer connects each PE to the DRAM main memory for the purpose of storing data. Each PE in the DianNao architecture has a single register to store weight data (see Figure 10b). Here, a PE receives data from three shared memories, NBin, NBout, and SB, which contain input feature maps, output feature maps, and weights, respectively. DianNao and Eyeriss organize PEs as two-dimensional matrices. The GAP8 architecture (note that the CNN accelerator of the GAP8 chip is abstracted here in the same way as the other cores due to the lack of information on its corresponding microarchitecture) consists of parallel PEs arranged in rows (see Figure 10a). The PEs in GAP8 are similarly composed of three registers that contain the CNN input filter weights, the input data, and the MAC output data. DRAM main memory is accessed by the PEs via Level 1 (64 KB) and Level 2 (512 KB) scratchpad memory levels.

For each accelerator, we execute the AlexNet CNN [6] and report the energy consumption distribution in Figure 12.



(**a**) GAP8. (**b**) DianNao. (**c**) Eyeriss.

Figure 12. Energy breakdown based on considered accelerators models.

With the exception of the main memory, the obtained results indicate that the on-chip energy consumption of the MAC units is not negligible. Accordingly, this is consistent with the CNN parameter complexity mentioned in Table 1. Eyeriss and GAP8 consume approximately 41% of energy for their respective MAC unit models, while DianNao consumes approximately 75%. This distribution of energy indicates that the MAC unit is a viable candidate for improvement in order to increase energy efficiency.

Figure 12 only shows the energy breakdown of on-chip components. Nevertheless, we can point out that the energy consumption of the main memory (which is typically located off-chip) represents approximately 60%, 95%, and 23% of the entire accelerator model, respectively, for Eyeriss, DianNao, and GAP8.

5.2. Evaluation with Regard to GAP8 RISC-V Architecture

Having assessed the relevance of the above analytical model, we now assess its global impact on the GAP8 RISC-V architecture model. The integration of our proposed MAC unit into this architecture is compared with that of the native MAC unit of GAP8 (Figures 13 and 14).

As a workload, we use AlexNet CNN in the next experiment. Figure 13 shows the energy efficiency of the different convolution layers with different uniform bit precision. CNN layers perform different operations and manipulate data in different ways. The obtained results indicate that energy efficiency varies across the considered configurations. In particular, the energy consumed by MAC units increases as bit precision increases. Especially interesting is the fact that the energy efficiency of convolution layers varies with bit precision.

It is worth mentioning that Conv4 is the most expensive layer for 16 and 32-bit precision (see Figure 13d), whereas Conv2 is the most expensive layer for 8-bit precision (see Figure 13b). These variations suggest that mixing different bit precision levels for different CNN layers may improve the efficiency of a network by selecting the most appropriate precision for each layer.

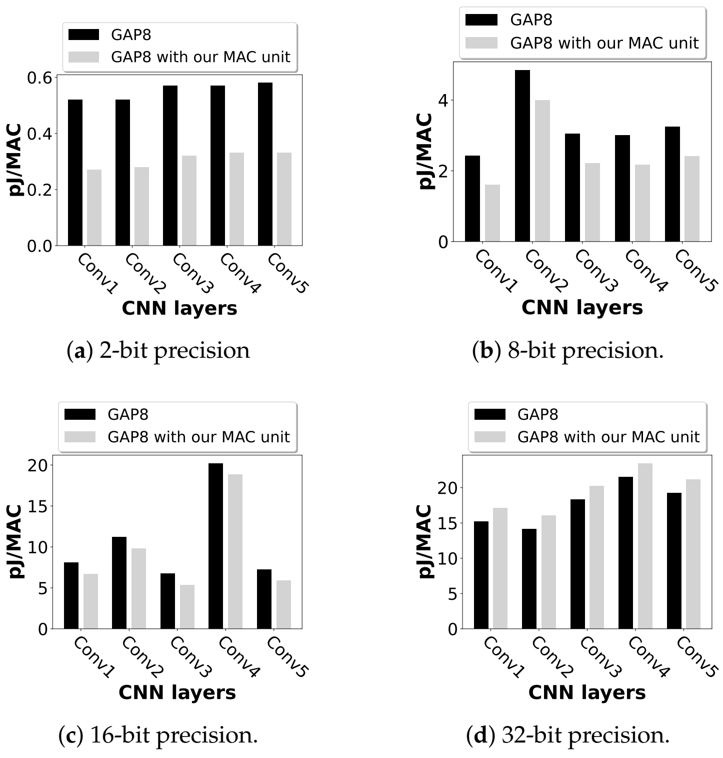


Figure 13. Details of GAP8 architecture energy efficiency for different data accuracy.

