ProU-SC: An Area-Efficient Random Number Source via Progressive Spatial Uniformity for Stochastic Computing

Jiangyuan Wang[1], Chang Meng[2], Jinhua Cui[1], Kuncai Zhong[1,*], Weikang Qian[3]

[1]College of Semiconductors, Hunan University, Changsha, China; [2]Dept. of Mathematics and Computer Science, Eindhoven University of Technology, The Netherlands; [3]Global College, Shanghai Jiao Tong University, Shanghai, China

Emails: jiangyuanwang@hnu.edu.cn, kczhong@hnu.edu.cn; *Corresponding author

Abstract

Stochastic computing (SC) requires long bit streams for high accuracy, causing significant latency. Progressive precision mitigates this issue via early truncation. However, existing random number sources (RNSs) struggle to balance hardware cost and early-estimate accuracy. Low-cost RNSs generally lack short-length spatial uniformity, while low-discrepancy sequence-based RNSs incur prohibitive hardware overhead. To address this, we formalize the principle of progressive spatial uniformity, ensuring uniform multi-dimensional sample point distributions across intermediate truncation levels. Guided by this principle, we propose **ProU-SC**, an area-efficient RNS architecture utilizing a coordinated chain of non-linear feedback shift registers and a hardwired bit-interleaving scheme to physically guarantee hierarchical nesting uniformity. Furthermore, a decoupled heuristic search algorithm is formulated to efficiently navigate the massive configuration space for accuracy optimization. Experimental results demonstrate that compared to state-of-the-art low-discrepancy sequence-based designs, **ProU-SC** reduces area and power by 93.8% and 92.4%, respectively. Crucially, it delivers the lowest computation error at ultra-short bit streams (e.g., 16 and 32 bits) among all evaluated baselines.

Keywords

Stochastic computing, Random number Source, Progressive spatial uniformity, Progressive precision.

1 Introduction

Stochastic computing (SC) [10] has emerged as a compelling low-power paradigm for resource-constrained applications like digital filtering [8, 13, 23, 28] and edge AI [9, 16, 25, 27]. Instead of using conventional binary radix, SC encodes numerical values as the ratio of 1s in stochastic bit streams (SBSs) [1]. For example, the stream 01101110 encodes the value $\frac{5}{8}$. This unique representation allows complex arithmetic to be implemented with highly compact logic, such as a single AND gate for multiplication [1]. However, SC faces a fundamental trade-off between latency and accuracy. To match the precision of an n-bit binary system, conventional SC requires a stream length of 2^n bits. This exponential latency growth poses a severe bottleneck for real-time applications.

To mitigate this, progressive precision (PP) has been widely explored. As an inherent advantage of SC, PP allows a circuit to truncate the SBS early, outputting a lower-resolution estimate from a short prefix. For instance, evaluating only the first 4 bits (0110) of the aforementioned 8-bit stream (01101110) yields an early estimate of $\frac{2}{4}$ (0.5) for the target value $\frac{5}{8}$ (0.625). Recent runtime early termination strategies leverage this property to dynamically drop unused precision bits, effectively reducing both computation latency and energy consumption [11, 12, 26].

However, the accuracy of these early estimates is fundamentally governed by the random number source (RNS). In an m-input SC circuit, the m random numbers generated per cycle form a single sample point in an m-dimensional space. Operating inherently as a Monte Carlo integration process [3], SC achieves high accuracy only when these sample points distribute evenly across the entire space. This crucial mathematical property is termed *spatial uniformity* (or low discrepancy) [3]. Under PP, early truncation drastically shrinks the available sample size, rendering the system highly vulnerable to spatial clustering. As illustrated in Fig. 1, while ideal spatial uniformity guarantees a perfectly balanced sample distribution from early truncation boundaries (e.g., precision level $k = 2$, producing $2^2 = 4$ points) all the way to full precision ($k = 4$, producing 16 points), severe clustering at short sequences introduces massive sampling bias. This bias will permanently degrade both the early-estimate and the final accuracy.

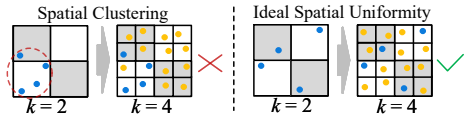


Figure 1: Spatial distribution across PP levels ($k = 2, 4$): severe clustering (left) vs. ideal spatial uniformity (right).

Consequently, to make PP viable, spatial uniformity must be strictly maintained not just at the full stream length, but across intermediate truncation boundaries. Unfortunately, existing RNS designs fail to achieve a favorable trade-off between hardware efficiency and this multi-scale uniformity [3, 4, 6, 15, 17, 19, 21]. Low-cost RNSs, such as linear feedback shift registers (LFSRs), lack short-length spatial uniformity and incur large errors for the short prefixes used in PP [4, 19, 21]. Conversely, low-discrepancy sequence-based RNSs [3, 15, 17], such as the Sobol sequence generator [15], provide high uniformity at power-of-two truncation points but rely on complex generation logic. This reliance results in prohibitive hardware overhead. For instance, a Sobol sequence

This work was supported by the National Natural Science Foundation of China (Grant No. 62402169), the Natural Science Foundation of Hunan Province (Grant No. 2026JJ40056), and the Research Foundation of Education Bureau of Hunan Province, China (No. 25B0049). The authors used Gemini exclusively for language polishing. The authors reviewed all changes and bear full responsibility for the content.

ICCAD '26, San Jose, CA, USA

ACM ISBN 979-8-4007-2873-0/2026/11
https://doi.org/10.1145/3831252.3833956

generator can occupy over 90% of the total area in a simple SC multiplier. Recent hardware-efficient architectures [29] achieve spatial uniformity at the full stream length (2^n bits) with low cost. However, they fail to guarantee this property at intermediate truncation points, where early truncation disrupts their spatial structure and severely degrades accuracy. Therefore, designing an RNS that guarantees progressive spatial uniformity while preserving the extreme hardware efficiency of SC remains a critical open challenge.

To address these limitations, we propose **ProU-SC**, an area-efficient RNS architecture that deterministically maintains progressive spatial uniformity. Instead of a monolithic generator, our core intuition is to decompose the sequence generation into a hierarchical chain of smaller sub-engines. Concurrently, we re-architect the spatial routing to dynamically map the outputs of these locally exhausted sub-engines into hierarchical spatial nesting. This ensures that uniformity is strictly preserved at designated intermediate boundaries without requiring complex control logic.

The main contributions of this work are summarized as follows:

- We formalize the principle of *progressive spatial uniformity*, establishing a theoretical foundation for uniform sample distribution at intermediate truncation points (Section 3).
- We propose the **ProU-SC** architecture, which utilizes a coordinated non-linear feedback shift register (NLFSR) chain and a hardwired bit-interleaving scheme to physically guarantee hierarchical nesting uniformity with high hardware efficiency (Section 4).
- We formulate a decoupled heuristic search algorithm to efficiently navigate **ProU-SC**'s massive configuration space, independently optimizing for both progressive and final accuracy (Section 5).

