FQP: A Fibonacci Quantization Processor with Multiplication-Free Computing and Topological-Order Routing

Xiaolong Yang†, Yang Wang†, Yubin Qin, Jiachen Wang, Shaojun Wei, Yang Hu and Shouyi Yin*
School of Integrated Circuits, Tsinghua University, Beijing, China, 100084

ABSTRACT

With the continuous advancement of artificial intelligence, neural networks exhibit an escalating parameter size, demanding increased computational power and excessive memory access. Low bit-width quantization emerges as a viable solution to address this challenge. However, conventional low bit-width uniform quantization suffers from a mismatch with the weight and activation data distribution in neural networks, resulting in accuracy degradation.

We propose Fibonacci Quantization, which matches the distribution of weights and activations by using Fibonacci numbers. It achieves negligible accuracy loss for ResNet50 on ImageNet1k with both activations and weights quantized to 4-bit. Based on the Fibonacci Quantization, we present the Fibonacci Quantization Processor. It comprises two types of multiplication-free computing units: the Dualistic-Transformation Adder (DTA) and the Bit-Exclusive Adder (BEA), both capable of transforming the multiplication of Fibonacci numbers into simple addition. In addition, to effectively map multiplications of small and large Fibonacci numbers onto BEA and DTA, we propose Topological-Order Routing (TOR) that routes data either to the previous or current position. Our 4-bit Fibonacci quantization achieves a 0.98% higher accuracy compared with 4-bit uniform quantization for ResNet50 on ImageNet1k. For equivalent accuracy, our proposed processor outperforms uniform quantization with 2.17× higher energy efficiency.

KEYWORDS

Quantization, Fibonacci Numbers, DNN Accelerator, BERT, ResNet

ACM Reference Format:
Xiaolong Yang†, Yang Wang†, Yubin Qin, Jiachen Wang, Shaojun Wei, Yang Hu and Shouyi Yin. 2024. FQP: A Fibonacci Quantization Processor with Multiplication-Free Computing and Topological-Order Routing. In *61st ACM/IEEE Design Automation Conference (DAC '24), June 23–27, 2024, San Francisco, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3649329.3656502

*Corresponding author: Shouyi Yin (yinsy@tsinghua.edu.cn).
† indicates equal contribution.
This work was supported in part by the National Science and Technology Major Project under Grant 2021ZD0114402; in part by the NSFC under Grant 62125403, Grant 92164301, and Grant 62304121; in part by Beijing S&T Project Z221100007722023; in part by the project funding for the 2022 Special Project on Industrial Foundation Reconstruction and High Quality Development of Manufacturing Industry CEIEC-2022-ZM02-0245; in part by the Beijing National Research Center for Information Science and Technology; and in part by the Beijing Advanced Innovation Center for Integrated Circuits.

DAC '24, June 23–27, 2024, San Francisco, CA, USA

ACM ISBN 979-8-4007-0601-1/24/06.
https://doi.org/10.1145/3649329.3656502

1 INTRODUCTION

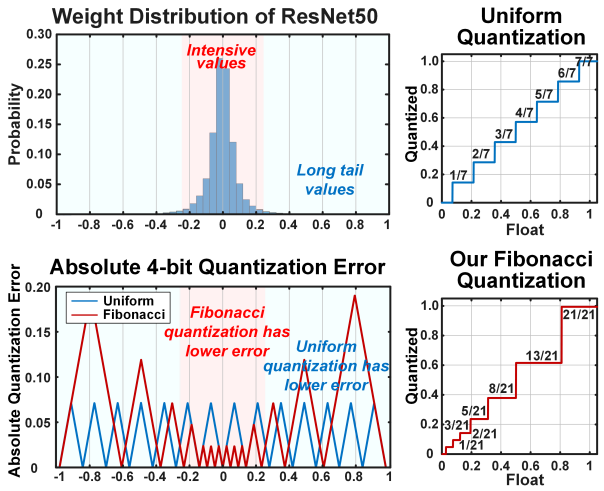


Figure 1: Fibonacci Quantization Matches Better to the Weight Distribution of ResNet50

In the field of artificial intelligence, there is a continuous escalation in the parameter size of neural networks, which in turn demands increased computational power and memory access. For example, ViT-B [1] has 86.6 million parameters and demands 17.6 GFLOPS for computation, which are respectively 3.46 × and 4.30 × more than ResNet50 [2]. This trend presents significant challenges to the application of neural networks.