Figure 14 illustrates the performance of some alternative CNN models combining 2-bit precision levels. As a result of combining 8-bit and 16-bit precision, CNN execution is more energy efficient than it would be with 16-bit precision alone (see Figure 14a). Furthermore, the previous uniform 32-bit representation (see Figure 13d) can be improved by combining it with the 16-bit precision, as shown in Figure 14b.

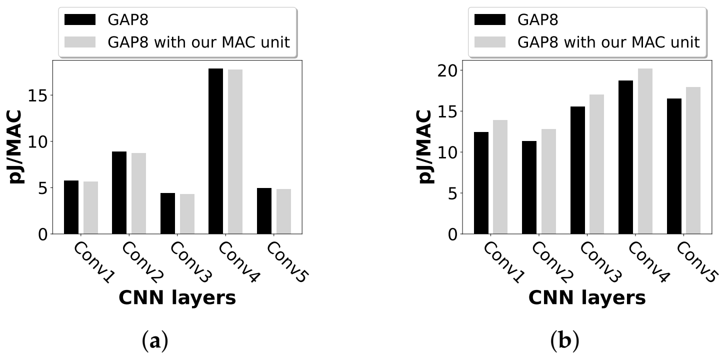


Figure 14. Details of GAP8 architecture energy efficiency for different data accuracy: (**a**) Eight- and 16-bit precision; (**b**) 16- and 32-bit precision.

Based on the above evaluations, the Timeloop model of our MAC unit appears to be less energy efficient than the original MAC unit of GAP8 for high bit precision, such as 32 bits. In fact, this tendency was inherited from the RTL-level implementation we used in our evaluation (see Section 4.2), where we observed a higher power consumption for our MAC unit than GAP8's, when using 32-bit precision. In order to maintain design consistency (here, behavioral monotonicity), we calibrated our MAC unit's Timeloop model with the same overhead power consumption. Our proposal may be made more efficient by means of applying various power optimization techniques, either offered by standard physical synthesis design flows from RTL or arithmetic unit-specific techniques. It is also possible to mitigate its lower energy efficiency compared to 32 bits by mixing different precision levels. This is observed in the next section for the MAC unit in DianNao.

Meanwhile, a noteworthy insight is that the proposed MAC unit improves overall energy efficiency with regard to smaller bit widths. Because a majority of CNN accelerators are designed to handle smaller bit widths, as illustrated in the samples in Table 2, this is a significant advantage. None of the popular CNN accelerators reported in this table feature 32-bit width. The first third of these accelerators supports a maximum 8-bit width, whereas the remaining accelerators support up to 16-bit width. The support for 32-bit width may further come in handy as per-layer quantization is increasingly popular, and offering the opportunity to avoid quantization for specific layers can help minimize accuracy loss.

5.3. Evaluation with Regard to Further Architectures

Beyond the GAP8 model, we now evaluate the Eyeriss and DianNao models in Timeloop to assess their energy efficiency when integrating our MAC into their respective architectures. For this purpose, we compare configurations with each accelerator's native MAC unit against our proposed MAC unit. In addition, we assess the effects of the most appropriate mapping choices determined by Timeloop on PE utilization ratios. Note that for the sake of convenience, the results presented in the sequel rely on system configurations including 96 PEs. Larger configurations could be explored, of course.

Impact of our MAC unit on energy-efficiency at chip level. Figure 15 summarizes the energy efficiency of AlexNet CNNs executed on the three target architectures. Eyeriss and DianNao show the greatest improvement, of 32% and 19%, respectively. This is a result of our MAC unit's ability to parallelize operations on data, especially with reduced data precision. Neither DianNao nor Eyeriss have native MAC units that are capable of handling more than one MAC operation at a time.

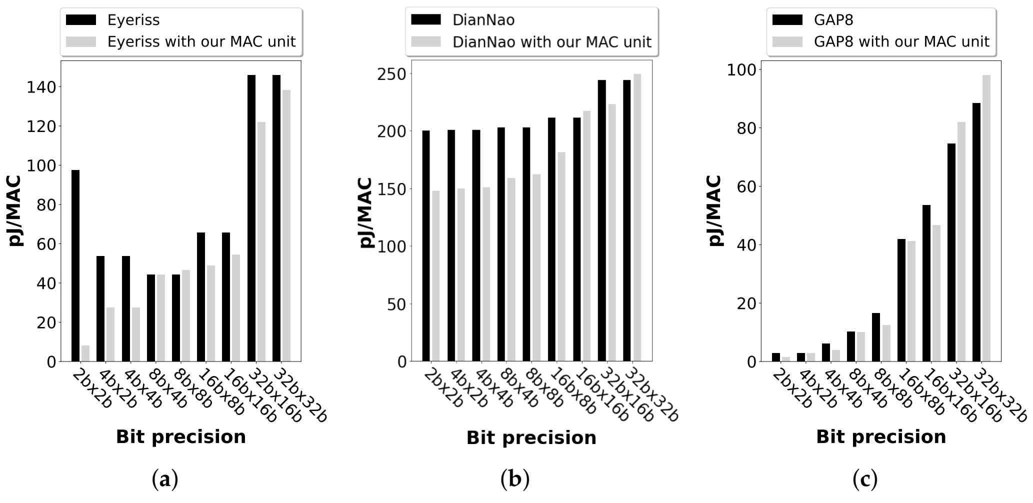


Figure 15. Impact of our MAC unit on the energy efficiency of architectures: (**a**) The Eyeriss architecture. (**b**) The DianNao architecture. (**c**) The GAP8 architecture.