Experimental results demonstrate that **ProU-SC** reduces hardware area and power by 93.8% and 92.4%, respectively, compared to state-of-the-art low-discrepancy sequence designs. Crucially, it delivers the lowest computation error at ultra-short bit streams (e.g., 16 and 32 bits) among all evaluated architectures.

2 Background

In this section, we first introduce the foundational background of stochastic number generators (SNGs). We then review LFSRs and NLFSRs, the metric of spatial uniformity, and the rotational generation mechanism, all of which form the structural basis of our architecture.

2.1 Stochastic Number Generators

An SC circuit requires stochastic number generators (SNGs) to convert input binary numbers into SBSs [1]. As illustrated in Fig. 2(a), a typical SNG comprises an RNS and a comparator. In each clock cycle, the RNS generates an n-bit random binary number R. The comparator evaluates R against the n-bit input value X, outputting a '1' if $R < X$, and a '0' otherwise. This process generates an SBS with a ratio of 1s strictly equal to the input value X.

As the core component of the SNG, the RNS fundamentally determines the statistical quality of the SBS, which inherently dictates the overall computation accuracy of the SC circuit. We quantify this accuracy using the mean absolute error (MAE) [2], defined as:

$$\mathrm{MAE} = \frac{1}{K}\sum_{i=1}^{K}\left|y_{ideal}^{(i)} - y_{sc}^{(i)}\right|, \tag{1}$$

where K is the total number of evaluated patterns. The term $y_{ideal}^{(i)}$ represents the ideal expected result for the i-th output, and $y_{sc}^{(i)}$ denotes the actual numerical value encoded by the SC circuit's output SBS.

2.2 LFSR/NLFSR and Configuration Space Optimization

The linear feedback shift register (LFSR) is a hardware-efficient choice for RNS designs, constructed from D flip-flops (DFFs) and XOR gates, as presented in Fig. 2(a). However, a standard LFSR inherently excludes the all-zero state, preventing it from generating all 2^n possible values. To overcome this limitation, the LFSR is augmented into a non-linear feedback shift register (NLFSR) [5]. As shown in Fig. 2(b), an NLFSR incorporates a non-linear logic gate (typically a NOR gate) into its feedback path, enabling the valid generation of the all-zero pattern. This full-cycle property is critical for generating all 2^n states without omission.

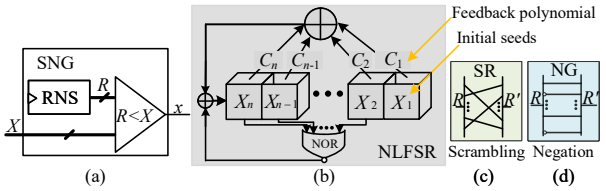


Figure 2: Illustration of (a) the SNG architecture, (b) the NLFSR architecture [5], (c) a scrambling (SR) module, and (d) a negation (NG) module.

Despite generating all states, the inherent linearity of (N)LFSRs often causes suboptimal accuracy due to inter-stream correlations [2]. To mitigate this, designers employ post-processing techniques such as scrambling (SR) [4, 21] and negation (NG) [22], which permute and selectively invert the output bits, respectively (Fig. 2(c) and (d)). However, co-optimizing these techniques requires navigating a massive configuration space encompassing feedback polynomials, initial seeds, scrambling patterns, and negation masks. Finding a high-accuracy configuration within this space remains computationally challenging.

2.3 The Metric of Spatial Uniformity

While an NLFSR guarantees the generation of all 2^n individual states, a critical metric governing the accuracy of multiple RNSs is *spatial uniformity*. For an m-input SC circuit operating at n-bit precision, the m random numbers generated by m RNSs per cycle form a single sample point in an m-dimensional (m-D) discrete integer space $\Omega = \{0, 1, \ldots, 2^n - 1\}^m$.

Recent work [29] formalized this concept by partitioning this m-D space into 2^n grid cells. This partitioning decouples the n-bit RNS outputs into a set of most significant bits (MSBs), defined as *leading-bits*, and the remaining *trailing-bits*. The leading-bits serve as a unique address assigning the sample point to a specific grid cell, while the trailing-bits define the precise coordinate within that cell.

A spatially uniform distribution is strictly achieved when these leading-bits exhaustively and uniquely generate all 2^n grid cell addresses over a full 2^n-cycle period, ensuring exactly one sample point lands in each cell. To satisfy this, the conventional architecture [29] employs a central n-bit uniform generator, whose outputs are hardwired to form the leading-bits of all m RNSs. Specifically, the first p RNSs receive $\lceil n/m \rceil$ leading-bits, while the remaining $(m-p)$ RNSs receive $\lfloor n/m \rfloor$ leading-bits, where $p = n \bmod m$. While this static architecture is highly effective for fixed-precision SC, its rigid hardwiring renders it fundamentally incompatible with the dynamic truncation requirements of PP, where uniformity must be maintained at multiple intermediate lengths.

2.4 Rotational RNS Generation

While spatial uniformity addresses the spatial distribution of sample points across multiple dimensions, the rotational approach is a deterministic technique used to reduce cross-correlation between multiple SBSs over time [14]. As illustrated in Fig. 3, this method selectively inhibits (pauses) the generation of one stream for a single clock cycle relative to another repeating reference stream (e.g., $a_0a_1a_2a_3$). This periodic pause effectively forces the second stream to rotate its sequence (e.g., from $b_0b_1b_2b_3$ to $b_3b_0b_1b_2$). This systematic shifting guarantees that over a full computational period, every bit of one stream interacts with every bit of the other stream exactly once, thereby structurally eliminating cross-correlation bias [14].

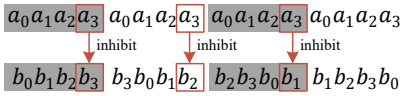


Figure 3: Illustration of the rotational generation mechanism [14].

3 Principle of Progressive Spatial Uniformity

Spatial uniformity is critical for minimizing computational bias in SC. Traditional RNS designs typically optimize for a fixed sequence length. In contrast, PP requires SBSs to maintain the uniformity of output sample points across multiple lower precision levels. We term this dynamic property *progressive spatial uniformity*. This principle ensures that the hardware can provide accurate early estimates before the entire bit stream is processed.

To establish a rigorous theoretical basis for this principle, we define the m-dimensional sample space as a discrete integer grid $\Omega = \{0, 1, \ldots, 2^n - 1\}^m$, corresponding to the n-bit outputs of m RNSs. We support PP from a minimum precision level m up to full precision n, enforcing uniformity at a selected subset of intermediate levels. Let $\mathcal{K} \subseteq \{m, m+1, \ldots, n\}$ strictly include both boundaries m and n. The physical space Ω remains fixed. Changing the precision level $k \in \mathcal{K}$ merely refines the partition granularity on Ω.

