Received 2 November 2023; revised 18 January 2024 and 13 February 2024; accepted 9 April 2024. Date of publication 12 April 2024; date of current version 8 May 2024.

Digital Object Identifier 10.1109/OJCAS.2024.3388210

Exploiting Neural-Network Statistics for Low-Power DNN Inference

LENNART BAMBERG [1], ARDALAN NAJAFI [1], AND ALBERTO GARCIA-ORTIZ [2] (Senior Member, IEEE)

1 NXP Semiconductors, 22529 Hamburg, Germany
2 Integrated Digital Systems, ITEM Institute, University of Bremen, 28359 Bremen, Germany

This article was recommended by Associate Editor M. Cagnazzo.

CORRESPONDING AUTHOR: L. BAMBERG (e-mail: lennart.bamberg@nxp.com)

This work was supported in part by the IPCEI ME/CT; in part by the European Union Next Generation EU; in part by the German Federal Ministry for Economic Affairs and Climate Action; in part by the Bavarian Ministry of Economic Affairs, Regional Development and Energy; in part by the Free State of Saxony with the Help of Tax Revenue Based on the Budget Approved by the Saxon State Parliament; and in part by the Free and Hanseatic City of Hamburg.

ABSTRACT Specialized compute blocks have been developed for efficient DNN execution. However, due to the vast amount of data and parameter movements, the interconnects and on-chip memories form another bottleneck, impairing power and performance. This work addresses this bottleneck by contributing a low-power technique for edge-AI inference engines that combines overhead-free coding with a statistical analysis of the data and parameters of neural networks. Our approach reduces the power consumption of the logic, interconnect, and memory blocks used for data storage and movements by up to 80 % for state-of-the-art benchmarks while providing additional power savings for the compute blocks by up to 39 %. These power improvements are achieved with no loss of accuracy and negligible hardware cost.

INDEX TERMS Artificial intelligence, edge-AI inference, low-power coding, low-power digital design, neural networks.

I. INTRODUCTION

THE INCREASING demand for AI at the edge has led to the development of various low-cost, yet high-performance Deep Neural Network (DNN) inference engines. However, due to the compute requirements and the large memory footprint of modern DNNs (e.g., the famous *ResNet50* requires 25.6 M parameters and 4.1 GFLOPS/frame [1]), the power consumption of these inference engines remains a major challenge. This is particularly true for edge-AI deployment in small, battery-operated devices (e.g., hearing aids and IoT nodes). The main contributors to the power consumption of modern edge-AI inference engines are typically the memories and global interconnects used for data storage and exchange. Also, the required parallel multiply-accumulate (MAC) units noticeably impact power consumption.

Therefore, reducing the power consumption of these components is crucial for enabling energy-efficient edge-AI inference. A standard technique in the *TinyML* space (i.e., AI/ML on low-cost/battery-powered devices) is to quantize parameters and activations from 32-bit floats to lower-bit-width integers. However, quantization impacts network accuracy. Thus, to quantize below 8 bits is not possible without harming the accuracy for many applications. Full-integer 8-bit quantization reduces the power consumption of the memories and interconnects by a factor 4 ×. The savings for the MAC units are even higher as integer MAC blocks are cheaper than the floating-point equivalents [2]. Moreover, the complexity of a MAC unit scales quadratically with the bit width. Another technique to reduce power consumption at the application level is to use a lighter-weight DNN (e.g., [3]). Again, the drawback is a reduced maximum accuracy.

A further advantage of the "lossy" techniques mentioned above is that they substantially reduce the overall memory footprint. This enhances their suitability for systems with small memory capacities. Further techniques to reduce the memory footprint are weight pruning combined with variable-length entropy encoding. Thereby, the tool prunes weights with a small magnitude (effectively setting them to 0) during training. The large peak in zero-valued weights is then exploited by the entropy encoding, using a lower bit-width for zeros. Variable-length encoding, however, harms performance and area due to the decoding hardware and the varying decoder throughput for a given bus bandwidth.

Thus, parameters are typically only compressed for higher memory levels (e.g., off-chip DRAM or Flash), keeping them uncompressed in memory that is the closest to the data path of the AI engine.

The power issue can also be addressed through compute-in-memory (CIM) or neuromorphic architectures. However, these techniques are likely years away from commercial mass adoption. Near-memory computing and systolic/data-driven architectures are already deployed at a larger scale. Hence, local SRAM blocks near the compute engines—called tightly coupled memories (TCMs)—store the weights and activations during processing for cheap data access. Moreover, activations and weights are fetched from the TCMs once and used for multiple computations in parallel and/or sequential. To run larger DNNs or to run multiple networks in a time-multiplexed fashion, additional memory levels are still required to swap parameters in the TCM or spill activation from the TCM. Furthermore, even though better than more traditional architectures, accessing the relatively large local memories still strongly impairs the overall power consumption despite the high reuse from the systolic computing style.

In this paper, we propose a novel approach to further reduce the power consumption of edge-AI inference engines by combining lossless and overhead-free low-power coding techniques with a comprehensive analysis of the statistical properties of the activations and parameters in neural networks. In particular, our approach exploits the non-maximized entropy of the activation and parameter streams caused by the non-uniform distribution of the data incl. activation and weight sparsity. Our approach is meant to extend existing approaches such as light-weight DNNs, full-integer quantization, pruning, and near-memory-compute, to further optimize the power consumption. Our coding technique is “lossless” which implies that it does not affect the DNN accuracy. Thus, the technique does not impact the outputs of the neural networks at all. This means that no change to the DNN architectures, the training, or the quantization step is required. On top, our proposed technique has no noticeable hardware cost and does not increase the memory footprint (i.e., overhead-free coding).

These properties allow designers and architects to apply the technique to their AI engines “blindly” (i.e., without an analysis of trade-offs). To the best of the authors’ knowledge, it is the first low-power technique for DNN engines with the property of being universally applicable without drawbacks on any abstraction level. Despite the easy application of the proposed technique at negligible silicon cost, it reduces the power consumption of interconnects and on-chip memories by over 80 % for real benchmarks in combination with weight-pruning [4]—initially proposed as a pure compression technique. On top, we can achieve additional power savings by over 35 % for the processing elements.

The rest of this paper is structured as follows. Related work is outlined in Section II. The background is presented in Section III. Subsequently, the bit-level statistics in modern DNNs are analyzed. In Section V, our proposed technique is presented. Afterward, the gains of the technique are qualitatively and quantitatively assessed in Section VI. Finally, a conclusion is drawn in Section VII.

II. RELATED WORK

Low-power coding, reducing the power consumption by changing the statistical properties of the bits, has been widely researched for various components and applications. Due to the large parasitic capacitance, most low-power encoding focuses on power savings in planar metal wires and 3D through-silicon-via interconnect [5], [6], [7], [8], [9], [10], [11]. Also, low-power coding for memories and memory buses has been researched [12], [13].

