LAXOR: A Bit-Accurate BNN Accelerator with Latch-XOR Logic for Local Computing

Dongrui Li[1, 2*], Tomomasa Yamasaki[1*], Aarthy Mani[2], Anh Tuan Do[2], Niangjun Chen[1], Bo Wang[1]
[1]Singapore University of Technology and Design
[2]Institute of Microelectronics, Agency for Science, Technology & Research (A*STAR), Singapore

***Abstract*—Binary Neural Network (BNN) accelerators are attractive solutions for Artificial Internet-of-Things (AIoT) applications thanks to the compact models and low computational cost while maintaining satisfactory classification performance. Various analog/mix-signal compute-in-memory macros have been proposed to boost the energy efficiency of binary convolution tasks. However, this approach incurs inaccurate computation due to its sensitivity to temperature, noise, and process variations. In this work, we present a full-digital BNN architecture that leverages a novel Latch-XOR logic array for local bitwise multiplication, suppressing massive data movement and achieving 4.2× lower energy per operation compared to the decoupled standard cell approach. An optimized population count circuitry is also proposed for data accumulation, which obtains 1.37× Energy-Delay-Area saving compared to Binary-Adder-Tree-based implementation. To enable seamless hardware-software co-optimization, we have developed an in-house simulator for design space exploration as well as flexible mapping with various network topologies and kernel sizes. Our experiment shows the Latch-XOR-based architecture in 28nm CMOS technology achieves an enhanced energy efficiency of 2315 $TOPS/W$, 3.4× higher compared to the state-of-the-art synthesized digital architecture. This manifests that the proposed accelerator is highly suited for AIoT applications.**

I. Introduction

Domain-specific hardware accelerators for Deep Neural Networks (DNNs) have become prevailing due to their optimized architectures, data flow, and circuits to enhance critical design metrics. Binary Neural Networks (BNNs), known as single-bit-precision DNNs, have demonstrated satisfactory accuracy in a variety of workloads [1], [2]. The weights and activations of BNNs are binarized to either '+1' or '–1' resulting in tiny model size and lightweight convolution implemented by XNOR/XOR and population count operations [1]. Consequently, its hardware implementation requires a much smaller memory capacity and dissipates substantially less power compared to multi-bit precision, real-valued neural networks [3], [4].

Contemporary on-chip BNN acceleration falls into two major categories: the analog/mixed-signal Compute-In-Memory (CIM) circuitry and the full-digital approach. In CIM-based architectures [5]–[7], parallel read-out of multiple memory rows allows for one-shot weighted sum accumulation (i.e., convolution), leading to considerably improved energy efficiency. However, the analog/mixed-signal approach is susceptible to temperature, noise, and process variation, causing potential inaccurate computation and thus accuracy degradation [4]. To avoid this, pure-digital BNN accelerators were proposed [4], [8]. Nevertheless, these works usually separate local weight storage from XNOR/XOR operation, resulting in intensive data movement, which deteriorates energy efficiency.

Another challenge in neural network accelerator design is how to effectively and accurately predict the hardware performance at the application level, under a wide variety of datasets and design parameters of the architecture. Moreover, designers must capture the trade-offs between performance and energy consumption early to reduce the rounds for hardware optimization. To achieve this, various simulators for CNN accelerators have been proposed [9]–[11]. Nevertheless, these works rarely model BNN topologies and their bitwise multiplication (i.e., XNOR or XOR). HAWIS [12] provides a generator for BNN models and estimates their energy consumption. However, the architecture primitive is based upon a ReRAM model and thus not very useful for synthesized, bit-accurate digital implementation.

In view of the above, we propose LAXOR, a BNN accelerator with a cycle-accurate simulator for hardware-software co-optimization. The essence of LAXOR lies in a novel local computing paradigm that fuses the weight storage (i.e., latch) and the compute unit (i.e., XOR gate) in a single logic to minimize data movement, achieving 4.2× lower energy consumption. We have validated our architecture with a variety of workloads, showing an accuracy of 65.21% for CIFAR-100 and 85.25% for CIFAR-10 with a total energy per classification of 103.74 μJ and 3.82 μJ, respectively in 28nm CMOS technology. In particular, LAXOR consums 1.8× lower energy compared to the state-of-the-art digital implementation [3] with the CIFAR-10 workload. Assisted with the optimized population count circuits, the proposed accelerator attains an energy efficiency of 2315 $TOPS/W$, 3.4× $\sim$ 37.8× higher compared to the advanced BNN accelerator architectures [13], [3], [14], respectively.

