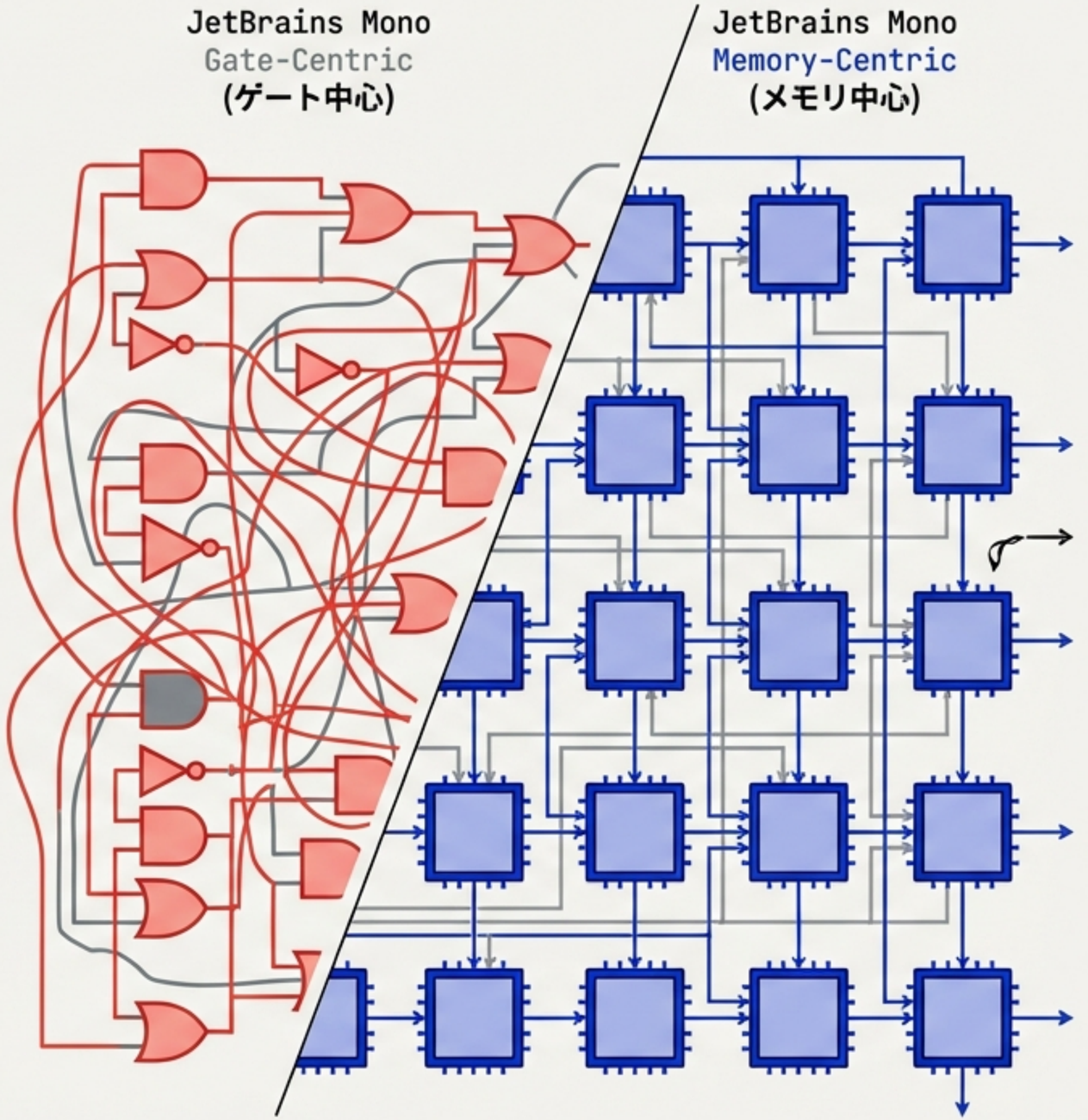
メモリを用いた論理合成：理論から応用へ

Logic Synthesis Using Memory: From Theory to Application

高レベルな関数分解理論とFPGAアーキテクチャの融合



KEY CONCEPTS:

- **パラダイムシフト:** 論理関数を「ゲートの集合」ではなく「メモリ（LUT）のネットワーク」として捉える。
- **目的:** 現代のFPGAが持つ膨大な組み込みメモリを最大限に活用し、高性能・低消費電力な回路を実現する。

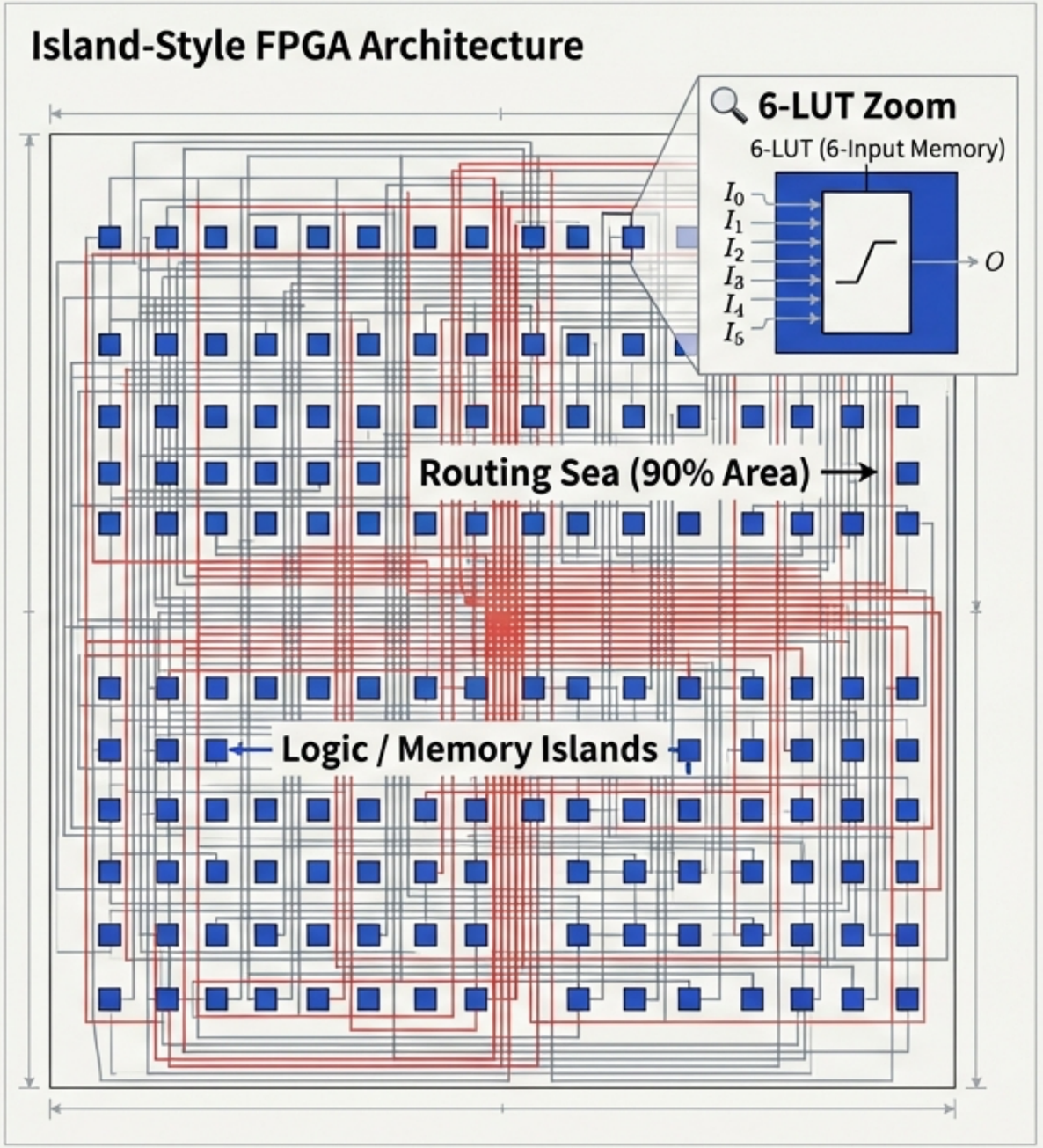
Based on the work of Prof. Tsutomu Sasao (Meiji University)