Traditional low-power codes are tailored for a power reduction for arbitrary or random data. The most popular coding techniques here are bus-invert or inversion codes (e.g., [6], [7]). These techniques increase the bit-width of the codewords over the plain data words by one inversion line. Also, application-specific low-power codes, for example for signal processing [9], [13] or address buses [8], [10], [11], were designed. Compared to inversion codes, they are overhead-free (i.e., codeword bit-width equal to dataword bit-width) and/or yield much higher power savings while exhibiting a lower implementation complexity. These gains are achieved by exploiting the statistical properties of the involved data. The drawback is that they are not universally applicable due to their application-specific nature.

Recent research already investigated low-power coding to decrease the power consumption in DNNs accelerators [14]. In particular, it uses bus-invert for coding the mantissa of the *bfloat16* numbers used in a systolic array. The authors demonstrated an overall power reduction of 9.4 % for *ResNet50*. Although the motivation is similar to ours, this work provides substantial novelty.

First, we design the first application-specific low-power codes for DNN inference. Thereby, we substantially increase the power savings at reduced implementation cost. Second, [14] focuses only on the data-path of the core accelerator, disregarding potential savings in the memories and interconnects—which however typically exhibit the highest power needs. Moreover, [14] focuses on floating point numbers, while AI inference at the low-power edge is dominantly uses integer quantization [2]. Thus, this work presents the first, holistic and application-specific low-power coding approach for integer-quantized DNN inference.

III. BACKGROUND

A. FULL-INTEGER DYNAMIC-RANGE QUANTIZATION

In the following, we describe the standard way of quantizing a neural net to full-integer operations described in [2].

The core operation of the DNN layers is a MAC series of a set of weights $\mathbf{w}$ with a set of activations, $\mathbf{x}$, followed by the addition of a bias, b, and a non-linear activation function, $\sigma()$, on the result:

$$y_i = \sigma\left(\sum_j w_{i,j}x_{i,j} + b\right) \tag{1}$$

Different DNN layer types (e.g., convolutional vs. fully connected) differ in how the weights and the activations are gathered and reused for a set of activations to be generated.

Full-integer dynamic-range quantization quantizes different sets of activations or weights to *int8* (i.e., $q_i^x \in [-127, 128]$) through a *float* scale factor, S, and an *int8* zeropoint (ZP). Thereby, the quantization has a dynamic range based on the value range of each set, determined during training time. The de-quantization from a quantized integer, q, back to the *float* is described as:

$$x_i = S^x\left(q_i^x - ZP^x\right) \tag{2}$$

Hence, the multiply-accumulate can be expressed as:

$$S^xS^w\left(\sum_j q_{i,j}^w q_{i,j}^x - q_{i,j}^w ZP^x - q_{i,j}^x ZP^w + ZP^x ZP^w + q^b\right) \tag{3}$$

Here, the term $-q_{i,j}^w ZP^x$ is constant post-training and can therefore be calculated at compile time and merged into the 32-bit bias for the accumulation. This cannot be done for the term $q_{i,j}^x ZP^w$ due to varying quantized data q^x at inference time. To avoid the extra compute, weights are thus mean-free quantized (i.e., $ZP^w = 0$; $q_i^w \in [-127, 127]$). It is empirically found that this does not harm accuracy as weight kernels tend to be mean-free due to the applied weight regularisation during training.

Highly non-linear activation functions are hardware-costly and reduce the accuracy of quantized inference. Thus, full-integer quantization typically focuses on activation functions that are a mere clipping, as the saturation of quantized values to the (min, max) values $(-128, 127)$ makes this a no operation (*NOP*). The most common activation function for quantized inference is $ReLU = \max(x, 0)$, clipping negative values to zero [2]. With activation functions that are a mere clipping and symmetrically quantized weights, the core operation becomes:

$$q_i^y = M\left(\sum_j q_{i,j}^w q_{i,j}^x + q'^b\right), \tag{4}$$

where q'^b is the effective bias equal to $quant32(b/S_xS_y + ZP^y/M - q_{i,j}^w ZP^x)$. M is the rescale factor to change to the scale of the quantized output S_y (i.e., $M = S_wS_x/S_y$). The multiplication with this "float" rescale factor $M \leq 1$ is realized through integer multiplication and shifting. For rescale factors and biases, typically 16 or 32 bits are used. Since typically hundreds to thousands of 8-bit activations are weighted and accumulated per output activation, the cost of having wider rescale factors and biases is negligible.

B. LOW-POWER CODING

Due to the large parasitic capacitances of modern interconnects, data encoding is a promising low-power technique in advanced technology nodes [5]. Low-power encoding adopts the bit-level properties of the data to reduce the power consumption of the physical: interconnects (e.g., metal wires or vias) for data transmission; memory for data storage; or logic for data processing. In this work, we only consider lossless and overhead-free low-power encoding such that it can be applied without further analysis of the impact on the memory footprint, bandwidth requirements, or network accuracy.

The interconnects in today's integrated circuits (ICs) are made up of metal wires and vias. Metal wires and vias entail a physical capacitance that dominates their power consumption. Whenever the logical value transmitted on a metal interconnect toggles, the capacitance is charged or discharged resulting in an energy loss of $V_{\mathrm{dd}}^2C/2$. Thus—as with logic—reducing the bit-switching activities of the lines yields a proportional reduction in the dynamic interconnect power consumption. For 3D interconnects, also increasing the 1-bit probability on the lines reduces the power consumption due to the MOS effect [15].

In modern DNN engines, the dominating factor for power consumption is reading on-chip memory. On-chip memories in today's ICs are predominantly SRAMs. The SRAM read power consumption is dominated by the bitline discharging if the stored value differs from the pre-charge voltage [13]. For single ported SRAMs, low-power coding cannot effectively reduce power consumption due to the complementary read behavior. Always, either the regular or complemented bitline is charged or discharged from $V_{\text{pre-charge}}$ to zero by the read resulting in a power consumption that is independent of the bits to be read. This changes for single-ended reads in dual-ported SRAMs [13]. Thus, if dual-ported SRAMs are used (e.g., to load parameters in parallel to compute) low-power encoding can effectively reduce power consumption via an increased 1-bit probability of the encoded data stored in the memory. This is illustrated in Fig. 1.

Lastly, the dynamic power of the logic and sequential elements in the processing elements is again dominated by bit switching of the bits due to internal short-currents when gate inputs switch and the gate input capacitances. Some flip-flops also show to have a power consumption that depends on the logical bit probabilities.

Leakage is another important power contributor in modern ICs, especially in modes of low activity. The leakage power of SRAM, logic, and flip-flops can be optimized again by tuning bit probabilities [5]. Leakage for interconnects is negligible.

