

# Unary Positional System: Flexible Balance of Hardware Area and Performance

Zeshi Liu<sup>1,\*</sup>, Zheng Weng<sup>1,2</sup>, Ruijie Tan<sup>1</sup>, Guangming Tang<sup>1</sup>, Haihang You<sup>1,\*</sup>

<sup>1</sup>Institute of Computing Technology, Chinese Academy of Sciences

<sup>2</sup>University of Chinese Academy of Sciences, Beijing, China

{liuzeshi,wengzheng25e,tanruijie,tanguangming,youhaihang}@ict.ac.cn

**Abstract**—Modern computer architectures face challenges in balancing hardware overhead and performance. Binary computing, known for its compactness, requires hardware area that scales quadratically with precision, while unary computing, despite its simplicity, suffers from exponentially increasing computation time. This paper introduces Unary Positional System (UPS), a paradigm that combines spatial and temporal characteristics to address this trade-off. We develop UPS-based architectures to perform fundamental arithmetic operations, and apply it to GEMM and superconductor FFT processor. Experimental results show that UPS bridges the gap between binary and unary computing, offering a balanced solution with flexibility for further optimization.

## I. INTRODUCTION

In modern computing systems, hardware overhead is receiving increasing attention. Binary computing relies on parallel processing of spatial bits, which leads to a significant increase in hardware area as data width grows [1]. It becomes less suitable for scenarios where hardware area or power are constrained and latency requirements are relatively relaxed [2].

Alternatively, unary computing has re-emerged as it represents data as a temporary series of bit stream. With its extremely simple computational logic, unary computing has garnered significant research interest [3]–[6]. It is well-suited for hardware-constrained scenarios, such as bionic algorithms [7], [8] and emerging hardware [9]–[12]. However, unary computing relies solely on the temporal dimension for data representation, leading to a rapid increase in computational latency as data precision grows.

Binary and unary computing represent two extremes in the spectrum of spatial and temporal representation in digital computing. As shown in Fig. 1, binary relies entirely on the spatial dimension, enabling extremely short computation times but demanding substantial hardware resources. In contrast, unary computing maximizes hardware simplicity by relying exclusively on the temporal nature. However, this simplicity comes at the cost of significantly increased computational time. The central portion of Fig. 1 remains underutilized.

In many emerging scenarios, the limitations of binary and unary computing pose challenges to the advancement of digital hardware. For instance, in optical and superconductor hardware, binary computing is difficult to implement due to its high hardware logic requirements [9], [10]. On the other hand, unary computing, while hardware-efficient, incurs significant

computational latency for high-precision tasks. This latency undermines the benefits of emerging hardware.

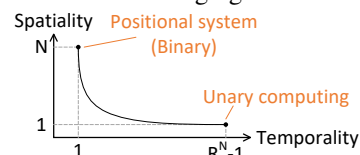


Fig. 1. Extreme characteristics in binary and unary computing.

In practice, resource-constrained and emerging scenarios rarely exhibit such polarized demands. With the aim of balancing spatial and temporal overheads to adapt to varying scenarios, we introduce the Unary Positional System (UPS), a paradigm that integrates both the spatial and temporal properties of number representation. Leveraging the UPS, we can flexibly adjust the area and latency of computing hardware, enabling more efficient utilization across various scenarios.

It is important to note that the objective of this paper is not to assert the superiority of binary, unary computing, or UPS. Instead, we want to demonstrate that the UPS method can be flexibly adjusted to navigate between the two extremes of binary and unary computing, by presenting a comprehensive design and detailed circuit structure, as well as qualitative and quantitative analyses.

First, we define the number representation in UPS based on radix and width, to integrate spatial and temporal properties. We develop unipolar (unsigned) and bipolar (signed) integer addition and multiplication within UPS, providing a foundation for extending the method to more complex designs. Second, we design the key hardware components required to implement UPS. Our design is entirely based on logic units, enabling the practical realization and application of UPS. Third, as practical applications, we implement two important structures – General Matrix Multiplication and superconductor FFT processor using UPS, to validate the feasibility of applying UPS to real-world scenarios. Finally, we conduct a thorough evaluation to compare the differences among UPS, binary, and unary computing.

With dual support for bit-stream and positional representations, UPS enables greater flexibility for constrained scenarios, making it well-suited for applications with specific requirements. The contributions of this work are as follows:

- We present a comprehensive design paradigm for the Unary Positional System (UPS), covering number representation, its complement, and methods for performing basic arithmetic operations. This framework enables the effective exploitation of the inherent flexibility of UPS.

\* Corresponding author.

- We introduce Counting Memory (CMem) as a key component for UPS hardware. Then we fully implement the fundamental arithmetic structures of UPS, including both unipolar and bipolar addition and multiplication.
- As application examples, we implement General Matrix Multiplication (GEMM) and superconductor FFT processors using UPS, demonstrating the feasibility of the UPS.
- The area of the UPS-based GEMM is 46% – 98% of that of the binary counterpart. Its performance surpasses unary computing by  $122\times$  to  $11,203\times$ . For superconductor FFT processors, the UPS-based design occupies only 39% and 60% of the area used in existing work, while achieving  $26\times$  to  $138\times$  higher performance than unary computing with equivalent accuracy. The results confirm the flexibility of UPS in balancing hardware area and performance.

## II. BACKGROUND AND MOTIVATION

### A. Binary Computing Characterization

Binary is fundamentally a positional system that represents numbers using multiple spatial positions, each assigned a value and a corresponding weight. By combining these values with their respective weights, a number can be expressed, as illustrated in Fig. 2(a). Due to its spatial nature, binary can be stored compactly and processed in parallel, leading to high computational speeds [1].

