ISSN:1738-6764, eISSN:2093-7504

Reconfigurable Multi-Precision Multipliers for Energy-Efficient CNN Acceleration for Visual AI in ICT Systems

Muhammad Hamis Haider [1], Hyuk-Jae Lee [2] and Seokbum Ko [3,*]

[1] Department of ECE, University of Saskatchewan; hamis.haider@usask.ca
[2] Department of ECE, Seoul National University; hyukjae@snu.ac.kr
[3] Department of ECE, University of Saskatchewan; seokbum.ko@usask.ca
* Correspondence

https://doi.org/10.5392/IJoC.2025.21.4.001

Manuscript Received 17 September 2025; Received 2 October 2025; Accepted 16 October 2025

Abstract: *Modern convolutional neural networks (CNNs) are essential in information and communication technology (ICT) applications, including edge computing, IoT devices, and mobile platforms, where energy efficiency and throughput are critical. These systems increasingly utilize multi-precision arithmetic to optimize accuracy and resource efficiency. However, traditional methods that assign separate fixed-precision multipliers for different bit-widths are inefficient, as the largest multiplier often dominates the critical path, limiting overall performance. In this paper, we introduce two scalable, power-efficient multiplier architectures with runtime reconfigurability: R4RC16 and R4RC32. These architectures are designed for CNN acceleration under multi-precision pruning. Each design features a low-power mode (8-bit) and a default mode (16-bit for R4RC16 and 32-bit for R4RC32), allowing for dynamic precision adjustment during inference with minimal overhead. When operating in low-power mode, our proposed multipliers achieve up to 7.6× greater energy efficiency compared to state-of-the-art approximate logarithmic multipliers, and up to 13.8× compared to approximate Booth-based designs. Additionally, they provide 2× (for 16-bit) and 4× (for 32-bit) higher throughput than exact 8-bit multipliers when processing pruned CNN workloads. Notably, the overhead in low-power mode is nearly independent of the full bit-width, resulting in a nearly constant power-delay product across both 16-bit and 32-bit designs. These findings highlight the significance of reconfigurable arithmetic units as critical components of ICT infrastructure that support healthcare, education, and multimedia, enabling CNNs to dynamically balance accuracy, energy, and throughput with less than 1% area overhead.*

Keywords: CNN Acceleration; Multi-precision Pruning; Reconfigurable Multipliers; Energy Efficiency; Hardware-aware Inference

1. Introduction

General-purpose processors (GPPs) provide wide programmability but often struggle to match the efficiency of application-specific instruction-set processors (ASIPs) or accelerators when workloads demand both high throughput and low energy. To compensate for this gap, modern platforms increasingly integrate heterogeneous compute units. CPUs are paired with GPUs or tensor cores, where CPUs handle irregular workloads and accelerators provide dense multiply-accumulate operations. Deep neural networks (DNNs), particularly convolutional neural networks (CNNs), are a prime example of workloads that benefit from this approach. CNN inference is dominated by dot-products and convolutions, which are computationally intensive and power hungry. As CNNs are deployed on edge devices, where thermal and energy constraints are critical, the efficiency of hardware multipliers becomes a bottleneck [1-5].

Quantization and pruning are widely used to reduce CNN compute cost. Quantization compresses weights and activations to formats such as 8-bit integers, while pruning eliminates redundant parameters. These techniques reduce memory and energy requirements, but they also create irregular multi-precision needs at the hardware level. For example, many CNN layers operate effectively at 8-bit precision, but certain layers require higher precision for accuracy. Multipliers in modern accelerators must therefore handle variable precisions

efficiently. Implementing both 8-bit and 32-bit multipliers in parallel is wasteful in area and power, and approximate multipliers introduce errors that are unacceptable in accuracy-sensitive layers.

Reconfigurable multipliers offer an effective alternative by providing exact arithmetic across multiple precisions at run-time. Modified Booth multipliers are particularly attractive as a base due to their modular partial-product encoding and scalable architecture [6, 7]. By integrating reconfigurable 4–2 compressors into the Booth array, it is possible to dynamically select which inputs participate in accumulation, thereby reducing switching activity in low-power mode while preserving the timing efficiency of the full-precision mode [8, 9].