In summary, low-power coding should reduce the switching and the bit probabilities. Which metric is more important depends on the components whose power consumption must be reduced. Generally, SRAMs, MACs, and global interconnects dominate the power consumption in modern AI engines. For the former, the bit probabilities must be

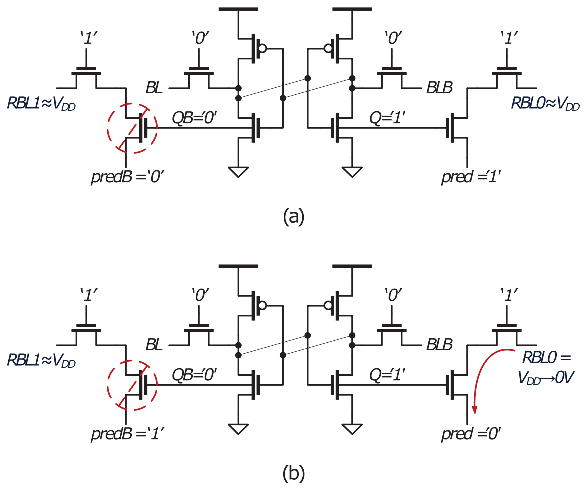


FIGURE 1. Reading a 0-bit *(a)* and 1-bit *(b)* from a dual-ported 6T SRAM cell. Due to the discharging of the bitline, *(a)* results in a higher power consumption. Figure adopted from [13].

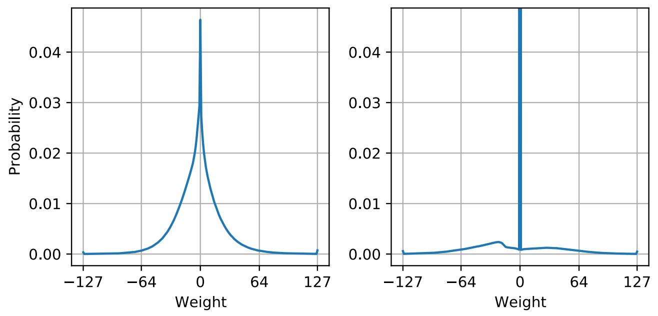


FIGURE 2. Probability density function of all the weights of an 8-bit-quantized *ResNet50* before (left) and after (right) pruning. After pruning, the zero weight contains almost all the probability (80 %).

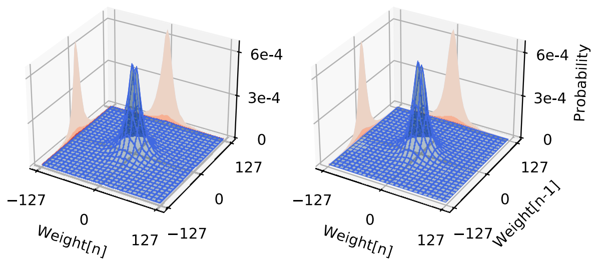


FIGURE 3. Two-dimensional PDF of the weights of a quantized *ResNet50*. Results for two different output layers. The example illustrates the un-correlation between pattern pairs and the leptokurtic characteristics.

optimized, and for the latter two, the switching. The worst case is a completely random switching and 1-bit probability of 0.5 (50 %). Everything above 0.5 can be transformed to $1 - x$ through inverting buffers if the bit-level statistics are understood. This is why our encoding focuses on reducing the 1-bit probabilities even though SRAMs actually needs high 1-bit probabilities for minimal power consumption.

IV. ANALYSIS OF THE BIT-LEVEL STATISTICS

In this section, we analyze the bit-level statistics of the streams of *int8* (two's complement number representation) quantized weights and activations found in modern Edge-AI inference engines. We moreover, compare the statistics to the ones for data found in more traditional DSP applications. Data streams sampled from the physical environment—such as audio, radio signals, etc.— tend to be Gaussian distributed. Consequently, the bit-level statistics follow a systematic pattern with two characteristic parts: a group of LSBs which are strongly un-correlated and have a switching activity of 0.5, and a group of strongly correlated MSBs with lower switching activity for temporally correlated signals [16]. These bit-level characteristics can be used to decrease the power consumption during transmission using low-cost low-power codes [5]. It is worth noting that the bit-level probability is equal to 50 % for all bits, only the switching activity changes; and second, that the decrease in switching activity happens only when the samples are temporally correlated—the typical case in DSP signals.

The characteristics of the data streams in DNN engines are however different. The left-hand side of Fig. 2 shows the probability density function (PDF) for a stream containing all 25 M, per-channel, 8-bit quantized weights of *ResNet50*. Analyzing various DNNs for this work showed that light-weight DNNs such as *MobileNet*, exhibit a typical Gaussian-like distribution with spikes at the edges -127 and 127 due to truncation of the Gaussian tails. However, more heavy DNNs such as *ResNets* and Transformers depict PDFs with a larger kurtosis and lower variance, σ_{var}. Also, per-tensor instead of per-channel quantization yields a smaller variance, as a single outlier in one channel reduces the integer values in all other channels. Bit pruning heavily increases the number of zero values such that it dominates the distribution.

In any case, the switching activity and bit activity are almost exactly 0.5 because the individual weights show to be uncorrelated as shown in Fig. 3. This yields a particularly high power consumption. The contribution of this paper is to understand the bit-level characteristics of the weights and activations by analytical models and to use these models to derive efficient low-power coding strategies.

For the activation statistics, we constrain our analysis to the *ReLU* activation function (and its derivatives like *ReLU6*), applying a merge clipping of the values. It is not only the most common activation function for quantized *TinyML* applications; it also has great characteristics for low-power processing, as we will show in the remainder of this paper. A *ReLU* clips all negative values to zero. Thus, a *ReLU* only outputs positive values while 8-bit-quantized activations are in $[-128, 127]$, resulting in the ZP for *ReLU* activations shows to be always -128, transforming the range linearly to a purely positive one, $[0, 255]$. Since the weighted, accumulated values are normally distributed, after activation, about 50 % of activations are equal to the ZP -128 or 0x80.

V. PROPOSED TECHNIQUE

As outlined above, the statistics of the quantized weights and activations show a large potential to decrease power consumption by an appropriate low-power coding. In this section, we discuss our proposed approach to achieve this reduction. First, we give an overview of the type of systems we are targeting; then, we describe the proposed techniques for encoding the weights and activations; and finally, we analyze the impacts of the encoding variants on the hardware implementation of the MAC.

A. SYSTEM OVERVIEW

Certainly, the exact low-level micro-architectural details of different AI accelerators vary notably; however, the structure of the memory hierarchy and the interconnect architecture is, in the fundamental aspects, very similar.

From bottom to top: AI architectures are composed of a set of processing elements (PEs), typically including a private TCM, separated for parameters and activations. Multiple PEs are clustered in groups that commonly contain one or more additional levels of shared, unified memory dedicated to the AI engine. The individual PEs and the dedicated memories are connected over a massively parallel and high bandwidth interconnect architecture—often a network-on-chip (NoC). The AI engine is connected to the rest of the system over a system-level interconnect. This interconnect architecture can be the same as for the local interconnect between the PEs (typically for NoCs), or a different one such as an AHB or AXI bus. For large models, the weights are saved outside the chip in DRAM, normally in compressed form, and are read on-demand into the on-chip global memories with on-the-fly decompression.