Our MAC unit for the GAP8 architecture is more efficient for reduced bit precision. It is in accordance with the observations made in Section 4. Our MAC unit reduces energy consumption per MAC operation by 23% as MAC precision increases from 2 bits to 16 bits. When operating at 32-bits precision, our MAC unit increases the power consumption by 10% for both DianNao and GAP8.

Impact on the number of PEs utilized in the best CNN mappings. Figure 16 shows the PE utilization ratios obtained from the best CNN mappings on the three architectures, with and without our proposed MAC unit. In other words, it represents the number of PEs that are required for the most energy-efficient execution of CNNs.

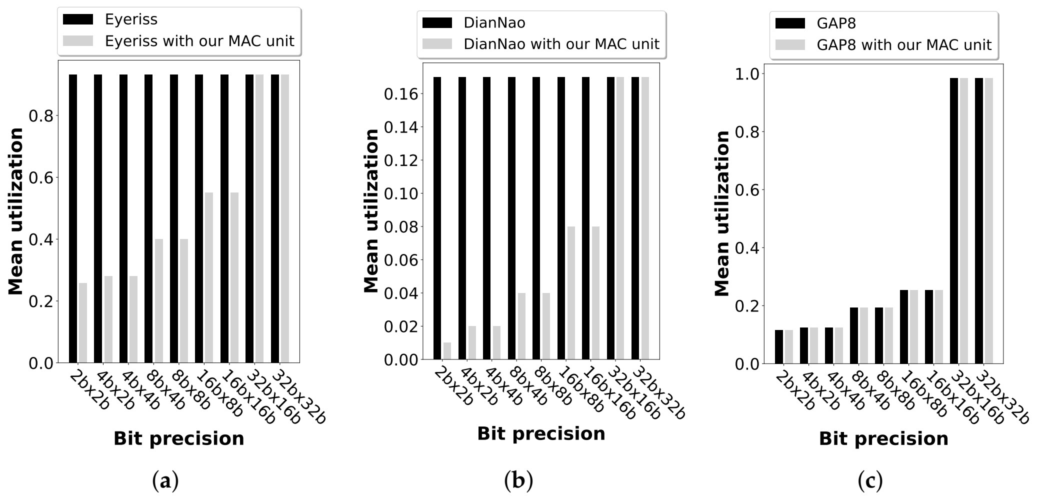


Figure 16. Comparison of PE utilization ratios of the three architectures with and without our MAC unit: (**a**) The Eyeriss architecture. (**b**) The DianNao architecture. (**c**) The GAP8 architecture.

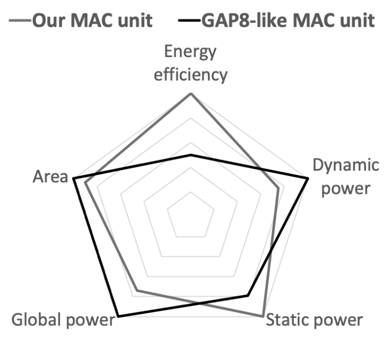
When using Eyeriss and DianNao's native single-precision MAC units, PE utilization ratios are greater than 80%. The integration of our proposed MAC unit into these architectures results in variable PE utilization ratios according to data precision. Particularly for smaller data bit widths, PE utilization ratios are low. This is due to the parallel MAC operations supported by GAP8 MAC units (including the native MAC unit and the proposed variant), inside each PE. Therefore, by distributing the workload and data over fewer PEs, the Timeloop mapper minimizes overall energy consumption.

As illustrated in Figure 16c, it is interesting to note that the PE utilization ratio for the GAP8 system is similar regardless of whether we consider either the native MAC unit or our proposed MAC unit. This is due to the fact that both MAC units support similar parallelism in MAC operations.

