

Yoshan: Algebraic-Circuit Co-Synthesis and Search Methodology of Hardware-Efficient S-Boxes

Robert Hanhan, Binyamin Alony and Itamar Levi

Bar-Ilan University, Ramat Gan, Israel

{robert.hanhan, binyamin.alony, itamar.levi}@biu.ac.il

Abstract. Choosing an S-box for a hardware-oriented cipher means balancing crypt-analytic quality (differential uniformity δ , linearity L , algebraic degree), circuit complexity (the number of AND gates and the AND depth D), and implementation cost (silicon area and, in masked designs, registers and fresh randomness). Under Boolean masking the non-linear gates dominate this cost and D sets the latency, so automated S-box searches have concentrated on quadratic permutations, which admit depth-1 circuits but make the algebraic degree of a cipher grow slowly.

We present YOSHAN, a co-synthesis system that optimizes an S-box and its gate-level circuit together instead of fixing the lookup table first. It combines structural algebraic seeding (subfield towers, Feistel, Misty, and product-round Feistel constructions, affine wrappers, CCZ graph automorphisms, bilinear and shear seeds), cryptographically guided evolution of the circuit DAG by algebraic rewrites, a differentiable Piling-Up relaxation with Hungarian LSAP projection, and multi-level circuit optimization (SMT sub-DAG shaving, stage-preserving common-subexpression elimination, exact depth-1 SAT synthesis, and algebraic-structural inverse compilation) inside a quality-diversity archive across $n \in \{5, 6, 7, 8\}$. Every archived circuit is exported gate for gate to SystemVerilog and certified by exhaustive simulation and a Z3 miter. The archive holds 31,129 certified netlists covering 56 $(n, \delta, L, \text{deg})$ profiles; three extended-affine invariants and an exact affine-equivalence solver split them into members of known classes (Category A) and permutations provably not extended-affine equivalent to any power map or to the cipher S-boxes we compare against (Category B, 15 benchmark classes $\text{YSH}_{n,i}$, 13 with stratified subspace-filtration degree spectra). All 30 reported S-boxes and their explicit inverses are synthesized gate-exactly in 45 nm CMOS, unmasked and as pipelined first-order DOM modules, and every benchmark circuit is traced to its seed and the search stage that shaped it. Results include a gate-exact AES-equivalent circuit at the optimal AND depth 3 (42 AND, 101 XOR) with an explicit depth-3 inverse (42 AND, 106 XOR), a 6-bit degree-5 permutation $\text{YSH}_{6,5}$ ($\delta = 6$, $L = 24$, boomerang uniformity $\beta = 14$) with 12 AND gates, a 5-bit quartic permutation $\text{YSH}_{5,4}$ with 5 AND gates at depth 2 in both directions, and 8-bit degree-7 permutations with 28 and 18 AND gates. All forward and inverse circuits are compared with the best published implementations.

Keywords: Substitution Boxes · Multiplicative Complexity · Algebraic Degree · Quality-Diversity Search · Hardware Synthesis · Masking

1 Introduction

Symmetric-key block ciphers, hash functions, cryptographic permutations, and authenticated encryption schemes rely on Substitution Boxes (S-boxes) $S : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ as their

primary source of non-linearity [Sha49]. A strong S-box combines low differential uniformity (δ) [BS91] and low linearity (L) [Mat94] with high algebraic degree ($\deg(S)$) to resist higher-order differential [Lai94, Knu95], division-property [Tod15], zero-sum [BCDC11], and cube attacks [DS09].

In side-channel-protected hardware or bitsliced software, an S-box cannot be evaluated via its lookup table alone. Under t -th order Boolean masking schemes such as Threshold Implementations (TI) [NRR06], Domain-Oriented Masking (DOM) [GMK16], and Hardware-Private Circuits (HPC) [CGLS21, FGDP⁺18], every sensitive bit is split into $d \geq t + 1$ shares. Linear gates (XOR, NOT) act share by share without randomness or registers, whereas each 2-input AND gate requires d^2 cross-share products refreshed with random bits and separated from the next non-linear layer by a register stage [GMK16, DCBG⁺17, LBS19]. The AND count thus governs the area, randomness, and power of a masked S-box, and its **multiplicative depth** D sets its latency in clock cycles (Section 5.2).

1.1 The Search Problem and the Algebraic-Degree Compromise

Optimizing cryptographic quality ($\delta, L, \deg$) and masked hardware cost (AND, D) jointly faces three obstacles:

1. **Combinatorial explosion.** While 4-bit permutations form 302 affine equivalence classes [LP07], widths $n = 5$ ($32! \approx 2.6 \times 10^{35}$) to $n = 8$ ($256! \approx 8.6 \times 10^{506}$) defy exhaustive enumeration, and bijections have vanishing density $2^n!/(2^n)^{2^n}$ among all n -bit Boolean maps.
2. **Fixed-target circuit minimization.** Existing minimizers synthesize a fixed lookup table or circuit via SAT [Sto16], A* multiplicative-complexity search [JBK24], whole-S-box Boyar–Peralta heuristics [JBKK25], segmented low-depth search [KBJ⁺26], or top-down polynomial division [BDM25, BDM26] (whose division tables scale as $2^{\binom{n}{5} + \binom{n}{4}}$ and stop at degree 3 for $n = 7$ [BDM26, p. 619]), while unified SAT co-synthesis of cryptographic properties and gate area is restricted to $n = 4$ [JCL⁺25]. A fixed-target tool cannot trade the lookup table for a cheaper permutation with the same or better profile when its equivalence class resists shallow factorization.
3. **The quadratic compromise.** To keep $D = 1$, automated searches [BDMD⁺20, BDM25] focus on quadratic permutations, whose slow degree growth forces high round counts (24 rounds in KECCAK- f [BDPVA11] and 12 in ASCON [DEMS21], versus 5–7 rounds in SPEEDY [LMMR21] with a degree-5 S-box). We therefore target *low-depth, low-AND circuits for higher-degree S-boxes* ($\deg \in \{3, \dots, 7\}$) that *reduce total masked latency* ($\text{rounds} \times D$) *and energy*.

1.2 Contributions

We introduce **Yoshan**, a co-synthesis system that unites constructive algebra, evolutionary DAG rewriting, continuous gradient relaxation, and formal synthesis to co-evolve an S-box S and its Boolean circuit DAG G :

1. **Algebraic-circuit co-synthesis (Section 4).** Search episodes are seeded from published circuits, fractional subfield towers, or structural constructions (Feistel, Misty, and product-round Feistel networks, sparse affine wrappers, linear/CCZ graph automorphisms, bilinear and shear seeds), then evolved by cryptographically guided DAG rewrites (AND-tree rotations, distributive folding, AND erasers, macro-mutations) and by a differentiable Piling-Up relaxation with Hungarian LSAP projection, inside a MAP-Elites archive [MC15] compressed by SMT cut-6 sub-DAG shaving, Paar CSE [Paa97], Boyar–Peralta wire reuse [BP10], and depth-1 SAT re-synthesis.

2. **Certified export, inverse synthesis, and 45 nm silicon evaluation (Sections 5, 6.3, and 7).** Every archived DAG is emitted gate for gate to SystemVerilog and certified by exhaustive 2^n -input simulation and a Z3 miter [dMB08]. For all 30 S-boxes of Table 2 we synthesize explicit inverse circuits through a four-tier algebraic/structural pipeline, print all 60 lookup tables and netlists (Supplementary Material (SM) C), map the 60 circuits 1-to-1 onto the NanGate 45 nm library in Yosys, unmasked and as D -stage pipelined $d = 2$ DOM-indep modules, and prove post-synthesis gate equivalence and timing in Berkeley ABC.
3. **Exact affine-equivalence classification (Section 6.1).** Three extended-affine (EA) invariants and an exact affine-equivalence solver in the style of Biryukov et al. [BDCBP03] partition the archive into known classes (**Category A**: 1,761 of 1,763 permutations carry certified (A, a, B, b) witnesses, the remaining two form a 6-bit χ_3 -Feistel cubic involution class with the forward invariants of x^{13}) and permutations provably not EA-equivalent to any catalogue entry (**Category B**, whose 15 benchmark classes are denoted $\text{YSH}_{n,i}$). Boomerang uniformity β separates two structural families within Category B: non-Feistel subspace-partition S-boxes with $\beta \in [6, 20]$ and Feistel-type S-boxes with a probability-one boomerang.
4. **Benchmarking across $n \in \{5, 6, 7, 8\}$ (Section 6).** Across 31,129 certified netlists in 56 profiles, co-synthesis complements fixed-target minimizers: an AES-class circuit at the optimal depth $D = 3$ (42/101/4 forward, 42/106/4 inverse, equal to FIPS-197 AES up to a wire permutation and a gate-exact realization of the depth-3 inverter architecture of Hadžić and Bloem [HB25]) that, as a pipelined DOM-indep module, saves 66/60 forward/inverse flip-flops and 11.5%/8.9% area at 25% lower latency than our $D = 4$ FIPS-197 circuit; $\text{YSH}_{6,5}$, a 6-bit degree-5 S-box with $(\delta, L, \beta) = (6, 24, 14)$ at 12 AND gates, strictly improving on SPEEDY’s (8, 24, 16) at its own 12-AND cost; the 5-bit quartic $\text{YSH}_{5,4}$ with 5 AND gates at $D = 2$ in both directions; 9-AND 6-bit quartic and cubic S-boxes with symmetric forward/inverse cost; and 8-bit degree-7 S-boxes $\text{YSH}_{8,2}$ and $\text{YSH}_{8,3}$ with 28 and 18 AND gates. Table 10 traces every benchmark circuit from Stage 1 through Stage 4.

2 Related Work

YOSHAN connects fixed-target circuit minimization, SAT and metaheuristic S-box search, and equivalence classification.

2.1 Circuit Minimization for a Fixed S-box

Fixed-target synthesis minimizes a given S-box S for multiplicative complexity (AND count), linear-layer (XOR/XNOR) count, combinational depth (LTP), or AND depth $D \geq \lceil \log_2 \deg(S) \rceil$ [BDM26]. Canright [Can05] decomposed the $\mathbb{F}_{2^8}$ AES inverter over $\mathbb{F}_{((2^2)^2)^2}$; Boyar and Peralta combined MC minimization with linear heuristics (RNBP/BPD) to reach 32 AND gates [BP10] and depth 16 [BP12], refined in [ME19]. Stoffelen [Sto16] used SAT, Testa et al. [TSADM19] used cut-based XAG rewriting, Rasoolzadeh [Ras22] optimized gate latency, Bilgin et al. [BDMD⁺20] targeted low-AND-depth and shared S/S^{-1} circuits, and Jeon, Baek, and Kim [JBK24] used A* graph search to find optimal-MC circuits for $n \leq 5$ and selected 6-bit S-boxes. Recently, Jeon et al. [JBKK25] extended Boyar-Peralta heuristics to whole S-boxes (ERNBP/EBPD) by reconstructing node-level XOR expressions over non-linear gate outputs and applying non-linear input transformations at invariant $(\#NL, D)$, optimizing 8-bit AES (32/81–32/97 at $D \in \{5, 6\}$, $LTP \in [17, 26]$; 33/85 at $LTP = 23$ and 34/81–34/100 at $D = 4$, $LTP \in [15, 28]$), SNOW3G S_Q , and Saturnin’s 16-bit super S-box. Kim et al. [KB⁺26] further introduced SLICE, combining bitsliced subcircuit exploration over 2-fanin $\{\text{AND}, \text{OR}, \text{NAND}, \text{NOR}\}$ ($\#NL$) and

$\{\text{XOR}, \text{XNOR}\}$ ($\#L$) gates with bit-width segmenting, two-stage boundary merging, and refined EBPd to set low-depth records on AES (LTP $\in \{12, 13\}$ at $D = 3$, LTP = 14 at $D = 4$), 5-bit ASCON (LTP = 4, 5/11, $D = 1$), Dillon APN, and 7-bit x^3 (Table 4).

At minimum AND depth $D = \lceil \log_2 \deg(S) \rceil$, Bolzer, Duval, and Minier [BDM25, BDM26] give the state-of-the-art (AND, XOR) reference via depth-1 quadratic synthesis [BDM25] and top-down polynomial division $P = Q_1 Q_2 + R$ [BDM26] across 5-bit ($\deg \leq 4$), 6-bit ($\deg \leq 5$), 7-bit ($\deg = 3$), and 8-bit AES (33/84, $D = 4$) benchmarks, noting two limits [BDM26, Sections 4.1.2, 4.2, 6, 7.2]: precomputation grows as $2^{\binom{n}{5} + \binom{n}{4}}$ while factorable quadratics drop from ≈ 0.50 to ≈ 0.15 at $n = 7$ (restricting $n = 7$ to $\deg \leq 3$), and 6-bit inversion costs 29/109 at $D = 3$ by division versus 21/63 by tower fields. Hadžić and Bloem [HB25] give the first depth-3 $\mathbb{F}_{2^8}$ inverter ($9 + 21 + 12 = 42$ AND) with a 139-gate AES S-box. While fixed-target division, EBPd, and SLICE achieve smaller linear layers or gate counts on several fixed classical targets, YOSHAN scales to $n = 7, \deg = 6$ and $n = 8, \deg = 7$ without division tables, cuts AND counts by up to 40% when the lookup table can vary across a profile, and reproduces the depth-3 AES architecture of [HB25] gate-exactly with its inverse (Table 4).

2.2 SAT Co-Synthesis, Metaheuristic Search, and Equivalence Classification

At $n = 4$, Jia et al. [JCL⁺25] combined Prim-like upper bounds, balanced-Boolean lower bounds, and a unified SAT model of $(\delta, L, \deg, \text{FP})$ and UMC 180 nm area/depth to prove the exact minimum area of optimal 4-bit S-boxes ($\delta = 4, L = 8$, i.e., NL = 4): 12 GE (8 gates, depth 4) with 2/3-input gates (12.33 GE without fixed points), 11 GE (6 gates, depth 3) with 4-input gates, and 14 GE (depth 8) at minimum coordinate degree 3, while optimizing fixed 5-bit χ_5 (17.00 GE) and 8-bit SKINNY-128 (26.67 GE); without bound pruning, the joint SAT model does not solve even at $n = 4$ [JCL⁺25, Section 1]. Conversely, structural Feistel/Misty constructions [LW14, CDL15, BPU16] and 6-bit cipher designs (SC2000 [SY⁺02], FIDES [BBK⁺13], SPEEDY [LMMR21], BIPBIP [BDD⁺23]) fix a template, whereas evolutionary searches [PČR16, KPF⁺24] and MAP-Elites [MC15] optimize truth-table non-linearity in isolation. YOSHAN uses the Boolean DAG as the genotype, jointly evolving $(\delta, L, \deg)$ and (AND, D , XOR) for $n \in \{5, 6, 7, 8\}$. Because exhaustive affine classification is known only for $n = 4$ (302 classes [LP07]) and quadratic $n \in \{5, 6\}$ (75 and 2,263 classes [BSŠ17, DMB19]), we combine three EA/CCZ invariants [CCZ98, Kal21] with an exact affine-equivalence solver [BDCBP03] (Section 6.1) against all cyclotomic power maps of $\mathbb{F}_{2^n}$ and KECCAK [BDPVA11], ASCON [DEMS21], SPEEDY [LMMR21], AES [Nat23], Dillon’s APN [BDMW10], and their inverses.

3 Algebraic Foundations and Cipher-Level Security Criteria

In this work, we focus on n -bit bijective S-boxes, which are defined as bijective vectorial Boolean functions $S : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ with coordinate functions $S(x) = (f_0(x), \dots, f_{n-1}(x))$ belonging to the symmetric group $\mathcal{S}_{2^n}$.

3.1 Differential Uniformity, Linearity, and Algebraic Degree

Resistance to differential [BS91] and linear cryptanalysis [Mat94] is governed by the Difference Distribution Table (DDT) for $\Delta x \in \mathbb{F}_2^n \setminus \{0\}$, $\Delta y \in \mathbb{F}_2^n$ and the Walsh–Hadamard Transform for masks $\alpha \in \mathbb{F}_2^n$, $v \in \mathbb{F}_2^n \setminus \{0\}$ ($\langle u, v \rangle = \bigoplus_i u_i v_i$):

$$\text{DDT}_S(\Delta x, \Delta y) = \#\{x \in \mathbb{F}_2^n : S(x) \oplus S(x \oplus \Delta x) = \Delta y\}, \quad W_S(\alpha, v) = \sum_x (-1)^{\langle \alpha, x \rangle \oplus \langle v, S(x) \rangle}. \quad (1)$$

The **differential uniformity** $\delta(S) = \max_{\Delta x \neq 0, \Delta y} \text{DDT}_S(\Delta x, \Delta y)$ bounds single-S-box differential transitions by $DP_{\max} = \delta(S)/2^n$. The optimum $\delta(S) = 2$ defines **Almost Perfect Nonlinear (APN)** functions [Nyb94]; among bijections, APN permutations are realized in odd dimensions ($n \in \{5, 7\}$) by Gold (x^{2^k+1}) and Kasami ($x^{2^{2k}-2^k+1}$) power maps with $\gcd(k, n) = 1$ and in dimension $n = 6$ by a single known CCZ class—Dillon’s permutation [BDMW10]—while existence of APN permutations for even $n \geq 8$ (the Big APN Problem) remains open. The **linearity** $L(S) = \max_{\alpha, v \neq 0} |W_S(\alpha, v)|$ sets the maximum correlation $C_{\max} = L(S)/2^n$, the linear potential $LP_{\max} = C_{\max}^2$, and the non-linearity $N_L(S) = 2^{n-1} - L(S)/2$. For odd n , Almost Bent (AB) permutations achieve the Sidelnikov–Chabaud–Vaudenay bound [CV95] $L(S) = 2^{(n+1)/2}$ ($L = 8$ for $n = 5$; $L = 16$ for $n = 7$); for even n , where no bijection can be bent ($L(S) > 2^{n/2}$, so $N_L \leq 2^{n-1} - 2^{n/2-1} - 2$ for balanced functions [Car21]), the best known bijective values are $L = 16$ at $n = 6$ [BDMW10] and $L = 32$ ($N_L = 112$) at $n = 8$. Boomerang and rectangle attacks are governed by the Boomerang Connectivity Table $\text{BCT}_S(a, b) = \#\{x : S^{-1}(S(x) \oplus b) \oplus S^{-1}(S(x \oplus a) \oplus b) = a\}$ and the **boomerang uniformity** $\beta(S) = \max_{a, b \neq 0} \text{BCT}_S(a, b)$ [CHP⁺18], with $\delta(S) \leq \beta(S) \leq 2^n$ and $\beta(S) = 2$ exactly for APN permutations [BC18].

Every coordinate f_i has a unique Algebraic Normal Form (ANF), $f_i(x) = \bigoplus_{u \in \mathbb{F}_2^n} a_u^{(i)} x^u$ with $x^u = \prod_j x_j^{u_j}$, computed in $O(n 2^n)$ operations by the binary Möbius transform. The **algebraic degree** $\deg(S)$ is the maximum degree among $f_0, \dots, f_{n-1}$ ($\deg(S) \leq n-1$ for any bijection with $n \geq 2$, and $\deg(S) \leq (n+1)/2$ for any odd- n AB function [CCZ98]), and $\deg_{\min}(S) = \min_{v \neq 0} \deg\langle v, S(x) \rangle \geq 1$. The **component degree spectrum** $\mathcal{D}(S) = \{\deg\langle v, S(x) \rangle : v \in \mathbb{F}_2^n \setminus \{0\}\}$, written $[d_1^{m_1}, d_2^{m_2}, \dots]$, records the degrees of all $2^n - 1$ non-zero components; because $V_{\leq d} = \{v : \deg\langle v, S(x) \rangle \leq d\}$ forms a flag of $\mathbb{F}_2$ -linear subspaces $0 = V_{\leq 0} \subset V_{\leq d_1} \subset \dots \subset \mathbb{F}_2^n$ with $k_j = \dim V_{\leq d_j}$ and $m_j = 2^{k_j} - 2^{k_{j-1}}$, any two-level spectrum with $d_1 = \deg_{\min}(S) < d_2 = \deg(S)$ takes the subspace-partition form $\mathcal{D}(S) = [d_1^{2^k-1}, d_2^{2^n-2^k}]$ with $k = \dim V_{\leq d_1}$ (and the uniform spectrum $[d_2^{2^n-1}]$ when $d_1 = d_2$).

3.2 Algebraic Degree, Higher-Order Derivatives, and Masked Total Complexity

While Wide-Trail bounds [DR01] bound single-trail probabilities $(DP_{\max})^{A_r}$ and $(LP_{\max})^{A_r}$ across A_r active S-boxes, resistance to boomerang [CHP⁺18], integral and division-property [Tod15], zero-sum [BCDC11], cube [DS09], and interpolation attacks [JK97] is governed by algebraic degree growth. Following Lai [Lai94] and Knudsen [Knu95], the r -th order derivative of S along linearly independent $a_1, \dots, a_r$ sums S over an r -dimensional cube and lowers the degree by at least r :

$$D_{a_1} \cdots D_{a_r} S(x) = \bigoplus_{I \subseteq \{1, \dots, r\}} S\left(x \oplus \bigoplus_{i \in I} a_i\right), \quad r > \deg(S) \implies D_{a_1} \cdots D_{a_r} S(x) \equiv 0. \quad (2)$$

When $\deg(S) = 2$, third derivatives vanish on a single round and the r -round degree grows at most as 2^r [BCDC11], which forces high round counts in ciphers with quadratic S-boxes (ASCON [DEMS21], KECCAK- f [BDPVA11]) relative to degree-5 designs such as SPEEDY [LMMR21]. In hardware, multiplicative complexity (MC , minimum AND gates) and multiplicative depth (MD) satisfy $MC(S) \geq \deg(S) - 1$ and $MD(S) \geq \lceil \log_2 \deg(S) \rceil$ [BFP16, GR16]. Realizing degree-3 to degree-7 S-boxes at $D \in \{2, 3, 4\}$ with near-quadratic AND counts therefore allows steeper degree growth, fewer rounds, and lower total masked latency ($R_{\text{rounds}} \times D$).

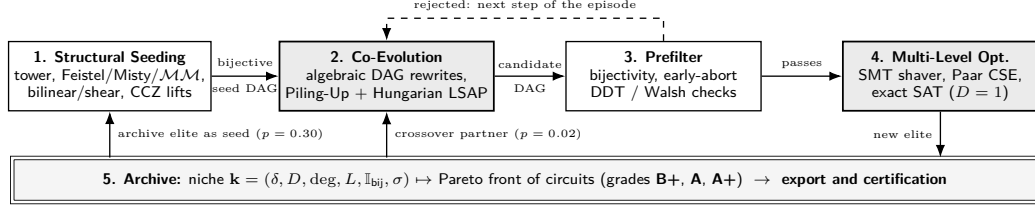


Figure 1: One search episode of YOSHAN. Stages 2–4 repeat until the episode stops; the archive is shared by all workers. σ is a signature of the DDT and $|\text{LAT}|$ value histograms.

3.3 Tower Fields, Affine Equivalence, and Forward/Inverse Duality

Composite tower fields. Direct polynomial evaluation of finite-field inversion $x \mapsto x^{2^n-2}$ produces deep circuits, whereas decomposing $\mathbb{F}_{2^8} \cong \mathbb{F}_{((2^2)^2)^2}$ via a basis isomorphism $M \in \text{GL}(8, \mathbb{F}_2)$ reduces $\mathbb{F}_{2^8}$ inversion to a norm $\Delta \in \mathbb{F}_{2^4}$, one $\mathbb{F}_{2^4}$ inversion, and two $\mathbb{F}_{2^4}$ multiplications, in the polynomial basis of Satoh et al. [SMTM01] or in Canright’s normal basis [Can05, BP10]. Recursing down to $\mathbb{F}_{2^2}$, where inversion is a free wire swap and multiplication uses 3 AND gates at $D = 1$, yields Canright’s AES architecture, Boyar–Peralta’s 32-AND circuit [BP10, BP12], the 33-AND ($D = 4$) circuit of Bolzer et al. [BDM26], and the depth-3 inverter of Hadžić and Bloem [HB25]; analogously, $\mathbb{F}_{2^6} \cong \mathbb{F}_{(2^3)^2}$ yields the 21-AND, 63-XOR ($D = 3$) 6-bit inverter of [BDM26, Table 5].

Affine equivalence and forward/inverse duality. Two S-boxes are **affine equivalent** ($S_1 \sim_{\text{aff}} S_2$) if $S_2(x) = B \cdot S_1(Ax \oplus c) \oplus d$ for $A, B \in \text{GL}(n, \mathbb{F}_2)$ and $c, d \in \mathbb{F}_2^n$. While δ , L , $\deg$, $\mathcal{D}(S)$, $MC(S)$, and the minimum multiplicative depth $MD(S)$ are affine invariants on $\mathcal{S}_{2^n}$ (shared across each cyclotomic coset $C_e = \{e \cdot 2^k \bmod (2^n - 1)\}$ of power permutations x^e), the fixed-point count $\text{FP}(S)$, the SAC deviation, and the exact XOR count depend on the representative. For decryption, self-inverse classes and balanced Feistel networks reuse the forward non-linear core or reverse its rounds at identical (AND, D), whereas quadratic S-boxes have higher-degree inverses ($\deg(S^{-1}) \in \{3, 4\}$) requiring $D^{-1} \geq 2$ (Section 6.3). Because $(\delta, L, \deg)$ are properties of the permutation S while masked area, randomness (N_{AND}), and pipeline registers ($4N_{\text{AND}} + 2N_{\text{sync}}$, Section 5.2) depend on the DAG factorization G , YOSHAN co-optimizes S and G in one loop.

4 The Yoshan Search Engine

Searching for a cryptographically strong S-box and synthesizing its circuit are traditionally separate problems: a permutation S is chosen for $(\delta(S), L(S))$, then the gate count of the fixed lookup table is minimized. Under masking this separation is costly because latency and area/randomness are set by the AND depth D and AND count (Section 5.2)—properties of the circuit DAG, not of the lookup table. YOSHAN therefore searches directly over circuits. A candidate is a DAG G of 2-input AND, 2-input XOR, and NOT gates (plus small lookup-table nodes during search) computing $S = \text{LUT}(G)$. Workers run search episodes in parallel and share elites through a common archive across five stages (Figure 1): (1) **seeding** (Section 4.1) from a bijective circuit; (2) **circuit evolution** (Section 4.2) under annealed cryptographic and cost rules; (3) a fast **prefilter and grade gate** (Section 4.3); (4) **compression** (Section 4.3) via sub-DAG shaving, CSE, and SAT; and (5) the **quality-diversity archive** (Section 4.4), followed by gate-exact export and formal certification (Sections 5 and 7).

4.1 Stage 1: Seeding

Because random tables are almost never bijective and random gate graphs are deep, every episode starts from a bijective circuit. Finite-field constructions use $\mathbb{F}_{2^n} \cong \mathbb{F}_2[x]/(p_n(x))$ with $p_5 = x^5 + x^2 + 1$, $p_6 = x^6 + x + 1$, $p_7 = x^7 + x + 1$, and $p_8 = x^8 + x^4 + x^3 + x + 1$ (a change of polynomial basis is a linear isomorphism and preserves affine classes). A worker draws its seed from one of three sources: (1) **an archive elite** with probability 0.30 (once the archive is non-empty); otherwise (2) **a published or algebraic circuit** with conditional probability ρ_g (0.7 in optimal campaigns, 0.6 in relaxed/lightweight campaigns)—the cipher S-boxes of Table 6, Gold and inverse-Gold circuits, DB₁ [AL26], Q_{2256}/Q_{2258} [BDM25] and their inverses, fractional subfield towers, composite-field inversion over $\mathbb{F}_{(2^3)^2}$ (21 AND at $D = 3$), and, for $n = 8$, Camellia, Gold, tower-field AES over $\mathbb{F}_{((2^2)^2)^2}$ (36 AND at $D = 4$ [Can05]) and the 42-AND, $D = 3$ inverter architecture of Hadžić and Bloem [HB25]; or (3) **a structural seed** (conditional probability $1 - \rho_g$): **Feistel**, **Misty**, and **product-round Feistel (MM-type)** constructions $S(x_L, x_R) = (x_R, x_L \oplus \pi(x_R) \odot \phi(x_R))$ (3-round balanced Feistel seeds carry a structural probability-one boomerang, Section 6.6), sparse affine wrappers $S(x) = A_1 R(A_2 x \oplus c_1) \oplus c_2$ with A_1, A_2 formed by $\lfloor n/2 \rfloor$ to $2n$ row additions from I_n , **linear/CCZ graph automorphisms** $\mathcal{L}(\begin{smallmatrix} x \\ F(x) \end{smallmatrix}) = \begin{pmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} x \\ F(x) \end{pmatrix}$ with $\mathcal{L} \in \text{GL}(2n, \mathbb{F}_2)$, yielding the CCZ-equivalent permutation $S_{\text{lift}} = \pi_2 \circ \pi_1^{-1}$ whenever $\pi_1(x) = L_{11}x \oplus L_{12}F(x)$ and $\pi_2(x) = L_{21}x \oplus L_{22}F(x)$ are both bijective (also applied as Stage 2 mutations), the differentiable templates of Section 4.2, single lookup-table nodes holding a power map (mutated by LUT swaps until lowered to gates, Section 5.1), and, at $n = 5$, a depth-1 quadratic template with five parallel AND gates, $y_k = M_k x \oplus (\alpha_k \cdot x \oplus a_k)(\beta_k \cdot x \oplus b_k) \oplus c_k$, whose bijective parameter sets are drawn by a bounded-random SAT query. Two further seed techniques are specific to quadratics:

Bilinear (i) and shear (ii) seeds. (i) For a quadratic target, a randomized solver builds depth-1 seed circuits directly by extracting the constant $c_k = S_k(0)$, linear part $L_{k,i} = S_k(e_i) \oplus c_k$, and quadratic coefficients $Q_{k,i,j} = S_k(e_i \oplus e_j) \oplus c_k \oplus L_{k,i} \oplus L_{k,j}$ ($i < j$) of each output bit via finite differences and sampling R products of linear forms $a_r(x) = (u_r^\top x) \wedge (v_r^\top x)$:

$$a_r(x) = \bigoplus_{0 \leq i < j < n} (u_{r,i} v_{r,j} \oplus v_{r,i} u_{r,j}) x_i x_j \oplus \bigoplus_{j=0}^{n-1} (u_{r,j} \wedge v_{r,j}) x_j, \quad (3)$$

$$L_{k,j}^{\text{req}} = L_{k,j} \oplus \bigoplus_{r=1}^R W_{k,r}(u_{r,j} \wedge v_{r,j}), \quad S_k(x) = c_k \oplus \bigoplus_{j=0}^{n-1} L_{k,j}^{\text{req}} x_j \oplus \bigoplus_{r=1}^R W_{k,r} a_r(x). \quad (4)$$

Whenever each Q_k lies in the $\mathbb{F}_2$ -span of $(u_{r,i} v_{r,j} \oplus v_{r,i} u_{r,j})_{i < j}$, Gaussian elimination yields $W_{k,r}$, and $L_{k,j}^{\text{req}}$ cancels the diagonal $x_j^2 = x_j$ terms, giving a depth-1 circuit with R AND gates (e.g., $R = 7$ for 5-bit Gold x^5). (ii) A *shear* adds $y_{n-1} \leftarrow y_{n-1} \oplus u(y) \wedge v(y)$ with linear forms u, v independent of y_{n-1} , preserving bijectivity while adding one AND gate and raising the degree of a quadratic base up to 4 (Section 6.6).

4.2 Stage 2: Circuit Evolution

An episode evolves G over many steps (Algorithm 1), proposing G' via: (1) **crossover** (prob. 0.02) with an elite from another niche—a homotopy crossover of the two parents' template weights when both are template-based, otherwise both parent DAGs are grafted onto shared inputs and their outputs combined by XOR or AND (non-bijective children are discarded by the prefilter); (2) **function-preserving rewrites** (CSE, distributive folding $(u_1 \wedge v) \oplus (u_2 \wedge v) \rightarrow (u_1 \oplus u_2) \wedge v$, and every 25 steps AND-tree rotation and factoring; Figure 2); or (3) **function-changing mutations**: before the (δ, L) targets are met, LUT

swaps and Feistel, Misty, or Lai–Massey moves; once met, an *AND eraser* (replacing one or two AND gates by XORs), *depth collapse* (rewiring a deepest AND input to a wire ≥ 2 levels shallower), and affine or CCZ graph automorphisms. Moves are kept only if G' is acyclic, bijective, and (for erasers/collapses) maintains the (δ, L) targets. Candidates are compared lexicographically: first by distance to targets, maximal DDT count, deg, and SAC deviation; once targets are met, by higher deg and lower hierarchical masked-cost score $10^5 N_{\text{AND}} + 1.5 \cdot 10^4 D + 100 N_{\text{XOR}} + 10 N_{\text{NOT}}$ (whose decade weights enforce $(N_{\text{AND}}, D) \succ N_{\text{XOR}} \succ N_{\text{NOT}}$, so non-linear gates and pipeline stages are never traded for linear gates) under cosine simulated annealing (period 2000, $T \in [0.05, 5]$), restarting from an elite after 50 duplicates and stopping after $\max(P, 1500(n - 3))$ non-improving steps. Lookup-table nodes carry no gate cost in this score and occur only while the (δ, L) targets are unmet; every candidate is lowered to explicit gates before compression, certification, and archive accounting (Section 5.1).

Differentiable Piling-Up relaxation and Hungarian LSAP projection. To complement the discrete rewrites with gradient guidance, YOSHAN includes a differentiable template module over parameterized multi-round networks (a Keccak-like SPN with a fixed invertible diffusion matrix, unbalanced Feistel, and dual-branch Misty networks, all with learned round wire permutations). Relaxing wires to $p_i \in [0, 1]$, AND gates to $p_i p_j$, and k -input XOR sums to the multilinear Piling-Up relaxation $\widetilde{\text{XOR}}(p_1, \dots, p_k) = \frac{1}{2} - \frac{1}{2} \prod_{m=1}^k (1 - 2p_m)$ [Mat94] (exact on $\{0, 1\}^k$ and under mutually independent inputs), and multiplying by the Hadamard matrix H_{2^n} yields the relaxed Walsh spectrum $\widetilde{W}_S(\alpha, v) = H_{2^n}(1 - 2\widetilde{S}_v(x))$ of every relaxed component on all 2^n inputs, which the multi-objective Adam loss

$$\mathcal{L}(\Theta; \tau) = 20 \left(\sum_{\alpha \neq 0, v \neq 0} |\widetilde{W}_S(\alpha, v)|^{32} \right)^{1/32} + 10 \mathcal{L}_{\text{SAC}} + 15 \mathcal{L}_{\text{sym}} + \lambda(\tau)(\mathcal{L}_{\text{ent}} + \mathcal{L}_{\ell_1}) \quad (5)$$

drives down together with the SAC deviation, fixed points of the round wire permutations, and (with the annealed weight $\lambda(\tau)$) non-binary wires and dense polynomials. After gradient descent, the weights are projected to a discrete circuit by the Hungarian algorithm for the Linear Sum Assignment Problem (LSAP) [Kuh55]: one LSAP per round permutation and K masked rectangular LSAP passes per branch polynomial. Bijectivity is structural for the additive branch update $L \leftarrow L \oplus F(R)$ of the Feistel templates and is checked at runtime by Stage 3 for SPN projections. The module serves as one of the structural seed generators of Stage 1 and as the crossover operator of Algorithm 1; its exact terms, schedule, and projection are listed in SM A.2, and Section 8 discusses the interplay between Piling-Up attenuation and discrete projection.

4.3 Stages 3 and 4: Prefilter, Grade Gate, and Circuit Compression

Each candidate first passes a fast bijectivity check and row-by-row DDT and Walsh scans that abort as soon as an entry exceeds a cap above the target. The archive accepts only circuits with $\text{deg} \geq 2$ meeting one of three grade thresholds (**A+**, **A**, **B+** in Table 1). Besides the rewrites of Stage 2 (associative AND-tree rotation, distributive AND folding, and common sub-expression sharing; Figure 2), accepted circuits are compressed without changing S by three tools: (1) **Paar CSE and Boyar–Peralta wire reuse** [Paa97, BP10] merge identical gates, repeatedly factor shared XOR operand pairs, and synthesize intra-stage linear blocks via distance-1/2 wire reuse; (2) the **sub-DAG shaver** tests whether a pair of AND gates (g_1, g_2) supported on a cut C ($|C| \leq 6$) can be replaced by a single product $a(z) = (u^\top z) \wedge (v^\top z)$ plus affine functions of $z \in \mathbb{F}_2^{|C|}$, i.e., whether for each $k \in \{1, 2\}$ some $w_k \in \{0, 1\}$ makes $g_k \oplus w_k a$ affine on the cut—decided exactly by a Z3 query [dMB08] over all $2^{|C|}$ cut assignments, or by a fast filter that checks, for each $k \in$

Algorithm 1: One YOSHAN search episode (per worker)

Input: width n , targets (δ_t, L_t) , shared archive $\mathcal{A}$ **Ensure:** new elites inserted into $\mathcal{A}$

- 1: $G \leftarrow$ seed from $\mathcal{A}$ ($p = 0.30$); otherwise from the circuit library ($p = \rho_g$) or a structural construction ($p = 1 - \rho_g$) (Section 4.1)
- 2: **repeat**
- 3: $G' \leftarrow$ crossover with an elite ($p = 0.02$); otherwise a function-preserving rewrite if one applies (CSE or distributive folding; AND-tree rotation and factoring every 25th step); otherwise a function-changing mutation of G
- 4: **if** G' is cyclic, or fails the prefilter, or $\text{LUT}(G')$ is not a bijection **then continue**
- 5: **end if**
- 6: compute δ, L, deg , SAC deviation, and gate counts of G' ; **if** G' is better than G or accepted by annealing **then** $G \leftarrow G'$
- 7: **if** G has grade $\geq \mathbf{B+}$ and is non-dominated in its niche **then** compress G (CSE) and insert into $\mathcal{A}$
- 8: **end if**
- 9: **until** targets met and no improvement for $\max(P, 1500(n - 3))$ steps
- 10: final compression of G (sub-DAG shaver, CSE, and exact $D = 1$ SAT for quadratic $n \leq 7$); insert G into $\mathcal{A}$ if non-dominated

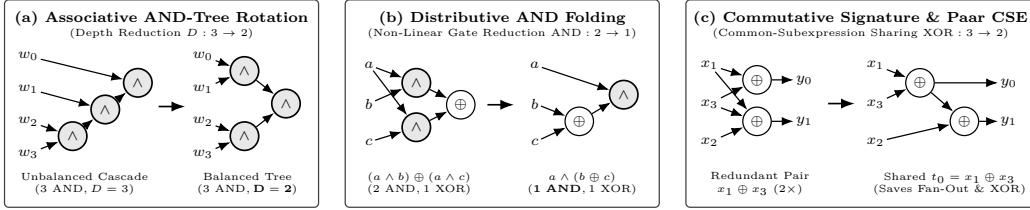


Figure 2: Examples of the circuit rewrites used in Stages 2 and 4 (Sections 4.2 and 4.3): associative AND-tree rotation, distributive folding, and common-subexpression elimination. Shaded nodes ($\wedge$) are AND gates, white nodes ($\oplus$) are XOR gates.

$\{1, 2\}$, whether at least one $f \in \{g_k, g_k \oplus a\}$ satisfies $f(e_{b_1} \oplus e_{b_2}) \oplus f(e_{b_1}) \oplus f(e_{b_2}) \oplus f(0) = 0$ for all $b_1 < b_2$ (complete only when $\deg_z g_k \leq 2$); every accepted rewrite is re-simulated on all 2^n inputs and reverted unless $\text{LUT}(G)$ is unchanged, so the shaver is sound independently of the filter; and (3) **exact SAT re-synthesis**, which searches for a $D = 1$ circuit with at most 8, 13, 14 AND gates on quadratic results for $n = 5, 6, 7$. The same pass, applied once more to every benchmark netlist before printing (sub-DAG shaver and stage-preserving Paar CSE), is what Section 6.5 calls the *Stage-4 pass*.

4.4 Stage 5: Quality-Diversity Archive

Without diversity, cheap quadratic circuits ($\text{deg} = 2, D = 1$) would crowd out higher-degree circuits before they are compressed. Following MAP-Elites [MC15], a candidate competes only within its niche

$$\mathbf{k}(S) = \left(\delta(S), D(G), \text{deg}(S), L(S), \mathbb{I}_{\text{bij}}(S), \sigma(S) \right), \quad (6)$$

where $D(G)$ is the depth of the circuit computing S , $\mathbb{I}_{\text{bij}}(S)$ is constant for archived circuits (non-bijective candidates are rejected in Stage 3), and $\sigma(S)$ hashes the DDT and $|\text{LAT}|$ histograms so that inequivalent S-boxes with the same (δ, L, deg) do not collide. A candidate enters $\mathcal{A}[\mathbf{k}]$ if it has grade $\geq \mathbf{B+}$ and is not Pareto-dominated in $(N_{\text{AND}}, D, N_{\text{AND}} + N_{\text{XOR}}, c)$ with $c = 1000D + 0.8N_{\text{AND}} + 1.6N_{\text{XOR}} + 0.5N_{\text{NOT}}$ (inside a niche D is fixed, so dominance reduces to $(N_{\text{AND}}, N_{\text{AND}} + N_{\text{XOR}}, 0.8N_{\text{AND}} + 1.6N_{\text{XOR}} + 0.5N_{\text{NOT}})$, weighting gates approximately by their unmasked CMOS area

Table 1: Grade thresholds of the archive gate (all grades also require $\deg \geq 2$).

n	A+	A	B+
5	$\delta \leq 2, L \leq 8$	$\delta \leq 2, L \leq 12$; or $\delta \leq 4, L \leq 8$; or $\delta \leq 4, L \leq 12, \deg \geq 3$	$\delta \leq 8, L \leq 16$
6	$\delta \leq 2, L \leq 16$	$\delta \leq 4, L \leq 24$	$\delta \leq 6, L \leq 24$; or $\delta \leq 8, L \leq 28$
7	$\delta \leq 2, L \leq 16$	$\delta \leq 4, L \leq 32$	$\delta \leq 6, L \leq 40$; or $\delta \leq 8, L \leq 44$
8	$\delta \leq 4, L \leq 32$	$\delta \leq 8, L \leq 48$	$\delta \leq 8, L \leq 64$; or $\delta \leq 16, L \leq 80$

ratios 1.064 : 1.596 : 0.532 μm^2 of Eq. (7)); dominated entries are pruned, keeping at most eight circuits per niche (Figure 4 in SM A).