However, as the data width of a binary computation increases, the amount of logic required grows rapidly, especially when multiplying, where the growth is quadratic. Therefore, hardware area and power consumption also increase significantly, leading to substantial overhead. This characteristic of binary makes it challenging to apply in emerging hardware fields, where area and power are often limited.

### B. Pros and Cons of Unary Computing

Unary computing employs extremely simple logic to perform computations on serial bit streams in the time dimension [13]. Fig. 2(b) illustrates its number representation, where data is typically represented by a series of “1”s in a bit stream. In unary computing, multiplication can be performed using a single AND gate, while addition is carried out with a counter.

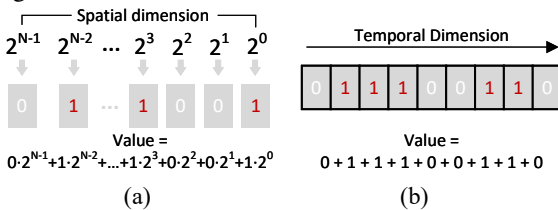


Fig. 2. (a) Binary representation and (b) unary bit stream.

Despite its logical simplicity, unary computing faces bottlenecks due to the encoding of unary bit streams. Unlike positional systems, unary computing requires a long bit stream that corresponds to the range of data being processed. This leads to bottlenecks in both computation time and storage [5].

Unary computing also includes time encoding [14] and stochastic computing [15], [16]. Additionally, stochastic and deterministic methods of unary computing are well-established [17], and as this paper focuses on achieving flexibility between unary and positional computing, we use deterministic methods as a representative method [18].

### C. Time-Space Balance Challenges and Opportunities – Potential for a Hybrid Approach

Binary represents numbers with bit weights of  $2^{n-1}$  and a time length of 1, while unary computing uses a constant bit weight of 1 but with a longer time length. These two approaches represent extremes in spatial and temporal utilization, respectively. However, existing research rarely achieves a practical balance between these two characteristics.

To combine spatial and temporal properties of both binary and unary, there are three key challenges: (1) Balancing the temporal and spatial paradigms and considering both the space and the time where each “1” occurs. (2) The system should be closed, meaning it could input and output directly in UPS encoding, avoiding conversions. (3) The method must be supported by appropriate application scenarios.

Alongside the challenges are three opportunities: (1) UPS is adaptable to temporal and spatial constraints, offering significant flexibility. (2) UPS provides a more natural number representation, supporting different radices. (3) In resource-constrained and emerging scenarios, for example, the Internet of Things [19] and superconductor computing [10], UPS provides performance potential while maintaining low hardware area requirements.

## III. UNARY POSITIONAL SYSTEM (UPS)

To enable a flexible representation of the spatio-temporal nature, we introduce Radix and Width, denoted as  $R$  and  $N$ .  $R$  (radix) represents the base of the UPS number. A positional value that reaches  $R$  will reset to zero and increment the next position by 1.  $N$  (width) represents the number of parallel positions in a UPS number. E.g., the representation  $3212_{UP(4,4)}$  denotes the UPS number 3212 with  $R = 4$  and  $N = 4$ .

### A. UPS Number

1) *Unsigned integer*: In UPS, there are  $N$  spatial positions within the same temporal dimension, each with a distinct weight,  $R^n$ , where  $n = 0, 1, \dots, N - 1$ . Each UPS position is associated with a bit stream of length  $R$  in the temporal dimension, which represents the value at that position by the number of “1”s it contains. The last bit of each stream must be 0, due to the propagation delay inherent in unary computing.

Fig. 3 illustrates an example of a UPS number with  $N = 4$  and  $R = 4$ . The corresponding decimal value is  $3212_{UP(4,4)} = 3 \cdot 4^3 + 2 \cdot 4^2 + 1 \cdot 4^1 + 2 \cdot 4^0 = 230$ .

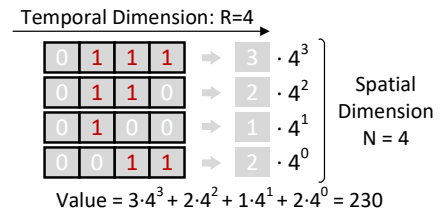


Fig. 3. Example of a UPS number with  $N = 4$ ,  $R = 4$ .

2) *Signed integer*: The signed numbers in UPS use the highest position to represent the sign. Negative numbers in UPS are then represented by the complement. Fig. 4 illustrates the procedure for complementing a UPS number, where  $212_{UP(4,4)}$  is complemented to  $-212_{UP(4,4)} = 3122_{UP(4,4)}$ . For a bit stream in the temporal dimension, its inversion can be considered as

simply inverting the first  $R - 1$  digits. The results in Fig. 4 can be easily verified with  $Complement(0212_{UP(4,4)}) = 4^4 - 0212_{UP(4,4)} = 4^4 - 38 = 218 = 3122_{UP(4,4)}$ .