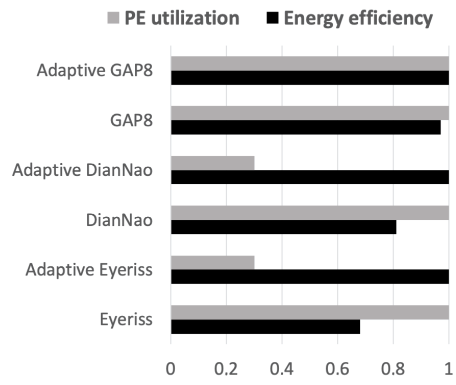
6. Concluding Remarks

This paper introduced a two-step bottom-up design methodology for assessing the energy efficiency of CNNs executed on accelerator architectures. An RTL design of a flexible and energy-efficient MAC unit is the first step in this methodology. In this way, we are able to efficiently handle CNNs with a large number of MAC operations. Our solution uses an open-source RISC-V core, also found in commercial chips. In order to support multiple quantization modes simultaneously for different CNN convolution layers, we developed a mixed-precision MAC unit. This implementation reduces the dynamic power and area by approximately 25% and 10% respectively, when compared with the native MAC unit of the considered RI5C-V processor, which is similar to that of the GAP8 SoC. Figure 17a summarizes the benefits of our solution with respect to the main figures of merit. For the energy-efficiency metric, the higher the value, the better, whereas for the power consumption and area footprint metric, the lower the value, the better. With the exception of static power consumption, our solution is generally more efficient than the existing RISC-V MAC unit design. Furthermore, because the static power dissipation in these designs is smaller than the dynamic one, we obtain an overall improvement in power consumption.

As a second step, we abstracted the devised MAC component by using Timeloop/Accelergy. At the system level, we tested the energy efficiency of the enhanced MAC unit when integrated with typical architectures such as GAP8, Eyeriss, and DianNao. The results confirmed that system-level energy efficiency improved for both uniform and mixed precision data bit widths from 2 bits to 16 bits. Figure 17b summarizes the main figures of merit when evaluating our MAC unit at the system level when integrated into the three CNN-dedicated architectures. In comparison with Eyeriss and DianNao, we have observed notable gains in energy efficiency as a result of the higher execution parallelism of our MAC unit. As a result, the number of processing units required for execution is drastically reduced, as illustrated by the PE utilization ratio (here, the lower the utilization ratio, the better for power savings). However, the benefits observed at the MAC unit level are moderate compared to GAP8. The native MAC unit of the latter also supports execution parallelism, while being less area efficient.



(**a**)



(**b**)

Figure 17. Normalized figures of merit comparing our proposal with regard to considered approaches: (**a**) MAC unit-level evaluation and comparison with regard to GAP8-like MAC unit. (**b**) System-level evaluation and comparison with regard to Eyeriss, DianNao, and GAP8 by using Timeloop. Here, "adaptive" denotes a design variant wherein our MAC unit replaces the original unit in a system.

As demonstrated in our study, the proposed bottom-up design methodology can facilitate the evaluation of CNN implementations on RISC-V and other architectures in the future.

Author Contributions: Conceptualization, G.D., G.S. and A.G.; methodology, G.D. and A.G.; software, G.D.; validation, G.D., G.S. and A.G.; writing—original draft preparation, G.D., G.S. and A.G.; writing—review and editing, A.G.; supervision, G.S. and A.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to thank Maxime France-Pillois for his helpful participation to early discussion on some parts of the present work.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MAC	Multiply-and-Accumulate
CNN	Convolutional Neural Network
DNN	Deep Neural Network
RTL	Register Transfer Level
TLM	Transaction Level Modeling

ISA	Instruction Set Architecture
ASIC	Application-Specific Integrated Circuit
FPGA	Field-Programmable Gate Array
SIMD	Single Instruction Multiple Data
ML	Machine Learning
SoC	System-on-Chip
PE	Processing Element