5 Netlist Export and Gate-Exact Hardware Synthesis

Standard logic optimizers such as Berkeley ABC [BM10] and Yosys [WGK13] operate on And-Inverter Graphs (AIGs), expanding each 2-input XOR into three AND gates and inverters; their algebraic and Boolean rewrites (**strash**, **rewrite**, **refactor**) mix linear and non-linear cones and trade XOR gates for AND gates or compound cells (A0I21, OAI21). Under Boolean masking every non-linear gate costs cross-share products, registers, and fresh randomness while linear gates act share by share, so YOSHAN never applies AIG restructuring to archived circuits: it emits each DAG gate for gate and maps every Boolean operator 1-to-1 to its standard cell.

5.1 Gate-for-Gate Export and Technology Mapping

Each archived DAG is emitted as a structural SystemVerilog module with one assignment per AND, XOR, and NOT gate. Any remaining lookup-table node is expanded first via sub-circuit synthesis or the algebraic normal form (each degree- d monomial as a balanced AND tree of depth $\lceil \log_2 d \rceil$ with shared sub-products and XOR trees). Before printing in SM C, structurally identical gates are merged, fan-out-free gates are pruned, and the resulting netlist is exhaustively re-verified; all gate counts in this paper refer to these printed netlists.

To measure silicon area and timing without altering a single gate, we synthesize all 60 forward and inverse netlists in Yosys against the NanGate 45 nm Open Cell Library (Nangate45_typ.lib) [Nan11] via 1-to-1 structural mapping: **simplemap** lowers operators to primitives, **techmap** and **dfflibmap** bind each primitive to its unit-drive cell (AND2_X1, XOR2_X1, INV_X1, DFF_X1) without invoking ABC’s AIG converter, **stat -liberty** reports standard-cell area in μm^2 and gate equivalents (1 GE = $\text{Area}(\text{NAND2_X1}) = 0.798 \mu\text{m}^2$), and **ltp** extracts the longest topological path. Berkeley ABC (**read_verilog -m; topo; stime**) then computes the Non-Linear Delay Model (NLDLM) critical-path delay in picoseconds, capturing pin-to-pin slew and fan-out capacitance while leaving the gate graph untouched.

5.2 Unmasked and Pipelined First-Order Masked Silicon Synthesis

Under 1-to-1 NanGate 45 nm mapping, the unmasked combinational cell area of a netlist with $(N_{\text{AND}}, N_{\text{XOR}}, N_{\text{NOT}})$ gates is

$$\text{Area}_{\text{unmasked}} (\mu\text{m}^2) = 1.064 \cdot N_{\text{AND}} + 1.596 \cdot N_{\text{XOR}} + 0.532 \cdot N_{\text{NOT}} \quad (7)$$

(1.333, 2.0, and 0.667 GE per cell). To measure the physical cost of first-order Boolean masking for all 30 forward and 30 inverse circuits, we compile each depth- D DAG into a fully pipelined D -stage, two-share ($t = 1, d = 2$) Domain-Oriented Masking (DOM-indep) [GMK16] SystemVerilog module (Figure 3 in SM A) and synthesize it gate for gate in Yosys:

1. **Stage assignment and pipeline synchronization.** Every wire w is assigned its As-Soon-As-Possible (ASAP) multiplicative depth $\ell(w) \in \{0, \dots, D\}$ ($\ell = 0$ for primary inputs, $\ell = \max(\ell(a), \ell(b))$ for linear gates, $\ell = \max(\ell(a), \ell(b)) + 1$ for AND gates). Whenever a wire produced at stage ℓ_0 feeds a gate at stage $\ell_1 > \ell_0$ (or a primary output at stage D), its two shares are routed through a deduplicated shift-register chain of $\ell_1 - \ell_0$ flip-flop pairs (2 DFF_X1 per stage crossed, $4.522 \mu\text{m}^2$ per DFF_X1). Because linear XOR trees compress wires before crossing stage boundaries, ASAP scheduling uses fewer or equal synchronization flip-flops than ALAP scheduling on 23 of the 30 forward circuits (138 vs. 144 on $D = 4$ AES; on $D = 3$ AES ALAP trims them from 48 to 42).
2. **Linear and non-linear gadgets.** Each 2-input XOR is duplicated across the two share domains (2 XOR2_X1); each NOT inverts share 0 only (1 INV_X1). Each AND gate crossing into stage ℓ becomes a DOM-indep gadget with two inner-domain and two cross-domain products (4 AND2_X1), two resharing XORs with one fresh random bit (2 XOR2_X1), four stage- ℓ registers (4 DFF_X1), and two domain-compression XORs (2 XOR2_X1).

A compiled $d = 2$ DOM-indep module with N_{sync} synchronized wire-stage crossings therefore instantiates

$$N_{\text{AND}2} = 4N_{\text{AND}}, N_{\text{XOR}2} = 2N_{\text{XOR}} + 4N_{\text{AND}}, N_{\text{INV}} = N_{\text{NOT}}, N_{\text{DFF}} = 4N_{\text{AND}} + 2N_{\text{sync}}, \quad (8)$$

consumes N_{AND} fresh random bits per evaluation, and completes one evaluation per clock cycle at latency D . Here DOM-indep serves as an architectural register/area baseline: at $D \geq 2$ its inner-domain products ($a_0 \wedge b_0, a_1 \wedge b_1$) are unregistered before domain compression and carry no multi-stage PINI composability guarantee [CGLS21, MMSS19], while Tables 3 and 9 also report intra-stage fan-in reconvergence N_{reconv} (Section 7.3); SM A.1 gives the exact $d = 2$ cell/register scaling and analytical AES comparison under operand resharing (DOM-dep [GMK16]), two-cycle PINI HPC2 [CGLS21], and single-cycle HPC3/HPC3.1 [KM22, HB25]. Every generated DOM-indep netlist is verified on all 2^n inputs ($y_0 \oplus y_1 = S(x_0 \oplus x_1)$ and $x_0 \oplus x_1 = S^{-1}(y_0 \oplus y_1)$), and all 60 unmasked and 60 DOM-indep post-synthesis netlists are formally proven equivalent to their SystemVerilog modules in Yosys with 1-to-1 cell counts matching Eqs. (7)–(8).

6 Results

We ran eight campaigns across $n \in \{5, 6, 7, 8\}$ (Table 5 in SM B), pairing one campaign steered toward the near-optimal profiles with one steered toward relaxed or lightweight profiles per width; the grade of every archived circuit is decided by Table 1 regardless of campaign. The archive holds 31,129 certified netlists of 30,786 distinct permutations across 56 $(n, \delta, L, \text{deg})$ profiles (Table 7, broken down by profile in Table 11); every netlist passed exhaustive simulation and the Z3 check of Section 7. At $n = 7$, where the optimal campaign is seeded with APN power maps, 865 of the 895 distinct permutations lie in APN power-map classes, while the lightweight campaign contributes compact quartic and degree-6 permutations with $\delta = 4$. At $n = 6$ and $n = 8$ the relaxed campaigns populate a few very large niches of grade **B+**: at $n = 6$, 19,380 netlists in the $(\delta, L, \text{deg}) = (6, 24, 5)$ profile (min-AND 12 at $D = 4$) and 1,728 and 1,601 in $(8, 24, 5)$ and $(8, 28, 5)$ (min-AND 16 and 11); at $n = 8$, 3,925 netlists in the $\delta \in \{10, 12\}$, $L \in \{64, 68, 72\}$ profiles, where ANF-synthesized circuits need 239–246 AND gates and the Feistel-type circuits of the YSH_{8,2} family 24–28. The compact hardware candidates concentrate along the 104 non-dominated (AND, XOR, D) Pareto points of Table 11. Each of the 30 circuits evaluated below (Table 2) is printed in full in SM C with its inverse, and Table 10 traces every one of them through the pipeline stages (Section 6.5).

Table 2: Cryptographic properties and gate counts of the 30 forward (S) and inverse (S^{-1}) circuits (SM C). δ : differential uniformity; L : linearity ($NL = 2^{n-1} - L/2$); β : boomerang uniformity (Section 3.1; $\dagger$ Feistel-type permutation with a probability-one boomerang, $\beta = 2^n$: the 3-round Feistel circuits and the two-round unbalanced-Feistel network $YSH_{5,4}$, Section 6.6); d/d^{-1} : degree of S and S^{-1} ; $\mathcal{D}(S)$: degree spectrum; FP: fixed points; SAC: max avalanche bias $|\Pr_x[S_j(x) \neq S_j(x \oplus e_i)] - \frac{1}{2}|$; A/X/N, D : AND/XOR/NOT gates and AND depth. Category A: equal ($=$), affine-equivalent ($\sim_{\text{aff}}$), or invariant-consistent ($\sim$) to Table 6; Category B ($YSH_{n,i}$): EA-separated from Table 6. *3-round χ_3 -Feistel involution ($S^{-1} = S, \mathcal{D}(S^{-1}) = [3^{63}]$).

S-box (SM)	n	δ	L	β	d/d^{-1}	$\mathcal{D}(S)$	FP	SAC	Forward (S)		Inverse (S^{-1})		Cat. / Class
									A/X/N	D	A/X/N	D^{-1}	
$YSH_{5,1}$: quartic, $\delta=4$ (C.1)	5	4	12	6	4/4	$[2^{15}, 4^{16}]$	0	.125	11/20/3	2	13/38/6	2	B ($YSH_{5,1}$)
$YSH_{5,2}$: quadratic, $\delta=4$ (C.2)	5	4	16	12	2/3	$[2^{31}]$	0	.5	5/12/3	1	9/32/2	2	B ($YSH_{5,2}$)
$YSH_{5,3}$: cubic, $\delta=8$ (C.3)	5	8	16	20	3/3	$[2^{15}, 3^{16}]$	2	.5	6/11/0	2	9/22/2	2	B ($YSH_{5,3}$)
$YSH_{5,4}$: quartic, $\delta=8$ (C.4)	5	8	16	32 $\dagger$	4/4	$[2^7, 3^8, 4^{16}]$	2	.5	5/12/1	2	5/11/2	2	B ($YSH_{5,4}$)
Gold x^5 (C.5)	5	2	8	2	2/3	$[2^{31}]$	2	.5	7/22/0	1	10/37/4	2	A ($= x^5$)
Gold class, SAC-ideal (C.6)	5	2	8	2	2/3	$[2^{31}]$	1	0	7/21/4	1	10/41/10	2	A ($\sim_{\text{aff}} x^5$)
χ_5 class (C.7)	5	8	16	16	2/3	$[2^{31}]$	5	.5	5/8/1	1	9/21/3	2	A ($\sim_{\text{aff}} \chi_5$)
inverse-Gold (C.8)	5	2	8	2	3/2	$[3^{31}]$	1	.125	14/31/2	2	7/26/5	1	A ($\sim_{\text{aff}} x^{25}$)
x^{15} (C.9)	5	2	12	2	4/4	$[4^{31}]$	2	.125	14/45/0	2	14/57/0	2	A ($= x^{15}$)
$YSH_{6,1}$: quartic, $\delta=4$ (C.10)	6	4	16	64 $\dagger$	4/4	$[3^7, 4^{56}]$	0	.125	9/27/1	3	9/27/1	3	B ($YSH_{6,1}$)
$YSH_{6,2}$: quartic, $k=5$ (C.11)	6	4	16	12	4/3	$[3^{31}, 4^{32}]$	2	.25	27/64/0	3	9/37/0	2	B ($YSH_{6,2}$)
$YSH_{6,3}$: cubic, $\delta=4$ (C.12)	6	4	16	64 $\dagger$	3/4	$[3^{63}]$	2	.5	9/17/9	3	9/17/8	3	B ($YSH_{6,3}$)
$YSH_{6,4}$: deg. 5, $\delta=8$ (C.13)	6	8	28	18	5/5	$[2^{31}, 5^{32}]$	2	.5	11/22/1	3	22/49/1	3	B ($YSH_{6,4}$)
$YSH_{6,5}$: deg. 5, $\delta=6$ (C.14)	6	6	24	14	5/5	$[2^{31}, 5^{32}]$	1	.125	12/31/0	4	21/57/0	3	B ($YSH_{6,5}$)
$YSH_{6,6}$: quartic, $\delta=8$ (C.15)	6	8	24	16	4/4	$[2^{31}, 4^{32}]$	2	.5	10/22/0	2	27/50/3	2	B ($YSH_{6,6}$)
Q_{2258} class (C.16)	6	4	16	4	2/3	$[2^{63}]$	2	.5	8/34/0	1	18/37/0	2	A ($\sim_{\text{aff}} x^5$)
cubic invol. (C.17)	6	4	16	64 $\dagger$	3/3	$[3^{63}]$	8	.5	9/15/9	3	9/15/9	3	A ($\sim x^{13}$) [*]
inversion (C.18)	6	4	16	4	5/5	$[5^{63}]$	1	.125	21/48/0	3	21/50/0	3	A ($\sim_{\text{aff}} x^{62}$)
Dillon APN (C.19)	6	2	16	2	4/4	$[3^7, 4^{56}]$	1	.5	28/68/0	2	28/78/7	2	A ($=$ Dillon)
$YSH_{7,1}$: quartic, $\delta=4$ (C.20)	7	4	32	12	4/5	$[2^{63}, 4^{64}]$	1	.125	15/70/10	2	30/158/12	3	B ($YSH_{7,1}$)
$YSH_{7,2}$: deg. 6, $\delta=4$ (C.21)	7	4	20	10	6/6	$[2^{63}, 6^{64}]$	2	.0312	51/76/1	3	34/164/9	3	B ($YSH_{7,2}$)
Gold x^3 (C.22)	7	2	16	2	2/4	$[2^{127}]$	2	.5	14/57/0	1	29/149/15	2	A ($= x^3$)
Kasami x^{13} (C.23)	7	2	16	2	3/3	$[3^{127}]$	2	.0625	30/114/0	2	37/139/0	2	A ($= x^{13}$)
inversion (C.24)	7	2	20	2	6/6	$[6^{127}]$	0	.0938	42/182/6	3	42/229/6	3	A ($\sim_{\text{aff}} x^{126}$)
power map x^7 (C.25)	7	6	40	10	3/5	$[3^{127}]$	2	.125	56/98/0	2	51/220/15	3	A ($= x^7$)
$YSH_{8,1}$: deg. 7, $\delta=6$ (C.26)	8	6	56	12	7/7	$[6^1, 7^{254}]$	1	.0781	43/107/7	4	43/109/10	4	B ($YSH_{8,1}$)
$YSH_{8,2}$: deg. 7, $\delta=10$ (C.27)	8	10	72	22	7/7	$[6^3, 7^{252}]$	0	.1094	28/61/1	6	28/84/2	6	B ($YSH_{8,2}$)
$YSH_{8,3}$: deg. 7, $\delta=14$ (C.28)	8	14	72	38	7/7	$[4^1, 5^2, 6^{60}, 7^{192}]$	0	.1094	18/71/2	5	18/77/8	6	B ($YSH_{8,3}$)
AES, $D=3$ (C.29)	8	4	32	6	7/7	$[7^{255}]$	3	.0625	42/101/4	3	42/106/4	3	A ($\sim_{\text{aff}}$ AES)
AES FIPS-197 (C.30)	8	4	32	6	7/7	$[7^{255}]$	0	.0625	36/99/4	4	36/95/4	4	A ($=$ AES)

6.1 Equivalence Classification

To relate each archived permutation S to known S-boxes, we compare three invariants computed from its lookup table: the component degree spectrum $\mathcal{D}(S)$ (Section 3.1), the differential spectrum (multiset of $DDT(a, b)$ for $a \neq 0$), and the extended Walsh spectrum (multiset of $|W_S(\alpha, v)|$ for $v \neq 0$). The last two are invariant under EA ($S' = A_2 \circ S \circ A_1 \oplus A_3$) and CCZ equivalence [CCZ98, Car21], and $\mathcal{D}(S)$ is affine-invariant on $\mathcal{S}_{2^n}$ and EA-invariant whenever $\deg_{\min}(S) \geq 2$. We test every permutation against our reference catalogue (Table 6): one power permutation x^d per cyclotomic coset with $\gcd(d, 2^n - 1) = 1$, five cipher S-boxes (KECCAK χ_5 and ASCON for $n = 5$, SPEEDY and Dillon’s APN permutation for $n = 6$, AES for $n = 8$), and their inverses. A permutation is in **Category A** (1,763 distinct archive permutations) if it matches a catalogue entry on all three invariants, and in **Category B** (29,023 permutations) if at least one invariant differs from every catalogue entry, which certifies EA-inequivalence to Table 6. The 15 Category B benchmark S-boxes of Table 2 have pairwise distinct $(n, \delta, L, \mathcal{D}(S))$ invariants and therefore belong to 15 pairwise disjoint EA classes outside the catalogue; we denote each concrete permutation (and its EA class) by $YSH_{n,i}$ ($i \in \{1, \dots, 4\}$ for $n = 5$; $i \in \{1, \dots, 6\}$ for $n = 6$; $i \in \{1, 2\}$ for $n = 7$; $i \in \{1, 2, 3\}$ for $n = 8$) and its explicit inverse by $YSH_{n,i}^{-1}$.

Because matching forward invariants are necessary but not sufficient for affine equivalence, an exact affine-equivalence solver following Biryukov et al. [BDCBP03] (bidirectional linear-equivalence closure between A and B combined with finite-field multiplicative and

Table 3: Gate-exact Yosys synthesis and ABC NLDM timing of the 30 forward (S) circuits on NanGate 45 nm (inverse circuits: Table 9 in SM B; GE in SM C). **Unmasked**: $\text{AND2_X1/XOR2_X1/INV_X1}$ counts (A/X/N), longest topological path (LTP), area (μm^2), and NLDM delay (ps). **Masked** ($d = 2$ DOM-indep): latency D (cycles) and random bits/eval $R = N_{\text{AND}}$, cells (A/X/N), flip-flops FF (DFF_X1: total = gadget G + sync S), area (μm^2), stage delay (ps), and N_{reconv} : number of AND gates with intra-stage fan-in reconvergence (Section 7.3), to which the mitigations of Section 5.2 apply.

S-box (SM)	n	Unmasked Combinational (S)					Masked ($d = 2$ DOM-indep, S)				
		A/X/N	LTP	μm^2	Delay (ps)	D/R	A/X/N	FF (G+S)	μm^2	Stage (ps)	N_{reconv}
YSH _{5,1} : quartic, $\delta=4$ (C.1)	5	11/20/3	7	45.22	334.4	2/11	44/84/3	66 (44+22)	480.93	310.0	0
YSH _{5,2} : quadratic, $\delta=4$ (C.2)	5	5/12/3	5	26.07	198.3	1/5	20/44/3	30 (20+10)	228.76	202.1	4
YSH _{5,3} : cubic, $\delta=8$ (C.3)	5	6/11/0	5	23.94	231.6	2/6	24/46/0	48 (24+24)	316.01	187.6	0
YSH _{5,4} : quartic, $\delta=8$ (C.4)	5	5/12/1	7	25.00	295.4	2/5	20/44/1	40 (20+20)	272.92	246.2	1
Gold x^5 (C.5)	5	7/22/0	6	42.56	311.2	1/7	28/72/0	38 (28+10)	316.54	288.5	2
Gold class, SAC-ideal (C.6)	5	7/21/4	6	43.09	282.0	1/7	28/70/4	38 (28+10)	315.48	243.3	0
χ_5 class (C.7)	5	5/8/1	4	18.62	141.2	1/5	20/36/1	28 (20+8)	205.88	139.0	1
inverse-Gold (C.8)	5	14/31/2	8	65.44	439.6	2/14	56/118/2	88 (56+32)	646.91	346.2	0
x^{15} (C.9)	5	14/45/0	13	86.72	710.9	2/14	56/146/0	76 (56+20)	636.27	518.9	7
YSH _{6,1} : quartic, $\delta=4$ (C.10)	6	9/27/1	16	53.20	916.7	3/9	36/90/1	74 (36+38)	517.10	561.1	3
YSH _{6,2} : quartic, $\delta=5$ (C.11)	6	27/64/0	10	130.87	502.4	3/27	108/236/0	172 (108+64)	1269.35	434.1	1
YSH _{6,3} : cubic, $\delta=4$ (C.12)	6	9/17/9	11	41.50	411.0	3/9	36/70/9	72 (36+36)	480.40	254.4	0
YSH _{6,4} : deg. 5, $\delta=8$ (C.13)	6	11/22/1	9	47.35	427.0	3/11	44/88/1	80 (44+36)	549.56	358.1	0
YSH _{6,5} : deg. 5, $\delta=6$ (C.14)	6	12/31/0	10	62.24	453.9	4/12	48/110/0	98 (48+50)	669.79	368.8	1
YSH _{6,6} : quartic, $\delta=8$ (C.15)	6	10/22/0	7	45.75	362.8	2/10	40/84/0	66 (40+26)	475.08	317.9	1
Q_{2258} class (C.16)	6	8/34/0	9	62.78	527.6	1/8	32/100/0	44 (32+12)	392.62	354.6	6
cubic invol. (C.17)	6	9/15/9	10	38.30	336.8	3/9	36/66/9	72 (36+36)	474.01	254.4	0
inversion (C.18)	6	21/48/0	15	98.95	797.5	3/21	84/180/0	152 (84+68)	1064.00	367.4	3
Dillon APN (C.19)	6	28/68/0	10	138.32	538.6	2/28	112/248/0	154 (112+42)	1211.36	460.5	0
YSH _{7,1} : quartic, $\delta=4$ (C.20)	7	15/70/10	14	133.00	750.9	2/15	60/200/10	88 (60+28)	786.30	478.9	14
YSH _{7,2} : deg. 6, $\delta=4$ (C.21)	7	51/76/1	16	176.09	840.4	3/51	204/356/1	284 (204+80)	2070.01	802.7	0
Gold x^3 (C.22)	7	14/57/0	9	105.87	558.8	1/14	56/170/0	72 (56+16)	656.49	409.5	5
Kasami x^{13} (C.23)	7	30/114/0	16	213.86	1007.6	2/30	120/348/0	148 (120+28)	1352.34	644.6	19
inversion (C.24)	7	42/182/6	33	338.35	2043.5	3/42	168/532/6	212 (168+44)	1989.68	718.5	42
power map x^7 (C.25)	7	56/98/0	14	215.99	772.4	2/56	224/420/0	280 (224+56)	2174.82	765.9	0
YSH _{8,1} : deg. 7, $\delta=6$ (C.26)	8	43/107/7	25	220.25	1529.9	4/43	172/386/7	236 (172+64)	1869.98	574.7	13
YSH _{8,2} : deg. 7, $\delta=10$ (C.27)	8	28/61/1	17	127.68	969.0	6/28	112/234/1	208 (112+96)	1433.74	275.1	2
YSH _{8,3} : deg. 7, $\delta=14$ (C.28)	8	18/71/2	18	133.53	1084.7	5/18	72/214/2	160 (72+88)	1142.74	431.6	3
AES, $D=3$ (C.29)	8	42/101/4	24	208.01	1516.4	3/42	168/370/4	216 (168+48)	1748.15	574.7	12
AES FIPS-197 (C.30)	8	36/99/4	21	198.44	1148.6	4/36	144/342/4	282 (144+138)	1976.38	371.0	12

Frobenius automorphisms and second-derivative conjugate signatures) tests every Category A permutation for $S(x) = B \cdot S_{\text{ref}}(Ax \oplus a) \oplus b$ with invertible A, B . Three relations result: **exact lookup-table equality** ('=' ; 12 archive permutations, 7 of the 15 Category A circuits of Table 2); **certified affine equivalence** ($\text{'}\sim_{\text{aff}}\text{'}$; 1,749 archive permutations, 7 of the 15 circuits, with explicit (A, a, B, b) in SM C), which also resolves every profile in which several cyclotomic cosets share identical invariants (SM B.1); and **invariant-consistent but not affine equivalent** ($\text{'}\sim\text{'}$), which occurs for only two of the 1,763 Category A permutations, both in the affine class of our 9-AND 3-round χ_3 -Feistel cubic involution (SM C.17): they match the three forward invariants of x^{13} (Q_{2258}^{-1}) but differ in inverse spectrum ($\mathcal{D}(S^{-1}) = [3^{63}]$ vs. $[2^{63}]$), boomerang uniformity ($\beta = 64$ vs. 4), and the EA-invariant third-derivative rank multiset $\{\text{rank}(w \mapsto D_u D_v D_w S(x)) : 0 \neq u \neq v \neq 0\}$ (constant in x with $\text{rank} \leq n - 2 = 4$ since $\deg(S) = 3$: $[2^{273}, 3^{840}, 4^{840}]$ vs. $[3^{189}, 4^{1764}]$), which certifies that they are not EA-equivalent to x^{13} . Most notably, for our 8-bit $D = 3$ AES-class circuit (SM C.29, 42/101/4) the affine equivalence to FIPS-197 AES has $A = I_8$, $a = 0$, B a pure wire permutation (reversing bit order within each 4-bit output nibble), and $b = 0x0F$: moving two of its four output NOT gates and reordering output wires yields the exact FIPS-197 AES S-box at depth $D = 3$ with zero extra gates.

6.2 Gate-Exact NanGate 45 nm Silicon Synthesis

Table 3 reports the post-synthesis standard-cell area and timing of all 30 forward circuits mapped gate for gate onto the NanGate 45 nm Open Cell Library [Nan11] in Yosys [WGK13] and timed in Berkeley ABC [BM10], unmasked and as D -stage pipelined first-order ($t = 1, d = 2$) DOM-indep modules (Section 5.2); Table 9 in SM B gives the same data for the 30 inverse circuits.

6.3 Inverse S-Box Circuit Synthesis (S^{-1})

To obtain compact, low-depth realizations of S^{-1} for all 30 S-boxes without generic ANF expansion, we synthesize and verify the 30 inverse circuits (printed in SM C, synthesized in Table 9) through a four-tier algebraic and structural pipeline followed by the Stage-4 pass; the identities behind each tier are derived in SM B.2.

1. **Tier 1: involutions and affine-folded core reuse (9 circuits).** For the 6-bit cubic involution $S^{-1} = S$ (9/15/9, $D = 3$). Whenever S^{-1} is affine equivalent to S (the five inversion-class circuits and Dillon’s APN permutation) or to another archive circuit (the inverse-Gold S-box, whose inverse $\sim_{\text{aff}} x^5$ re-uses our 7-AND Gold core, and the SAC-ideal Gold-class circuit, whose inverse re-uses the x^{25} core of the Gold x^5 row), the certified identity $S^{-1}(y) = B \cdot S_{\text{ref}}(Ay \oplus a) \oplus b$ reuses the non-linear core of S_{ref} with zero extra AND gates and depth. Rather than wrapping (A, a) and (B, b) around S_{ref} as separate XOR matrices, we split the DAG into its input linear cone, non-linear core, and output linear cone, compose the affine maps with the two cones symbolically over $\mathbb{F}_2$, and re-synthesize the linear layers by Paar’s CSE [Paa97]. This yields, e.g., 21/50/0 ($D = 3$; 13 fewer XOR gates than the published 21/63) for 6-bit inversion, 42/106/4 ($D = 3$) for the $D = 3$ AES-class S-box, 36/95/4 ($D = 4$) for FIPS-197 AES, and 7/26/5 ($D = 1$) for the inverse-Gold S-box, halving both its AND count and depth.
2. **Tier 2: structural reversal, shear unwinding, directional-derivative and transposition inversion, and backward unrolling (13 circuits).** Every Category B circuit is either a multi-round generalized Feistel network or a shear or transposition extension of a lower-degree core Q . The 3-round Feistel networks $\text{YSH}_{6,1}$ and $\text{YSH}_{6,3}$ and the six-round single-bit-branch network $\text{YSH}_{8,2}$ are inverted by reversing their rounds (9/27/1, 9/17/8 at $D = 3$; 28/84/2 at $D = 6$). For the output-sheared cores of $\text{YSH}_{6,2}$, $\text{YSH}_{7,1}$, and $\text{YSH}_{8,1}$ ($Q \sim_{\text{aff}} x^{38}, x^3, x^{254}$), unwinding the involution shear with one AND gate and feeding the inverse core of Q gives 9/37/0 ($D = 2$; a $3\times$ AND reduction over the 27-AND forward circuit), 30/158/12 ($D = 3$), and 43/109/10 ($D = 4$). For the shear extensions of $Q \sim_{\text{aff}} x^5$ ($\text{YSH}_{6,6}$, $\text{YSH}_{6,4}$, $\text{YSH}_{6,5}$) the exact directional-derivative identity $S^{-1}(y) = Q^{-1}(y) \oplus (s(y) \wedge \Delta_c Q^{-1}(y))$ evaluates Q^{-1} (an 18-AND, $D = 2$ tower-field norm circuit) in parallel with the correction term and reaches the minimum depth $D = 2$ (27/50/3) for the quartic $\text{YSH}_{6,6}^{-1}$ and $D = 3$ (22/49/1, 21/57/0) for the two degree-5 inverses, one level shallower than the $D = 4$ forward circuit $\text{YSH}_{6,5}$. For $\text{YSH}_{5,1}$ and $\text{YSH}_{7,2}$, which differ from a quadratic Gold permutation by a single transposition, a two-point indicator evaluated in parallel with Q^{-1} gives 13/38/6 ($D = 2$) and 34/164/9 ($D = 3$; 17 fewer AND gates than the 51-AND forward circuit). For $\text{YSH}_{5,4}$ and $\text{YSH}_{8,3}$, backward unrolling with linear-span closure—a wire is recovered as soon as it is an affine combination of recovered wires or the AND of two of them—re-uses every forward AND gate once: 5/11/2 at $D = 2$ (the same AND count and depth as S) and 18/77/8 at $D = 6$ (one level deeper than S).
3. **Tier 3: composite-field norm, Gold composition, and Karatsuba power-map factorizations (4 circuits).** For Category A power maps with higher-degree inverses we factor the inverse exponent into low-depth algebraic stages: the 6-bit Q_{2258} (x^5) inverse $x^{38} = y^2 \cdot N^4$ via the relative norm $N = y^9 \in \mathbb{F}_{2^3}$ (18/37/0 at $D = 2$); the 7-bit Gold (x^3) inverse x^{85} with $85 \equiv 5 \times 9 \times 16 \pmod{127}$, composed from our $D = 1$ circuits for y^5 and z^9 (29/149/15 at $D = 2$, the minimum depth for degree 4); and, with a 23-AND, $D = 1$ Karatsuba multiplier over $\mathbb{F}_2[z]/(z^7 + z + 1)$, the Kasami (x^{13}) inverse $y^{88} = (y^3 \cdot y^8)^8$ (37/139/0 at $D = 2$) and the x^7 inverse $y^{109} = ((y^5)^5)^{64} \cdot (y^5)^{32}$ (51/220/15 at $D = 3$, five fewer AND gates than forward).
4. **Tier 4: 5-bit Hodge-dual coset-span synthesis (4 circuits).** The remain-

Table 4: Comparison with published circuits (AND/XOR/ D , NOT in parentheses; Δ = published – ours). Baselines are from [BDM26] unless noted ([JBKK25, KBJ+26] count 2-fanin #NL $\in \{\text{AND, OR, NAND, NOR}\}$ and #L $\in \{\text{XOR, XNOR}\}$): ^adepth-1 minimizer [BDM25] (x^3 on $\mathbb{F}_{2^5}$ is 7/19/1; on 7-bit Gold x^3 , SLICE [KBJ+26] reaches 11/41/1 at LTP = 6 and 18/40/1, 20/37/1 at LTP = 5); ^bTier-4 inverse of Gold x^5 (C.5), archive circuit (C.8) is 14/31/2 (2) with S^{-1} at 7/26/1 (5); ^cexact x^{38} inverse (C.16) is 18/37/2; ^dsmallest random $(\delta, L) = (4, 16)$ quadratic 5-bit S-box in [BDM25] (5/20, 5/24, 5/24); ^eTier-4 inverse of χ_5 -class (C.7); ^fdepth-3 AES of [HB25] (42/97/3, LTP = 14), refined by SLICE [KBJ+26] to 42/94/3 (LTP = 13, $\Delta = 0/-7$) and 42/107/3 (LTP = 12, $\Delta = 0/+6$); ^gSLICE [KBJ+26] also reaches 20/49/2 (LTP = 7) and 12/39/3 (LTP = 10); ^hat $D = 4$, [JBKK25] reaches 33/85/4 (LTP = 23) and 34/81/4–34/100/4 (LTP $\in [15, 28]$); SLICE [KBJ+26] reaches 33/95/4 (LTP = 14); at $D \in \{5, 6\}$, [JBKK25] reaches 32/81.

n	S-box or profile	Cat.	(δ, L, d)	Published	Ours (S)	Ours (S^{-1})	Δ (S)	Class (SM)
5	KECCAK χ_5 [BDM25+20]	A	(8, 16, 2)	5/5/1 (5)	5/8/1 (1)	9/21/2 (3)	0/-3	$\sim_{\text{aff}} \chi_5$ (C.7)
5	Gold x^5	A	(2, 8, 2)	7/21/1 ^a	7/22/1	10/37/2 (4)	0/-1	$= x^5$ (C.5)
5	Gold class, SAC-ideal	A	(2, 8, 2)	7/21/1 ^a	7/21/1 (4)	10/41/2 (10)	0/0	$\sim_{\text{aff}} x^5$ (C.6)
5	YSH _{5,2} (quadratic, $\delta=4$)	B	(4, 16, 2)	5/20/1 ^d	5/12/1 (3)	9/32/2 (2)	0/+8	YSH _{5,2} (C.2)
5	χ_5^{-1} class	A	(8, 16, 3)	9/27/2	9/21/2 (3) ^e	5/8/1 (1)	0/+6	$\sim_{\text{aff}} \chi_5^{-1}$ (C.7 inv.)
5	YSH _{5,3} (cubic, $\delta=8$)	B	(8, 16, 3)	9/27/2	6/11/2	9/22/2 (2)	+3/+16	YSH _{5,3} (C.3)
5	YSH _{5,4} (quartic, $\delta=8$)	B	(8, 16, 4)	–	5/12/2 (1)	5/11/2 (2)	–	YSH _{5,4} (C.4)
5	inverse Gold x^{25}	A	(2, 8, 3)	10/37/2	10/37/2 (4)	7/22/1	0/0	$= x^{25}$ (C.5 inv.) ^b
5	inversion x^{30}	A	(2, 12, 4)	12/51/2	14/45/2	14/57/2	-2/+6	$= x^{15}$ (C.9)
5	DB ₁ profile [AL26]	B	(4, 12, 4)	9/36/2	11/20/2 (3)	13/38/2 (6)	-2/+16	YSH _{5,1} (C.1)
6	Q_{2258} (x^5 class)	A	(4, 16, 2)	8/25/1 ^a	8/34/1	18/37/2	0/-9	$\sim_{\text{aff}} x^5$ (C.16)
6	Q_{2258}^{-1} (x^{13} inv.)	A	(4, 16, 3)	11/42/2	9/15/3 (9)	9/15/3 (9)	+2/+27	$\sim x^{13}$ invol. (C.17) ^c
6	YSH _{6,1} (quartic, $\delta=4$)	B	(4, 16, 4)	–	9/27/3 (1)	9/27/3 (1)	–	YSH _{6,1} (C.10)
6	Dillon APN	A	(2, 16, 4)	17/75/2 ^g	28/68/2	28/78/2 (7)	-11/+7	$=$ Dillon (C.19)
6	inversion x^{62} (div.)	A	(4, 16, 5)	29/109/3	21/48/3	21/50/3	+8/+61	$\sim_{\text{aff}} x^{62}$ (C.18)
6	inversion x^{62} (tower)	A	(4, 16, 5)	21/63/3	21/48/3	21/50/3	0/+15	$\sim_{\text{aff}} x^{62}$ (C.18)
6	SPEEDY [LMR21]	B	(8, 24, 5)	20/51/3	12/31/4	21/57/3	+8/+20	YSH _{6,5} , $\delta=6$ (C.14)
7	Gold x^3	A	(2, 16, 2)	11/45/1 ^a	14/57/1	29/149/2 (15)	-3/-12	$= x^3$ (C.22)
7	Kasami x^{13}	A	(2, 16, 3)	27/115/2	30/114/2	37/139/2	-3/+1	$= x^{13}$ (C.23)
7	power map x^7	A	(6, 40, 3)	30/140/2	56/98/2	51/220/3 (15)	-26/+42	$= x^7$ (C.25)
7	inversion x^{126}	A	(2, 20, 6)	–	42/182/3 (6)	42/229/3 (6)	–	$\sim_{\text{aff}} x^{126}$ (C.24)
7	YSH _{7,1} (quartic, $\delta=4$)	B	(4, 32, 4)	–	15/70/2 (10)	30/158/3 (12)	–	YSH _{7,1} (C.20)
7	YSH _{7,2} (deg. 6, $\delta=4$)	B	(4, 20, 6)	–	51/76/3 (1)	34/164/3 (9)	–	YSH _{7,2} (C.21)
8	AES (FIPS-197)	A	(4, 32, 7)	33/84/4 ^h	36/99/4 (4)	36/95/4 (4)	-3/-15	$=$ AES (C.30)
8	AES class ($D=3$)	A	(4, 32, 7)	42/97/3 ^f	42/101/3 (4)	42/106/3 (4)	0/-4	$\sim_{\text{aff}}$ AES (C.29)
8	YSH _{8,2} (deg. 7, $\delta=10$)	B	(10, 72, 7)	–	28/61/6 (1)	28/84/6 (2)	–	YSH _{8,2} (C.27)
8	YSH _{8,3} (deg. 7, $\delta=14$)	B	(14, 72, 7)	–	18/71/5 (2)	18/77/6 (8)	–	YSH _{8,3} (C.28)

ing 5-bit S-boxes (YSH_{5,2}, YSH_{5,3}, Gold x^5 , and the χ_5 -class circuit) have cubic inverses with uniform spectrum $\mathcal{D}(S^{-1}) = [3^{31}]$; decomposing every cubic component via Hodge duality on $\mathbb{F}_2^5$ as $L_c(y) \cdot Q_c(y) \oplus R_c(y)$ with a linear form L_c and $\deg Q_c, \deg R_c \leq 2$ and sharing $r \in \{4, 5\}$ level-1 AND gates among five such components yields exact $D = 2$ circuits: 9/32/2, 9/22/2, 9/21/3 for the χ_5 -class inverse (6 fewer XOR gates than the 9/27/2 circuit of [BDM26]), and 10/37/4 for the Gold inverse x^{25} (matching the 10-AND, 37-XOR count of [BDM26] and saving 4 AND gates, at 6 more XOR gates, over our 14/31/2 archive circuit of SM C.8).

6.4 Comparison with Published Circuits

Table 4 compares our circuits with the best published implementations, including top-down polynomial division [BDM25, BDM26], the depth-3 AES inverter of Hadžić and Bloem [HB25], whole-S-box Boyar–Peralta optimization (ERNBP/EBPD) [JBKK25], and segmented low-depth synthesis (SLICE) [KBJ+26]. Category A rows compare circuits for the same function or proven affine class (or forward invariant class for the 6-bit cubic involution); Category B rows compare a YSH _{n,i} circuit for a *different* permutation with the same or better (δ, L, deg) .

6.5 Provenance and Stage Ablation of the Benchmark Circuits

Table 10 (SM B) traces each of the 30 benchmark circuits through the pipeline: the Stage-1 seed with its gate count, the certified netlist admitted to the archive after Stages 2–3, and the netlist printed in SM C after the Stage-4 pass (sub-DAG shaver and stage-preserving Paar CSE, Section 4.3). Three facts follow from it.

Where the circuits come from (Stage 1). Apart from the Category A rows that reproduce a catalogue or library function exactly (the seven ‘=’ rows of Table 2 and the Q_{2258} circuit, whose permutation is the published Q_{2258} itself), no printed lookup table equals an entry of the circuit library, the external 5-bit database, or the catalogue; in particular none of the 15 Category B permutations does. Fifteen of the 30 circuits start from a library circuit (source (2) of Section 4.1), eleven from a structural seed (source (3)), and four from an archive elite (source (1)). Among the 15 Category B circuits, eight descend from a library circuit that Stage 2 moved into a new EA class: $YSH_{6,2}$ from the cubic Q_{2256}^{-1} seed ($Q_{2258}^{-1} \sim_{\text{aff}} x^{38}$ inverse coset), $YSH_{7,1}$ from the fractional subfield tower, $YSH_{8,2}$ from the Canright AES tower [Can05], $YSH_{6,1}$ from a 9-AND 3-round Feistel library circuit (`golden_6b_row26`), $YSH_{5,3}$ from the KECCAK χ_5 circuit, and $YSH_{6,4}$ – $YSH_{6,6}$ from the 9-AND Q_{2258} seed; five descend from structural seeds: a power-map LUT node ($YSH_{5,1}$), the depth-1 quadratic template ($YSH_{5,2}$), the 3-round Feistel template ($YSH_{6,3}$), the unbalanced Feistel template ($YSH_{7,2}$), and CCZ-lifted inversion ($YSH_{8,1}$); and two descend from archive elites: $YSH_{5,4}$ from a 16-AND, $D = 2$, $\delta = 6$ elite of which Stage 2 kept five AND gates, and $YSH_{8,3}$ from $YSH_{8,2}$ via Stage-2 AND erasure (–10 AND, –1 level). Category A circuits start from the library circuit or power-map node of their own class, except the Kasami and χ_5 -class circuits (from archive elites). For $YSH_{5,3}$ and $YSH_{6,4}$ – $YSH_{6,6}$ the seed core remains inside the archived netlist (the depth-1 χ_5 core in $YSH_{5,3}$ and all nine depth-1 products of Q_{2258} in $YSH_{6,4}$ – $YSH_{6,6}$); $YSH_{5,2}$ is a depth-1 quadratic template instance whose $(\delta, L) = (4, 16)$ profile passed the grade gate directly.

