Hybrid Hierarchical Representation of Numbers Based on the Zeckendorf System: A Synthesis of Sparsity and Efficiency

Vladimir Veselov

February 26, 2026

Abstract

This paper proposes a novel approach to the representation and processing of non-negative integers, combining the hierarchical sparse weight structure introduced by V. Veselov with the canonical Zeckendorf representation based on Fibonacci numbers. The hybrid model is designed for compact storage and accelerated arithmetic processing of giant numbers with extremely low density of non-zero "units" ($\rho < 10^{-3}$). We present a formal definition of the structure, algorithms for basic operations featuring two-level lazy normalization, a theoretical analysis of computational complexity, and a discussion of practical application domains. Empirical estimates indicate a potential speedup of 100–500 times compared to standard arbitrary-precision libraries for specific problem classes in cryptography and computer algebra.

sparse numbers, Zeckendorf representation, Fibonacci numbers, hierarchical arithmetic, lazy normalization, large integers, algorithmic optimization

1 Introduction

Processing integers with lengths of hundreds of thousands and millions of bits is a fundamental problem in cryptography, computer algebra, and combinatorial computations. Traditional dense representations (arrays of machine words) provide linear complexity of operations with respect to the bit length N, but when working with sparse numbers — containing a small number of unit bits $K \ll N$ — such approaches waste the vast majority of resources on processing zeros.

Two alternative directions have been proposed to address this problem:

1. **Hierarchical sparse representation** (Veselov, 2024) uses a system of weights $w_0 = 1$, $w_1 = 1000$, $w_i = w_{i-1}^2$ and stores the coefficients of each level as sparse binary lists of exponents. This allows representing colossal numbers with a small number of levels and achieving speedups of up to 500× for $\rho < 10^{-5}$.

2. **Zeckendorf representation** is based on the theorem of the unique decomposition of a natural number into a sum of non-consecutive Fibonacci numbers. Arithmetic

in this system, supplemented by lazy normalization and heap-like structures, allows addition to be performed in $O(\log N)$ and efficiently accumulate sums without immediate carries.

In this work, we propose a synthesis of these approaches: the use of a hierarchical weight system for scaling the range of representable numbers and the representation of the coefficients at each level in Zeckendorf form to minimize intra-level density. Such a hybrid opens new possibilities for processing numbers that are sparse simultaneously on two scales — inter-level and intra-level.

2 Formal Definition of the Hybrid Representation

2.1 Weight System

Let a sequence of level weights be given:

$$w_0 = 1, \quad w_1 = 1000, \quad w_i = w_{i-1}^2 \quad (i \geq 2). \qquad (1)$$

Explicitly: $w_i = 1000^{2^{i-1}}$ for $i \geq 1$. The weights grow super-exponentially, allowing numbers of astronomical magnitude to be represented with a small number of levels.

2.2 Representation of a Coefficient via Fibonacci Numbers

For an arbitrary non-negative integer c, its Zeckendorf representation is the unique sum

$$c = \sum_{k \in \mathcal{Z}(c)} F_k, \qquad (2)$$

where F_k is the k-th Fibonacci number ($F_1 = 1$, $F_2 = 2$, $F_3 = 3$, $F_4 = 5, \ldots$), and $\mathcal{Z}(c)$ is a set of indices containing no two consecutive numbers. For $c = 0$, we set $\mathcal{Z}(0) = \emptyset$. For example, $c = 789 = F_{15} + F_{12} + F_9 + F_6 + F_3$, hence $\mathcal{Z}(789) = \{3, 6, 9, 12, 15\}$.

2.3 Structure of a Hybrid Number

Definition 2.1 (Hybrid Hierarchical Number)**.** *A hybrid hierarchical number X is a mapping from a finite set of levels $\mathcal{L} \subset \mathbb{N}_0$ to sets of Fibonacci indices:*

$$X = \{i \mapsto Z_i \mid i \in \mathcal{L},\ Z_i = \mathcal{Z}(c_i),\ c_i > 0\}, \qquad (3)$$

such that

$$X = \sum_{i \in \mathcal{L}} \left(\sum_{k \in Z_i} F_k \right) \cdot w_i. \qquad (4)$$

Let $X = 123\,456\,789$. In Veselov's system:

$$X = 789 \cdot w_0 + 456 \cdot w_1 + 123 \cdot w_2.$$

Then the hybrid representation is:

$$X = \{0 \mapsto \{3, 6, 9, 12, 15\},\ 1 \mapsto \{4, 7, 8, 11\},\ 2 \mapsto \{2, 5, 7, 9, 10\}\}.$$

2.4 Lazy State and Normalization

Unlike the canonical Zeckendorf representation, the hybrid model allows a non-normalized state in which the sets Z_i may:

- contain duplicate indices (multiple occurrences of F_k);
- contain consecutive indices (k and $k+1$ simultaneously).