II. Architectural Overview

A. Many-core Architecture

To enable modular hardware implementation and network mapping, we propose a many-core architecture with compact XOR arrays and energy-efficient population count (i.e., pop-count) logic for local computing, whose detailed implementa-

This research is supported by the Ministry of Education, Singapore (MOE-T2EP50122-0024), National Research Foundation, Singapore (NRF-CRP23-2019-0003), and SUTD (SRT3IS20162, SKI 2021_02_06). *Both authors contributed equally to this work.

979-8-3503-1175-4/23/$31.00 ©2023 IEEE

tion will be discussed in Section III. As shown in Fig.1(a), the architecture consists of 4 Processing Engine (PE) clusters, a global controller and configuration unit, an accumulation unit to calculate total sums for activation layers or Fully-Connected (FC) layers, and a comparison block to determine the final inference result based on the maximum value. Specifically, the PE clusters account for most of the compute workload, the power consumption as well as the silicon area. Each cluster consists of 64 PEs where every single PE incorporates a Latch-XOR array (i.e., 1024 cells) for bitwise multiplications, a popcount unit that performs kernel-level accumulation, and a 9-bit bias unit for kernel mapping compensation or activation bias. The output of the popcount logic is connected to a local activation unit for the binarization, generating output feature maps. The three components are closely integrated into a single PE to minimize data movement. We also employ an input buffer (i.e. IB_{Ci} in Fig.1) to load weight or input activation before loading them to the PEs in a time-multiplexing manner. The datapath supports various kernel sizes (e.g., 3×3, 4×4, etc.), and their mapping methodology will be illustrated hereafter.

B. Dataflow Mapping

As most BNN models adopt small kernel dimensions, ranging from 1×1 to 11×11 [15], we enable the proposed LAXOR architecture with a flexible dataflow mapping scheme to support them correspondingly.

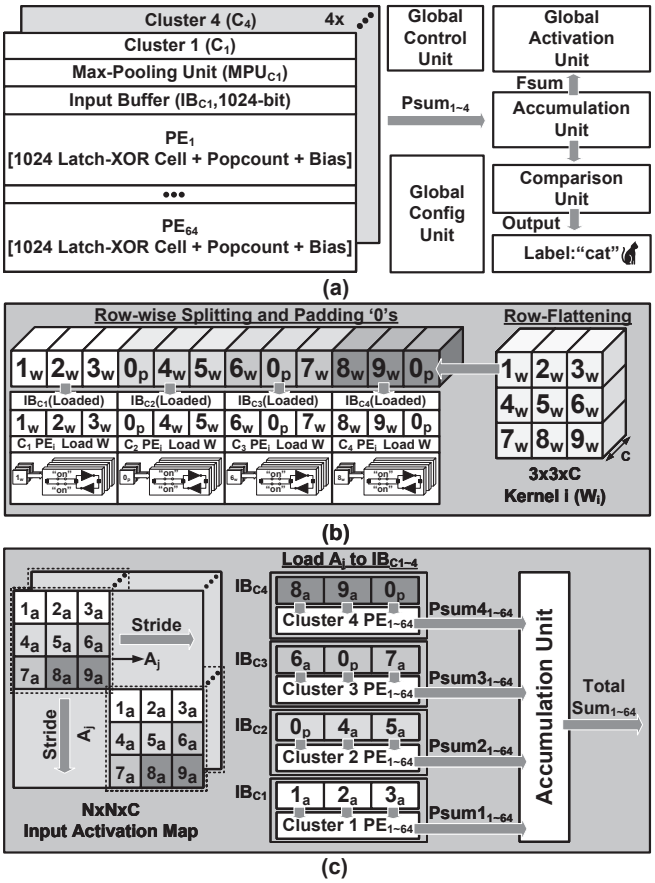

Fig. 1. Block diagram of the (a) many-core architecture (b) Dataflow mapping scheme for odd-sized weight and (c) Dataflow mapping scheme for convolution operation.

Our mapping strategy is to leverage the parallelism of the 4 clusters. For a 3D odd-sized kernel W_i (e.g., $(2N+1)^2\times C$, where C is the number of channels and N is a non-negative integer), we can rewrite the equation into $(4N^2+4N+4)\times C-3\times C$ while 3 groups of '0' are padded, each with a size of C. By doing so, we transform it into a $(4N^2+4N+4)\times C$ shape before equally splitting it into four segments and mapping them onto the corresponding PEs in the clusters.