References

1. Mahdavinejad, M.S.; Rezvan, M.; Barekatain, M.; Adibi, P.; Barnaghi, P.; Sheth, A.P. Machine learning for internet of things data analysis: A survey. *Digit. Commun. Netw.* **2018**, *4*, 161–175. [CrossRef]
2. Li, H.; Ota, K.; Dong, M. Learning IoT in Edge: Deep Learning for the Internet of Things with Edge Computing. *IEEE Netw.* **2018**, *32*, 96–101. [CrossRef]
3. Moons, B.; Goetschalckx, K.; Van Berckelaer, N.; Verhelst, M. Minimum energy quantized neural networks. In Proceedings of the 2017 51st Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, USA, 29 October–1 November 2017; pp. 1921–1925. [CrossRef]
4. Pimentel, A.D. Exploring Exploration: A Tutorial Introduction to Embedded Systems Design Space Exploration. *IEEE Des. Test* **2017**, *34*, 77–90. [CrossRef]
5. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *CoRR* **2017**, abs/1704.04861. Available online: https://arxiv.org/abs/1704.04861 (accessed on 4 January 2023).
6. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
7. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.E.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going Deeper with Convolutions. *CoRR* **2014**, abs/1409.4842. Available online: https://arxiv.org/abs/1409.4842 (accessed on 4 January 2023).
8. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. *CoRR* **2015**, abs/1512.03385. Available online: https://arxiv.org/abs/1512.03385 (accessed on 4 January 2023).
9. Karen Simonyan, A.Z. Very Deep Convolutional Networks for Large-Scale Image Recognition. In Proceedings of the 3rd International Conference on Learning Representations (ICLR), San Diego, CA, USA, 7–9 May 2015. Available online: http://arxiv.org/abs/1409.1556 (accessed on 4 January 2023).
10. Gamatié, A.; Devic, G.; Sassatelli, G.; Bernabovi, S.; Naudin, P.; Chapman, M. Towards Energy-Efficient Heterogeneous Multicore Architectures for Edge Computing. *IEEE Access* **2019**, *7*, 49474–49491. [CrossRef]
11. Apvrille, L.; Bécoulet, A. Prototyping an Embedded Automotive System from its UML/SysML Models. In Proceedings of the Embedded Real Time Software and Systems (ERTS'2012), Toulouse, France, 29 January–1 February 2012.
12. Dekeyser, J.L.; Gamatié, A.; Etien, A.; Ben Atitallah, R.; Boulet, P. Using the UML Profile for MARTE to MPSoC Co-Design. In Proceedings of the First International Conference on Embedded Systems Critical Applications (ICESCA'08), Tunis, Tunisia, May 2008.
13. Quadri, I.R.; Gamatié, A.; Boulet, P.; Dekeyser, J.L. Modeling of Configurations for Embedded System Implementations in MARTE. In Proceedings of the 1st workshop on Model Based Engineering for Embedded Systems Design-Design, Automation and Test in Europe (DATE 2010), Dresden, Germany, 8–12 March 2010.
14. Yu, H.; Gamatié, A.; Rutten, É.; Dekeyser, J. Safe design of high-performance embedded systems in an MDE framework. *Innov. Syst. Softw. Eng.* **2008**, *4*, 215–222. [CrossRef]
15. Breuer, M.; Friedman, A.; Iosupovicz, A. A Survey of the State of the Art of Design Automation. *Computer* **1981**, *14*, 58–75. [CrossRef]
16. Binkert, N.; Beckmann, B.; Black, G.; Reinhardt, S.K.; Saidi, A.; Basu, A.; Hestness, J.; Hower, D.R.; Krishna, T.; Sardashti, S.; et al. The Gem5 Simulator. *SIGARCH Comput. Archit. News* **2011**, *39*, 1–7. [CrossRef]
17. Butko, A.; Gamatié, A.; Sassatelli, G.; Torres, L.; Robert, M. Design Exploration for next Generation High-Performance Manycore On-chip Systems: Application to big.LITTLE Architectures. In *Proceedings of the ISVLSI: International Symposium on Very Large Scale Integration*; IEEE: Montpellier, France, 2015; pp. 551–556. [CrossRef]
18. Nocua, A.; Bruguier, F.; Sassatelli, G.; Gamatié, A. ElasticSimMATE: A fast and accurate gem5 trace-driven simulator for multicore systems. In Proceedings of the 12th International Symposium on Reconfigurable Communication-centric Systems-on-Chip, ReCoSoC 2017, Madrid, Spain, 12–14 July 2017; pp. 1–8. [CrossRef]
19. Ghenassia, F. *Transaction-Level Modeling with SystemC: TLM Concepts and Applications for Embedded Systems*; Springer: New York, NY, USA, 2006.
20. Latif, K.; Selva, M.; Effiong, C.; Ursu, R.; Gamatie, A.; Sassatelli, G.; Zordan, L.; Ost, L.; Dziurzanski, P.; Indrusiak, L.S. Design Space Exploration for Complex Automotive Applications: An Engine Control System Case Study. In Proceedings of the 2016 Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools, Prague, Czech Republic, 18 January 2016; RAPIDO '16; Association for Computing Machinery: New York, NY, USA, 2016. [CrossRef]