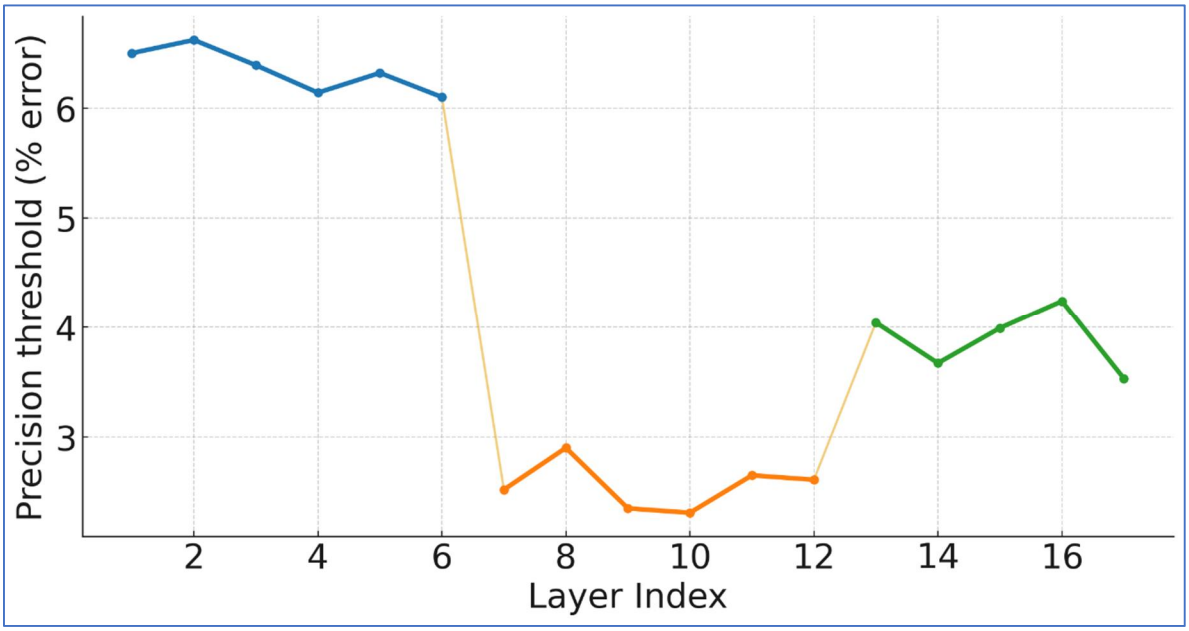

Figure 1. The tolerance of each layer in ResNet-18 model for low-precision computing.

The importance of such hardware innovations extends beyond the immediate domain of CNN acceleration and directly contributes to the broader goals of information and communication technology convergence. Modern ICT applications increasingly rely on embedded intelligence, from real-time vision in autonomous vehicles and smart surveillance to medical diagnostics, natural language interfaces, and immersive multimedia. Each of these domains imposes unique constraints on computation, communication, and energy budgets, yet they all share the need for efficient multi-precision arithmetic at their core processing units. By enabling CNNs to dynamically adjust precision at run time without sacrificing accuracy, reconfigurable multipliers strengthen the ICT infrastructure that supports these diverse applications. They make it feasible to deploy advanced AI models within the tight resource envelopes of edge devices, ensuring that ICT systems can adapt to application-specific requirements while maintaining global scalability and interoperability across domains such as healthcare, education, mobile computing, and industrial automation [10].

In this paper, we propose two reconfigurable Booth multipliers, R4RC16 and R4RC32. Each supports a default precision (16-bit or 32-bit) and a low-power mode (8-bit). The designs target CNN acceleration with multi-precision pruning: low-power mode provides energy savings and throughput gains for quantized layers, while default mode enables exact higher-precision operations when necessary. Key contributions include:

1. Novel reconfigurable 4–2 compressors that reduce switching activity during low-power operation.
2. Scalable multiplier architectures that maintain near-constant power and delay in 8-bit mode, independent of full bit-width.
3. Run-time reconfigurability with minimal configuration overhead, enabling per-layer or per-tile precision scheduling.
4. Energy savings in low-power mode compared to state-of-the-art approximate multipliers, without loss of correctness.

The remainder of the paper is organized as follows. Section 2 reviews related work on multiplier designs and network pruning. Section 3 describes the proposed reconfigurable architectures. Section 4 provides experimental setup details, Section 5 discusses the evaluation results, and finally Section 6 concludes with future directions.