For any target level $k \in \mathcal{K}$, Ω is partitioned into 2^k disjoint grid cells $C_j^{(k)}$ $(0 \le j < 2^k)$. The partition of dimension i is governed by extracting b_i leading-bits from the RNS output (as defined in Section 2.3). To ensure the most uniform division across dimensions, these bits are allocated such that $b_i = \lceil k/m \rceil$ for the first $k \bmod m$ dimensions, and $b_i = \lfloor k/m \rfloor$ for the remaining dimensions, strictly satisfying $\sum_{i=1}^{m} b_i = k$. Dimension i is thus divided into 2^{b_i} equal intervals. Let $D_{j,i} \in \{0, \ldots, 2^{b_i} - 1\}$ denote the integer value of these b_i leading-bits. The Cartesian product of these intervals mathematically defines the grid cell $C_j^{(k)}$ as:

$$C_j^{(k)} = \bigtimes_{i=1}^{m} \left[D_{j,i} \cdot 2^{n-b_i},\ (D_{j,i}+1) \cdot 2^{n-b_i} - 1 \right]. \tag{2}$$

The k-bit cell address $L(C_j^{(k)})$ is the concatenation $(D_{j,1}, \ldots, D_{j,m})$. A hardware-generated sample point $P_g = (R_1, \ldots, R_m)$ locates within $C_j^{(k)}$ if and only if the b_i leading-bits of each coordinate R_i precisely match $D_{j,i}$.

The principle of progressive spatial uniformity on $\mathcal{K}$ is governed by two strict conditions:

(a) Uniformity at supported levels. For every $k \in \mathcal{K}$, the first 2^k sample points generated by the RNSs occupy all 2^k cells $\{C_j^{(k)}\}$ exactly once.

(b) Hierarchical nesting. For any supported levels $k' < k$ in $\mathcal{K}$, each of the first $2^{k'}$ sample points must remain within its original coarse grid cell when evaluated at the finer resolution k. Formally, if a point $P_g \in C_{j'}^{(k')}$, then its newly assigned finer cell $C_j^{(k)}$ at level k must satisfy $P_g \in C_j^{(k)} \subseteq C_{j'}^{(k')}$.

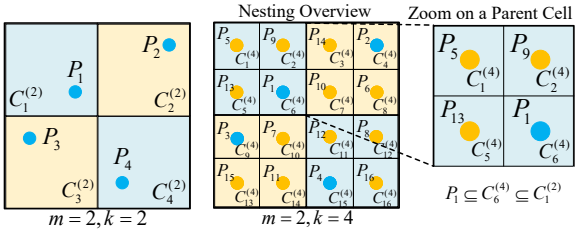


Figure 4: Illustration of the progressive spatial uniformity principle with $k = 2$ to $k = 4$.

To illustrate this, consider $m = 2$ and $n = 4$ (Fig. 4), yielding two 4-bit RNS outputs (R_1, R_2) per cycle. Suppose the first PP level is $k = m = 2$, where $b_1 = b_2 = 1$. Here, only the single MSB of each output forms a 2-bit cell address. Thus, Ω is divided into four level-2 cells, addressed by the four MSB pairs $(0,0), (0,1), (1,0), (1,1)$. To satisfy **Condition (a)** at $k = 2$, the first four cycles must generate MSB pairs covering these four addresses exactly once. For instance, the sequence may output $(R_1, R_2) \in \{(0\mathbf{xxx}, 0\mathbf{xxx}), (0\mathbf{xxx}, 1\mathbf{xxx}), (1\mathbf{xxx}, 0\mathbf{xxx}), (1\mathbf{xxx}, 1\mathbf{xxx})\}$, where '**x**' denotes an unspecified bit. This guarantees the induced point set uniformly occupies all four coarse cells.

When increasing to full precision $k = n = 4$, we refine each coordinate using two MSBs, dividing Ω into 16 finer level-4 cells. Uniformity at $k = 4$ requires the first 16 cycles to cover all 16 address pairs uniquely. **Condition (b)** further constrains the sequence traversal order: the initial four sample points must remain bounded within their original level-2 cells. The newly evaluated MSBs merely subdivide each existing coarse cell into four level-4 sub-cells. Advancing from $k = 2$ to $k = 4$ thus deterministically refines the distribution without causing any point to migrate across the established coarse boundaries.

4 Proposed ProU-SC Architecture

In this section, we propose the **ProU-SC** architecture to implement the principle of progressive spatial uniformity with low hardware

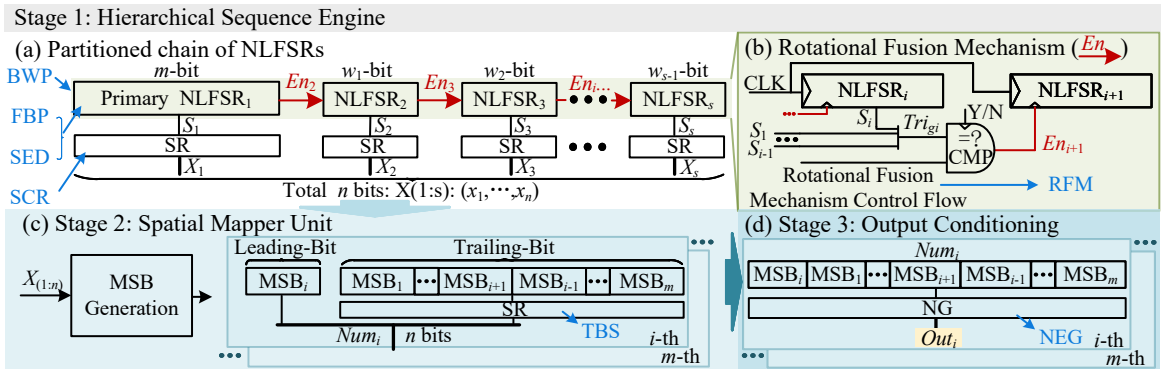


Figure 5: Architecture of ProU-SC. The decoupled generation pipeline comprises (a) a partitioned chain of NLFSRs, (b) a rotational fusion mechanism (RFM) for hierarchical sequence control, (c) a spatial mapper via the hardwired bit-interleaving scheme, and (d) an output conditioning stage.

cost. To explain this design clearly, we first introduce the design intuition. Then, we detail the decoupled architecture, which separates the generation process into a three-stage pipeline, as shown in Fig. 5.

4.1 Design Rationale: From Static to Progressive Uniformity

As discussed in Section 2.3, existing hardware-efficient architectures achieve static spatial uniformity by using a central n-bit uniform generator and statically partitioning its outputs [29]. However, this rigid hardwiring forces a pseudo-random state traversal. When the bit stream is truncated early, this traversal causes severe spatial clustering.

To break this static bottleneck and achieve progressive spatial uniformity, the system must guarantee exhaustive address generation at intermediate truncation boundaries (e.g., 2^m cycles). This requirement drives the design of **ProU-SC**. Instead of using a single large generator, our architecture decomposes the sequence generation into a chain of smaller sub-engines. This ensures that at any intermediate truncation boundary, the active sub-engines can complete a full cycle of all their possible states. Concurrently, we design a deterministic bit-interleaving scheme. It maps these fully-traversed bits directly to the MSB positions of the spatial coordinates. By strictly locking the highest-significance addresses first, any subsequently generated bits merely subdivide the existing coarse grid cells, naturally realizing the hierarchical spatial nesting required by PP.