With the complement, the addition of signed UPS numbers can be directly converted to the addition of its complement. Besides, for bipolar multiplication, conclusions similar to binary can be drawn, as shown in Eq. 1 (where  $C()$  denotes the complement and  $Sgn()$  represents the sign function). This allows arithmetic to be performed directly using complements.

$$\begin{aligned} & \text{for } C(Y_{UP(R,N)}) = y_{N-1}y_{N-2}\dots y_1y_0, \text{ there is} \\ & C(X_{UP(R,N)} \cdot Y_{UP(R,N)}) = C(X_{UP(R,N)}) \cdot \\ & [-Sgn(y_{N-1}) \cdot R^{N-1} + y_{N-2} \cdot R^{N-2} + \dots + y_1 \cdot R^1 + y_0 \cdot R^0] \end{aligned} \quad (1)$$

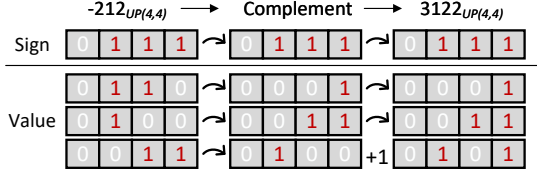


Fig. 4. Complement procedure in UPS.

## B. UPS Computation

1) *Addition*: In the temporal dimension of UPS, it performs addition by counting to obtain the sum across each spatial position. Due to the radix, each spatial position in UPS must reset and carry over when it reaches the radix.

Fig. 5 presents an example of UPS addition. In this example,  $3102_{UP(4,4)}$  is added to  $0222_{UP(4,4)}$ , where the first position produces a carry, resulting in the final sum of  $3330_{UP(4,4)}$ .

Leveraging the advantages of complement representation, both unipolar and bipolar UPS addition and subtraction can be implemented by combining addition with sign determination.

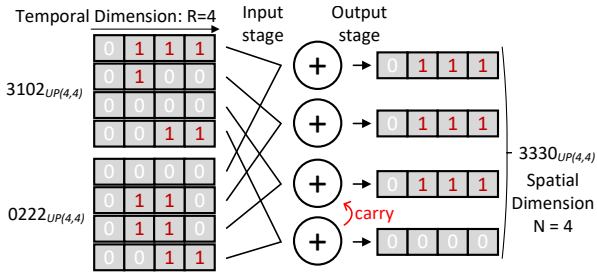


Fig. 5. Example of UPS addition.

2) *Multiplication*: For single-bit case, as shown in Fig. 6(a), each bit of the two multipliers must be ANDed once. So we extend the two multipliers to align with each pair of bits.

Then, unipolar UPS multiplication is built on the single-bit case. Fig. 6(b) shows an example with  $3102_{UP(4,4)} \times 0222_{UP(4,4)}$ . The numbers in the vertical columns of the figure indicate the frequency of the digit '1' appearing in the bit stream. Each position of the multiplier is multiplied by the entire multiplicand, resulting in  $4 = N$  partial sums.

By introducing Eq. 1, we can perform bipolar UPS multiplication. Fig. 6(c) shows  $3102_{UP(4,4)} \times 0222_{UP(4,4)}$ , when a negative number is multiplied by a positive number, the higher positions of the partial sum must be filled with sign digits. Fig. 6(d) illustrates  $0222_{UP(4,4)} \times 3102_{UP(4,4)}$ , when a positive number is multiplied by a negative number, the final partial

sum must be subtracted. Additionally, when multiplying two negative numbers in UPS, both of these steps are required.

## C. Discussion

UPS can employ various techniques to reduce complexity. The representation and computation of floating-point numbers can be naturally achieved by combining UPS's integer with binary floating-point representation [20]. Tricks like Booth's algorithm [21] and negative weights can also be implemented in UPS by mimicking binary. As this paper focuses on the fundamental representation and computation of UPS, these techniques will not be further discussed here. Finally, although the examples all use  $R = 4$  and  $N = 4$ , in practice, both  $R$  and  $N$  can be any arbitrary positive integers.

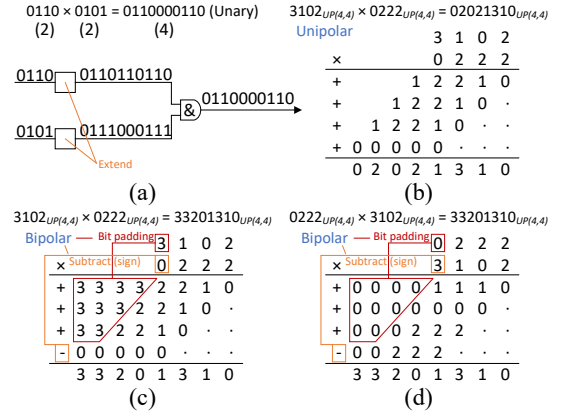


Fig. 6. Multiplication examples in UPS. (a) Single-bit multiplication. (b) Example of unipolar UPS multiplication. (c) and (d) Bipolar UPS multiplication.

## IV. UPS ARCHITECTURE

### A. Counting Memory (CMem)

We introduce Counting Memory (CMem) as a key component of UPS. The function of CMem is to count and storage the input bit stream of "1"s and generate a carry/overflow signal when the radix ( $R$ ) is reached. Fig. 7 presents the symbols for two types of radix- $R$  CMem, with single and dual inputs, respectively. In particular, the dual-port CMem can meet the requirement of dual inputs for arithmetic operations.

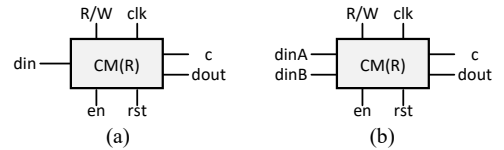


Fig. 7. Symbol of radix- $R$  CMem with (a) single input and (b) dual inputs.