FPGAアーキテクチャの現実と課題

- **VLSI設計の現状:**
  微細化に伴う設計コストと時間の増大。
  プログラム可能なLSI（FPGA）が主流解となる。

- **FPGAの構造:**
  - **LUT (Look-Up Table):** 実態は**6入力メモリ**（現在は $K = 6$ が標準）。
  - **配線リソース:** チップ面積の9割以上を占める。**配線遅延が性能のボトルネック。**

- **合成の課題:**
  従来のゲートレベル合成ではなく、LUT（メモリ）の特性を活かし、配線を**最小化する「関数分解」が必要不可欠。**



数学的基盤：関数分解

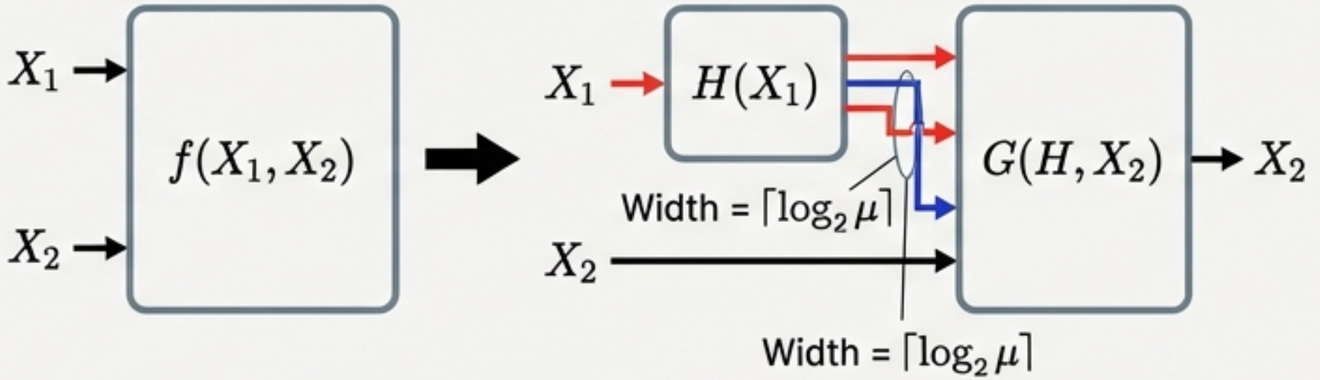
Mathematical Foundation: Function Decomposition

- **基本原理:** 大規模な論理関数 f を、より小さなブロック G と H に分割する。
- **束縛変数 (X_1) vs 自由変数 (X_2):** 入力変数 X を二つのセットに分割。
- **重要指標: 列複雑度** (Column Multiplicity μ)。分解表における「異なる列パターン」の数が回路規模を決定する。
- **接続線数:** ブロック間の配線数は $\lceil \log_2 \mu \rceil$ で決まる。

分解表 (Decomposition Chart)

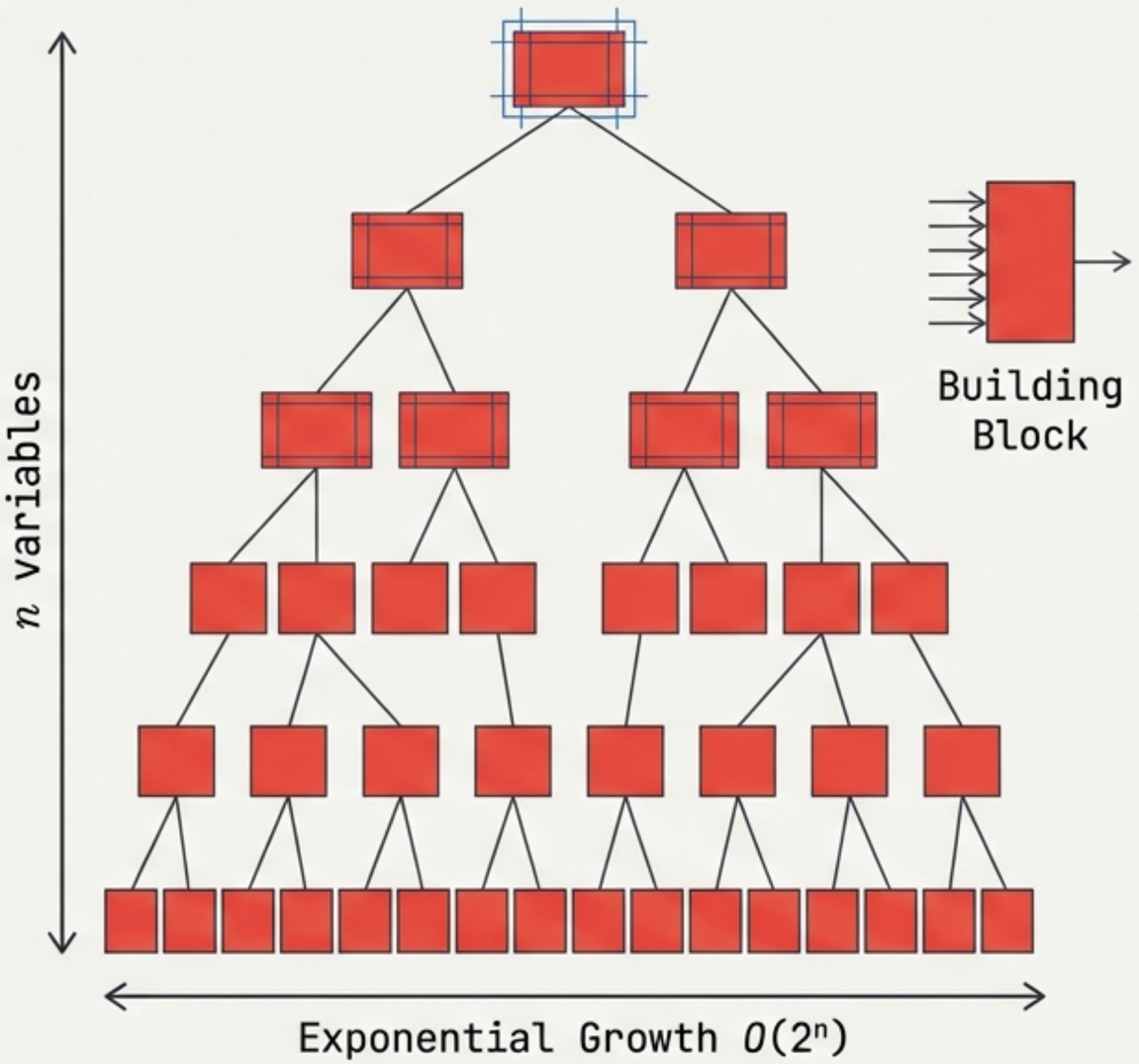
X_2 \ X_1	00	01	10	11
00	0	1	1	0
01	1	0	0	1
10	0	1	1	0
11	1	1	0	1