Specifically, by taking a 3D $3\times3\times C$ kernel as an example, we partition the transformed $12\times C$ kernel into four cluster windows evenly. As the size of each partition does not exceed the input buffer partition size, we then map the kernel data onto the 4 input buffers (i.e., $IB_{C1}\sim IB_{C4}$), each corresponding to an input feature pixel in the window (Fig. 1(b)). As shown in Fig. 1(c), after all the kernels have been loaded into the PEs, the input activation patch A_j will also be transformed, following the same way as W_i. The input buffer is reused to load the input activation and broadcast them to all the PEs within one cluster. Concurrently, the bias is loaded to each cluster to compensate for the transformation loss (i.e., pad $3\times C$ with '0' to Psums). Thereafter, the PEs are enabled to perform convolution operations and generate 4 groups of 64 independent partial sums in parallel (i.e., Psum1 $\sim$ Psum4) from the clusters of each 64 PEs. At last, the accumulation unit sums up all the partial sums to produce 64 total sums (i.e., Fsums). On the other hand, for a 3D even-sized kernel W_i (e.g., $4\times4\times C$), it can be partitioned into four segments and mapped onto four clusters directly.

III. Local Computing in PE

A. Compact 10-Transistor Latch-XOR Computing Cell

Convolution is the most energy-exhausting operation in neural network computation [16]. A convolution consists of multiple weight-input multiplications, followed by an accumulation. In BNN, due to its single-bit precision, we can implement the bitwise multiplication efficiently with an inverted-XOR gate. Besides, the accumulation can be realized by popcount logic which counts the number of '0' of the XOR output. Existing BNN accelerators such as [3], [4] implement this binary convolution with separate latches/flip-flops and XOR/XNOR gates, which can be seen in Fig. 2(a). However, this approach results in significant energy overhead due to frequent data movement between the local memory unit and the compute unit.

To improve the overall energy efficiency of the accelerator, we propose a tightly coupled 10T Latch-XOR cell (Fig. 2(b)) in which the computation is in-situ with the data storage for local computing. It comprises a transmission gate, a cross-coupled latch, a two-transistor (i.e., M1, M2) switching path, and an inverter. During the weight loading, the transmission gate is turned on to write the weight to the latch by asserting XOR_EN to high and sending the value to the cell via the IN signal. After all the weights have been programmed to the array, the transmission gates are turned off to isolate IN and the storage

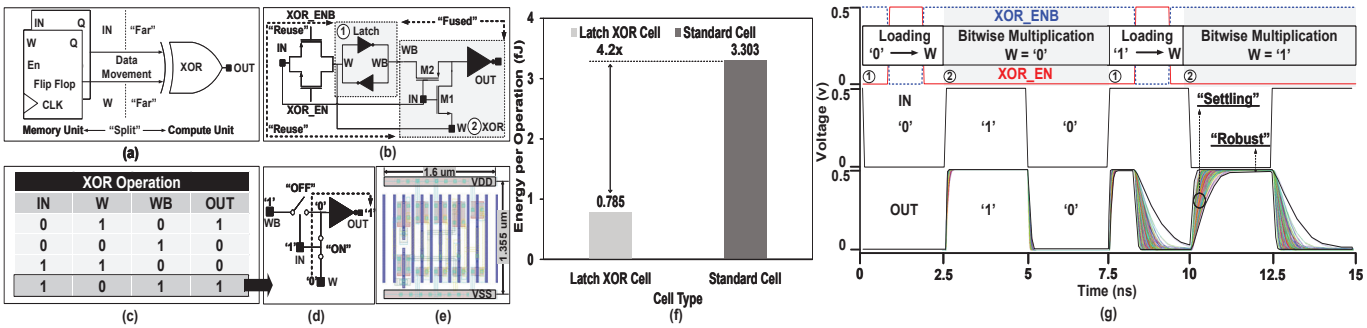

Fig. 2. (a) Standard cell-based separate implementation of weight storage and XOR logic. (b) Proposed 10T Latch XOR cell design. (c) The truth table of XOR operation. (d) Operating principle of (*Weight* $\oplus$ *Input*) logic using 2 transistors and 1 inverter. (e) The layout of the proposed 10T cell. (f) Energy gain of the proposed 10T cell with respect to the design in (a). (g) A waveform sample of 10,000-point Monte-Carlo simulation of the proposed 10-T cell at 0.5 V.

node of the latch. During inference, input data is sent to the cell to perform the XOR operation with two transistors M1, M2, and an inverter, as shown in Fig. 2(b). Note that the IN signal is reused to deliver both weight and data to the cell in a time-multiplexing manner for area-saving purposes.