Based on this intuition, we implement **ProU-SC** through three pipelined stages:

(1) **Stage 1: Hierarchical Sequence Engine** (Fig. 5(a) and (b)). It coordinates a partitioned chain of NLFSRs and a rotational fusion mechanism (RFM) to control the state traversal sequence (Section 4.2).
(2) **Stage 2: Spatial Mapper** (Fig. 5(c)). It hardwires these sequence states into multi-dimensional spatial coordinates via a bit-interleaving scheme (Section 4.3).
(3) **Stage 3: Output Conditioning** (Fig. 5(d)). It further reduces the correlation among the m outputs through selective negation to maximize final computation accuracy (Section 4.4).

4.2 Stage 1: Hierarchical Sequence Engine

Guided by the design rationale, the core of the sequence engine is a partitioned chain of hardware-efficient NLFSRs, as shown in Fig. 5(a). We decompose the generation logic into a single m-bit primary NLFSR ($NLFSR_1$) and $s-1$ secondary NLFSRs ($NLFSR_2$ through $NLFSR_s$). The bit-width partitioning (BWP) satisfies $n = m + \sum_{i=1}^{s-1} w_i$, where w_i denotes the bit-width of $NLFSR_{i+1}$. Each NLFSR is independently configurable with its own feedback polynomial (FBP) and seed initialization (SED). This decomposition establishes a generation hierarchy that aligns precisely with the progressive spatial truncation boundaries.

For this chain to function as a unified n-bit generator, the state transitions must be carefully controlled. Inspired by rotational sequence alignment [14], we propose the RFM to govern the secondary engines, as shown in Fig. 5(b).

Let $S_i(t)$ denote the state of $NLFSR_i$ at clock cycle t. The primary $NLFSR_1$ operates continuously. For any subsequent $NLFSR_{i+1}$, its clock enable signal $EN_{i+1}(t)$ is generated by a comparator (CMP_i) and deterministically controlled by the preceding engines:

$$EN_{i+1}(t) = \begin{cases} 1, & \text{if } (S_1(t) \parallel \cdots \parallel S_i(t)) = Trig_i, \\ 0, & \text{otherwise,} \end{cases} \tag{3}$$

where $\parallel$ denotes bitwise concatenation, and $Trig_i$ is a pre-configured trigger value.

Analogous to a multi-radix counter, this hierarchical pause mechanism guarantees that the concatenated global state vector ($S_1 \parallel S_2 \parallel \cdots \parallel S_s$) exhaustively traverses all possible state combinations over exactly 2^n clock cycles.

This cascaded architecture naturally provides two critical features for SC. First, it enables sequence reconfigurability. Altering the trigger values ($Trig_i$) dynamically reconfigures the global state traversal sequence. This helps minimize inter-sequence correlation without changing the underlying FBP. Second, it supports dynamic power-gating. When computing at lower precision bounds, the comparators CMP_i can be disabled. This keeps the higher-order secondary NLFSRs inactive to save power.

Finally, an optional scrambling (SCR) module can be applied to each engine's output to inject additional non-linearity, as shown in Fig. 5(a). The individually scrambled outputs are then concatenated to form the final n-bit global sequence state vector $X_{(1:s)} =$

$\{x_1, x_2, \ldots, x_n\}$. This sequence vector serves as the direct output of the hierarchical sequence engine and is then forwarded to the spatial mapping stage.

4.3 Stage 2: Spatial Mapper

Taking the n-bit sequence state vector $X_{(1:s)}$ as input, the spatial mapper explicitly partitions and routes this sequence state into m independent n-bit spatial coordinates, as shown in Fig. 5(c). These coordinates are denoted as final output numbers Out_i $(1 \le i \le m)$.

Each output Out_i concatenates one leading-bit group (MSB_i) and $(m-1)$ trailing-bit groups. The leading group determines the spatial grid-cell address, while the trailing groups act as fine-grained coordinate adjustments within that cell.

The core of this mapping is the MSB generation block, which constructs all m MSB_i groups directly from $X_{(1:s)}$ via a hardwired bit-interleaving scheme. Let x_l denote the l-th most significant bit of the source vector $X_{(1:s)}$ $(1 \le l \le n)$. The j-th bit of the i-th MSB group, denoted as $MSB_i[j]$, is statically routed from x_l:

$$MSB_i[j] \xleftarrow{\text{routed from}} x_l, \quad \text{where } l = (j-1)m + i, \tag{4}$$

for $1 \le i \le m$ and $1 \le j \le \lceil (n-i+1)/m \rceil$. The unified upper bound for j mathematically encapsulates the exact p and $(m-p)$ fractional bit distribution established in Section 2.3. To clearly illustrate this mapping, Fig. 6 visualizes an example of a 7-bit source vector ($n = 7$) mapped into three spatial dimensions ($m = 3$).

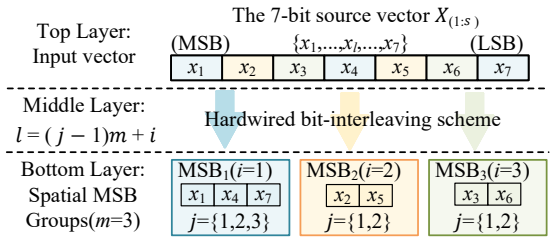


Figure 6: Example of Hardwired bit-interleaving scheme: a 7-bit state vector mapped to $m = 3$ spatial MSB groups.

Structurally, Eq. (4) dictates that the highest-activity bits of $X_{(1:s)}$ generated by the primary NLFSR are unconditionally allocated to the highest spatial significance positions across all m dimensions. Therefore, this specific hardwired topology inherently satisfies both progressive uniformity constraints established in Section 3.

First, this routing scheme ensures spatial uniformity at every supported level (**Condition a**). Consider a truncation boundary k corresponding to the activation of the primary NLFSR and the first u secondary NLFSRs. The RFM guarantees that the first k bits of $X_{(1:s)}$ exhaustively traverse all 2^k possible states. Crucially, Eq. (4) directly routes these exact k bits to form the complete k-bit cell address vector. This one-to-one mapping immediately translates the 2^k temporal states into the exhaustion of all 2^k grid cell addresses in the m-dimensional space Ω. This fulfills the uniformity requirement.

Second, this MSB-first allocation strictly guarantees hierarchical nesting (**Condition b**). As the precision level increases, newly generated source bits are deterministically routed to lower-significance bit positions. This leaves the higher-order leading-bit address completely unchanged. Consequently, the newly added bits merely subdivide the existing grid cells of Ω without altering the previously assigned cell address of any sample point.

Finally, for a given Out_i, the remaining $(m-1)$ MSB groups are assembled as trailing bits. A trailing-bit scrambling (TBS) module is structurally embedded to provide degrees of freedom for further reducing correlation between the output streams.

4.4 Stage 3: Output Conditioning

The fully assembled n-bit random numbers eventually pass to the output conditioning stage. The goal of this stage is to further reduce the correlation among the m outputs and improve the final computation accuracy, as shown in Fig. 5(d).