It is important to note that we recognize level-based binary as one of the best counter logic for current semiconductor processes. Moreover, the binary-based implementation of CMem can act as a bridge between the binary system and UPS, enabling direct conversion between them. CMem can also be designed to fully leverage hardware characteristics for efficient counting, holding the potential to be implemented using emerging non-binary principles. For example, in optical or superconductor computing, counting can be performed using accumulated optical photons or single flux quantum, enabling CMem differs from binary logic [9]–[12]. Furthermore, CMem

offers the potential for more natural carry-over and fan-in in emerging domains such as neuromorphic computing [7], [8].

### B. UPS Adder

Based on dual-input CMem, the basic UPS adder could be derived directly. Fig. 8 provides an example of a UPS adder with  $R = 4$ ,  $N = 4$ . The UPS adder should first read the inputs and then process the carry before it can output the result (via the R/W signal).

The key to UPS addition is the handling of the carry signal, which requires an additional  $N$  (width) time step. Furthermore, based on the UPS complement, both unipolar and bipolar UPS addition can use the same adder.

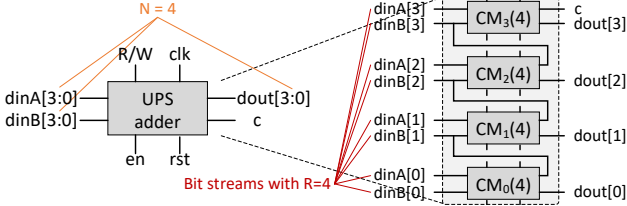


Fig. 8. Symbol and structure of the UPS adder with  $N = 4$ ,  $R = 4$ .

### C. UPS Multiplier

The multiplier best exemplifies temporal-spatial nature of UPS. In spatial dimension, each position pair from the two multipliers is multiplied in parallel, leading to faster computation. In temporal dimension, UPS represents values through a time-serialized bit stream, enabling a larger range for each position and requiring less logic. Here we present the most basic UPS array multiplier, with the goal of verifying the feasibility of it.

1) *UPS Unipolar Multiplier*: Fig. 9(a) shows the symbol for the UPS multiplier, while Fig. 9(b) presents an example of a unipolar UPS multiplier with  $R = 4$  and  $N = 4$ .

The UPS multiplier performs sequential read/write (via CMem) on the partial products of each row to obtain the final result. The CMem in the UPS multiplier functions as an accumulator, meaning it does not require extra carry time. The total time for a UPS multiplication is  $\text{multiplication\_time} + N \cdot \text{addition\_time}$ .

2) *UPS Bipolar Multiplier*: Fig. 9(c) illustrates an example of a UPS array multiplier using complements, with  $R = 4$  and  $N = 4$ . Following Eq. 1, the bipolar multiplier performs sign position padding and complements the last partial product, requiring slightly more logic than the unipolar multiplier. However, the bipolar multiplication completes processing in the same time as the unipolar multiplication.

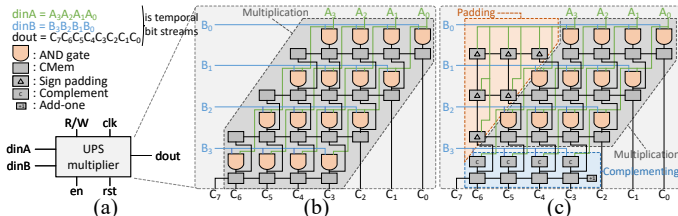


Fig. 9. UPS array multiplier. (a) Symbol. (b) Unipolar. (c) Bipolar.

### D. Discussion

Since UPS has a certain spatiality, many techniques from binary systems can be adapted.

1) *Efficiency of different  $R$ ,  $N$* : The parameters  $R$  and  $N$  in the UPS reflect the temporal-spatial efficiency of its representation. When  $R$  is fixed and  $N$  is increased, the spatial efficiency improves. Conversely, when  $N$  is fixed and  $R$  is increased, the temporal efficiency increases. This flexibility allows UPS to adapt to different scenarios.

2) *More parallel structures*: In traditional binary architectures, parallel structures such as carry-look-ahead and multi-input adders are leveraged to enhance efficiency. These parallel techniques can also be implemented in UPS to improve performance due to its spatial nature.

3) *Pipeline design*: In unary computing, pipelining is challenging to implement due to the rapid increase in bit stream length. In contrast, UPS allows pipeline to be applied in a coarse-grained manner. By treating the temporal portion ( $R$ ) as a unified whole, we introduce a pipeline by blocks of time ( $N$ ) to UPS. Its efficiency will be evaluated in Section VI.

## V. UPS APPLICATION

### A. UPS-based GEMM

General Matrix Multiplication (GEMM) is a critical operation, which accounts for over 90% of the computation in Deep Neural Networks [22]. The core of GEMM is multiply-accumulate (MAC). By merging these two operations, greater efficiency can be achieved, particularly in UPS, where the direct accumulation capability reduces the required logic.

We first explore the feasibility and flexibility of UPS by implementing a UPS-based GEMM. Fig. 10 illustrates the structure, where matrices  $A$ ,  $B$ , and  $C$  have dimensions of  $m \times k$ ,  $k \times n$ , and  $m \times n$ , respectively. The core of this GEMM structure is a processing element composed of multiple UPS-based multiplication and accumulation, as shown in Fig. 10(b).

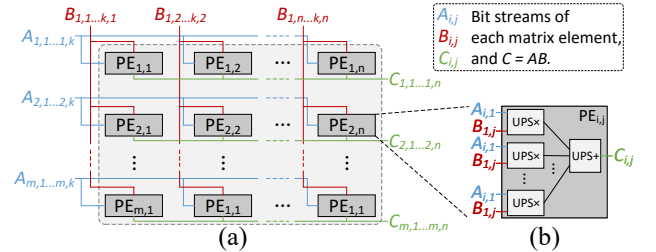


Fig. 10. UPS-based GEMM structure. (a) Overview. (b) Processing element.