A common approach to mitigate these challenges is low bit-width quantization [3-5]. This technique involves reducing the precision of the network's parameters, thereby decreasing the computational resources and memory footprint required. However, as shown in Figure 1, a critical limitation of the widely-used uniform quantization is its mismatch with the weight and activation data distribution in neural networks. Typically, weights and activations in neural networks follow a distribution akin to a Gaussian curve, with a dense concentration of smaller weights and a sparse distribution of larger weights. Uniform quantization, failing to account for this non-uniform distribution, often results in significant accuracy degradation. For CNN models, 4-bit uniform quantized ResNet50 has a 1.04% accuracy loss on the ImageNet1k dataset. For Transformer models, 4-bit uniform quantized ViT-B and Bert-Base [6] have respective accuracy losses of 1.22% and 1.17% on the ImageNet1k dataset and SQuAD V1.1 dataset.

To address the shortcomings of uniform quantization, Arbitrary Quantization [7] and Power-Of-Two [8-9] (POT) Quantization have been proposed. However, these methods come with their drawbacks. Arbitrary Quantization, while offering flexibility in value representation, often leads to increased complexity in hardware

implementation due to its lack of a regular pattern. On the other hand, POT Quantization simplifies multiplication operations but can result in suboptimal utilization of the quantization range, leading to potential accuracy losses.

We introduce Fibonacci Quantization to both effectively represent the data distributions by utilizing the non-uniform distribution of Fibonacci numbers and boost hardware efficiency by transforming multiplication to addition. This innovative quantization method exhibits a distribution that is densely populated in the area of smaller values, closely aligning with the typical distribution of neural network weights and activations. It not only offers higher accuracy in low-value regions but also maintains a broader dynamic range, making it highly compatible with the inherent data distribution in neural networks. Remarkably, applying this quantization to ResNet50 and ViT-B results in only 0.04% and 0.51% accuracy loss on the ImageNet1k dataset, with both activations and weights quantized to 4-bit. This demonstrates the method's efficacy in preserving model precision despite reduced bit-width.

We propose two techniques to effectively transform the multiplication of Fibonacci numbers into addition with lower hardware cost: the Dualistic-Transformation Adder (DTA) and the Bit Exclusive Adder (BEA). The DTA is particularly adept at handling the multiplication of large Fibonacci numbers. It achieves this by mathematically transforming the multiplication into the addition of Lucas numbers, thus significantly reducing hardware overhead. On the other hand, the BEA utilizes the characteristic that no more than one of the higher three bits in small Fibonacci numbers is 1, converting the multiplication of smaller Fibonacci numbers into addition, which results in reductions in power consumption and hardware area.

Furthermore, we introduce the Topological-Order Router (TOR) to efficiently map multiplications onto the DTA and BEA. By intelligently routing data to either the current or previous position, the TOR minimizes hardware costs and manages the flow of data in the most resource-effective manner. By integrating the DTA, BEA, and TOR, the power consumption of a single Fibonacci multiplication is only 68% of that of an INT4 multiplication.

In summary, this paper introduces several key innovations in the field of neural network computation and hardware design:

- **Fibonacci Quantization:** We propose Fibonacci Quantization, which aligns with the distribution of weights and activations in neural networks, significantly improving the precision of low bit-width quantization.
- **Multiplication-Free Computing:** We propose the Dualistic-Transformation Adder (DTA) and the Bit Exclusive Adder (BEA) to transform multiplication into addition. The DTA converts the multiplication of large Fibonacci numbers into simple addition of Lucas numbers. Concurrently, the BEA leverages the unique property of the higher three bits in small Fibonacci numbers to transform the multiplication of small Fibonacci numbers into single shifting and addition. The combination of DTA and BEA effectively eliminates the need for resource-intensive multiplication.
- **Topological-Order Routing (TOR):** The TOR intelligently maps multiplications onto the DTA and BEA by restricting routing data to the current or previous position. This routing strategy minimizes hardware costs and efficiently manages the flow of data, ensuring that operations are executed in the most resource-effective manner.
- **Design of the Fibonacci Quantization Processor (FQP):** We design the FQP in TSMC 22nm technology. This processor not only validates the practicality of our approach but also demonstrates superior energy efficiency. When compared to uniform quantization neural network accelerators, our Fibonacci processor achieves 2.17 × greater energy efficiency at equivalent accuracy.

These contributions together open new avenues for energy-efficient AI quantization algorithms and accelerators.