This is achieved by applying a standard negation (NG) module. Each Out_i is controlled by a statically configured n-bit output negation (NEG) mask, denoted as $M \in \{0,1\}^n$. In this hardware-level module, a bit '1' in the mask dictates that the corresponding bit of Out_i is inverted, while a '0' preserves the original value. In our NLFSR-based hardware, each DFF simultaneously provides both a non-inverting output (Q) and an inverting output ($\bar{Q}$). Consequently, this negation module incurs zero hardware overhead, as any specific bit is inverted simply by routing from the $\bar{Q}$ terminal rather than the Q terminal.

Crucially, because this bitwise inversion acts as a constant, global permutation across the entire m-dimensional space, it merely swaps the assigned grid cells for all sample points simultaneously. Since the underlying sequence exhaustively covers the space, this one-to-one mapping strictly preserves the progressive spatial uniformity established in the previous stage.

5 Configuration Search and Optimization

Although the **ProU-SC** hardware structurally guarantees spatial uniformity, its large configuration space leaves the precise sequence traversal order unconstrained. Randomly configuring these parameters may introduce computational bias, degrading both early-estimate and final accuracy. Consequently, explicit configuration optimization is necessary. This section quantifies the configuration space, introduces a mathematical pruning technique, and formulates a decoupled heuristic search algorithm to navigate this space efficiently.

Table 1: Config Subspaces Breakdown.

Sub-space	Configuration (Abbr.)	Size of Sub-space
C_{NLFSR}	Bit-width Partitioning (BWP)	2^{n-m-1}
	Feedback Polynomial (FBP)	$\prod_{i=1}^{s} \lvert A_i \rvert$
	Seed Initialization (SED)	2^n
	Scrambling Application (SCR)	$\prod_{i=1}^{s} (w_i!)$
C_{RFM}	RFM Trigger	$2^{n-w_{s-1}}$
C_{TBS}	Trailing-Bit Scrambling (TBS)	$((n-\lceil n/m \rceil)!)^p \times ((n-\lfloor n/m \rfloor)!)^{m-p}$
C_{NEG}	Output Negation Mask (NEG)	$(2^n)^m$

5.1 Configuration Space Complexity

We define a configuration as a tuple $c = (c_{\text{NLFSR}}, c_{\text{RFM}}, c_{\text{TBS}}, c_{\text{NEG}})$, which encapsulates the parameters for the NLFSR sequence engines, the rotational fusion trigger logic, the trailing-bit scrambling, and the output negation masks, respectively. The set of all valid configurations, C_{Total}, is upper-bounded by the product of its sub-spaces: $|C_{\text{Total}}| \le |C_{\text{NLFSR}}| \times |C_{\text{RFM}}| \times |C_{\text{TBS}}| \times |C_{\text{NEG}}|$.

Assume $|A_i|$ is the number of possible feedback polynomials for the i-th NLFSR. Table 1 summarizes a quantitative breakdown of the configuration sub-spaces. Even for a standard 8-bit 2-input SC multiplier, the configuration space scales factorially due to sequence scrambling (C_{SCR}, C_{TBS}) and exponentially due to output negation (C_{NEG}). This combinatorial explosion readily expands the unpruned global search space to a computationally intractable scale, rendering monolithic search heuristics ineffective.

5.2 Negation Space Pruning via SNG Symmetry

To make this massive space navigable, it is imperative to first eliminate structural redundancies. While the output negation subspace (C_{NEG}) is essential for reducing the correlation among the outputs, it inherently contains symmetrically equivalent configurations that stall heuristic optimization. Unlike sequence permutations where redundancy is difficult to isolate, the negation space can be mathematically folded without losing optimality by exploiting the intrinsic symmetry of SNGs.

As established in Section 4.4, the output conditioning stage applies an n-bit negation mask $M \in \{0,1\}^n$ to selectively invert the bits of $R^{(i)}$. Let $R^{(i)}_{mask}$ denote the resultant numerical value after applying mask M. Mathematically, applying the all-bit complement of this mask ($\bar{M} = M \oplus \{1\}^n$) unconditionally inverts every bit of $R^{(i)}_{mask}$. This operation equates to a full bitwise NOT on the numerical value, expressed as $\bar{R}^{(i)}_{mask} = (2^n - 1) - R^{(i)}_{mask}$.

Lemma 1. Let the random number domain be $\mathcal{R} = \{0, \ldots, 2^n - 1\}$ and the threshold domain be $\mathcal{Y} = \{0, \ldots, 2^n\}$. Evaluating the fully negated sequence $\bar{R}_{mask}$ against a threshold $Y \in \mathcal{Y}$ yields the exact bitwise complement of the bit stream generated by the original R_{mask} evaluated against $Y' = 2^n - Y$.

Proof: For cycle i, the comparator outputs $b^{(i)} = 1$ if $R^{(i)}_{mask} < Y$. Evaluating $\bar{R}^{(i)}_{mask}$ against Y' yields output $\bar{b}^{(i)} = 1$ when $(2^n - 1) - R^{(i)}_{mask} < 2^n - Y \iff Y - 1 < R^{(i)}_{mask}$. Since variables are integers, this simplifies to $Y \leq R^{(i)}_{mask}$, the exact logical complement of the original trigger condition. Thus, $\bar{b}^{(i)} = \neg b^{(i)}$.

By Lemma 1, applying an all-bit complement to mask M inherently inverts the output bit stream. Under exhaustive uniform evaluation, this bit stream inversion merely permutes the input test vectors, keeping the globally aggregated MAE strictly invariant. Consequently, applying an all-bit complement to the n-bit negation mask of any output yields an exactly MAE-equivalent configuration.

By extending this symmetry to all m outputs, we identify 2^m equivalent mask choices per global state. To fold this space, we enforce a canonical representation. For each output, the MSB of mask M is fixed to 0 (i.e., $M < 2^{n-1}$), effectively discarding its complement. This explicit rule prunes C_{NEG} from $(2^n)^m$ to $2^{(n-1)m}$. By eliminating these symmetrically redundant configurations, we significantly reduce the size of the search space, thereby accelerating the heuristic optimization process.

5.3 Decoupled Heuristic Search Algorithm

To quantify early-estimate accuracy under the PP paradigm, we first define a weighted score (WS) across B truncation boundaries:

$$WS = \sum_{i=0}^{B} 2^{-i} \cdot \text{MAE}_{2^{n-i}}, \tag{5}$$

where $\text{MAE}_{2^{n-i}}$ denotes the MAE evaluated using only the first 2^{n-i} bits of the generated SBS. The decaying weight 2^{-i} penalizes the larger errors that naturally occur at shorter truncation boundaries, ensuring that a lower WS reflects better early-estimate accuracy.

To efficiently minimize such target metrics within the large configuration space, our proposed heuristic search algorithm exploits an inherent structural asymmetry in parameter sensitivities. Specifically, the front-end generation parameters (C_{NLFSR}, C_{RFM}) dictate the sequence traversal order, which primarily governs the accuracy at early truncation boundaries. Conversely, the back-end parameters (C_{TBS}, C_{NEG}) perform spatial value mapping, which primarily determines the final accuracy. Based on this structural decoupling, we partition the configuration tuple into two blocks: the sequence generation block $\mathcal{P}_{pp} = \{C_{\text{NLFSR}}, C_{\text{RFM}}\}$ and the spatial mapping block $\mathcal{P}_{acc} = \{C_{\text{TBS}}, C_{\text{NEG}}\}$.