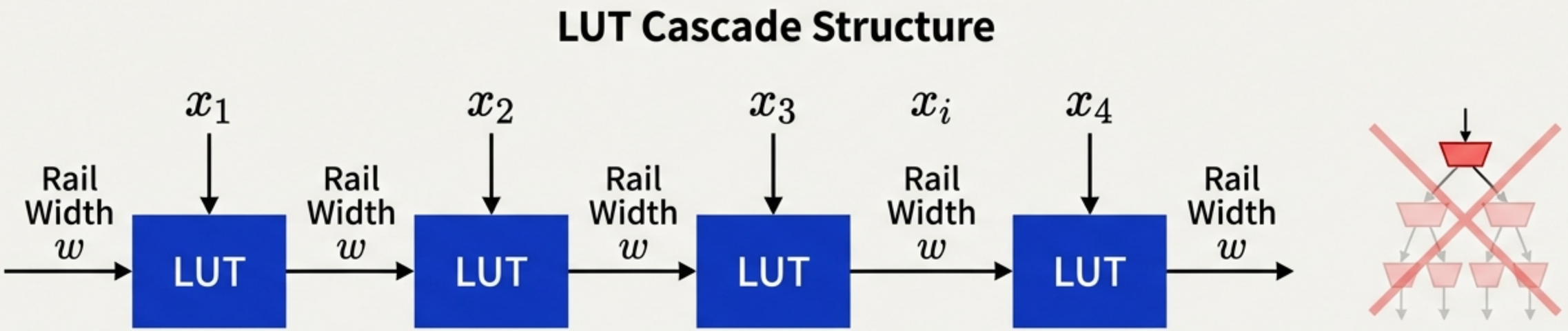
μ: Number of unique column patterns.
($\mu = 2$ in this example)



万能合成法：マルチプレクサ (MUX) による実現

- **シャノン展開**: 任意の関数はMUXの木構造で実現可能。
- **コストの壁**: 必要なハードウェア量は変数の数 n に対して指数関数的 ($O(2^n)$) に増加。
- **具体的限界 (Theorem 4.2.3)**: 6-LUTを用いた場合、n 変数関数の実現には約 $(2^{n-4}-1)/3$ 個のLUTが必要。
- **結論**: ランダムな論理には有効だが、大規模回路には非効率。