2. Background Knowledge

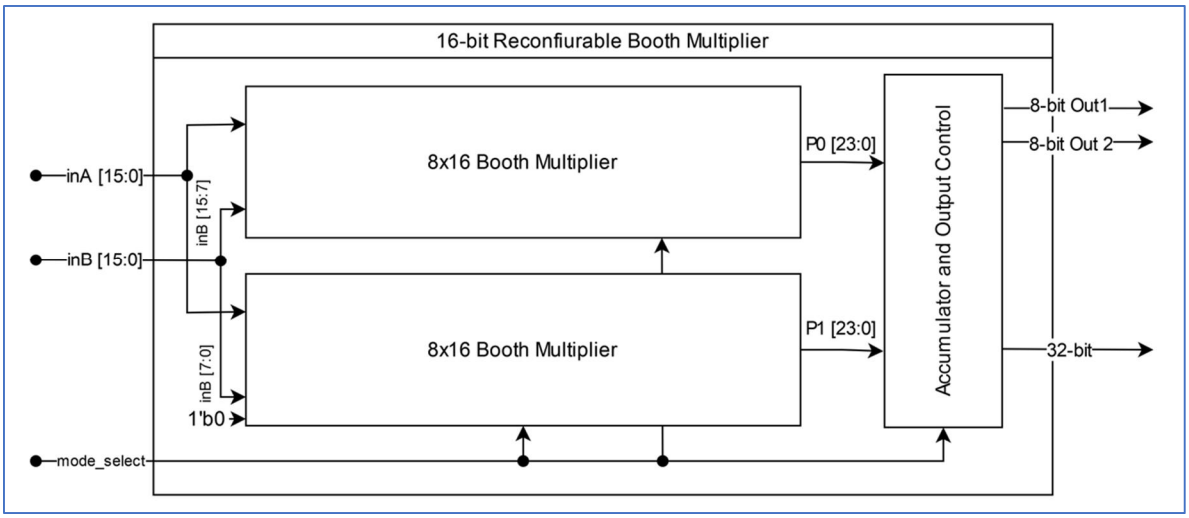

Figure 2. Block diagram of the 16-bit version of the proposed scalable

CNN accelerators typically rely on algorithm–hardware co-optimization to achieve high performance within energy constraints. Techniques such as pruning and quantization have proven effective at reducing both the compute and memory requirements. However, such techniques impose irregular multi-precision demands on hardware. To exploit these opportunities, multipliers must efficiently adapt to different precision requirements without incurring prohibitive overheads.

Modified Booth multiplication remains a well-established architecture for implementing multipliers in modern processors [6]. Booth encoding allows for parallel and modular generation of partial products. The introduction of regular partial product arrays has enabled more efficient adder tree designs [7]. These properties make modified Booth multipliers especially suitable for reconfigurable designs, as the encoding scheme does not change with bit-width and the partial product arrays can be reorganized to support different operating modes.

Compressor networks are crucial in reducing partial products in multipliers. Among them, 4–2 compressors are widely used because they provide efficient parallel reduction. MUX-based 4–2 compressors have been proposed for fast and power-efficient implementations [8]. Furthermore, ultra-low-voltage 4–2 compressors demonstrate that carefully designed gating mechanisms can significantly reduce dynamic power [9]. This makes compressor design a promising target for reconfigurability, enabling low-power operation without sacrificing correctness.

Prior work on reconfigurable multipliers has explored configurable Booth multipliers, recursive decomposition, and asymmetric designs [10-14]. However, many of these approaches introduce considerable area or delay overhead or fail to scale efficiently with bit-width. Approximate multipliers, including both Booth-based and logarithmic multipliers [15-23], achieve energy savings by tolerating errors. While suitable for error-tolerant workloads, they are less general and often unsuitable for mixed-precision inference where accuracy-sensitive layers coexist with quantized layers [24].

Figure 1 illustrates how sensitivity to reduced numerical precision varies across the convolutional layers of ResNet-18. In general, CNNs do not respond uniformly to quantization or rounding errors. different layers tolerate approximation differently depending on their role in the feature hierarchy. The earliest layers, which extract basic edges and textures, show moderate robustness to low-precision computation because their learned filters capture redundant low-level structures [18]. The middle layers, however, exhibit the lowest tolerance: these stages transform low-level patterns into more abstract and discriminative features, and small perturbations here accumulate and propagate, leading to significant drops in classification accuracy. The final layers are somewhat less sensitive than the middle ones, but still less tolerant than the first layers, since distortions introduced near the output cannot be corrected downstream [19].

The proposed R4RC16 and R4RC32 multipliers build upon these foundations by combining the structural advantages of modified Booth encoding with novel reconfigurable 4–2 compressors. The result is a design that provides exact arithmetic across multiple precisions, aligning hardware efficiency with algorithmic multi-precision pruning in CNNs.

