# Mathematical Foundations for Stochastic Computing Formal Verification in Lean 4

This report outlines the core mathematical theorems, proof strategies, and Lean 4 (Mathlib) components required to formally verify a Stochastic Computing (SC) system. The verification focuses on the correctness of expected values and the analysis of error bounds for finite-length bitstreams.

## 1. Core Stochastic Computing Theory

#### Fundamental Theorem of Stochastic Representation

A real number $v \in [0, 1]$ (unipolar format) is represented by an infinite sequence of independent and identically distributed (i.i.d.) random variables (bits) $X = (X_1, X_2, X_3, \dots)$, where each $X_i \sim \text{Bernoulli}(v)$.

- **Formal Statement:** The value $v$ is defined as the expected value of any single bit:

$$\forall i, v = \mathbb{E}[X_i] = \mathbb{P}(X_i = 1)$$

- **Correctness Proof:** The law of large numbers (specifically the Strong Law of Large Numbers, SLLN) proves that the empirical mean (the measured value) converges to the true value $v$ almost surely as the bitstream length $N \to \infty$.

$$\hat{V}_N = \frac{1}{N} \sum_{i=1}^{N} X_i \xrightarrow{a.s.} v$$

- **Independence Assumption:** The core mathematical justification for simplicity in SC circuits is the assumption that input bitstreams are **statistically independent** *and* that the bits within each stream are **i.i.d.** (independent and identically distributed). This is the **minimum assumption** required for both multiplication and scaled addition correctness.
- **LFSR Theorems:** LFSRs generate pseudo-random sequences. Their statistical justification relies on theorems about maximum-length sequences ($m$-sequences), detailed in Section 5.

## 2. Stochastic Multiplication (AND Gate)

#### Theorem 2.1: Correctness of Stochastic Multiplication (Unipolar)

Let $A$ and $B$ be two independent stochastic bitstreams representing $P_A$ and $P_B$ respectively. The output stream $Z = A \wedge B$ (bitwise AND) represents $P_Z = P_A \cdot P_B$.

- **Proof Sketch (Expected Value):**
  1. Define the output bit $Z_i = A_i \wedge B_i$.
  2. The expected value of the output bit is $\mathbb{E}[Z_i] = \mathbb{P}(Z_i = 1)$.
  3. By the definition of the AND operation: $\mathbb{P}(Z_i = 1) = \mathbb{P}(A_i = 1 \text{ and } B_i = 1)$.
  4. **Crucial Condition (Minimal Assumption):** Assume the bits $A_i$ and $B_i$ are statistically independent.
  5. By the definition of independence: $\mathbb{P}(A_i = 1 \text{ and } B_i = 1) = \mathbb{P}(A_i = 1) \cdot \mathbb{P}(B_i = 1)$.

6. Since $\mathbb{P}(A_i = 1) = P_A$ and $\mathbb{P}(B_i = 1) = P_B$, we have $\mathbb{P}(Z_i = 1) = P_A P_B$.
7. Therefore, the expected value is $\mathbb{E}[Z_i] = P_A P_B$.

#### Correlation Effects (Question 2)

If $A_i$ and $B_i$ are correlated, the multiplication theorem fails:

$$\mathbb{P}(Z_i = 1) = \mathbb{P}(A_i = 1 \text{ and } B_i = 1) = P_A P_B + \text{Cov}(A_i, B_i)$$

The output value deviates from $P_A P_B$ by the covariance of the input bits. Correlation introduces a **bias** error, which does not diminish with increasing bitstream length $N$.

#### Error and Convergence (Finite Length $N$)

For a finite bitstream of length $N$, the Mean Squared Error (MSE) of the measured product $\hat{P}_Z$ is:

$$\text{Var}[\hat{P}_Z] = \frac{P_Z(1 - P_Z)}{N} = \frac{P_A P_B(1 - P_A P_B)}{N}$$

- **Theorem:** The accuracy convergence for finite-length bitstreams follows the **Inverse Length Law**, $\text{MSE} \propto O(1/N)$.