What Stage 2 changes. Stage 2 is a function-changing search, so its effect is visible in the profile rather than the gate count. Its degree-raising shear mutation accounts for four Category B classes: one shear gate on the χ_5 circuit (5/10, $D = 1$) gives the cubic $YSH_{5,3}$ (6/11/0, $D = 2$; δ, L unchanged, $\mathcal{D}(S)$ from $[2^{31}]$ to $[2^{15}, 3^{16}]$), and one, two, and three chained shear gates on the Q_{2258} circuit (9/30, $D = 1$, $(\delta, L, \deg) = (4, 16, 2)$) give $YSH_{6,6}$ ((8, 24, 4), $D = 2$), $YSH_{6,4}$ ((8, 28, 5), $D = 3$), and $YSH_{6,5}$ ((6, 24, 5), $D = 4$)—each shear gate buys one degree for one AND gate and one level, at the price of δ . Other function-changing moves raised the degree of a depth-1 power-map node from 2 to 4 for $YSH_{5,1}$ (11/20, $D = 1 \rightarrow 11/20/3$, $D = 2$), re-wired `golden_6b_row26` into the quartic $YSH_{6,1}$ (9/27, $D = 3 \rightarrow 9/29/1$, $D = 3$, $\mathcal{D}(S) = [3^7, 4^{56}]$), turned the cubic Q_{2256}^{-1} seed into the quartic $YSH_{6,2}$ (26/65, $D = 2 \rightarrow 27/64/0$, $D = 3$, whose inverse unwinds one output shear onto the 8-AND quadratic Q_{2258} core), grew the unbalanced Feistel skeleton into $YSH_{7,2}$ (51/7, $D = 1 \rightarrow 51/77/1$, $D = 3$), reduced a 16-AND, $D = 2$ elite to the 5-AND $YSH_{5,4}$ (16/39 $\rightarrow$ 5/15/1, $D = 3$, degree 4), and moved the Canright AES tower via AND erasure and re-wiring to $YSH_{8,2}$ (36/136, $D = 4 \rightarrow 28/61/1$, $D = 6$: –8 AND, –75 XOR at +2 levels and $(\delta, L) = (10, 72)$). Where the seed already had the target function, Stage 2 and in-loop CSE trimmed linear gates (67 $\rightarrow$ 58 XOR for 6-bit inversion, 116 $\rightarrow$ 103 for $D = 3$ AES, 120 $\rightarrow$ 107 for $YSH_{8,1}$) or one AND gate (9 $\rightarrow$ 8 for Q_{2258} , 43 $\rightarrow$ 42 at –1 level for 7-bit inversion). Applied to $YSH_{8,2}$, AND erasure alone traces the certified degree-7 family of Table 8 (SM B) from 28 AND ($\delta = 10$) to 18 AND ($\delta = 14$, $YSH_{8,3}$). Within a Category A class, Stage 2 also reached the SAC-ideal x^5 representative of SM C.6 (7/21/4, $D = 1$). The Piling-Up/LSAP seeder (Section 4.2) contributed no benchmark circuit: its direct graduates lie in the 5-bit (8, 16, 4) family of $YSH_{5,4}$ or re-wire Gold x^3 and AES seeds without improving on Table 2.

What Stage 4 changes. The Stage-4 pass is function-preserving and depth-non-increasing

(every rewrite is re-simulated on all 2^n inputs). Over the 30 printed forward circuits it removed 182 XOR gates and one AND gate ($\text{YSH}_{6,3}$, $10 \rightarrow 9$) relative to the archived netlists, left 10 circuits unchanged, and reduced the depth of one circuit ($\text{YSH}_{5,4}$, $5/15/1$ at $D = 3 \rightarrow 5/12/1$ at $D = 2$, as its level-3 AND operand was available at level 1); the largest single reductions are $177 \rightarrow 98$ XOR for x^7 , $119 \rightarrow 99$ for FIPS-197 AES, $50 \rightarrow 34$ for Q_{2258} , and $43 \rightarrow 31$ for $\text{YSH}_{6,5}$. The same pass is applied to the 30 inverse netlists before synthesis.

6.6 Discussion

Co-synthesis vs. fixed-target minimization. When a designer specifies the profile $(\delta, L, \deg)$ rather than a fixed truth table, co-synthesis yields substantial non-linear savings: on SPEEDY’s profile, $\text{YSH}_{6,5}$ improves (δ, β) from $(8, 16)$ to $(6, 14)$ at $L = 24$ while cutting AND gates from 20 to 12 and XOR gates from 51 to 31; on the χ_5^{-1} profile $\text{YSH}_{5,3}$ saves 3 AND and 16 XOR gates; the 3-round χ_3 -Feistel involution with x^{13} ’s forward invariants saves 2 AND and 27 XOR gates against Q_{2258}^{-1} while being self-inverse; and $\text{YSH}_{6,1}$ reaches $(4, 16, 4)$ with $\mathcal{D}(S) = \mathcal{D}(S^{-1}) = [3^7, 4^{56}]$ at $9/27/1$ both ways. In Category A, co-synthesis pays when multiplicative depth governs masked cost: compared with $D = 4$ AES $(33/84/4)$ [BDM26], $33/85/4$ at $\text{LTP} = 23$ or $34/81/4$ at $\text{LTP} = 28$ [JBKK25], $33/95/4$ at $\text{LTP} = 14$ [KBJ+26] and $D \in \{5, 6\}$ circuits $(32/81)$ [JBKK25]), our $D = 3$ AES-class circuit $(42/101/4, \text{LTP} = 24)$ trades 9–10 AND gates for 1–3 fewer pipeline stages. Its non-linear core is the depth-3 inverter of Hadžić and Bloem [HB25] $(42/97/3, \text{LTP} = 14)$, which SLICE [KBJ+26] refines via whole-circuit eBPD to $42/94/3$ ($\text{LTP} = 13$) and $42/107/3$ (record $\text{LTP} = 12$); our contribution is the gate-exact netlist pair with its depth-3 inverse $(42/106/4)$ and masked-silicon evaluation. As pipelined $d = 2$ DOM-indep modules (Tables 3–9), moving from our $D = 4$ FIPS-197 circuit $(36/99/4)$ to $D = 3$ removes 90 forward sync flip-flops against 24 extra gadget flip-flops, cutting flip-flops from 282 to 216 (276 to 216 for S^{-1}) and masked area by 11.5%/8.9% ($33/84/4$ with 4 NOT [BDM26] needs 132 gadget flip-flops and $1,218.28 \mu\text{m}^2$ before sync registers by Eq. (8); composable HPC2 and HPC3/HPC3.1 likewise save 516 vs. 636 and 216 vs. 282 flip-flops, SM A.1); at stage delay 574.7 vs. 371.0 ps, three cycles reduce latency for clocks ≥ 575 ps. We also cut 6-bit tower inversion from $21/63$ to $21/48$ (S^{-1} : $21/50$) at $D = 3$, and reach SAC deviation 0 in the x^5 class $(7/21/4, D = 1)$, SM C.6). Conversely, on fixed targets, quadratic synthesis [BDM25], A^* MC search [JBK24], polynomial division [BDM26], and SLICE [KBJ+26] lead on Q_{2258} $(8/25$ vs. $8/34)$, 5-bit x^5 $(7/21$ vs. $7/22)$, 7-bit x^3 $(11/45)$ [BDM25] and $11/41, \text{LTP} = 6$ or $18/40, 20/37, \text{LTP} = 5$ [KBJ+26] vs. $14/57)$, Dillon’s APN $(17/75/2)$ [BDM26] and $20/49/2, \text{LTP} = 7$ or $12/39/3, \text{LTP} = 10$ [KBJ+26] vs. $28/68/2)$, 7-bit x^7 $(30/140$ vs. $56/98)$, 5-bit inversion $(12/51$ vs. $14/45)$, Kasami $(27/115$ vs. $30/114)$, and $D = 4$ AES $(33\text{--}34$ vs. 36 AND). The two paradigms are synergistic: YOSHAN finds low-AND Category B S-boxes where fixed classes are costly, and ERNBP/eBPD [JBKK25] and SLICE [KBJ+26] can polish its DAGs at invariant (AND, D) .

Algebraic structure of the Category B circuits. The $\text{YSH}_{n,i}$ of Table 2 follow the subspace-partition spectrum $\mathcal{D}(S) = [d_1^{2^k-1}, d_2^{2^n-2^k}]$ with $k = \dim V_{\leq d_1}$ (uniform $[d_2^{2^n-1}]$ for $\text{YSH}_{5,2}, \text{YSH}_{6,3}$; nested over a two- and three-step flag for $\text{YSH}_{5,4}$ and $\text{YSH}_{8,3}$) and split into two structural families. The *non-Feistel subspace-partition S-boxes* combine a wide quadratic first level with a short chain of degree-raising shears or transpositions: the hyperplane-split circuits ($k = n - 1$) reach high degree at low AND cost ($10+1$ gates at $D = 2$ for $\text{YSH}_{5,1}$, $14+1$ for $\text{YSH}_{7,1}$, $9+1$ for $\text{YSH}_{6,6}$, $9+2$ and $9+3$ for the degree-5 $\text{YSH}_{6,4}$ and $\text{YSH}_{6,5}$); $\text{YSH}_{6,2}$ ($k = 5$) and $\text{YSH}_{8,1}$ ($k = 1$) apply one output shear to a Q_{2258}^{-1} and an 8-bit inversion core to eliminate quadratic components without a Feistel structure, $\text{YSH}_{6,2}^{-1}$ unwinding to $9/37/0$ at $D = 2$; the six-round generalized Feistel network $\text{YSH}_{8,2}$,

evolved from the Canright AES tower (Section 6.5), trades $(\delta, L) = (10, 72)$ and two levels for 15 fewer AND gates than $\text{YSH}_{8,1}$ at degree 7, and its AND-erasure descendant $\text{YSH}_{8,3}$ gives up ten more AND gates and one level for $(\delta, L, \beta) = (14, 72, 38)$ —as a pipelined **DOM-indep** module the smallest 8-bit circuit of the set $(1, 142.74 \mu\text{m}^2, 72 \text{ gadget flip-flops})$. The *Feistel-type S-boxes* have symmetric forward/inverse cost: $\text{YSH}_{5,4}$ reaches degree 4 at the 5-AND count of χ_5 and depth $D = 2$ both ways (5/12/1 and 5/11/2), and the balanced 3-round networks $\text{YSH}_{6,1}$ and $\text{YSH}_{6,3}$ (3/3/3 AND gates per level) eliminate every quadratic component at 9 AND gates and $D = 3$ with $\delta = 4, L = 16$; $\text{YSH}_{6,3}$ (9/17/9, $D(S) = [3^{63}]$) also achieves $N_{\text{reconv}} = 0$ and a 55% shorter stage delay than $\text{YSH}_{6,1}$.

Boomerang uniformity and the price of 3-round Feistel seeds. The β column of Table 2 separates the two families sharply: the ten non-Feistel circuits have $\beta \in [6, 20]$ (comparable to AES at $\beta = 6$ and χ_5/SPEEDY at $\beta = 16$), the six-round $\text{YSH}_{8,2}$ and $\text{YSH}_{8,3}$ have $\beta = 22$ and 38, whereas $\text{YSH}_{6,1}$, $\text{YSH}_{6,3}$, and the χ_3 -Feistel involution reach $\beta = 2^n = 64$ and $\text{YSH}_{5,4}$ reaches $\beta = 32$. This is a classical property of 3-round Feistel permutations [LW14, CDL15, BPU16, BC18]: over two m -bit branches ($L_{i+1} = R_i, R_{i+1} = L_i \oplus F_{i+1}(R_i)$), the input difference $(\alpha, 0)$ leaves R_0 unchanged and shifts $R_1 = L_0 \oplus F_1(R_0)$ by α ; applying output difference $(0, \alpha)$ and decrypting swaps R_1 and $R_1 \oplus \alpha$, so both returning texts have right half $R_0 \oplus F_2(R_1) \oplus F_2(R_1 \oplus \alpha)$ and left halves differing by α . Hence $\text{BCT}((\alpha, 0), (0, \alpha)) = 2^{2m}$ for all $2^m - 1$ non-zero α —the 7 maximal entries of each 6-bit 3-round Feistel circuit. Because Eq. (6) indexes $(\delta, D, \deg, L)$ without $\beta(S)$, these circuits were retained for their profile and forward/inverse symmetry; a cipher using them must obtain boomerang resistance from its linear layer and round count, whereas the non-Feistel $\text{YSH}_{n,i}$ resist boomerang switches directly (Section 8).

7 Verification

Because the search transforms circuits via stochastic mutations, SMT sub-DAG re-synthesis, and CSE, soundness is enforced by an independent post-search verification pipeline: exported SystemVerilog netlists are certified against their lookup tables by exhaustive simulation and an SMT miter (Section 7.1); every metric and post-synthesis standard-cell netlist is independently recomputed and formally equivalence-checked (Section 7.2); the equivalence classification of Section 6.1 is solved directly on the lookup tables; and every forward and inverse circuit is audited for symbolic mask independence and intra-stage combinational support disjointness (Section 7.3).

7.1 Certification of the Exported Netlists

Before a candidate is archived, the engine emits its structural SystemVerilog netlist (Section 5.1) and certifies the emitted text rather than the in-memory DAG. Behavioral constructs (e.g., `case`) are rejected; the continuous AND/XOR/NOT assignments are evaluated bit-parallel on all 2^n input vectors, and the candidate is archived only if every output bit resolves, the resulting table is a permutation, and it matches the simulated truth table of the DAG. For $n \leq 8$ this exhaustive check is a complete proof of functional equivalence. As a second check independent of the simulator, the engine solves a Z3 miter [dMB08] $\exists x \in \mathbb{F}_2^n : G(x) \oplus \text{LUT}(x) \neq 0$ between the DAG G and the lookup table, requiring `unsat`; all 31,129 archived netlists of Section 6 carry both a passed certificate and an `unsat` miter.

7.2 Independent Recomputation and Verification Guarantees

For each of the 60 listings in SM C, an independent script parses the printed netlist, simulates it on all 2^n inputs, verifies $S(\text{LUT}_{S^{-1}}[y]) = y$ and $\text{LUT}_{S^{-1}}[S(x)] = x$, counts total and

per-level AND gates, and recomputes δ , L , $\deg$, $\deg^{-1}$, $\mathcal{D}(S)$, $\mathcal{D}(S^{-1})$, β , FP, and SAC (generating Table 2 and the “Ours” columns of Table 4). For Tables 3 and 9, every post-synthesis NanGate 45 nm netlist emitted by Yosys (60 unmasked and 60 DOM-indep modules) is checked for exact 1-to-1 cell counts and formally proven equivalent to its pre-synthesis SystemVerilog module via `equiv_status -assert` (combinational) and `miter -equiv` with `sat -verify -tempinduct` (sequential DOM-indep). Every Category A permutation in the archive is tested for exact affine equivalence against Table 6 with its (A, a, B, b) witness checked on all 2^n inputs, and all representatives in Table 11 are re-verified from their netlists. Every reported circuit is therefore a gate-exact AND/XOR/NOT netlist with certified gate counts, depth D , 45 nm silicon equivalence, cryptographic spectra, and (for entries marked ‘=’ or ‘ $\sim_{\text{aff}}$ ’) certified equality or affine equivalence to Table 6; evolutionary gate counts remain upper bounds, not lower bounds.

7.3 Masking-Related Design Rules and Glitch-Reconvergence Audit

The search budget rejects gate-level candidates whose fresh-randomness demand (counted conservatively at two bits per AND gate) exceeds 128 bits per pipeline stage or 1,280 bits per evaluation, and any linear gate with fan-out above 16, to bound capacitive coupling [DCBG⁺17, LBS19]; the cap is applied to the whole gate-level circuit (no mutated candidate with more than 64 explicit AND gates is accepted; seeds and lookup-table nodes lowered at export are exempt), and all 60 reported netlists use at most 35 AND gates per pipeline stage. Across all 30 forward and 30 inverse circuits we also audited two structural conditions governing first-order ($t = 1, d = 2$) glitch-extended probing security [GMK16, FGDP⁺18]. *Linear mask independence*: tracking the symbolic mask vector $\mu(w) \in \mathbb{F}_2^{n+N_{\text{AND}}}$ of every wire as a linear combination of the n input masks and the N_{AND} fresh gadget masks ($\mu(x_i) = e_i$, $\mu(u \oplus v) = \mu(u) \oplus \mu(v)$, $\mu(\neg u) = \mu(u)$, $\mu(c_k) = e_{n+k}$ for the output of the k -th gadget) confirms that across all 593 forward and 649 inverse AND gates (1,242 in total) no AND gate receives an unmasked operand ($\mu(a) = 0$) or two operands with identical steady-state masks ($\mu(a) = \mu(b)$). *Intra-stage glitch-extended support disjointness*: tracing the combinational fan-in cone $\text{Supp}_{\text{st}\ell}(w)$ of each AND-gate operand back to the stage- ℓ register boundary shows that 10 of the 30 forward circuits (YSH_{5,1}, YSH_{5,3}, the SAC-ideal Gold-class circuit, the inverse-Gold circuit, YSH_{6,3}, YSH_{6,4}, the 6-bit cubic involution, Dillon APN, YSH_{7,2}, and 7-bit x^7) and 3 of the 30 inverse circuits (YSH_{5,3}⁻¹, the χ_5 -class inverse, and the cubic involution) satisfy $\text{Supp}_{\text{st}\ell}(a) \cap \text{Supp}_{\text{st}\ell}(b) = \emptyset$ at every AND gate, so no single glitch-extended probe at $a_0 \wedge b_1$ observes both shares of any stage- ℓ variable or both domains of any preceding gadget; in the remaining circuits intra-stage linear reconvergence occurs at the $N_{\text{reconv}} > 0$ gates listed per circuit in Tables 3 and 9 (e.g., 3 of the 9 gates in YSH_{6,1} and $N_{\text{reconv}} \in \{9, 11, 12\}$ in the AES circuits). Both conditions are necessary single-stage checks, not a proof: the inner-domain products of stage- $(\ell - 1)$ gadgets are registered unblinded, so two gadgets that shared a stage- $(\ell - 1)$ input can still be observed jointly by one glitch-extended probe in stage ℓ . Removing reconvergence requires one of the mitigations of SM A.1, and full verification with SILVER [KSM20] or PROLEAD [MM22] is future work.

Artifact. SM C prints the forward and inverse lookup tables and netlists of all 30 S-boxes, and Table 10 records the Stage-1 seed and stage-by-stage gate counts of each circuit so every metric, invariant, and provenance claim can be re-derived independently.

8 Design Considerations and Future Directions

Equivalence classification and cipher-level cryptanalysis. While 1,761 of the 1,763 Category A archive permutations are proven affine equivalent or identical to Table 6, testing whether the remaining two (the 6-bit χ_3 -Feistel cubic involution class, proven

EA-inequivalent to x^{13} by its third-derivative rank spectrum in Section 6.1) are CCZ-equivalent to x^{13} , and classifying Category B permutations against broader composite families [Kal21], are natural follow-ups. For the hyperplane-split Category B circuits ($\mathcal{D}(S) = [2^{2^{n-1}-1}, d_2^{2^{n-1}}]$ for $\text{YSH}_{5,1}$, $\text{YSH}_{5,3}$, $\text{YSH}_{6,4}$ – $\text{YSH}_{6,6}$, $\text{YSH}_{7,1}$ – $\text{YSH}_{7,2}$, and nested $[2^7, 3^8, 4^{16}]$ for $\text{YSH}_{5,4}$, whose $2^{n-1} - 1$ lower-degree components form a hyperplane $H = \langle v^* \rangle^\perp$), the SPN diffusion matrix M must ensure $M^\top$ does not map active next-round input masks entirely into H (preventing multi-round higher-order differential, integral, or algebraic trails from staying in H); the 3-round Feistel $\text{YSH}_{6,1}$ and $\text{YSH}_{6,3}$ have minimum component degree 3 in all directions but, like any 3-round Feistel permutation [CDL15, BPU16, BC18], $\beta = 2^n$ (Section 6.6), as does $\text{YSH}_{5,4}$ ($\beta = 32$), so boomerang resistance must come from the linear layer and round count. Adding $\beta(S)$ to Eq. (6) with 4-round Feistel/Misty seeds is a direct extension (non-Feistel $\text{YSH}_{n,i}$ satisfy $\beta \leq 20$).

Linear-layer polishing, physical place-and-route, and co-evolution. Coupling YOSHAN’s archive with linear-network minimizers [BP10, ME19, Sto16] and whole-S-box eRNPB/eBPD [JBKK25] and segmented SLICE [KBJ⁺26] passes can further compress cross-stage XOR cones and LTP at invariant (AND, D), and SAT technology mapping [JCL⁺25] can exploit multi-input cells in unmasked flows; place-and-route parasitics, SILVER/PROLEAD [KSM20, MM22], and joint (S, S^{-1}) fitness are next.

Continuous-to-discrete relaxation. In the Piling-Up module (Eq. (5)), interior wires with $p_m \approx 0.5$ drive $\prod_m (1 - 2p_m) \rightarrow 0$ exponentially with depth, shrinking the relaxed Walsh spectrum before binarization, while Hungarian projection crosses sharp algebraic boundaries; straight-through evaluation, interleaved Hungarian/Sinkhorn steps, and a joint differential-linear objective via the Chabaud–Vaudenay identity [CV95, CCZ98, Car21] $\sum_{\alpha,v} W_S(\alpha, v)^4 = 2^{2n} \sum_{a,b} \text{DDT}(a, b)^2$ remedy this.

9 Conclusion

Under masking, S-box area and randomness scale with AND count and pipeline latency with AND depth, confining prior automated synthesis to depth-1 quadratics or fixed-table minimization. YOSHAN co-synthesizes the permutation and its circuit in one quality-diversity loop uniting structural algebraic seeding, cryptographically guided DAG evolution, differentiable Piling-Up relaxation with Hungarian LSAP projection, and multi-level circuit optimization with four-tier inverse compilation. Across $n \in \{5, 6, 7, 8\}$ it produced 31,129 certified netlists in 56 profiles, with all 60 forward and inverse circuits verified, synthesized in 45 nm CMOS (unmasked and $d = 2$ DOM-indep), and traced in Table 10. In Category A, co-synthesis yields a gate-exact realization of the depth-3 AES-class architecture of Hadžić and Bloem (42/101/4 forward, 42/106/4 inverse at $D = 3$) saving 66/60 flip-flops and 11.5%/8.9% DOM-indep area at 25% lower latency than our $D = 4$ FIPS-197 circuit, and a 21/48/0 6-bit inversion-class circuit, while on fixed targets polynomial division [BDM25, BDM26], whole-S-box eBPD [JBKK25], and SLICE [KBJ⁺26] lead in linear gates and combinational depth. In Category B, relaxing the fixed-table constraint uncovers 15 EA classes outside the reference catalogue with lower non-linear cost in two families: non-Feistel subspace-partition S-boxes with $\beta \in [6, 20]$ —the 12-AND degree-5 $\text{YSH}_{6,5}$ ($\delta = 6, L = 24, \beta = 14$), the 11-AND, $D = 2$ quartic $\text{YSH}_{5,1}$ ($\delta = 4, L = 12, \beta = 6$), the 15-AND, $D = 2$ 7-bit quartic $\text{YSH}_{7,1}$, the 6-bit quartic $\text{YSH}_{6,2}$ with its 9-AND, $D = 2$ inverse, and the degree-7 8-bit $\text{YSH}_{8,2}$ and $\text{YSH}_{8,3}$ with 28 and 18 AND gates—and Feistel-type S-boxes with symmetric forward/inverse cost and $\beta = 2^n$: the 9-AND, $D = 3$ 3-round Feistel S-boxes $\text{YSH}_{6,1}$ and $\text{YSH}_{6,3}$ and the 5-bit quartic $\text{YSH}_{5,4}$ (5 AND, $D = 2$ both ways).

Acknowledgments

The authors thank Google for providing the computational infrastructure used to conduct this research. R. H., B. A., and I. L. were partly supported by the Israel Science Foundation (ISF), grants 2569/21 and 3276/25.

References

- [AL26] Binyamin Alony and Itamar Levi. Optimal 5-bit S-boxes dataset. SelecSys Labs, Faculty of Engineering, Bar-Ilan University, <https://github.com/Selecsys-Lab/optimal-5bit-sboxes>, 2026.
- [BBK⁺13] Begül Bilgin, Andrey Bogdanov, Miroslav Knežević, Florian Mendel, and Qingju Wang. Fides: Lightweight authenticated cipher with side-channel resistance for constrained hardware. In *Cryptographic Hardware and Embedded Systems — CHES 2013*, volume 8086 of *LNCS*, pages 142–158. Springer, 2013.
- [BBS⁺17] Dušan Božilov, Begül Bilgin, and Hacı Ali Şahin. A note on 5-bit quadratic permutations’ classification. *IACR Transactions on Symmetric Cryptology*, 2017(1):398–404, 2017.
- [BC18] Christina Boura and Anne Canteaut. On the boomerang uniformity of cryptographic Sboxes. *IACR Transactions on Symmetric Cryptology*, 2018(3):290–310, 2018.
- [BCDC11] Christina Boura, Anne Canteaut, and Christophe De Cannière. Higher-order differential properties of Keccak and Luffa. In *Fast Software Encryption — FSE 2011*, volume 6733 of *LNCS*, pages 252–269. Springer, 2011.
- [BDCBP03] Alex Biryukov, Christophe De Cannière, An Braeken, and Bart Preneel. A toolbox for cryptanalysis: Linear and affine equivalence algorithms. In *Advances in Cryptology — EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 33–50. Springer, 2003.
- [BDD⁺23] Yanis Belkheyar, Joan Daemen, Christoph Dobraunig, Santosh Ghosh, and Shahram Rasoolzadeh. BipBip: A low-latency tweakable block cipher with small dimensions. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(1):326–368, 2023.
- [BDM25] Marie Bolzer, Sébastien Duval, and Marine Minier. A new tool to find lightweight (AND, XOR) implementations of quadratic vectorial Boolean functions up to dimension 9. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 73(1):478–491, 2025.
- [BDM26] Marie Bolzer, Sébastien Duval, and Marine Minier. A tool for lightweight (AND, XOR) implementations of large-degree S-boxes. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2026(2):605–629, 2026.
- [BDMD⁺20] Begül Bilgin, Lauren De Meyer, Sébastien Duval, Itamar Levi, and François-Xavier Standaert. Low AND depth and efficient inverses: A guide on S-boxes for low-latency masking. *IACR Transactions on Symmetric Cryptology*, 2020(1):144–184, 2020.

- [BDMW10] K. A. Browning, J. F. Dillon, M. T. McQuistan, and A. J. Wolfe. An APN permutation in dimension six. In *Finite Fields: Theory and Applications (FQ9)*, volume 518 of *Contemporary Mathematics*, pages 33–42. American Mathematical Society, 2010.
- [BDPVA11] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. The Keccak reference, version 3.0. <https://keccak.team/files/Keccak-reference-3.0.pdf>, 2011.
- [BFP16] Joan Boyar, Magnus Gausdal Find, and René Peralta. On various nonlinearity measures for Boolean functions. *Cryptography and Communications*, 8(3):313–330, 2016.
- [BM10] Robert Brayton and Alan Mishchenko. ABC: An academic industrial-strength verification tool. In *Computer Aided Verification, 22nd International Conference (CAV 2010)*, volume 6174 of *LNCs*, pages 24–40. Springer, 2010.
- [BP10] Joan Boyar and René Peralta. A new combinational logic minimization technique with applications to cryptology. In *Experimental Algorithms, 9th International Symposium (SEA 2010)*, volume 6049 of *LNCs*, pages 178–189. Springer, 2010.
- [BP12] Joan Boyar and René Peralta. A small depth-16 circuit for the AES S-box. In *Information Security and Privacy Research (SEC 2012)*, volume 376 of *IFIP Advances in Information and Communication Technology*, pages 287–298. Springer, 2012.
- [BPU16] Alex Biryukov, Léo Perrin, and Aleksei Udovenko. Reverse-engineering the S-box of Streebog, Kuznyechik and STRIBOBr1. In *Advances in Cryptology — EUROCRYPT 2016, Part I*, volume 9665 of *LNCs*, pages 372–402. Springer, 2016.
- [BS91] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, 4(1):3–72, 1991.
- [Can05] David Canright. A very compact S-box for AES. In *Cryptographic Hardware and Embedded Systems — CHES 2005*, volume 3659 of *LNCs*, pages 441–455. Springer, 2005.
- [Car21] Claude Carlet. *Boolean Functions for Cryptography and Coding Theory*. Cambridge University Press, 2021.
- [CCZ98] Claude Carlet, Pascale Charpin, and Victor Zinoviev. Codes, bent functions and permutations suitable for DES-like cryptosystems. *Designs, Codes and Cryptography*, 15(2):125–156, 1998.
- [CDL15] Anne Canteaut, Sébastien Duval, and Gaëtan Leurent. Construction of lightweight S-boxes using Feistel and MISTY structures. In *Selected Areas in Cryptography — SAC 2015*, volume 9566 of *LNCs*, pages 373–393. Springer, 2015.
- [CGLS21] Gaëtan Cassiers, Benjamin Grégoire, Itamar Levi, and François-Xavier Standaert. Hardware private circuits: From trivial composition to full verification. *IEEE Transactions on Computers*, 70(10):1677–1690, 2021.

- [CHP⁺18] Carlos Cid, Tao Huang, Thomas Peyrin, Yu Sasaki, and Ling Song. Boomerang connectivity table: A new cryptanalysis tool. In *Advances in Cryptology — EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 683–714. Springer, 2018.
- [CV95] Florent Chabaud and Serge Vaudenay. Links between differential and linear cryptanalysis. In *Advances in Cryptology — EUROCRYPT '94*, volume 950 of *LNCS*, pages 356–365. Springer, 1995.
- [DCBG⁺17] Thomas De Cnudde, Begül Bilgin, Benedikt Gierlichs, Ventzislav Nikov, Svetla Nikova, and Vincent Rijmen. Does coupling affect the security of masked implementations? In *Constructive Side-Channel Analysis and Secure Design — COSADE 2017*, volume 10348 of *LNCS*, pages 1–18. Springer, 2017.
- [DEMS21] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2: Lightweight authenticated encryption and hashing. *Journal of Cryptology*, 34(3):33, 2021.
- [dMB08] Leonardo Mendonça de Moura and Nikolaj S. Bjørner. Z3: An efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2008)*, volume 4963 of *LNCS*, pages 337–340. Springer, 2008.
- [DMB19] Lauren De Meyer and Begül Bilgin. Classification of balanced quadratic functions. *IACR Transactions on Symmetric Cryptology*, 2019(2):169–192, 2019.
- [DR01] Joan Daemen and Vincent Rijmen. The wide trail design strategy. In *Cryptography and Coding*, volume 2260 of *LNCS*, pages 222–238. Springer, 2001.
- [DS09] Itai Dinur and Adi Shamir. Cube attacks on tweakable black box polynomials. In *Advances in Cryptology — EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 278–299. Springer, 2009.
- [FGDP⁺18] Sebastian Faust, Vincent Grosso, Santos Merino Del Pozo, Clara Paglialonga, and François-Xavier Standaert. Composable masking schemes in the presence of physical defaults & the robust probing model. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(3):89–120, 2018.
- [GMK16] Hannes Groß, Stefan Mangard, and Thomas Korak. Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order. In *Proceedings of the 2016 ACM Workshop on Theory of Implementation Security (TIS '16)*, page 3. ACM, 2016.
- [GR16] Dahmun Goudarzi and Matthieu Rivain. On the multiplicative complexity of Boolean functions and bitsliced higher-order masking. In *Cryptographic Hardware and Embedded Systems — CHES 2016*, volume 9813 of *LNCS*, pages 457–478. Springer, 2016.
- [HB25] Vedad Hadžić and Roderick Bloem. Efficient and composable masked AES S-box designs using optimized inverters. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2025(1):656–683, 2025.
- [JBK24] Yongjin Jeon, Seungjun Baek, and Jongsung Kim. Towards finding S-box circuits with optimal multiplicative complexity. *IEEE Transactions on Computers*, 73(8):2036–2050, 2024.

- [JBKK25] Yongjin Jeon, Seungjun Baek, Giyoon Kim, and Jongsung Kim. A framework for generating S-box circuits with Boyar–Peralta algorithm-based heuristics, and its applications to AES, SNOW3G, and Saturnin. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2025(1):586–631, 2025.
- [JCL⁺25] Chenhao Jia, Tingting Cui, Qing Ling, Yan He, Kai Hu, Yu Sun, and Meiqin Wang. How small can S-boxes be? *IACR Transactions on Symmetric Cryptology*, 2025(1):592–622, 2025. Cryptology ePrint Archive, Paper 2025/386; <https://github.com/ChenhaoJia/Area-Optimized-Implementation-for-S-box>.
- [JK97] Thomas Jakobsen and Lars R. Knudsen. The interpolation attack on block ciphers. In *Fast Software Encryption — FSE 1997*, volume 1267 of *LNCS*, pages 28–40. Springer, 1997.
- [Kal21] Nikolay S. Kaleyski. Invariants for EA- and CCZ-equivalence of APN and AB functions. *Cryptography and Communications*, 13(6):945–969, 2021.
- [KBJ⁺26] Giyoon Kim, Seungjun Baek, Yongjin Jeon, Vedad Hadžić, and Jongsung Kim. Bitsliced segment-based search technique for low-depth and hardware-efficient S-box circuits. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2026. Cryptology ePrint Archive, Paper 2026/726, <https://eprint.iacr.org/2026/726>; <https://github.com/sboxcircuit/slice>.
- [KM22] David Knichel and Amir Moradi. Low-latency hardware private circuits. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS 2022)*, pages 1799–1812. ACM, 2022.
- [Knu95] Lars R. Knudsen. Truncated and higher order differentials. In *Fast Software Encryption — FSE 1994*, volume 1008 of *LNCS*, pages 196–211. Springer, 1995.
- [KPF⁺24] Alexandr Kuznetsov, Nikolay Poluyanenko, Emanuele Frontoni, Sergey Kandi, and Olha Pieshkova. Optimized simulated annealing for efficient generation of highly nonlinear S-boxes. *Soft Computing*, 28(5):3905–3920, 2024.
- [KSM20] David Knichel, Pascal Sasdrich, and Amir Moradi. SILVER — statistical independence and leakage verification. In *Advances in Cryptology — ASIACRYPT 2020, Part I*, volume 12491 of *LNCS*, pages 787–816. Springer, 2020.
- [Kuh55] Harold W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1–2):83–97, 1955.
- [Lai94] Xuejia Lai. Higher order derivatives and differential cryptanalysis. In *Communications and Cryptography: Two Sides of One Tapestry*, pages 227–233. Springer, 1994.
- [LBS19] Itamar Levi, Davide Bellizia, and François-Xavier Standaert. Reducing a masked implementation’s effective security order with setup manipulations: And an explanation based on externally-amplified couplings. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(2):293–317, 2019.

- [LMMR21] Gregor Leander, Thorben Moos, Amir Moradi, and Shahram Rasoolzadeh. The SPEEDY family of block ciphers: Engineering an ultra-low-latency cipher from gate level for secure processor architectures. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(4):510–545, 2021.
- [LP07] Gregor Leander and Axel Poschmann. On the classification of 4-bit S-boxes. In *Arithmetic of Finite Fields (WAIFI 2007)*, volume 4547 of *LNCS*, pages 159–176. Springer, 2007.
- [LW14] Yongqiang Li and Meiqin Wang. Constructing S-boxes for lightweight cryptography with Feistel structure. In *Cryptographic Hardware and Embedded Systems — CHES 2014*, volume 8731 of *LNCS*, pages 127–146. Springer, 2014.
- [Mat94] Mitsuru Matsui. Linear cryptanalysis method for DES cipher. In *Advances in Cryptology — EUROCRYPT ’93*, volume 765 of *LNCS*, pages 386–397. Springer, 1994.
- [MC15] Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *CoRR*, abs/1504.04909, 2015.
- [ME19] Alexander Maximov and Patrik Ekdahl. New circuit minimization techniques for smaller and faster AES SBoxes. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(4):91–125, 2019.
- [MM22] Nicolai Müller and Amir Moradi. PROLEAD — a probing-based hardware leakage detection tool. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(4):311–348, 2022.
- [MMSS19] Thorben Moos, Amir Moradi, Tobias Schneider, and François-Xavier Standaert. Glitch-resistant masking revisited: or why proofs in the robust probing model are needed. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(2):256–292, 2019.
- [Nan11] Nangate Inc. NanGate FreePDK45 open cell library. OpenROAD-flow-scripts distribution of NangateOpenCellLibrary_typical.lib, 2011.
- [Nat23] National Institute of Standards and Technology. Advanced encryption standard (AES). Technical Report FIPS 197, NIST, 2023.
- [NRR06] Svetla Nikova, Christian Rechberger, and Vincent Rijmen. Threshold implementations against side-channel attacks and glitches. In *Information and Communications Security (ICICS 2006)*, volume 4307 of *LNCS*, pages 529–545. Springer, 2006.
- [Nyb94] Kaisa Nyberg. Differentially uniform mappings for cryptography. In *Advances in Cryptology — EUROCRYPT ’93*, volume 765 of *LNCS*, pages 55–64. Springer, 1994.
- [Paa97] Christof Paar. Optimized arithmetic for Reed-Solomon encoders. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT 1997)*, page 250. IEEE, 1997.
- [PČR16] Stjepan Picek, Marko Čupić, and Leon Rotim. A new cost function for evolution of S-boxes. *Evolutionary Computation*, 24(4):695–718, 2016.
- [Ras22] Shahram Rasoolzadeh. Low-latency boolean functions and bijective S-boxes. *IACR Transactions on Symmetric Cryptology*, 2022(3):403–447, 2022.

- [Sha49] Claude E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4):656–715, 1949.
- [SMTM01] Akashi Satoh, Sumio Morioka, Kohji Takano, and Seiji Munetoh. A compact Rijndael hardware architecture with S-box optimization. In *Advances in Cryptology — ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 239–254. Springer, 2001.
- [Sto16] Ko Stoffelen. Optimizing S-box implementations for several criteria using SAT solvers. In *Fast Software Encryption — FSE 2016*, volume 9783 of *LNCS*, pages 140–160. Springer, 2016.
- [SY+02] Takeshi Shimoyama, Hitoshi Yanami, Kazuhiro Yokoyama, Masahiko Takenaka, Kouichi Itoh, Jun Yajima, Naoya Torii, and Hidema Tanaka. The block cipher SC2000. In *Fast Software Encryption — FSE 2001*, volume 2355 of *LNCS*, pages 312–327. Springer, 2002.
- [Tod15] Yosuke Todo. Structural evaluation by generalized integral property. In *Advances in Cryptology — EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 287–314. Springer, 2015.
- [TSADM19] Eleonora Testa, Mathias Soeken, Luca Gaetano Amarù, and Giovanni De Micheli. Reducing the multiplicative complexity in logic networks for cryptography and security applications. In *Proceedings of the 56th Annual Design Automation Conference 2019 (DAC '19)*, pages 74:1–74:6. ACM, 2019.
- [WGK13] Clifford Wolf, Johann Glaser, and Johannes Kepler. Yosys — a free Verilog synthesis suite. In *Proceedings of the 21st Austrian Workshop on Microelectronics (Austrochip 2013)*, 2013.

Supplementary Material (SM)

Yoshan: Algebraic-Circuit Co-Synthesis and Search Methodology of Hardware-Efficient S-Boxes

This Supplementary Material (SM) is organized into three sections: **SM A** details the masked AND gadgets, PINI composability, and differentiable Piling-Up/LSAP hyper-parameters; **SM B** provides the campaign configurations, reference catalogue, archive breakdown across all 56 profiles, S^{-1} silicon synthesis, provenance table, AND-erasure family, and algebraic inverse derivations; and **SM C** prints the complete hexadecimal lookup tables and verified gate-level netlists for all 30 forward (S) and 30 inverse (S^{-1}) benchmark S-boxes.

A Methodology Details

A.1 Masked AND Gadget, Composability, and Mitigation Costs

Figure 3 shows the first-order ($t = 1, d = 2$) domain-oriented masking (DOM-indep) AND gadget [GMK16] synthesized in Section 5.2. The inner-domain products $a_0 \wedge b_0$ and $a_1 \wedge b_1$ need no randomness, while the cross-domain products $a_0 \wedge b_1$ and $a_1 \wedge b_0$ are blinded with one fresh random bit r and registered in four flip-flops (DFF_X1) before each domain compresses its two terms into an output share, preventing glitches from combining shares of different domains. When the two inputs (a, b) of an AND gate share a common variable in their intra-stage combinational fan-in cone ($\text{Supp}_{\text{stl}}(a) \cap \text{Supp}_{\text{stl}}(b) \neq \emptyset$), a glitch propagating through both linear cones into $a_0 \wedge b_1$ can observe both shares of that common variable. Three mitigations exist; we state their $d = 2$ per-gadget costs on the NanGate cells of Section 5.2 and their analytical effect on our AES circuits: (i) Resharing one operand (DOM-dep [GMK16]) costs +1 random bit, +2 XOR2_X1, and +2 DFF_X1 per reconvergent gate and inserts one register cut on that path (a two-cycle multiplication [GMK16, §3.3]), which in a synchronous pipeline also requires 2 DFF_X1 per parallel stage wire to align the stage; moreover, DOM gadgets carry no general multi-stage PINI composability guarantee [CGLS21, MMSS19]. (ii) The glitch-robust PINI HPC2 gadget [CGLS21, Alg. 5] keeps one random bit per gate (N_{AND} in total) at a two-cycle latency ($2D$ pipeline stages, doubling the synchronization registers of Eq. (8) to $4N_{\text{sync}}$): at $d = 2$ it uses $2d^2 - d = 6$ AND2_X1, $d = 2$ INV_X1, $2d(d - 1) = 4$ XOR2_X1 (or 6 XOR2_X1 when counting explicit fresh-mask blinding on both cross-domain paths), and 8 DFF_X1 per gate (or 10 DFF_X1 when both operands a and b are independently latched in the first cycle rather than sharing the stage-input synchronization registers). Under composable $d = 2$ HPC2, moving from our $D = 4$ FIPS-197 AES circuit (36/99/4, $N_{\text{sync}} = 69$) to our $D = 3$ AES-class circuit (42/101/4, $N_{\text{sync}} = 24$) reduces pipeline latency from 8 to 6 cycles, synchronization flip-flops from $4 \times 69 = 276$ to $4 \times 24 = 96$, and total flip-flops from $360 + 276 = 636$ to $420 + 96 = 516$ under the 10-FF convention (3,802.24 vs. 3,341.00 μm^2 , saving 120 DFF_X1 and 12.1% area) or from $288 + 276 = 564$ to $336 + 96 = 432$ under the 8-FF convention (3,368.13 vs. 2,885.18 μm^2 , saving 132 DFF_X1 and 14.3% area). (iii) Single-cycle PINI gadgets—HPC3 [KM22] (4 AND2_X1, 6 XOR2_X1, 4 DFF_X1) and Hadžić and Bloem’s HPC3.1 [HB25] (4 AND2_X1, 4 XOR2_X1, 4 DFF_X1)—restore latency D at 2 random bits per gate ($2N_{\text{AND}}$ in total) with the exact register count $N_{\text{DFF}} = 4N_{\text{AND}} + 2N_{\text{sync}}$ of Eq. (8): our $D = 3$ AES circuit then uses 216 DFF_X1 at 3 cycles (1,748.15 μm^2 with HPC3.1, 1,882.21 μm^2 with HPC3) versus 282 DFF_X1 at 4 cycles for $D = 4$ FIPS-197 AES (1,976.38 and 2,091.29 μm^2), saving 66 DFF_X1 and 11.5%/10.0%

area. Thus the register and latency advantage of reducing D from 4 to 3 holds unchanged under composable PINI masking.