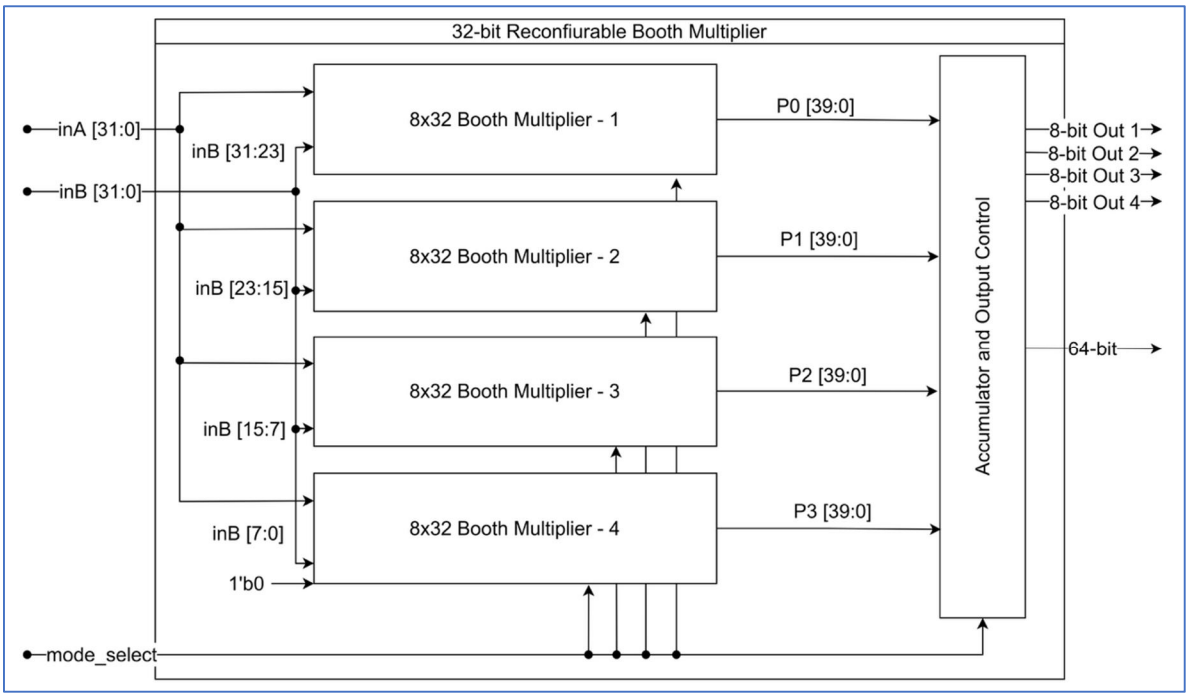

Figure 3. Block diagram of the 32-bit version of the proposed scalable

3. Proposed Reconfigurable Multipliers

3.1 Proposed 16-bit Reconfigurable Multiplier (R4RC16)

The proposed reconfigurable multiplier is a 16-bit Booth multiplier with radix-4 encoding, as shown in Figure 2. It operates in two modes: default (16-bit) and low-power (8-bit) mode. The mode select signal controls the mode of operation of the multiplier. The multiplier is designed using the proposed reconfigurable 4-2 compressor blocks which provide near-constant delay characteristics and enable an area-efficient implementation of the partial product accumulation adder network. The proposed R4RC16 is a reconfigurable signed 16 × 16 multiplier composed of two 8 × 16 radix-4 Booth multipliers. The multipliers are configured as follows.

(a) Radix-4 8 × 16 Multiplier 1: The multiplier takes the 16-bit input at port “inA” and the 9 most significant bits at port “inB”. Fig. 3(b) shows the last four rows of the original partial product array and the 8-bit version with the differences marked in bold cells. The mode is selected via the “mode select” signal.

- When mode select is ON: The internal 8 × 16 multiplier generates the result of the inA[15:0] × inB[15:8] multiplication.
- When mode select is OFF: The internal 8 × 16 multiplier generates the result of the inA[15:8] × inB[15:8] multiplication.

(b) Radix-4 8 × 16 Multiplier 2: The multiplier takes the 16-bit input at port “inA” and the 8 least significant bits at port “inB” with a 0-bit padded at the end. The mode is selected via the “mode select” signal.

- When mode select is ON: The internal 8 × 16 multiplier generates the result of the inA[15:0] × inB[7:0] multiplication.
- When mode select is OFF: The internal 8 × 16 multiplier generates the result of the inA[7:0] × inB[7:0] multiplication.

3.2 Proposed 32-bit Reconfigurable Multiplier (R4RC32)

The same design methodology is applied to the 32-bit version of the proposed reconfigurable multipliers. The architecture is shown in Figure 3. The modules from R4RC16 are reused thanks to their modular and scalable design. The only increase in the critical path occurs due to the extension of the output bit-width in the 8 × 32 internal multipliers. The two inputs for these decoders are the Booth encoding signals generated from the “ib” slices and the “ia”. Both the inputs are relatively concurrent, so there is only a slight increase in the

delay of the multiplier when it is being operated in the low-power (8-bit) mode. It has the following internal components.

(a) Radix-4 8 × 32 Multiplier 1: The multiplier takes the 32-bit input at port "inA" and the 9 most significant bits at port "inB". The mode is selected via the "mode select" signal.

- When mode select is ON: The internal 8 × 32 multiplier generates the result of the inA[31:0] × inB[31:24] multiplication.
- When mode select is OFF: The internal 8 × 32 multiplier generates the result of the inA[31:24] × inB[31:24] multiplication.

(b) Radix-4 8 × 32 Multiplier 2: The multiplier takes the 32-bit input at port "inA" and the 8 least significant bits at port "inB". The mode is selected via the "mode select" signal.

- When mode select is ON: The internal 8 × 32 multiplier generates the result of the inA[31:0] × inB[23:16] multiplication.
- When mode select is OFF: The internal 8 × 32 multiplier generates the result of the inA[23:16] × inB[23:16] multiplication.

(c) Radix-4 8 × 32 Multiplier 3: The multiplier takes the 32-bit input at port "inA" and the 9 most significant bits at port "inB". The mode is selected via the "mode select" signal.

- When mode select is ON: The internal 8 × 32 multiplier generates the result of the inA[31:0] × inB[15:8] multiplication.
- When mode select is OFF: The internal 8 × 32 multiplier generates the result of the inA[15:8] × inB[15:8] multiplication.

(d) Radix-4 8 × 32 Multiplier 4: The multiplier takes the 32-bit input at port "inA" and the 8 least significant bits at port "inB". The mode is selected via the "mode select" signal.

- When mode select is ON: The internal 8 × 32 multiplier generates the result of the inA[31:0] × inB[7:0] multiplication.
- When mode select is OFF: The internal 8 × 32 multiplier generates the result of the inA[7:0] × inB[7:0] multiplication.