## 3. Stochastic Addition (MUX-Based)

#### Theorem 3.1: Correctness of Scaled Addition (MUX)

Let $A$ and $B$ be two independent stochastic bitstreams representing $P_A$ and $P_B$. Let $S$ be a third independent stream where $S_i \sim \text{Bernoulli}(P_S)$, with $P_S = 0.5$. The MUX operation, $Z = (S \wedge A) \vee (\neg S \wedge B)$, implements a scaled sum: $P_Z = P_S P_A + (1 - P_S) P_B$.

- **Expected Value Proof Sketch:**
  1. The output bit $Z_i$ is a disjoint union of two events: $(S_i \wedge A_i)$ or $(\neg S_i \wedge B_i)$.
  2. $\mathbb{P}(Z_i = 1) = \mathbb{P}(S_i \wedge A_i) + \mathbb{P}(\neg S_i \wedge B_i)$
  3. Assuming mutual independence of $S_i$, $A_i$, and $B_i$:

$$\mathbb{P}(Z_i = 1) = \mathbb{P}(S_i)\mathbb{P}(A_i) + \mathbb{P}(\neg S_i)\mathbb{P}(B_i)$$

$$\mathbb{P}(Z_i = 1) = P_S P_A + (1 - P_S) P_B$$

  4. Since $P_S = 0.5$, we get the desired scaled addition: $P_Z = \frac{P_A + P_B}{2}$.

#### Variance and Error Propagation

The output bit $Z_i$ also follows a Bernoulli distribution $\text{Bernoulli}(P_Z)$. Therefore, the variance of the measured sum $\hat{P}_Z$ is:

$$\text{Var}[\hat{P}_Z] = \frac{P_Z(1 - P_Z)}{N} = \frac{(\frac{P_A + P_B}{2})(1 - \frac{P_A + P_B}{2})}{N}$$

#### Unscaled Addition Limitations

Unscaled addition ($P_A + P_B$) cannot be implemented with simple logic gates while remaining restricted to the $[0, 1]$ range. A common approach is to use a counter/accumulator and a saturation function, which shifts the problem back to the binary domain or requires more complex, non-purely stochastic logic.

## 4. Error Analysis and Convergence

#### Concentration Inequalities (Question 3)

Since the bitstream count is the sum of i.i.d. Bernoulli random variables, concentration inequalities provide the tightest probabilistic error bounds for finite bitstreams of length $N$. These bounds quantify the probability of the empirical mean $\hat{P}_N$ deviating from the true mean $P$.

- **Hoeffding's Inequality (Tightest Bound):** For a bitstream $X$ of length $N$ representing $P_X$:

  $$\mathbb{P}(|\hat{P}_X - P_X| \geq \epsilon) \leq 2e^{-2N\epsilon^2}$$

  This inequality answers **Question 3** by providing the tightest theoretical upper bound on the probability of a deviation $\epsilon$ for a given length $N$. It is independent of $P_X$.

- **Application:** To guarantee $\epsilon$ accuracy, the required bitstream length $N$ can be calculated. For example, to ensure $\epsilon = 0.01$ with $99\%$ confidence ($\delta = 0.01$):

  $$N \geq \frac{1}{2\epsilon^2} \ln\left(\frac{2}{\delta}\right)$$

- **Central Limit Theorem (CLT) Application:** For large $N$, the empirical mean $\hat{P}_X$ is approximately normally distributed:

  $$\hat{P}_X \sim \mathcal{N}\left(P_X, \frac{P_X(1-P_X)}{N}\right)$$

  The CLT provides better error estimates near the true value but requires a large $N$ assumption, making Hoeffding's bound generally more robust for formal verification of *finite* systems.

#### Cumulative Error

Chained operations (e.g., $A \times B \times C$) cause the variance to accumulate. Since the variance scales as $O(1/N)$, the error accumulates linearly with the number of operations $k$.