A.2 Differentiable Template Module: Terms and Hyper-Parameters

This subsection lists the exact definitions behind Eq. (5) (Section 4.2). *Parameterization.* Round wire permutations are parameterized by $P^{(r)} = \text{softmax}(W_{\text{perm}}^{(r)}/\tau)$ and branch polynomials by temperature-sharpened sigmoids of weights $W_{\text{poly}}^{(r)}$; the relaxed Walsh spectrum $\widetilde{W}_S(\alpha, v) = H_{2^n}(1 - 2\widetilde{S}_v(x))$ is evaluated for every relaxed component $\widetilde{S}_v$ ($v \neq 0$) on all 2^n inputs. After gradient descent (standalone, or as a 100-epoch homotopy $W_{\text{child}} = \alpha W_A + (1 - \alpha)W_B$ between two parents' weights when the module acts as the crossover operator) the weights are projected to a discrete circuit: one LSAP per round permutation turns the row-stochastic softmax matrix into a permutation matrix (a vertex of the Birkhoff polytope), and K sequential masked rectangular LSAP passes select the monomials of each branch polynomial. *Templates.* The Keccak-like SPN template uses a fixed invertible diffusion matrix $M_{\text{diff}} \in \text{GL}(n, \mathbb{F}_2)$ (for $n = 7$ the circulant matrix with first row $(1, 1, 1, 1, 1, 0, 0)$); the Misty template uses $(n - 3, 3)$ or balanced $(\lceil n/2 \rceil, \lfloor n/2 \rfloor)$ branch splits. Branch polynomials are parameterized by temperature-sharpened sigmoids $\sigma(\kappa(\tau)(\sigma(W_{\text{poly}}^{(r)}) - \frac{1}{2}))$ with $\kappa(\tau) = \kappa_0 + \kappa_1 \max(0, 1 - 2\tau)$, $(\kappa_0, \kappa_1) = (5, 25)$ for the SPN and $(4, 16)$ for the Misty template. The relaxed component for an output mask $v \neq 0$ is $\widetilde{S}_v(x) = \text{XOR}(\{\widetilde{S}_j(x) : v_j = 1\})$. *Loss terms.* $\mathcal{L}_{\text{SAC}} = \sum_{i,k} (2^{-n} \sum_x \text{XOR}(\widetilde{S}_k(x), \widetilde{S}_k(x \oplus e_i)) - \frac{1}{2})^2$; $\mathcal{L}_{\text{sym}} = \sum_r \sum_i (P_{ii}^{(r)})^2$ (SPN template only; it penalizes fixed points of the round wire permutation); $\mathcal{L}_{\text{ent}} = c_{\text{ent}} \cdot \text{mean}_{x,k} \widetilde{S}_k(x)(1 - \widetilde{S}_k(x))$ with $c_{\text{ent}} = 300$ for the SPN and 25 for the Misty template; $\mathcal{L}_{\ell_1} = c \sum \sigma(W_{\text{poly}})$ with $c = 3 \cdot 10^{-3}$ for the SPN and $3 \cdot 10^{-4}$ for the Misty template; and $\lambda(\tau) = \max(0.01, 1 - 2\tau)$. *Schedule.* $\tau_0 = 0.5$ and $\tau_{t+1} = \max(0.01, \rho \tau_t)$ with $\rho = 0.02^{1/(0.9E)}$ over E epochs; Adam with a cosine-decayed learning rate of 0.12 for the SPN/Feistel and $2 \cdot 10^{-3}$ for the Misty template, gradient norm clipped at 5. The in-episode Misty variant evaluates the spectral term over the coordinate and adjacent-pair components only, with exponent 8 instead of 32, for speed. *Projection.* Each round permutation is obtained by one LSAP on the row-stochastic softmax matrix; each branch polynomial by K sequential rectangular LSAP passes on its sigmoid weight matrix (candidate monomials $\times$ branch outputs), where the entries selected in a pass are masked with -10^9 before the next pass so that every pass adds one disjoint assignment ($K = 5$; $K = 2$ for Misty branches; bijectivity is structural for additive $L \leftarrow L \oplus F(R)$ branch updates and checked at runtime by Stage 3 for SPN projections).

B Campaign Suite, Archive Breakdown, and Reference Tables

This section of the SM lists the campaign configurations and targets (Table 5), the reference catalogue of Section 6.1 (Table 6), the archive summary (Table 7) and its profile-by-profile breakdown (Table 11), the synthesis of the 30 inverse circuits (Table 9), the provenance of the 30 benchmark circuits (Table 10), the AND-erasure family of YSH_{8,2} (Table 8), and the derivations behind the inverse circuits of Section 6.3.

B.1 Multi-Orbit Invariant Collisions

Five profiles of Table 11 contain several cyclotomic cosets with identical forward invariants (the three EA invariants of Section 6.1 cannot separate power maps that share the differential and Walsh value histograms and the degree spectrum); the exact affine-equivalence solver

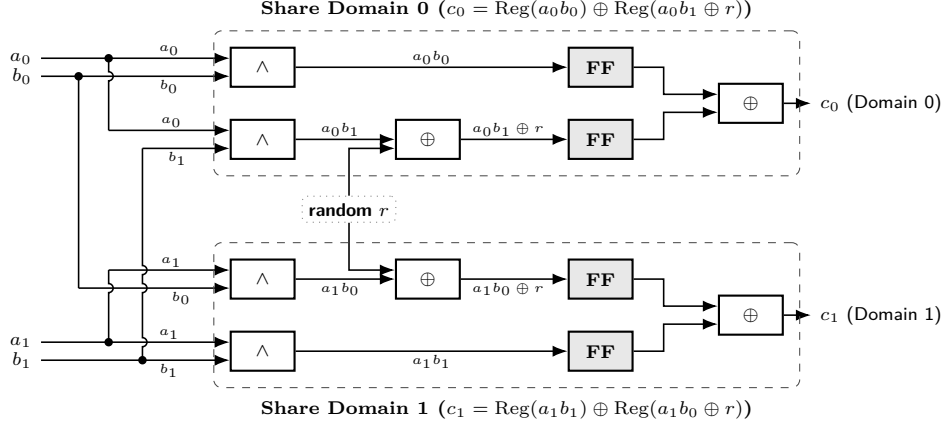


Figure 3: First-order ($t = 1, d = 2$) DOM-indep AND gadget [GMK16]. The cross-domain products $a_0 \wedge b_1$ and $a_1 \wedge b_0$ are blinded with a fresh random bit r , and all four products are stored in registers (FF) before each domain compresses its two terms into an output share.

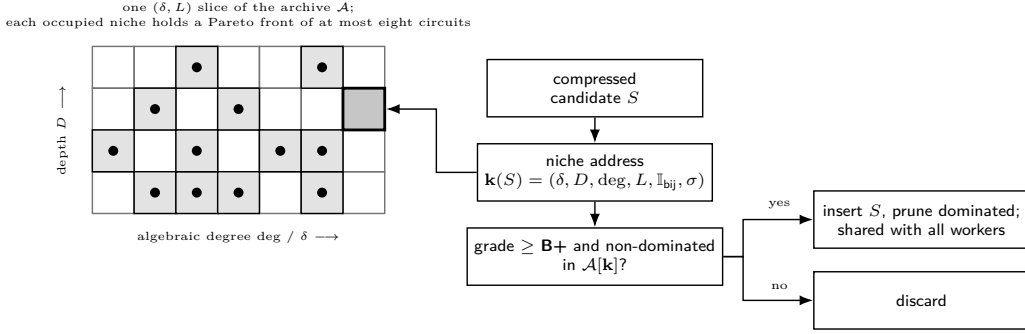


Figure 4: The quality-diversity archive (Section 4.4). A candidate is addressed to its niche by Eq. (6) and kept if it has grade **B+** or better and is not Pareto-dominated there.

separates them. For $n = 5$, the quadratic APN cosets x^3 and x^5 in $(2, 8, 2)$ split the 81 archive permutations into $69 \sim_{\text{aff}} x^3$ and $12 \sim_{\text{aff}} x^5$ (the printed Gold circuit of SM C.5 is the table of x^5 itself and the SAC-ideal circuit of SM C.6 is affine equivalent to it, not to x^3), and the inverse-Gold cosets x^7 ($x^{25} = (x^5)^{-1}$) and x^{11} ($x^{21} = (x^3)^{-1}$) in $(2, 8, 3)$ split the 160 permutations into $133 \sim_{\text{aff}} x^7$ (among them our 14-AND circuit of SM C.8) and $27 \sim_{\text{aff}} x^{11}$. For $n = 7$, the three Gold cosets x^3, x^5, x^9 in $(2, 16, 2)$ split the 15 permutations into 6, 6, and 3; the Kasami and Welch cosets x^{13} and x^{11} in $(2, 16, 3)$ split the 151 permutations into $76 \sim_{\text{aff}} x^{13}$ and $75 \sim_{\text{aff}} x^{11}$; and the five quartic AB cosets $x^{15}, x^{23}, x^{27}, x^{29}, x^{43}$ in $(2, 16, 4)$ split the 581 permutations into $436 \sim_{\text{aff}} x^{29}$, $144 \sim_{\text{aff}} x^{23}$, and one $\sim_{\text{aff}} x^{43}$ (the x^{85} inverse of the Gold x^3 circuit, whose 29/149/15, $D = 2$ netlist is the min-AND representative of the profile). In every other Category A profile all permutations lie in a single coset. Over the whole archive, 12 Category A permutations equal a catalogue table, 1,749 carry a certified affine-equivalence witness, and two (the 6-bit χ_3 -Feistel cubic involution class of SM C.17) match the forward invariants of x^{13} without being affine equivalent to it.

Table 5: The eight campaigns of the suite. Each width n has one campaign aiming at near-optimal profiles and one aiming at relaxed or lightweight profiles. All campaigns use a budget of 30,000 steps and a base patience $P = 3,500$; ρ_g is the probability of choosing a library seed when the episode does not start from an archive elite (Section 4.1). The target (δ, L) steers Stage 2 only: it decides when an episode switches from cryptographic to circuit-shrinking mutations and when the lexicographic order flips to the cost score (in **optimal** mode the annealing band is additionally relaxed to $(\delta + 2, 2L)$); admission to the archive is independent of the campaign and always governed by the grade gate of Table 1, so a candidate that only reaches the steering target but misses grade **B+** is never archived.

Campaign	n	Mode	Target (δ, L)	ρ_g
c01_5b_optimal	5	optimal	(2, 12)	0.7
c02_5b_relaxed	5	relaxed	(16, 24)	0.6
c03_6b_optimal	6	optimal	(4, 32)	0.7
c04_6b_relaxed	6	relaxed	(16, 48)	0.6
c05_7b_apn_optimal	7	optimal	(6, 40)	0.7
c06_7b_lightweight	7	lightweight	(8, 36)	0.6
c07_8b_aes_optimal	8	optimal	(4, 32)	0.7
c08_8b_lightweight	8	lightweight	(16, 80)	0.6

Table 6: The reference catalogue of Section 6.1. For each width n , the catalogue contains one power map x^d of $\mathbb{F}_{2^n} \cong \mathbb{F}_2[x]/(p_n(x))$ for every cyclotomic class of exponents with $\gcd(d, 2^n - 1) = 1$ (represented by its smallest exponent; this already includes the inverse of every power map), together with the named cipher S-boxes and their inverses. Tables 2 and 4 name the same cosets by their conventional exponents: inversion x^{2^n-2} (x^{62}, x^{126}) lies in the coset of x^{31} and x^{63} , and 5-bit inverse Gold x^{25} in the coset of x^7 .

n	$p_n(x)$	Catalogue entries
5	$x^5 + x^2 + 1$ (0x25)	x^1, x^3, x^5 (Gold APN), x^7, x^{11} (inverse Gold), x^{15} (inversion); KECCAK χ_5 and χ_5^{-1} [BDPVA11]; ASCON and ASCON ⁻¹ [DEMS21]
6	$x^6 + x + 1$ (0x43)	x^1, x^5 (Q_{2258}), x^{11}, x^{13} (Q_{2258}^{-1}), x^{23}, x^{31} (inversion); SPEEDY and SPEEDY ⁻¹ [LMMR21]; Dillon’s APN permutation and its inverse [BDMW10]
7	$x^7 + x + 1$ (0x83)	x^1, x^3, x^5 (Gold APN), x^7, x^9, x^{11} (Welch), x^{13} (Kasami), $x^{15}, x^{19}, x^{21}, x^{23}, x^{27}, x^{29}, x^{31}, x^{43}, x^{47}, x^{55}, x^{63}$ (inversion)
8	$x^8 + x^4 + x^3 + x + 1$ (0x11B)	$x^1, x^7, x^{11}, x^{13}, x^{19}, x^{23}, x^{29}, x^{31}, x^{37}, x^{43}, x^{47}, x^{53}, x^{59}, x^{61}, x^{91}, x^{127}$ (inversion); AES and AES ⁻¹ [Nat23]

Table 7: Archive summary across $n \in \{5, 6, 7, 8\}$. Netlists: distinct certified netlists (every entry passed exhaustive simulation and the Z3 miter of Section 7); Distinct: distinct permutations; Profiles: distinct $(n, \delta, L, \deg)$; Cat. A/B: distinct permutations per category (Section 6.1); **A+**/**A/B+**: netlists per grade of Table 1; Pareto: distinct (AND, XOR, D) triples non-dominated (componentwise) among the netlists of the same profile, summed over the profiles of Table 11 (which lists only min-AND and min- D representatives).

n	Netlists	Distinct	Profiles	Cat. A	Cat. B	A+	A	B+	Pareto
5	2,211	2,031	12	391	1,640	291	995	925	18
6	23,943	23,878	12	472	23,406	464	28	23,451	21
7	976	895	14	865	30	775	153	48	17
8	3,999	3,982	18	35	3,947	41	9	3,949	48
All	31,129	30,786	56	1,763	29,023	1,571	1,185	28,373	104

Table 8: Stage-2 AND erasure applied to $\text{YSH}_{8,2}$ (Section 6.5): the certified family of 8-bit degree-7 permutations it produces, after the Stage-4 pass, ordered by AND count; its 18-AND member is printed in full as $\text{YSH}_{8,3}$ (SM C.28). All members passed exhaustive simulation and the Z3 miter; β , FP, and SAC were recomputed from the lookup tables.

A/X/N	D	δ	L	$\mathcal{D}(S)$	β	FP	SAC
28/61/1	6	10	72	$[6^3, 7^{252}]$	22	0	.1094
27/62/1	6	12	72	$[6^3, 7^{252}]$	22	0	.1094
26/63/1	6	12	72	$[6^7, 7^{248}]$	22	0	.1094
25/63/1	6	16	72	$[6^7, 7^{248}]$	24	1	.1875
24/64/1	6	12	72	$[6^3, 7^{252}]$	32	2	.125
23/65/1	6	14	72	$[6^{15}, 7^{240}]$	36	1	.125
22/66/2	6	12	76	$[6^{15}, 7^{240}]$	36	3	.125
21/67/2	6	14	76	$[6^{31}, 7^{224}]$	36	2	.125
20/68/2	6	14	76	$[6^{31}, 7^{224}]$	36	3	.125
19/68/2	6	14	72	$[4^1, 6^{62}, 7^{192}]$	32	2	.125
18/71/2	5	14	72	$[4^1, 5^2, 6^{60}, 7^{192}]$	38	0	.1094

Table 9: Gate-exact Yosys synthesis and ABC NLDLM timing of the 30 inverse (S^{-1}) circuits of Section 6.3 on NanGate 45 nm (1 GE = $0.798 \mu\text{m}^2$, GE in SM C). $\mathcal{D}(S^{-1})$: degree spectrum; n/T : width/Tier (Section 6.3); FF/S: total/sync DFF_X1 ($G = 4R$ gadget flip-flops; cf. Table 3); N_{reconv} as in Table 3.

S-box (SM)	n/T	$D(S^{-1})$	Unmasked (S^{-1})				Masked ($d = 2$ DOM, S^{-1})					
			A/X/N	LTP	μm^2	Delay (ps)	D/R	A/X/N	FF/S	μm^2	Stage (ps)	N_{reconv}
$\text{YSH}_{5,1}^{-1}$: quartic $\delta=4$ (C.1)	5/T2	$[3^{15}, 4^{16}]$	13/38/6	12	77.67	623.2	2/13	52/128/6	84/32	642.66	323.9	5
$\text{YSH}_{5,2}^{-1}$: quad. $\delta=4$ (C.2)	5/T4	$[3^{31}]$	9/32/2	10	61.71	530.1	2/9	36/100/2	68/32	506.46	302.7	2
$\text{YSH}_{5,3}^{-1}$: cubic $\delta=8$ (C.3)	5/T4	$[3^{31}]$	9/22/2	8	45.75	366.8	2/9	36/80/2	62/26	447.41	329.7	0
$\text{YSH}_{5,4}^{-1}$: quartic $\delta=8$ (C.4)	5/T2	$[2^7, 3^8, 4^{16}]$	5/11/2	7	23.94	286.9	2/5	20/42/2	40/20	270.26	306.3	1
Gold x^5 (C.5)	5/T4	$[3^{31}]$	10/37/4	11	71.82	595.9	2/10	40/114/4	70/30	543.17	353.4	2
Gold class, SAC-ideal (C.6)	5/T1	$[3^{31}]$	10/41/10	13	81.40	652.6	2/10	40/122/10	86/46	631.48	346.9	5
χ_5 class (C.7)	5/T4	$[3^{31}]$	9/21/3	8	44.69	329.7	2/9	36/78/3	62/26	444.75	258.1	0
inverse-Gold (C.8)	5/T1	$[2^{31}]$	7/26/5	10	51.60	485.6	1/7	28/80/5	38/10	331.97	358.2	5
x^{15} (C.9)	5/T1	$[4^{31}]$	14/57/0	14	105.87	794.1	2/14	56/170/0	88/32	728.84	517.0	7
$\text{YSH}_{6,1}^{-1}$: quartic $\delta=4$ (C.10)	6/T2	$[3^7, 4^{56}]$	9/27/1	13	53.20	764.6	3/9	36/90/1	78/42	535.19	407.4	3
$\text{YSH}_{6,2}^{-1}$: quartic $k=5$ (C.11)	6/T2	$[2^{15}, 3^{48}]$	9/37/0	8	68.63	468.2	2/9	36/110/0	76/40	557.54	387.3	4
$\text{YSH}_{6,3}^{-1}$: cubic $\delta=4$ (C.12)	6/T2	$[3^{31}, 4^{32}]$	9/17/8	10	40.96	399.2	3/9	36/70/8	72/36	479.86	317.0	1
$\text{YSH}_{6,4}^{-1}$: deg. 5 $\delta=8$ (C.13)	6/T2	$[3^{15}, 4^{16}, 5^{32}]$	22/49/1	12	102.14	632.8	3/22	88/186/1	132/44	987.92	480.4	2
$\text{YSH}_{6,5}^{-1}$: deg. 5 $\delta=6$ (C.14)	6/T2	$[3^{15}, 4^{16}, 5^{32}]$	21/57/0	15	113.32	869.9	3/21	84/198/0	134/50	1011.33	381.2	7
$\text{YSH}_{6,6}^{-1}$: quartic $\delta=8$ (C.15)	6/T2	$[3^7, 4^{56}]$	27/50/3	10	110.12	561.3	2/27	108/208/3	140/32	1081.56	479.7	2
Q_{2258} class (C.16)	6/T3	$[3^{63}]$	18/37/0	9	78.20	473.2	2/18	72/146/0	96/24	743.74	480.1	2
cubic invol. (C.17)	6/T1	$[3^{63}]$	9/15/9	10	38.30	336.8	3/9	36/66/9	72/36	474.01	254.0	0
inversion (C.18)	6/T1	$[5^{63}]$	21/50/0	14	102.14	771.4	3/21	84/184/0	152/68	1070.38	317.0	3
Dillon APN (C.19)	6/T1	$[3^7, 4^{56}]$	28/78/7	15	158.00	819.8	2/28	112/268/7	168/56	1310.32	423.5	13
$\text{YSH}_{7,1}^{-1}$: quartic $\delta=4$ (C.20)	7/T2	$[4^7, 5^{120}]$	30/158/12	24	290.47	1511.7	3/30	120/436/12	226/106	1851.89	759.6	23
$\text{YSH}_{7,2}^{-1}$: deg. 6 $\delta=4$ (C.21)	7/T2	$[4^{63}, 6^{64}]$	34/164/9	21	302.71	1263.9	3/34	136/464/9	216/80	1866.79	698.8	28
Gold x^3 (C.22)	7/T3	$[4^{127}]$	29/149/15	22	276.64	1191.7	2/29	116/414/15	166/50	1542.80	772.0	28
Kasami x^{13} (C.23)	7/T3	$[3^{127}]$	37/139/0	17	261.21	1056.7	2/37	148/426/0	204/56	1759.86	735.9	6
inversion (C.24)	7/T1	$[6^{127}]$	42/229/6	33	413.36	2118.5	3/42	168/626/6	306/138	2564.77	716.5	39
power x^7 (C.25)	7/T3	$[5^{127}]$	51/220/15	33	413.36	1904.0	3/51	204/644/15	284/80	2537.11	976.9	27
$\text{YSH}_{8,1}^{-1}$: deg. 7 $\delta=6$ (C.26)	8/T2	$[6^1, 7^{254}]$	43/109/10	24	225.04	1380.5	4/43	172/390/10	264/92	2004.58	578.1	9
$\text{YSH}_{8,2}^{-1}$: deg. 7 $\delta=10$ (C.27)	8/T2	$[6^3, 7^{252}]$	28/84/2	28	164.92	1524.5	6/28	112/280/2	296/184	1905.62	431.9	7
$\text{YSH}_{8,3}^{-1}$: deg. 7 $\delta=14$ (C.28)	8/T2	$[5^1, 6^{62}, 7^{192}]$	18/77/8	28	146.30	1389.2	6/18	72/226/8	274/202	1680.59	438.1	5
AES ($D=3$) (C.29)	8/T1	$[7^{255}]$	42/106/4	22	215.99	1337.5	3/42	168/380/4	216/48	1764.11	576.4	11
AES (FIPS) (C.30)	8/T1	$[7^{255}]$	36/95/4	23	192.05	1281.9	4/36	144/334/4	276/132	1936.48	462.5	9

B.2 Inverse Derivations

This subsection collects the algebraic identities behind the Tier-2, Tier-3, and Tier-4 inverse circuits of Section 6.3.

Tier 2: shear extensions of a lower-degree core. For $\text{YSH}_{6,6}$, $\text{YSH}_{6,4}$, and $\text{YSH}_{6,5}$, $S(x) = T(Q(x))$ where $Q \sim_{\text{aff}} x^5$ (Q_{2258}) and $T(y) = y \oplus (s(y) \cdot c)$ is an involution shear along $c \in \mathbb{F}_2^6$ with $s(y) \in \mathbb{F}_2$. Rather than composing $Q^{-1}(T(y))$ sequentially, we use the

exact directional-derivative identity

$$S^{-1}(y) = Q^{-1}(y \oplus s(y)c) = Q^{-1}(y) \oplus (s(y) \wedge \Delta_c Q^{-1}(y)), \quad (9)$$

where $\Delta_c Q^{-1}(y) = Q^{-1}(y) \oplus Q^{-1}(y \oplus c)$ has degree $\leq \deg(Q^{-1}) - 1 = 2$ (depth ≤ 1), while $s(y)$ is quadratic (depth 1) for $\text{YSH}_{6,6}$ and cubic (depth 2) for $\text{YSH}_{6,4}$ and $\text{YSH}_{6,5}$, so $s(y) \wedge \Delta_c Q^{-1}(y)$ sits at depth 2 and 3, respectively. Evaluating $Q^{-1}(y)$ via the 18-AND, $D = 2$ tower-field norm circuit below in parallel with $s(y) \wedge \Delta_c Q^{-1}(y)$, followed by the Stage-4 pass, gives the theoretical minimum depth $D = 2$ (27/50/3) for $\text{YSH}_{6,6}^{-1}$ and $D = 3$ (22/49/1 and 21/57/0) for the two degree-5 inverses. For the output-sheared circuits $\text{YSH}_{6,2}$, $\text{YSH}_{7,1}$, and $\text{YSH}_{8,1}$ the shear $T(y) = y \oplus ((y_i \wedge y_j) \cdot c)$ with $c_i = c_j = 0$ is unwound with one AND gate; for the transposition extensions $\text{YSH}_{5,1}$ and $\text{YSH}_{7,2}$, $S^{-1}(y) = Q^{-1}(y) \oplus \mathbb{I}_{\{y \in \{y_\alpha, y_\beta\}\}} \cdot (\alpha \oplus \beta)$ with the two-point indicator of the swapped values evaluated in parallel.

Tier 2: backward unrolling. For $\text{YSH}_{5,4}$, whose two quartic coordinates are $y_2 = x_0 \oplus y_4 \oplus y_0 y_1$ and $y_3 = (x_0 \oplus x_4) \oplus y_0 \oplus y_0 y_4$, three coordinates of S^{-1} are quadratic in y : $x_0 = y_2 \oplus y_4 \oplus y_0 y_1$, $x_3 = y_1 \oplus y_2 \oplus y_0 y_1 \oplus y_3 y_4$, and $x_4 = x_0 \oplus y_3 \oplus y_0 \oplus y_0 y_4$ (one level, three AND gates); the remaining two, $x_1 = y_4 \oplus u \oplus x_0 u$ with $u = x_0 \oplus x_4$ and $x_2 = y_0 \oplus x_3 \oplus x_3 x_4$, re-use them at level 2, so $\text{YSH}_{5,4}^{-1}$ costs 5/11/2 at $D = 2$. The same propagation recovers the eight inputs of $\text{YSH}_{8,3}$ with every forward AND gate re-used once (18/77/8 at $D = 6$).

Tier 3: power-map factorizations. The four Tier-3 inverse circuits factor the inverse exponent of a Category A power map into low-depth algebraic stages:

- *6-bit Q_{2258} (x^5) inverse (x^{38}):* Writing $y = A + B\gamma \in \mathbb{F}_{2^3}[\gamma]/(\gamma^2 + \gamma + \nu)$, the relative norm $N = y^9 = A^2 + AB + B^2\nu \in \mathbb{F}_{2^3}$ costs one 6-AND $\mathbb{F}_{2^3}$ -product AB at level 1, and $y^{38} = y^2 \cdot N^4$ costs two 6-AND $\mathbb{F}_{2^3}$ -products $(A^2 + B^2)N^4$ and B^2N^4 at level 2, giving an 18/37/0, $D = 2$ circuit (SM C.16).
- *7-bit Gold (x^3) inverse (x^{85}):* Factoring $85 \equiv 5 \times 9 \times 16 \pmod{127}$ composes our 14-AND, $D = 1$ circuit for y^5 with the 15-AND, $D = 1$ circuit for z^9 and folds the Frobenius w^{16} by Paar CSE, giving a 29/149/15, $D = 2$ circuit at the theoretical minimum depth for degree 4 (SM C.22).
- *7-bit Kasami (x^{13}) inverse (x^{88} , $88 \equiv 11 \times 8 \pmod{127}$) and x^7 inverse (x^{109} , $109 \equiv 25 \times 64 + 5 \times 32 \pmod{127}$):* Using a 23-AND, $D = 1$ Karatsuba multiplier over $\mathbb{F}_2[z]/(z^7 + z + 1)$, we factor $y^{88} = (y^3 \cdot y^8)^8$ (14 + 23 = 37/139/0 at $D = 2$, SM C.23) and $y^{109} = ((y^5)^5)^{64} \cdot (y^5)^{32}$ via two 14-AND y^5 stages and one 23-AND Karatsuba product (51/220/15 at $D = 3$, five fewer ANDs than forward, SM C.25).

Tier 4: Hodge-dual coset-span synthesis. By Hodge duality $(\bigwedge^3(\mathbb{F}_2^5))^* \cong \bigwedge^2 \mathbb{F}_2^5$, where every bivector on a 5-space has rank ≤ 4 and thus a non-zero radical vector L_c , the cubic part of every degree-3 Boolean polynomial on $\mathbb{F}_2^5$ is divisible by a non-zero linear form $L_c(y)$, decomposing each component $\langle c, S^{-1}(y) \rangle$ as $L_c(y) \cdot Q_c(y) \oplus R_c(y)$ with $\deg Q_c, \deg R_c \leq 2$. Selecting a basis of five output components $c_1, \dots, c_5 \in \mathbb{F}_2^5$ whose quadratic quotients and residuals are spanned by $r \in \{4, 5\}$ shared level-1 AND gates (plus 5 level-2 AND gates multiplying each L_{c_i} by its quotient) yields the exact $D = 2$ inverses of $\text{YSH}_{5,2}$, $\text{YSH}_{5,3}$, Gold x^5 , and the χ_5 -class circuit listed in Section 6.3.

Table 10: Provenance of the 30 benchmark circuits (Section 6.5). **Seed**: the Stage-1 seed of the episode that produced the archived circuit—a published or algebraic library circuit (L), a structural template or power-map lookup-table node (T), or an archive elite (A); generator strings in parentheses are internal library/template keys whose nominal suffixes reflect pre-expansion or base templates (e.g., `q2256_inv_6b_11and_d2` expands the cubic Q_{2258}^{-1} LUT to 26/65, $D=2$; `dillon_apn_6b_12and_d3` and `gold_x3_7b_11and_d1` expand to 28/75, $D=2$ and 14/59, $D=1$; `ccz_lifted_8b_inversion_aes` appends a 1-AND shear to the 42-AND, $D=3$ inverter), whereas **Seed A/X, D** reports the actual AND/XOR count and AND depth of the expanded initial DAG at step 0 (– where the seed is a template or lookup-table node lowered during Stage 2); **Archived**: the certified netlist admitted to the archive after Stages 2–3; **Printed**: the netlist of SM C after the Stage-4 pass (Section 6.5).

S-box (SM)	Stage-1 seed (generator)	Seed A/X, D	Archived A/X/N, D	Printed A/X/N, D
YSH _{5,1} (C.1)	T power-map LUT node (<code>novel_power_map_5b</code>)	11/20, 1	11/20/3, 2	11/20/3, 2
YSH _{5,2} (C.2)	T depth-1 quadratic template, SAT-sampled (<code>mc5_md1</code>)	5/–, 1	5/14/3, 1	5/12/3, 1
YSH _{5,3} (C.3)	L KECCAK χ_5 circuit (<code>keccak_chi5</code>), +1 shear gate	5/10, 1	6/11/0, 2	6/11/0, 2
YSH _{5,4} (C.4)	A archive elite, $\delta=6$ niche (<code>swarm_transfection</code>)	16/39, 2	5/15/1, 3	5/12/1, 2
Gold x^5 (C.5)	T power-map LUT node (<code>novel_power_map_5b</code>)	7/22, 1	7/22/0, 1	7/22/0, 1
x^5 class, SAC = 0 (C.6)	T power-map LUT node (<code>novel_power_map_5b</code>)	7/21, 1	7/23/4, 1	7/21/4, 1
χ_5 class (C.7)	A archive elite (<code>swarm_transfection</code>)	5/11, 1	5/8/1, 1	5/8/1, 1
inverse-Gold (C.8)	T power-map LUT node (<code>novel_power_map_5b</code>)	15/32, 1	14/32/2, 2	14/31/2, 2
x^{15} (C.9)	T power-map LUT node (<code>novel_power_map_5b</code>)	14/45, 1	14/45/0, 2	14/45/0, 2
YSH _{6,1} (C.10)	L algebraic library circuit (<code>golden_6b_row26</code>)	9/27, 3	9/29/1, 3	9/27/1, 3
YSH _{6,2} (C.11)	L cubic Q_{2256}^{-1} circuit (<code>q2256_inv_6b_11and_d2</code>)	26/65, 2	27/64/0, 3	27/64/0, 3
YSH _{6,3} (C.12)	T 3-round Feistel template (<code>structural_feistel_6b</code>)	9/18, 3	10/19/9, 3	9/17/9, 3
YSH _{6,4} (C.13)	L Q_{2258} circuit (<code>q2258_6b_8and_d1</code>), +2 shear gates	9/30, 1	11/28/1, 3	11/22/1, 3
YSH _{6,5} (C.14)	L Q_{2258} circuit (<code>q2258_6b_8and_d1</code>), +3 shear gates	9/30, 1	12/43/0, 4	12/31/0, 4
YSH _{6,6} (C.15)	L Q_{2258} circuit (<code>q2258_6b_8and_d1</code>), +1 shear gate	9/30, 1	10/32/0, 2	10/22/0, 2
Q_{2258} class (C.16)	L Q_{2258} circuit (<code>q2258_6b_8and_d1</code>)	9/30, 1	8/50/0, 1	8/34/0, 1
cubic invol. (C.17)	T Misty template (<code>structural_misty_6b</code>)	–	9/18/9, 3	9/15/9, 3
inversion (C.18)	L $\mathbb{F}_{2^{3,2}}$ inversion (<code>algebraic_6b_sota_inversion_x31_d3</code>)	21/67, 3	21/58/0, 3	21/48/0, 3
Dillon APN (C.19)	L Dillon APN circuit (<code>dillon_apn_6b_12and_d3</code>)	28/75, 2	28/74/0, 2	28/68/0, 2
YSH _{7,1} (C.20)	L fractional subfield tower (<code>fractional_tower_7b_9and_d2</code>)	15/71, 2	15/71/10, 2	15/70/10, 2
YSH _{7,2} (C.21)	T unbalanced Feistel template (<code>unbalanced_feistel_7b</code>)	51/7, 1	51/77/1, 3	51/76/1, 3
Gold x^9 (C.22)	L Gold x^9 circuit (<code>gold_x3_7b_11and_d1</code>)	14/59, 1	14/59/0, 1	14/57/0, 1
Kasami x^{13} (C.23)	A archive elite from a Kasami power-map node (<code>apn_kasami_k2_x13</code>)	30/115, 2	30/114/0, 2	30/114/0, 2
inversion (C.24)	L sheared 7-bit inversion (<code>sheared_7b_inversion_standard</code>)	43/182, 4	42/182/6, 3	42/182/6, 3
power map x^7 (C.25)	T power-map LUT node (<code>power_map_x7_7b_d2</code>)	–	56/177/0, 2	56/98/0, 2
YSH _{8,1} (C.26)	T CCZ-lifted 8-bit inversion (<code>ccz_lifted_8b_inversion_aes</code>)	43/120, 4	43/107/7, 4	43/107/7, 4
YSH _{8,2} (C.27)	L Canright tower-field AES [Can05] (<code>canright_aes_8b_32and_d4</code>)	36/136, 4	28/61/1, 6	28/61/1, 6
YSH _{8,3} (C.28)	A archive elite YSH _{8,2} (<code>swarm_transfection</code>), + AND erasure	28/62, 6	18/73/2, 5	18/71/2, 5
AES, $D=3$ (C.29)	L depth-3 AES inverter [HB25] (<code>aes_8b_42and_d3</code>)	42/116, 3	42/103/4, 3	42/101/4, 3
AES FIPS-197 (C.30)	L Camellia circuit, same inversion class (<code>camellia_8b_32and_d4</code>)	–	36/119/4, 4	36/99/4, 4

Table 11: The 56 $(n, \delta, L, \deg)$ profiles present in the Stages 1–3 search archive of Table 7 (analytical Tier-4 Hodge-dual inverses synthesized outside the Stage-2 archive in Section 6.3, such as the 10/37/4, $D = 2$ inverse of 5-bit Gold x^5 in Table 4, are not counted here). Gr.: threshold tier of Table 1; Netlists: distinct certified netlists; Dist.: distinct permutations; Cat. A/B: distinct permutations in each category (1,761 of the 1,763 Category A permutations are proven equal or affine equivalent to their catalogue reference; only two—in the 6-bit (4, 16, 3) profile—match the forward invariants of x^{13} without being affine or EA equivalent to it). Min-AND rep.: the archived netlist of the profile with the fewest AND gates (ties broken by smaller depth, then in favour of a forward or inverse benchmark circuit of Section 6 with equal priority, then by fewer XOR gates); its counts A/X/N and depth D are those after the function-preserving Stage-4 pass of Section 4.3, re-verified by exhaustive simulation (for the ANF-synthesized representatives with more than 100 AND gates, after merging duplicate gates and pruning dead wires only). Class: as in Table 2, with power-map cosets labeled by their minimal cyclotomic representative (x^{31} , x^{63} vs. conventional x^{62} , x^{126} in Tables 2–4; see Table 6); the archived Tier-1–3 inverses of benchmark circuits are archive members in their own right and are named as such ($\text{YSH}_{n,i}^{-1}$). Min- D rep.: the lowest-depth netlist of the profile, when its depth is smaller than that of the min-AND representative. The 56 profiles contain 104 per-profile non-dominated (AND, XOR, D) points in total (Table 7).

n	$(\delta, L, \deg)$	Gr.	Netlists	Dist.	Cat.	A/B	Min-AND (A/X/N)	D	Class	Min- D (if shallower)
5	(2, 8, 2)	A+	126	81		81/0	7/21/4	1	A ($\sim_{\text{aff}} x^5$)	–
5	(2, 8, 3)	A+	165	160		160/0	14/31/2	2	A ($\sim_{\text{aff}} x^7$)	–
5	(2, 12, 4)	A	157	143		143/0	14/45/0	2	A ($= x^{15}$)	–
5	(4, 12, 4)	A	838	823		0/823	11/20/3	2	B ($\text{YSH}_{5,1}$)	–
5	(4, 16, 2)	B+	16	7		0/7	5/12/3	1	B ($\text{YSH}_{5,2}$)	–
5	(4, 16, 3)	B+	1	1		0/1	9/32/2	2	B ($\text{YSH}_{5,2}^{-1}$)	–
5	(4, 16, 4)	B+	567	546		0/546	12/26/2	2	B	–
5	(6, 12, 4)	B+	6	3		0/3	16/36/2	2	B	–
5	(6, 16, 4)	B+	20	11		0/11	7/18/2	4	B	15/33/3, $D = 2$, B
5	(8, 16, 2)	B+	12	8		6/2	5/8/1	1	A ($\sim_{\text{aff}} \chi_5$)	–
5	(8, 16, 3)	B+	16	10		1/9	5/12/1	3	B	6/11/0, $D = 2$, B ($\text{YSH}_{5,3}$)
5	(8, 16, 4)	B+	287	238		0/238	5/12/1	2	B ($\text{YSH}_{5,4}$)	–
6	(2, 16, 4)	A+	464	460		460/0	28/68/0	2	A ($= \text{Dillon}$)	–
6	(4, 16, 2)	A	6	4		4/0	8/34/0	1	A ($\sim_{\text{aff}} x^5$)	–
6	(4, 16, 3)	A	12	6		4/2	9/37/0	2	B ($\text{YSH}_{6,2}^{-1}$)	–
6	(4, 16, 4)	A	3	3		0/3	9/17/8	3	B ($\text{YSH}_{6,3}^{-1}$)	–
6	(4, 16, 5)	A	6	4		4/0	21/48/0	3	A ($\sim_{\text{aff}} x^{31}$)	–
6	(4, 20, 5)	A	1	1		0/1	29/71/0	3	B	–
6	(6, 20, 5)	B+	2	1		0/1	55/90/2	3	B	–
6	(6, 24, 5)	B+	19,380	19,369		0/19,369	12/31/0	4	B ($\text{YSH}_{6,5}$)	21/57/0, $D = 3$, B ($\text{YSH}_{6,5}^{-1}$)
6	(6, 28, 5)	B+	738	723		0/723	52/84/4	3	B	–
6	(8, 24, 4)	B+	2	2		0/2	10/22/0	2	B ($\text{YSH}_{6,6}$)	–
6	(8, 24, 5)	B+	1,728	1,716		0/1,716	16/47/1	6	B	49/80/2, $D = 3$, B
6	(8, 28, 5)	B+	1,601	1,589		0/1,589	11/22/1	3	B ($\text{YSH}_{6,4}$)	–
7	(2, 16, 2)	A+	19	15		15/0	14/57/0	1	A ($= x^3$)	–
7	(2, 16, 3)	A+	160	151		151/0	30/114/0	2	A ($= x^{13}$)	–
7	(2, 16, 4)	A+	596	581		581/0	29/149/15	2	A ($\sim_{\text{aff}} x^{43}$; inverse of Gold x^3)	–
7	(2, 20, 6)	A	124	116		116/0	42/182/6	3	A ($\sim_{\text{aff}} x^{63}$)	–
7	(4, 20, 6)	A	8	3		0/3	34/164/9	3	B ($\text{YSH}_{7,2}^{-1}$)	–
7	(4, 24, 6)	A	3	2		0/2	114/199/0	3	B	–
7	(4, 32, 4)	A	9	4		0/4	15/70/10	2	B ($\text{YSH}_{7,1}$)	–
7	(4, 32, 5)	A	9	5		0/5	30/158/12	3	B ($\text{YSH}_{7,1}^{-1}$)	–
7	(4, 36, 6)	B+	10	5		0/5	43/179/11	4	B	–
7	(4, 40, 6)	B+	4	2		0/2	43/179/5	4	B	–
7	(6, 20, 6)	B+	10	3		0/3	55/76/5	3	B	–
7	(6, 24, 6)	B+	21	6		0/6	56/79/3	3	B	–
7	(6, 40, 3)	B+	2	1		1/0	56/98/0	2	A ($= x^7$)	–
7	(6, 40, 5)	B+	1	1		1/0	51/220/15	3	A ($\sim_{\text{aff}} x^{55}$; inverse of power map x^7)	–
8	(4, 32, 7)	A+	41	35		35/0	36/99/4	4	A ($= \text{AES}$)	42/101/4, $D = 3$, A ($\sim_{\text{aff}} \text{AES}$)
8	(6, 36, 7)	A	5	3		0/3	245/411/2	3	B	–
8	(6, 40, 7)	A	4	2		0/2	246/418/4	3	B	–
8	(6, 56, 7)	B+	9	9		0/9	37/106/13	5	B	43/107/7, $D = 4$, B ($\text{YSH}_{8,1}$)
8	(8, 56, 7)	B+	2	1		0/1	246/417/5	3	B	–
8	(8, 60, 7)	B+	2	1		0/1	246/415/3	3	B	–
8	(10, 68, 7)	B+	1,167	1,167		0/1,167	239/851/3	3	B	–
8	(10, 72, 7)	B+	600	596		0/596	28/61/1	6	B ($\text{YSH}_{8,2}$)	239/876/4, $D = 3$, B
8	(12, 64, 7)	B+	519	518		0/518	240/845/2	3	B	–
8	(12, 68, 7)	B+	1,113	1,113		0/1,113	240/847/0	3	B	–
8	(12, 72, 7)	B+	526	526		0/526	24/64/1	6	B	240/848/2, $D = 3$, B
8	(12, 76, 7)	B+	1	1		0/1	22/66/2	6	B	–
8	(14, 68, 7)	B+	1	1		0/1	26/61/1	6	B	–
8	(14, 72, 7)	B+	3	3		0/3	18/71/2	5	B ($\text{YSH}_{8,3}$)	–
8	(14, 76, 7)	B+	2	2		0/2	20/68/2	6	B	–
8	(16, 72, 7)	B+	1	1		0/1	25/63/1	6	B	–
8	(16, 76, 7)	B+	1	1		0/1	27/60/1	6	B	–
8	(16, 80, 7)	B+	2	2		0/2	27/60/1	6	B	–

C Lookup Tables and Gate Netlists

This section of the SM lists every S-box that the body names together with its inverse S^{-1} : for both directions we give the lookup table in hexadecimal and the complete gate netlist, in the order in which the gates are emitted (`in_val` is the input word, bit 0 first; `out_val` is the output word, written most significant bit first). For every listing we recomputed δ , L , NL, $\deg(S)$, $\mathcal{D}(S)$, the number of fixed points (FP) and the SAC deviation from the printed lookup table, and we simulated the printed netlist on all 2^n inputs: each netlist computes its lookup table exactly. The gate counts and AND depths below are those of the printed netlists, which contain no gate that feeds no output. Unmasked and first-order pipelined DOM-indep ($d = 2$) silicon areas (μm^2 and Gate Equivalents, 1 GE = $0.798 \mu\text{m}^2$) and NLDM critical-path delays (ps) are obtained by gate-exact Yosys and ABC synthesis on the NanGate 45 nm Open Cell Library (Section 5.2). Category A entries are either equal to a catalogue S-box (=), certified affine equivalent to it ($\sim_{\text{aff}}$, with the explicit witness $S(x) = B \cdot S_{\text{ref}}(Ax \oplus a) \oplus b$ given by the column vectors $[L(1 \ll 0), \dots, L(1 \ll (n-1))]$ of $A, B \in \text{GL}(n, \mathbb{F}_2)$ and constants $a, b \in \mathbb{F}_2^n$), or invariant-consistent only ($\sim$); Category B entries ($\text{YSH}_{n,i}$, with inverses $\text{YSH}_{n,i}^{-1}$) are separated from every catalogue S-box by the invariants of Section 6.1 and lie in 15 pairwise disjoint EA classes. Two equivalent wrapper conventions appear in the listings: either the n -bit input word `in_val` is split into `a_in[i:i]` bits ($i = 0$ LSB to $n-1$ MSB) and the output word `out_val` is assembled by a Verilog concatenation $\{\text{y_n-1}, \dots, \text{y_0}\}$ (most significant bit first, rightmost wire LSB), or the bits are addressed directly by index as `x[i]` and `y[i]` ($i = 0$ LSB to $n-1$ MSB); `sync_buf_*` assignments are zero-cost wire aliases emitted by the stage scheduler and are not counted as gates.

