Geometric Constraint Circuits for Symbolic Prime Growth

Abstract

We reformulate the symbolic lattice framework for iterated prime growth as a *non-uniform, constraint-based computational model* that explicitly trades spatial resources for evaluation time. Arithmetic is not executed at runtime; instead, it is *compiled into structure.* Once constructed, the system supports constant- or near-constant-time evaluation, yielding what we call a *free second read.* This framework does not contradict SETH or standard lower-bound conjectures, as it operates via non-uniform compilation and super-polynomial spatial resources. Rather, it provides a principled account of how geometric, optical, or hardware-based systems achieve fast evaluation through precomputation.

1 Axioms and Motivation

1.1 Axioms

We make the computational assumptions of the framework explicit.

Axiom 1 (Non-Uniform Compilation). For each bound N, a one-time construction of a structure C_N is permitted. The size of C_N may be super-polynomial in $\log N$. The construction cost is not charged to individual queries.

Axiom 2 (Compiled Arithmetic). All arithmetic relations relevant to the task are embedded into the static structure of C_N. No arithmetic operations are performed at evaluation time.

Axiom 3 (Evaluation by Propagation). Evaluation consists solely of bounded-depth signal propagation or constraint satisfaction within C_N.

Axiom 4 (Amortized Queries). The structure C_N may be queried arbitrarily many times after construction.

2 Motivation

Classical algorithmic models treat time as the dominant scarce resource. In contrast, physical and engineered systems routinely trade space, structure, and fabrication cost for speed. Examples include ASICs, ROM tables, optical correlators, and neural accelerators. In all such systems, arithmetic is not repeatedly executed; it is embedded into structure.

The original symbolic lattice framework models the asymptotic growth of iterated prime indices without explicit arithmetic evaluation. In this paper, we reinterpret that framework as a *compiler*: symbolic growth rules are translated into geometric or circuit-level constraints. Evaluation then reduces to signal propagation or passive measurement.

We compile arithmetic into structure and trade space for time.

This principle underlies the entire construction. No computational work is eliminated; rather, it is displaced from runtime into one-time construction.

3 Non-Uniform Compilation Model

3.1 Circuit Families

For each bound N, we define a circuit or geometric structure C_N with the following properties:

- **Input:** An encoding of an integer $n \leq N$
- **Output:** A predicate (e.g., prime/composite or symbolic growth layer)
- **Depth:** $O(1)$ or polylogarithmic
- **Size:** Super-polynomial or exponential in $\log N$

The family C_N is *non-uniform*: each circuit is constructed with full knowledge of the bound N. This places the model squarely within standard circuit complexity and advice-based computation.

3.2 Space–Time Tradeoff

Evaluation time is constant or near-constant, while spatial resources scale rapidly with N. This is the classical tradeoff exploited by hardware acceleration and does not violate known lower bounds.

4 Constraint-Based Primality Filtering

4.1 From Decision to Filtering

Instead of deciding primality algorithmically, we eliminate composites via constraints. A number n is composite if and only if there exists a divisor $1 < d \leq \sqrt{n}$. Each potential divisor corresponds to a constraint channel. Activation of any channel destabilizes the configuration, signaling compositeness.

Primality is identified with global stability under all constraints.

4.2 Geometric Interpretation

Constraints may be realized geometrically:

- Shapes encode candidate numbers
- Alignment, overlap, or interference encodes divisibility
- Failure propagates physically (leaks, collapses, shadow overlap)

No arithmetic occurs during evaluation; all arithmetic is compiled into the geometry.

5 A Toy Optical Shadow Model

Consider an optical system in which:

- A shape encoding n is illuminated
- A fixed mask encodes all divisor constraints up to $\sqrt{N}$
- Shadows interfere if and only if a divisor exists

If light reaches the detector, the number is prime; if blocked, it is composite. This is equivalent to a depth-1 Boolean circuit with many parallel gates. The shadow is merely signal propagation; the computation occurred at construction.

6 The Free Second Read

We now make the space–time tradeoff precise.

[Compilation–Evaluation Tradeoff] Let C_N be a non-uniform family of structures of depth $d(N)$ constructed with full knowledge of N. If all arithmetic relations required to answer queries on inputs $n \leq N$ are embedded into C_N, then each query can be evaluated in time $O(d(N))$, independent of the construction cost.

Proof. By construction, all arithmetic relations are statically encoded in C_N. Evaluation consists only of propagation through a structure of bounded depth $d(N)$. The size and construction cost of C_N do not affect the evaluation path length. Hence each query runs in time $O(d(N))$. $\square$

Once the structure C_N is built, evaluation does not depend on the cost of construction. The same input may be queried repeatedly with negligible additional cost. This is the *free second read*:

- The first "read" pays the full construction cost
- All subsequent reads reuse the compiled structure

Formally, this corresponds to amortized constant-time queries after non-uniform preprocessing.

Once the structure C_N is built, evaluation does not depend on the cost of construction. The same input may be queried repeatedly with negligible additional cost. This is the *free second read*:

- The first "read" pays the full construction cost
- All subsequent reads reuse the compiled structure

Formally, this corresponds to amortized constant-time queries after non-uniform preprocessing.
This phenomenon is well known in hardware systems but is often obscured in purely algorithmic discussions.

7 Relation to the Symbolic Lattice Framework

The symbolic lattice can be reinterpreted as a resource diagram:

- Vertical layers correspond to compilation depth
- Horizontal growth corresponds to spatial resource scaling
- Renormalization absorbs local irregularities into fixed constraints

Symbolic iteration $T(x) = x \log x$ describes how required resources scale across layers, not how long computation runs.

8 Limits and Non-Claims

This framework:

- Does not provide a uniform algorithm for unbounded inputs
- Does not yield polynomial-space solutions
- Does not contradict SETH or ETH

The construction cost grows super-polynomially, and the model is explicitly non-uniform. SETH already allows such advice-based and circuit-family constructions.

9 Connections to Hardware Accelerators

The model aligns with existing systems:

- ASICs: arithmetic compiled into silicon
- Optical correlators: pattern matching via interference
- Neural accelerators: learned weights as structure
- ROM tables: precomputed answers

All achieve speed by trading space and construction for time.

10 Conclusion

Symbolic iteration does not bypass computation; it reorganizes it. By compiling arithmetic into structure, we obtain fast evaluation at the cost of large, non-uniform construction. The resulting free second read is not a violation of lower bounds but a direct consequence of space–time tradeoffs long exploited in physical computation.