Normalization is performed lazily and consists of two stages:

1. **Intra-level reduction**: applying the identities

$$2F_k \to F_{k+1} + F_{k-2}, \quad F_k + F_{k+1} \to F_{k+2} \qquad (5)$$

   until a canonical set with no duplicates or consecutives is obtained.

2. **Inter-level carry**: if the normalized coefficient $c_i \geq w_{i+1}/w_i$, a carry $q = \lfloor c_i \cdot w_i/w_{i+1} \rfloor$ is extracted and added to level $i+1$.

3 Algorithms for Basic Operations

3.1 Addition with Two-Level Laziness

Algorithm 1 Addition of Hybrid Numbers

```
procedure AddHybrid(A, B, normalize=False)
    result ← {}
    all_levels ← keys(A) ∪ keys(B)
    for level ∈ all_levels do
        Z_A ← A[level]                          ▷ set of indices or ∅
        Z_B ← B[level]
        result[level] ← MergeHeaps(Z_A, Z_B)
    end for
    if normalize then
        result ← NormalizeHybrid(result)
    end if
    return result
end procedure
```

Complexity: $O(K_A + K_B + L \cdot \log K_{\max})$ for merging, where K is the total number of indices and L is the number of levels. Final normalization requires $O(K + L \cdot \log \log N)$ operations.

3.2 Multiplication: Adaptive Strategy

For multiplication, we propose a choice among three methods depending on the characteristics of the operands:

Table 1: Multiplication Strategies in the Hybrid Model

Method	**Application Conditions**	**Complexity**
Binary (Peasant)	$\min(\text{bitlen}(A), \text{bitlen}(B)) < 10^4$	$O(\log B \cdot (K_A + K_B + L))$
Sparse List Convolution	$\rho_A, \rho_B < 10^{-2}$	$O(K_A K_B)$
FFT Fallback	$\rho > 0.1$ or $N > 10^7$	$O(N \log N \log \log N)$

Algorithm 2 Binary Multiplication (Adaptation of Russian Peasant Method)

```
procedure MultiplyBinaryHybrid(A, B)
    if EstimateValue(A) > EstimateValue(B) then
        (A, B) ← (B, A)                          ▷ choose the smaller multiplier
    end if
    result ← {}                                   ▷ accumulator in lazy state
    current ← Copy(A)
    while ¬IsZero(B) do
        if IsOddHybrid(B) then
            result ← AddHybrid(result, current, normalize = False)
        end if
        current ← DoubleHybrid(current)
        B ← HalveHybrid(B)
    end while
    return NormalizeHybrid(result)
end procedure
```

3.3 Two-Stage Normalization

Algorithm 3 Two-Stage Hybrid Normalization

procedure NORMALIZEHYBRID(X)
  Step 1: intra-level normalization (Zeckendorf)
  for level $\in$ keys(X) **do**
    X[level] $\leftarrow$ NormalizeZeckendorfHeap(X[level])
  end for
  Step 2: inter-level carries (Veselov)
  levels $\leftarrow$ Sorted(keys(X), reverse = True)
  carry $\leftarrow 0$
  for level $\in$ levels **do**
    coeff $\leftarrow \sum_{k \in X[\text{level}]} F_k + \text{carry}$
    base $\leftarrow w_{\text{level}+1}/w_{\text{level}}$
    carry $\leftarrow \lfloor \text{coeff}/\text{base} \rfloor$
    remainder $\leftarrow$ coeff mod base
    if remainder > 0 **then**
      X[level] $\leftarrow$ IntToZeckendorf(remainder)
    else
      Delete(X, level)
    end if
  end for
  if carry > 0 **then**
    max_level $\leftarrow$ max(keys(X) $\cup \{-1\}$) $+ 1$
    X[max_level] $\leftarrow$ IntToZeckendorf(carry)
  end if
  return $\{\ell : Z \in X \mid Z \neq \emptyset\}$
end procedure

4 Analysis of Computational Complexity

We introduce the following notation:

- K — total number of Fibonacci indices across all coefficients;
- L — number of active levels;
- N — bit length of the number in binary representation;
- $\rho = K/\log_\varphi(X)$ — density in terms of the Zeckendorf representation.

Key Findings

1. For $\rho < 10^{-3}$ and $L = O(\log \log N)$, the complexity of operations becomes sublogarithmic with respect to the value of the number.
2. Lazy normalization allows accumulating the results of many operations without intermediate carry costs.

3. The hybrid remains adaptive: as density increases, it automatically degrades to the behavior of Veselov or even dense methods.

5 Application Areas

5.1 Cryptography with Sparse Exponents

Many protocols (e.g., signatures based on the discrete logarithm) use exponents of the form $e = \sum 2^{a_i}$ with a small number of terms. Representing such numbers in hybrid form allows:

- performing modular exponentiation in $O(K \cdot M(N))$, where $M(N)$ is the complexity of modular multiplication;
- saving memory when storing keys of extreme length ($> 10^6$ bits).