The two key streams of information in an AI architecture are the weights and the activations. The interpretation of these two streams requires three levels of abstraction; from the low level of abstraction to the higher one, they are:

word The physical stream of bits moving through the interconnects, or the values saved into on-chip memories. Can be encoded using approaches such as Bus-Invert, K0, etc. Thus, for processing, the bits have to be converted into (interpreted as) an integer, which may imply some hardware modules.

int. Once decoded, the words need to be interpreted as integer numbers. Here again, there are different alternatives, as for example, the use of sign-magnitude, unsigned numbers, or the more common two's complement. Note that this interpretation of words into integers determines the actual ALU that is required in the PEs (e.g., a standard signed-signed MAC or a sign-magnitude approach).

real Finally, in an AI architecture, the integer numbers are interpreted as a corresponding real number using a linear mapping characterized by a zero point, *ZP*, and a scale, *S*. The rescale process after the integer MAC-series is responsible for the right interpretation.

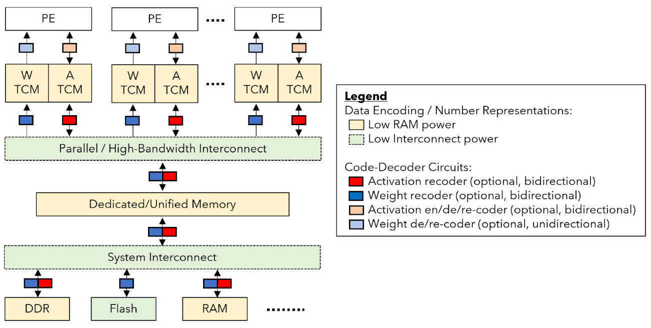

PE
PE
PE
W TCM
A TCM
W TCM
A TCM
W TCM
A TCM
Parallel / High-Bandwidth Interconnect
Dedicated/Unified Memory
System Interconnect
DDR
Flash
RAM
Legend
Data Encoding / Number Representations:
Low RAM power
Low Interconnect power
Code-Decoder Circuits:
Activation recoder (optional, bidirectional)
Weight recoder (optional, bidirectional)
Activation en/de/re-coder (optional, bidirectional)
Weight de/re-coder (optional, unidirectional)

FIGURE 4. Illustrative system overview with low-power encoding/decoding for weights and activations. The bus width depends on the system (e.g., 4 bytes or 8 bytes) and encoding or decoding is done per byte lane. The encoding is overhead-free (i.e., code word width equal to data word width, 1 byte) such that the bus width and memory requirements are not affected by the coding. The memory hierarchy can have any number of levels (we show two as an example).

The fundamental idea of this work is to exploit the statistical characteristics of the activation and weights, as well as the degrees of freedom provided by the interpretation as integer and later as real numbers of the words to decrease the power consumption in the interconnect architecture and the memory hierarchy of AI inference engines.

Fig. 4 shows an illustrative system overview with the proposed low-power-coding technique in place. Weights are low-power encoded at compile time and placed in the on-chip memory or DDR with optional additional compression. Activations are dynamic and not known at compile time. They are generated and read by the PEs. Ideally, activations are only stored on the lowest level of the memory hierarchy, made up of the TCMs of the PEs, but are spilled to higher hierarchies if a TCM capacity is exceeded. Weights move from the higher memory levels into the TCM when the respective layer of the neural network is processed by the PEs. From the TCM, the PEs typically access weight multiple times per inference, depending on the weight reuse in the layer.

The goal of our technique is to improve the power consumption across all levels of the interconnect and memory hierarchy. Thus, only at the input of the PEs the weights and activations are potentially decoded into the standard *int8* format to perform the MACs computations. At the output of the PEs, the activations are low-power encoded again.

However, the weights encoded in the parameter TCM, can be also used directly by the PEs (without decoding) by adapting the architecture of the PEs to a different integer representation, as shown in the experimental results section of this paper. Similar to the weights, at the input of the PEs, the activations can be either decoded to *int8* or processed in the low-power representation.

At the interface between a memory and an interconnect block, the number encoding/representation is potentially changed to one that optimizes the power of the respective physical medium for transmission or storage. Thus, going from a memory to an interconnect the data encoding is changed to one that optimizes the interconnect power consumption and vice versa. It is also possible that the same low-power encoding is used for interconnects and memories. Either because a single encoding sufficiently optimizes both power metrics or because the prime optimization goal is only the power consumption of one structure. In this scenario, there is at most encoder/decoder circuit between each TCM and the corresponding PE.

For our approach, it is essential to use low-power codes with negligible hardware implementation costs to ensure that the technique is universally applicable, as it comes in variants where multiple en- or decoder circuits are required in the system. The only exception is weight encoding, which always happens offline at compile time. It thus can exhibit a larger complexity than weight decoding or activation en- and decoding.

In summary, we need low-power codes that exploit statistics of activations and weights for maximized power savings through minimizing the 1-bit probabilities (for dual-ported SRAMs) or the switching probabilities (for logic and interconnects). The low-power codes need to be designed to change from encodings with low switching activities to encodings with low 1-bit probabilities at low hardware costs. On top, the MAC hardware must either be capable of directly processing the encoded weights or activations, or a low-cost decoding to the standard *int8* format is required.

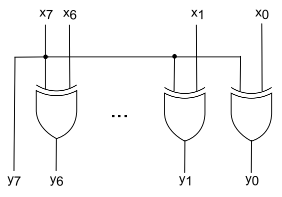
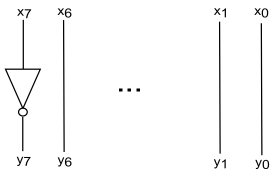
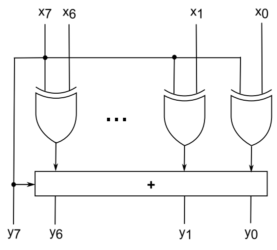
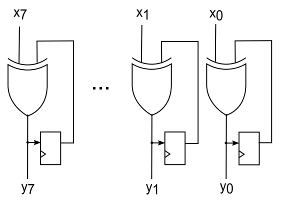
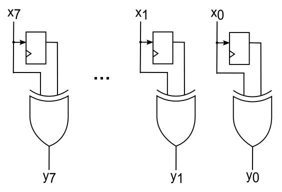
B. LOW-POWER CODING FOR WEIGHTS

In this work, we only consider the most area-efficient coding approaches, inspired by our existing work on more traditional low-power coding [5]. Our contribution is a coding framework that, at ultra-low-cost, can change the signal representation to one with optimized switching characteristics or one with optimized bit probabilities, and back. Thereby, IC designers can pick for each component the signal representation that results in the best power consumption.

Two separate encoding schemes are used: first, a probability coding that decreases the 1-bit probabilities of the patterns, and second, an *XOR-decorrelator* that maps 1-bits at the input x to transitions at the output y; i.e., $y = y_{\text{prev}} \oplus x$. To go back from the minimized-switching representation to a minimized-1-bit representation, an *XOR-correlator* is used, mathematically expressed as $y = x_{\text{prev}} \oplus x$. If 0 bits are minimized by the probability encoding instead of 1 bits, in decorrelator and correlator, XOR operations must be swapped with XNORs to map zeros to transitions. As can be seen in the last two rows of Table 1, in both variants, the encoder and decoder only need $B = 8$ flip-flops and X(N)OR gates and have a logic depth of 1.