The truth table (Fig. 2(c)) lists all the input patterns and the corresponding output values for an XOR operation. Specifically, when the weight value is logic '0' (i.e., W is '0' and WB is '1') and the input activation (IN) is logic '1', the NMOS transistor M1 is turned on whereas the PMOS transistor M2 is off. Consequently, the weight passes through M1 and gets inverted so that a logic '1' is generated at the output terminal (OUT) as illustrated in Fig. 2(d). Fig. 2(e) captures the layout of our 10T custom latch-XOR cell, occupying only 1.6μm $\times$ 1.355μm in a 28nm CMOS technology, which is $5\times$ smaller than the overall area of a standard XOR gate with a latch cell using the same technology. As shown in Fig. 2(f), our proposed 10T latch XOR cell only consumes 0.785 fJ energy per operation, $4.2\times$ lower compared to the decoupled standard cell approach. We also verified the reliability of the proposed cell through 10,000 Monte-Carlo simulations with 3-sigma variations on top of the TT corner. Fig. 2(g) plots the timing diagram of the XOR operations as the truth table lists. The output inverter improves the driving capability of the output when a weak '0' passes, enabling the cell to be fully functional at 0.5V, 200MHz against process variations.

B. Popcount Unit PCL Design

Three design strategies have been investigated for the popcount unit implementation, which are Binary Adder Tree (BAT), Look-Up Table (LUT), and Parallel-counter-Carry-Look-ahead (PCL) circuitry. They are illustrated in Fig. 3(a), (b) and (c), respectively. BAT is a common technique for the addition of multiple input operands through a $\lceil N/2 \rceil$ layer reduction tree hierarchy, where N is the number of the operands. However, the computation depends on the height of the tree, leading to a long latency if the hierarchy is deep. LUT can store the numbers of '0's in the 8-bit data patterns without the need to count the bits. For instance, it outputs 8 if the input data pattern is '00000000'. When the data pattern is longer than 8-bit, the logic will compare the first 8-bit section followed

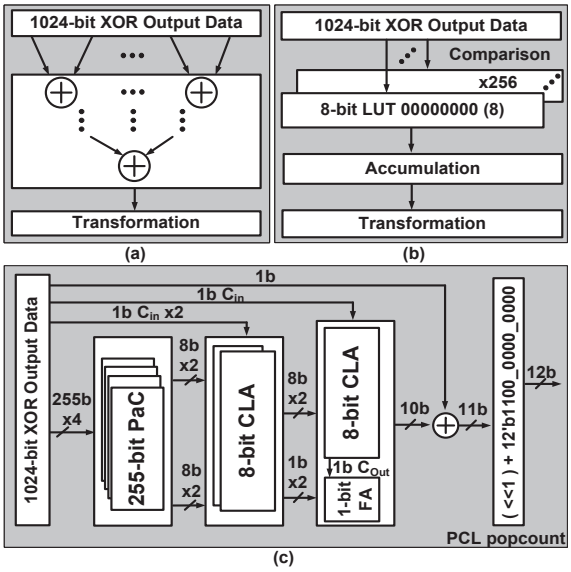

Fig. 3. Block diagram of (a) Binary adder tree (b) Popcount logic using LUT approach and (c) Proposed popcount design using PCL unit.

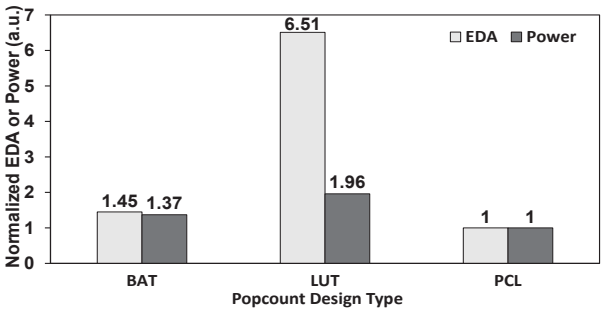

Fig. 4. Popcount design analysis of normalized EDA product and power values of proposed PCL unit vs. conventional BAT and LUT design at 0.9V.

by the next and repeat it till the last section is processed. An accumulator sums up the outputs from all the sections before transforming the counting result into a total sum. However, the LUT approach incurs more power and delay compared to the BAT counterpart, causing a higher Energy-Delay-Area (EDA) product.

As Fig. 3(c) illustrates, the PCL unit is a hybrid combinational circuitry of the parallel counter (PaC) and Carry

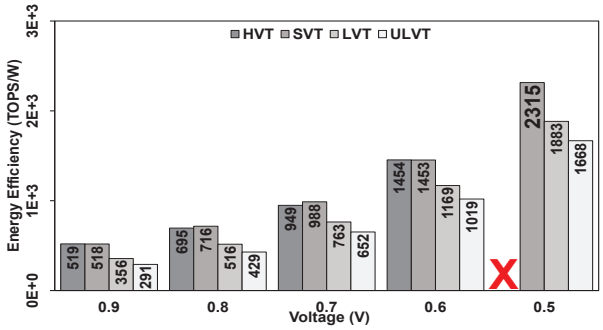

Fig. 5. Energy efficiency of the PE evaluated by sweeping the voltage from 0.9 V down to the minimum voltage required for successful operations at 200 MHz. Note that HVT devices fail at this operating condition.