### B. UPS-based superconductor FFT Processor

The Fast Fourier Transform (FFT) is a widely used algorithm that demands substantial computation [23]. We implement FFT using superconductor computing based on UPS, highlighting its flexibility in hardware-constrained scenarios.

1) *Features of superconductor Computing*: superconductor computing is a promising emerging technologies in the post-Moore era, which has ultra-low switching delay ( $\sim 10^{-12}s$ ) and switching energy ( $\sim 10^{-19}J$ ) [24], [25], enabling highly efficient hardware for domain-specific computing [26].

Relying on Josephson junctions (JJs) for information storage and processing [27], superconductor Computing faces certain limitations, the most critical is the integration density of JJs. State-of-the-art fabrication processes can produce circuits comprising 10,000–20,000 JJs [28]–[30]. However, each logic cell generally require 5–10 JJs, as shown in table I.

2) *superconductor FFT Processor*: Most existing superconductor FFT processors adopt binary design methodologies. We employ a typical FFT butterfly structure integrated with UPS arithmetic units, aiming to balance hardware overhead and performance by leveraging the advantages of UPS. Fig. 11 illustrates our design, where input buffers generate UPS input, data converters create complements based on signs, the multipliers are UPS multipliers, and the adders are UPS adders. The only difference lies in replacing traditional arithmetic units with their UPS-based counterparts. This highlights the practicality and adaptability of the UPS methodology, demonstrating its ease of integration into real-world applications.

TABLE I  
JJS OVERHEAD FOR COMMON LOGIC SUPERCONDUCTOR CELLS [31].

Cell name	# of JJs	Cell name	# of JJs
JTL (Josephson transmission line)	2	AND	10
SPL (splitter)	3	NOT	7
CB (confluence buffer)	2	XOR	12
NDRO (non-destructive read-out)	12	OR	11
DFE	4	TFF	6

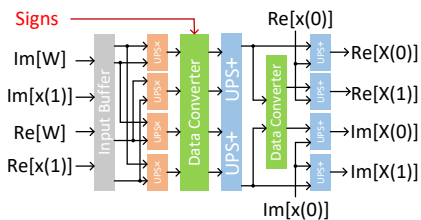


Fig. 11. The UPS-based superconductor butterfly operator.

## VI. EVALUATION

### A. Experimental Setup

We evaluate UPS, binary, and unary computing using the same structure. For semiconductor processes, we use Verilog RTL to synthesize the area and power via Synopsys Design Compiler, targeting the FreePDK 45nm technology node [32] and a 400 MHz frequency. For superconductor processes, we simulate designs with AIST Advanced Process (ADP2) [33].

Since widths of UPS and unary computing are more flexible than binary, our evaluation is based on the width of binary. We use the suffixes “-u” and “-b” to indicate unipolar and bipolar cases, respectively. Furthermore, We evaluate every  $(R, N)$  combinations of UPS to demonstrate its flexibility.

### B. Comparison between Additions

As Fig. 12 shows, the area and power of binary adders is lower than that of UPS and unary computing for the same precision. This is because counting and storing the bit stream needs additional overhead, especially CMem, which integrates counting, storage, and conversion functions.

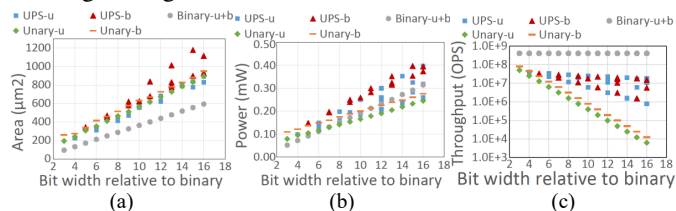


Fig. 12. Comparison of adders based on relative binary bit width. (a) Area. (b) Power. (c) Throughput.

However, UPS offers a significant advantage: it avoids the rapid drop in throughput that unary computing suffers as

data precision increases. The benefits of CMem are worth its additional overhead, because addition logic typically constitutes a small fraction of the overall cost in large-scale computation. As shown in Fig. 12(c), at the same width, the performance of UPS adder ranges from  $1.42\times$ – $2979\times$  higher than that of unary computing.

### C. Comparison between Multiplication

Fig. 13 presents a comparison of the multipliers. As shown, although binary offers high performance, its area and power consumption scale quadratically with data width. Unary computing incurs lower area and power overhead but suffers from a sharp decline in performance as data width increases.

UPS effectively bridges the gap between binary and unary computing in terms of both overhead and performance. In Fig. 13(a)(b), the area and power of UPS-based multipliers range from 15% to 92% and 6.7% to 56% of those of their binary counterparts, respectively. In Fig. 13(c), the throughput of UPS is at least  $2.8\times$  higher than that of unary computing, and the gap widens rapidly.

The spatial nature of UPS enables the use of pipelines, and Fig. 13(d) illustrates the improvement achieved through our block pipeline design.