4. Experimental Setup

Table 1. Resource analysis of R4RC16 compared to 16-bit multipliers

Multiplier (16-bit)	Power (µW)	Area (µm²)	Delay (ns)	PDP (fJ)	Rel. PDP	NMED (×10⁻³)
Exact 16-bit	25.71	2146.00	1.54	39.59	0.845	None
R4RC16*	28.73	2160.00	1.63	46.84	1.000	None
Recursive 16-bit	42.00	2678.87	1.81	76.08	1.624	None
TL16-8/3 [22]	18.02	962.39	1.20	21.57	0.461	17.14
Mitchell [20]	27.26	1374.96	1.70	46.26	0.988	10.04
ILM-AA [19]	21.41	1170.27	1.57	33.65	0.718	6.85

The experimental setup for evaluating the proposed multipliers was designed to capture both algorithmic-level performance in convolutional neural network inference and circuit-level efficiency using standard design flows. To assess the impact of the proposed multipliers on a representative deep learning workload, the CIFAR-100 dataset was selected. CIFAR-100 contains 100 classes with 600 images each, making it a challenging benchmark for image classification tasks due to the fine granularity of the categories and the limited number of training examples per class. This dataset stresses the representational capacity of neural networks and provides a realistic evaluation of hardware multipliers operating under quantization and pruning constraints.

For the convolutional neural network model, ResNet-18 was selected as the benchmark architecture due to its widespread use in ICT applications. ResNet-18 has become a standard backbone for image classification and recognition because it achieves a balance between accuracy and computational efficiency, making it suitable for both cloud-scale servers and resource-constrained edge devices [1, 2]. Its residual block design with shortcut connections enables stable training while limiting network depth, ensuring that it remains computationally affordable without sacrificing representational capacity. ResNet-18 has been integrated into a

variety of ICT-driven domains, including medical image diagnostics, surveillance and security systems, autonomous driving, multimedia content retrieval, and IoT-based smart environments [3-6]. These areas demand efficient convolutional operations under strict power and latency requirements, aligning closely with the hardware-level optimizations targeted in this paper. Compared to larger architectures such as VGG, ResNet-18 avoids excessive parameter counts while still offering generalization power, and unlike lightweight models such as MobileNet, it maintains closer alignment with canonical CNN structures. Since the central aim of this study is to evaluate multipliers under convolutional workloads, ResNet-18 provides a representative and practically relevant model that reflects the needs of ICT systems. Furthermore, as most CNNs depend on multiply-accumulate kernels at their core, the conclusions drawn using ResNet-18 extend broadly to other architectures deployed in modern ICT infrastructures.

Table 2. Resource analysis of R4RC32 compared to 32-bit multipliers

Multiplier (32-bit)	Power (μW)	Area (μm²)	Delay (ns)	PDP (fJ)	Rel. PDP
Exact 32-bit	155.00	8643.00	3.05	472.75	0.837
R4RC32*	178.25	8647.00	3.17	565.05	1.000
Recursive 32-bit	187.16	9425.23	3.52	658.57	1.166

Referring back to Figure 1, we can understand that during inference of ResNet-18 on hardware equipped with the proposed multi-precision multiplier, the arithmetic unit would dynamically adjust its precision mode as the computation moves through different sections of the network. Based on the tolerance profile in the figure, the early convolutional layers (layers 1–6) can be computed in a reduced-precision mode with minimal accuracy loss, the middle layers (7–12) require higher precision due to their low tolerance to error, and the final layers (13–17) demand the maximum precision to protect output fidelity. This segmentation implies that the multiplier would switch modes twice during a single inference pass: once when moving from the early section to the middle section (low precision → higher precision), and again when transitioning from the last section of the previous inference to the start of the new inference sample (higher precision → low precision). Such structured switching reduces unnecessary use of high-precision computation in tolerant regions while still safeguarding accuracy in sensitive stages, striking a balance between efficiency and reliability. This same activity logic is applied to all multi-precision multipliers included in our experiments.

The hardware evaluation of the proposed R4RC16 and R4RC32 multipliers was performed using a standard 65 nm TSMC CMOS technology library. The designs were synthesized with Synopsys Design Compiler to generate gate-level netlists and timing reports. After synthesis, simulation was carried out using Synopsys VCS to capture the switching activity of the circuits under representative workloads. The generated waveform files from VCS were provided to Synopsys Power Compiler for accurate power estimation. This flow ensures that the reported power and delay values reflect realistic implementation characteristics in silicon, as switching activity is derived from actual execution traces rather than static assumptions. The choice of 65 nm technology was made to allow comparison with prior works in the literature, which frequently use the same technology node for benchmarking.

5. Results and Discussion

The proposed designs are evaluated against both exact and approximate multipliers from the literature in terms of power, area, delay, power delay product (PDP), and accuracy. The evaluation is organized into three main comparisons corresponding to the 16-bit reconfigurable multiplier, the 32-bit reconfigurable multiplier, and the energy-to-accuracy trade-offs during neural network inference.