Look-Ahead adder (CLA) to accomplish the counting and transforming functions. Specifically, the PaC works as a large parallel counter to compress the output of the 1020-bit XOR operation into four 8-bit data so that the two pairs of data packets can be sent to the two CLAs for addition. Thereafter, the results are processed by the third CLA with 1-bit carry-in, generating 10-bit data before being added with the last 1-bit data of the XOR output. Finally, the 11-bit data undergoes a shift-and-add operation to be transformed into the exact 12-bit result by left-shifting one bit and adding with 1100_0000_0000, which is the 2's complement of 1024 in binary.

To achieve higher energy efficiency, we compare the three techniques for the 1024-bit popcount operation on Energy-Delay-Area (EDA) product and power consumption. Fig. 4 reveals that our proposed PCL scheme outperforms the counterparts by 6.51× and 1.45× on EDA, respectively. For power consumption, it can reduce power by a factor of 1.96 and 1.37 compared to them, respectively.

To optimize the energy efficiency of the PE for binary convolution, we have investigated the implementation of the Latch-XOR array using four distinct threshold voltage (i.e., V_{th}) devices and varied the supply voltage from the nominal voltage (i.e., 0.9 V) to the minimum voltage that supports a frequency of 200MHz. As Fig. 5 shows, the Standard V_{th} (SVT) array is fully functional at 0.5 V with the specified frequency and achieves the highest energy efficiency of 2315 $TOPS/W$ among all the device types. Note that the High V_{th} (HVT) macro fails at the operating corner. Consequently, we implement the array with SVT devices for optimal energy efficiency.

IV. Design Analysis with LAXOR Simulator

We design a Python-based simulator for the proposed LAXOR accelerator to (i) map and verify the functionality of a BNN model onto the proposed architecture and (ii) generate application-specific, cycle-accurate results (e.g., latency, energy, utilization, etc.) for design analysis. Apart from that, the simulator supports a variety of BNN models and architectural parameters, ensuring a framework to adapt to a wide range of workloads and hardware configurations for scalability. Specifically, the simulator can emulate all the computation and dataflow in convolution (CONV), max-pooling (MP), and FC layers.

Fig. 6 depicts the overview of the LAXOR simulator, which consists of a front-end tool, Areca, and a back-end tool, Bits-Island. Areca interfaces with the pre-trained model and user configurations before generating the data stream in a format tailored to the accelerator. Bits-Island replicates the LAXOR architecture, maps the data stream onto different PEs, and simulates the functionality layer by layer. Eventually, the toolchain reports the mapping results, layer output, and other critical design metrics. To ensure accurate estimation, latency and energy per atomic operation (e.g., 2-input XOR, 1-bit weight loading) are provided after post-layout simulation as in [17].

Previous DNN energy models [9], [10] emphasize layer-wise dynamic energy but rarely take leakage energy into account. As leakage energy could dominate in low voltage operations, we rectify this and incorporate the leakage energy in our model. As Equation (1) shows, the total energy consumption E_{total} consists of the computation energy E_{cp}, the data movement energy E_{dm}, and the leakage energy E_L. The computations are decomposed into different types of atomic operations so that E_{cp} can be calculated as a sum of the contributions from all the operation categories (e.g., XOR, popcount, 4-input OR, etc). Similarly, the evaluation of data movement energy E_{dm} is based upon the contributions from all dataflow, each is considered as the product of the data amount and its unit movement energy. Total leakage energy E_L is obtained by multiplying total leakage power with the latency of the entire application. This allows us to accurately capture the realistic energy consumption of LAXOR when simulating different networks, datasets and eventually to obtain the optimal design choice. It is worth noting that the simulator runs more than 8×10^6 faster than the Cadence Spectre tool when simulating a 1024b binary convolution, which significantly improves the efficiency of the architectural exploration.

$$E_{total} = \sum_{l \in Layers} (E^l_{cp} + E^l_{dm}) + E_L \tag{1}$$

LAXOR simulator supports a variety of BNN models and architectural parameters. Users can perform a design analysis to profile the impact of these parameters at the architecture level. For instance, we can configure the number of PEs, the