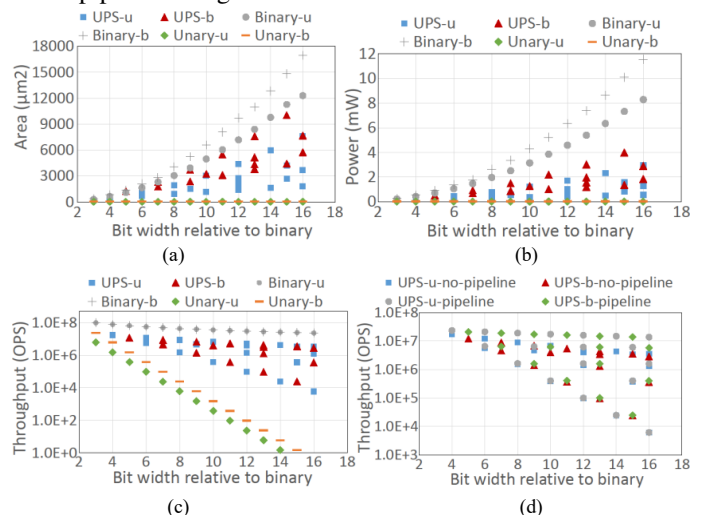


Fig. 13. Comparison of multipliers based on relative binary bit width. (a) Area. (b) Power. (c) Throughput. And (d) the effect of pipeline on UPS.

### D. Comparison between GEMM

Our evaluation uses matrices with sizes  $m = n = k = 4, 8, 16$ , and compares them against binary implementations with 7, 8, and 9 bits. Fig. 14 shows the result. It can be observed that, since the overhead of multipliers is higher than that of adders, the overhead and performance of GEMM implementations follow a similar trend to those of the multipliers.

As shown in Fig. 14(a)(b), the UPS-GEMM exhibits flexible hardware characteristics, requiring only 46% – 98% of the area and 24% – 62% of the power compared to the binary. In terms of throughput, UPS significantly outperforms unary computing and approaches the performance level of binary, as illustrated in Fig. 14(c). Specifically, the throughput of the UPS-based GEMM is  $122\times$  –  $11,203\times$  higher than that of unary computing. Also, we assess the impact of the block pipeline design on GEMM, with the results given in Fig. 14(d).

### E. Comparison between superconductor FFT Processors

We evaluate superconductor FFT processors as a representative example of applying UPS in emerging constrained scenarios. Two UPS configurations are used: one with  $R = 8, N = 2$  plus a sign position, and another with  $R = 4, N = 2$ , comparing against the signed 7-bit and unsigned 4-bit designs presented in [34] and [29], respectively. The total JJ count is employed as an indicator of a design’s overall cost [31], [33]. We assess the overhead and performance of a unary computing implementation based on UPS at  $N = 1$ , serving as an extreme example of temporal-dimension-dominant design.

From Table II, it is evident that the number of JJs used by UPS amounts to only 60% and 39% of those required by two binary implementations, respectively.

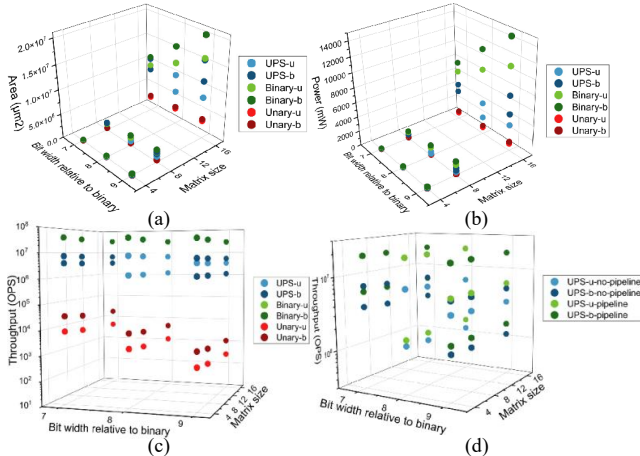


Fig. 14. Comparison of GEMM based on relative binary bit width and matrix size. (a) Area. (b) Power. (c) Throughput. And (d) pipeline’s effect on UPS.

TABLE II  
COMPARISON OF BINARY, UPS, AND UNARY COMPUTING IMPLEMENTATIONS OF SUPERCONDUCTOR FFT PROCESSORS.

Method	Binary [29] (bit-serial)	UPS	Unary
Width	7 (signed)	<b>N=2, R=8, plus 1-position sign</b>	1
JJ w/ wiring	17455	<b>10488</b>	5796
cycles@4-point FFT	113	<b>263</b>	36368
cycles@8-point FFT	335	<b>735</b>	100896
Method	Binary [34] (2-bit bit-slice)	UPS	Unary
Width	4 (unsigned)	<b>N=2, R=2</b>	1
JJ w/ wiring	20523	<b>8076</b>	4767
cycles@4-point FFT	43	<b>94</b>	2448
cycles@8-point FFT	59	<b>238</b>	6304

While the binary implementations are reaching the integration limits of JJs of superconductor circuits (10,000–20,000 JJs). The spatial efficiency of UPS enables the full implementation of the butterfly operator within current superconductor process constraints. Compared to unary computing, UPS is substantially more practical, achieving  $26\times$ – $138\times$  the throughput of unary computing. These results highlight the advantage of UPS in constrained scenarios

### F. UPS with Different ( $R, N$ )

As observed in the experiments, the temporal-spatial nature of UPS provides impressive flexibility. From Fig. 13 and 14, for a fixed precision, decreasing  $R$  and increasing  $N$  reduces the temporal aspect of UPS while enhancing its spatial aspect. This trade-off enables higher performance at the cost of increased area and power consumption, with its extreme case being full

positional representation, typically binary. Conversely, increasing  $R$  and decreasing  $N$  can reduce area and power overhead, but this comes at the expense of latency. The extreme case of this trade-off is unary computing.

The advantage of UPS lies in its ability to shift only part of the resolution to the temporal dimension, enabling spatial savings while maintaining a manageable increase in latency. Notably, as data width increases, the number of viable ( $N, R$ ) combinations also grows, providing UPS with greater flexibility for emerging hardware and resource-constrained scenarios.