2 METHOD

2.1 Fibonacci Quantization

Fibonacci Quantization quantizes floating-point numbers into discrete Fibonacci numbers. This paper discusses 4-bit Fibonacci Quantization (FIB4). Given that the Fibonacci sequence is 0 1 1 2 3 5 8 13 21 34..., we eliminate one redundant 1 to extend the representational range from -34 to 21. To simplify hardware implementation, we exclude -34, which has a negligible impact on accuracy, thereby setting the maximum range of FIB4 quantization from -21 to 21. The format of FIB4 is shown in Figure 2.

Sign [3]					Fibonacci number id [2-0]											
code	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
val	0	1	2	3	5	8	13	21	0	-1	-2	-3	-5	-8	-13	-21

Figure 2: The Format of FIB4

The quantization of weights is conducted using the following equations:

$$Scale = Clip\_Ratio * Max(|W|)/21 \tag{1}$$

$$W_Q = Fib\_Round(W/Scale) \tag{2}$$

In Equations (1-2), W is the original floating-point weight and W_Q is the quantized weight. $Clip\_Ratio$ is a parameter for adjusting Scale, and Fib_Round is the Fibonacci Quantization function that rounds data to the nearest Fibonacci number. If the data is equidistant from two Fibonacci numbers, it is rounded off randomly. To further simplify hardware implementation, we adjust $Clip\_Ratio$ to ensure that, in each non-overlapping segment with eight quantized weights, at most one quantized weight exceeds 8. Within this constraint, $Clip\_Ratio$ is linearly swept to find the optimal point that minimizes the quantization error. Experiments show that maintaining at most one quantized weight greater than 8 in each non-overlapping segment does not affect accuracy.

For the quantization of activations, we employ the following equations:

$$Zero\_Point = EMA(Mean(A)) \tag{3}$$

$$Scale = Clip\_Ratio * EMA(Max(|A - Zero\_Point|))/21 \tag{4}$$

$$A_Q = Fib\_Round((A - Zero\_Point)/Scale) \tag{5}$$

In Equations (3-5), A represents the floating-point activation, and A_Q denotes the quantized activation values. EMA stands for Exponential Moving Average. We run a set of samples to determine the $Zero\_Point$ and the maximum absolute value of $A - Zero\_Point$.

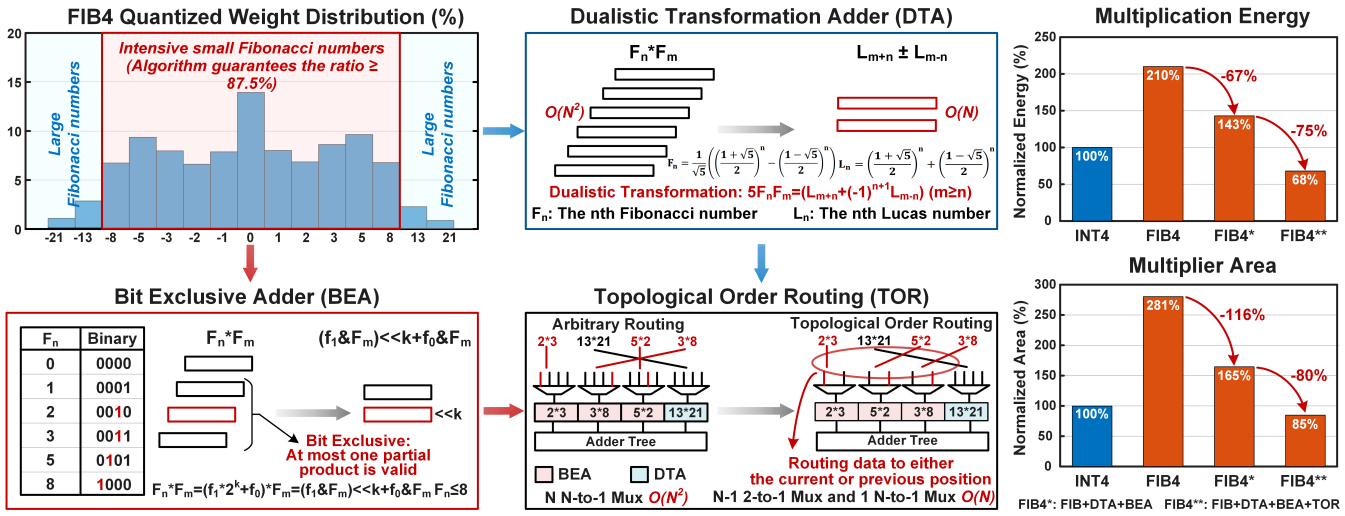


Figure 3: Overview of Our Method

Clip_Ratio is also linearly swept to minimize the quantization error. The activation is quantized before performing convolution or matrix multiplication and is dequantized before entering the nonlinear functions like GeLU, Softmax, and Layernorm.