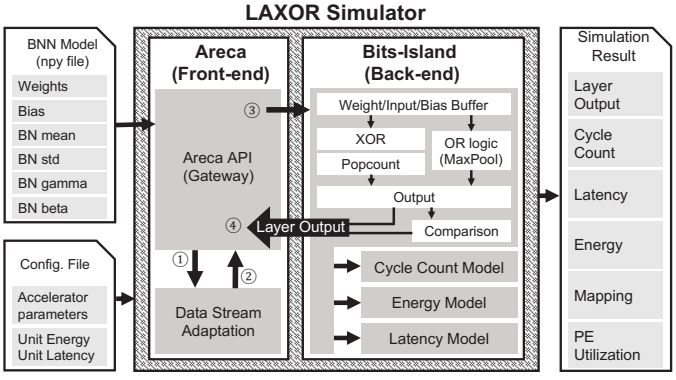

Fig. 6. High-level overview of the LAXOR Simulator.

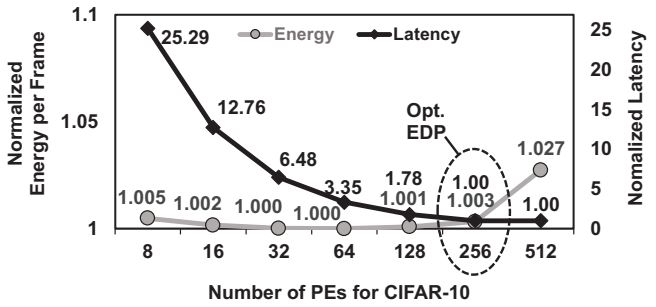

Fig. 7. Design space exploration regarding the number of PEs with 1024 Latch-XOR cells per one PE and 1024-bit buffer with respect to [3].

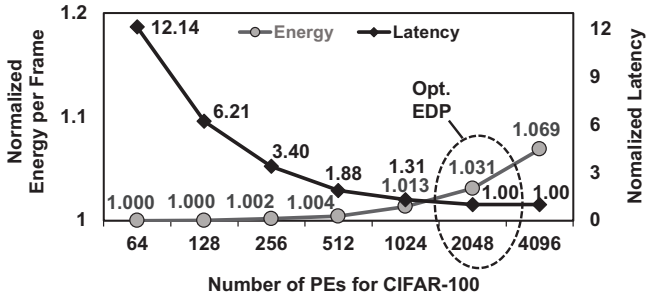

Fig. 8. Design space exploration regarding the number of PEs with 1024 Latch-XOR cells per one PE and 1024-bit buffer with respect to CIFAR-100.

number of Latch-XOR per PE, buffer size, etc. to investigate the performance and energy of the architecture with a given benchmark. Hereafter, we utilize CIFAR-10 as our driving application to demonstrate the design analysis process. The model consists of 8 CONV (256 filters, 2×2 kernel), 2 MP (2×2 kernel), and 1 FC layer. Firstly, we decide the number of PE to deploy in the architecture. We start with 1024 Latch-XOR cells per PE as this size can adequately fit one (2×2×256) kernel in the model. Fig. 7 shows the normalized energy per inference and the latency with different numbers of PEs. With 256 PEs, the mapped application can achieve the lowest Energy-Delay Product (EDP). This is because the 256 PEs configuration provides the highest concurrency to compute all the filters while avoiding leakage overhead from idle ones compared to using 512 PEs. The simulator can search for the optimal configuration subject to network topologies and workloads. Fig. 8 exhibits the normalized energy and latency when running a CIFAR-100 task. The optimal EDP is obtained with 2048 PEs where each PE is configured with 1024 10T cells. Regardless of network models, we adopt a 1024-bit global buffer in our architecture and divide the PEs into 4 clusters due to layout considerations. Our source code for LAXOR simulator is available at https://github.com/tomomasayamasaki/LAXOR.

V. Experimental results

We have implemented various BNN workloads on the LAXOR architecture with the assistance of the in-house developed simulator. The accelerator buffers a 9-bit bias for each filter and accommodates 256 4-input OR logic for 2×2 max pooling operation. Each parameter is set according to the optimal design analysis of the CIFAR-10 workload which is our driving application (see Section IV). As Table I shows, we utilize a four-layer FC network with 1024 neurons per layer for the MNIST inference task and achieve 98.31% accuracy. For the fashion-MNIST workload, the BNN model has more FC layers and can achieve an accuracy of 87.58%. We also implement two BNN models for CIFAR-10 workloads and CIFAR-100 workloads, respectively. The BNN model for CIFAR-10 is from [3] and the BNN topology for CIFAR-100 is inspired by ResNet-18. Our experiment shows that the former can predict the result with an accuracy of 85.25% while the latter can achieve 65.21% and 86.89% for top-1 and top-5 accuracy, respectively.