- **Theorem:** If an operation chain has $k$ components, the total MSE of the result is $\text{MSE}_{total} \approx k \cdot \text{MSE}_{single\_op}$. This rapid error accumulation is a known limitation of SC.

#### Information-Theoretic Limits (Question 4)

There are no "information-theoretic limits" on *computable* accuracy beyond the $O(1/N)$ relationship established by the Central Limit Theorem and Hoeffding's bounds. The main limit is **time/latency**—accuracy scales linearly with time (length $N$), which is significantly slower than the exponential scaling ($O(2^{-B})$) of fixed-point binary systems, where $B$ is the bit width.

## 5. Bitstream Generation

### LFSR Mathematical Properties

An $n$-stage Linear Feedback Shift Register (LFSR) is mathematically modeled by a linear recurrence relation over the Galois Field $\mathbb{F}_2$.

- **Theorem 5.1: Maximum-Length Sequence ($m$-sequence):** An $n$-stage LFSR produces a maximum-length pseudo-random sequence (period $2^n - 1$) if and only if its characteristic polynomial $f(x)$ is **primitive** over the field $\mathbb{F}_2$.
- **Statistical Properties (Solomon Golomb's Postulates):** For $m$-sequences, these properties justify their use as practical sources of independence:
  1. **Balance Property:** In one cycle, the number of 1s is exactly $2^{n-1}$ and the number of 0s is $2^{n-1} - 1$. This ensures the empirical probability $\hat{P}_X$ is $\frac{2^{n-1}}{2^n-1} \approx 0.5$.
  2. **Run Property:** The lengths of consecutive 1s and 0s follow a near-perfect geometric distribution.
  3. **Autocorrelation Property:** The circular autocorrelation function is nearly a Dirac delta function. This proves that the sequence is highly uncorrelated with shifted copies of itself, which is vital when using multiple taps from a single LFSR to generate different, supposedly independent bitstreams.

## 6. Comparison and Conversion Operations

### Binary-to-Stochastic Conversion

The core method compares the target binary number $B$ (representing $P_B$) against a sequence of random numbers $R_i \sim \text{Uniform}[0, 1)$. The output bit $X_i = (B > R_i)$.

- **Theorem (Expected Value):** $\mathbb{P}(X_i = 1) = \mathbb{P}(B > R_i)$. Since $B$ is fixed and $R_i$ is uniform, this probability is exactly $B$. This conversion is statistically exact.
- **Bitstream-to-Binary Conversion:** This is simply calculating the empirical mean: $\hat{P}_B = \frac{1}{N}\sum_{i=1}^{N} X_i$. The accuracy is bounded by the concentration inequalities in Section 4.

### Comparison Operation

Stochastic comparison ($P_A > P_B$) is performed by bitwise subtraction and checking the sign bit of an accumulator, or by using a dedicated circuit called a stochastic comparator (XOR + counter/integrator). The output of an XOR gate, $Z = A \oplus B$, has an expected value $P_Z = P_A(1 - P_B) + P_B(1 - P_A)$. This value is minimized when $P_A = P_B$, allowing comparison via minimizing the XOR output's frequency.

## 7. Advanced Operations

| Operation | Stochastic Implementation | Expected Value Theorem | Theoretical Foundation |
|---|---|---|---|
| Division ($A/B$) | Successive approximation/ recursive feedback using multiplication and subtraction primitives. | Requires convergence proof of the feedback loop. | Geometric series and control theory principles. |

| | | | |
|---|---|---|---|
| Square Root ($\sqrt{A}$) | Iterative methods (e.g., Newton-Raphson) implemented with SC primitives. | Requires proof of convergence of the iterative process. | Standard numerical analysis on functions $f(x) = x^2 - A$. |
| Exponentiation | Approximated by cascades of multipliers (for powers $x^k$) or specialized circuits (for $e^x$). | Relies on the identity $e^x = \sum_{k=0}^{\infty} x^k/k!$. | Taylor series approximation and implementation of polynomial evaluation. |

## 8. Lean 4 Formalization Considerations

### Probability Space Construction (Question 5)

For verifying SC, the probability space $\Omega$ must be formally defined to contain the bitstreams.