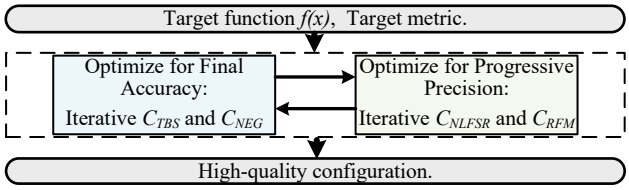


Figure 7: Flowchart of the decoupled iterative heuristic search algorithm.

Leveraging this partitioned structure, we propose an alternating iterative search algorithm, as shown in Fig. 7. The algorithm alternates between two phases to minimize a designated target metric. In Phase 1, while keeping $\mathcal{P}_{pp}$ fixed, the algorithm randomly samples $\mathcal{P}_{acc}$ to greedily minimize the target metric. In Phase 2, while fixing the updated $\mathcal{P}_{acc}$, it randomly samples $\mathcal{P}_{pp}$ to further minimize the same target metric. Any configuration yielding a superior metric immediately updates the current best state for the subsequent search iteration.

Iterating until no improvements occur in either block, this decoupled approach effectively finds a high-quality configuration within the intractable search space. Crucially, this algorithm provides two distinct operating modes by simply switching the target metric. For systems prioritizing final accuracy, the target metric is restricted to the full-length MAE_{2^n}. For systems requiring robust progressive precision, the target metric is set to the global WS, thereby physically co-optimizing early estimation and final accuracy.

6 Experimental Results

This section evaluates the performance of the proposed **ProU-SC** architecture against several existing RNS designs.

6.1 Experimental Setup

We evaluated our design across nine benchmarks: 2-, 3-, and 4-input SC multipliers (denoted as MUL-2, MUL-3, MUL-4, respectively); three non-linear functions ($\sin(x)$, $\cos(x)$, $\tanh(x)$) [20]; and three FIR filters of 8th, 16th, and 32nd orders (FIR-8, FIR-16, FIR-32).

To establish a comprehensive baseline, **ProU-SC** is compared against five representative SNG architectures. For clarity, we denote these prior works as follows: **Ind-LFSR** [4], the conventional approach assigning independent LFSRs; **Share-LFSR** [21], an area-efficient design sharing a single LFSR via SR modules; **SU-LFSR** [29], a state-of-the-art design achieving static spatial uniformity; **LDS-RNS** [15], a high-accuracy design based on the Sobol sequence; and **VDC-RNS** [17], the latest low-discrepancy generator based on the Van der Corput sequence. For an m-input SC circuit, all evaluated baselines adhere to the independent topology, assigning m distinct SNGs to drive the m inputs. Consequently, the sole architectural differentiator is the underlying RNS generation mechanism.

To evaluate the two operating modes defined in Section 5.3, we generate two distinct configurations. **ProU-SC (Acc)** optimizes exclusively for the full-length MAE_{2^n}. It employs a coarse-grained hardware decomposition consisting of only two engines: an m-bit primary NLFSR and a single $(n-m)$-bit secondary NLFSR. Conversely, **ProU-SC (PP)** optimizes the global WS by employing a fine-grained generation chain: an m-bit primary NLFSR followed by $(n-m)$ individual 1-bit secondary NLFSRs. This fine-grained partitioning maximizes the resolution of intermediate truncation boundaries.

All experiments were executed on a 4.30GHz CPU with 32GB RAM. For a fair comparison, the configuration search time allocated for the baselines on each benchmark matched the exact runtime of our heuristic algorithm for **ProU-SC (PP)**. These bounded search times were: 3, 3, and 7 minutes for the 2-, 3-, and 4-input multipliers; 4, 5, and 5 minutes for the $\sin(x)$, $\cos(x)$, and $\tanh(x)$ functions; and 5, 5, and 6 minutes for the FIR-8, FIR-16, and FIR-32 filters. All accuracy evaluations target a standard 8-bit precision, establishing a full stream length of 256 bits. For hardware evaluation, all designs were synthesized using Synopsys Design Compiler [24] and the Nangate 45nm library [18]. Because SNGs typically dominate the hardware cost of SC circuits, our evaluations strictly report the total area, power, and delay of the SNGs required by each benchmark.

6.2 Final Accuracy Comparison

Fig. 8 evaluates the MAE at the full 256-bit precision. **ProU-SC (Acc)** achieves a competitive average MAE of 0.477%, demonstrating significant error reduction compared to Ind-LFSR (0.508%) and Share-LFSR (0.873%). Furthermore, it surpasses structured generators like VDC-RNS (1.143%) and matches SU-LFSR (0.454%). While the hardware-intensive LDS-RNS yields a marginally lower global average (0.395%) due to its performance in high-order FIR filters, **ProU-SC (Acc)** achieves the lowest MAE in fundamental benchmarks, including MUL-2, $\sin(x)$, and $\tanh(x)$.

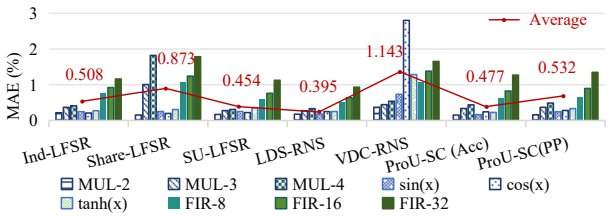


Figure 8: MAE (%) comparison at the full 256-bit precision for different RNS designs.

This accuracy advantage stems from the hardwired spatial uniformity of **ProU-SC** at the full 2^n-bit length, which eliminates the spatial clustering degrading conventional LFSRs. Consequently, even under the strict constraints imposed to co-optimize for PP, **ProU-SC (PP)** maintains a robust 0.532% average MAE, validating its stability at full precision.

6.3 Progressive Precision Comparison

Fig. 9 evaluates the PP performance via WS. **ProU-SC (PP)** consistently dominates across all nine benchmarks, yielding an extremely low average WS of 1.85%. This significantly outperforms the leading low-discrepancy sequence baseline, LDS-RNS (2.29%).

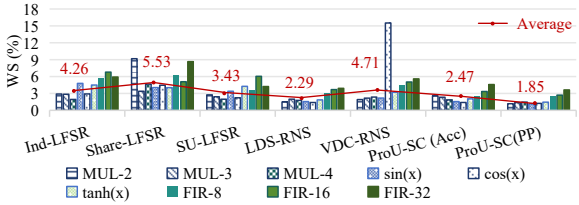


Figure 9: WS (%) comparison for different RNS designs.

Furthermore, the data highlights the inherent difficulty of early estimation in complex non-linear applications. For instance, while VDC-RNS achieves an average WS of 4.71%, its error metric increases to 15.52% when evaluating the $\cos(x)$ function. In contrast, **ProU-SC (PP)** maintains a stable WS of 1.24% on this identical challenging target, confirming the robustness of our sequence optimization against varying Boolean sensitivities.