2.2 Multiplication-Free Computing

2.2.1 Dualistic-Transformation Adder (DTA). In 4-bit Fibonacci Quantization, representing the maximum value necessitates a 6-bit representation, requiring the use of a 6-bit multiplier, which leads to substantial hardware expenditure. Nevertheless, there is a clever Dualistic Transformation Equation (8) to mitigate this issue. The Fibonacci sequence F_n and the Lucas sequence L_n are twin sequences, both fulfilling a recursive relationship where each term is the sum of the two preceding terms. The distinction lies in their initial terms: 0 and 1 for the Fibonacci sequence, compared to 2 and 1 for the Lucas sequence. The general expression for the Fibonacci sequence is presented in Equation (6) and for the Lucas sequence in Equation (7). The Dualistic Transformation Equation (8) converts the multiplication of two Fibonacci numbers, multiplied by five, into the addition of two corresponding terms in the Lucas sequence. A challenge arises in applying this equation: the addition in the Lucas sequence results in a value five times that of the original multiplication. We address this by incorporating the factor of five into the quantization scale, ensuring accurate results during the subsequent dequantization process.

$$F_n = \frac{1}{\sqrt{5}}((\frac{1+\sqrt{5}}{2})^n - (\frac{1-\sqrt{5}}{2})^n) \tag{6}$$

$$L_n = (\frac{1+\sqrt{5}}{2})^n + (\frac{1-\sqrt{5}}{2})^n \tag{7}$$

$$5F_nF_m = L_{n+m} + (-1)^{n+1}L_{m-n}, n \le m \tag{8}$$

2.2.2 Bit-Exclusive Adder (BEA). The DTA, while effective, still incurs significant power and area costs, making it suitable primarily for multiplications involving larger Fibonacci numbers. As shown in Figure 3, after quantizing neural network weights using Fibonacci Quantization, more than 87.5% of the quantized weights fall within small Fibonacci numbers such as 0, 1, 2, 3, 5, and 8. These numbers exhibit a Bit Exclusive property in their high three bits – at most one of these three bits is set to 1, while the others are 0, as demonstrated in Equation (9). This characteristic allows for the optimization of the multiplier by combining three partial products (PP) of high three bits into one PP. Consequently, multiplication can be transformed into a combination of shifting and addition operations, as illustrated in Equation (10).

$$F_n = f_1 * 2^k + f_0 \quad f_1, f_0 \in \{0,1\}, k \in \{1,2,3\}, F_n \le 8 \tag{9}$$

$$F_nF_m = (f_1 \& F_m) << k + f_0 \& F_m \tag{10}$$

2.3 Topological-Order Routing (TOR)

Effectively combining the DTA and BEA significantly reduces the computational overhead of multiplication. Our approach involves using a Processing Element Line (PE Line) consisting of seven BEAs and one DTA. When processing eight inputs, it's essential to map the multiplications of smaller Fibonacci numbers to the BEAs and those of larger Fibonacci numbers to the DTA. However, mapping multiplications to both DTA and BEA poses considerable hardware costs. Traditional 8-to-8 routing methods, requiring 8 8-to-1 Multiplexers (Mux), lead to substantial hardware expenses. The 8-to-8 routing approach entails significant redundancy. In reality, only the DTA needs to select any one multiplication operand from the eight multiplications, while the remaining multiplications can utilize the Topological-Order Routing (TOR). In TOR, each multiplication operand is routed to BEA in either the current position or the previous position. Switching to the BEA perspective, it only needs to select a multiplication operand from its current or next position. This is analogous to having eight cards, where the DTA picks one, and the remaining seven cards are either at their original position or the position of the card immediately below them. Employing this method, we only need 1 8-to-1 Mux and 7 2-to-1 Mux. Such an approach drastically reduces the routing costs.

An effective combination of DTA, BEA, and TOR can reduce the power consumption of a FIB4 multiplication to just 68% of that required for an INT4 multiplication.