1. **Idealized SC (Infinite Streams):**
   - **Space:** $\Omega = \{0, 1\}^{\mathbb{N}}$, the space of infinite binary sequences.
   - **Measure:** The product measure $\mu$ on $\Omega$, where each coordinate is a Bernoulli variable. This uses measure theory.
   - **Mathlib Module:** `Mathlib.Probability.Measure.Product`, `Mathlib.Probability.Independence`
2. **Practical SC (Finite Streams - Recommended):**
   - **Space:** $\Omega = \{0, 1\}^N$, the finite set of length $N$ bitstreams.
   - **Measure:** The discrete measure. This shifts the focus from "convergence" (SLLN) to "error bounds" (Hoeffding).
   - **Mathlib Module:** `Mathlib.Probability.Distributions.Bernoulli`, `Mathlib.Probability.IdentDistrib`, `Mathlib.Data.Fintype.Card`

### Data Representation Strategy

- **Input Values:** Represent $P_A$ and $P_B$ as `Real` numbers in $[0, 1]$.
- **Bitstreams:** Represent finite streams as `Fin N → Bool` or `Vector N Bool`.
- **Operators:** Define the logic gates (`\land`, `\lor`, `\neg`) as functions on the bitstream type.
- **Measured Value:** The empirical probability $\hat{P}_X$ is a function from the bitstream to a `Real` number.

### Proof Strategies (Question 6)

- **Expected Value Correctness (Ideal):** Use the definition of expected value (`measure.integral`) and properties of product measures (`measure_theory.measure.prod`).
- **Expected Value Correctness (Finite):** Use combinatorics and finite set cardinality, or the moment calculation theorem for the Bernoulli distribution.
- **Error Bounds (Finite):**
  - **Hoeffding/Chernoff:** These are formalized in Mathlib and are the most useful. Search for modules related to `hoeffding` or `chernoff` in the `Probability` namespace. You will need the version of the inequality applicable to sums of bounded random variables.
  - **Central Limit Theorem:** `Mathlib.Probability.Limit.Central` exists, but is less useful for

proving *guarantees* for a specific, finite $N$.

| Mathlib Component | Utility for SC Formalization |
| --- | --- |
| `Data.Real.Basic` | Representation of numeric values $P_X \in [0, 1]$. |
| `Probability.Distributions.Bernoulli` | Definition of the underlying probability distribution for bit generation. |
| `Probability.Independence` | Formalizing the assumption of independence for multiplication/addition inputs. |
| `Probability.Concentration.Hoeffding` | **Crucial** for proving practical accuracy bounds (Question 3). |
| `FieldTheory.Finite.Basic` | Needed for the LFSR mathematical proofs (primitive polynomials over $\mathbb{F}_2$). |

**Alternative Frameworks (Question 7)**

While **Measure Theory** provides the rigorous foundation for the infinite case (SLLN), the **Algebraic/Computable** framework (focused on $\mathbb{F}_2$ for LFSRs and finite probability spaces) is better suited for hardware verification and constructive Lean 4 proofs. Your focus should be on formalizing the **Hoeffding bound** on finite bitstreams, as this directly translates to a quantifiable guarantee for your FPGA system.

**Known Limitations / Open Problems**

- **Correlation Management:** Generating multiple, truly independent pseudo-random streams on an FPGA is expensive. The correlation between two LFSR outputs (even from different taps) is an open area of practical research, which limits the real-world accuracy guarantee.
- **Complex Functions:** Proving the convergence and bounds for advanced, iterative functions (like division or square root) is significantly more complex than the basic operations and often requires bounds on function derivatives (Lipschitz continuity) to track error propagation.

**Key Papers to Inform Lean 4 Proofs:**

- **SC Primer:** *Stochastic Computing: A Review* by *Alaghi* and *Hayes* (Modern survey).
- **Error Analysis:** *Stochastic Computing Systems* by *B. R. Gaines* (Original foundations, 1967).