After being mapped onto the LAXOR accelerator, the MNIST-MLP workload is estimated to consume a total energy of 0.042 μJ per classification with 0.4 μs latency while the Fashion-MNIST workload can consume an energy of 0.069 μJ per classification with 0.7 μs latency. The CIFAR-10 benchmark consumes 3.82 μJ per inference in 0.02 *ms* while the CIFAR-100 task dissipates 103.74 μJ per inference in 0.56 *ms*. Particularly, LAXOR has a 1.8× lower energy consumption compared to the estimated hand-designed digital implementation in [3] when running the CIFAR-10 workload with the same model. Table I lists the simulation output from the tool including accuracy, energy consumption, latency and PE utilization of various workloads. It is worth noting that the energy consumption does not necessarily increase with the model size as different computation types such as CONV or FC also plays an important role. The PE utilization ratios show that the PE clusters in the LAXOR accelerator are efficiently mapped thanks to our mapping scheme.

TABLE I
The Simulated Result of LAXOR

Dataset	MNIST	Fashion MNIST	CIFAR-10	CIFAR-100
Accuracy (%)	98.31	87.58	85.25	65.21 (top-1) 86.89 (top-5)
No. of CONV	0	0	8	15
No. of FC	4	6	1	1
No. of MP	0	0	2	1
Model size (*MB*)	0.79	1.29	0.51	7.45
Dynamic Energy (*uJ*/classification)	0.024	0.040	3.73	101.50
Leakage Energy (*uJ*/classification)	0.018	0.029	0.09	2.24
Total Energy (*uJ*/classification)	0.042	0.069	3.82	103.74
Cycle count	887	1469	4628	112787
Latency (*ms*)	0.004	0.007	0.02	0.56
PE utilization (%)	92.61	95.42	89.45	96.6

Table. II compares the proposed BNN accelerator with state-of-the-art implementations. LAXOR achieves an energy efficiency of 2315 $TOPS/W$ with respect to Multiply-Accumulate (MAC) operation, outperforming all the synthesized digital BNN accelerators in the table, i.e., [3], [13], [14]. In particular, the energy efficiency of LAXOR is 3.4× higher compared to [13] which was implemented with a more advanced technology node. As a mixed-signal design, [19] is energy efficient but

TABLE II
BENCHMARKING LAXOR WITH STATE-OF-THE-ART BNN ACCELERATORS

	ISSCC'22 [18]	ISSCC'18 [19]	CICC'18 [20]	ISCAS'18 [13]	TCAD'18 [14]	JSSC'19 [3]	**This Work**
CMOS Technology	28nm	28nm	28nm	22nm	65nm	28nm	**28nm**
Design Type	Compute-In-Memory	Mixed-Signal	Digital	Digital	Digital	Digital	**Digital**
Result Type	Silicon	Silicon	Silicon	Synthesis	Synthesis	Synthesis	**Synthesis**
VDD (V)	0.5-1.1	0.6-0.8	0.66-0.9	0.65	0.6, 1.2	0.6, 0.8	**0.5-0.9**
Bit Width	1	1	1	1	1	1	**1**
Frequency (MHz)	280M	1.5M-10M	1.5-48M	-	480M	10M	**200M**
Core Area (mm^2)	0.033 (Macro Area)	4.6	1.4	0.46	1.9	-	**2.73**
Performance ($TOPS$)	9.175-20.032 (0.9-1.1V)	0.072-0.478	0.09-2.8	91.12	1.5	0.478	**104.8**
Compute Density ($TOPS/mm^2$)	-	0.0157-0.1039	0.064-2	198.1	0.79	-	**38.388**
MAC Energy Efficiency ($TOPS/W$)	625 @0.5V* (100% toggle rate)	532-772 (0.8-0.6V)	145-230 (0.9-0.66V)	672.6 @0.65V	61.2 @0.6V	299 @0.8V	**2315 @0.5V (100% toggle rate)**
Bit Accurate	No	No	Yes	Yes	Yes	Yes	**Yes**

Note: * Estimated energy efficiency from the graph of the paper [18].

its computation precision can be limited by process, voltage and temperature variations as explained in Section I. CIM design in [18] shows an impressive energy efficiency in silicon with a 25% input toggle rate. However, the metric downgrades significantly when increasing the toggle rate to 100%, which is much lower than ours.