We study two alternatives for the probability coding for DNN parameters: Sign-Magnitude (SM) representation and the *XOR-MSB-coding*. The latter approach consists of XORing the $B - 1$ LSBs with the MSB value, leaving the MSB untouched, for a minimized 1-bit probability. An advantage of this *XOR-MSB-coding* is that the encoder and decoder circuits are identical. Also, it again has ultra-low cost as the coding circuit is $B - 1$ parallel XOR gates (i.e., logic depth of 1) as shown in the first row of Table 1. To achieve a minimized 0-bit probability, again simply XNOR gates are used instead of XOR gates.

TABLE 1. Used coding techniques. Note that encoders and decoders of the proposed techniques are identical, except for the *XOR-decorrelator* coding.

Coding	Equation & Hardware
XOR-MSB	$\mathbf{y} = \mathbf{x} \oplus x_{\text{MSB}}$ 
XOR-ZP (for ZP=-128)	$\mathbf{y} = \mathbf{x} \oplus \text{0x80}$ 
Sign-Magnitude	$\mathbf{y} = (\mathbf{x} \oplus x_{\text{MSB}}) + x_{\text{MSB}}$ 
XOR-decorrelator	$\mathbf{y} = \mathbf{y}_{\text{prev}} \oplus \mathbf{x}$ 
XOR-correlator (XOR-decorrelator decoder)	$\mathbf{y} = \mathbf{x}_{\text{prev}} \oplus \mathbf{x}$ 

The rationale behind the XOR-MSB technique is to exploit that the upper-most bits in each pattern are typically equal (i.e., high spatial correlation), but with equal probability 0 and 1 (i.e., no temporal correlation). By XORing or XNORing each bit with the MSB, the upper-most bits but the MSB become mostly 0 or 1, respectively. This by itself does not only optimize the bit probabilities. It already reduces the switching. Thus, the technique optimizes both key metrics at once. Hence, without additional *XOR-decorrelator coding* it can improve the power consumption of all kinds of components without additional re-coding steps.

Thus, for the typical case of $B = 8$, our proposed coding technique approach only needs 7 XOR-gates when the *XOR-decorrelator* is not used. With the *decorrelator-coding*, an additional 8 XOR gates and flip-flops are needed per re-coding. Note that the data is typically already encoded once at compile time with an *XNOR-MSB encoding* to optimize the power consumption of the memories in which the parameters will be stored before they are first used.

The second variant, *Sign-Magnitude (SM) encoding*, works for DNNs weights without causing any coding overhead, despite the redundant representation of 0 (0x80 and 0x00), as weights are only symmetrically quantized to integers in $[-127,\ 127]$. Thus, -128 does not occur.

Sign-magnitude coding has higher encoder-decoder costs (see the third row of Table 1), but is based on the same coding idea as the XOR-MSB encoding. Hence, it yields similar optimization in terms of bit probabilities and switching. Despite the higher en/decoding complexity, *SM encoding* for weights is promising, as, for the actual MAC execution, weights either need to be decoded or the hardware adjusted to the new weight representation. If the weights are in sign-magnitude format, the hardware becomes even simpler by the modification compared to the initial $8b \times 8b$ signed MAC. The multiplication becomes 7-bit unsigned (7 LSBs of SM weight) by 8-bit unsigned activation, followed by a selective addition or subtraction of the result to the accumulator based on the sign-bit of the weight. Thus, no on-chip SM encoding or decoding of the weights is needed at all while the MAC hardware is minimized. For this scenario, SM weights are proposed.

C. LOW-POWER CODING FOR ACTIVATIONS

For activations, the proposed coding is even simpler than for weigths. The absence of negative numbers due to the *ReLU* activation already heavily reduces the MSB switching. Moreover, all bits but the MSB have low bit probabilities. This is because 50 % or even more patterns are equal to the ZP (0x80), which only has one 1-bit at the MSB. Thus, X(N)ORing with the constant ZP effectively optimizes the bit probabilities. For a ZP of 0x80, this just requires negating the MSB to reduce the number of 1 bits and negating the 7 LSBs to reduce 0 bits. Thus, the hardware cost for this coding we refer to as *XOR-ZP* is approximately zero (see the second row of the Table 1). Effectively, the coding moves the number representation from *int8* to *uint8* with a quantization zero-point of 0. This enables directly processing the *uint8* numbers in a *uint* MAC block without decoding by adjusting the accumulator bias calculated at compile time.

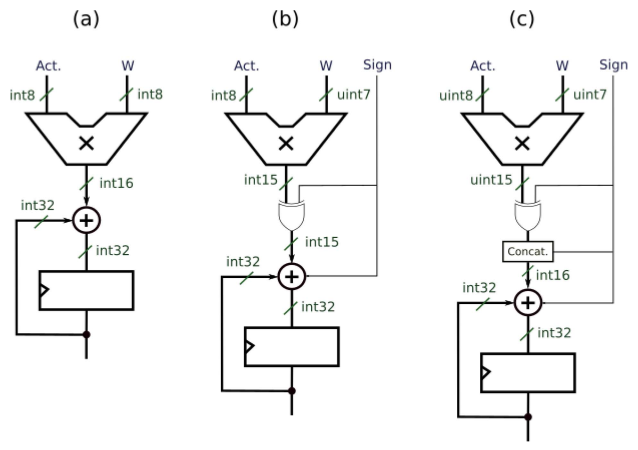


FIGURE 5. The MAC architectures with inputs: (a) *int8*−*int8*, (b) *int8*−*uint7*, and (c) *uint8*−*uint7*. For the cases where weights are coded with sign-magnitude (*uint7*), the sign bit is used as an additional input to the adder.

As for weights, *XOR-decorrelator coding* is used to transform optimized bit probabilities into optimized bit switching. Again, for $B = 8$, the proposed decoding approach just needs a couple of gates and 8 flip-flops if the XOR-decorrelator is present, while the logic depth is one.

D. IMPLEMENTATION OF MAC WITH CODING

To demonstrate that the above-mentioned coding schemes result in overall energy savings, we study their effects on the MAC units as well. Based on the coding schemes discussed earlier, we consider three alternatives for the MAC implementation, as shown in Fig. 5. Note that in the case of *XOR-MSB coding*, an extra addition is required to obtain the correct result. Since the irregular architecture results in an excessive overhead, the inputs to the MAC unit should be decoded first for this specific coding. As a result, the corresponding MAC unit would be a normal two's complement (Fig. 5a), and we do not consider an extra design alternative for *XOR-MSB coding*.

In Fig. 5a, the conventional two's complement MAC implementation is shown. This design employs 8-bit signed multiplication. Fig. 5b illustrates the case where the activations are not coded (*int8*), and the weights are in SM (*uint7*). In this case, the multiplier has one less partial product row, resulting in a smaller architecture. The accumulator input width is reduced by 1 in exchange for the addition of a carry-in and an input XORing (to decode into two's complement). The last alternative shown in Fig. 5c is the implementation of the MAC unit where the activations are coded with XOR-ZP (*uint8*), while the weights are in SM (*uint7*). In this alternative, an unsigned multiplier is used.