効率的合成法：LUTカスケード

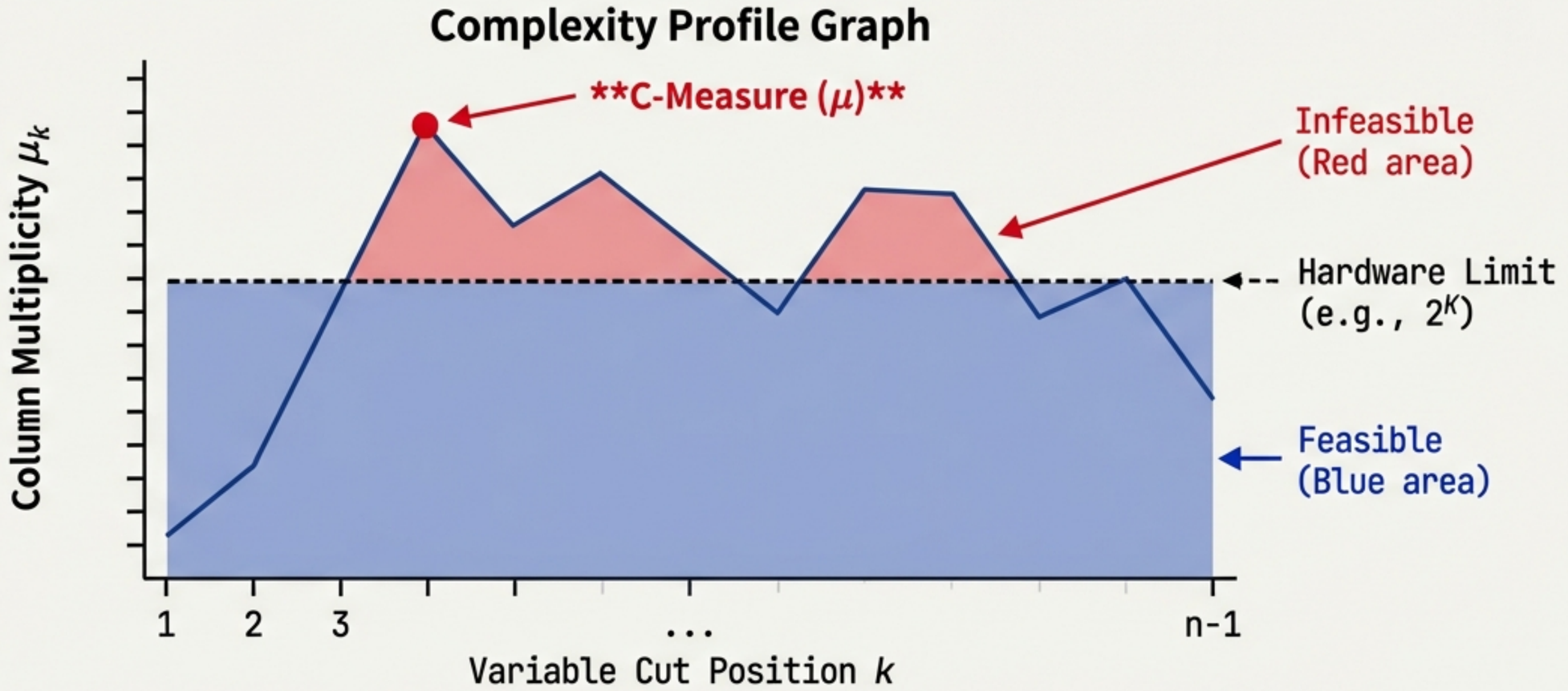
Efficient Synthesis: The LUT Cascade

- **Key Attributes**
  - **構造**: LUT（メモリ）を一列に接続。配線が局所的で規則的。
  - **スケーラビリティ**: 条件を満たす関数ならば、LUT数は変数 n に対して線形（$O(n)$）に比例。
  - **レイル数**（Rail Width）w:
    - $w = \lceil \log_2 \mu \rceil$
    - この w がFPGAのLUT入力数 K（通常6）に収まるかが鍵。

複雑度の尺度：C尺度 (The C-Measure)

Definition: C尺度 (μ): 入力変数の順序を固定した際、あらゆる分割位置 k における列複雑度 μ_k の最大値。
数式: $\mu(f) = \max_k (\mu_k)$

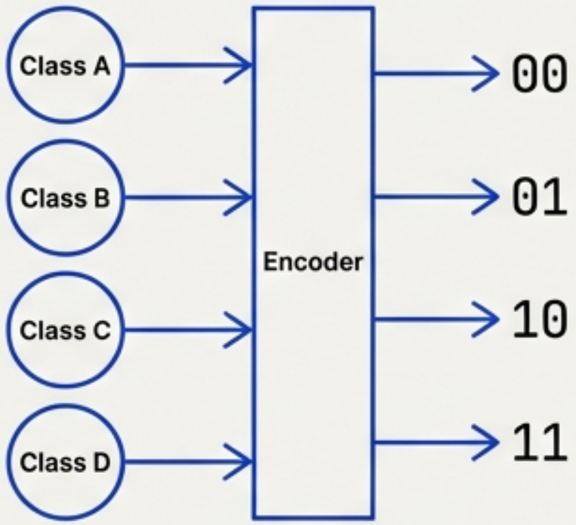
Criterion: μ が小さい → LUTカスケードで実現可能 (Feasible).
μ が大きい → 他の手法が必要 (Infeasible).



