# Philox Engine Correction Proposal

Ensuring Consistent Interpretation of the w Template Parameter


| | |
|---|---|
| Document #: | D0000R1 |
| Date: | 2025-04-26 |
| Project: | Programming Language C++ |
| Audience: | Library Evolution<br>Library |
| Revises: | D0000R0 |


| | |
|---|---|
| Reply-to: | Author Juan Lucas Rey<br><juanlucasrey@gmail.com> |


## 1 Introduction

Several random number engine classes in the C++ Standard Library make use of a `w` template parameter to define a "word size." In particular:

- mersenne_twister_engine
- subtract_with_carry_engine
- philox_engine

Each engine's documentation describes `w` as controlling the width, in bits, of the values it produces. In all three cases, the engine's `max()` member function returns a static value equal to $2^w - 1$.

However, in `philox_engine`, the current specification allows the generation of values greater than $2^w - 1$ under certain circumstances, violating the expected bound implied by `max()`.

## 2 Motivation and Problem Description

Consider the following instantiation of `philox_engine`:

```
using custom_philox = philox_engine<
    std::uint_fast32_t, 31, 4, 10,
    0xD2511F53, 0x9E3779B9, 0xCD9E8D57, 0xBB67AE85
>;
```

Here, `w = 31`, meaning that `max()` equals $2^{31} - 1$. Nevertheless, the engine, as currently specified, can produce values exceeding this limit, due to the size of the multiplier constants and the absence of modular reduction within the algorithm.

This inconsistency breaks the guarantees expected from the `w` parameter and can lead to surprising and erroneous behavior in client code relying on the specified output range.

## 3 Design Alternatives

Two solutions were considered:

### 3.1 A. Constrain multiplier constants (`M_k`) to fit within $[0, 2^w - 1]$

Require that each multiplier constant be no greater than $2^w - 1$. This solution would limit the possible choice of multipliers but ensure range compliance.

### 3.2 B. Apply modular reduction to products during computation

Modify the algorithm so that all products involving multipliers are reduced modulo $2^w$ immediately after multiplication. This maintains full generality in choosing constants, ensures correctness, and matches the behavior of similar engines.

**We recommend Solution B.**

## 4 Proposed Wording

The following changes are relative to the working draft of the C++ Standard, section [rand.eng.philox].

### 4.1 Modifications to `philox_engine` description

**Change** the description of the computation in `philox_engine::operator()` (from its current form) **to the following**:

> Each round consists of a sequence of calculations, where for each $k$ in $[0, n)$:
>
> — Compute $z_k = (M_k \times x_{k'}) \bmod 2^w$, where $x_{k'}$ denotes a selected component of the state.
> — Update each state component according to the specified permutation and addition of constants.
>
> All intermediate and final results are reduced modulo $2^w$.

### 4.2 Modifications to `philox_engine` remarks

**Add** a new remark:

> [Note: Applying modular reduction after each multiplication guarantees that all generated values are within the range $[0, 2^w - 1]$, as implied by `max()`. — end note]

## 5 Impact on Existing Implementations

Implementations conforming to the current wording but not applying modular reduction may observe behavioral changes for certain engine instantiations with $w < 32$ (or $w < 64$, depending on `UIntType`). These changes are deemed desirable to ensure compliance with the Standard's stated bounds.

No existing usage of standard Philox configurations (e.g., `philox4x32`) is affected.

## 6 Acknowledgements

Thanks to discussions within Library Evolution Working Group members for feedback and insights.

## 7 References

— Random Number Generation Standardization Papers, N3551, N3552
— Philox Design: Random123 library documentation