To study if the coding ideas mentioned earlier affect the energy consumption of PEs, we consider an inner-product unit (IPU) performing eight MAC operations in parallel. For the baseline case in which no coding is applied, the

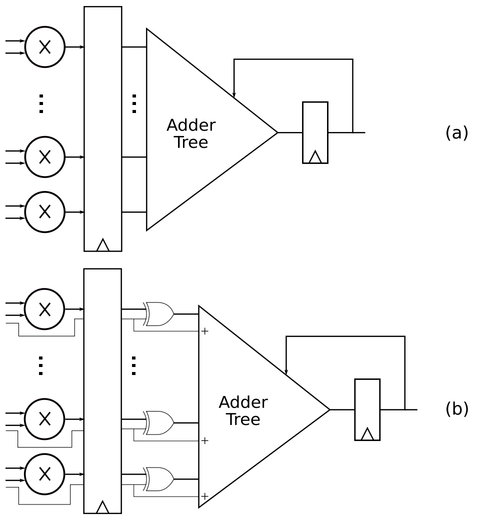


FIGURE 6. The adder tree-based inner-product units.

Algorithm 1: Description of the Proposed Low-Power Technique in the Different Variants

```
OFFLINE (AT COMPILE TIME):
Weights encoded with either SM or XOR-MSB

ONLINE (USING LOW-COST CIRCUITRY):
Activations XOR-ZP encoded at PE output
if decorrelator used then
    XOR-decorrelator at memory read and PE output
    XOR-correlator at memory write and PE input
end
if int8–int8 MAC used then
    Weights XOR-MSB or SM decoded at PE input
end
if not uint8–uint7 MAC used then
    Activations XOR-ZP decoded at PE input
end
```

inner-product unit works with two's complement numbers (see Fig. 6(a)). When the weights are in SM, the sign bits are used to negate the outputs of the multiplier employing parallel XOR gates (cf. Fig. 6(b)). The additions of the sign bits are integrated into the adder tree so that they do not introduce an overhead. As a result, for the applied codings, the overhead is the XOR gates (8×15 gates for our IPU) and the saving is one partial-product addition for each of the 8 multipliers.

The detailed algorithm of the proposed low-power technique for different variants when using a specific MAC unit is shown in Algorithm 1. The algorithm also shows which coding is done offline at compile time and to which coding hardware is dedicated.

VI. EXPERIMENTAL RESULTS

The experimental results section is divided into three parts: a qualitative analysis, a quantitative assessment, and the effects

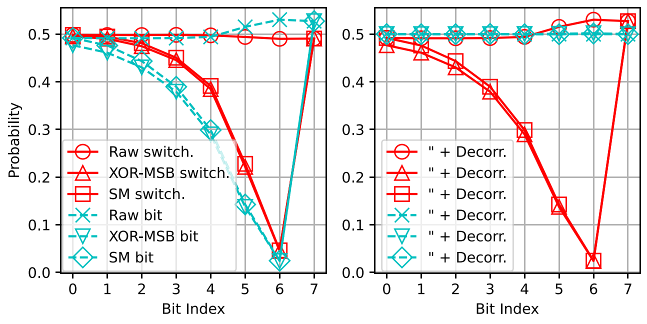


FIGURE 7. Impact of proposed low-power coding on bit-level statistics for 8-bit quantized *ResNet50* weights (non-pruned): (left) without XOR-decorrelator; (right) with XOR-decorrelator.

of the codings on the energy consumption of the MAC unit. For a fair, unbiased experimental setup, throughout this work, we analyze only DNNs that were professionally trained by others. In detail, we use CNN trained for *ImageNet1k* classification provided by the application module of *Keras* in *TF2.11*. This ensures maximized reproducibility of the experimental results without any neural-network IP issues. With the paper, we further publish a *Python* package to test the power gains of our technique for any quantized *TFlite* model: https://github.com/ardalan-ids/interconnect-statistics-nn.git. This further enhances reproducibility.

We investigate per-channel quantized neural networks. Per-tensor quantization (the less advanced technique) results in smaller standard deviations for the weight distribution. This enhances the gains of our technique (cf. Section VI-B). Thereby, we avoid reporting overly optimistic gains for our technique that are only seen in some applications.

A. QUALITATIVE CODING ANALYSIS

Our qualitative analysis employs an analysis of the probabilities of a logical 1 on a bit line of a code word and the switching activity of the bits, carried out with *Python* and *Tensorflow*. We present results for the vanilla *ResNet50* architecture as a representative example. The same analysis for other DNNs shows no conceptual differences. Quantitative results for several DNNs are provided in the next subsection. In this qualitative analysis, we investigate the bit-level switching and bit probabilities of the activation and weight data. The order in which the data is processed, read, or transmitted is irrelevant to this analysis, as the values tend to be uncorrelated. Hence, without loss of generality, we can investigate a single random shuffle of the activations and parameters in each tensor.

The results for the 25 M weights of *ResNet50* are presented in Fig. 7. The standard raw values result in high power consumption quantities, as both the bit switching as well as the bit probabilities seem to be random (i.e., 50 %/bit)—which is the worst case. Due to the random bit probabilities an XOR-decorrelator alone cannot reduce the power consumption, as shown in the raw-switching line of the right panel.

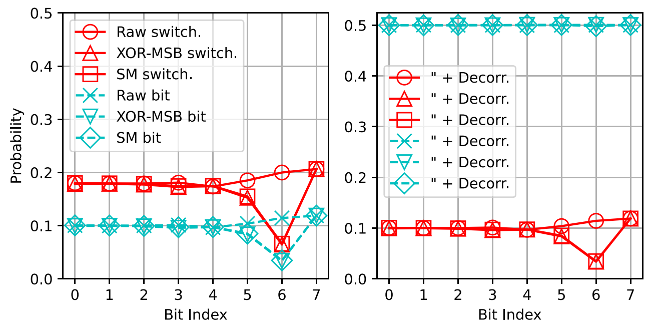


FIGURE 8. Impact of proposed low-power coding on bit-level statistics for 8-bit quantized *ResNet50* weights that are 80 % pruned: (left) without XOR-decorrelator; (right) with XOR-decorrelator.

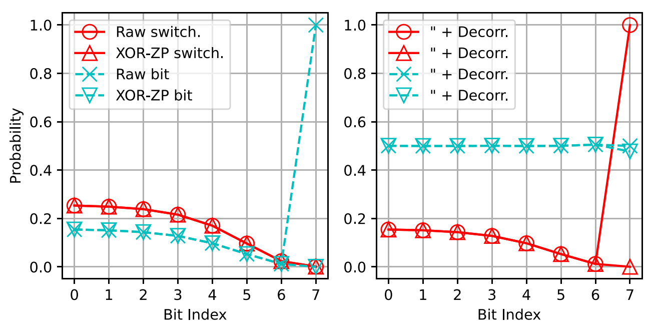


FIGURE 9. Impact of proposed low-power coding on bit-level statistics for all activations within an 8-bit quantized *ResNet50* weights (non-pruned, one input).