C.1 $\text{YSH}_{5,1}$: 5-bit quartic, $\delta = 4$, $L = 12$ (5-bit, Category B)

- **Profile:** $\delta = 4$, $L = 12$ (NL = 10), $\deg(S) = 4$, $\mathcal{D}(S) = [2^{15}, 4^{16}]$, FP = 0, SAC deviation 0.125.
- **Cost:** 11 AND, 20 XOR, 3 NOT, AND depth $D = 2$ (unmasked NanGate 45 nm: $45.22 \mu\text{m}^2$ / 56.7 GE, delay 334.4 ps, logic depth 7; pipelined $d = 2$ DOM-indep: 66 DFF_X1 [44 gadget + 22 sync], $480.93 \mu\text{m}^2$ / 602.7 GE, stage delay 310.0 ps).
- **Remark:** Same $(\delta, L, \deg)$ as DB_1 [AL26] (9/36/2 in [BDM26]): two more AND gates and 16 fewer XOR gates at the same depth. Its quadratic components form a hyperplane, and its AND gates are 10 at level 1 and one at level 2 (Section 6.6).
- **Inverse** ($\text{YSH}_{5,1}^{-1}$): $\deg(S^{-1}) = 4$, $\mathcal{D}(S^{-1}) = [3^{15}, 4^{16}]$; cost: 13 AND, 38 XOR, 6 NOT, AND depth $D = 2$ (Tier 2: Parallel Transposition Inversion on 10-AND Gold inverse core; unmasked NanGate 45 nm: $77.67 \mu\text{m}^2$ / 97.3 GE, delay 623.2 ps, logic depth 12; pipelined $d = 2$ DOM-indep: 84 DFF_X1 [52 gadget + 32 sync], $642.66 \mu\text{m}^2$ / 805.3 GE, stage delay 323.9 ps).

Lookup table (LUT[0...31]):

```
0x15, 0x0f, 0x19, 0x14, 0x1d, 0x16, 0x0b, 0x17, 0x02, 0x1a, 0x1e, 0x11, 0x07, 0x0e, 0x01, 0x1f
0x00, 0x12, 0x09, 0x0c, 0x1b, 0x18, 0x08, 0x10, 0x13, 0x03, 0x0a, 0x0d, 0x05, 0x04, 0x06, 0x1c
```

Netlist (11 AND, 20 XOR, 3 NOT, $D = 2$):

```
a_in = in_val[4:0]; split_2 = a_in[0:0]; split_3 = a_in[1:1];
split_4 = a_in[2:2]; split_5 = a_in[3:3]; split_6 = a_in[4:4];
and_7 = split_2 & split_3; and_8 = split_2 & split_4; and_9 = split_3 & split_4;
and_10 = split_2 & split_5; and_11 = split_3 & split_5;
and_12 = split_4 & split_5; and_13 = split_2 & split_6;
and_14 = split_3 & split_6; and_15 = split_4 & split_6;
and_16 = split_5 & split_6; xor_18 = split_5 ^ and_7; xor_19 = split_2 ^ and_9;
xor_21 = split_3 ^ and_12; xor_34 = split_4 ^ and_13; and_17 = and_7 & and_15;
```

```

xor_22 = and_8 ^ and_15; xor_28 = and_10 ^ and_15; xor_20 = split_6 ^ xor_18;
xor_29 = xor_18 ^ xor_19; xor_37 = and_11 ^ xor_19; xor_35 = xor_19 ^ xor_34;
xor_24 = and_17 ^ xor_21; xor_25 = and_12 ^ xor_22; xor_23 = and_14 ^ xor_20;
xor_38 = xor_22 ^ xor_20; xor_30 = xor_28 ^ xor_29; xor_36 = xor_35 ^ xor_24;
xor_26 = xor_25 ^ xor_23; xor_31 = and_16 ^ xor_23; xor_39 = xor_37 ^ xor_38;
not_27 = ~xor_26; xor_32 = xor_31 ^ xor_24; not_40 = ~xor_39; not_33 = ~xor_32;
concat_41 = { not_40, xor_36, not_33, xor_30, not_27 }; out_val = concat_41;

```

Inverse lookup table ($\text{LUT}_{S^{-1}}[0 \dots 31]$):

```

0x10, 0x0e, 0x08, 0x19, 0x1c, 0x1e, 0x0c, 0x16, 0x12, 0x1a, 0x06, 0x13, 0x1b, 0x0d, 0x01
0x17, 0x0b, 0x11, 0x18, 0x03, 0x00, 0x05, 0x07, 0x15, 0x02, 0x09, 0x14, 0x1f, 0x04, 0x0a, 0x0f

```

Inverse netlist (13 AND, 38 XOR, 6 NOT, $D = 2$):

```

xor_0 = x[1] ^ x[4]; xor_1 = x[2] ^ x[3]; xor_2 = x[0] ^ x[2];
xor_3 = x[0] ^ xor_1; xor_4 = x[4] ^ xor_3; xor_5 = xor_0 ^ xor_3;
xor_6 = x[4] ^ xor_1; xor_7 = x[2] ^ xor_0; xor_8 = x[4] ^ xor_2;
xor_9 = x[1] ^ xor_1; xor_10 = x[1] ^ xor_2; xor_11 = x[1] ^ x[2];
xor_12 = x[2] ^ x[4]; not_0 = ~x[0]; not_1 = ~x[1]; not_2 = ~xor_1;
and_0 = xor_2 & xor_4; and_1 = x[3] & xor_5; and_2 = xor_2 & xor_6;
and_3 = xor_1 & xor_7; and_4 = xor_1 & xor_8; and_5 = not_0 & not_1;
and_6 = not_2 & x[4]; and_7 = and_5 & and_6; xor_13 = and_2 ^ xor_0;
xor_14 = and_1 ^ and_4; xor_15 = and_3 ^ xor_13; xor_16 = and_0 ^ and_1;
xor_17 = xor_13 ^ xor_16; xor_18 = x[2] ^ and_1; xor_19 = xor_15 ^ xor_18;
xor_20 = and_1 ^ and_3; xor_21 = x[4] ^ and_0; xor_22 = and_4 ^ xor_21;
xor_23 = and_0 ^ xor_15; xor_24 = xor_15 ^ and_7; xor_25 = x[3] ^ and_0;
xor_26 = and_2 ^ xor_25; not_3 = ~xor_19; not_4 = ~xor_20;
and_8 = xor_17 & xor_9; and_9 = not_3 & xor_3; and_10 = not_4 & xor_10;
and_11 = xor_22 & xor_11; and_12 = xor_12 & xor_14; xor_27 = and_4 ^ and_9;
xor_28 = and_10 ^ and_11; xor_29 = xor_14 ^ xor_28; xor_30 = xor_0 ^ and_10;
xor_31 = xor_27 ^ xor_30; xor_32 = xor_13 ^ xor_29; xor_33 = xor_23 ^ and_8;
xor_34 = xor_27 ^ xor_24; xor_35 = xor_26 ^ and_9; xor_36 = and_12 ^ xor_29;
xor_37 = xor_35 ^ xor_36; not_5 = ~xor_37; y[0] = xor_31; y[1] = xor_32;
y[2] = xor_33; y[3] = xor_34; y[4] = not_5;

```

C.2 YSH_{5,2}: 5-bit quadratic, $\delta = 4$, $L = 16$ (5-bit, Category B)

- **Profile:** $\delta = 4$, $L = 16$ (NL = 8), $\deg(S) = 2$, $\mathcal{D}(S) = [2^{31}]$, FP = 0, SAC deviation 0.5.
- **Cost:** 5 AND, 12 XOR, 3 NOT, AND depth $D = 1$ (unmasked NanGate 45 nm: $26.07 \mu\text{m}^2$ / 32.7 GE, delay 198.3 ps, logic depth 5; pipelined $d = 2$ DOM-indep: 30 DFF_X1 [20 gadget + 10 sync], $228.76 \mu\text{m}^2$ / 286.7 GE, stage delay 202.1 ps).
- **Remark:** Differential uniformity $\delta = 4$ instead of 8 at the AND count and depth of χ_5 (5 AND, $D = 1$). Structure and origin: an instance of the depth-1 quadratic template $y_k = M_k x \oplus (\alpha_k \cdot x \oplus a_k)(\beta_k \cdot x \oplus b_k)$ (one product per output), drawn by the template's SAT sampler with this profile directly (Section 6.5); the linear-layer passes reduced the sampled netlist from 5/20/3 to 5/12/3.
- **Inverse** ($\text{YSH}_{5,2}^{-1}$): $\deg(S^{-1}) = 3$, $\mathcal{D}(S^{-1}) = [3^{31}]$; cost: 9 AND, 32 XOR, 2 NOT, AND depth $D = 2$ (Tier 4: Hodge-dual linear-divisor coset-span $D = 2$ synthesis; unmasked NanGate 45 nm: $61.71 \mu\text{m}^2$ / 77.3 GE, delay 530.1 ps, logic depth 10; pipelined $d = 2$ DOM-indep: 68 DFF_X1 [36 gadget + 32 sync], $506.46 \mu\text{m}^2$ / 634.7 GE, stage delay 302.7 ps).

Lookup table ($\text{LUT}[0 \dots 31]$):

```

0x02, 0x04, 0x0e, 0x19, 0x0a, 0x0f, 0x17, 0x03, 0x11, 0x06, 0x1c, 0x1a, 0x01, 0x15, 0x1d, 0x18
0x08, 0x0c, 0x10, 0x05, 0x00, 0x07, 0x09, 0x1f, 0x0b, 0x1e, 0x12, 0x16, 0x1b, 0x0d, 0x13, 0x14

```

Netlist (5 AND, 12 XOR, 3 NOT, $D = 1$):

```

xor_0 = x[2] ^ x[4]; xor_1 = x[2] ^ x[3]; xor_2 = x[0] ^ xor_0;
xor_3 = x[1] ^ x[3]; xor_4 = x[0] ^ x[1]; xor_5 = x[1] ^ xor_1;
xor_6 = x[1] ^ x[4]; not_0 = ~xor_6; not_1 = ~xor_0; not_2 = ~xor_2;
and_0 = x[2] & xor_1; and_1 = xor_3 & xor_2; and_2 = xor_4 & xor_5;
and_3 = not_0 & x[4]; and_4 = not_1 & not_2; xor_7 = and_2 ^ xor_3;
xor_8 = and_4 ^ xor_1; xor_9 = and_3 ^ xor_4; xor_10 = and_0 ^ xor_6;
xor_11 = x[3] ^ and_1; y[0] = xor_7; y[1] = xor_8; y[2] = xor_9; y[3] = xor_10;
y[4] = xor_11;

```

Inverse lookup table ($\text{LUT}_{S^{-1}}[0 \dots 31]$):

```
0x14, 0x0c, 0x00, 0x07, 0x01, 0x13, 0x09, 0x15, 0x10, 0x16, 0x04, 0x18, 0x11, 0x1d, 0x02, 0x05
0x12, 0x08, 0x1a, 0x1e, 0x1f, 0x0d, 0x1b, 0x06, 0x0f, 0x03, 0x0b, 0x1c, 0x0a, 0x0e, 0x19, 0x17
```

Inverse netlist (9 AND, 32 XOR, 2 NOT, $D = 2$):

```
xor_0 = x[0] ^ x[4]; xor_1 = x[1] ^ x[3]; xor_2 = x[0] ^ x[2];
xor_3 = x[2] ^ x[3]; xor_4 = x[2] ^ xor_0; xor_5 = x[1] ^ x[2];
xor_6 = xor_1 ^ xor_2; xor_7 = x[1] ^ x[4]; xor_8 = x[1] ^ xor_0;
not_0 = ~x[2]; and_0 = xor_3 & xor_4; and_1 = xor_5 & x[4];
and_2 = x[3] & x[4]; and_3 = xor_6 & xor_7; xor_9 = x[2] ^ and_0;
xor_10 = and_1 ^ xor_1; xor_11 = x[2] ^ and_2; xor_12 = x[3] ^ and_3;
xor_13 = and_2 ^ xor_10; xor_14 = x[4] ^ xor_9; xor_15 = not_0 ^ xor_12;
xor_16 = xor_0 ^ xor_13; xor_17 = xor_0 ^ and_1; xor_18 = and_3 ^ xor_17;
xor_19 = xor_11 ^ xor_12; not_1 = ~xor_18; and_4 = xor_14 & x[1];
and_5 = xor_15 & x[1]; and_6 = xor_2 & xor_16; and_7 = not_1 & x[3];
and_8 = xor_19 & x[0]; xor_20 = and_6 ^ and_8; xor_21 = and_4 ^ and_5;
xor_22 = and_4 ^ xor_20; xor_23 = and_7 ^ not_0; xor_24 = xor_11 ^ xor_21;
xor_25 = x[4] ^ and_8; xor_26 = xor_21 ^ xor_25; xor_27 = xor_8 ^ xor_22;
xor_28 = xor_23 ^ xor_27; xor_29 = xor_9 ^ xor_22; xor_30 = xor_13 ^ xor_20;
xor_31 = xor_23 ^ xor_30; y[0] = xor_24; y[1] = xor_26; y[2] = xor_28;
y[3] = xor_29; y[4] = xor_31;
```

C.3 YSH_{5,3}: 5-bit cubic, $\delta = 8$, $L = 16$ (5-bit, Category B)

- **Profile:** $\delta = 8$, $L = 16$ (NL = 8), $\deg(S) = 3$, $\mathcal{D}(S) = [2^{15}, 3^{16}]$, FP = 2, SAC deviation 0.5.
- **Cost:** 6 AND, 11 XOR, 0 NOT, AND depth $D = 2$ (unmasked NanGate 45 nm: $23.94 \mu\text{m}^2$ / 30.0 GE, delay 231.6 ps, logic depth 5; pipelined $d = 2$ DOM-indep: 48 DFF_X1 [24 gadget + 24 sync], $316.01 \mu\text{m}^2$ / 396.0 GE, stage delay 187.6 ps).
- **Remark:** Same $(\delta, L, \deg)$ as χ_5^{-1} (9/27/2 in [BDM26]): three fewer AND gates and 16 fewer XOR gates at the same depth. Structure and origin: the five level-1 products $x_1x_2, x_2x_3, x_3x_4, x_4x_0, x_0x_1$ are those of the KECCAK χ_5 seed circuit (without the level-2 gate the netlist computes that seed's lookup table exactly), and the single level-2 AND gate is a Stage-2 output shear that raises the degree to 3 (Section 6.5).
- **Inverse** (YSH_{5,3}⁻¹): $\deg(S^{-1}) = 3$, $\mathcal{D}(S^{-1}) = [3^{31}]$; cost: 9 AND, 22 XOR, 2 NOT, AND depth $D = 2$ (Tier 4: Hodge-dual linear-divisor coset-span $D = 2$ synthesis; unmasked NanGate 45 nm: $45.75 \mu\text{m}^2$ / 57.3 GE, delay 366.8 ps, logic depth 8; pipelined $d = 2$ DOM-indep: 62 DFF_X1 [36 gadget + 26 sync], $447.41 \mu\text{m}^2$ / 560.7 GE, stage delay 329.7 ps).

Lookup table ($\text{LUT}[0 \dots 31]$):

```
0x00, 0x12, 0x09, 0x18, 0x14, 0x06, 0x0d, 0x1c, 0x0a, 0x1a, 0x03, 0x10, 0x16, 0x04, 0x0f, 0x1e
0x05, 0x15, 0x0c, 0x1f, 0x11, 0x01, 0x08, 0x1b, 0x0b, 0x19, 0x02, 0x13, 0x17, 0x07, 0x0e, 0x1d
```

Netlist (6 AND, 11 XOR, 0 NOT, $D = 2$):

```
and_6 = x[1] & x[2]; xor_11 = x[0] ^ x[2]; and_7 = x[2] & x[3];
xor_13 = x[1] ^ x[3]; and_8 = x[3] & x[4]; xor_15 = x[2] ^ x[4];
xor_19 = x[4] ^ x[1]; xor_12 = xor_11 ^ and_6; xor_14 = xor_13 ^ and_7;
xor_16 = xor_15 ^ and_8; and_9 = x[4] & x[0]; and_29 = x[0] & xor_14;
xor_17 = x[3] ^ x[0]; xor_26 = and_29 ^ xor_17; xor_18 = xor_26 ^ and_9;
and_10 = x[0] & x[1]; xor_20 = xor_19 ^ and_10; y[0] = xor_20; y[1] = xor_18;
y[2] = xor_16; y[3] = xor_14; y[4] = xor_12;
```

Inverse lookup table ($\text{LUT}_{S^{-1}}[0 \dots 31]$):

```
0x00, 0x15, 0x1a, 0x0a, 0x0d, 0x10, 0x05, 0x1d, 0x16, 0x02, 0x08, 0x18, 0x12, 0x06, 0x1e, 0x0e
0x0b, 0x14, 0x01, 0x1b, 0x04, 0x11, 0x0c, 0x1c, 0x03, 0x19, 0x09, 0x17, 0x07, 0x1f, 0x0f, 0x13
```

Inverse netlist (9 AND, 22 XOR, 2 NOT, $D = 2$):

```

xor_0 = x[0] ^ x[3]; xor_1 = x[1] ^ x[4]; xor_2 = x[0] ^ x[2];
xor_3 = x[1] ^ x[3]; xor_4 = x[2] ^ x[4]; xor_5 = x[1] ^ xor_0;
and_0 = x[0] & x[1]; and_1 = x[0] & x[2]; and_2 = xor_0 & x[4];
and_3 = xor_3 & xor_4; xor_6 = x[2] ^ and_1; xor_7 = and_2 ^ xor_1;
xor_8 = and_0 ^ xor_2; xor_9 = and_3 ^ xor_6; xor_10 = xor_7 ^ xor_9;
xor_11 = x[4] ^ xor_6; xor_12 = xor_0 ^ xor_9; xor_13 = x[4] ^ xor_8;
xor_14 = xor_1 ^ xor_6; not_0 = ~xor_10; not_1 = ~xor_12; and_4 = xor_8 & x[3];
and_5 = xor_7 & x[2]; and_6 = xor_5 & not_0; and_7 = xor_0 & xor_11;
and_8 = not_1 & x[4]; xor_15 = and_5 ^ xor_7; xor_16 = xor_13 ^ and_4;
xor_17 = x[3] ^ xor_15; xor_18 = x[2] ^ and_6; xor_19 = xor_14 ^ and_7;
xor_20 = xor_0 ^ and_8; xor_21 = xor_15 ^ xor_20; y[0] = xor_16; y[1] = xor_17;
y[2] = xor_18; y[3] = xor_19; y[4] = xor_21;

```

C.4 YSH_{5,4}: 5-bit quartic, $\delta = 8$, $L = 16$, $D = 2$ (5-bit, Category B)

- **Profile:** $\delta = 8$, $L = 16$ (NL = 8), $\deg(S) = 4$, $\mathcal{D}(S) = [2^7, 3^8, 4^{16}]$, FP = 2, SAC deviation 0.5.
- **Cost:** 5 AND, 12 XOR, 1 NOT, AND depth $D = 2$ (unmasked NanGate 45 nm: $25.00 \mu\text{m}^2 / 31.3 \text{ GE}$, delay 295.4 ps, logic depth 7; pipelined $d = 2$ DOM-indep: 40 DFF_X1 [20 gadget + 20 sync], $272.92 \mu\text{m}^2 / 342.0 \text{ GE}$, stage delay 246.2 ps).
- **Remark:** Algebraic degree 4 at the 5-AND count of χ_5 and at depth $D = 2$, the minimum for degree 4: the three quadratic coordinates $y_0 = x_2 \oplus x_3 \oplus x_3x_4$, $y_4 = x_1 \oplus u \oplus x_0u$ ($u = x_0 \oplus x_4$), and $y_1 = \overline{x_3 \oplus x_1 \oplus ux_1} \oplus x_0$ are computed at level 1, and the two quartic coordinates reuse them in the two level-2 products $y_2 = x_0 \oplus y_4 \oplus y_0y_1$ and $y_3 = u \oplus y_0 \oplus y_0y_4$, so S and S^{-1} both cost 5 AND at $D = 2$. The 16 quartic components are the complement of the hyperplane $V_{\leq 3}$, inside which the 7 quadratic components form a 3-dimensional subspace. As a two-round unbalanced Feistel-type network, it has maximal boomerang uniformity $\beta = 32$ (e.g., BCT(2, 2) = 32), like the 3-round Feistel circuits of Section 6.6. Stage-1 seed: an archive elite with 16 AND gates at $D = 2$ and $\delta = 6$, of which Stage 2 kept five AND gates (Section 6.5); Stage-4 post-processing reduced the archived 5/15/1, $D = 3$ netlist to 5/12/1, $D = 2$.
- **Inverse** (YSH_{5,4}⁻¹): $\deg(S^{-1}) = 4$, $\mathcal{D}(S^{-1}) = [2^7, 3^8, 4^{16}]$; cost: 5 AND, 11 XOR, 2 NOT, AND depth $D = 2$ (Tier 2: Structural backward unrolling (three quadratic coordinates at depth 1, two quartic coordinates at depth 2); unmasked NanGate 45 nm: $23.94 \mu\text{m}^2 / 30.0 \text{ GE}$, delay 286.9 ps, logic depth 7; pipelined $d = 2$ DOM-indep: 40 DFF_X1 [20 gadget + 20 sync], $270.26 \mu\text{m}^2 / 338.7 \text{ GE}$, stage delay 306.3 ps).

Lookup table (LUT[0...31]):

```

0x02, 0x0c, 0x14, 0x18, 0x0f, 0x05, 0x15, 0x19, 0x09, 0x03, 0x13, 0x1f, 0x00, 0x0e, 0x16, 0x1a
0x1e, 0x04, 0x0a, 0x12, 0x1b, 0x0d, 0x07, 0x17, 0x1c, 0x06, 0x08, 0x10, 0x1d, 0x0b, 0x01, 0x11

```

Netlist (5 AND, 12 XOR, 1 NOT, $D = 2$):

```

xor_0 = x[2] ^ x[3]; xor_1 = x[3] ^ x[1]; and_0 = x[3] & x[4];
xor_2 = x[0] ^ x[4]; xor_3 = xor_0 ^ and_0; and_1 = xor_2 & x[1];
xor_4 = x[1] ^ xor_2; xor_5 = xor_2 ^ xor_3; xor_6 = xor_1 ^ and_1;
not_0 = ~xor_6; and_2 = x[0] & xor_2; xor_7 = not_0 ^ x[0];
xor_8 = xor_4 ^ and_2; and_3 = xor_3 & xor_7; xor_9 = x[0] ^ xor_8;
and_4 = xor_8 & xor_3; xor_10 = xor_9 ^ and_3; xor_11 = xor_5 ^ and_4;
y[0] = xor_3; y[1] = xor_7; y[2] = xor_10; y[3] = xor_11; y[4] = xor_8;

```

Inverse lookup table (LUT _{S^{-1}} [0...31]):

```

0x0c, 0x1e, 0x00, 0x09, 0x11, 0x05, 0x19, 0x16, 0x1a, 0x08, 0x12, 0x1d, 0x01, 0x15, 0x0d, 0x04
0x1b, 0x1f, 0x13, 0x0a, 0x02, 0x06, 0x0e, 0x17, 0x03, 0x07, 0x0f, 0x14, 0x18, 0x1c, 0x10, 0x0b

```

Inverse netlist (5 AND, 11 XOR, 2 NOT, $D = 2$):

```

xor_0 = x[0] ^ x[3]; and_0 = x[0] & x[1]; and_1 = x[3] & x[4];
and_2 = x[0] & x[4]; xor_1 = x[2] ^ and_0; xor_2 = x[4] ^ xor_1;
xor_3 = and_2 ^ xor_0; xor_4 = x[1] ^ and_1; xor_5 = xor_1 ^ xor_4;
xor_6 = xor_2 ^ xor_3; xor_7 = x[4] ^ xor_3; xor_8 = x[0] ^ xor_5;
not_0 = ~xor_5; and_3 = xor_2 & xor_3; and_4 = not_0 & xor_6;
xor_9 = and_3 ^ xor_7; xor_10 = and_4 ^ xor_8; not_1 = ~xor_10; y[0] = xor_2;
y[1] = xor_9; y[2] = not_1; y[3] = not_0; y[4] = xor_6;

```

C.5 5-bit Gold APN x^5 (5-bit, Category A)

- **Profile:** $\delta = 2$, $L = 8$ (NL = 12), $\deg(S) = 2$, $\mathcal{D}(S) = [2^{31}]$, FP = 2, SAC deviation 0.5.
- **Cost:** 7 AND, 22 XOR, 0 NOT, AND depth $D = 1$ (unmasked NanGate 45 nm: $42.56 \mu\text{m}^2 / 53.3$ GE, delay 311.2 ps, logic depth 6; pipelined $d = 2$ DOM-indep: 38 DFF_X1 [28 gadget + 10 sync], $316.54 \mu\text{m}^2 / 396.7$ GE, stage delay 288.5 ps).
- **Remark:** The exact lookup table of x^5 over $\mathbb{F}_2[x]/(x^5 + x^2 + 1)$ (the invariants do not separate x^3 and x^5); 7/24/1 in [BDM26] and 7/19/1 in [BDM25].
- **Inverse** (S^{-1}): $\deg(S^{-1}) = 3$, $\mathcal{D}(S^{-1}) = [3^{31}]$; cost: 10 AND, 37 XOR, 4 NOT, AND depth $D = 2$ (Tier 4: Hodge-dual linear-divisor coset-span $D = 2$ synthesis; unmasked NanGate 45 nm: $71.82 \mu\text{m}^2 / 90.0$ GE, delay 595.9 ps, logic depth 11; pipelined $d = 2$ DOM-indep: 70 DFF_X1 [40 gadget + 30 sync], $543.17 \mu\text{m}^2 / 680.7$ GE, stage delay 353.4 ps).

Lookup table (LUT[0...31]):

```
0x00, 0x01, 0x05, 0x16, 0x11, 0x19, 0x04, 0x1e, 0x1f, 0x18, 0x12, 0x07, 0x14, 0x1a, 0x09, 0x15
0x0c, 0x06, 0x17, 0x0f, 0x10, 0x13, 0x1b, 0x0a, 0x0e, 0x02, 0x1d, 0x03, 0x08, 0x0d, 0x0b, 0x1c
```

Netlist (7 AND, 22 XOR, 0 NOT, $D = 1$):

```
a_in = in_val[4:0]; split_2 = a_in[0:0]; split_3 = a_in[1:1];
split_4 = a_in[2:2]; split_5 = a_in[3:3]; split_6 = a_in[4:4];
xor_7 = split_2 ^ split_4; xor_9 = split_3 ^ split_4; and_14 = split_3 & split_5;
and_16 = split_4 & split_5; xor_26 = split_2 ^ split_5;
xor_8 = split_3 ^ split_6; xor_13 = split_4 ^ split_6;
and_15 = split_3 & split_6; xor_12 = split_5 ^ xor_7; and_17 = split_2 & xor_9;
xor_24 = split_3 ^ and_16; xor_10 = split_5 ^ xor_8; xor_11 = split_2 ^ xor_8;
xor_21 = split_5 ^ and_15; and_20 = xor_12 & xor_13; and_18 = split_2 & xor_10;
and_19 = xor_7 & xor_11; xor_22 = and_20 ^ xor_21; xor_27 = xor_26 ^ and_20;
xor_29 = and_16 ^ and_18; xor_31 = and_17 ^ and_18; xor_35 = split_2 ^ and_19;
xor_25 = xor_22 ^ xor_13; xor_28 = xor_24 ^ xor_27; xor_30 = xor_21 ^ xor_29;
xor_32 = xor_24 ^ xor_31; xor_36 = xor_35 ^ xor_22; xor_34 = and_14 ^ xor_25;
xor_33 = xor_32 ^ xor_25; concat_37 = { xor_36, xor_34, xor_33, xor_30, xor_28 };
out_val = concat_37;
```

Inverse lookup table (LUT $_{S^{-1}}$ [0...31]):

```
0x00, 0x01, 0x19, 0x1b, 0x06, 0x02, 0x11, 0x0b, 0x1c, 0x0e, 0x17, 0x1e, 0x10, 0x1d, 0x18, 0x13
0x14, 0x04, 0x0a, 0x15, 0x0c, 0x0f, 0x03, 0x12, 0x09, 0x05, 0x0d, 0x16, 0x1f, 0x1a, 0x07, 0x08
```

Inverse netlist (10 AND, 37 XOR, 4 NOT, $D = 2$):

```
xor_0 = x[1] ^ x[4]; xor_1 = x[2] ^ xor_0; xor_2 = x[0] ^ x[3];
xor_3 = x[0] ^ x[1]; xor_4 = x[2] ^ xor_2; xor_5 = x[0] ^ x[4];
xor_6 = x[0] ^ x[2]; xor_7 = xor_3 ^ xor_4; and_0 = xor_2 & xor_1;
and_1 = xor_4 & xor_5; and_2 = xor_3 & xor_6; and_3 = xor_0 & x[3];
and_4 = xor_7 & x[4]; xor_8 = and_2 ^ and_3; xor_9 = and_4 ^ xor_1;
xor_10 = and_0 ^ and_2; xor_11 = x[0] ^ and_0; xor_12 = x[3] ^ xor_8;
xor_13 = x[4] ^ xor_12; xor_14 = x[4] ^ and_0; xor_15 = and_1 ^ xor_14;
xor_16 = xor_3 ^ and_1; xor_17 = xor_8 ^ xor_16; xor_18 = xor_9 ^ xor_10;
xor_19 = and_3 ^ xor_11; xor_20 = xor_3 ^ and_3; xor_21 = xor_8 ^ xor_11;
xor_22 = x[2] ^ xor_12; not_0 = ~xor_13; not_1 = ~xor_17; not_2 = ~xor_18;
not_3 = ~xor_19; and_5 = not_0 & x[0]; and_6 = xor_15 & x[1];
and_7 = not_1 & x[2]; and_8 = not_2 & x[3]; and_9 = not_3 & x[4];
xor_23 = and_1 ^ and_6; xor_24 = and_7 ^ xor_23; xor_25 = xor_9 ^ xor_24;
xor_26 = and_5 ^ and_6; xor_27 = xor_20 ^ xor_26; xor_28 = xor_21 ^ and_7;
xor_29 = xor_26 ^ xor_28; xor_30 = xor_22 ^ and_5; xor_31 = and_9 ^ xor_23;
xor_32 = xor_30 ^ xor_31; xor_33 = xor_10 ^ and_8; xor_34 = and_9 ^ xor_25;
xor_35 = xor_33 ^ xor_34; xor_36 = xor_2 ^ xor_25; y[0] = xor_27;
y[1] = xor_29; y[2] = xor_32; y[3] = xor_35; y[4] = xor_36;
```

C.6 5-bit Gold class ($\sim_{\text{aff}} x^5$) with ideal avalanche, 7 AND (5-bit, Category A)

- **Profile:** $\delta = 2$, $L = 8$ (NL = 12), $\deg(S) = 2$, $\mathcal{D}(S) = [2^{31}]$, FP = 1, SAC deviation 0.
- **Cost:** 7 AND, 21 XOR, 4 NOT, AND depth $D = 1$ (unmasked NanGate 45 nm: $43.09 \mu\text{m}^2$ / 54.0 GE, delay 282.0 ps, logic depth 6; pipelined $d = 2$ DOM-indep: 38 DFF_X1 [28 gadget + 10 sync], $315.48 \mu\text{m}^2$ / 395.3 GE, stage delay 243.3 ps).
- **Remark:** Affine-equivalent to x^5 (certified by the solver of Section 6.1: $S(x) = B S_{\text{ref}}(Ax \oplus a) \oplus b$ with $S_{\text{ref}}(u) = u^5 \bmod (x^5 + x^2 + 1)$, $A = [0x03, 0x08, 0x05, 0x0a, 0x12]$, $a = 0x01$, $B = [0x08, 0x1b, 0x18, 0x1c, 0x1e]$, $b = 0x16$), hence Category A; it is not affine-equivalent to x^3 . Unlike the power map itself (SAC deviation 0.5), this affine representative satisfies the strict avalanche criterion exactly: each of the 25 single-bit input changes flips each output bit for exactly 16 of the 32 inputs, with one fixed point, at the same 7 AND gates and depth 1 as x^5 (7/21/4 against 7/22/0). Stage-1 seed: a power-map lookup-table node (`novel_power_map_5b`) that Stage 2 lowered and re-wired (Section 6.5); Stage-4 post-processing reduced the archived 7/23/4 netlist to 7/21/4. The inverse re-uses the x^{25} core of the Gold x^5 row inside the two affine layers (Tier 1).
- **Inverse (S^{-1}):** $\deg(S^{-1}) = 3$, $\mathcal{D}(S^{-1}) = [3^{31}]$; cost: 10 AND, 41 XOR, 10 NOT, AND depth $D = 2$ (Tier 1: Affine-folded reuse of the x^{25} inverse core of the Gold x^5 circuit; unmasked NanGate 45 nm: $81.40 \mu\text{m}^2$ / 102.0 GE, delay 652.6 ps, logic depth 13; pipelined $d = 2$ DOM-indep: 86 DFF_X1 [40 gadget + 46 sync], $631.48 \mu\text{m}^2$ / 791.3 GE, stage delay 346.9 ps).

Lookup table (LUT[0...31]):

```
0x1e, 0x06, 0x14, 0x13, 0x00, 0x17, 0x10, 0x18, 0x1d, 0x1f, 0x0b, 0x16, 0x02, 0x0f, 0x0e, 0x1c
0x01, 0x12, 0x05, 0x09, 0x07, 0x1b, 0x19, 0x1a, 0x0d, 0x04, 0x15, 0x03, 0x0a, 0x0c, 0x08, 0x11
```

Netlist (7 AND, 21 XOR, 4 NOT, $D = 1$):

```
and_0 = x[0] & x[1]; xor_0 = x[0] ^ x[2]; xor_1 = x[1] ^ x[2];
xor_2 = x[1] ^ x[3]; xor_3 = x[0] ^ x[3]; and_1 = x[2] & x[3];
xor_4 = x[3] ^ x[4]; xor_5 = x[2] ^ x[4]; and_2 = x[1] & x[4];
and_3 = xor_0 & x[4]; and_4 = xor_0 & xor_2; and_5 = xor_1 & xor_4;
and_6 = xor_3 & xor_5; xor_6 = xor_5 ^ and_3; xor_7 = and_4 ^ xor_5;
xor_8 = x[0] ^ and_5; xor_9 = x[1] ^ and_6; xor_10 = and_0 ^ and_6;
xor_11 = and_2 ^ xor_7; xor_12 = and_1 ^ xor_8; xor_13 = x[3] ^ xor_9;
xor_14 = xor_9 ^ xor_7; xor_15 = xor_4 ^ xor_10; xor_16 = and_5 ^ xor_10;
xor_17 = xor_8 ^ xor_11; xor_18 = xor_13 ^ xor_11; xor_19 = xor_12 ^ xor_14;
xor_20 = xor_6 ^ xor_16; not_0 = ~xor_17; not_1 = ~xor_18; not_2 = ~xor_19;
not_3 = ~xor_20; y[0] = xor_15; y[1] = not_1; y[2] = not_3; y[3] = not_2;
y[4] = not_0;
```

Inverse lookup table (LUT $_{S^{-1}}$ [0...31]):

```
0x04, 0x10, 0x0c, 0x1b, 0x19, 0x12, 0x01, 0x14, 0x1e, 0x13, 0x1c, 0x0a, 0x1d, 0x18, 0x0e, 0x0d
0x06, 0x1f, 0x11, 0x03, 0x02, 0x1a, 0x0b, 0x05, 0x07, 0x16, 0x17, 0x15, 0x0f, 0x08, 0x00, 0x09
```

Inverse netlist (10 AND, 41 XOR, 10 NOT, $D = 2$):

```
xor_0 = x[0] ^ x[1]; xor_1 = x[3] ^ x[4]; xor_2 = x[2] ^ xor_0;
xor_3 = xor_0 ^ xor_1; xor_4 = x[0] ^ xor_1; xor_5 = x[4] ^ xor_2;
xor_6 = x[0] ^ x[2]; xor_7 = xor_1 ^ xor_2; xor_8 = x[1] ^ x[3];
xor_9 = x[3] ^ xor_6; xor_10 = x[4] ^ xor_0; xor_11 = x[4] ^ xor_6;
xor_12 = x[1] ^ x[4]; not_0 = ~xor_7; not_1 = ~xor_5; not_2 = ~xor_8;
not_3 = ~xor_4; not_4 = ~xor_9; not_5 = ~x[1]; not_6 = ~xor_0; not_7 = ~xor_1;
and_0 = not_0 & not_1; and_1 = not_2 & xor_3; and_2 = not_3 & not_4;
and_3 = not_5 & xor_2; and_4 = xor_10 & not_6; xor_13 = and_2 ^ and_3;
xor_14 = and_1 ^ xor_0; xor_15 = x[2] ^ xor_13; xor_16 = and_0 ^ and_3;
xor_17 = and_0 ^ xor_14; xor_18 = xor_4 ^ and_1; xor_19 = xor_13 ^ xor_18;
xor_20 = xor_5 ^ and_0; xor_21 = and_2 ^ and_4; xor_22 = xor_20 ^ xor_21;
xor_23 = xor_1 ^ xor_16; xor_24 = x[3] ^ and_2; xor_25 = xor_3 ^ and_4;
```

```

not_8 = ~xor_17; and_5 = xor_15 & not_7; and_6 = not_8 & x[0];
and_7 = xor_19 & xor_11; and_8 = xor_22 & xor_2; and_9 = xor_23 & not_6;
xor_26 = and_1 ^ and_6; xor_27 = and_7 ^ xor_26; xor_28 = x[3] ^ xor_27;
xor_29 = and_5 ^ and_9; xor_30 = xor_14 ^ and_9; xor_31 = xor_24 ^ xor_30;
xor_32 = xor_15 ^ and_5; xor_33 = and_8 ^ xor_28; xor_34 = xor_32 ^ xor_33;
xor_35 = xor_13 ^ xor_26; xor_36 = xor_29 ^ xor_12; xor_37 = xor_35 ^ xor_36;
xor_38 = xor_16 ^ xor_29; xor_39 = xor_25 ^ xor_38; xor_40 = and_4 ^ xor_28;
not_9 = ~xor_39; y[0] = xor_31; y[1] = xor_34; y[2] = xor_37; y[3] = not_9;
y[4] = xor_40;

```

C.7 5-bit χ_5 class (5-bit, Category A)

- **Profile:** $\delta = 8$, $L = 16$ (NL = 8), $\deg(S) = 2$, $\mathcal{D}(S) = [2^{31}]$, FP = 5, SAC deviation 0.5.
- **Cost:** 5 AND, 8 XOR, 1 NOT, AND depth $D = 1$ (unmasked NanGate 45 nm: $18.62 \mu\text{m}^2$ / 23.3 GE, delay 141.2 ps, logic depth 4; pipelined $d = 2$ DOM-indep: 28 DFF_X1 [20 gadget + 8 sync], $205.88 \mu\text{m}^2$ / 258.0 GE, stage delay 139.0 ps).
- **Remark:** Certified affine equivalent to KECCAK χ_5 ($\sim_{\text{aff}} \chi_5$, with $A = [0x01, 0x04, 0x02, 0x0c, 0x11]$, $a = 0x1f$, $B = [0x14, 0x04, 0x02, 0x08, 0x01]$, $b = 0x1b$), whose usual circuit has 5 AND, 5 XOR and 5 NOT gates [BDMD⁺20].
- **Inverse** (S^{-1}): $\deg(S^{-1}) = 3$, $\mathcal{D}(S^{-1}) = [3^{31}]$; cost: 9 AND, 21 XOR, 3 NOT, AND depth $D = 2$ (Tier 4: Hodge-dual linear-divisor coset-span $D = 2$ synthesis; unmasked NanGate 45 nm: $44.69 \mu\text{m}^2$ / 56.0 GE, delay 329.7 ps, logic depth 8; pipelined $d = 2$ DOM-indep: 62 DFF_X1 [36 gadget + 26 sync], $444.75 \mu\text{m}^2$ / 557.3 GE, stage delay 258.1 ps).