3 ARCHITECTURE

3.1 Overall Architecture

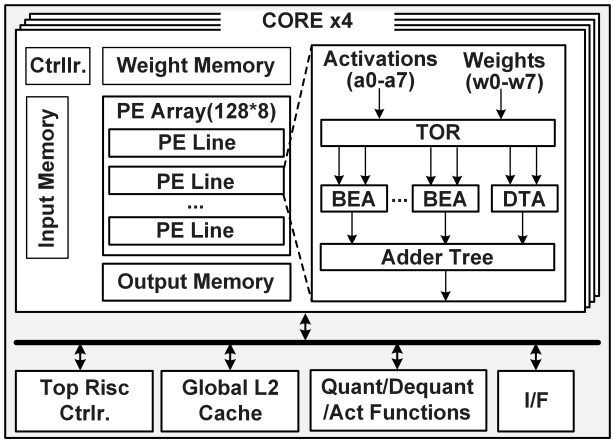


Figure 4: Overall Architecture of the FQP

Figure 4 presents the comprehensive architecture of the FQP, which is controlled at the top level by a tiny RISC CPU. It incorporates a 128KB Global L2 Cache for caching the inputs and outputs of the Core. Additionally, there is a module tasked with executing quantization, dequantization processes, and complex nonlinear functions. The processor features four Cores that perform matrix multiplication and convolution calculations. Each Core is equipped with 20KB of Input Memory, 20KB of Weight Memory, 16KB of Output Memory, and 128 PE Lines. Each PE Line includes a TOR for mapping multiplications onto seven BEAs and one DTA. The outputs from these seven BEAs and the DTA are fed into an addition tree, resulting in an 8-element vector inner product. Matrix multiplication can be obtained through multiple accumulations of this 8-element vector inner product.

3.2 DTA Unit

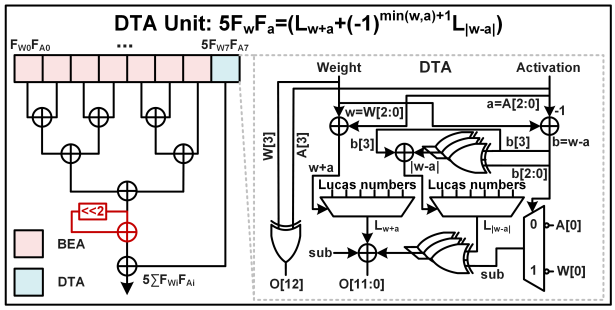


Figure 5: Circuit of DTA Unit

Figure 5 delineates the detailed hardware implementation of the DTA unit. The output of DTA is five times the multiplication of the weight and activation, necessitating modifying the adder tree. This modification involves first obtaining the sum of the outputs of seven BEA units, followed by adding the sum to a right-shifted of itself by two, effectively multiplying the sum by five. This adjustment allows for direct addition to the DTA's output. In the DTA unit's design, we transformed Equation (8) into Equation (11). The initial step is to calculate $w + a$ and $w - a$. We then perform an XOR operation between $w - a$ and its sign, and add this result to the sign of $w - a$, yielding $|w - a|$. Based on the sign of $w - a$, we can identify the smaller of w and a, and determine the value of $(-1)^{min(w,a)+1}$ from the lowest bit of the smaller number. $w+a$ and $|w - a|$ are then processed through a Mux to yield the corresponding L_{w+a} and L_{w-a}. Finally, the correct addition or subtraction based on $(-1)^{min(w,a)+1}$ is performed to obtain the final result.

$$5F_wF_a = L_{w+a} + (-1)^{min(w,a)+1} L_{|w-a|} \tag{11}$$

3.3 BEA Unit

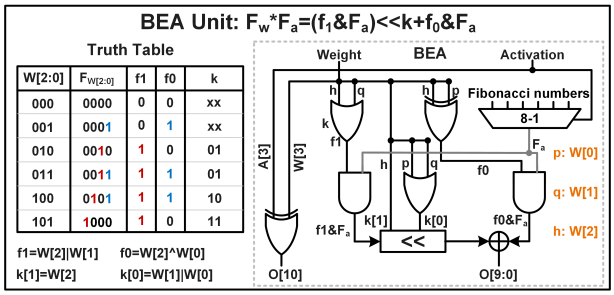


Figure 6: Circuit of BEA Unit

Figure 6 shows the circuit of the BEA unit. The BEA has a relatively simple structure, with its primary focus on generating control signals $f1$, $f0$, and k. Here, $f1$ indicates the presence of a 1 in the high 3 bits of the Fibonacci number decoded from the weight. $f0$ determines if the lowest bit of the Fibonacci number decoded from the weight is 1. The 2-bit k identifies which of the high 3 bits is 1; if $f1 = 0$, it implies there is no 1 in the top 3 bits, allowing k to take any value. A truth table can be used to derive $f1 = W[2] \vee W[1]$, $f0 = W[2] \oplus W[0]$, $k[1] = W[2]$, and $k[0] = W[1] \vee W[0]$. Using these control signals, we first decode the activation into a Fibonacci number using a Mux. An AND operation is then performed with this decoded number and $f1$ and $f0$ respectively. The result from the AND operation with $f1$ is left-shifted by k bits and then added to the result from the AND operation with $f0$ to produce the final output.