Table 1 presents the resource analysis of the proposed R4RC16 compared to exact, recursive, and state-of-the-art approximate 16-bit multipliers. The exact 16-bit multiplier consumes 25.71 μW of power, occupies an area of 2146.00 μm², and has a delay of 1.54 ns, resulting in a PDP of 39.59 fJ and a relative PDP of 0.845. The proposed R4RC16 design consumes 28.73 μW of power, has an area of 2160.00 μm², and a delay of 1.63 ns, giving a PDP of 46.84 fJ with a normalized relative PDP of 1.000. The recursive 16-bit multiplier is less efficient, consuming 42.00 μW of power, 2678.87 μm² of area, and 1.81 ns of delay, which results in a PDP of 76.08 fJ and a relative PDP of 1.624. Approximate multipliers such as TL16-8/3 [22] and ILM-AA [19] show lower PDP values of 21.57 fJ and 33.65 fJ respectively, but they introduce notable numerical mean error distances (NMED) of 17.14×10^{-3} and 6.85×10^{-3}. The Mitchell multiplier [20] demonstrates a PDP of 46.26

fJ but introduces an NMED of 10.04×10^{-3}. The Mitchell-trunc6-C1 [21] has a PDP of 37.29 fJ but an NMED of 11.82×10^{-3}. Similarly, DR-ALM5 [18] achieves a PDP of 44.43 fJ with an NMED of 6.93×10^{-3}. From Table 1, while approximate multipliers reduce PDP, they do so at the cost of accuracy, whereas the proposed R4RC16 provides exact arithmetic with only a modest overhead compared to the exact 16-bit baseline and a much lower overhead compared to recursive reconfigurable designs.

Table 2 evaluates the proposed R4RC32 design against exact and recursive 32-bit multipliers. The exact 32-bit multiplier consumes 155.00 μW of power, has an area of 8643.00 μm², and a delay of 3.05 ns, producing a PDP of 472.75 fJ and a relative PDP of 0.837. The proposed R4RC32 consumes 178.25 μW, requires 8647.00 μm² of area, and achieves a delay of 3.17 ns, resulting in a PDP of 565.05 fJ with a relative PDP of 1.000. By comparison, the recursive 32-bit multiplier consumes 187.16 μW, occupies 9425.23 μm², and has a delay of 3.52 ns, giving a PDP of 658.57 fJ and a relative PDP of 1.166. Table 2 demonstrates that, similar to the 16-bit case, the proposed 32-bit design achieves exact functionality with a lower PDP overhead than recursive reconfigurable multipliers, while maintaining area scalability.

Table 3. Experimental results: Energy efficiency and accuracy

Multiplier	NPM	Bit-widths (I, W)	Energy (fJ)	Accuracy (%)	Energy increase
Exact 8-bit	1	(8,8)	1.420	95.90	0.0000
R4RC16-LP*	2	(8,8)	1.512	95.90	0.0648
R4RC32-LP*	4	(8,8)	1.532	95.90	0.0648
Recursive 16-bit (LP)	2	(8,8)	1.812	95.90	0.2760
Recursive 32-bit (LP)	4	(8,8)	2.362	95.90	0.6634
Reconfigurable [11]	2	(8,8)	1.910	88.45	0.3451
Exact 16-bit	1	(16,16)	9.910	96.40	4.7113
Mitchell [20]	1	(16,16)	8.823	77.11	5.2136
Exact 32-bit	1	(32,32)	114.951	97.20	111.4648

Table 3 provides results for neural network inference, showing the energy consumption and accuracy of multipliers when running an ResNet-18 test case on the CIFAR-100 dataset. The exact 8-bit multiplier achieves 95.90 percent accuracy with an energy of 1.420 fJ per multiplication, normalized to an energy increase of zero. The proposed R4RC16-LP achieves 95.90 percent accuracy with an energy of 1.512 fJ, corresponding to an energy increase of 0.0648. The R4RC32-LP achieves the same accuracy of 95.90 percent with 1.532 fJ of energy, also corresponding to an energy increase of 0.0648. The recursive 16-bit and recursive 32-bit multipliers, in low-power mode, achieve the same accuracy of 95.90 percent but with higher energy consumptions of 1.812 fJ and 2.362 fJ, which correspond to increases of 0.2760 and 0.6634 respectively. The reconfigurable design from [11] consumes 1.910 fJ but suffers a reduced accuracy of 88.45 percent, while the design from [12] consumes 2.070 fJ and achieves only 85.54 percent accuracy. Larger multipliers show dramatic increases in energy consumption. The exact 16-bit multiplier consumes 9.910 fJ with an accuracy of 96.40 percent, which is a 4.7113 energy increase compared to the 8-bit baseline. The Mitchell multiplier [20] consumes 8.823 fJ and achieves 77.11 percent accuracy, corresponding to an energy increase of 5.2136. The exact 32-bit multiplier requires 114.951 fJ of energy to achieve 97.20 percent accuracy, which corresponds to an energy increase of 111.4648.