## VII. RELATED WORK

The field of computing is undergoing a re-focus on unary computing due to the rise of hardware-limited and emerging scenarios [5], [9], [10], yet unary computing is hindered by high latency and low accuracy. Although previous research has explored methods incorporating stochastic, unary, and positional approaches, none have successfully harnessed the flexibility of these techniques for practical applications.

For example, [35] introduced the idea of unary positional computing but only provided conceptual discussion and abstract computational designs. They did not explore a generalized approach or conduct in-depth experimental analyses, which limits the practical adoption of the method in real-world scenarios.

Furthermore, [36] applied unary computing to represent floating-point numbers. But their approach relies on stochastic computing for the temporal dimension, making it suitable only for handling fractions. Moreover, the exclusive use of radix-2 significantly limited the flexibility of their representation.

Likewise, [37] also adopted radix-2, restricting their method to a binary system extension. To simplify implementation, they employed scaled addition as the adder, which further reduced the generality and practical utility of their approach.

Existing research has also explored methods for achieving temporal-spatial balance through Hybrid Unary-Binary (HUB) approaches, such as [5], [38], [39]. However, HUB methods require compatibility with both unary and binary computing, which contrasts with the UPS of providing a unified structure. Therefore, we do not consider HUB methods in this paper.

## VIII. CONCLUSION

This paper introduces UPS. By combining temporal and spatial nature, UPS avoids the quadratic area growth seen in binary computing and the rapid latency growth typical of unary computing. We design UPS-based GEMM and superconductor FFT processors as application examples to verify its feasibility. The results of applying UPS to GEMM demonstrate that its area overhead is 46%–98% of that of binary designs, and its power consumption is 24%–62%. Besides, UPS has  $122\times$ – $11,203\times$  performance over unary computing. By offering a flexible balance between hardware area and performance, UPS is well-suited for resource-constrained scenarios and applications with specialized requirements.

## ACKNOWLEDGMENT

This work was supported by the Postdoctoral Fellowship Program (Grade B) of China Postdoctoral Science Foundation under grant number GZB20240760.

