Linear Feedback Shift Register (LFSR)

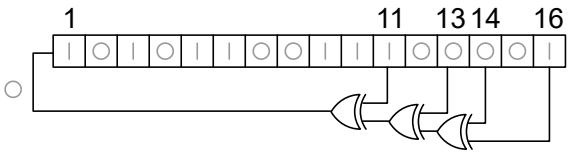
Definition

An LFSR is a shift register where the input bit is determined by a linear function of its current state. In digital logic (GF(2)), this linear function is strictly XOR.

Structure and Operation

The LFSR shifts bits at every clock cycle. The new bit entering the register is calculated by XORing specific bits currently in the register.

- Taps:
  These are the specific bit positions that are XORed together to determine the next state.
- Polynomial Representation:



  The tap configuration is often expressed as a "characteristic polynomial." For example, a 16-bit LFSR using taps at positions 16, 14, 13, and 11 is written as:

$$S(x) = x^{16} + x^{14} + x^{13} + x^{11} + 1$$

  The exponents correspond to the tap positions.

- Seed:
  This is the initial value loaded into the register. The LFSR must be initialized with a non-zero seed to operate (In XOR logic).

Properties

When designed with a "primitive polynomial," an LFSR exhibits properties ideal for pseudo-random number generation (PRNG):

- Maximum Length Period:
  For an n-bit register, the sequence will repeat only after $2^n - 1$ cycles. It cycles through every possible state except the all-zero state (which would cause a lock-up).
- Deterministic:
  While the output looks random, the sequence is deterministic. If you know the current state and the polynomial, you can predict the next bit.
- Distribution Balance:
  The output stream is nearly unbiased (50% probability). In a full period, there are exactly 2^{n-1} ones and $2^{n-1} - 1$ zeros.
- Run-Length Properties:
  The distribution of consecutive "runs" (consecutive 1s or 0s) follows a specific statistical pattern: 50% of runs are length 1, 25% are length 2, 12.5% are length 3, etc.

Generate pseudo-random number

Since the LFSR natively outputs a 1-bit stream with a ~50% uniform distribution, we use specific digital logic structures to transform this into multi-bit values or weighted probabilities.

1. Generating Multi-bit Values (Bit-Width)

To generate an N-bit random integer (e.g., an 8-bit byte or 32-bit word), two primary methods are used:

- **Serial Accumulation (High Quality):**
  Wait for N clock cycles, shifting the single output bit into a separate N-bit shift register.
  - *Pros:* The bits in the generated word are mathematically uncorrelated.
  - *Cons:* Throughput is lower (requires N cycles per word).
- **Parallel Slicing (High Speed):**
  Take N bits directly from the LFSR's internal state in a single clock cycle.
  - *Pros:* 1 word per clock cycle.
  - *Cons:* Consecutive words are highly correlated (they are just shifted versions of the previous state).

2. Tunable Probability (Weighted Randomness)

To generate a random bit that is **logic '1' with probability** P (where $P \neq 50\%$), we use the **Comparator Method**.

- **Structure:**
  1. Generate an N-bit uniform random integer (R) using the LFSR (as described above).
  2. Define a Threshold (T) such that $T = P \times 2^N$.
  3. Compare:

$$Output = \begin{cases} 1 & \text{if } R < T \\ 0 & \text{if } R \geq T \end{cases}$$

- **Precision (Resolution):**
  The "Probability Precision" is determined strictly by the **bit-width (N)** of the random integer used in the comparison.
  - **Low Precision:** Using 8 bits ($N = 8$) gives a resolution of $1/256 \approx 0.39\%$.
  - **High Precision:** Using 32 bits ($N = 32$) gives a resolution of $1/2^{32} \approx 2.3 \times 10^{-10}$.