Figure 4 visualizes the results of Table 3 by plotting the accuracy against the energy consumption for all multipliers except the exact 32-bit design, which was excluded due to its extreme energy consumption of 114.951 fJ. In the figure, the proposed R4RC16-LP and R4RC32-LP are positioned very close to the exact 8-bit multiplier, with only slight increases in energy consumption (1.512 fJ and 1.532 fJ compared to 1.420 fJ) while maintaining the same accuracy of 95.90 percent. Recursive designs are shifted further to the right in the figure, indicating higher energy usage at the same accuracy level. Approximate designs, such as those from [11] and [12], appear lower on the vertical axis due to their reduced accuracies of 88.45 percent and 85.54 percent, despite having energy consumptions of 1.910 fJ and 2.070 fJ. The exact 16-bit multiplier is located further right on the energy axis at 9.910 fJ, showing that while it achieves a slightly higher accuracy of 96.40 percent, it requires over six times the energy of the proposed reconfigurable multipliers operating in low-power mode. The Mitchell multiplier [20] falls even lower in the accuracy dimension at 77.11 percent while consuming 8.823 fJ of energy.

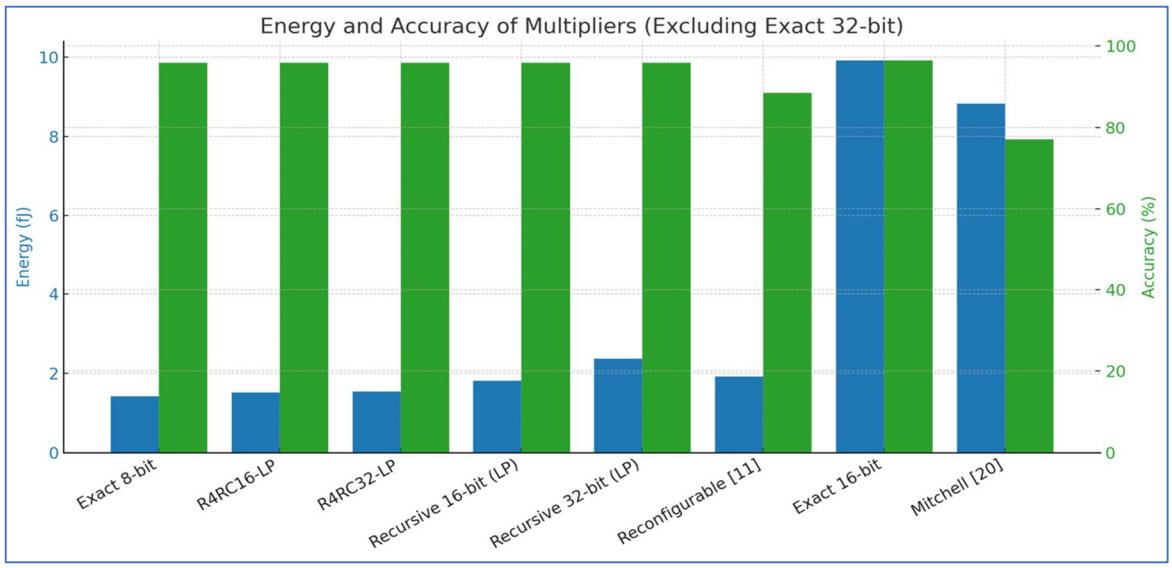

Figure 4. Bar chart visualizing the data from Table 3, showing that the proposed reconfigurable multipliers R4RC16 and R4RC32 perform better than the recursive state-of-the-art works.

6. Conclusions

The results of this paper show that reconfigurable multipliers can deliver exact arithmetic across multiple precisions while maintaining efficiency that is competitive with or superior to existing approaches. The proposed designs demonstrate that it is possible to reduce energy consumption substantially when operating in low-power mode without sacrificing accuracy, while also retaining the ability to perform higher-precision computations when required. This combination of efficiency and flexibility makes the multipliers suitable for a wide range of ICT workloads, particularly those built on convolutional neural networks such as ResNet-18, which is widely adopted in computer vision, image classification, and recognition tasks across different domains.

For ICT systems deployed at the edge, where power and area constraints are critical, multi-precision capability is increasingly necessary. Many convolutional neural networks can tolerate reduced precision in most layers, but they still require higher precision in specific operations to maintain accuracy. Supporting both modes within the same hardware reduces redundancy, avoids the need for separate dedicated multipliers, and improves overall utilization of resources. The use of ResNet-18 in this evaluation highlights its relevance to ICT convergence, since it is commonly used in applications such as medical image analysis, surveillance systems, autonomous navigation, and multimedia content processing. These diverse fields share the requirement of balancing energy efficiency with accuracy, making hardware support for multi-precision inference highly practical.

By offering scalable and power-efficient reconfigurable multipliers, this paper provides a path toward hardware that can more closely support the dynamic requirements of multi-precision AI models within ICT infrastructures. Such designs can enable more practical deployment of deep learning across resource-constrained edge devices, smart sensors, and connected platforms, where flexibility, efficiency, and accuracy must all be preserved within limited energy budgets while supporting the broader convergence of ICT applications.

Acknowledgments: The authors would like to thank NSERC, Canada for the financial support for this project. This research was also supported by Brain Pool program funded by the Ministry of Science and ICT through the National Research Foundation of Korea (RS-2024-00408565, Differentially private AI on edge system through approximate computing).

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

References

[1] A. Reuther, P. Michaleas, M. Jones, V. Gadepally, S. Samsi, and J. Kepner, “Survey and benchmarking of machine learning accelerators,” in 2019 IEEE High Performance Extreme Computing Conference (HPEC), pp. 1–9, 2019.