21. Mello, A.; Maia, I.; Greiner, A.; Pecheux, F. Parallel simulation of systemC TLM 2.0 compliant MPSoC on SMP workstations. In Proceedings of the 2010 Design, Automation Test in Europe Conference Exhibition (DATE 2010), Dresden, Germany, 8–12 March 2010; pp. 606–609. [CrossRef]
22. Schirner, G.; Dömer, R. Quantitative Analysis of the Speed/Accuracy Trade-off in Transaction Level Modeling. *ACM Trans. Embed. Comput. Syst.* **2009**, *8*, 1–29. [CrossRef]
23. Russo, E.; Palesi, M.; Monteleone, S.; Patti, D.; Lahdhiri, H.; Ascia, G.; Catania, V. Exploiting the Approximate Computing Paradigm with DNN Hardware Accelerators. In Proceedings of the 2022 11th Mediterranean Conference on Embedded Computing (MECO), Budva, Montenegro, 7–10 June 2022; pp. 1–4. [CrossRef]
24. Corvino, R.; Gamatié, A.; Geilen, M.; Józwiak, L. Design space exploration in application-specific hardware synthesis for multiple communicating nested loops. In Proceedings of the 2012 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation, SAMOS XII, Samos, Greece, 16–19 July 2012; pp. 128–135. [CrossRef]
25. An, X.; Boumedien, S.; Gamatié, A.; Rutten, E. CLASSY: A Clock Analysis System for Rapid Prototyping of Embedded Applications on MPSoCs. In Proceedings of the 15th International Workshop on Software and Compilers for Embedded Systems, St. Goar, Germany, 27–28 June 2011; SCOPES '12; Association for Computing Machinery: New York, NY, USA, 2012; pp. 3–12. [CrossRef]
26. Caliri, G.V. Introduction to analytical modeling. In Proceedings of the 26th International Computer Measurement Group Conference, Orlando, FL, USA, 10–15 December 2000; pp. 31–36.
27. Garofalo, A.; Tagliavini, G.; Conti, F.; Rossi, D.; Benini, L. XpulpNN: Accelerating Quantized Neural Networks on RISC-V Processors Through ISA Extensions. In Proceedings of the 2020 Design, Automation Test in Europe Conference Exhibition (DATE), Grenoble, France, 9–13 March 2020; pp. 186–191. [CrossRef]
28. Gautschi, M.; Schiavone, P.D.; Traber, A.; Loi, I.; Pullini, A.; Rossi, D.; Flamand, E.; Gürkaynak, F.K.; Benini, L. Near-Threshold RISC-V Core With DSP Extensions for Scalable IoT Endpoint Devices. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2017**, *25*, 2700–2713. [CrossRef]
29. Parashar, A.; Raina, P.; Shao, Y.S.; Chen, Y.H.; Ying, V.A.; Mukkara, A.; Venkatesan, R.; Khailany, B.; Keckler, S.W.; Emer, J. Timeloop: A Systematic Approach to DNN Accelerator Evaluation. In Proceedings of the 2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Madison, WI, USA, 24–26 March 2019; pp. 304–315. [CrossRef]
30. Wu, Y.N.; Emer, J.S.; Sze, V. Accelergy: An Architecture-Level Energy Estimation Methodology for Accelerator Designs. In Proceedings of the 2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Westminster, CO, USA, 4–7 November 2019; pp. 1–8. [CrossRef]
31. Delobelle, T.; Péneau, P.Y.; Gamatié, A.; Bruguier, F.; Senni, S.; Sassatelli, G.; Torres, L. MAGPIE: System-level Evaluation of Manycore Systems with Emerging Memory Technologies. In Proceedings of the 2nd International Workshop on Emerging Memory Solutions-Technology, Manufacturing, Architectures, Design and Test at Design Automation and Test in Europe (DATE'2017), Lausanne, Switzerland, 27–31 March 2017.
32. Devic, G.; France-Pillois, M.; Salles, J.; Sassatelli, G.; Gamatié, A. Highly-Adaptive Mixed-Precision MAC Unit for Smart and Low-Power Edge Computing. In Proceedings of the 2021 19th IEEE International New Circuits and Systems Conference (NEWCAS), Toulon, France, 13–16 June 2021; pp. 1–4. [CrossRef]
33. Dally, W.J.; Turakhia, Y.; Han, S. Domain-Specific Hardware Accelerators. *Commun. ACM* **2020**, *63*, 48–57. [CrossRef]
34. Peccerillo, B.; Mannino, M.; Mondelli, A.; Bartolini, S. A survey on hardware accelerators: Taxonomy, trends, challenges, and perspectives. *J. Syst. Archit.* **2022**, *129*, 102561. [CrossRef]
35. Gwennap, L. Esperanto maxes out RISC-V: High-End Maxion CPU Raises RISC-V Performance Bar. *Microprocess. Rep. Tech. Rep.* **2018**. Available online: https://www.esperanto.ai/wp-content/uploads/2018/12/Esperanto-Maxes-Out-RISC-V.pdf (accessed on 4 January 2023).
36. Conti, F.; Benini, L. A ultra-low-energy convolution engine for fast brain-inspired vision in multicore clusters. In Proceedings of the 2015 Design, Automation Test in Europe Conference Exhibition (DATE), Grenoble, France, 9–13 March 2015; pp. 683–688. [CrossRef]
37. DSP for Cortex-M. Available online: https://developer.arm.com/architectures/instruction-sets/dsp-extensions/dsp-for-cortex-m (accessed on 28 January 2012).
38. Venieris, S.I.; Bouganis, C.S. fpgaConvNet: A Framework for Mapping Convolutional Neural Networks on FPGAs. In Proceedings of the 2016 IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), Washington, DC, USA, 1–3 May 2016; pp. 40–47. [CrossRef]
39. Umuroglu, Y.; Fraser, N.J.; Gambardella, G.; Blott, M.; Leong, P.; Jahre, M.; Vissers, K. FINN: A Framework for Fast, Scalable Binarized Neural Network Inference. In Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, CA, USA, 22–24 February 2017; FPGA '17; Association for Computing Machinery: New York, NY, USA, 2017; pp. 65–74. [CrossRef]
40. Dundar, A.; Jin, J.; Martini, B.; Culurciello, E. Embedded Streaming Deep Neural Networks Accelerator With Applications. *IEEE Trans. Neural Netw. Learn. Syst.* **2017**, *28*, 1572–1583. [CrossRef] [PubMed]
41. Dundar, A.; Jin, J.; Gokhale, V.; Martini, B.; Culurciello, E. Memory access optimized routing scheme for deep networks on a mobile coprocessor. In Proceedings of the 2014 IEEE High Performance Extreme Computing Conference (HPEC), Waltham, MA, USA, 9–11 September 2014; pp. 1–6. [CrossRef]