## REFERENCES

- [1] D. E. Knuth, *The art of computer programming*. Pearson Education, 1997, vol. 3.
- [2] D. Wu, Y. Chen, Q. Zhang, Y.-I. Ueng, and X. Zeng, "Strategies for reducing decoding cycles in stochastic ldpc decoders," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 63, no. 9, pp. 873–877, 2016.
- [3] A. Ren, Z. Li, C. Ding, Q. Qiu, Y. Wang, J. Li, X. Qian, and B. Yuan, "Sc-dcnn: Highly-scalable deep convolutional neural network using stochastic computing," in *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '17. Association for Computing Machinery, 2017, p. 405–418.
- [4] D. Wu, J. Li, R. Yin, H. Hsiao, Y. Kim, and J. S. Miguel, "Ugemm: Unary computing architecture for gemm applications," in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, 2020, pp. 377–390.
- [5] H. Guo, Y. Zhao, Z. Li, Y. Hao, C. Liu, X. Song, X. Li, Z. Du, R. Zhang, Q. Guo, T. Chen, and Z. Xu, "Cambricon-u: A systolic random increment memory architecture for unary computing," in *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 424–437.
- [6] D. Wu and J. S. Miguel, "usystolic: Byte-crawling unary systolic array," in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2022, pp. 12–24.
- [7] D. Wu, J. Li, Z. Pan, Y. Kim, and J. S. Miguel, "ubrain: a unary brain computer interface," in *Proceedings of the 49th Annual International Symposium on Computer Architecture*, ser. ISCA '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 468–481.
- [8] X. Jia, H. Gu, Y. Liu, J. Yang, X. Wang, W. Pan, Y. Zhang, S. Cotofana, and W. Zhao, "An energy-efficient bayesian neural network implementation using stochastic computing method," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 9, pp. 12 913–12 923, 2024.
- [9] S. S. Vatsavai, V. S. P. Karempudi, I. Thakkar, A. Salehi, and T. Hastings, "Sconna: A stochastic computing based optical accelerator for ultra-fast, energy-efficient inference of integer-quantized cnns," in *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2023, pp. 546–556.
- [10] P. Gonzalez-Guerrero, M. G. Bautista, D. Lyles, and G. Micheliogiannakis, "Temporal and sfq pulse-streams encoding for area-efficient superconducting accelerators," in *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 963–976.
- [11] R. Cai, A. Ren, O. Chen, N. Liu, C. Ding, X. Qian, J. Han, W. Luo, N. Yoshikawa, and Y. Wang, "A stochastic-computing based deep learning framework using adiabatic quantum-flux-parametron superconducting technology," in *Proceedings of the 46th International Symposium on Computer Architecture*, ser. ISCA '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 567–578.
- [12] H. El-Derhalli, L. Constans, S. Le Beux, A. De Rossi, F. Raineri, and S. Tahar, "Towards all-optical stochastic computing using photonic crystal nanocavities," *J. Emerg. Technol. Comput. Syst.*, vol. 18, no. 1, Oct. 2021.
- [13] W. J. Poppelbaum, A. Dollas, J. Glickman, and C. O'Toole, "Unary processing," in *Advances in computers*. Elsevier, 1987, vol. 26, pp. 47–92.
- [14] A. Madhavan, T. Sherwood, and D. Strukov, "Race logic: A hardware acceleration for dynamic programming algorithms," in *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, 2014, pp. 517–528.
- [15] B. R. Gaines, *Stochastic Computing Systems*. Boston, MA: Springer US, 1969, pp. 37–172.
- [16] A. Alaghi and J. P. Hayes, "Survey of stochastic computing," *ACM Trans. Embed. Comput. Syst.*, vol. 12, no. 2s, May 2013.
- [17] P. Mars and W. Poppelbaum, *Stochastic and Deterministic Averaging Processors*, ser. Computing Series. P. Peregrinus, 1981.
- [18] D. Jenson and M. Riedel, "A deterministic approach to stochastic computation," in *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2016, pp. 1–8.
- [19] M. Merenda, C. Porcaro, and D. Iero, "Edge machine learning for ai-enabled iot devices: A review," *Sensors*, vol. 20, no. 9, p. 2533, 2020.
- [20] W. Kahan, "Ieee standard 754 for binary floating-point arithmetic," *Lecture Notes on the Status of IEEE*, vol. 754, no. 94720-1776, p. 11, 1996.
- [21] A. D. Booth, "A signed binary multiplication technique," *The Quarterly Journal of Mechanics and Applied Mathematics*, vol. 4, no. 2, pp. 236–240, 1951.
- [22] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM International Conference on Multimedia*, ser. MM '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 675–678.
- [23] P. Duhamel and M. Vetterli, "Fast fourier transforms: A tutorial review and a state of the art," *Signal Processing*, vol. 19, no. 4, pp. 259–299, 1990.
- [24] T. V. Duzer and C. W. Turner, *Principles of Superconductive Devices and Circuits, (Second Ed.)*. USA: Prentice Hall PTR, 1998.
- [25] D. S. Holmes, A. L. Ripple, and M. A. Manheimer, "Energy-Efficient Superconducting Computing—Power Budgets and Requirements," *IEEE Transactions on Applied Superconductivity*, vol. 23, no. 3, pp. 1 701 610–1 701 610, 2013.
- [26] Z. Liu, S. Chen, P. Qu, H. Liu, M. Niu, L. Ying, J. Ren, G. Tang, and H. You, "SUSHI: Ultra-High-Speed and Ultra-Low-Power Neuro-morphic Chip Using Superconducting Single-Flux-Quantum Circuits," in *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 614–627.
- [27] K. Likharev and V. Semenov, "Rsfq logic/memory family: a new josephson-junction technology for sub-terahertz-clock-frequency digital systems," *IEEE Transactions on Applied Superconductivity*, vol. 1, no. 1, pp. 3–28, 1991.
- [28] Z. Liu, S. Chen, P. Qu, G. Tang, and H. You, "Toward superconducting neuromorphic computing using single-flux-quantum circuits," *IEEE Transactions on Applied Superconductivity*, vol. 35, no. 3, pp. 1–14, 2025.
- [29] F. Ke, O. Chen, Y. Wang, and N. Yoshikawa, "Demonstration of a 47.8 ghz high-speed fft processor using single-flux-quantum technology," *IEEE Transactions on Applied Superconductivity*, vol. 31, no. 5, pp. 1–5, 2021.
- [30] L. Ying, X. Zhang, M. Niu, J. Ren, W. Peng, M. Maezawa, and Z. Wang, "Development of multi-layer fabrication process for sfq large scale integrated digital circuits," *IEEE Transactions on Applied Superconductivity*, vol. 31, no. 5, pp. 1–4, 2021.
- [31] X. Gao, Q. Qiao, M. Wang, M. Niu, H. Liu, M. Maezawa, J. Ren, and Z. Wang, "Design and verification of SFQ cell library for superconducting LSI digital circuits," *IEEE Transactions on Applied Superconductivity*, vol. 31, no. 5, pp. 1–5, 2021.
- [32] NCSU, "Freepdk45," <https://eda.ncsu.edu/freepdk/freepdk45/> Accessed November 01, 2024.
- [33] S. NAGASAWA, K. HINODE, T. SATOH, M. HIDAKA, H. AKAIKE, A. FUJIMAKI, N. YOSHIKAWA, K. TAKAGI, and N. TAKAGI, "Nb 9-layer fabrication process for superconducting large-scale sfq circuits and its process evaluation," *IEICE Transactions on Electronics*, vol. E97.C, no. 3, pp. 132–140, 2014.
- [34] G.-M. Tang, P.-Y. Qu, X.-Y. Zheng, J.-H. Yang, X.-C. Ye, D.-R. Fan, and N.-H. Sun, "Bit-slice butterfly processing units for 64-point rsfq fft processors," *IEEE Transactions on Applied Superconductivity*, vol. 30, no. 1, pp. 1–6, 2020.
- [35] M. van der Hagen and M. Riedel, "Unary positional computing," in *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2017, pp. 1335–1339.
- [36] Y. Zhu, P. Suo, and K. Bazargan, "Binary stochastic implementation of digital logic," in *Proceedings of the 2014 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 171–180.
- [37] M. H. Najafi, S. R. Faraji, B. Li, D. J. Lilja, and K. Bazargan, "Accelerating deterministic bit-stream computing with resolution splitting," in *20th International Symposium on Quality Electronic Design (ISQED)*, 2019, pp. 157–162.
- [38] Z. Chen, Y. Ma, and Z. Wang, "Hybrid stochastic-binary computing for low-latency and high-precision inference of cnns," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 7, pp. 2707–2720, 2022.
- [39] S. R. Faraji and K. Bazargan, "Hybrid binary-unary hardware accelerator," *IEEE Transactions on Computers*, vol. 69, no. 9, pp. 1308–1319, 2020.