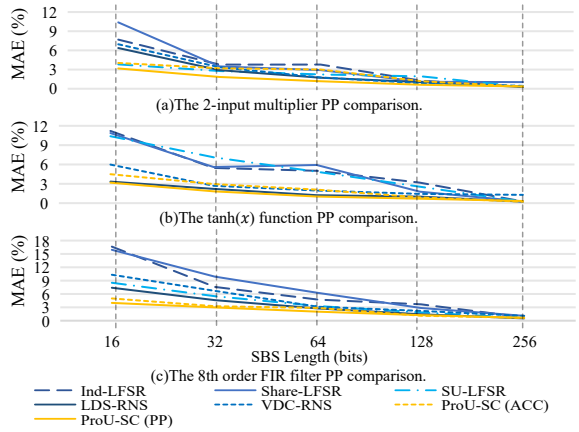


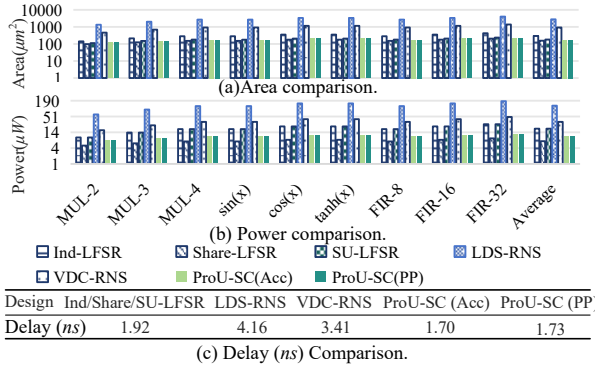
Figure 10: PP performance comparison for (a) 2-input multiplier, (b) $\tanh(x)$ function, and (c) the 8th order FIR filter (FIR-8) across different bit stream lengths.

To illustrate this PP performance, Fig. 10 plots MAE trajectories across different SBS lengths for three representative benchmarks. As explicitly demonstrated, **ProU-SC (PP)** consistently achieves the fastest error reduction, particularly at ultra-short SBSs (2^4 and 2^5 bits). In the complex FIR-8 filter (Fig. 10(c)), the traditional Share-LFSR exhibits an MAE exceeding 16% at 16 bits. Similarly, while conventional LFSRs suffer severe degradation in $\tanh(x)$ (Fig. 10(b)), **ProU-SC (PP)** suppresses the MAE to highly usable levels (~3%) at this identical short length. This evidence corroborates the aggregated WS metrics, proving **ProU-SC** outperforms even the hardware-intensive LDS-RNS in early estimation.

6.4 Hardware Cost Comparison

As shown in Fig. 11, **ProU-SC (PP)** delivers substantial area and power savings over LDS-RNS (93.8%/92.4%), VDC-RNS (81.5%/72.2%), Ind-LFSR (41.6%/48.3%), and SU-LFSR (5.7%/48.3%). Concurrently, it maintains a hardware cost approaching that of Share-LFSR, the most lightweight baseline evaluated.

Since the delay of an SNG is typically identical across different circuits for a fixed bit-width, Fig. 11(c) directly compares the absolute delay of each RNS design. **ProU-SC (PP)** operates at 1.73 *ns*, achieving significant delay reductions over LDS-RNS (58.4%), VDC-RNS (49.3%), and the conventional LFSR baselines (9.9%). Furthermore, **ProU-SC (Acc)** optimizes the delay to 1.70 *ns*. Benefiting from this hierarchical partitioning, **ProU-SC (Acc)** yields a strictly lower average area-delay product (ADP) than even the lightweight Share-LFSR, confirming its high hardware efficiency.



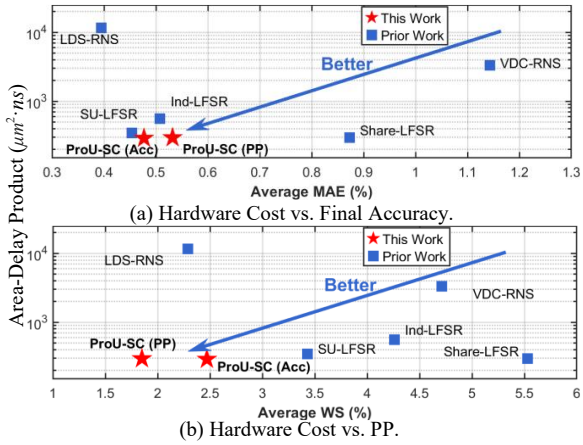
Design	Ind/Share/SU-LFSR	LDS-RNS	VDC-RNS	ProU-SC (Acc)	ProU-SC (PP)
Delay (*ns*)	1.92	4.16	3.41	1.70	1.73

(c) Delay (*ns*) Comparison.

Figure 11: Hardware cost comparison of different SNG architectures across all benchmarks: (a) Area (μm^2), (b) Power (μW), and (c) Delay (*ns*).

6.5 Accuracy-Cost Trade-off Analysis

Fig. 12 visualizes the accuracy-cost trade-offs by synthesizing the multi-dimensional metrics evaluated above. The y-axis represents the hardware efficiency via ADP, while the x-axis denotes the final MAE and the progressive WS, respectively.



(a) Hardware Cost vs. Final Accuracy.

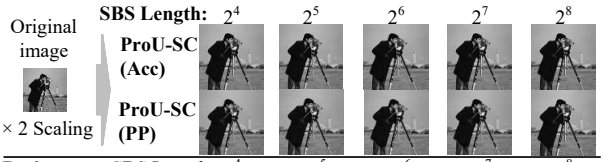
(b) Hardware Cost vs. PP.

Figure 12: Accuracy-cost trade-off analysis: (a) Hardware Cost vs. Final Accuracy, and (b) Hardware Cost vs. PP.

These scatter plots clearly illustrate the two extremes of existing design spaces. Hardware-intensive designs like LDS-RNS cluster in the extreme top-left, whereas lightweight variants like Share-LFSR occupy the bottom-right due to substantial accuracy degradation. In stark contrast, both **ProU-SC (Acc)** and **ProU-SC (PP)** exclusively anchor the optimal bottom-left quadrant. Operating with a low ADP ($< 300\mu m^2 \cdot ns$), **ProU-SC** delivers a massive efficiency gain over LDS-RNS. Concurrently, it achieves a significantly lower MAE and WS compared to Share-LFSR. By occupying the most favorable trade-off position across both final accuracy and progressive precision domains, **ProU-SC** demonstrates high cost-effectiveness and architectural balance.

6.6 Application-Level Case Study: Bilinear Interpolation

Finally, we applied our design to bilinear image interpolation for ×2 scaling [7]. Fig. 13 illustrates the visual quality across intermediate truncation lengths. For quantitative comparison, we evaluated other baseline designs, with their peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM) listed in the embedded table. The results show that **ProU-SC** maintains high visual fidelity even at early truncation (2^4 bits), approaching near-perfect accuracy at the full stream length.