42. Chen, Y.; Emer, J.S.; Sze, V. Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks. In Proceedings of the ISCA. IEEE Computer Society, Seoul, Republic of Korea, 18–22 June 2016; pp. 367–379.
43. Chen, Y.H.; Yang, T.J.; Emer, J.; Sze, V. Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices. *IEEE J. Emerg. Sel. Top. Circuits Syst.* **2019**, *9*, 292–308. [CrossRef]
44. Chen, T.; Du, Z.; Sun, N.; Wang, J.; Wu, C.; Chen, Y.; Temam, O. DianNao: A Small-Footprint High-Throughput Accelerator for Ubiquitous Machine-Learning. In Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems, Salt Lake City, UT, USA, 1–5 March 2014; ASPLOS '14; Association for Computing Machinery: New York, NY, USA, 2014; pp. 269–284. [CrossRef]
45. Han, S.; Liu, X.; Mao, H.; Pu, J.; Pedram, A.; Horowitz, M.A.; Dally, W.J. EIE: Efficient Inference Engine on Compressed Deep Neural Network. *CoRR* **2016**, abs/1602.01528. Available online: https://arxiv.org/abs/1602.01528 (accessed on 4 January 2023).
46. Cavigelli, L.; Gschwend, D.; Mayer, C.; Willi, S.; Muheim, B.; Benini, L. Origami: A Convolutional Network Accelerator. *CoRR* **2015**, abs/1512.04295. Available online: https://arxiv.org/pdf/1512.04295.pdf (accessed on 4 January 2023).
47. Yin, S.; Ouyang, P.; Tang, S.; Tu, F.; Li, X.; Liu, L.; Wei, S. A 1.06-to-5.09 TOPS/W reconfigurable hybrid-neural-network processor for deep learning applications. In Proceedings of the 2017 Symposium on VLSI Circuits, Kyoto, Japan, 5–8 June 2017; pp. C26–C27. [CrossRef]
48. Ando, K.; Ueyoshi, K.; Orimo, K.; Yonekawa, H.; Sato, S.; Nakahara, H.; Ikebe, M.; Asai, T.; Takamaeda-Yamazaki, S.; Kuroda, T.; et al. BRein memory: A 13-layer 4.2 K neuron/0.8 M synapse binary/ternary reconfigurable in-memory deep neural network accelerator in 65 nm CMOS. In Proceedings of the 2017 Symposium on VLSI Circuits, Kyoto, Japan, 5–8 June 2017; pp. C24–C25. [CrossRef]
49. Sharma, H.; Park, J.; Suda, N.; Lai, L.; Chau, B.; Kim, J.K.; Chandra, V.; Esmaeilzadeh, H. Bit Fusion: Bit-Level Dynamically Composable Architecture for Accelerating Deep Neural Networks. *CoRR* **2017**, abs/1712.01507. Available online: https://arxiv.org/pdf/1712.01507.pdf (accessed on 4 January 2023).
50. Sharify, S.; Lascorz, A.D.; Judd, P.; Moshovos, A. Loom: Exploiting Weight and Activation Precisions to Accelerate Convolutional Neural Networks. *CoRR* **2017**, abs/1706.07853. Available online: https://arxiv.org/abs/1706.07853 (accessed on 4 January 2023).
51. Lee, J.; Kim, C.; Kang, S.; Shin, D.; Kim, S.; Yoo, H.J. UNPU: A 50.6TOPS/W unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision. In Proceedings of the 2018 IEEE International Solid - State Circuits Conference-(ISSCC), San Francisco, CA, USA, 11–15 February 2018; pp. 218–220. [CrossRef]
52. Ueyoshi, K.; Ando, K.; Hirose, K.; Takamaeda-Yamazaki, S.; Hamada, M.; Kuroda, T.; Motomura, M. QUEST: Multi-Purpose Log-Quantized DNN Inference Engine Stacked on 96-MB 3-D SRAM Using Inductive Coupling Technology in 40-nm CMOS. *IEEE J. -Solid-State Circuits* **2019**, *54*, 186–196. [CrossRef]
53. Moons, B.; Uytterhoeven, R.; Dehaene, W.; Verhelst, M. 14.5 Envision: A 0.26-to-10TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable Convolutional Neural Network processor in 28nm FDSOI. In Proceedings of the 2017 IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, CA, USA, 5–9 February 2017; pp. 246–247. [CrossRef]
54. Shin, D.; Lee, J.; Lee, J.; Yoo, H.J. 14.2 DNPU: An 8.1TOPS/W reconfigurable CNN-RNN processor for general-purpose deep neural networks. In Proceedings of the 2017 IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, CA, USA, 5–9 February 2017; pp. 240–241. [CrossRef]
55. Yuan, Z.; Yang, Y.; Yue, J.; Liu, R.; Feng, X.; Lin, Z.; Wu, X.; Li, X.; Yang, H.; Liu, Y. 14.2 A 65nm 24.7μJ/Frame 12.3mW Activation-Similarity-Aware Convolutional Neural Network Video Processor Using Hybrid Precision, Inter-Frame Data Reuse and Mixed-Bit-Width Difference-Frame Data Codec. In Proceedings of the 2020 IEEE International Solid- State Circuits Conference-(ISSCC), San Francisco, CA, USA, 16–20 February 2020; pp. 232–234. [CrossRef]
56. Wang, J.; Lou, Q.; Zhang, X.; Zhu, C.; Lin, Y.; Chen, D. Design Flow of Accelerating Hybrid Extremely Low Bit-width Neural Network in Embedded FPGA. *CoRR* **2018**, abs/1808.04311. Available online: https://arxiv.org/abs/1808.04311 (accessed on 4 January 2023).
57. Sze, V.; Chen, Y.H.; Yang, T.J.; Emer, J.S. Efficient Processing of Deep Neural Networks: A Tutorial and Survey. *Proc. IEEE* **2017**, *105*, 2295–2329. [CrossRef]
58. Zhang, D.; Yang, J.; Ye, D.; Hua, G. LQ-Nets: Learned Quantization for Highly Accurate and Compact Deep Neural Networks. In *Proceedings of the Computer Vision—ECCV 2018*; Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 373–390.
59. Wang, K.; Liu, Z.; Lin, Y.; Lin, J.; Han, S. HAQ: Hardware-Aware Automated Quantization With Mixed Precision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019.
60. Jin, Q.; Yang, L.; Liao, Z. AdaBits: Neural Network Quantization with Adaptive Bit-Widths. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020.
61. Soufleri, E.; Roy, K. Network Compression via Mixed Precision Quantization Using a Multi-Layer Perceptron for the Bit-Width Allocation. *IEEE Access* **2021**, *9*, 135059–135068. [CrossRef]
62. Camus, V.; Mei, L.; Enz, C.; Verhelst, M. Review and Benchmarking of Precision-Scalable Multiply-Accumulate Unit Architectures for Embedded Neural-Network Processing. *IEEE J. Emerg. Sel. Top. Circuits Syst.* **2019**, *9*, 697–711. [CrossRef]
63. Roelke, A.; Stan, M.R. Risc5: Implementing the RISC-V ISA in gem5. In Proceedings of the First Workshop on Computer Architecture Research with RISC-V (CARRV), Boston, MA, USA, 14 October 2017