Lookup table (LUT[0...31]):

```

0x00, 0x15, 0x06, 0x13, 0x10, 0x04, 0x02, 0x16, 0x08, 0x1d, 0x0a, 0x1f, 0x0c, 0x18, 0x1a, 0x0e
0x14, 0x09, 0x12, 0x0f, 0x05, 0x19, 0x17, 0x0b, 0x1e, 0x03, 0x1c, 0x01, 0x1b, 0x07, 0x0d, 0x11

```

Netlist (5 AND, 8 XOR, 1 NOT, $D = 1$):

```

xor_11 = x[1] ^ x[2]; xor_7 = x[1] ^ x[3]; and_10 = x[0] & x[4];
and_14 = x[3] & x[4]; xor_15 = x[0] ^ x[4]; not_0 = ~xor_7;
and_13 = xor_11 & xor_7; xor_19 = x[3] ^ and_10; xor_22 = x[1] ^ and_14;
and_16 = xor_15 & x[2]; and_9 = not_0 & x[2]; xor_23 = x[0] ^ and_16;
xor_21 = xor_15 ^ and_13; xor_18 = xor_15 ^ and_9; y[0] = xor_23; y[1] = xor_22;
y[2] = xor_21; y[3] = xor_19; y[4] = xor_18;

```

Inverse lookup table (LUT $_{S^{-1}}$ [0...31]):

```

0x00, 0x1b, 0x06, 0x19, 0x05, 0x14, 0x02, 0x1d, 0x08, 0x11, 0x0a, 0x17, 0x0c, 0x1e, 0x0f, 0x13
0x04, 0x1f, 0x12, 0x03, 0x10, 0x01, 0x07, 0x16, 0x0d, 0x15, 0x0e, 0x1c, 0x1a, 0x09, 0x18, 0x0b

```

Inverse netlist (9 AND, 21 XOR, 3 NOT, $D = 2$):

```

xor_0 = x[0] ^ x[1]; xor_1 = x[3] ^ x[4]; xor_2 = x[2] ^ x[3];
xor_3 = x[2] ^ x[4]; and_0 = x[0] & x[3]; and_1 = xor_0 & xor_1;
and_2 = xor_2 & x[1]; and_3 = x[2] & x[4]; xor_4 = and_1 ^ x[4];
xor_5 = and_0 ^ x[0]; xor_6 = xor_4 ^ xor_5; not_0 = ~xor_6;
xor_7 = xor_2 ^ and_2; xor_8 = xor_4 ^ xor_7; not_1 = ~xor_8;
xor_9 = and_3 ^ x[0]; not_2 = ~xor_9; xor_10 = and_1 ^ x[2];
xor_11 = xor_5 ^ xor_10; xor_12 = x[1] ^ x[3]; xor_13 = xor_12 ^ and_0;
and_4 = not_0 & x[1]; and_5 = not_1 & x[0]; and_6 = not_2 & x[3];
and_7 = xor_11 & x[4]; and_8 = xor_3 & xor_13; xor_14 = and_5 ^ x[3];
xor_15 = and_7 ^ x[0]; xor_16 = and_8 ^ x[2]; xor_17 = xor_15 ^ xor_16;
xor_18 = and_6 ^ x[1]; xor_19 = xor_14 ^ xor_18; xor_20 = xor_3 ^ and_4;
y[0] = xor_17; y[1] = xor_19; y[2] = xor_20; y[3] = xor_14; y[4] = xor_15;

```

C.8 5-bit inverse-Gold class (x^{25}) (5-bit, Category A)

- **Profile:** $\delta = 2$, $L = 8$ (NL = 12), $\deg(S) = 3$, $\mathcal{D}(S) = [3^{31}]$, FP = 1, SAC deviation 0.125.
- **Cost:** 14 AND, 31 XOR, 2 NOT, AND depth $D = 2$ (unmasked NanGate 45 nm: $65.44 \mu\text{m}^2$ / 82.0 GE, delay 439.6 ps, logic depth 8; pipelined $d = 2$ DOM-indep: 88 DFF_X1 [56 gadget + 32 sync], $646.91 \mu\text{m}^2$ / 810.7 GE, stage delay 346.2 ps).
- **Remark:** Certified affine equivalent to x^7 (and hence to $x^{25} = (x^5)^{-1}$, $\sim_{\text{aff}} x^{25}$, with $S_{\text{ref}}(u) = u^7 \bmod (x^5 + x^2 + 1)$, $A = [0x02, 0x01, 0x08, 0x07, 0x17]$, $a = 0x01$, $B = [0x0c, 0x1b, 0x15, 0x0f, 0x10]$, $b = 0x14$), and not affine equivalent to x^{11} (x^{21}). Stage-1 seed: a power-map lookup-table node (`novel_power_map_5b`); Stage-4 post-processing reduced the archived 14/32/2 netlist to 14/31/2. Compared with the 10/37/2 circuit of [BDM26] it uses four more AND gates and six fewer XOR gates. The inverse is quadratic ($\sim_{\text{aff}} x^5$) and re-uses the 7-AND Gold x^5 core of SM C.5 inside two affine layers (Tier 1).
- **Inverse (S^{-1}):** $\deg(S^{-1}) = 2$, $\mathcal{D}(S^{-1}) = [2^{31}]$; cost: 7 AND, 26 XOR, 5 NOT, AND depth $D = 1$ (Tier 1: 3-stage affine-folded core reuse; unmasked NanGate 45 nm: $51.60 \mu\text{m}^2$ / 64.7 GE, delay 485.6 ps, logic depth 10; pipelined $d = 2$ DOM-indep: 38 DFF_X1 [28 gadget + 10 sync], $331.97 \mu\text{m}^2$ / 416.0 GE, stage delay 358.2 ps).

Lookup table (LUT[0...31]):

```
0x18, 0x01, 0x14, 0x11, 0x13, 0x1b, 0x0b, 0x16, 0x10, 0x12, 0x09, 0x04, 0x1d, 0x1c, 0x00, 0x07
0x08, 0x03, 0x0c, 0x1f, 0x1e, 0x17, 0x15, 0x0d, 0x02, 0x06, 0x0a, 0x05, 0x0e, 0x1a, 0x19, 0x0f
```

Netlist (14 AND, 31 XOR, 2 NOT, $D = 2$):

```
xor_0 = x[1] ^ x[3]; xor_1 = x[0] ^ x[3]; and_0 = x[0] & x[1];
and_1 = x[0] & x[2]; and_2 = x[0] & x[3]; and_3 = x[1] & x[3];
and_4 = x[2] & x[3]; and_5 = x[0] & x[4]; and_6 = x[1] & x[4];
and_7 = x[2] & x[4]; and_8 = x[3] & x[4]; xor_2 = x[0] ^ and_7;
xor_3 = x[1] ^ and_4; xor_4 = and_0 ^ and_3; xor_5 = and_1 ^ and_6;
xor_6 = x[0] ^ and_3; xor_7 = x[2] ^ and_6; xor_8 = and_0 ^ and_7;
xor_9 = x[1] ^ and_8; xor_10 = x[2] ^ and_5; xor_11 = and_3 ^ xor_5;
xor_12 = and_4 ^ and_8; xor_13 = and_7 ^ xor_3; xor_14 = x[4] ^ xor_2;
and_9 = xor_6 & xor_7; and_10 = xor_0 & xor_5; and_11 = xor_1 & xor_8;
and_12 = xor_2 & xor_3; and_13 = xor_9 & xor_10; xor_15 = and_2 ^ and_12;
xor_16 = x[2] ^ xor_15; xor_17 = and_5 ^ and_11; xor_18 = and_10 ^ xor_2;
xor_19 = and_13 ^ xor_4; xor_20 = xor_16 ^ xor_18; xor_21 = and_11 ^ xor_20;
xor_22 = xor_11 ^ xor_21; xor_23 = xor_16 ^ xor_17; xor_24 = xor_12 ^ xor_23;
xor_25 = xor_19 ^ xor_13; xor_26 = xor_0 ^ xor_20; xor_27 = and_9 ^ xor_15;
xor_28 = xor_17 ^ xor_19; xor_29 = xor_14 ^ xor_27; xor_30 = xor_28 ^ xor_29;
not_0 = ~xor_26; not_1 = ~xor_30; y[0] = xor_22; y[1] = xor_24; y[2] = xor_25;
y[3] = not_0; y[4] = not_1;
```

Inverse lookup table (LUT $_{S^{-1}}$ [0...31]):

```
0x0e, 0x01, 0x18, 0x11, 0x0b, 0x1b, 0x19, 0x0f, 0x10, 0x0a, 0x1a, 0x06, 0x12, 0x17, 0x1c, 0x1f
0x08, 0x03, 0x09, 0x04, 0x02, 0x16, 0x07, 0x15, 0x00, 0x1e, 0x1d, 0x05, 0x0d, 0x0c, 0x14, 0x13
```

Inverse netlist (7 AND, 26 XOR, 5 NOT, $D = 1$):

```
xor_0 = x[1] ^ x[2]; xor_1 = x[3] ^ x[4]; xor_2 = x[3] ^ xor_0;
xor_3 = x[4] ^ xor_0; xor_4 = x[2] ^ xor_1; xor_5 = x[2] ^ x[4];
xor_6 = xor_0 ^ xor_1; xor_7 = x[0] ^ xor_0; xor_8 = x[0] ^ x[1];
xor_9 = x[2] ^ x[3]; xor_10 = x[1] ^ x[3]; not_0 = ~xor_2; not_1 = ~xor_7;
not_2 = ~xor_0; not_3 = ~xor_9; and_0 = xor_3 & x[0]; and_1 = x[3] & x[0];
and_2 = xor_3 & xor_5; and_3 = not_0 & xor_6; and_4 = not_1 & xor_4;
and_5 = not_0 & xor_8; and_6 = not_2 & not_3; xor_11 = and_0 ^ and_4;
xor_12 = and_1 ^ and_6; xor_13 = and_5 ^ xor_11; xor_14 = and_3 ^ xor_12;
xor_15 = and_2 ^ xor_13; xor_16 = x[0] ^ xor_14; xor_17 = xor_4 ^ and_1;
xor_18 = xor_13 ^ xor_17; xor_19 = x[2] ^ and_5; xor_20 = xor_14 ^ xor_19;
xor_21 = and_0 ^ xor_16; xor_22 = and_6 ^ xor_15; xor_23 = xor_10 ^ xor_22;
xor_24 = xor_1 ^ xor_15; xor_25 = xor_16 ^ xor_24; not_4 = ~xor_25;
y[0] = xor_18; y[1] = xor_20; y[2] = xor_21; y[3] = xor_23; y[4] = not_4;
```

C.9 5-bit x^{15} (inversion class) (5-bit, Category A)

- **Profile:** $\delta = 2$, $L = 12$ (NL = 10), $\deg(S) = 4$, $\mathcal{D}(S) = [4^{31}]$, FP = 2, SAC deviation 0.125.
- **Cost:** 14 AND, 45 XOR, 0 NOT, AND depth $D = 2$ (unmasked NanGate 45 nm: $86.72 \mu\text{m}^2$ / 108.7 GE, delay 710.9 ps, logic depth 13; pipelined $d = 2$ DOM-indep: 76 DFF_X1 [56 gadget + 20 sync], $636.27 \mu\text{m}^2$ / 797.3 GE, stage delay 518.9 ps).
- **Remark:** The exact lookup table of x^{15} over $\mathbb{F}_2[x]/(x^5 + x^2 + 1)$, which is linearly equivalent to the inversion $x^{30} = (x^{15})^2$; 12/51/2 in [BDM26]: two more AND gates and six fewer XOR gates.
- **Inverse** (S^{-1}): $\deg(S^{-1}) = 4$, $\mathcal{D}(S^{-1}) = [4^{31}]$; cost: 14 AND, 57 XOR, 0 NOT, AND depth $D = 2$ (Tier 1: 3-stage affine-folded core reuse; unmasked NanGate 45 nm: $105.87 \mu\text{m}^2$ / 132.7 GE, delay 794.1 ps, logic depth 14; pipelined $d = 2$ DOM-indep: 88 DFF_X1 [56 gadget + 32 sync], $728.84 \mu\text{m}^2$ / 913.3 GE, stage delay 517.0 ps).

Lookup table (LUT[0...31]):

```
0x00, 0x01, 0x1f, 0x15, 0x12, 0x1c, 0x0a, 0x11, 0x1d, 0x02, 0x16, 0x04, 0x18, 0x0b, 0x19, 0x10, 0x09, 0x17, 0x1b, 0x14, 0x0e, 0x0c, 0x13, 0x03, 0x05, 0x08, 0x07, 0x0f, 0x1a, 0x1e, 0x06, 0x0d
```

Netlist (14 AND, 45 XOR, 0 NOT, $D = 2$):

```
a_in = in_val[4:0]; split_2 = a_in[0:0]; split_3 = a_in[1:1];
split_4 = a_in[2:2]; split_5 = a_in[3:3]; split_6 = a_in[4:4];
xor_7 = split_2 ^ split_3; xor_8 = split_2 ^ split_4; xor_9 = split_2 ^ split_5;
and_14 = split_4 & split_5; xor_21 = split_4 ^ split_5;
xor_11 = split_4 ^ split_6; and_13 = split_3 & split_6; xor_10 = split_6 ^ xor_7;
xor_12 = split_5 ^ xor_8; and_16 = xor_8 & split_6; and_15 = split_3 & xor_9;
xor_22 = split_6 ^ xor_21; and_18 = xor_9 & xor_11; xor_33 = split_5 ^ and_13;
and_17 = split_5 & xor_10; and_19 = xor_7 & xor_12; xor_23 = split_2 ^ and_16;
xor_30 = split_3 ^ xor_22; xor_20 = and_15 ^ and_17; xor_27 = split_6 ^ and_17;
xor_28 = and_19 ^ xor_23; xor_34 = and_14 ^ xor_23; xor_31 = xor_30 ^ and_14;
xor_24 = and_19 ^ xor_20; xor_26 = xor_22 ^ xor_20; xor_29 = xor_27 ^ xor_28;
xor_32 = and_16 ^ xor_31; xor_25 = and_18 ^ xor_24; xor_35 = xor_33 ^ xor_24;
xor_39 = xor_26 ^ xor_29; and_44 = xor_26 & xor_32; and_46 = xor_29 & xor_32;
xor_37 = xor_25 ^ xor_29; xor_56 = xor_25 ^ xor_32; xor_36 = xor_34 ^ xor_35;
and_47 = xor_25 & xor_39; xor_54 = xor_26 ^ and_46; xor_42 = xor_32 ^ xor_37;
xor_38 = xor_26 ^ xor_36; xor_43 = xor_29 ^ xor_36; and_45 = xor_26 & xor_36;
xor_40 = xor_32 ^ xor_38; xor_41 = xor_25 ^ xor_38; and_50 = xor_42 & xor_43;
xor_51 = xor_32 ^ and_45; and_48 = xor_25 & xor_40; and_49 = xor_37 & xor_41;
xor_57 = xor_56 ^ and_50; xor_52 = and_50 ^ xor_51; xor_59 = and_46 ^ and_48;
xor_61 = and_47 ^ and_48; xor_65 = xor_25 ^ and_49; xor_58 = xor_54 ^ xor_57;
xor_55 = xor_52 ^ xor_43; xor_60 = xor_51 ^ xor_59; xor_62 = xor_54 ^ xor_61;
xor_66 = xor_65 ^ xor_52; xor_64 = and_44 ^ xor_55; xor_63 = xor_62 ^ xor_55;
concat_67 = { xor_66, xor_64, xor_63, xor_60, xor_58 }; out_val = concat_67;
```

Inverse lookup table (LUT $_{S^{-1}}$ [0...31]):

```
0x00, 0x01, 0x09, 0x17, 0x0b, 0x18, 0x1e, 0x1a, 0x19, 0x10, 0x06, 0x0d, 0x15, 0x1f, 0x14, 0x1b, 0x0f, 0x07, 0x04, 0x16, 0x13, 0x03, 0x0a, 0x11, 0x0c, 0x0e, 0x1c, 0x12, 0x05, 0x08, 0x1d, 0x02
```

Inverse netlist (14 AND, 57 XOR, 0 NOT, $D = 2$):

```
xor_0 = x[2] ^ x[3]; xor_1 = x[0] ^ x[1]; xor_2 = xor_0 ^ x[4];
xor_3 = x[0] ^ x[2]; xor_4 = x[0] ^ x[3]; xor_5 = x[2] ^ x[4];
xor_6 = xor_1 ^ x[4]; xor_7 = xor_0 ^ x[0]; xor_8 = xor_2 ^ x[1];
and_0 = x[2] & x[3]; and_1 = x[1] & x[4]; and_2 = xor_3 & x[4];
and_3 = xor_4 & x[1]; and_4 = xor_4 & xor_5; xor_9 = and_1 ^ x[3];
and_5 = xor_6 & x[3]; and_6 = xor_1 & xor_7; xor_10 = and_2 ^ x[0];
xor_11 = and_3 ^ and_5; xor_12 = and_5 ^ x[4]; xor_13 = and_6 ^ xor_10;
xor_14 = and_0 ^ xor_10; xor_15 = and_0 ^ xor_8; xor_16 = and_6 ^ xor_11;
xor_17 = xor_2 ^ xor_11; xor_18 = xor_12 ^ xor_13; xor_19 = and_2 ^ xor_15;
xor_20 = and_4 ^ xor_16; xor_21 = xor_16 ^ xor_9; xor_22 = xor_17 ^ xor_18;
and_7 = xor_17 & xor_19; and_8 = xor_18 & xor_19; xor_23 = xor_20 ^ xor_18;
xor_24 = xor_14 ^ xor_21; and_9 = xor_20 & xor_22; xor_25 = xor_19 ^ xor_23;
xor_26 = xor_17 ^ xor_24; xor_27 = xor_18 ^ xor_24; and_10 = xor_17 & xor_24;
xor_28 = xor_19 ^ xor_26; xor_29 = xor_20 ^ xor_26; and_11 = xor_25 & xor_27;
and_12 = xor_20 & xor_28; and_13 = xor_23 & xor_29; xor_30 = and_9 ^ and_12;
xor_31 = and_4 ^ and_6; xor_32 = and_13 ^ and_11; xor_33 = and_3 ^ x[2];
```

```

xor_34 = x[3] ^ x[4]; xor_35 = and_5 ^ and_7; xor_36 = xor_30 ^ xor_33;
xor_37 = xor_34 ^ xor_35; xor_38 = and_2 ^ x[1]; xor_39 = and_0 ^ xor_31;
xor_40 = xor_32 ^ xor_38; xor_41 = and_1 ^ xor_36; xor_42 = xor_40 ^ xor_41;
xor_43 = and_1 ^ and_13; xor_44 = xor_43 ^ xor_37; xor_45 = xor_44 ^ xor_39;
xor_46 = and_8 ^ xor_36; xor_47 = xor_46 ^ xor_37; xor_48 = and_7 ^ and_8;
xor_49 = and_10 ^ xor_48; xor_50 = xor_30 ^ xor_49; xor_51 = xor_50 ^ xor_39;
xor_52 = xor_40 ^ xor_51; xor_53 = and_8 ^ xor_11; xor_54 = and_12 ^ xor_53;
xor_55 = xor_31 ^ xor_54; xor_56 = xor_32 ^ xor_55; y[0] = xor_42;
y[1] = xor_45; y[2] = xor_47; y[3] = xor_52; y[4] = xor_56;

```

C.10 YSH_{6,1}: 6-bit quartic, $[3^7, 4^{56}]$, $\delta = 4$, $D = 3$ (6-bit, Category B)

- **Profile:** $\delta = 4$, $L = 16$ (NL = 24), $\deg(S) = 4$, $\mathcal{D}(S) = \mathcal{D}(S^{-1}) = [3^7, 4^{56}]$, FP = 0, SAC deviation 0.125.
- **Cost:** 9 AND, 27 XOR, 1 NOT, AND depth $D = 3$ (unmasked NanGate 45 nm: $53.20 \mu\text{m}^2 / 66.7$ GE, delay 916.7 ps, logic depth 16; pipelined $d = 2$ DOM-indep: 74 DFF_X1 [36 gadget + 38 sync], $517.10 \mu\text{m}^2 / 648.0$ GE, stage delay 561.1 ps).
- **Remark:** Balanced 3-round generalized Feistel permutation with an asymmetric degree-3 middle round: degree 4 at $\delta = 4$, $L = 16$ with 9 AND gates (3 at each of the three multiplicative levels) and minimum component degree $d_{\min}(S) = d_{\min}(S^{-1}) = 3$, eliminating all quadratic components ($\mathcal{D}(S) = \mathcal{D}(S^{-1}) = [3^7, 4^{56}]$, matching Dillon's APN degree spectrum), while inheriting the structural 3-round Feistel boomerang uniformity $\beta = 64$ (Section 6.6).
- **Inverse (YSH_{6,1}⁻¹):** $\deg(S^{-1}) = 4$, $\mathcal{D}(S^{-1}) = [3^7, 4^{56}]$; cost: 9 AND, 27 XOR, 1 NOT, AND depth $D = 3$ (Tier 2: Exact 3-round generalized Feistel layer reversal; unmasked NanGate 45 nm: $53.20 \mu\text{m}^2 / 66.7$ GE, delay 764.6 ps, logic depth 13; pipelined $d = 2$ DOM-indep: 78 DFF_X1 [36 gadget + 42 sync], $535.19 \mu\text{m}^2 / 670.7$ GE, stage delay 407.4 ps).

Lookup table (LUT[0...63]):

```

0x18, 0x32, 0x0d, 0x05, 0x3a, 0x10, 0x02, 0x2f, 0x31, 0x28, 0x19, 0x3d, 0x30, 0x08, 0x1b, 0x25
0x38, 0x20, 0x09, 0x00, 0x11, 0x2e, 0x0c, 0x1a, 0x23, 0x14, 0x2c, 0x15, 0x36, 0x07, 0x3b, 0x22
0x0a, 0x39, 0x1e, 0x2d, 0x0b, 0x26, 0x16, 0x01, 0x1f, 0x03, 0x34, 0x37, 0x3f, 0x1c, 0x33, 0x2b
0x0e, 0x12, 0x3c, 0x13, 0x04, 0x21, 0x3e, 0x0f, 0x29, 0x06, 0x27, 0x24, 0x1d, 0x2a, 0x35, 0x17

```

Netlist (9 AND, 27 XOR, 1 NOT, $D = 3$):

```

and_0 = x[0] & x[1]; and_1 = x[1] & x[2]; and_2 = x[2] & x[0];
xor_0 = x[4] ^ and_1; xor_1 = x[5] ^ and_2; xor_2 = xor_0 ^ x[0];
xor_3 = xor_0 ^ x[2]; xor_4 = xor_0 ^ x[3]; xor_5 = xor_2 ^ and_0;
xor_6 = xor_2 ^ x[2]; xor_7 = xor_4 ^ xor_5; xor_8 = xor_7 ^ x[1];
xor_9 = xor_1 ^ xor_8; and_3 = xor_2 & xor_9; not_0 = ~xor_9;
xor_10 = xor_7 ^ xor_6; xor_11 = not_0 ^ x[1]; and_4 = xor_9 & xor_10;
xor_12 = x[2] ^ and_4; xor_13 = xor_11 ^ xor_3; xor_14 = xor_12 ^ xor_2;
xor_15 = xor_13 ^ and_3; xor_16 = xor_15 ^ xor_10; and_5 = xor_16 & xor_14;
and_6 = xor_10 & xor_2; xor_17 = xor_0 ^ and_6; xor_18 = xor_17 ^ xor_9;
and_7 = xor_14 & xor_18; xor_19 = not_0 ^ and_7; and_8 = xor_18 & xor_16;
xor_20 = xor_10 ^ and_8; xor_21 = xor_19 ^ xor_16; xor_22 = xor_20 ^ xor_14;
xor_23 = xor_21 ^ xor_2; xor_24 = xor_19 ^ xor_18; xor_25 = xor_23 ^ and_5;
xor_26 = xor_25 ^ xor_18; y[0] = xor_22; y[1] = xor_26; y[2] = xor_21;
y[3] = xor_24; y[4] = xor_19; y[5] = xor_14;

```

Inverse lookup table (LUT_{S⁻¹}[0...63]):

```

0x13, 0x27, 0x06, 0x29, 0x34, 0x03, 0x39, 0x1d, 0x0d, 0x12, 0x20, 0x24, 0x16, 0x02, 0x30, 0x37
0x05, 0x14, 0x31, 0x33, 0x19, 0x1b, 0x26, 0x3f, 0x00, 0x0a, 0x17, 0x0e, 0x2d, 0x3c, 0x22, 0x28
0x11, 0x35, 0x1f, 0x18, 0x3b, 0x0f, 0x25, 0x3a, 0x09, 0x38, 0x3d, 0x2f, 0x1a, 0x23, 0x15, 0x07
0x0c, 0x08, 0x01, 0x2e, 0x2a, 0x3e, 0x1c, 0x2b, 0x10, 0x21, 0x04, 0x1e, 0x32, 0x0b, 0x36, 0x2c

```

Inverse netlist (9 AND, 27 XOR, 1 NOT, $D = 3$):

```

xor_0 = x[2] ^ x[4]; xor_1 = x[3] ^ x[4]; and_0 = x[5] & xor_0;
and_1 = x[5] & xor_1; and_2 = xor_1 & xor_0; xor_2 = x[0] ^ x[5];
xor_3 = and_2 ^ xor_2; xor_4 = and_1 ^ x[4]; not_0 = ~xor_4;
xor_5 = x[1] ^ xor_1; xor_6 = and_0 ^ xor_5; xor_7 = x[2] ^ xor_6;

```

```

and_3 = xor_7 & not_0; and_4 = xor_3 & not_0; and_5 = xor_3 & xor_7;
xor_8 = not_0 ^ xor_1; xor_9 = and_5 ^ xor_8; xor_10 = x[5] ^ xor_7;
xor_11 = and_4 ^ xor_10; xor_12 = xor_3 ^ xor_0; xor_13 = and_3 ^ xor_12;
xor_14 = xor_11 ^ xor_9; xor_15 = xor_14 ^ xor_13; xor_16 = xor_15 ^ xor_4;
xor_17 = xor_7 ^ xor_9; and_6 = xor_17 & xor_16; and_7 = xor_16 & xor_11;
and_8 = xor_17 & xor_11; xor_18 = and_7 ^ xor_9; xor_19 = xor_11 ^ xor_7;
xor_20 = xor_3 ^ xor_19; xor_21 = and_6 ^ xor_7; xor_22 = xor_20 ^ xor_21;
xor_23 = xor_9 ^ xor_22; xor_24 = xor_16 ^ xor_20; xor_25 = xor_24 ^ not_0;
xor_26 = and_8 ^ xor_25; y[0] = xor_17; y[1] = xor_16; y[2] = xor_11;
y[3] = xor_23; y[4] = xor_18; y[5] = xor_26;

```

C.11 YSH_{6,2}: 6-bit quartic, $[3^{31}, 4^{32}]$, $\delta = 4$, $D = 3$ (6-bit, Category B)

- **Profile:** $\delta = 4$, $L = 16$ (NL = 24), $\deg(S) = 4$, $\mathcal{D}(S) = [3^{31}, 4^{32}]$, FP = 2, SAC deviation 0.25.
- **Cost:** 27 AND, 64 XOR, 0 NOT, AND depth $D = 3$ (unmasked NanGate 45 nm: $130.87 \mu\text{m}^2$ / 164.0 GE, delay 502.4 ps, logic depth 10; pipelined $d = 2$ DOM-indep: 172 DFF_X1 [108 gadget + 64 sync], $1269.35 \mu\text{m}^2$ / 1590.7 GE, stage delay 434.1 ps).
- **Remark:** Bipartite spectrum $[3^{31}, 4^{32}]$ at $\delta = 4$, $L = 16$. YSH_{6,1} (SM C.10) has the same $(\delta, L, \deg, D)$ with far fewer forward gates (9 AND, 27 XOR) and a $k = 3$ spectrum $[3^7, 4^{56}]$, whereas YSH_{6,2}⁻¹ unwinds to depth $D^{-1} = 2$.
- **Inverse** (YSH_{6,2}⁻¹): $\deg(S^{-1}) = 3$, $\mathcal{D}(S^{-1}) = [2^{15}, 3^{48}]$; cost: 9 AND, 37 XOR, 0 NOT, AND depth $D = 2$ (Tier 2: Output shear unwinding + 8-AND Q_{2258} quadratic inverse core; unmasked NanGate 45 nm: $68.63 \mu\text{m}^2$ / 86.0 GE, delay 468.6 ps, logic depth 8; pipelined $d = 2$ DOM-indep: 76 DFF_X1 [36 gadget + 40 sync], $557.54 \mu\text{m}^2$ / 698.7 GE, stage delay 387.3 ps).

Lookup table (LUT[0...63]):

```

0x00, 0x01, 0x07, 0x02, 0x03, 0x04, 0x05, 0x06, 0x0b, 0x08, 0x1f, 0x1e, 0x2d, 0x2a, 0x39, 0x3c
0x3d, 0x35, 0x38, 0x33, 0x16, 0x1b, 0x14, 0x1a, 0x12, 0x2e, 0x29, 0x10, 0x0c, 0x37, 0x31, 0x0f
0x19, 0x17, 0x26, 0x2c, 0x18, 0x15, 0x22, 0x2b, 0x24, 0x3a, 0x11, 0x0d, 0x3f, 0x20, 0x0e, 0x13
0x2f, 0x23, 0x34, 0x3b, 0x32, 0x3e, 0x28, 0x27, 0x36, 0x1c, 0x0a, 0x25, 0x21, 0x09, 0x1d, 0x30

```

Netlist (27 AND, 64 XOR, 0 NOT, $D = 3$):

```

a_in = in_val[5:0];
split_2 = a_in[0:0]; split_3 = a_in[1:1]; split_4 = a_in[2:2];
split_5 = a_in[3:3]; split_6 = a_in[4:4]; split_7 = a_in[5:5];
and_8 = split_2 & split_3; and_9 = split_2 & split_4; and_13 = split_3 & split_4;
paar_shared_xor_1_split_3_split_4 = split_3 ^ split_4;
and_10 = split_2 & split_5; and_14 = split_3 & split_5;
and_17 = split_4 & split_5; and_11 = split_2 & split_6;
and_15 = split_3 & split_6; and_18 = split_4 & split_6;
and_20 = split_5 & split_6; xor_34 = split_3 ^ split_6;
xor_49 = split_5 ^ split_6; and_12 = split_2 & split_7;
and_16 = split_3 & split_7; and_19 = split_4 & split_7;
and_21 = split_5 & split_7; and_22 = split_6 & split_7;
xor_43 = split_4 ^ split_7; xor_83 = split_3 ^ and_14; xor_29 = split_3 ^ and_17;
xor_37 = split_3 ^ and_11; xor_44 = and_10 ^ and_15; xor_25 = split_2 ^ and_18;
xor_38 = and_10 ^ and_18; xor_26 = split_7 ^ and_20; xor_32 = and_11 ^ and_12;
xor_41 = and_12 ^ and_18; xor_50 = and_12 ^ and_14; xor_55 = split_6 ^ and_12;
paar_shared_xor_0_and_12_and_17 = and_12 ^ and_17; xor_47 = and_10 ^ and_16;
xor_57 = and_14 ^ and_16; xor_28 = split_2 ^ and_19; xor_35 = and_17 ^ and_19;
xor_52 = split_6 ^ and_19; xor_46 = split_4 ^ and_21; xor_40 = split_3 ^ and_22;
xor_65 = split_5 ^ and_22; xor_73 = split_2 ^ paar_shared_xor_1_split_3_split_4;
xor_79 = split_6 ^ paar_shared_xor_1_split_3_split_4; and_45 = xor_43 & xor_44;
and_39 = xor_37 & xor_38; and_27 = xor_25 & xor_26;
and_33 = paar_shared_xor_1_split_3_split_4 & xor_32; and_51 = xor_49 & xor_50;
xor_60 = and_18 ^ xor_57; xor_91 = and_21 ^ xor_57; and_30 = xor_28 & xor_29;
and_36 = xor_34 & xor_35; and_48 = xor_46 & xor_47; and_42 = xor_40 & xor_41;
xor_74 = xor_73 ^ and_13; xor_67 = and_20 ^ and_45; xor_66 = and_9 ^ and_39;
xor_84 = xor_83 ^ and_33; xor_61 = and_17 ^ and_51;
and_24 = split_3 & paar_shared_xor_0_and_12_and_17;
and_54 = xor_52 & paar_shared_xor_0_and_12_and_17; xor_56 = and_30 ^ xor_55;

```

```

xor_94 = xor_60 ^ and_36; xor_58 = split_7 ^ and_48; xor_62 = and_21 ^ and_42;
xor_75 = xor_74 ^ and_33; xor_85 = xor_61 ^ xor_67; xor_68 = and_24 ^ xor_61;
xor_80 = xor_79 ^ and_54; xor_59 = and_8 ^ xor_56; xor_63 = and_11 ^ xor_58;
xor_92 = xor_91 ^ xor_58; xor_69 = xor_62 ^ xor_66; xor_95 = xor_62 ^ xor_67;
xor_87 = xor_84 ^ xor_85; xor_71 = and_27 ^ xor_59; xor_72 = xor_59 ^ xor_65;
xor_70 = xor_60 ^ xor_63; xor_76 = xor_63 ^ xor_68; xor_86 = xor_56 ^ xor_69;
xor_96 = xor_68 ^ xor_95; xor_93 = xor_92 ^ xor_71; xor_81 = xor_69 ^ xor_70;
xor_89 = and_51 ^ xor_70; xor_77 = xor_76 ^ xor_72; xor_88 = xor_86 ^ xor_87;
xor_97 = xor_94 ^ xor_96; xor_82 = xor_80 ^ xor_81; xor_90 = xor_72 ^ xor_89;
xor_78 = xor_75 ^ xor_77; xor_98 = xor_71 ^ xor_97;
and_shear = xor_78 & xor_90; xor_shear = xor_82 ^ and_shear;
concat_99 = { xor_98, xor_93, xor_90, xor_88, xor_shear, xor_78 };
out_val = concat_99;

```

Inverse lookup table ($\text{LUT}_{S^{-1}}[0 \dots 63]$):

```

0x00, 0x01, 0x03, 0x04, 0x05, 0x06, 0x07, 0x02, 0x09, 0x3d, 0x3a, 0x08, 0x1c, 0x2b, 0x2e, 0x1f
0x1b, 0x2a, 0x18, 0x2f, 0x16, 0x25, 0x14, 0x21, 0x24, 0x20, 0x17, 0x15, 0x39, 0x3e, 0x0b, 0x0a
0x2d, 0x3c, 0x26, 0x31, 0x28, 0x3b, 0x22, 0x37, 0x36, 0x1a, 0x0d, 0x27, 0x23, 0x0c, 0x19, 0x30
0x3f, 0x1e, 0x34, 0x13, 0x32, 0x11, 0x38, 0x1d, 0x12, 0x0e, 0x29, 0x33, 0x0f, 0x10, 0x35, 0x2c

```

Inverse netlist (9 AND, 37 XOR, 0 NOT, $D = 2$):

```

xor_0 = x[3] ^ x[4]; xor_1 = x[0] ^ xor_0; xor_2 = x[2] ^ x[5];
xor_3 = x[4] ^ xor_2; xor_4 = x[3] ^ xor_2; xor_5 = x[4] ^ x[5];
xor_6 = x[5] ^ xor_1; xor_7 = xor_1 ^ xor_2; xor_8 = x[2] ^ x[3];
xor_9 = x[0] ^ x[5]; xor_10 = x[0] ^ x[2]; and_0 = x[0] & x[3];
and_1 = xor_1 & xor_4; and_2 = xor_5 & xor_6; and_3 = xor_0 & xor_7;
xor_11 = x[1] ^ and_0; xor_12 = x[5] ^ xor_11; xor_13 = x[2] ^ xor_11;
xor_14 = x[3] ^ xor_12; xor_15 = xor_11 ^ xor_1; xor_16 = x[0] ^ xor_14;
xor_17 = xor_3 ^ xor_11; xor_18 = x[3] ^ xor_15; xor_19 = xor_0 ^ xor_13;
xor_20 = and_2 ^ xor_12; xor_21 = and_3 ^ xor_15; and_4 = x[3] & xor_13;
and_5 = xor_8 & xor_15; and_6 = xor_9 & xor_16; and_7 = x[0] & xor_17;
and_8 = xor_18 & xor_19; xor_22 = and_3 ^ and_7; xor_23 = and_1 ^ xor_22;
xor_24 = and_2 ^ and_4; xor_25 = and_5 ^ and_8; xor_26 = xor_10 ^ and_8;
xor_27 = xor_22 ^ xor_24; xor_28 = xor_26 ^ xor_27; xor_29 = xor_20 ^ and_7;
xor_30 = xor_3 ^ xor_23; xor_31 = xor_21 ^ and_6; xor_32 = xor_25 ^ xor_31;
xor_33 = x[5] ^ xor_24; xor_34 = xor_14 ^ and_4; xor_35 = xor_23 ^ xor_25;
xor_36 = xor_34 ^ xor_35; y[0] = xor_28; y[1] = xor_29; y[2] = xor_30;
y[3] = xor_32; y[4] = xor_33; y[5] = xor_36;

```

C.12 YSH_{6,3}: 6-bit cubic, $\delta = 4$, 9 AND (6-bit, Category B)

- **Profile:** $\delta = 4$, $L = 16$ (NL = 24), $\deg(S) = 3$, $\mathcal{D}(S) = [3^{63}]$, FP = 2, SAC deviation 0.5.
- **Cost:** 9 AND, 17 XOR, 9 NOT, AND depth $D = 3$ (unmasked NanGate 45 nm: $41.50 \mu\text{m}^2 / 52.0$ GE, delay 411.0 ps, logic depth 11; pipelined $d = 2$ DOM-indep: 72 DFF_X1 [36 gadget + 36 sync], $480.40 \mu\text{m}^2 / 602.0$ GE, stage delay 254.4 ps).
- **Remark:** Same $\mathcal{D}(S) = [3^{63}]$ as the x^{13} class, from which the DDT and |LAT| histograms separate it. The circuit of SM C.17 has the same $(\delta, L, \deg, D)$ and 9 AND gates with two fewer XOR gates.
- **Inverse** ($\text{YSH}_{6,3}^{-1}$): $\deg(S^{-1}) = 4$, $\mathcal{D}(S^{-1}) = [3^{31}, 4^{32}]$; cost: 9 AND, 17 XOR, 8 NOT, AND depth $D = 3$ (Tier 2: Output shear unwinding + 3-round χ_3 -Feistel reversal; unmasked NanGate 45 nm: $40.96 \mu\text{m}^2 / 51.3$ GE, delay 399.2 ps, logic depth 10; pipelined $d = 2$ DOM-indep: 72 DFF_X1 [36 gadget + 36 sync], $479.86 \mu\text{m}^2 / 601.3$ GE, stage delay 317.0 ps).