Design	SBS Length	2^4	2^5	2^6	2^7	2^8
Ind-LFSR	PSNR	25.618	39.698	31.548	32.780	45.946
	SSIM	0.643	0.919	0.919	0.954	0.996
Share-LFSR	PSNR	25.420	31.146	32.005	35.916	46.121
	SSIM	0.693	0.884	0.952	0.978	0.997
SU-LFSR	PSNR	25.247	25.511	26.449	36.587	54.558
	SSIM	0.803	0.912	0.910	0.966	0.998
LDS-RNS	PSNR	40.688	39.333	38.378	40.194	48.005
	SSIM	0.862	0.905	0.934	0.970	0.998
VDC-RNS	PSNR	42.699	41.490	40.416	39.655	53.201
	SSIM	0.842	0.905	0.942	0.956	0.997
ProU-SC (Acc)	PSNR	33.170	38.342	43.597	43.360	54.387
	SSIM	0.797	0.914	0.944	0.975	0.998
ProU-SC (PP)	PSNR	39.925	40.359	40.308	40.338	54.258
	SSIM	0.803	0.900	0.961	0.986	0.998

Figure 13: Visual and quantitative evaluation of different SNG architectures for bilinear image interpolation (×2 scaling).

7 Conclusion

In this paper, we introduced **ProU-SC**, an area-efficient architecture that physically implements the principle of progressive spatial uniformity for SC. By decoupling sequence generation from spatial routing, **ProU-SC** achieves a highly favorable accuracy-cost trade-off. Experimental results confirm that our co-optimized configuration, **ProU-SC (PP)**, delivers the lowest computation error at ultra-short bit streams (e.g., 16 and 32 bits) across all evaluated baselines, while requiring a hardware cost comparable to the most lightweight designs. Consequently, **ProU-SC** provides a highly effective and scalable hardware solution for resource-constrained SC applications. In the future, we will explore dynamic on-chip reconfigurability for real-time precision control.

References

[1] A. Alaghi et al. 2018. The Promise and Challenge of Stochastic Computing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 37, 8 (2018), 1515–1531.

[2] A. Alaghi and J. P. Hayes. 2013. Exploiting Correlation in Stochastic Circuit Design. In *International Conference on Computer Design*. 39–46.

[3] A. Alaghi and J. P. Hayes. 2014. Fast and Accurate Computation Using Stochastic Circuits. In *Design, Automation & Test in Europe Conference & Exhibition*. 1–4.

[4] J. H. Anderson et al. 2016. Effect of LFSR Seeding, Scrambling and Feedback Polynomial on Stochastic Computing Accuracy. In *Design, Automation & Test in Europe Conference & Exhibition*. 1550–1555.

[5] B. Arazi. 1981. On the Synthesis of de-Bruijn Sequences. *Information and Control* 49, 2 (1981), 81–90.

[6] S. Asadi et al. 2021. A Low-Cost FSM-based Bit-Stream Generator for Low-discrepancy Stochastic Computing. In *Design, Automation & Test in Europe Conference & Exhibition*. IEEE, 908–913.

[7] S. Aygun et al. 2023. Agile Simulation of Stochastic Computing Image Processing With Contingency Tables. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 42, 10 (2023), 3474–3478.

[8] T. J. Baker et al. 2021. Benefits of Stochastic Computing in Hearing Aid Filterbank Design. In *IEEE Biomedical Circuits and Systems Conference*. 1–5.

[9] K.-C. Chen and C.-H. Wu. 2021. High-Accurate Stochastic Computing for Artificial Neural Network by Using Extended Stochastic Logic. In *International Symposium on Circuits and Systems*. 1–4.

[10] B. R. Gaines. 1967. Stochastic Computing. In *AFIPS Spring Joint Computer Conference*. 149–156.

[11] O. S. Hoffend and J. P. Hayes. 2026. Runtime Termination Control in Stochastic Computing. *IEEE Transactions on Emerging Topics in Computing* 14, 1 (2026), 69–79.

[12] H. Hsiao et al. 2022. Streaming Accuracy: Characterizing Early Termination in Stochastic Computing. In *Asia and South Pacific Design Automation Conference*. 320–325.

[13] H. Ichihara et al. 2019. Compact and Accurate Digital Filters Based on Stochastic Computing. *IEEE Transactions on Emerging Topics in Computing* 7, 1 (2019), 31–43.

[14] D. Jenson and M. Riedel. 2016. A Deterministic Approach to Stochastic Computation. In *International Conference on Computer-Aided Design*. 1–8.

[15] S. Liu and J. Han. 2017. Energy Efficient Stochastic Computing with Sobol Sequences. In *Design, Automation & Test in Europe Conference & Exhibition*. 650–653.

[16] Y. Liu et al. 2021. A Survey of Stochastic Computing Neural Networks for Machine Learning Applications. *IEEE Transactions on Neural Networks and Learning Systems* 32, 7 (2021), 2809–2824.

[17] M. S. Moghadam et al. 2023. Accurate and Energy-Efficient Stochastic Computing with Van Der Corput Sequences. In *International Symposium on Nanoscale Architectures*. 1–6.

[18] Nangate Inc. 2021. http://www.nangate.com.

[19] F. Neugebauer et al. 2018. S-box-based Random Number Generation for Stochastic Computing. *Microprocessors and Microsystems* 61 (2018), 316–326.

[20] K. Parhi and Y. Liu. 2019. Computing Arithmetic Functions Using Stochastic Logic by Series Expansion. *IEEE Transactions on Emerging Topics in Computing* 7, 1 (2019), 44–59.

[21] S. A. Salehi. 2020. Low-Cost Stochastic Number Generators for Stochastic Computing. *IEEE Transactions on Very Large Scale Integration Systems* 28, 4 (2020), 992–1001.

[22] S. A. Salehi. 2024. Area-Efficient LFSR-Based Stochastic Number Generators With Minimum Correlation. *IEEE Design and Test* 41, 1 (2024), 50–59.

[23] R. Sengupta et al. 2024. Performance and Error Tolerance of Stochastic Computing-Based Digital Filter Design. In *International Symposium on Design & Diagnostics of Electronic Circuits & Systems*. 7–12.

[24] Synopsys Inc. 2021. http://www.synopsys.com.

[25] D. Wu et al. 2020. UGEMM: Unary Computing Architecture for GEMM Applications. In *International Symposium on Computer Architecture*. 377–390.

[26] D. Wu et al. 2021. Normalized Stability: A Cross-Level Design Metric for Early Termination in Stochastic Computing. In *Asia and South Pacific Design Automation Conference*. 254–259.

[27] T. Xie et al. 2024. ASCEND: Accurate yet Efficient End-to-End Stochastic Computing Acceleration of Vision Transformer. In *Design, Automation & Test in Europe Conference & Exhibition*. 1–6.

[28] B. Yuan and Y. Wang. 2016. High-Accuracy FIR Filter Design Using Stochastic Computing. In *IEEE Computer Society Annual Symposium on VLSI*. 128–133.

[29] K. Zhong et al. 2026. Low-Cost High-Accuracy Random Number Source Design for Stochastic Computing via Exploitation of Uniform Spatial Distribution. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 45, 2 (2026), 705–718.