[2] T. Fritzmann, G. Sigl, and J. Sepúlveda, “RISQ-V: Tightly coupled RISC-V accelerators for post-quantum cryptography,” IACR Transactions on Cryptographic Hardware and Embedded Systems, pp. 239–280, 2020.

[3] M. Asadikouhanjani and S. B. Ko, “Enhancing the utilization of processing elements in spatial deep neural network accelerators,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 40, no. 9, pp. 1947–1951, 2021.

[4] M. Asadikouhanjani, H. Zhang, L. Gopalakrishnan, H. J. Lee, and S. B. Ko, "A real-time architecture for pruning the effectual computations in deep neural networks," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 68, no. 5, pp. 2030–2041, 2021.

[5] S. R. Kuang, J. P. Wang, and C. Y. Guo, "Modified Booth multipliers with a regular partial product array," IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 56, no. 5, pp. 404–408, 2009.

[6] T. Yuan, W. Liu, J. Han, and F. Lombardi, "High performance CNN accelerators based on hardware and algorithm co-optimization," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 68, no. 1, pp. 250–263, 2021.

[7] A. Ardakani, C. Condo, and W. J. Gross, "Fast and efficient convolutional accelerator for edge computing," IEEE Transactions on Computers, vol. 69, no. 1, pp. 138–152, 2020.

[8] V. Mazzia, A. Khaliq, F. Salvetti, and M. Chiaberge, "Real-time apple detection system using embedded systems with hardware accelerators: An edge AI application," IEEE Access, vol. 8, pp. 9102–9114, 2020.

[9] H. Zhang and S. B. Ko, "Design of power efficient posit multiplier," IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 67, no. 5, pp. 861–865, 2020.

[10] H. Zhang, D. Chen, and S. B. Ko, "Efficient multiple-precision floating-point fused multiply-add with mixed-precision support," IEEE Transactions on Computers, vol. 68, no. 7, pp. 1035–1048, 2019.

[11] S. R. Kuang and J. P. Wang, "Design of power-efficient configurable Booth multiplier," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 57, no. 3, pp. 568–580, 2010.

[12] C. Guo, L. Zhang, X. Zhou, W. Qian, and C. Zhuo, "A reconfigurable approximate multiplier for quantized CNN applications," in 2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 235–240, 2020.

[13] F. Qureshi and O. Gustafsson, "Low-complexity reconfigurable complex constant multiplication for FFTs," in 2009 IEEE International Symposium on Circuits and Systems, pp. 1137–1140, 2009.

[14] Z. Shun, O. A. Pfander, H. J. Pfleiderer, and A. Bermak, "A VLSI architecture for a run-time multi-precision reconfigurable Booth multiplier," in 2007 14th IEEE International Conference on Electronics, Circuits and Systems, pp. 975–978, 2007.

[15] C. Guo, L. Zhang, X. Zhou, G. L. Zhang, B. Li, W. Qian, X. Yin, and C. Zhuo, "A reconfigurable multiplier for signed multiplications with asymmetric bit-widths," Journal of Emerging Technologies in Computing Systems, vol. 17, art. 48, 2021.

[16] M. H. Haider and S. B. Ko, "Booth encoding based energy efficient multipliers for deep learning systems," IEEE Transactions on Circuits and Systems II: Express Briefs, p. 1, 2023.

[17] W. Liu, J. Xu, D. Wang, C. Wang, P. Montuschi, and F. Lombardi, "Design and evaluation of approximate logarithmic multipliers for low power error-tolerant applications," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 65, no. 9, pp. 2856–2868, 2018.

[18] P. Yin, C. Wang, H. Waris, W. Liu, Y. Han, and F. Lombardi, "Design and analysis of energy-efficient dynamic range approximate logarithmic multipliers for machine learning," IEEE Transactions on Sustainable Computing, vol. 6, no. 4, pp. 612–625, 2020.

[19] M. S. Ansari, B. F. Cockburn, and J. Han, "An improved logarithmic multiplier for energy-efficient neural computing," IEEE Transactions on Computers, vol. 70, no. 4, pp. 614–625, 2020.

[20] J. N. Mitchell, "Computer multiplication and division using binary logarithms," IRE Transactions on Electronic Computers, no. 4, pp. 512–517, 1962.

[21] M. S. Kim, A. A. Del Barrio, L. T. Oliveira, R. Hermida, and N. Bagherzadeh, "Efficient Mitchell's approximate log multipliers for convolutional neural networks," IEEE Transactions on Computers, vol. 68, no. 5, pp. 660–675, 2018.

[22] R. Pilipović, P. Bulić, and U. Lotrič, "A two-stage operand trimming approximate logarithmic multiplier," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 68, no. 6, pp. 2535–2545, 2021.

[23] R. Pilipović and P. Bulić, "On the design of logarithmic multiplier using radix-4 Booth encoding," IEEE Access, vol. 8, pp. 64578–64590, 2020.

[24] V. Leon, G. Zervakis, D. Soudris, and K. Pekmestzi, "Approximate hybrid high radix encoding for energy-efficient inexact multipliers," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 26, no. 3, pp. 421–430, 2017.