Lookup table ($\text{LUT}[0 \dots 63]$):

```

0x00, 0x03, 0x06, 0x01, 0x05, 0x04, 0x02, 0x07, 0x19, 0x3c, 0x2b, 0x18, 0x0f, 0x0e, 0x3d, 0x2a
0x30, 0x17, 0x3a, 0x15, 0x1f, 0x38, 0x32, 0x1d, 0x08, 0x0b, 0x37, 0x2e, 0x34, 0x11, 0x2d, 0x12
0x28, 0x35, 0x27, 0x3b, 0x3f, 0x2c, 0x23, 0x31, 0x20, 0x1e, 0x1b, 0x36, 0x25, 0x33, 0x0c, 0x09
0x10, 0x29, 0x16, 0x22, 0x2f, 0x1a, 0x1c, 0x24, 0x39, 0x21, 0x0a, 0x0d, 0x14, 0x26, 0x13, 0x3e

```

Netlist (9 AND, 17 XOR, 9 NOT, $D = 3$):

```

not_0 = ~x[0]; not_1 = ~x[1]; not_2 = ~x[2]; and_0 = not_0 & x[1];
and_1 = not_1 & x[2]; and_2 = not_2 & x[0]; xor_0 = x[4] ^ and_2;
xor_1 = x[3] ^ and_1; xor_2 = x[5] ^ and_0; xor_3 = x[0] ^ xor_1;
xor_4 = x[1] ^ xor_0; xor_5 = x[2] ^ xor_2; not_3 = ~xor_5; not_4 = ~xor_3;
not_5 = ~xor_4; and_3 = not_3 & xor_3; and_4 = not_4 & xor_4;
and_5 = not_5 & xor_5; xor_6 = and_3 ^ xor_0; xor_7 = and_4 ^ xor_2;
xor_8 = and_5 ^ xor_1; xor_9 = x[0] ^ and_5; xor_10 = xor_6 ^ xor_9;
xor_11 = x[1] ^ and_3; xor_12 = x[2] ^ and_4; not_6 = ~xor_6; not_7 = ~xor_7;
not_8 = ~xor_8; and_6 = not_6 & xor_7; and_7 = not_7 & xor_8;
and_8 = not_8 & xor_6; xor_13 = xor_10 ^ and_6; xor_14 = and_8 ^ xor_13;
xor_15 = xor_11 ^ and_7; xor_16 = xor_12 ^ and_8; y[0] = xor_14; y[1] = xor_15;
y[2] = xor_16; y[3] = xor_8; y[4] = xor_6; y[5] = xor_7;

```

Inverse lookup table ($\text{LUT}_{S^{-1}}[0 \dots 63]$):

```

0x00, 0x03, 0x06, 0x01, 0x05, 0x04, 0x02, 0x07, 0x18, 0x2f, 0x3a, 0x19, 0x2e, 0x3b, 0x0d, 0x0c
0x30, 0x1d, 0x1f, 0x3e, 0x3c, 0x13, 0x32, 0x11, 0x0b, 0x08, 0x35, 0x2a, 0x36, 0x17, 0x29, 0x14
0x28, 0x39, 0x33, 0x26, 0x37, 0x2c, 0x3d, 0x22, 0x20, 0x31, 0x0f, 0x0a, 0x25, 0x1e, 0x1b, 0x34
0x10, 0x27, 0x16, 0x2d, 0x1c, 0x21, 0x2b, 0x1a, 0x15, 0x38, 0x12, 0x23, 0x09, 0x0e, 0x3f, 0x24

```

Inverse netlist (9 AND, 17 XOR, 8 NOT, $D = 3$):

```

xor_0 = x[2] ^ x[4]; not_0 = ~x[4]; not_1 = ~x[5]; and_0 = x[3] & x[4];
and_1 = not_0 & x[5]; and_2 = not_1 & x[3]; xor_1 = x[0] ^ and_0;
xor_2 = x[1] ^ and_2; xor_3 = and_0 ^ xor_0; xor_4 = and_1 ^ xor_1;
xor_5 = x[3] ^ xor_4; xor_6 = x[4] ^ xor_2; xor_7 = x[5] ^ xor_3;
not_2 = ~xor_7; not_3 = ~xor_5; not_4 = ~xor_6; and_3 = not_2 & xor_5;
and_4 = not_3 & xor_6; and_5 = not_4 & xor_7; xor_8 = and_3 ^ xor_2;
xor_9 = and_4 ^ xor_3; xor_10 = and_5 ^ xor_4; xor_11 = x[3] ^ and_5;
xor_12 = x[4] ^ and_3; xor_13 = x[5] ^ and_4; not_5 = ~xor_10; not_6 = ~xor_8;
not_7 = ~xor_9; and_6 = xor_8 & not_5; and_7 = xor_9 & not_6;
and_8 = xor_10 & not_7; xor_14 = xor_11 ^ and_7; xor_15 = xor_12 ^ and_8;
xor_16 = xor_13 ^ and_6; y[0] = xor_10; y[1] = xor_8; y[2] = xor_9;
y[3] = xor_14; y[4] = xor_15; y[5] = xor_16;

```

C.13 YSH_{6,4}: 6-bit degree 5, $\delta = 8$, $L = 28$ (6-bit, Category B)

- **Profile:** $\delta = 8$, $L = 28$ (NL = 18), $\deg(S) = 5$, $\mathcal{D}(S) = [2^{31}, 5^{32}]$, FP = 2, SAC deviation 0.5.
- **Cost:** 11 AND, 22 XOR, 1 NOT, AND depth $D = 3$ (unmasked NanGate 45 nm: $47.35 \mu\text{m}^2 / 59.3 \text{ GE}$, delay 427.0 ps, logic depth 9; pipelined $d = 2$ DOM-indep: 80 DFF_X1 [44 gadget + 36 sync], $549.56 \mu\text{m}^2 / 688.7 \text{ GE}$, stage delay 358.1 ps).
- **Remark:** Same δ and degree as the SPEEDY S-box (20/51/3 in [BDM26]) but a different class and a higher linearity (28; SPEEDY has $L = 24$): nine fewer AND gates and 29 fewer XOR gates at the same depth. Structure and origin: the nine level-1 products are exactly those of the Q_{2258} library seed circuit (9/30, $D = 1$; the un-sheared core is affine-equivalent to x^5), followed by two chained Stage-2 shear gates at levels 2 and 3 (Section 6.5).
- **Inverse** (YSH_{6,4}⁻¹): $\deg(S^{-1}) = 5$, $\mathcal{D}(S^{-1}) = [3^{15}, 4^{16}, 5^{32}]$; cost: 22 AND, 49 XOR, 1 NOT, AND depth $D = 3$ (Tier 2: Parallel $D = 3$ directional-derivative inversion on 18-AND tower core; unmasked NanGate 45 nm: $102.14 \mu\text{m}^2 / 128.0 \text{ GE}$, delay 632.8 ps, logic depth 12; pipelined $d = 2$ DOM-indep: 132 DFF_X1 [88 gadget + 44 sync], $987.92 \mu\text{m}^2 / 1238.0 \text{ GE}$, stage delay 480.4 ps).

Lookup table ($\text{LUT}[0 \dots 63]$):

```

0x02, 0x03, 0x01, 0x06, 0x07, 0x04, 0x05, 0x00, 0x1b, 0x0a, 0x38, 0x2f, 0x1e, 0x1d, 0x2c, 0x29
0x19, 0x28, 0x1a, 0x2d, 0x14, 0x27, 0x16, 0x23, 0x26, 0x17, 0x15, 0x22, 0x2b, 0x08, 0x09, 0x3c
0x3f, 0x3e, 0x24, 0x33, 0x2a, 0x39, 0x20, 0x35, 0x34, 0x25, 0x1f, 0x18, 0x21, 0x32, 0x0b, 0x0e
0x3d, 0x1c, 0x36, 0x11, 0x30, 0x13, 0x3a, 0x0f, 0x10, 0x31, 0x3b, 0x0c, 0x0d, 0x2e, 0x37, 0x12

```

Netlist (11 AND, 22 XOR, 1 NOT, $D = 3$):

```

xor_0 = x[1] ^ x[4]; xor_1 = x[2] ^ x[5]; xor_2 = x[0] ^ x[3];
and_0 = x[0] & x[1]; and_1 = x[0] & x[2]; and_2 = x[1] & x[2];
and_3 = x[3] & x[4]; and_4 = x[3] & x[5]; and_5 = x[4] & x[5];
and_6 = xor_2 & xor_0; and_7 = xor_2 & xor_1; and_8 = xor_0 & xor_1;
xor_3 = and_0 ^ and_3; xor_4 = and_1 ^ xor_0; xor_5 = x[4] ^ and_8;
xor_6 = xor_3 ^ xor_4; xor_7 = x[3] ^ xor_5; xor_8 = and_5 ^ xor_1;
xor_9 = and_0 ^ and_6; xor_10 = and_2 ^ xor_7; xor_11 = and_2 ^ xor_2;
xor_12 = and_4 ^ xor_6; xor_13 = xor_8 ^ xor_11; xor_14 = xor_0 ^ xor_13;
xor_15 = x[5] ^ xor_10; xor_16 = xor_1 ^ xor_3; xor_17 = x[4] ^ and_1;
xor_18 = and_7 ^ xor_9; xor_19 = xor_17 ^ xor_18; xor_20 = x[5] ^ xor_9;
not_0 = ~xor_12; and_9 = xor_15 & not_0; and_10 = and_9 & xor_14;
xor_21 = xor_19 ^ and_10; y[0] = xor_14; y[1] = not_0; y[2] = xor_16;
y[3] = xor_15; y[4] = xor_21; y[5] = xor_20;

```

Inverse lookup table ($\text{LUT}_{S^{-1}}[0 \dots 63]$):

```

0x07, 0x02, 0x00, 0x01, 0x05, 0x06, 0x03, 0x04, 0x1d, 0x1e, 0x09, 0x2e, 0x3b, 0x3c, 0x2f, 0x37
0x38, 0x33, 0x3f, 0x35, 0x14, 0x1a, 0x16, 0x19, 0x2b, 0x10, 0x12, 0x08, 0x31, 0x0d, 0x0c, 0x2a
0x26, 0x2c, 0x1b, 0x17, 0x22, 0x29, 0x18, 0x15, 0x11, 0x0f, 0x24, 0x1c, 0x0e, 0x13, 0x3d, 0x0b
0x34, 0x39, 0x2d, 0x23, 0x28, 0x27, 0x32, 0x3e, 0x0a, 0x25, 0x36, 0x3a, 0x1f, 0x30, 0x21, 0x20

```

Inverse netlist (22 AND, 49 XOR, 1 NOT, $D = 3$):

```

not_0 = ~x[1]; xor_0 = x[3] ^ x[5]; xor_1 = not_0 ^ x[0]; xor_2 = x[3] ^ x[4];
xor_3 = x[2] ^ xor_1; xor_4 = x[4] ^ xor_0; xor_5 = x[0] ^ x[2];
xor_6 = not_0 ^ x[2]; xor_7 = x[4] ^ x[5]; xor_8 = xor_1 ^ xor_2;
xor_9 = x[2] ^ x[5]; xor_10 = x[0] ^ x[3]; xor_11 = xor_3 ^ xor_4;
xor_12 = not_0 ^ x[4]; xor_13 = xor_0 ^ xor_5; and_0 = xor_1 & xor_2;
and_1 = xor_0 & xor_5; and_2 = x[2] & x[5]; and_3 = xor_6 & xor_7;
and_4 = xor_3 & xor_4; and_5 = x[0] & x[3]; xor_14 = and_0 ^ and_5;
xor_15 = and_1 ^ and_2; xor_16 = xor_14 ^ xor_15; xor_17 = and_3 ^ xor_14;
xor_18 = and_0 ^ and_1; xor_19 = and_4 ^ xor_18; xor_20 = xor_16 ^ xor_8;
xor_21 = xor_17 ^ xor_9; xor_22 = xor_10 ^ xor_19; xor_23 = xor_21 ^ xor_22;
and_6 = xor_20 & xor_8; and_7 = xor_23 & xor_9; and_8 = xor_10 & xor_21;
xor_24 = xor_20 ^ xor_23; and_9 = xor_11 & xor_24; xor_25 = xor_20 ^ xor_21;
and_10 = xor_12 & xor_25; and_11 = xor_13 & xor_22; and_12 = xor_2 & xor_20;
and_13 = x[5] & xor_23; and_14 = x[3] & xor_21; and_15 = xor_24 & xor_4;
and_16 = x[4] & xor_25; and_17 = xor_0 & xor_22; xor_26 = and_6 ^ and_8;
xor_27 = and_12 ^ and_14; xor_28 = and_10 ^ and_9; xor_29 = xor_26 ^ xor_28;
xor_30 = and_11 ^ and_7; xor_31 = xor_26 ^ xor_30; xor_32 = and_7 ^ and_8;
xor_33 = and_9 ^ xor_32; xor_34 = and_15 ^ and_16; xor_35 = xor_27 ^ xor_34;
xor_36 = and_13 ^ and_17; xor_37 = xor_27 ^ xor_36; xor_38 = and_13 ^ and_14;
xor_39 = and_15 ^ xor_38; and_18 = x[0] & x[1]; and_19 = and_18 & x[3];
and_20 = and_19 & x[2]; and_21 = and_2 & and_19; xor_40 = and_19 ^ and_21;
xor_41 = and_20 ^ and_21; xor_42 = xor_40 ^ and_20; xor_43 = and_20 ^ xor_29;
xor_44 = xor_42 ^ xor_31; xor_45 = xor_40 ^ xor_33; xor_46 = and_20 ^ xor_35;
xor_47 = xor_41 ^ xor_37; xor_48 = xor_42 ^ xor_39; y[0] = xor_43;
y[1] = xor_44; y[2] = xor_45; y[3] = xor_46; y[4] = xor_47; y[5] = xor_48;

```

C.14 YSH_{6,5}: 6-bit degree 5, $\delta = 6$, $L = 24$ (6-bit, Category B)

- **Profile:** $\delta = 6$, $L = 24$ (NL = 20), $\deg(S) = 5$, $\mathcal{D}(S) = [2^{31}, 5^{32}]$, FP = 1, SAC deviation 0.125.
- **Cost:** 12 AND, 31 XOR, 0 NOT, AND depth $D = 4$ (unmasked NanGate 45 nm: $62.24 \mu\text{m}^2 / 78.0 \text{ GE}$, delay 453.9 ps, logic depth 10; pipelined $d = 2$ DOM-indep: 98 DFF_X1 [48 gadget + 50 sync], $669.79 \mu\text{m}^2 / 839.3 \text{ GE}$, stage delay 368.8 ps).
- **Remark:** Lower δ than the SPEEDY S-box (6 vs. 8) at the same $L = 24$, with eight fewer AND and 20 fewer XOR gates than its circuit 20/51/3 in [BDM26], at depth 4 instead of 3. Structure and origin: the nine level-1 products are exactly those of the Q_{2258} library seed circuit (9/30, $D = 1$; the un-sheared core is affine-equivalent to x^5), followed by three chained Stage-2 shear gates at levels 2, 3, and 4 (Section 6.5).
- **Inverse** (YSH_{6,5}⁻¹): $\deg(S^{-1}) = 5$, $\mathcal{D}(S^{-1}) = [3^{15}, 4^{16}, 5^{32}]$; cost: 21 AND, 57 XOR, 0 NOT, AND depth $D = 3$ (Tier 2: Parallel $D = 3$ directional-derivative inversion on 18-AND tower core; unmasked NanGate 45 nm: $113.32 \mu\text{m}^2 / 142.0 \text{ GE}$, delay 869.9 ps, logic depth 15; pipelined $d = 2$ DOM-indep: 134 DFF_X1 [84 gadget + 50 sync], $1011.33 \mu\text{m}^2 / 1267.3 \text{ GE}$, stage delay 381.2 ps).

Lookup table (LUT[0...63]):

```
0x00, 0x0d, 0x3e, 0x04, 0x09, 0x37, 0x3a, 0x33, 0x2c, 0x21, 0x2e, 0x14, 0x39, 0x07, 0x36, 0x1f
0x03, 0x32, 0x3d, 0x1b, 0x2b, 0x29, 0x18, 0x2d, 0x24, 0x15, 0x26, 0x20, 0x10, 0x12, 0x3f, 0x2a
0x08, 0x19, 0x17, 0x31, 0x01, 0x23, 0x13, 0x06, 0x0b, 0x1a, 0x28, 0x0e, 0x1e, 0x3c, 0x30, 0x25
0x27, 0x0a, 0x38, 0x22, 0x0f, 0x11, 0x1d, 0x34, 0x2f, 0x02, 0x0c, 0x16, 0x3b, 0x05, 0x35, 0x1c
```

Netlist (12 AND, 31 XOR, 0 NOT, $D = 4$):

```
xor_0 = x[2] ^ x[5]; xor_1 = x[0] ^ x[3]; xor_2 = x[1] ^ x[4];
and_0 = x[0] & x[1]; and_1 = x[0] & x[2]; and_2 = x[1] & x[2];
and_3 = x[3] & x[4]; and_4 = x[3] & x[5]; and_5 = x[4] & x[5];
and_6 = xor_1 & xor_2; and_7 = xor_1 & xor_0; and_8 = xor_2 & xor_0;
xor_3 = x[1] ^ and_1; xor_4 = and_3 ^ and_4; xor_5 = and_6 ^ and_7;
xor_6 = x[3] ^ and_2; xor_7 = x[0] ^ and_5; xor_8 = and_0 ^ xor_3;
xor_9 = xor_5 ^ xor_6; xor_10 = x[4] ^ xor_4; xor_11 = xor_7 ^ xor_9;
xor_12 = x[1] ^ xor_4; xor_13 = and_0 ^ and_1; xor_14 = and_8 ^ xor_13;
xor_15 = xor_5 ^ xor_12; xor_16 = xor_8 ^ xor_0; xor_17 = xor_8 ^ xor_10;
xor_18 = xor_11 ^ xor_16; xor_19 = x[5] ^ xor_9; xor_20 = xor_14 ^ xor_19;
xor_21 = x[2] ^ xor_7; xor_22 = xor_10 ^ xor_14; xor_23 = xor_21 ^ xor_22;
xor_24 = and_3 ^ xor_3; xor_25 = xor_11 ^ xor_24; xor_26 = and_6 ^ and_8;
xor_27 = xor_3 ^ xor_4; xor_28 = xor_6 ^ xor_26; xor_29 = xor_27 ^ xor_28;
and_9 = xor_20 & xor_17; and_10 = xor_15 & and_9; and_11 = and_10 & xor_18;
xor_30 = xor_29 ^ and_11; y[0] = xor_23; y[1] = xor_17; y[2] = xor_25;
y[3] = xor_18; y[4] = xor_15; y[5] = xor_30;
```

Inverse lookup table (LUT_{S⁻¹}[0...63]):

```
0x00, 0x24, 0x39, 0x10, 0x03, 0x3d, 0x27, 0x0d, 0x20, 0x04, 0x31, 0x28, 0x3a, 0x01, 0x2b, 0x34
0x1c, 0x35, 0x1d, 0x26, 0x0b, 0x19, 0x3b, 0x22, 0x16, 0x21, 0x29, 0x13, 0x3f, 0x36, 0x2c, 0x0f
0x1b, 0x09, 0x33, 0x25, 0x18, 0x2f, 0x1a, 0x30, 0x2a, 0x15, 0x1f, 0x14, 0x08, 0x17, 0x0a, 0x38
0x2e, 0x23, 0x11, 0x07, 0x37, 0x3e, 0x0e, 0x05, 0x32, 0x0c, 0x06, 0x3c, 0x2d, 0x12, 0x02, 0x1e
```

Inverse netlist (21 AND, 57 XOR, 0 NOT, $D = 3$):

```
xor_0 = x[1] ^ x[4]; xor_1 = x[0] ^ x[3]; xor_2 = x[2] ^ x[5];
xor_3 = x[2] ^ x[3]; xor_4 = x[0] ^ xor_0; xor_5 = x[3] ^ x[4];
xor_6 = x[2] ^ xor_0; xor_7 = x[2] ^ x[4]; xor_8 = xor_2 ^ xor_4;
xor_9 = x[4] ^ xor_8; xor_10 = x[1] ^ xor_3; xor_11 = x[4] ^ xor_2;
xor_12 = xor_3 ^ xor_8; xor_13 = x[4] ^ x[5]; xor_14 = x[1] ^ x[5];
xor_15 = x[1] ^ xor_1; xor_16 = x[4] ^ xor_1; xor_17 = x[3] ^ xor_0;
and_0 = xor_3 & xor_8; and_1 = x[1] & x[4]; and_2 = xor_2 & xor_7;
and_3 = xor_9 & xor_10; and_4 = xor_4 & xor_5; and_5 = xor_11 & xor_6;
and_6 = x[0] & x[1]; xor_18 = x[5] ^ and_0; xor_19 = and_2 ^ xor_1;
xor_20 = and_3 ^ and_5; xor_21 = and_1 ^ xor_0; xor_22 = and_4 ^ xor_18;
xor_23 = x[1] ^ and_1; xor_24 = and_3 ^ xor_19; xor_25 = xor_20 ^ xor_21;
xor_26 = xor_18 ^ xor_19; xor_27 = xor_23 ^ xor_24; xor_28 = xor_22 ^ xor_23;
xor_29 = and_4 ^ xor_25; xor_30 = x[4] ^ xor_18; xor_31 = xor_22 ^ xor_24;
xor_32 = and_5 ^ xor_26; xor_33 = xor_20 ^ xor_30; xor_34 = xor_21 ^ xor_32;
xor_35 = and_0 ^ and_2; xor_36 = and_6 ^ xor_25; xor_37 = xor_35 ^ xor_36;
and_7 = xor_12 & xor_34; and_8 = xor_13 & xor_29; and_9 = xor_14 & xor_33;
and_10 = xor_15 & xor_31; and_11 = xor_16 & xor_27; and_12 = xor_0 & xor_28;
and_13 = xor_34 & xor_3; and_14 = xor_29 & xor_7; and_15 = xor_6 & xor_33;
and_16 = xor_31 & xor_5; and_17 = xor_17 & xor_27; and_18 = x[1] & xor_28;
and_19 = and_6 & xor_37; and_20 = and_19 & x[2]; xor_38 = and_7 ^ and_19;
xor_39 = and_11 ^ xor_38; xor_40 = and_10 ^ and_20; xor_41 = and_8 ^ xor_39;
xor_42 = and_12 ^ xor_40; xor_43 = and_13 ^ and_16; xor_44 = and_9 ^ xor_39;
xor_45 = xor_40 ^ xor_44; xor_46 = and_13 ^ and_14; xor_47 = and_17 ^ xor_41;
xor_48 = xor_46 ^ xor_47; xor_49 = and_18 ^ xor_38; xor_50 = xor_43 ^ xor_49;
xor_51 = xor_50 ^ xor_42; xor_52 = and_15 ^ and_17; xor_53 = xor_43 ^ xor_52;
xor_54 = xor_53 ^ xor_45; xor_55 = xor_41 ^ and_20; xor_56 = and_7 ^ xor_42;
y[0] = xor_48; y[1] = xor_51; y[2] = xor_54; y[3] = xor_55; y[4] = xor_56;
y[5] = xor_45;
```

C.15 YSH_{6,6}: 6-bit quartic, $\delta = 8$, $D = 2$ (6-bit, Category B)

- **Profile:** $\delta = 8$, $L = 24$ (NL = 20), $\deg(S) = 4$, $\mathcal{D}(S) = [2^{31}, 4^{32}]$, FP = 2, SAC deviation 0.5.
- **Cost:** 10 AND, 22 XOR, 0 NOT, AND depth $D = 2$ (unmasked NanGate 45 nm: $45.75 \mu\text{m}^2 / 57.3 \text{ GE}$, delay 362.8 ps, logic depth 7; pipelined $d = 2$ DOM-indep: 66 DFF_X1 [40 gadget + 26 sync], $475.08 \mu\text{m}^2 / 595.3 \text{ GE}$, stage delay 317.9 ps).

- **Remark:** Degree 4 at depth $D = 2$, the minimum for degree 4. Structure and origin: the nine level-1 products are exactly those of the Q_{2258} library seed circuit (9/30, $D = 1$; the un-sheared core is affine-equivalent to x^5), followed by one Stage-2 shear gate at level 2 (Section 6.5).
- **Inverse** ($\text{YSH}_{6,6}^{-1}$): $\deg(S^{-1}) = 4$, $\mathcal{D}(S^{-1}) = [3^7, 4^{56}]$; cost: 27 AND, 50 XOR, 3 NOT, AND depth $D = 2$ (Tier 2: Parallel $D = 2$ directional-derivative inversion on 18-AND tower core; unmasked NanGate 45 nm: $110.12 \mu\text{m}^2 / 138.0 \text{ GE}$, delay 561.3 ps, logic depth 10; pipelined $d = 2$ DOM-indep: 140 DFF_X1 [108 gadget + 32 sync], $1081.56 \mu\text{m}^2 / 1355.3 \text{ GE}$, stage delay 479.7 ps).

Lookup table ($\text{LUT}[0 \dots 63]$):

```
0x00, 0x01, 0x07, 0x02, 0x03, 0x04, 0x05, 0x06, 0x09, 0x08, 0x2e, 0x2b, 0x3b, 0x3c, 0x14, 0x17
0x3e, 0x16, 0x39, 0x15, 0x35, 0x1b, 0x33, 0x19, 0x1a, 0x32, 0x34, 0x18, 0x29, 0x0e, 0x0f, 0x2c
0x13, 0x2a, 0x1c, 0x21, 0x10, 0x2f, 0x1e, 0x25, 0x24, 0x1d, 0x0b, 0x3f, 0x1f, 0x20, 0x38, 0x0a
0x2d, 0x3d, 0x22, 0x36, 0x26, 0x30, 0x28, 0x3a, 0x37, 0x27, 0x11, 0x0c, 0x0d, 0x12, 0x23, 0x31
```

Netlist (10 AND, 22 XOR, 0 NOT, $D = 2$):

```
xor_0 = x[2] ^ x[5]; xor_1 = x[0] ^ x[3]; xor_2 = x[1] ^ x[4];
xor_3 = xor_0 ^ xor_1; xor_4 = xor_2 ^ xor_3; and_0 = x[0] & x[1];
and_1 = x[0] & x[2]; and_2 = x[1] & x[2]; and_3 = x[3] & x[4];
and_4 = x[3] & x[5]; and_5 = x[4] & x[5]; and_6 = xor_1 & xor_2;
and_7 = xor_1 & xor_0; and_8 = xor_2 & xor_0; xor_5 = x[4] ^ and_1;
xor_6 = and_7 ^ xor_5; xor_7 = x[1] ^ and_4; xor_8 = x[5] ^ xor_6;
xor_9 = and_0 ^ and_6; xor_10 = xor_5 ^ xor_7; xor_11 = xor_6 ^ xor_9;
xor_12 = xor_4 ^ and_5; xor_13 = xor_0 ^ xor_10; xor_14 = and_0 ^ and_3;
xor_15 = xor_10 ^ xor_14; xor_16 = x[3] ^ and_1; xor_17 = and_7 ^ and_8;
xor_18 = xor_16 ^ xor_17; and_9 = xor_8 & xor_11; xor_19 = and_2 ^ and_9;
xor_20 = xor_12 ^ xor_19; xor_21 = xor_18 ^ xor_19; y[0] = xor_20;
y[1] = xor_13; y[2] = xor_15; y[3] = xor_21; y[4] = xor_8; y[5] = xor_11;
```

Inverse lookup table ($\text{LUT}_{S^{-1}}[0 \dots 63]$):

```
0x00, 0x01, 0x03, 0x04, 0x05, 0x06, 0x07, 0x02, 0x09, 0x08, 0x2f, 0x2a, 0x3b, 0x3c, 0x1d, 0x1e
0x24, 0x3a, 0x3d, 0x20, 0x0e, 0x13, 0x11, 0x0f, 0x1b, 0x17, 0x18, 0x15, 0x22, 0x29, 0x26, 0x2c
0x2d, 0x23, 0x32, 0x3e, 0x28, 0x27, 0x34, 0x39, 0x36, 0x1c, 0x21, 0x0b, 0x1f, 0x30, 0x0a, 0x25
0x35, 0x3f, 0x19, 0x16, 0x1a, 0x14, 0x33, 0x38, 0x2e, 0x12, 0x37, 0x0c, 0x0d, 0x31, 0x10, 0x2b
```

Inverse netlist (27 AND, 50 XOR, 3 NOT, $D = 2$):

```
xor_0 = x[3] ^ x[4]; xor_1 = x[2] ^ x[5]; xor_2 = x[0] ^ xor_0;
xor_3 = x[0] ^ x[1]; xor_4 = x[1] ^ x[4]; xor_5 = x[0] ^ x[2];
xor_6 = x[5] ^ xor_0; xor_7 = x[1] ^ x[2]; xor_8 = x[4] ^ x[5];
xor_9 = xor_1 ^ xor_4; xor_10 = xor_1 ^ xor_2; xor_11 = x[3] ^ xor_1;
xor_12 = xor_11 ^ xor_3; xor_13 = x[3] ^ x[5]; and_0 = x[0] & xor_0;
and_1 = x[1] & x[4]; and_2 = x[2] & x[5]; and_3 = x[3] & xor_3;
and_4 = xor_5 & xor_6; and_5 = xor_7 & xor_8; xor_14 = and_0 ^ and_5;
xor_15 = and_1 ^ and_2; xor_16 = xor_14 ^ xor_15; xor_17 = and_3 ^ xor_14;
xor_18 = and_0 ^ and_1; xor_19 = and_4 ^ xor_18; xor_20 = xor_16 ^ xor_2;
xor_21 = xor_1 ^ xor_17; xor_22 = xor_19 ^ xor_9; xor_23 = xor_21 ^ xor_22;
and_6 = xor_2 & xor_20; and_7 = xor_1 & xor_23; and_8 = xor_21 & xor_9;
xor_24 = xor_20 ^ xor_23; and_9 = xor_10 & xor_24; xor_25 = xor_20 ^ xor_21;
and_10 = xor_12 & xor_25; and_11 = xor_22 & xor_4; and_12 = xor_0 & xor_20;
and_13 = x[5] & xor_23; and_14 = xor_21 & xor_8; and_15 = xor_24 & xor_6;
and_16 = xor_13 & xor_25; and_17 = x[4] & xor_22; xor_26 = and_11 ^ and_6;
xor_27 = and_12 ^ and_17; xor_28 = and_7 ^ and_8; xor_29 = xor_26 ^ xor_28;
xor_30 = and_9 ^ xor_26; xor_31 = and_6 ^ and_7; xor_32 = and_10 ^ xor_31;
xor_33 = and_13 ^ and_14; xor_34 = xor_27 ^ xor_33; xor_35 = and_15 ^ xor_27;
xor_36 = and_12 ^ and_13; xor_37 = and_16 ^ xor_36; and_18 = x[4] & x[5];
xor_38 = x[0] ^ x[3]; and_19 = x[1] & x[2]; and_20 = xor_38 & x[1];
and_21 = xor_38 & x[2]; not_0 = ~and_19; and_22 = not_0 & and_18;
xor_39 = xor_38 ^ x[1]; and_23 = xor_39 & and_18; xor_40 = and_23 ^ and_18;
xor_41 = xor_39 ^ and_20; not_1 = ~xor_41; and_24 = not_1 & and_18;
xor_42 = xor_38 ^ x[2]; xor_43 = xor_42 ^ and_21; not_2 = ~xor_43;
and_25 = not_2 & and_18; and_26 = and_18 & x[2]; xor_44 = and_22 ^ xor_29;
xor_45 = and_24 ^ xor_30; xor_46 = and_25 ^ xor_32; xor_47 = and_26 ^ xor_34;
xor_48 = and_23 ^ xor_35; xor_49 = xor_40 ^ xor_37; y[0] = xor_44;
y[1] = xor_45; y[2] = xor_46; y[3] = xor_47; y[4] = xor_48; y[5] = xor_49;
```

C.16 6-bit Q_{2258} class (x^5) (6-bit, Category A)

- **Profile:** $\delta = 4$, $L = 16$ ($NL = 24$), $\deg(S) = 2$, $\mathcal{D}(S) = [2^{63}]$, $FP = 2$, SAC deviation 0.5.
- **Cost:** 8 AND, 34 XOR, 0 NOT, AND depth $D = 1$ (unmasked NanGate 45 nm: $62.78 \mu\text{m}^2$ / 78.7 GE, delay 527.6 ps, logic depth 9; pipelined $d = 2$ DOM-indep: 44 DFF_X1 [32 gadget + 12 sync], $392.62 \mu\text{m}^2$ / 492.0 GE, stage delay 354.6 ps).
- **Remark:** Certified linearly equivalent to x^5 ($\sim_{\text{aff}} x^5$, with $S_{\text{ref}}(u) = u^5 \bmod (x^6 + x + 1)$, $A = [0x01, 0x0e, 0x17, 0x3a, 0x03, 0x12]$, $a = 0x00$, $B = [0x01, 0x11, 0x25, 0x36, 0x31, 0x1e]$, $b = 0x00$); 8/34/1 in [BDM26] and 8/25/1 in [BDM25]: identical 8 AND and 34 XOR count as [BDM26] (nine more XOR gates than [BDM25]) at depth $D = 1$.
- **Inverse (S^{-1}):** $\deg(S^{-1}) = 3$, $\mathcal{D}(S^{-1}) = [3^{63}]$; cost: 18 AND, 37 XOR, 0 NOT, AND depth $D = 2$ (Tier 3: Tower-field norm inversion $y^{38} = y^2(y^9)^4$ over $\mathbb{F}_{2^6}/\mathbb{F}_{2^3}$; unmasked NanGate 45 nm: $78.20 \mu\text{m}^2$ / 98.0 GE, delay 473.2 ps, logic depth 9; pipelined $d = 2$ DOM-indep: 96 DFF_X1 [72 gadget + 24 sync], $743.74 \mu\text{m}^2$ / 932.0 GE, stage delay 480.1 ps).

Lookup table ($\text{LUT}[0 \dots 63]$):

```
0x00, 0x01, 0x07, 0x02, 0x03, 0x04, 0x05, 0x06, 0x09, 0x08, 0x2e, 0x2b, 0x3a, 0x3d, 0x1c, 0x1f
0x3f, 0x1e, 0x38, 0x1d, 0x34, 0x13, 0x32, 0x11, 0x12, 0x33, 0x35, 0x10, 0x29, 0x0e, 0x0f, 0x2c
0x1b, 0x2a, 0x14, 0x21, 0x18, 0x2f, 0x16, 0x25, 0x24, 0x15, 0x0b, 0x3e, 0x17, 0x20, 0x39, 0x0a
0x2d, 0x3c, 0x22, 0x37, 0x26, 0x31, 0x28, 0x3b, 0x36, 0x27, 0x19, 0x0c, 0x0d, 0x1a, 0x23, 0x30
```

Netlist (8 AND, 34 XOR, 0 NOT, $D = 1$):

```
xor_0 = x[3] ^ x[4]; xor_1 = x[0] ^ xor_0; xor_2 = x[1] ^ x[2];
xor_3 = x[2] ^ x[3]; xor_4 = x[4] ^ x[5]; xor_5 = x[0] ^ x[5];
xor_6 = x[1] ^ xor_1; xor_7 = x[5] ^ xor_1; xor_8 = x[5] ^ xor_3;
xor_9 = xor_4 ^ xor_6; xor_10 = xor_2 ^ xor_4; xor_11 = x[3] ^ xor_6;
xor_12 = xor_0 ^ xor_2; xor_13 = x[2] ^ xor_7; xor_14 = x[0] ^ x[2];
xor_15 = x[4] ^ xor_2; and_0 = x[3] & xor_2; and_1 = xor_8 & xor_1;
and_2 = xor_3 & xor_6; and_3 = xor_9 & xor_5; and_4 = xor_10 & x[0];
and_5 = xor_11 & xor_12; and_6 = xor_4 & xor_7; and_7 = xor_13 & xor_0;
xor_16 = and_5 ^ and_7; xor_17 = and_4 ^ and_6; xor_18 = x[1] ^ xor_17;
xor_19 = and_2 ^ xor_16; xor_20 = xor_14 ^ and_0; xor_21 = xor_16 ^ xor_17;
xor_22 = xor_20 ^ xor_21; xor_23 = xor_15 ^ and_1; xor_24 = and_6 ^ and_7;
xor_25 = xor_23 ^ xor_24; xor_26 = x[5] ^ xor_18; xor_27 = xor_6 ^ and_3;
xor_28 = xor_19 ^ xor_27; xor_29 = x[3] ^ and_1; xor_30 = xor_18 ^ xor_19;
xor_31 = xor_29 ^ xor_30; xor_32 = x[5] ^ and_0; xor_33 = and_6 ^ xor_32;
y[0] = xor_22; y[1] = xor_25; y[2] = xor_26; y[3] = xor_28; y[4] = xor_31;
y[5] = xor_33;
```

Inverse lookup table ($\text{LUT}_{S^{-1}}[0 \dots 63]$):

```
0x00, 0x01, 0x03, 0x04, 0x05, 0x06, 0x07, 0x02, 0x09, 0x08, 0x2f, 0x2a, 0x3b, 0x3c, 0x1d, 0x1e
0x1b, 0x17, 0x18, 0x15, 0x22, 0x29, 0x26, 0x2c, 0x24, 0x3a, 0x3d, 0x20, 0x0e, 0x13, 0x11, 0x0f
0x2d, 0x23, 0x32, 0x3e, 0x28, 0x27, 0x34, 0x39, 0x36, 0x1c, 0x21, 0x0b, 0x1f, 0x30, 0x0a, 0x25
0x3f, 0x35, 0x16, 0x19, 0x14, 0x1a, 0x38, 0x33, 0x12, 0x2e, 0x0c, 0x37, 0x31, 0x0d, 0x2b, 0x10
```

Inverse netlist (18 AND, 37 XOR, 0 NOT, $D = 2$):

```
xor_0 = x[2] ^ x[5]; xor_1 = x[0] ^ x[3]; xor_2 = x[1] ^ x[4];
xor_3 = xor_0 ^ xor_1; xor_4 = x[3] ^ x[4]; xor_5 = x[0] ^ x[1];
xor_6 = x[0] ^ x[2]; xor_7 = x[3] ^ x[5]; xor_8 = x[1] ^ x[2];
xor_9 = x[4] ^ x[5]; xor_10 = xor_0 ^ xor_2; xor_11 = xor_2 ^ xor_3;
xor_12 = xor_4 ^ x[5]; and_0 = x[0] & x[3]; and_1 = x[1] & x[4];
and_2 = x[2] & x[5]; and_3 = xor_4 & xor_5; and_4 = xor_6 & xor_7;
and_5 = xor_8 & xor_9; xor_13 = and_0 ^ and_5; xor_14 = and_1 ^ and_2;
xor_15 = xor_13 ^ xor_14; xor_16 = and_3 ^ xor_13; xor_17 = and_0 ^ and_1;
xor_18 = and_4 ^ xor_17; xor_19 = xor_1 ^ xor_15; xor_20 = xor_0 ^ xor_16;
xor_21 = xor_10 ^ xor_18; xor_22 = xor_20 ^ xor_21; and_6 = xor_1 & xor_19;
and_7 = xor_0 & xor_22; and_8 = xor_10 & xor_20; xor_23 = xor_19 ^ xor_22;
and_9 = xor_3 & xor_23; xor_24 = xor_19 ^ xor_20; and_10 = xor_11 & xor_24;
and_11 = xor_2 & xor_21; and_12 = xor_19 & x[3]; and_13 = xor_22 & x[5];
and_14 = xor_9 & xor_20; and_15 = xor_7 & xor_23; and_16 = xor_12 & xor_24;
and_17 = xor_21 & x[4]; xor_25 = and_11 ^ and_6; xor_26 = and_12 ^ and_17;
```

```
xor_27 = and_7 ^ and_8; xor_28 = xor_25 ^ xor_27; xor_29 = and_9 ^ xor_25;
xor_30 = and_6 ^ and_7; xor_31 = and_10 ^ xor_30; xor_32 = and_13 ^ and_14;
xor_33 = xor_26 ^ xor_32; xor_34 = and_15 ^ xor_26; xor_35 = and_12 ^ and_13;
xor_36 = and_16 ^ xor_35; y[0] = xor_28; y[1] = xor_29; y[2] = xor_31;
y[3] = xor_33; y[4] = xor_34; y[5] = xor_36;
```

C.17 6-bit cubic, x^{13} invariants (6-bit, Category A)

- **Profile:** $\delta = 4$, $L = 16$ (NL = 24), $\deg(S) = 3$, $\mathcal{D}(S) = [3^{63}]$, FP = 8, SAC deviation 0.5.
- **Cost:** 9 AND, 15 XOR, 9 NOT, AND depth $D = 3$ (unmasked NanGate 45 nm: $38.30 \mu\text{m}^2$ / 48.0 GE, delay 336.8 ps, logic depth 10; pipelined $d = 2$ DOM-indep: 72 DFF_X1 [36 gadget + 36 sync], $474.01 \mu\text{m}^2$ / 594.0 GE, stage delay 254.4 ps).
- **Remark:** All three forward invariants equal those of x^{13} (the class of Q_{2258}^{-1}), but this S-box is a 3-round χ_3 -Feistel cubic involution ($\mathcal{D}(S^{-1}) = [3^{63}]$, $\beta = 64$, and third-derivative rank multiset $[2^{273}, 3^{840}, 4^{840}]$ vs. $\mathcal{D}(S^{-1}) = [2^{63}]$, $\beta = 4$, and $[3^{189}, 4^{1764}]$ for x^{13}); hence it is neither affine- nor EA-equivalent to x^{13} ($\sim x^{13}$ only, and together with its input-half-swapped companion is one of only two permutations among all 1,763 Category A entries in the archive that are not affine equivalent to their catalogue reference), and the same circuit also computes its inverse. 11/42/2 in [BDM26]: two fewer AND gates and 27 fewer XOR gates, at one more AND level and with 9 NOT gates.
- **Inverse (S^{-1}):** $\deg(S^{-1}) = 3$, $\mathcal{D}(S^{-1}) = [3^{63}]$; cost: 9 AND, 15 XOR, 9 NOT, AND depth $D = 3$ (Tier 1: Exact involution ($S^{-1} = S$); unmasked NanGate 45 nm: $38.30 \mu\text{m}^2$ / 48.0 GE, delay 336.8 ps, logic depth 10; pipelined $d = 2$ DOM-indep: 72 DFF_X1 [36 gadget + 36 sync], $474.01 \mu\text{m}^2$ / 594.0 GE, stage delay 254.4 ps).