3.4 TOR Unit

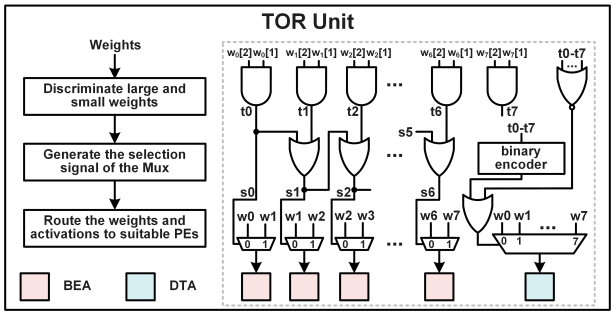


Figure 7: Circuit of TOR Unit

As shown in Figure 7, the TOR unit includes three steps. The first step is identifying which weight has an absolute value greater than

8. This is determined by checking if the pre-decoded low three bits are greater than 5, which can be conveniently done through $W[2] \wedge W[1]$. The second step involves setting the control signals for the Mux. For the DTA's Mux, the control signal converts the one-hot encoding from the first step into binary encoding. If the one-hot encoding is all zeros, indicating no large Fibonacci numbers, the control signal of the DTA's Mux is set to 111. For the BEA's Mux, a serial OR gate moving from left to right generates the control signal. Once a position with a larger Fibonacci number is identified, that position and all subsequent positions have their control signal set to 1. The third step uses the Mux to select the corresponding weights and activations as inputs for the DTA and BEA based on the control signals determined in the second step.

4 EVALUATION

4.1 Software Evaluation Setup

To mitigate the accuracy loss of quantized models, we employ Quantization Aware Training (QAT) for INT4-8 and FIB4 quantization, using the Straight-Through Estimator (STE) method for back-propagating gradients during training. In QAT of Fibonacci Quantization, at the end of each epoch, if any non-overlapping segment of eight quantized weights contains more than one weight with an absolute value greater than 8, the quantization scale is adjusted to ensure that no more than one weight in each segment exceeds 8. ResNet50 is retrained using stochastic gradient descent for 30 epochs, with a learning rate of 10^{-4} and a decay every 10 epochs. ViT-B is retrained on ImageNet1k using the AdamW optimizer for 20 epochs, with a learning rate of 10^{-5}. Bert-Base is fine-tuned on the SQuAD v1.1 for 5 epochs, using the Adam optimizer with a learning rate of 3×10^{-5}.

4.2 Hardware Evaluation Setup

We design the FQP using the TSMC 22nm ULP (Ultra Low Power) process and complete logic synthesis, layout and routing, parasitic parameter extraction from the layout, timing analysis, and power analysis. For the logic synthesis, Synopsys Design Compiler P-2019.03 is employed. The layout and routing are conducted using Synopsys IC Compiler II P-2019.03. Parasitic parameter extraction from the layout is performed with Synopsys StarRC Q-2019.12. Timing analysis is carried out using Synopsys PrimeTime P-2019.03, and the power analysis is executed utilizing the PTPX tool within PrimeTime. The energy efficiency of INT4 and INT8 is obtained by replacing the PE Line in the FQP and evaluating with the PTPX. The energy efficiency for INT5-7 is evaluated by precisely simulating the power consumption of the multiplier and the adder, combined with the write/read power consumption per bit of registers, SRAM, and DRAM.

4.3 Accuracy Evalution

As detailed in Figure 8, FIB4 quantization significantly outperforms INT4 in terms of accuracy, close to the accuracy of INT6. For the ResNet50 model quantized with FIB4 on the ImageNet1k dataset, the image classification accuracy is close to the FP32 baseline, equivalent to that of INT6 quantization, while INT4 quantization experiences a precision loss greater than 1%. This underscores