最適化技術：符号化法の工夫
Optimization: Smart Encoding Techniques

分離的符号化（Standard）

各同値類に一意のコードを割り当てる（Safe but Redundant）。

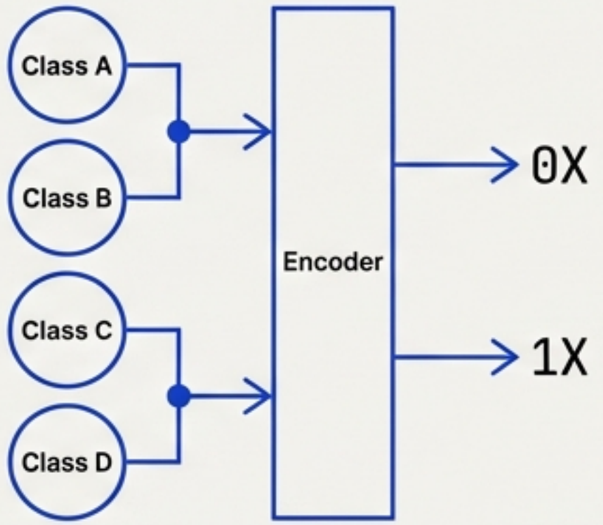


Result: 2 Wires Required.
Rail Width = 2

Compression

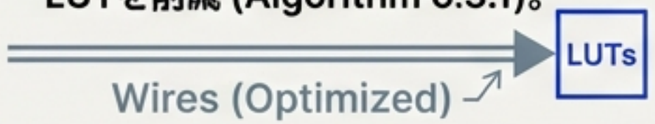
非分離的符号化（Smart）

異なる同値類でコードを共有する。



Result: Reduced Wires / LUTs.
Rail Width < 2

効果: 中間変数を入力変数そのものに置換し、
LUTを削減 (Algorithm 6.3.1)。

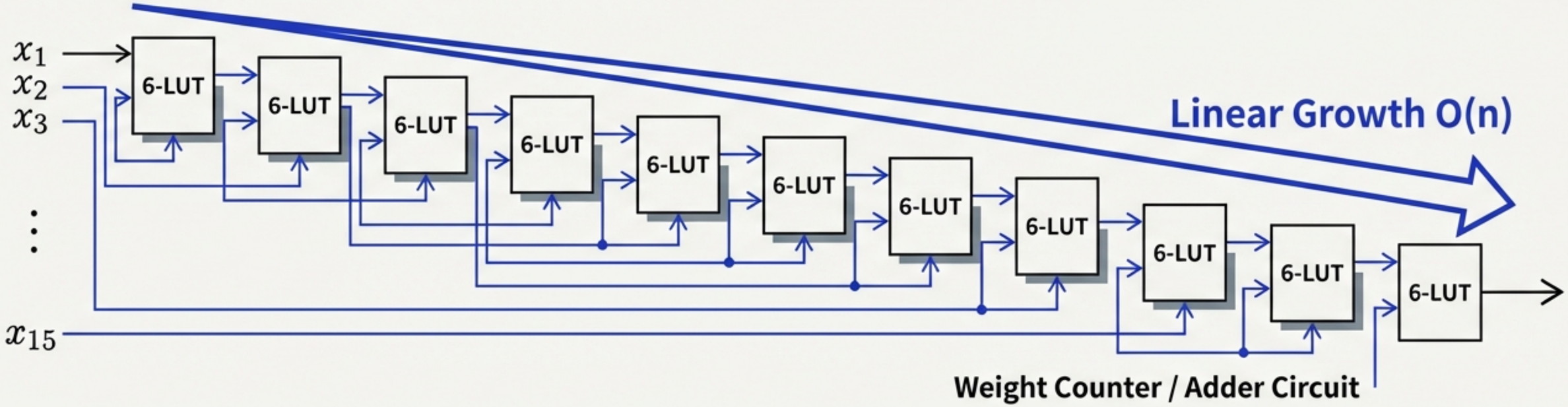


実現容易な関数 I：対称関数
Feasible Functions I: Symmetric Functions

Definition: 変数を置換しても出力が変わらない関数（例：多数決関数、パリティ、加算）。

C-Measure Property: $\mu(f) \leqq n+1$ (Lemma 5.3.1)
複雑度は n に対して線形にしか増えない。

Result: 15変数の対称関数 → わずか13個の6-LUTで実現可能。

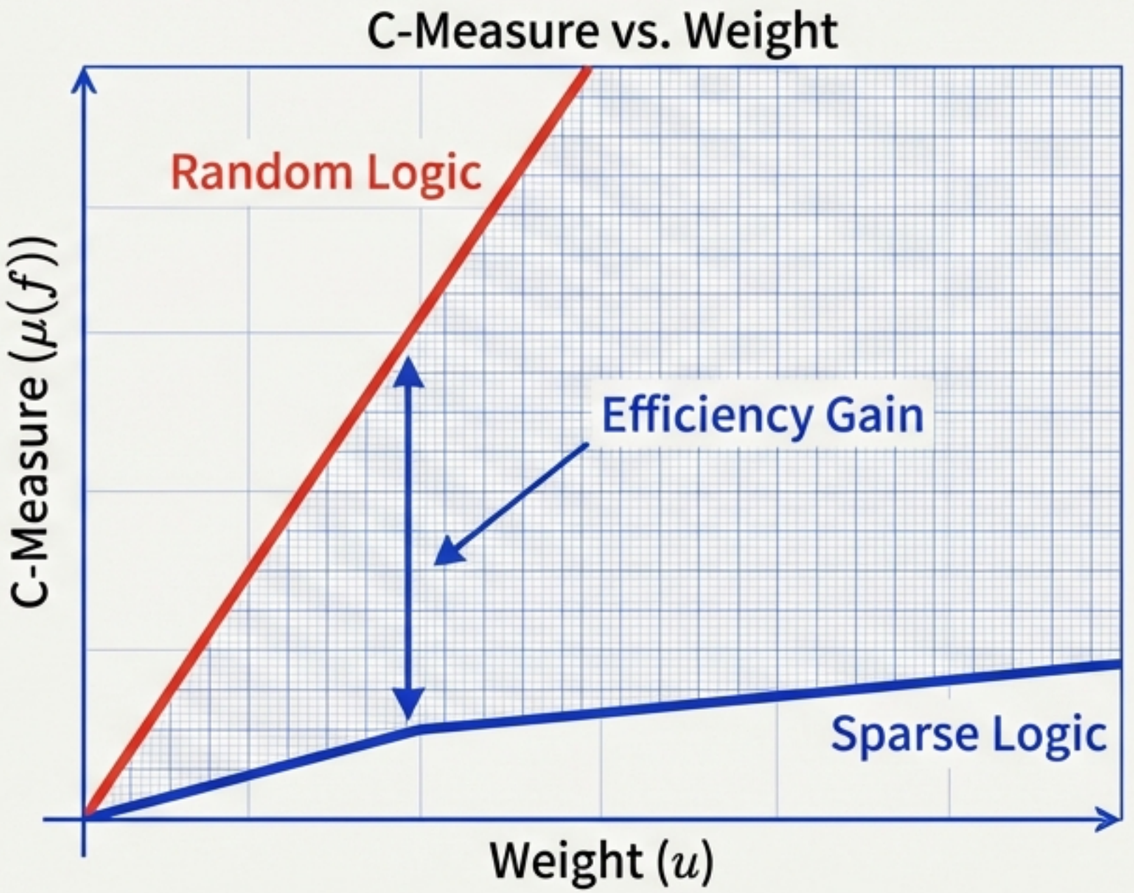


実現容易な関数 II：疎な関数
Feasible Functions II: Sparse Functions

Definition: 真理値表において、出力が '1' となる入力パターン（重み u）が極めて少ない関数。
Bound: $\mu(f) \leq u + 1$
Implication: ランダムな論理は困難だが、実世界の「特定パターン検出」はカスケードに最適。

Input Patterns	Output (f)
00...00	0
00...01	0
00...00	0
00...01	1
00...10	0
00...01	0
00...10	0
00...11	0
00...10	1
01...01	0
01...10	0
10...01	0
10...10	0
10...01	0
10...10	0
11...01	0
11...10	0
11...11	0

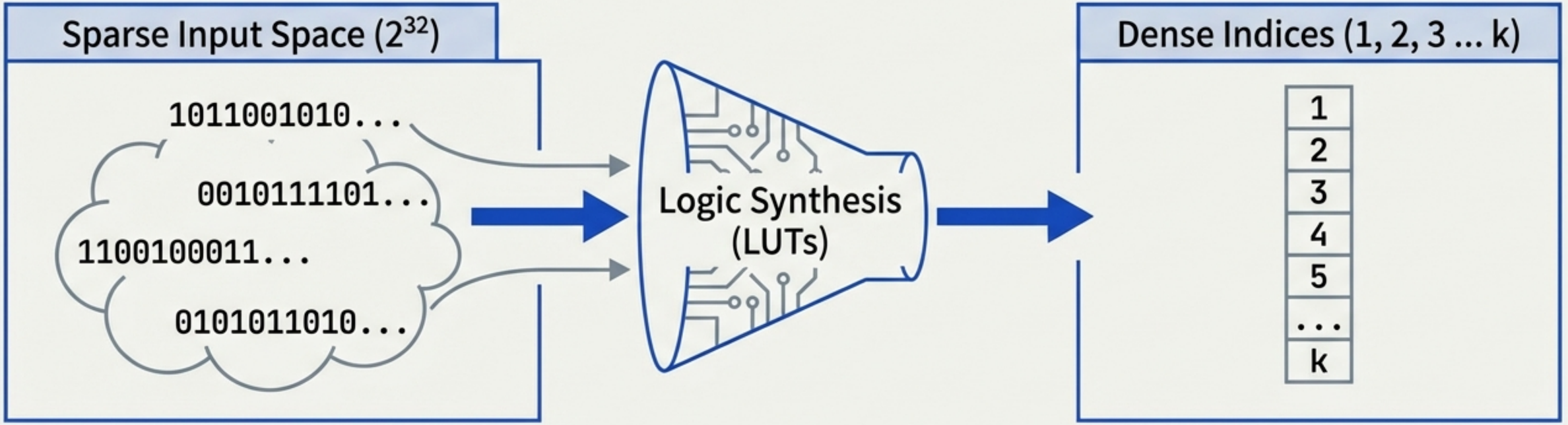
Sparse Distribution



キラーアプリ：インデックス生成関数

The Killer App: Index Generation Functions

Problem: 疎なベクトル集合（IPアドレス、ユーザID）を効率的に管理したい。
Solution: 登録ベクトルを「論理関数」として扱い、メモリ合成する。
Definition: 入力が登録ベクトルと一致すればインデックスを出力、不一致なら0。