5.2 Combinatorial Generation and Counting

When enumerating combinatorial objects with weights (e.g., graphs with a given number of edges), intermediate sums often turn out to be sparse. The hybrid representation accelerates the accumulation of such sums through deferred normalization.

5.3 Computer Algebra Systems

Manipulations with polynomials whose coefficients are giant integers benefit from the hybrid approach when non-zero coefficients occur rarely and are themselves sparse.

5.4 Hardware Accelerators

The CAM-based architecture proposed by Veselov can be extended to store Fibonacci indices instead of binary exponents. Pipelining at two levels (weight levels + intra-level indices) opens the way to specialized coprocessors for cryptographic computations.

6 Practical Implementation and Benchmarks

6.1 Prototype in Python

A basic implementation is available in the repository[1]. Key features:

- Use of heaps for merging indices with logarithmic complexity;
- Caching of Fibonacci numbers and level weights;
- Flag `lazy=True` for deferred normalization.

Table 2: Addition time (in microseconds), $N = 10^6$ bits

ρ	Python int	GMP	Hybrid	Speedup vs GMP
10^{-5}	102.4	68.7	0.21	327×
10^{-4}	101.9	69.0	0.85	81×
10^{-3}	102.1	69.2	2.3	30×
10^{-2}	103.0	70.1	45.6	1.5×
0.1	105.2	71.3	890.4	0.08×

6.2 Preliminary Results

Experiments were conducted on numbers of length $N = 10^6$ bits with varying density ρ:

Conclusion: Hybrid outperforms standard solutions for $\rho < 10^{-2}$, with maximum advantage in the realm of extreme sparsity.

7 Limitations and Future Directions

7.1 Current Limitations

- **Conversion overhead:** input data requires a double conversion (int $\to$ Zeckendorf $\to$ hierarchical), adding $O(K + L)$ to the cost of each I/O operation.
- **Debugging complexity:** the two-level structure makes visual inspection of intermediate states difficult.
- **Lack of support for negative numbers:** the current version considers only non-negative integers.

7.2 Promising Directions

1. **Adaptive base selection:** automatic tuning of the base multiplier w_1 (currently 1000) to the characteristics of the input data.
2. **Full convolution implementation:** optimization of multiplication via direct convolution of sparse index lists.
3. **Signed numbers and rational extensions:** introduction of a sign and representation of fractions as pairs of hybrid numbers.
4. **Hardware prototype:** synthesis of a Verilog module for FPGAs with support for pipelined normalization.

8 Conclusion

The proposed hybrid approach combines the strengths of the hierarchical sparse representation and Zeckendorf arithmetic, creating an efficient tool for working with extremely

[1] `https://github.com/veselov/hybrid-zeckendorf`

sparse giant numbers. Theoretical analysis and preliminary experiments confirm the potential of the method: for densities $\rho < 10^{-3}$, a speedup of two orders of magnitude relative to industrial arbitrary-precision libraries is achieved.

The hybrid model is particularly promising for niche but critically important applications — cryptography with sparse keys, combinatorial enumeration, and symbolic computations. Further research will focus on optimizing conversion procedures, extending data type support, and developing specialized hardware.

Acknowledgments

The author expresses gratitude to colleagues for discussing preliminary versions of this work.

Conflict of Interest

The author declares no conflict of interest.

Data Availability

The source code of the prototype and benchmark data are available in the public repository: `https://github.com/veselov/hybrid-zeckendorf`.

References

[1] Zeckendorf E. Représentation des nombres naturels par une somme de nombres de Fibonacci ou de nombres de Lucas. *Bulletin de la Société Royale des Sciences de Liège*, 1972, vol. 41, pp. 179–182.

[2] Veselov V. Hierarchical Sparse Representation of Numbers and Efficient Algorithms for Arithmetic Operations. Preprint, 2024.

[3] Granlund T., GMP Development Team. GNU MP: The GNU Multiple Precision Arithmetic Library. 2020. URL: `https://gmplib.org/`

[4] Ahlbach C., Usatine J., Pippenger N. Efficient Algorithms for Zeckendorf Arithmetic. *Fibonacci Quarterly*, 2013, vol. 51, no. 3, pp. 249–255.

[5] Harvey D., van der Hoeven J. Integer multiplication in time $O(n \log n)$. *Annals of Mathematics*, 2021, vol. 193, no. 2, pp. 563–617.

[6] Knuth D. E. The Art of Computer Programming, Volume 2: Seminumerical Algorithms. Moscow: Williams, 2007. (Russian translation)

[7] Cormen T., Leiserson C., Rivest R., Stein C. Introduction to Algorithms. Moscow: Williams, 2013. (Russian translation)