When we apply the XOR-MSB coding or the SM coding to the weights we observe a significant decrease in the bit probabilities, especially at bit indices 6 to 4. This reduction in the bit probability results in a reduction of switching activity, as $t_i = 2pr_i(1 - pr_i)$ for uncorrelated data. If an XOR-decorrelator is additionally used (right panel), the switching activity is reduced even further, since 1 bits are mapped to transitions. The drawback is random bit probabilities again. Note that there is no improvement in the MSB, since this bit is not modified by the probability coding schemes.

After the analysis for standard DNNs, we focus on the effect of weight pruning. The results for the *ResNet50* architecture after pruning 80 % of the weights using the standard *TFLite* pruning methodology are shown in Fig. 8. The first noticeable effect is that now all the bit probabilities of the unencoded raw data are close to 0.1 instead of 0.5. The reason is the increase of the zero values due to pruning (ZP = 0x00 for all weights). This effect reduces the switching as well as bit probabilities—and thereby the power consumption–dramatically. After applying our coding approach without an XOR-decorrelator, the switching activity of bits is further reduced by 2 ×, with a low gain in the LSBs. When the full coding approach is used with the XOR-decorrelator (cf. right panel), the switching activity of all bits goes below 0.1. Hence, the total power consumption can be reduced by over 5 × for both: components whose power consumption is proportional to the bit switching; as well as components whose power consumption is proportional to the bit probabilities. It is worth noting that even without further probability coding (e.g., XOR-MSB or SM), XOR-decorrelator coding is very beneficial for pruned weights.

In the last step of the qualitative analysis, we evaluate our coding for activations. Again, our analysis for various inferences shows that the gains seem to be relatively independent of the input data, why we represent only data for one randomly selected input. Also, weight pruning shows to have no noticeable impact on the activation distribution why it is not analyzed separately here. The results for all 9.4 M activations of a *ResNet50* inference are shown in Fig. 9.

The clipping to zero by the *ReLU* activation has a positive effect on the switching and bit probabilities. Compared to zero-mean Gaussian distributed data the switching is reduced by 2 × in the LSBs to about 10 × in the MSB. The bit probabilities in the LSBs are even reduced by 3 × for the LSB to over 41 × for the second MSB. However, the extremely high bit probability of the MSB reduces the overall optimization of the bit probabilities to only 2 ×. This is fixed by our XOR-ZP encoding which here just negates the MSB resulting in an overall bit-probability and switching optimization compared to random/Gaussian data by over 5 × and 10 ×. Our XOR-correlator enables us to get also 10 × switching savings, at no bit probability savings. This shows that *ReLU* activations alone already bring the power consumption noticeably down, but our coding can even further push the low-power limits.

B. QUANTITATIVE CODING ANALYSIS

We quantify the bit-switching and bit-probability optimization utilizing Python and the *Tensorflow* models of popular edge DNN. First, we analyze two of the most widely used CNNs in academia and industry *ResNet50* and *MobileNet*. As a cross-validation, we look at MobileNet V1 as well as V2 which differ in some concepts but are both light-weight DNNs. Furthermore, we look at a more recent DNN, *EfficientNet*, to show that the technique is future-proof. Activations are only investigated for *MobileNetV1* and *ResNet50*, as for others we found no professionally trained but yet publicly available variants with *ReLU* activations. Pruning is as well only reported for these two networks as the other two show the same behavior/gains.

Table 2 shows the total switching and bit probability reductions obtained with the proposed technique compared to random/Gaussian distributed data. The results for SM encoding are similar to the ones for XOR-MSB encoding, with a tendency to be slightly worse (always <2 percentage points). Thus, SM encoding is not listed separately.

For standard weights whose individual bits appear random, our technique achieves a power reduction of up to 33 %. This is a remarkable reduction considering that the technique here requires no changes to the training, the DNN itself, and comes at no accuracy or noticeable hardware cost. For

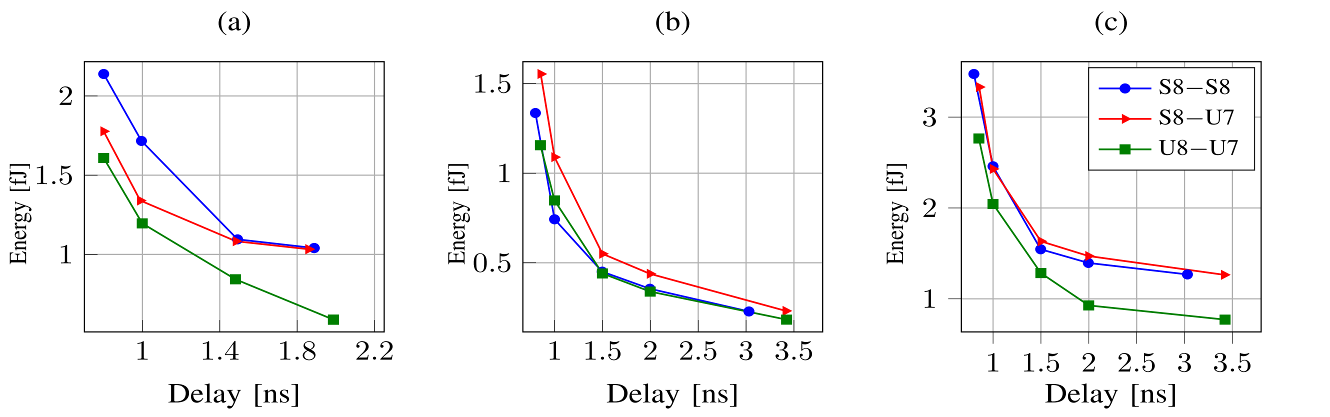


FIGURE 10. Energy versus delay in a commercial 40-nm technology of: (a) the multipliers, (b) the adder tree, and (c) the total energy of the inner-product unit including multipliers and the adder tree, for different coding schemes.

TABLE 2. Switching and bit probability optimization by the proposed versus the standard technique for various DNN data.