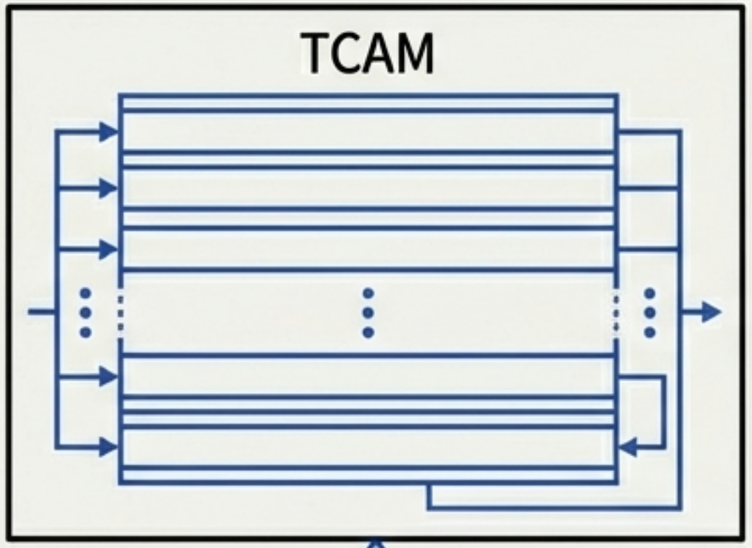
From Sparse Data to Dense Logic.

応用事例 I：IPルーティング（LPM）

Helvetiica Neue Bold： Noto Sans JP Bold

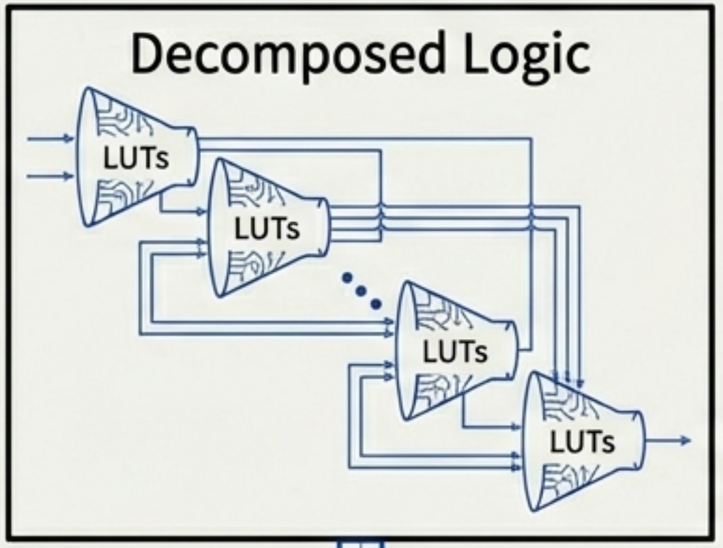
インターネットの核心技術：Longest Prefix Match (LPM)。

従来 (Traditional TCAM)



Power: High
電力消費大、
ビット単価高。

提案 (Logic Synthesis)