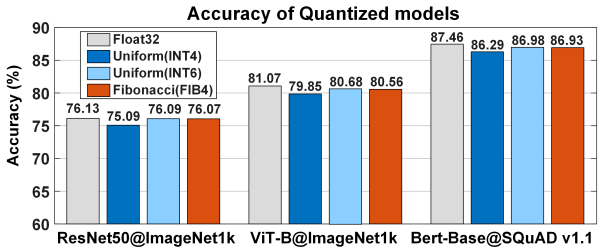


Figure 8: Accuracy of Quantized Models

the effectiveness of FIB4 quantization in CNN models. The ViT-B model, quantized with FIB4, shows a 0.71% higher accuracy on ImageNet1k compared to the model quantized with INT4, demonstrating the superiority of FIB4 quantization in visual transformer models over INT4. Bert-Base quantized with INT4 on the SQuAD V1.1 dataset for the reading comprehension task, incurs a 1.15% accuracy loss compared to the FP32 baseline, whereas FIB4 quantization results in only a 0.53% accuracy loss, closely matching the accuracy of INT6. These experimental results clearly illustrate that 4-bit Fibonacci Quantization significantly surpasses 4-bit uniform quantization in accuracy, achieving precision comparable to 6-bit uniform quantization.

4.4 Performance, Power, and Area Evalution

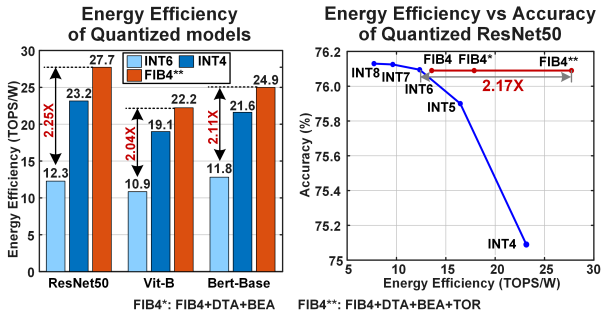
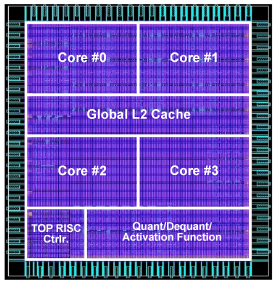


Figure 9: Energy Efficiency of Quantized Models



	Specifications
Technology	22nm CMOS
Die Area(mm²)	4.53
SRAM(KB)	352
Voltage(V)	0.6-0.9
Frequency(MHz)	150-800
Data Precision	FIB4
Power(mW)	49.2-366.2
Peak Performance (TOPS)	6.55
Energy Efficiency (TOPS/W)	17.9-31.7

Figure 10: Layout and Parameters of the FQP

Figure 9 shows the evaluation of energy efficiency across three models using different quantization types. It is evident that FIB4, optimized with DTA, BEA, and TOR, exhibits higher energy efficiency compared to INT4 and is more than twice as efficient as INT6 in terms of energy efficiency. This demonstrates that FIB4, when optimized with these components, consumes significantly less energy than INT6 while maintaining comparable accuracy, as

Table 1: Comparison with State-of-the-art Processors

	ISSCC'20 [10]	ISSCC'21 [11]	ISSCC'21 [12]	JSSC'23 [4]	This Work
Technology (nm)	7	28	7	5	**22**
Die Area (mm^2)	3.04	1.9	19.6	0.153	**4.53**
Supply Voltage (V)	0.575-0.825	0.6-0.9	0.55-0.75	0.46-1.05	**0.6-0.9**
Frequency (MHz)	290-880	100-470	1000-1600	152-1760	**150-800**
Precision	INT8	INT8	FP32/FP16/FP8, INT4/INT2	INT8/INT4	**FIB4**
Power (mW)	174-1053	19.4-131.6	0.87-74.9	NA	**49.2-366.2**
Peak Performance (TOPS)	3.6	1.43[2)]	64-102.4@INT4	3.6[3)]@INT4	**6.55**
Energy Efficiency (TOPS/W)	3.42-13.32[1)]	12.1[2)]-17.5[1)2)] 4.32[4)5)]-6.25[1)4)5)]	8.9-16.5[1)]@INT4	95.6[1)3)]@INT4 11.4[1)4)5)]@INT4	**17.9-31.7**[1)4)]
Area Efficiency (TOPS/mm^2)	1.19	0.75[2)] 0.31[4)5)]	3.27-5.22@INT4 0.71[5)]-1.12[5)]@INT4	23.3@INT4[3)] 1.33@INT4[4)5)]	**1.45**

1) Evaluated at minimal voltage. 2) Evaluated with input/output sparsity caused by ReLU. 3) Evaluated with 50% input sparsity.
4) Evaluated with the dense model. 5) Scaled to 22nm Technology.

observed in the previous section. Figure 9 depicts the accuracy versus energy efficiency curves, revealing that even unoptimized FIB4 is already positioned above the uniform quantization curve. With the optimization using DTA, BEA, and TOR, FIB4 achieves an energy efficiency of 2.17× that of uniform quantization at the same level of accuracy.