64. RISC-V. Spike RISC-V ISA Simulator. 2021. Available online: https://github.com/riscv/riscv-isa-sim (accessed on 24 October 2022).
65. Imperas. riscvOVPsim-Free Imperas RISC-V Instruction Set Simulator. Available online: https://www.imperas.com/riscvovpsim-free-imperas-risc-v-instruction-set-simulator (accessed on 24 October 2022).
66. RISC-V. RARS – RISC-V Assembler and Runtime Simulator. Available online: https://github.com/TheThirdOne/rars (accessed on 24 October 2022).
67. QEMU. RISC-V System Emulator. 2021. Available online: https://qemu.readthedocs.io/en/latest/system/target-riscv.html (accessed on 24 October 2022).
68. GAPuino GAP8 Development Board. Available online: https://greenwaves-technologies.com/product/gapuino/ (accessed on 24 October 2022).
69. Page Github du Coeur cv32e40p (RI5CY). [en Ligne] Consulté le. Available online: https://github.com/openhwgroup/cv32e40p (accessed on 20 August 2021).
70. Ottavi, G.; Garofalo, A.; Tagliavini, G.; Conti, F.; Di Mauro, A.; Benini, L.; Rossi, D. Dustin: A 16-Cores Parallel Ultra-Low-Power Cluster with 2b-to-32b Fully Flexible Bit-Precision and Vector Lockstep Execution Mode. 2022. Available online: https://arxiv.org/abs/2201.08656 (accessed on 4 January 2023).