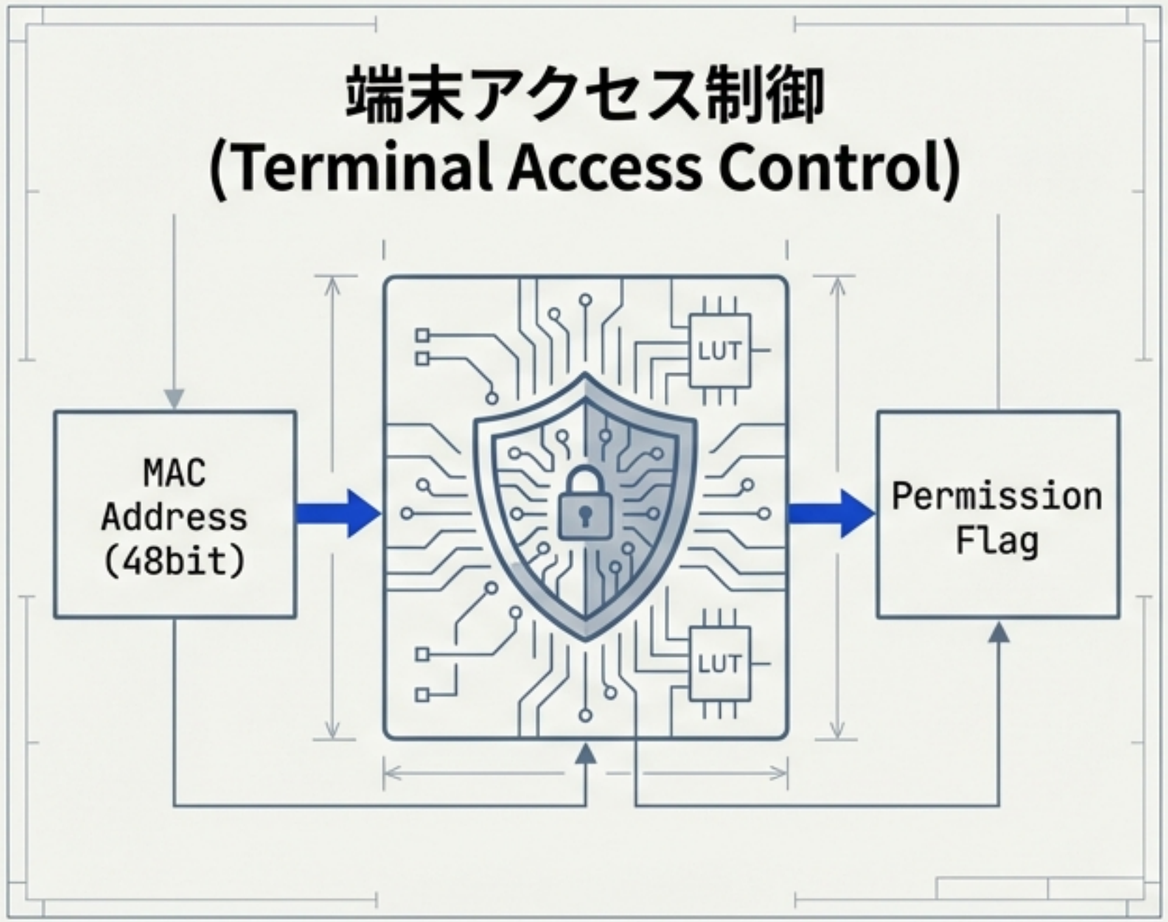
Power: Low
FPGAメモリを活用。
低消費電力、高速。

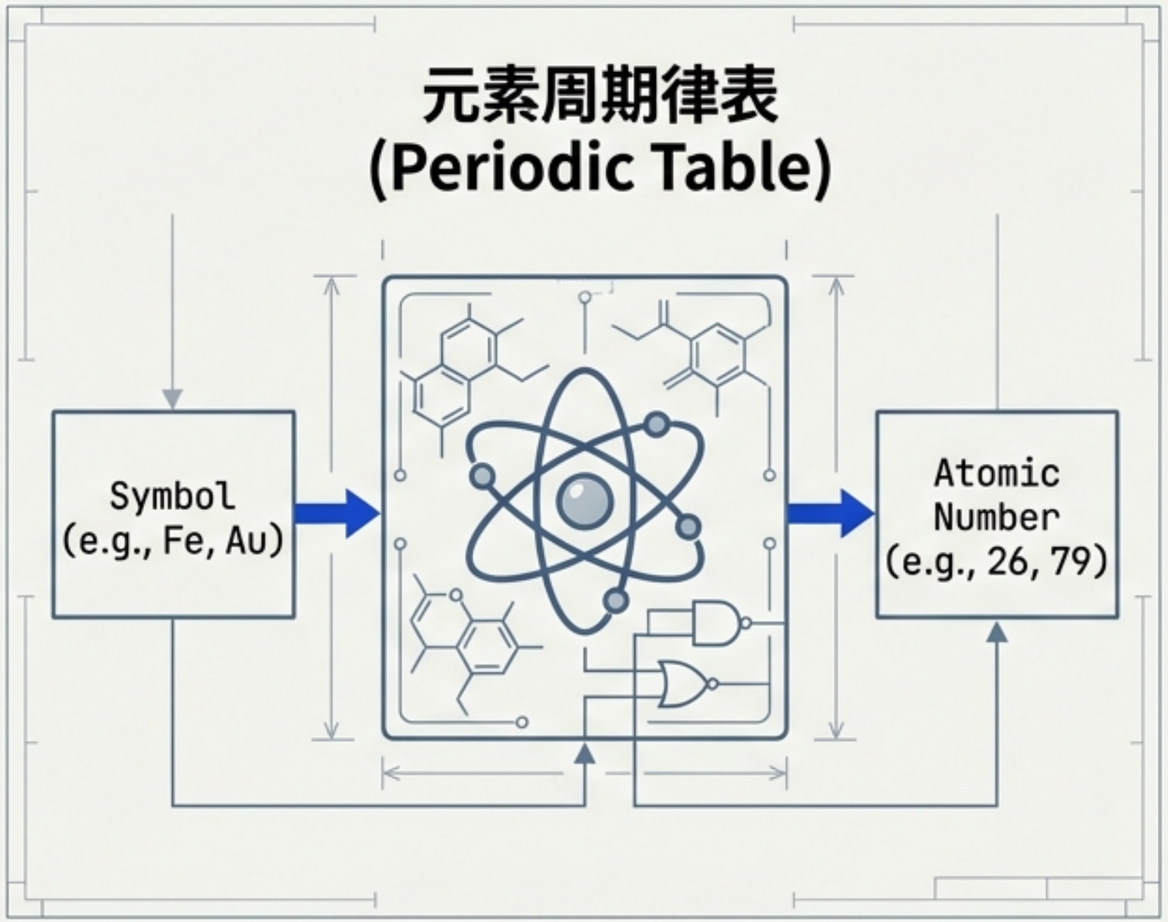
LPM関数のC尺度は $\mu \leq k+1$ (Theorem 7.4.1)。

Internet Backbone

応用事例 II：セキュリティと科学データベース

これらは数学的には同一の「インデックス生成問題」である。

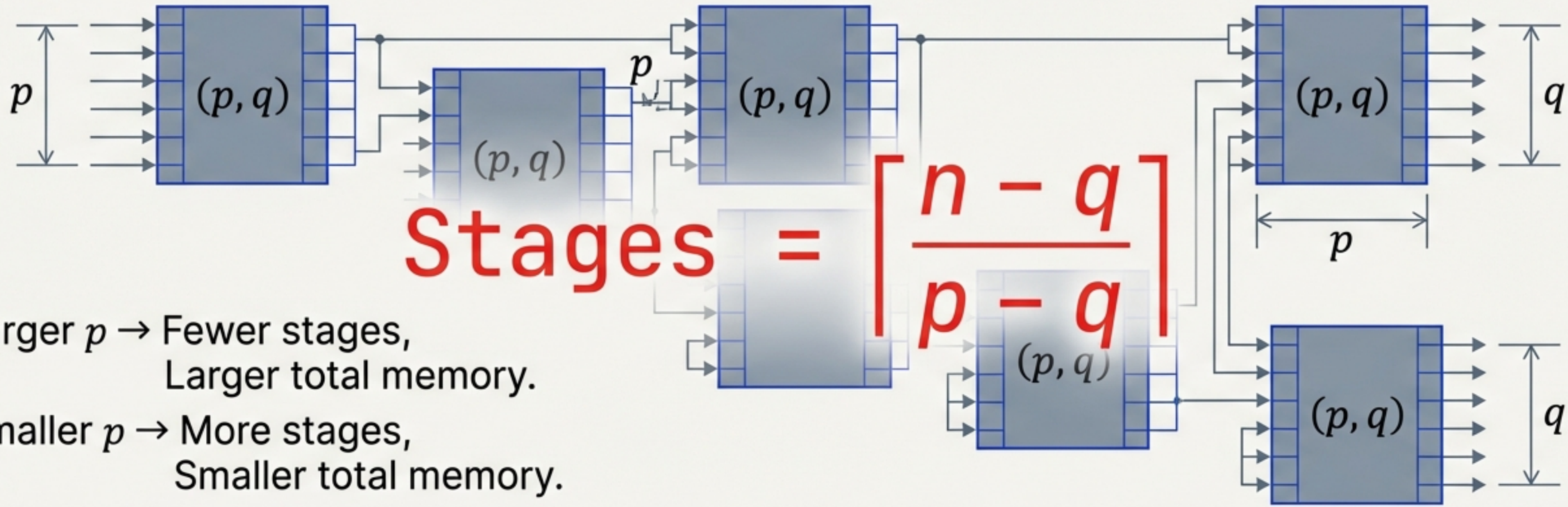




実装戦略：(p, q) 素子の利用

Ink Black Helvetica Neue Bold
Noto Sans JP Bold

(p, q) 素子: p-input, q-output memory block.



$$\text{Stages} = \left\lceil \frac{n - q}{p - q} \right\rceil$$

Larger p → Fewer stages,
Larger total memory.

Smaller p → More stages,
Smaller total memory.