Figure 10 presents the layout of the FQP and a summary of parameters. Our design occupies an area of 4.53 mm² and can achieve a maximum frequency of 800 MHz at 0.9 V, with a peak performance of 6.55 TOPS. At 0.6 V, it can achieve an energy efficiency of 31.7 TOPS/W. In these evaluations, both multiplication and addition are counted as a single operation.

4.5 Comparsion with State-of-the-art Processors

As detailed in Table 1, the peak energy efficiency of this work reaches 31.7 TOPS/W, markedly outperforming ISSCC'20 [10], ISSCC'21 [11], ISSCC'21[12], and JSSC'23 [4] when scaled to the 22nm process and dense model. The peak energy efficiency of this work stands at a significant 1.92× higher than the INT4 in ISSCC'21 [12] and 2.78× higher than the INT4 in JSSC'23 [4]. In the realm of area efficiency, our work achieves an impressive 1.45 TOPS/mm^2, surpassing ISSCC'20 [10], ISSCC'21 [11], as well as ISSCC'21 [12] and JSSC'23 [4] when scaled to the 22nm process and dense model. The area efficiency of this work is 1.29× higher than the INT4 in ISSCC'21 [12] and 1.09× higher than the INT4 in JSSC'23 [4].

These results demonstrate that the FQP possesses superior energy efficiency and area efficiency compared to state-of-the-art processors. Furthermore, as shown in Section 4.3, the accuracy of quantized models using FIB4 also largely surpasses the accuracy of quantized models using INT4. This indicates that the FQP leads in accuracy, energy efficiency, and area efficiency compared to state-of-the-art processors in the field.

5 CONCLUSION

In this paper, we propose Fibonacci Quantization, which closely aligns with the distribution of weights and activations in neural networks, achieving higher accuracy than uniform quantization at the same bit-width. We design the FQP with three hardware optimization techniques: DTA, BEA, and TOR. Compared to state-of-the-art processors, the FQP achieves leading energy and area efficiency. We anticipate that Fibonacci Quantization will become widely adopted in low bit-width quantization in the future.

REFERENCES

[1] Alexey Dosovitskiy, Lucas Beyer, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

[2] Kaiming He, Xiangyu Zhang, et al. Identity mappings in deep residual networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 630–645. Springer, 2016.

[3] Tailin Liang, John Glossner, et al. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, 461:370–403, 2021.

[4] Ben Keller, Rangharajan Venkatesan, et al. A 95.6-tops/w deep learning inference accelerator with per-vector scaled 4-bit quantization in 5 nm. *IEEE Journal of Solid-State Circuits*, 58(4):1129–1141, 2023.

[5] Wenhao Sun, Grace Li Zhang, et al. Class-based quantization for neural networks. In *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6, 2023.

[6] Jacob Devlin, Ming-Wei Chang, et al. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, pages 4171–4186, Minneapolis, Minnesota, June 2019.

[7] Sijie Zhao, Tao Yue, and Xuemei Hu. Distribution-aware adaptive multi-bit quantization. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9277–9286, 2021.

[8] Sugil Lee, Hyeonuk Sim, et al. Successive log quantization for cost-efficient neural networks using stochastic computing. In *Proceedings of the 56th Annual Design Automation Conference 2019*, DAC '19, New York, NY, USA, 2019. Association for Computing Machinery.

[9] Sebastian Vogel, Jannik Springer, et al. Self-supervised quantization of pre-trained neural networks for multiplierless acceleration. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1094–1099, 2019.

[10] Chien-Hung Lin, Chih-Chung Cheng, et al. 7.1 a 3.4-to-13.3tops/w 3.6tops dual-core deep-learning accelerator for versatile ai applications in 7nm 5g smartphone soc. In *2020 IEEE International Solid-State Circuits Conference - (ISSCC)*, pages 134–136, 2020.

[11] Huiyu Mo, Wenping Zhu, et al. 9.2 a 28nm 12.1tops/w dual-mode cnn processor using effective-weight-based convolution and error-compensation-based prediction. In *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, volume 64, pages 146–148, 2021.

[12] Ankur Agrawal, Sae Kyu Lee, et al. 9.1 a 7nm 4-core ai chip with 25.6tflops hybrid fp8 training, 102.4tops int4 inference and workload-aware throttling. In *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, volume 64, pages 144–146, 2021.