Due to approximate computing, its inference accuracy degrades compared to the exact arithmetic approach even after being compensated with additional approximate-aware training. On the contrary, LAXOR ensures bit-accurate, deterministic computation, maximally free from accuracy loss induced by hardware non-linearity and non-ideality.

VI. CONCLUSION

This paper introduces LAXOR, a flexible BNN accelerator with a novel local computing paradigm. Thanks to the proposed Latch-XOR logic and the architecture-circuit co-optimization, LAXOR achieves an energy efficiency of 2315 $TOPS/W$ at 0.5V, 200MHz for binary convolution operation, outperforming the digital state-of-the-art by 3.4×. Apart from that, we present a Python-based simulator that enables fast design analysis for optimal design points and flexible mapping for a variety of network layers and kernel sizes. With the hardware-software co-design methodology, we demonstrate a full-digital, bit-accurate BNN accelerator that is highly suited for AIoT applications.

REFERENCES

[1] I. Hubara *et al.*, "Binarized neural networks," *Advances in neural information processing systems*, vol. 29, 2016.

[2] M. Courbariaux *et al.*, "Binaryconnect: Training deep neural networks with binary weights during propagations," 2015.

[3] D. Bankman *et al.*, "An always-on 3.8 μJ/86% cifar-10 mixed-signal binary cnn processor with all memory on chip in 28-nm cmos," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 1, pp. 158–172, 2019.

[4] P. C. Knag *et al.*, "A 617-TOPS/W all-digital binary neural network accelerator in 10-nm finfet cmos," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 4, pp. 1082–1092, 2021.

[5] S. Yin *et al.*, "XNOR-SRAM: In-memory computing sram macro for binary/ternary deep neural networks," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 6, pp. 1733–1743, 2020.

[6] W.-S. Khwa *et al.*, "A 65nm 4kb algorithm-dependent computing-in-memory sram unit-macro with 2.3ns and 55.8TOPS/W fully parallel product-sum operation for binary dnn edge processors," in *2018 IEEE International Solid-State Circuits Conference - (ISSCC)*, pp. 496–498, 2018.

[7] S. Xie *et al.*, "16.2 edram-cim: Compute-in-memory design with reconfigurable embedded-dynamic-memory array realizing adaptive data converters and charge-domain computing," in *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 64, pp. 248–250, 2021.

[8] J. Lee *et al.*, "Unpu: An energy-efficient deep neural network accelerator with fully variable weight bit precision," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 1, pp. 173–185, 2019.

[9] T.-J. Yang *et al.*, "A method to estimate the energy consumption of deep neural networks," in *2017 51st Asilomar Conference on Signals, Systems, and Computers*, pp. 1916–1920, 2017.

[10] Y. Zhao *et al.*, "Dnn-chip predictor: An analytical performance predictor for dnn accelerators with various dataflows and hardware architectures," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1593–1597, 2020.

[11] A. Samajdar *et al.*, "Scale-sim: Systolic cnn accelerator simulator," in *https://arxiv.org/abs/1811.02883*, arXiv, 2018.

[12] Q. Tang *et al.*, "Hawis: Hardware-aware automated width search for accurate, energy-efficient and robust binary neural network on reram dot-product engine," in *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 226–231, 2022.

[13] M. Rusci *et al.*, "Design automation for binarized neural networks: A quantum leap opportunity?," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, 2018.

[14] R. Andri *et al.*, "YodaNN: An architecture for ultralow power binary-weight cnn acceleration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 1, pp. 48–60, 2018.

[15] T. Simons *et al.*, "A review of binarized neural networks," *Electronics*, vol. 8, no. 6, 2019.

[16] P. Judd *et al.*, "Stripes: Bit-serial deep neural network computing," in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 1–12, 2016.

[17] B. Wang *et al.*, "Shenjing: A low power reconfigurable neuromorphic accelerator with partial-sum and spike networks-on-chip," in *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 240–245, 2020.

[18] D. Wang *et al.*, "Dimc: 2219TOPS/W 2569F2/b digital in-memory computing macro in 28nm based on approximate arithmetic hardware," in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 65, pp. 266–268, 2022.

[19] D. Bankman *et al.*, "An always-on 3.8 μJ/86% cifar-10 mixed-signal binary cnn processor with all memory on chip in 28nm cmos," in *2018 IEEE International Solid-State Circuits Conference - (ISSCC)*, pp. 222–224, 2018.

[20] B. Moons *et al.*, "Binareye: An always-on energy-accuracy-scalable binary cnn processor with all memory on chip in 28nm cmos," in *2018 IEEE Custom Integrated Circuits Conference (CICC)*, pp. 1–4, 2018.