Design Space Trade-off

Speed (Stages) ⟷ Area (Memory Size)

← Trade-off →

定量的評価と理論的限界

Ink Black Helvetica Neue Bold
Noto Sans JP Bold

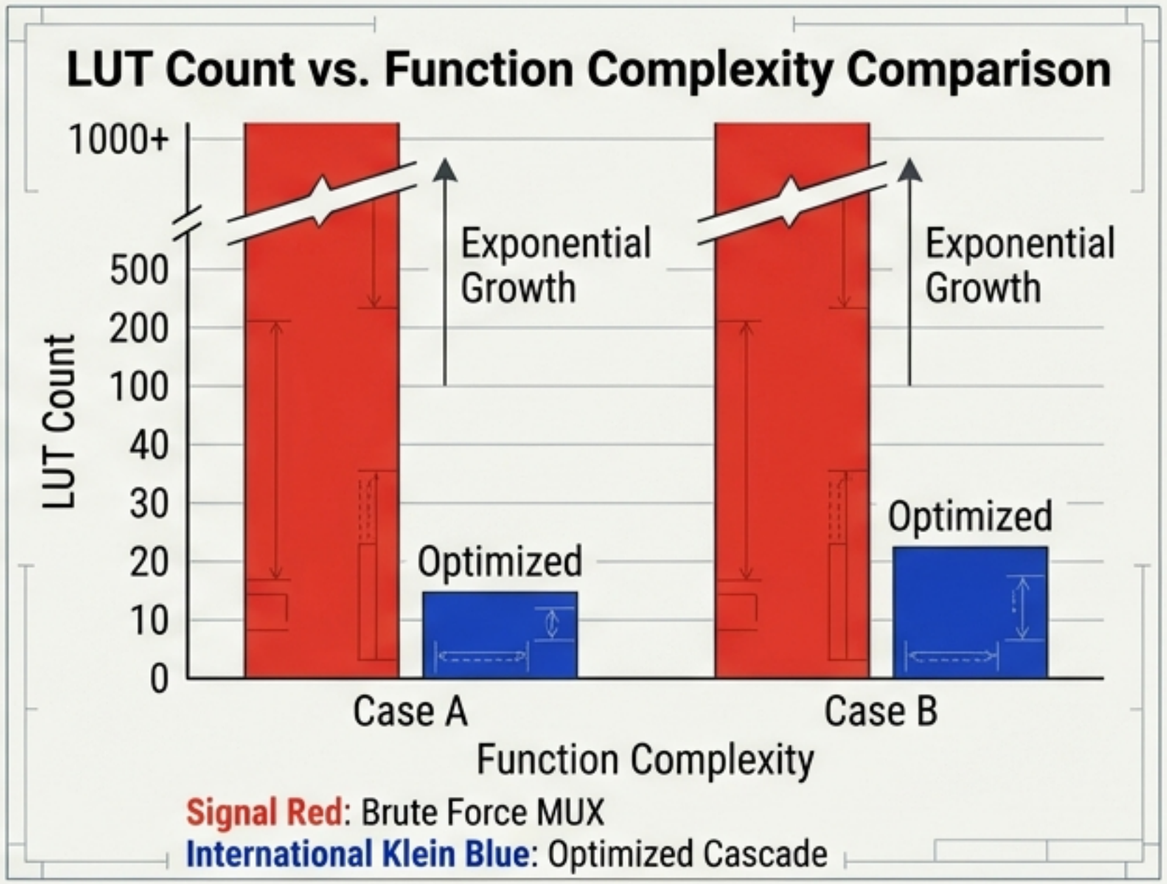
Quantitative Results & Theoretical Bounds

Case A:
$n = 10$,
Weight $u \leq 47$

Required 6-LUTs:
< 15

Case B:
$n = 12$,
Weight $u \leq 95$

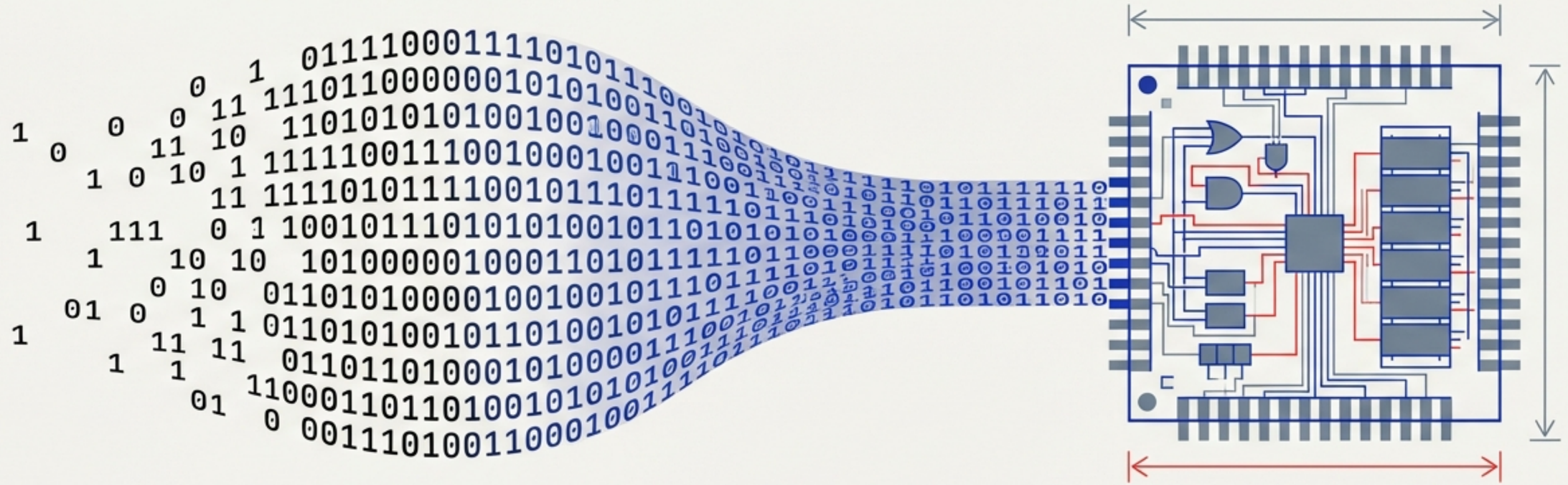
Required 7-LUTs:
< 23



疎な関数であれば、入力数が多くても現実的なハードウェア量で収まることが数学的に保証されている。

結論：メモリベース論理合成の未来の未来

- **データの論理化:** データストレージの問題 → 論理最小化の問題。
- **C尺度の重要性:** 実現可能性を判断する羅針盤。
- **次世代設計へ:** 大規模FPGAにおける高速・低消費電力システムの鍵。



“Memory is Logic. Logic is Memory.”