Type	Encoding	Switching	Bit Prob.
Weights ResNet50	Standard	-0.08 %	+0.08 %
	XOR-MSB	-23.8 %	-31.9 %
	XOR-MSB & Decorr.	-31.9 %	+0.03 %
Weights MobileNet	Standard	-0.16 %	+1.09 %
	XOR-MSB	-14.5 %	-22.6 %
	XOR-MSB & Decorr.	-22.6 %	+0.01 %
Weights MobileNet V2	Standard	-0.19 %	+0.97 %
	XOR-MSB	-10.8 %	-18.5 %
	XOR-MSB & Decorr.	-18.5 %	-0.02 %
Weights EfficientNet B0	Standard	-0.29 %	+0.98 %
	XOR-MSB	-14.7 %	-22.7 %
	XOR-MSB & Decorr.	-22.7 %	-0.25 %
Pruned Weights ResNet50	Standard	-62.9 %	-79.1 %
	XOR-MSB	-67.4 %	-81.8 %
	XOR-MSB & Decorr.	-81.8 %	-0.02 %
Pruned Weights MobileNet	Standard	-61.5 %	-78.2 %
	XOR-MSB	-64.7 %	-80.1 %
	XOR-MSB & Decorr.	-80.1 %	-0.15 %
Activations ResNet50	Standard	-68.1 %	-56.8 %
	XOR-MSB	-68.1 %	-81.8 %
	XOR-MSB & Decorr.	-81.8 %	-0.02 %
Activations MobileNet	Standard	-27.8 %	-28.9 %
	XOR-ZP	-27.8 %	-50.4 %
	XOR-ZP & Decorr.	-50.4 %	+0.01 %

the other lighter-weight DNNs, the standard deviation of the neural network is smaller. This reduces the possible coding gains. Still, we achieve an optimization by 22.6 %, 18.6 %, and 22.7 % in both power-relevant metrics for *MobileNet V1*, *MobileNet V2*, and *EfficientNet B0*, respectively.

If weight pruning is applied during DNN training—which in that case entails no extra cost nor an accuracy loss [4]—we even achieve power reductions of over 80 %, also for the lighter-weight DNN.

We stated in Section IV of this paper that using *ReLU* activations already reduces the power characteristics versus activation functions for which the 8-bit output tends to be normally distributed. The results show that in fact the power can be reduced by 28 % to 68.1 %. Our proposed technique, which again comes at no accuracy nor noticeable hardware cost, enhances the power savings further to 50.4 % to 81.8 %.

C. ENERGY CONSUMPTION OF MAC

This section studies the effects of the codings on the MAC units. The MACs shown in Fig. 6 are synthesized in a commercial 40-nm technology using different timing constraints, from relaxed to stringent. The power consumption is measured after back-annotated gate-level simulations. As a representative example, we consider the real activations and weights of an *EfficientNet B0* inference. We report the average energy consumption of the MAC units over all convolution layers (including the depthwise layers).

Fig. 10 shows the comparison of energy consumption of the multipliers, the adder tree, and the total energy of the MAC units versus the delay of the units. The energy consumptions in the adder trees are marginally different for the studied coding schemes (see Fig. 10b). In contrast, the coding can result in substantial energy savings in the multipliers (see Fig. 10a). Consequently, as shown in Fig. 10c, applying low-power coding for memories and interconnects, can even provide additional energy savings in the PEs instead of introducing an energy overhead for these components. As an instance, when the activations are coded with XOR-ZP (*uint8*), and the weights are in SM (*int7*), up to 39 % energy saving can be achieved in comparison to the case where no coding is applied (activations and weights in *int8*). It should be noted that this energy saving is in addition to the aforementioned improvements in memory and communication by the proposed low-power coding framework.

VII. CONCLUSION

This work presented a thorough analysis of the bit-level statistics of the parameters and activations in full-integer quantized, *ReLU*-based, neural networks and an ultra-low-cost coding technique, exploiting the gained insight for power savings. At no noticeable cost, the technique can reduce the power consumption of on-chip memories, interconnects, and buffers to up to over 80 %. The technique is effective for any DNNs with power savings expected in the range of 18 %-82 % (cf. Table 2). Achieving its full potential requires to apply standard weight pruning for the DNN weights. In this case, the power savings are shown to be constantly in the range of 80 % (cf. pruning entries of Table 2). On top, the technique can achieve power savings by almost 40 % for the MAC blocks in the processing elements.

Our proposed technique is also efficient for the fully connected layers in LLMs. However, the multi-headed dot-product attention modules due to the effect of SoftMax activations need further investigation in future work. For future work, we will also extend this work to further enhance the coding gains for other activations functions that are not a mere clipping such as leaky *ReLU*, or *(H)Swish*. Moreover, we plan to exploit the data statistics also for hardware- and power-efficient compression.

REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[2] B. Jacob et al., "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2704–2713.

[3] A. G. Howard et al., "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.

[4] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015, *arXiv:1510.00149*.

[5] A. Garcia-Ortiz, L. Bamberg, and A. Najafi, "Low-power coding: Trends and new challenges," *J. Low Power Electron.*, vol. 13, no. 3, pp. 356–370, 2017.

[6] M. Stan and W. Burleson, "Bus-invert coding for low-power I/O," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 3, no. 1, pp. 49–58, Mar. 1995.

[7] N. Jafarzadeh, M. Palesi, S. Eskandari, S. Hessabi, and A. Afzali-Kusha, "Low energy yet reliable data communication scheme for network-on-chip," *IEEE Trans. Comput. Aided Design Integr. Circuits Syst.*, vol. 34, no. 12, pp. 1892–1904, Dec. 2015.

[8] S. Ramprasad, N. Shanbhag, and I. Hajj, "A coding framework for low-power address and data busses," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 7, no. 2, pp. 212–221, Jun. 1999.

[9] T. Murgan, P. B. Bacinschi, A. G. Ortiz, and M. Glesner, "Partial bus-invert bus encoding schemes for low-power DSP systems considering inter-wire capacitance," in *Proc. 16th Int., Int. Workshop Power Timing Model., Optim. Simul., PATMOS*, 2006, pp. 169–180.

[10] L. Benini, G. De Micheli, E. Macii, D. Sciuto, and C. Silvano, "Asymptotic zero-transition activity encoding for address busses in low-power microprocessor-based systems," in *Proc. Great Lakes Symp. VLSI*, 1997, pp. 77–82.

[11] L. Benini, G. De Micheli, E. Macii, M. Poncino, and S. Quer, "Power optimization of core-based systems by address bus encoding," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 6, no. 4, pp. 554–562, Dec. 1998.

[12] H. Fujiwara et al., "A two-port SRAM for real-time video processor saving 53% of bitline power with majority logic and data-bit reordering," in *Proc. Int. Symp. Low Power Electron. Design*, 2006, pp. 61–66.

[13] M. E. Sinangil and A. P. Chandrakasan, "Application-specific SRAM design using output prediction to reduce bit-line switching activity and statistically gated sense amplifiers for up to 1.9x lower energy/access," *IEEE J. Solid-State Circuits*, vol. 49, no. 1, pp. 107–117, Jan. 2014.

[14] C. Peltekis, D. Filippas, G. Dimitrakopoulos, and C. Nicopoulos, "Low-power data streaming in systolic arrays with bus-invert coding and zero-value clock gating," in *Proc. 12th Int. Conf. Modern Circuits Syst. Technol. (MOCAST)*, 2023, pp. 1–4.

[15] L. Bamberg and A. Garcia-Ortiz, "High-level energy estimation for submicrometric TSV arrays," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 10, pp. 2856–2866, Oct. 2017.

[16] P. E. Landman and J. M. Rabaey, "Architectural power analysis: The dual bit type method," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 3, no. 2, pp. 173–187, Jun. 1995.