Lookup table (LUT[0...63]):

```
0x00, 0x18, 0x30, 0x08, 0x28, 0x20, 0x10, 0x38, 0x03, 0x3d, 0x17, 0x0b, 0x35, 0x1f, 0x29, 0x21
0x06, 0x2b, 0x3b, 0x37, 0x27, 0x1a, 0x16, 0x0a, 0x01, 0x19, 0x15, 0x2e, 0x3a, 0x36, 0x22, 0x0d
0x05, 0x0f, 0x1e, 0x34, 0x3e, 0x25, 0x2f, 0x14, 0x04, 0x0e, 0x39, 0x11, 0x2c, 0x33, 0x1b, 0x26
0x02, 0x3c, 0x32, 0x2d, 0x23, 0x0c, 0x1d, 0x13, 0x07, 0x2a, 0x1c, 0x12, 0x31, 0x09, 0x24, 0x3f
```

Netlist (9 AND, 15 XOR, 9 NOT, $D = 3$):

```
not_0 = ~x[3]; not_1 = ~x[4]; not_2 = ~x[5]; and_0 = not_0 & x[4];
and_1 = not_1 & x[5]; and_2 = not_2 & x[3]; xor_0 = x[0] ^ and_1;
xor_1 = x[1] ^ and_2; xor_2 = x[2] ^ and_0; xor_3 = x[3] ^ xor_0;
xor_4 = x[4] ^ xor_1; xor_5 = x[5] ^ xor_2; not_3 = ~xor_3; not_4 = ~xor_4;
not_5 = ~xor_5; and_3 = not_3 & xor_4; and_4 = not_4 & xor_5;
and_5 = not_5 & xor_3; xor_6 = and_3 ^ xor_2; xor_7 = and_4 ^ xor_0;
xor_8 = and_5 ^ xor_1; xor_9 = x[3] ^ and_4; xor_10 = x[4] ^ and_5;
xor_11 = x[5] ^ and_3; not_6 = ~xor_7; not_7 = ~xor_6; not_8 = ~xor_8;
and_6 = not_6 & xor_8; and_7 = not_7 & xor_7; and_8 = not_8 & xor_6;
xor_12 = xor_9 ^ and_8; xor_13 = xor_10 ^ and_7; xor_14 = xor_11 ^ and_6;
y[0] = xor_12; y[1] = xor_13; y[2] = xor_14; y[3] = xor_7; y[4] = xor_8;
y[5] = xor_6;
```

Inverse lookup table (LUT $_{S^{-1}}$ [0...63]):

```
0x00, 0x18, 0x30, 0x08, 0x28, 0x20, 0x10, 0x38, 0x03, 0x3d, 0x17, 0x0b, 0x35, 0x1f, 0x29, 0x21
0x06, 0x2b, 0x3b, 0x37, 0x27, 0x1a, 0x16, 0x0a, 0x01, 0x19, 0x15, 0x2e, 0x3a, 0x36, 0x22, 0x0d
0x05, 0x0f, 0x1e, 0x34, 0x3e, 0x25, 0x2f, 0x14, 0x04, 0x0e, 0x39, 0x11, 0x2c, 0x33, 0x1b, 0x26
0x02, 0x3c, 0x32, 0x2d, 0x23, 0x0c, 0x1d, 0x13, 0x07, 0x2a, 0x1c, 0x12, 0x31, 0x09, 0x24, 0x3f
```

Inverse netlist (9 AND, 15 XOR, 9 NOT, $D = 3$):

```
not_0 = ~x[3]; not_1 = ~x[4]; not_2 = ~x[5]; and_0 = not_0 & x[4];
and_1 = not_1 & x[5]; and_2 = not_2 & x[3]; xor_0 = x[0] ^ and_1;
xor_1 = x[1] ^ and_2; xor_2 = x[2] ^ and_0; xor_3 = x[3] ^ xor_0;
xor_4 = x[4] ^ xor_1; xor_5 = x[5] ^ xor_2; not_3 = ~xor_3; not_4 = ~xor_4;
not_5 = ~xor_5; and_3 = not_3 & xor_4; and_4 = not_4 & xor_5;
and_5 = not_5 & xor_3; xor_6 = and_3 ^ xor_2; xor_7 = and_4 ^ xor_0;
xor_8 = and_5 ^ xor_1; xor_9 = x[3] ^ and_4; xor_10 = x[4] ^ and_5;
```

```

xor_11 = x[5] ^ and_3; not_6 = ~xor_7; not_7 = ~xor_6; not_8 = ~xor_8;
and_6 = not_6 & xor_8; and_7 = not_7 & xor_7; and_8 = not_8 & xor_6;
xor_12 = xor_9 ^ and_8; xor_13 = xor_10 ^ and_7; xor_14 = xor_11 ^ and_6;
y[0] = xor_12; y[1] = xor_13; y[2] = xor_14; y[3] = xor_7; y[4] = xor_8;
y[5] = xor_6;

```

C.18 6-bit inversion class (6-bit, Category A)

- **Profile:** $\delta = 4$, $L = 16$ (NL = 24), $\deg(S) = 5$, $\mathcal{D}(S) = [5^{63}]$, FP = 1, SAC deviation 0.125.
- **Cost:** 21 AND, 48 XOR, 0 NOT, AND depth $D = 3$ (unmasked NanGate 45 nm: $98.95 \mu\text{m}^2 / 124.0$ GE, delay 797.5 ps, logic depth 15; pipelined $d = 2$ DOM-indep: 152 DFF_X1 [84 gadget + 68 sync], $1064.00 \mu\text{m}^2 / 1333.3$ GE, stage delay 367.4 ps).
- **Remark:** Certified linearly equivalent to the inversion $x^{31} \sim_{\text{aff}} x^{62}$ (with $S_{\text{ref}}(u) = u^{31} \bmod (x^6 + x + 1)$, $A = [0x01, 0x0e, 0x17, 0x2c, 0x02, 0x1c]$, $a = 0x00$, $B = [0x08, 0x3c, 0x16, 0x0a, 0x12, 0x05]$, $b = 0x00$); the tower-field circuit 21/63/3 of [BDM26] has the same AND count and depth and 15 more XOR gates.
- **Inverse** (S^{-1}): $\deg(S^{-1}) = 5$, $\mathcal{D}(S^{-1}) = [5^{63}]$; cost: 21 AND, 50 XOR, 0 NOT, AND depth $D = 3$ (Tier 1: 3-stage affine-folded core reuse; unmasked NanGate 45 nm: $102.14 \mu\text{m}^2 / 128.0$ GE, delay 771.4 ps, logic depth 14; pipelined $d = 2$ DOM-indep: 152 DFF_X1 [84 gadget + 68 sync], $1070.38 \mu\text{m}^2 / 1341.3$ GE, stage delay 317.0 ps).

Lookup table (LUT[0...63]):

```

0x00, 0x08, 0x28, 0x30, 0x38, 0x10, 0x18, 0x20, 0x36, 0x06, 0x32, 0x22, 0x29, 0x21, 0x0c, 0x2c
0x1b, 0x15, 0x03, 0x3d, 0x19, 0x3a, 0x11, 0x2a, 0x12, 0x35, 0x1d, 0x02, 0x2e, 0x2f, 0x17, 0x1e
0x24, 0x31, 0x0f, 0x25, 0x04, 0x39, 0x37, 0x0d, 0x3f, 0x3c, 0x0a, 0x13, 0x1c, 0x07, 0x0b, 0x1a
0x09, 0x3e, 0x1f, 0x3b, 0x27, 0x23, 0x01, 0x0e, 0x2d, 0x14, 0x33, 0x16, 0x26, 0x2b, 0x34, 0x05

```

Netlist (21 AND, 48 XOR, 0 NOT, $D = 3$):

```

xor_0 = x[0] ^ x[1]; xor_1 = x[3] ^ x[4]; xor_2 = x[0] ^ x[2];
xor_3 = x[4] ^ x[5]; xor_4 = x[1] ^ x[2]; xor_5 = x[3] ^ x[5];
and_0 = x[3] & x[0]; and_1 = x[4] & x[1]; and_2 = x[5] & x[2];
and_3 = xor_1 & xor_0; and_4 = xor_5 & xor_2; and_5 = xor_3 & xor_4;
xor_6 = x[4] ^ and_0; xor_7 = x[3] ^ and_5; xor_8 = x[0] ^ and_2;
xor_9 = xor_6 ^ xor_7; xor_10 = x[1] ^ and_4; xor_11 = x[5] ^ xor_8;
xor_12 = x[2] ^ and_1; xor_13 = xor_6 ^ xor_10; xor_14 = xor_12 ^ xor_13;
xor_15 = and_1 ^ xor_9; xor_16 = x[2] ^ and_3; xor_17 = xor_9 ^ xor_16;
xor_18 = xor_11 ^ xor_15; xor_19 = and_3 ^ xor_10; and_6 = xor_18 & xor_14;
and_7 = xor_17 & xor_14; and_8 = xor_18 & xor_17; xor_20 = and_7 ^ xor_11;
xor_21 = and_3 ^ xor_20; xor_22 = and_6 ^ xor_7; xor_23 = and_1 ^ xor_22;
xor_24 = and_8 ^ xor_12; xor_25 = xor_21 ^ xor_24; xor_26 = and_6 ^ xor_17;
xor_27 = and_8 ^ xor_14; xor_28 = xor_20 ^ xor_15; xor_29 = and_6 ^ xor_28;
xor_30 = xor_23 ^ xor_19; xor_31 = xor_13 ^ xor_21; xor_32 = and_8 ^ xor_26;
and_9 = x[4] & xor_27; and_10 = x[1] & xor_27; and_11 = x[5] & xor_30;
and_12 = x[2] & xor_30; and_13 = x[3] & xor_31; and_14 = x[0] & xor_31;
and_15 = xor_3 & xor_32; and_16 = xor_4 & xor_32; and_17 = xor_1 & xor_25;
and_18 = xor_0 & xor_25; and_19 = xor_5 & xor_29; and_20 = xor_2 & xor_29;
xor_33 = and_13 ^ and_15; xor_34 = and_9 ^ and_19; xor_35 = and_11 ^ xor_33;
xor_36 = and_14 ^ and_16; xor_37 = and_9 ^ xor_35; xor_38 = and_13 ^ xor_34;
xor_39 = and_17 ^ xor_33; xor_40 = and_10 ^ and_12; xor_41 = xor_36 ^ xor_37;
xor_42 = xor_40 ^ xor_41; xor_43 = and_18 ^ xor_36; xor_44 = xor_39 ^ xor_43;
xor_45 = and_10 ^ and_14; xor_46 = and_20 ^ xor_38; xor_47 = xor_45 ^ xor_46;
y[0] = xor_37; y[1] = xor_39; y[2] = xor_38; y[3] = xor_42; y[4] = xor_44;
y[5] = xor_47;

```

Inverse lookup table (LUT $_{S^{-1}}$ [0...63]):

```

0x00, 0x36, 0x1b, 0x12, 0x24, 0x3f, 0x09, 0x2d, 0x01, 0x30, 0x2a, 0x2e, 0x0e, 0x27, 0x37, 0x22
0x05, 0x16, 0x18, 0x2b, 0x39, 0x11, 0x3b, 0x1e, 0x06, 0x14, 0x2f, 0x10, 0x2c, 0x1a, 0x1f, 0x32
0x07, 0x0d, 0x0b, 0x35, 0x20, 0x23, 0x3c, 0x34, 0x02, 0x0c, 0x17, 0x3d, 0x0f, 0x38, 0x1c, 0x1d
0x03, 0x21, 0x0a, 0x3a, 0x3e, 0x19, 0x08, 0x26, 0x04, 0x25, 0x15, 0x33, 0x29, 0x13, 0x31, 0x28

```

Inverse netlist (21 AND, 50 XOR, 0 NOT, $D = 3$):

```

xor_0 = x[2] ^ x[3]; xor_1 = x[3] ^ x[4]; xor_2 = x[0] ^ x[1];
xor_3 = x[0] ^ x[2]; xor_4 = x[1] ^ x[2]; xor_5 = x[3] ^ x[5];
xor_6 = x[4] ^ x[5]; xor_7 = x[5] ^ xor_1; and_0 = xor_6 & x[0];
and_1 = xor_1 & x[1]; and_2 = xor_7 & x[2]; and_3 = xor_5 & xor_2;
and_4 = xor_3 & x[3]; and_5 = xor_4 & x[5]; xor_8 = x[4] ^ and_1;
xor_9 = x[5] ^ and_3; xor_10 = x[0] ^ and_2; xor_11 = x[1] ^ and_4;
xor_12 = and_0 ^ and_5; xor_13 = and_0 ^ xor_11; xor_14 = xor_8 ^ xor_0;
xor_15 = xor_8 ^ xor_12; xor_16 = xor_0 ^ xor_9; xor_17 = xor_12 ^ xor_16;
xor_18 = xor_13 ^ xor_14; xor_19 = and_5 ^ xor_8; xor_20 = xor_9 ^ xor_11;
xor_21 = xor_10 ^ xor_15; xor_22 = xor_19 ^ xor_20; and_6 = xor_21 & xor_18;
and_7 = xor_17 & xor_18; and_8 = xor_21 & xor_17; xor_23 = and_7 ^ xor_10;
xor_24 = xor_9 ^ xor_23; xor_25 = and_6 ^ and_8; xor_26 = and_6 ^ xor_23;
xor_27 = and_6 ^ xor_22; xor_28 = and_8 ^ xor_14; xor_29 = and_8 ^ xor_18;
xor_30 = xor_13 ^ xor_24; xor_31 = xor_15 ^ xor_26; xor_32 = xor_17 ^ xor_25;
xor_33 = xor_24 ^ xor_28; and_9 = xor_1 & xor_29; and_10 = xor_29 & x[1];
and_11 = xor_7 & xor_27; and_12 = xor_27 & x[2]; and_13 = xor_6 & xor_30;
and_14 = xor_30 & x[0]; and_15 = xor_32 & x[5]; and_16 = xor_4 & xor_32;
and_17 = xor_5 & xor_33; and_18 = xor_2 & xor_33; and_19 = xor_31 & x[3];
and_20 = xor_3 & xor_31; xor_34 = and_10 ^ and_18; xor_35 = and_9 ^ and_13;
xor_36 = and_12 ^ and_14; xor_37 = and_12 ^ xor_34; xor_38 = and_16 ^ and_20;
xor_39 = and_18 ^ and_20; xor_40 = xor_34 ^ xor_38; xor_41 = xor_36 ^ xor_39;
xor_42 = and_11 ^ and_15; xor_43 = xor_35 ^ xor_40; xor_44 = xor_42 ^ xor_43;
xor_45 = and_13 ^ and_15; xor_46 = and_17 ^ xor_37; xor_47 = xor_45 ^ xor_46;
xor_48 = and_19 ^ xor_35; xor_49 = xor_41 ^ xor_48; y[0] = xor_44;
y[1] = xor_47; y[2] = xor_49; y[3] = xor_40; y[4] = xor_37; y[5] = xor_41;

```

C.19 6-bit Dillon APN permutation (6-bit, Category A)

- **Profile:** $\delta = 2$, $L = 16$ (NL = 24), $\deg(S) = 4$, $\mathcal{D}(S) = [3^7, 4^{56}]$, FP = 1, SAC deviation 0.5.
- **Cost:** 28 AND, 68 XOR, 0 NOT, AND depth $D = 2$ (unmasked NanGate 45 nm: $138.32 \mu\text{m}^2$ / 173.3 GE, delay 538.6 ps, logic depth 10; pipelined $d = 2$ DOM-indep: 154 DFF_X1 [112 gadget + 42 sync], $1211.36 \mu\text{m}^2$ / 1518.0 GE, stage delay 460.5 ps).
- **Remark:** Equal to the APN permutation [BDMW10] of our reference library; 17/75/2 in [BDM26]: our circuit has 11 more AND gates and seven fewer XOR gates.
- **Inverse (S^{-1}):** $\deg(S^{-1}) = 4$, $\mathcal{D}(S^{-1}) = [3^7, 4^{56}]$; cost: 28 AND, 78 XOR, 7 NOT, AND depth $D = 2$ (Tier 1: 3-stage affine-folded core reuse; unmasked NanGate 45 nm: $158.00 \mu\text{m}^2$ / 198.0 GE, delay 819.8 ps, logic depth 15; pipelined $d = 2$ DOM-indep: 168 DFF_X1 [112 gadget + 56 sync], $1310.32 \mu\text{m}^2$ / 1642.0 GE, stage delay 423.5 ps).

Lookup table (LUT[0...63]):

```

0x00, 0x36, 0x30, 0x0d, 0x0f, 0x12, 0x35, 0x23, 0x19, 0x3f, 0x2d, 0x34, 0x03, 0x14, 0x29, 0x21
0x3b, 0x24, 0x02, 0x22, 0x0a, 0x08, 0x39, 0x25, 0x3c, 0x13, 0x2a, 0x0e, 0x32, 0x1a, 0x3a, 0x18
0x27, 0x1b, 0x15, 0x11, 0x10, 0x1d, 0x01, 0x3e, 0x2f, 0x28, 0x33, 0x38, 0x07, 0x2b, 0x2c, 0x26
0x1f, 0x0b, 0x04, 0x1c, 0x3d, 0x2e, 0x05, 0x31, 0x09, 0x06, 0x17, 0x20, 0x1e, 0x0c, 0x37, 0x16

```

Netlist (28 AND, 68 XOR, 0 NOT, $D = 2$):

```

xor_0 = x[0] ^ x[2]; xor_1 = x[2] ^ x[3]; xor_2 = x[5] ^ xor_0;
and_0 = x[0] & x[1]; and_1 = x[0] & x[2]; and_2 = x[1] & x[2];
and_3 = x[0] & x[3]; and_4 = x[1] & x[3]; and_5 = x[2] & x[3];
and_6 = x[0] & x[4]; and_7 = x[1] & x[4]; and_8 = x[2] & x[4];
and_9 = x[3] & x[4]; and_10 = x[0] & x[5]; and_11 = x[1] & x[5];
and_12 = x[2] & x[5]; and_13 = x[3] & x[5]; and_14 = x[4] & x[5];
xor_3 = and_3 ^ and_8; xor_4 = x[4] ^ and_5; xor_5 = and_6 ^ xor_3;
xor_6 = and_0 ^ and_1; xor_7 = x[0] ^ and_12; xor_8 = x[3] ^ and_8;
xor_9 = and_14 ^ xor_6; xor_10 = x[0] ^ and_0; xor_11 = and_7 ^ xor_1;
xor_12 = and_13 ^ xor_4; xor_13 = x[1] ^ xor_7; xor_14 = x[5] ^ and_2;
xor_15 = x[1] ^ and_2; xor_16 = x[3] ^ and_6; xor_17 = and_1 ^ and_6;
xor_18 = and_0 ^ and_8; xor_19 = x[1] ^ and_1; xor_20 = x[3] ^ and_10;
xor_21 = and_6 ^ and_11; xor_22 = x[3] ^ and_1; xor_23 = and_7 ^ and_11;
xor_24 = x[3] ^ and_11; xor_25 = and_7 ^ and_10; xor_26 = x[4] ^ and_13;
xor_27 = x[0] ^ and_11; xor_28 = x[3] ^ and_14; xor_29 = x[0] ^ and_14;
xor_30 = and_8 ^ and_11; xor_31 = xor_9 ^ xor_11; xor_32 = xor_12 ^ xor_31;
xor_33 = xor_2 ^ and_11; xor_34 = xor_9 ^ xor_33; xor_35 = xor_2 ^ and_5;
xor_36 = xor_3 ^ xor_35; xor_37 = and_12 ^ xor_6; xor_38 = xor_11 ^ xor_37;
xor_39 = xor_8 ^ xor_12; xor_40 = xor_13 ^ xor_39; and_15 = and_6 & xor_14;

```

```

and_16 = xor_15 & xor_16; and_17 = xor_1 & xor_17; and_18 = x[3] & xor_18;
and_19 = xor_19 & xor_20; and_20 = xor_0 & xor_21; and_21 = xor_22 & xor_23;
and_22 = xor_24 & xor_25; and_23 = xor_7 & xor_26; and_24 = xor_27 & xor_28;
and_25 = xor_29 & xor_30; and_26 = xor_10 & xor_8; and_27 = xor_10 & xor_4;
xor_41 = and_9 ^ and_26; xor_42 = x[4] ^ and_18; xor_43 = xor_42 ^ xor_5;
xor_44 = and_19 ^ xor_41; xor_45 = and_10 ^ xor_44; xor_46 = x[5] ^ and_20;
xor_47 = and_2 ^ and_21; xor_48 = and_4 ^ xor_45; xor_49 = and_23 ^ xor_48;
xor_50 = and_25 ^ xor_46; xor_51 = xor_43 ^ xor_47; xor_52 = xor_49 ^ xor_51;
xor_53 = and_15 ^ and_22; xor_54 = xor_50 ^ xor_53; xor_55 = xor_32 ^ xor_54;
xor_56 = xor_34 ^ xor_52; xor_57 = xor_36 ^ and_16; xor_58 = xor_41 ^ xor_57;
xor_59 = xor_38 ^ and_20; xor_60 = xor_52 ^ xor_59; xor_61 = xor_40 ^ and_16;
xor_62 = xor_44 ^ xor_61; xor_63 = xor_13 ^ and_17; xor_64 = and_24 ^ and_27;
xor_65 = xor_43 ^ xor_54; xor_66 = xor_63 ^ xor_64; xor_67 = xor_65 ^ xor_66;
y[0] = xor_55; y[1] = xor_56; y[2] = xor_58; y[3] = xor_60; y[4] = xor_62;
y[5] = xor_67;

```

Inverse lookup table ($\text{LUT}_{S^{-1}}[0 \dots 63]$):

```

0x00, 0x26, 0x12, 0x0c, 0x32, 0x36, 0x39, 0x2c, 0x15, 0x38, 0x14, 0x31, 0x3d, 0x03, 0x1b, 0x04
0x24, 0x23, 0x05, 0x19, 0x0d, 0x22, 0x3f, 0x3a, 0x1f, 0x08, 0x1d, 0x21, 0x33, 0x25, 0x3c, 0x30
0x3b, 0x0f, 0x13, 0x07, 0x11, 0x17, 0x2f, 0x20, 0x29, 0x0e, 0x1a, 0x2d, 0x2e, 0x0a, 0x35, 0x28
0x02, 0x37, 0x1c, 0x2a, 0x0b, 0x06, 0x01, 0x3e, 0x2b, 0x16, 0x1e, 0x10, 0x18, 0x34, 0x27, 0x09

```

Inverse netlist (28 AND, 78 XOR, 7 NOT, $D = 2$):

```

not_0 = ~x[0]; not_1 = ~x[2]; xor_0 = not_1 ^ x[3]; xor_1 = xor_0 ^ x[4];
xor_2 = xor_1 ^ not_0; xor_3 = x[1] ^ x[3]; xor_4 = xor_0 ^ x[1];
not_2 = ~xor_3; xor_5 = not_1 ^ x[4]; xor_6 = x[3] ^ x[5];
xor_7 = xor_2 ^ x[5]; xor_8 = xor_1 ^ x[5]; xor_9 = xor_3 ^ x[4];
and_0 = not_2 & xor_4; and_1 = xor_4 & xor_5; and_2 = not_2 & xor_5;
and_3 = xor_4 & xor_6; and_4 = not_2 & xor_6; and_5 = xor_5 & xor_6;
and_6 = xor_2 & xor_4; and_7 = not_2 & xor_2; and_8 = xor_2 & xor_5;
and_9 = xor_2 & xor_6; and_10 = xor_4 & xor_7; and_11 = not_2 & xor_7;
and_12 = xor_5 & xor_7; and_13 = xor_6 & xor_7; and_14 = xor_2 & xor_7;
xor_10 = and_0 ^ xor_4; xor_11 = and_1 ^ not_2; xor_12 = and_1 ^ xor_6;
xor_13 = and_2 ^ xor_7; xor_14 = and_2 ^ not_2; xor_15 = and_5 ^ xor_2;
xor_16 = and_6 ^ xor_6; xor_17 = and_6 ^ and_1; xor_18 = and_8 ^ and_0;
xor_19 = and_8 ^ xor_6; xor_20 = and_10 ^ xor_6; xor_21 = and_10 ^ and_7;
xor_22 = and_6 ^ and_11; xor_23 = and_11 ^ xor_4; xor_24 = and_11 ^ and_8;
xor_25 = and_7 ^ and_11; xor_26 = and_11 ^ xor_6; xor_27 = and_12 ^ xor_4;
xor_28 = and_13 ^ xor_2; xor_29 = and_14 ^ xor_6; xor_30 = and_14 ^ xor_4;
and_15 = and_6 & xor_13; and_16 = xor_14 & xor_16; and_17 = xor_8 & xor_17;
and_18 = xor_6 & xor_18; and_19 = xor_11 & xor_20; and_20 = xor_9 & xor_22;
and_21 = xor_12 & xor_25; and_22 = xor_26 & xor_21; and_23 = xor_27 & xor_28;
and_24 = xor_23 & xor_29; and_25 = xor_30 & xor_24; and_26 = xor_10 & xor_19;
and_27 = xor_10 & xor_15; xor_31 = and_16 ^ not_0; xor_32 = and_13 ^ and_19;
xor_33 = x[4] ^ x[5]; xor_34 = and_5 ^ and_12; xor_35 = and_8 ^ xor_31;
xor_36 = and_24 ^ and_17; xor_37 = and_27 ^ xor_36; xor_38 = xor_33 ^ xor_35;
xor_39 = and_3 ^ not_1; xor_40 = xor_32 ^ xor_39; xor_41 = and_9 ^ and_26;
xor_42 = xor_41 ^ xor_34; xor_43 = xor_37 ^ x[1]; xor_44 = and_14 ^ x[3];
xor_45 = and_2 ^ and_4; xor_46 = and_10 ^ and_21; xor_47 = and_15 ^ and_20;
xor_48 = and_25 ^ and_22; xor_49 = and_23 ^ xor_38; xor_50 = xor_45 ^ xor_46;
xor_51 = xor_47 ^ xor_48; xor_52 = xor_49 ^ xor_50; xor_53 = and_20 ^ not_0;
xor_54 = xor_53 ^ xor_40; xor_55 = xor_54 ^ xor_25; xor_56 = xor_44 ^ xor_55;
xor_57 = x[1] ^ x[4]; xor_58 = and_12 ^ xor_57; xor_59 = xor_58 ^ xor_40;
not_3 = ~xor_59; xor_60 = xor_32 ^ xor_38; xor_61 = xor_42 ^ xor_60;
not_4 = ~xor_61; xor_62 = and_6 ^ and_18; xor_63 = xor_31 ^ xor_62;
xor_64 = xor_42 ^ xor_63; xor_65 = xor_43 ^ xor_64; xor_66 = xor_51 ^ xor_65;
xor_67 = xor_43 ^ xor_25; xor_68 = xor_52 ^ xor_67; not_5 = ~xor_68;
xor_69 = and_0 ^ and_1; xor_70 = and_3 ^ xor_69; xor_71 = and_11 ^ xor_70;
xor_72 = and_19 ^ xor_71; xor_73 = xor_34 ^ xor_72; xor_74 = xor_73 ^ xor_37;
xor_75 = xor_44 ^ xor_74; xor_76 = xor_51 ^ xor_75; xor_77 = xor_52 ^ xor_76;
not_6 = ~xor_77; y[0] = xor_56; y[1] = not_3; y[2] = not_4; y[3] = xor_66;
y[4] = not_5; y[5] = not_6;

```

C.20 YSH_{7,1}: 7-bit quartic, $\delta = 4$, $L = 32$ (7-bit, Category B)

- **Profile:** $\delta = 4$, $L = 32$ (NL = 48), $\deg(S) = 4$, $\mathcal{D}(S) = [2^{63}, 4^{64}]$, FP = 1, SAC deviation 0.125.
- **Cost:** 15 AND, 70 XOR, 10 NOT, AND depth $D = 2$ (unmasked NanGate 45 nm: $133.00 \mu\text{m}^2 / 166.7 \text{ GE}$, delay 750.9 ps, logic depth 14; pipelined $d = 2$ DOM-indep: 88 DFF_X1 [60 gadget + 28 sync], $786.30 \mu\text{m}^2 / 985.3 \text{ GE}$, stage delay 478.9 ps).

- **Remark:** Bipartite spectrum $[2^{63}, 4^{64}]$: its quadratic components form a hyperplane. Its AND gates are 14 at level 1 and one at level 2 (Section 6.6).
- **Inverse** ($\text{YSH}_{7,1}^{-1}$): $\deg(S^{-1}) = 5$, $\mathcal{D}(S^{-1}) = [4^7, 5^{120}]$; cost: 30 AND, 158 XOR, 12 NOT, AND depth $D = 3$ (Tier 2: Output shear unwinding + 29-AND $D = 2$ Gold inverse core; unmasked NanGate 45 nm: $290.47 \mu\text{m}^2 / 364.0 \text{ GE}$, delay 1511.7 ps, logic depth 24; pipelined $d = 2$ DOM-indep: 226 DFF_X1 [120 gadget + 106 sync], $1851.89 \mu\text{m}^2 / 2320.7 \text{ GE}$, stage delay 759.6 ps).

Lookup table ($\text{LUT}[0 \dots 127]$):

```
0x3d, 0x7b, 0x7e, 0x34, 0x5b, 0x5d, 0x76, 0x7c, 0x5f, 0x2f, 0x4b, 0x37, 0x65, 0x55, 0x1f, 0x23
0x06, 0x56, 0x0c, 0x51, 0x3a, 0x2a, 0x5e, 0x42, 0x7d, 0x1b, 0x21, 0x4a, 0x1d, 0x3b, 0x2e, 0x04
0x79, 0x35, 0x32, 0x73, 0x25, 0x29, 0x00, 0x01, 0x69, 0x13, 0x75, 0x02, 0x68, 0x52, 0x1a, 0x2d
0x6d, 0x36, 0x6e, 0x38, 0x6a, 0x70, 0x07, 0x11, 0x64, 0x08, 0x31, 0x50, 0x3f, 0x12, 0x05, 0x24
0x7a, 0x19, 0x49, 0x26, 0x10, 0x33, 0x4d, 0x63, 0x2b, 0x7f, 0x4e, 0x16, 0x1c, 0x09, 0x17, 0x0f
0x45, 0x30, 0x3e, 0x46, 0x74, 0x41, 0x60, 0x58, 0x0d, 0x4f, 0x20, 0x6f, 0x61, 0x62, 0x22, 0x2c
0x77, 0x1e, 0x4c, 0x28, 0x27, 0x0e, 0x72, 0x57, 0x54, 0x0b, 0x39, 0x6b, 0x59, 0x47, 0x5a, 0x48
0x66, 0x18, 0x14, 0x67, 0x6c, 0x53, 0x71, 0x43, 0x5c, 0x15, 0x78, 0x3c, 0x0a, 0x03, 0x40, 0x44
```

Netlist (15 AND, 70 XOR, 10 NOT, $D = 2$):

```
not_0 = ~x[0]; xor_0 = x[1] ^ x[3]; xor_1 = x[2] ^ x[3]; not_1 = ~x[3];
xor_2 = not_0 ^ x[5]; xor_3 = x[4] ^ xor_0; xor_4 = x[2] ^ xor_0;
not_2 = ~xor_1; xor_5 = not_0 ^ not_1; xor_6 = not_1 ^ x[5];
xor_7 = not_1 ^ xor_2; xor_8 = x[0] ^ xor_3; xor_9 = x[6] ^ xor_3;
not_3 = ~xor_4; xor_10 = not_0 ^ not_2; xor_11 = not_2 ^ xor_2;
and_0 = not_2 & not_1; not_4 = ~xor_8; not_5 = ~xor_9; xor_12 = not_3 ^ xor_2;
xor_13 = not_3 ^ xor_5; xor_14 = x[5] ^ x[2]; and_1 = not_0 & x[2];
xor_15 = not_2 ^ and_0; xor_16 = not_3 ^ not_4; xor_17 = not_4 ^ xor_2;
xor_18 = not_1 ^ not_4; xor_19 = not_4 ^ xor_5; xor_20 = not_5 ^ xor_2;
xor_21 = not_0 ^ not_5; xor_22 = not_3 ^ not_5; xor_23 = not_5 ^ x[2];
and_2 = not_4 & not_5; and_3 = xor_10 & xor_12; and_4 = xor_13 & xor_11;
xor_24 = not_1 ^ and_1; xor_25 = not_4 ^ and_1; xor_26 = not_3 ^ xor_15;
xor_27 = not_2 ^ xor_16; xor_28 = not_5 ^ xor_16; and_5 = not_3 & xor_17;
and_6 = xor_18 & xor_2; and_7 = xor_19 & xor_6; and_8 = not_1 & xor_20;
and_9 = xor_16 & xor_21; and_10 = xor_11 & xor_22; and_11 = xor_13 & xor_23;
xor_29 = and_3 ^ xor_15; and_12 = xor_27 & xor_14; and_13 = xor_7 & xor_28;
xor_30 = and_2 ^ and_5; xor_31 = not_2 ^ and_5; xor_32 = not_0 ^ and_6;
xor_33 = and_8 & xor_7; xor_34 = not_4 ^ and_8; xor_35 = and_9 ^ and_4;
xor_36 = and_10 ^ and_11; xor_37 = and_10 ^ and_13; xor_38 = and_3 ^ and_13;
xor_39 = xor_30 ^ xor_26; xor_40 = and_6 ^ xor_30; xor_41 = and_12 ^ xor_32;
xor_42 = xor_32 ^ xor_25; xor_43 = xor_33 ^ xor_24; xor_44 = and_12 ^ xor_33;
xor_45 = not_5 ^ xor_35; xor_46 = and_4 ^ xor_36; xor_47 = xor_35 ^ xor_36;
xor_48 = xor_35 ^ xor_37; xor_49 = xor_34 ^ xor_40; xor_50 = x[5] ^ xor_41;
xor_51 = xor_31 ^ xor_41; xor_52 = xor_38 ^ xor_42; xor_53 = xor_29 ^ xor_43;
xor_54 = xor_36 ^ xor_44; xor_55 = xor_29 ^ xor_45; xor_56 = xor_43 ^ xor_46;
xor_57 = xor_43 ^ xor_48; xor_58 = xor_47 ^ xor_50; xor_59 = xor_50 ^ xor_53;
xor_60 = xor_55 ^ xor_54; xor_61 = xor_55 ^ xor_49; xor_62 = xor_51 ^ xor_56;
xor_63 = xor_39 ^ xor_58; not_6 = ~xor_59; xor_64 = xor_62 ^ xor_60;
xor_65 = xor_62 ^ xor_52; xor_66 = xor_63 ^ xor_60; not_7 = ~xor_63;
xor_67 = xor_63 ^ xor_61; not_8 = ~xor_64; xor_68 = xor_57 ^ xor_66;
not_9 = ~xor_67; and_14 = not_6 & not_8; xor_69 = xor_66 ^ and_14;
y[0] = xor_69; y[1] = not_6; y[2] = not_8; y[3] = not_7; y[4] = xor_68;
y[5] = xor_65; y[6] = not_9;
```

Inverse lookup table ($\text{LUT}_{S^{-1}}[0 \dots 127]$):

```
0x26, 0x27, 0x2b, 0x7d, 0x1f, 0x3e, 0x10, 0x36, 0x39, 0x4d, 0x7c, 0x69, 0x12, 0x58, 0x65, 0x4f
0x44, 0x37, 0x3d, 0x29, 0x72, 0x79, 0x4b, 0x4e, 0x71, 0x41, 0x2e, 0x19, 0x4c, 0x1c, 0x61, 0x0e
0x5a, 0x1a, 0x5e, 0x0f, 0x3f, 0x24, 0x43, 0x64, 0x63, 0x25, 0x15, 0x48, 0x5f, 0x2f, 0x1e, 0x09
0x51, 0x3a, 0x22, 0x45, 0x03, 0x21, 0x31, 0x0b, 0x33, 0x6a, 0x14, 0x1d, 0x7b, 0x00, 0x52, 0x3c
0x7e, 0x55, 0x17, 0x77, 0x7f, 0x50, 0x53, 0x6d, 0x6f, 0x42, 0x1b, 0x0a, 0x62, 0x46, 0x4a, 0x59
0x3b, 0x13, 0x2d, 0x75, 0x68, 0x0d, 0x11, 0x67, 0x57, 0x6c, 0x6e, 0x04, 0x78, 0x05, 0x16, 0x08
0x56, 0x5c, 0x5d, 0x47, 0x38, 0x0c, 0x70, 0x73, 0x2c, 0x28, 0x34, 0x6b, 0x74, 0x30, 0x32, 0x5b
0x35, 0x76, 0x66, 0x23, 0x54, 0x2a, 0x06, 0x60, 0x7a, 0x20, 0x40, 0x01, 0x07, 0x18, 0x02, 0x49
```

Inverse netlist (30 AND, 158 XOR, 12 NOT, $D = 3$):

```
and_0 = x[1] & x[2]; xor_0 = and_0 ^ x[0]; not_0 = ~x[1]; not_1 = ~x[2];
not_2 = ~x[3]; not_3 = ~x[5]; not_4 = ~x[6]; xor_1 = not_1 ^ x[4];
xor_2 = not_3 ^ xor_0; xor_3 = not_2 ^ not_4; xor_4 = not_0 ^ not_3;
xor_5 = not_0 ^ xor_2; xor_6 = xor_1 ^ xor_3; xor_7 = xor_0 ^ x[4];
xor_8 = not_2 ^ xor_1; xor_9 = not_4 ^ xor_5; xor_10 = xor_0 ^ xor_1;
xor_11 = not_2 ^ xor_4; xor_12 = xor_2 ^ xor_3; xor_13 = not_1 ^ xor_0;
```

```

xor_14 = xor_0 ^ xor_8; xor_15 = not_2 ^ xor_5; xor_16 = not_3 ^ not_4;
xor_17 = xor_1 ^ xor_16; xor_18 = x[4] ^ xor_11; xor_19 = xor_4 ^ xor_6;
xor_20 = x[4] ^ xor_9; xor_21 = not_3 ^ xor_6; xor_22 = not_0 ^ not_4;
xor_23 = xor_22 ^ xor_7; xor_24 = not_2 ^ xor_13; xor_25 = xor_2 ^ xor_6;
xor_26 = xor_1 ^ xor_2; xor_27 = x[4] ^ xor_3; xor_28 = xor_3 ^ xor_7;
xor_29 = not_0 ^ xor_8; xor_30 = not_1 ^ xor_5; xor_31 = xor_3 ^ xor_4;
xor_32 = not_4 ^ xor_10; xor_33 = not_1 ^ xor_12; xor_34 = not_0 ^ xor_13;
xor_35 = not_1 ^ not_2; xor_36 = xor_2 ^ xor_35; and_1 = xor_14 & xor_15;
and_2 = xor_17 & xor_18; and_3 = not_3 & xor_19; and_4 = xor_8 & xor_9;
and_5 = xor_14 & xor_20; and_6 = xor_21 & xor_23; and_7 = xor_24 & xor_25;
and_8 = xor_23 & xor_25; and_9 = xor_26 & xor_27; and_10 = x[4] & xor_7;
and_11 = xor_7 & xor_10; and_12 = xor_28 & xor_4; and_13 = xor_29 & xor_30;
and_14 = not_0 & xor_2; xor_37 = and_1 ^ xor_31; xor_38 = and_9 ^ xor_37;
xor_39 = xor_32 ^ xor_38; xor_40 = and_2 ^ xor_12; xor_41 = and_4 ^ and_5;
xor_42 = xor_17 ^ xor_39; xor_43 = and_13 ^ xor_40; xor_44 = and_8 ^ xor_11;
xor_45 = and_3 ^ and_6; xor_46 = xor_41 ^ xor_45; xor_47 = and_12 ^ xor_42;
xor_48 = and_11 ^ xor_44; xor_49 = xor_41 ^ xor_43; xor_50 = and_8 ^ xor_33;
xor_51 = xor_40 ^ xor_50; xor_52 = xor_46 ^ xor_51; xor_53 = and_2 ^ and_10;
xor_54 = and_14 ^ xor_53; xor_55 = xor_46 ^ xor_54; xor_56 = xor_47 ^ xor_55;
xor_57 = xor_48 ^ xor_56; not_5 = ~xor_57; xor_58 = and_7 ^ xor_47;
xor_59 = xor_49 ^ xor_58; xor_60 = xor_34 ^ xor_42; xor_61 = xor_43 ^ xor_60;
xor_62 = xor_48 ^ xor_61; not_6 = ~xor_62; xor_63 = and_3 ^ xor_36;
xor_64 = and_7 ^ xor_63; xor_65 = and_12 ^ xor_64; xor_66 = and_14 ^ xor_65;
xor_67 = xor_39 ^ xor_66; not_7 = ~xor_67; xor_68 = and_10 ^ xor_38;
xor_69 = xor_49 ^ xor_68; not_8 = ~xor_69; xor_70 = and_4 ^ xor_14;
xor_71 = and_6 ^ xor_70; xor_72 = and_7 ^ xor_71; xor_73 = and_9 ^ xor_72;
xor_74 = xor_44 ^ xor_73; not_9 = ~xor_74; xor_75 = not_7 ^ xor_52;
xor_76 = not_6 ^ xor_52; xor_77 = not_7 ^ xor_59; xor_78 = not_8 ^ not_9;
xor_79 = not_5 ^ not_7; xor_80 = not_5 ^ not_6; xor_81 = not_8 ^ xor_59;
xor_82 = not_5 ^ xor_59; xor_83 = not_7 ^ not_8; xor_84 = not_6 ^ not_8;
xor_85 = xor_59 ^ xor_76; xor_86 = not_5 ^ xor_76; xor_87 = not_8 ^ xor_75;
xor_88 = xor_59 ^ xor_75; xor_89 = not_9 ^ xor_77; xor_90 = not_6 ^ not_7;
xor_91 = not_9 ^ xor_90; xor_92 = not_9 ^ xor_76; xor_93 = xor_52 ^ xor_78;
xor_94 = not_8 ^ xor_79; xor_95 = not_9 ^ xor_59; and_15 = not_5 & not_6;
and_16 = xor_75 & xor_80; and_17 = xor_52 & xor_81; and_18 = xor_82 & xor_83;
and_19 = xor_77 & xor_84; and_20 = xor_59 & xor_78; and_21 = not_9 & xor_85;
and_22 = xor_84 & xor_86; and_23 = not_5 & xor_87; and_24 = not_6 & xor_88;
and_25 = xor_87 & xor_89; and_26 = xor_86 & xor_91; and_27 = xor_79 & xor_92;
and_28 = xor_75 & xor_93; and_29 = xor_94 & xor_95; xor_96 = and_15 ^ and_18;
xor_97 = and_24 ^ xor_96; xor_98 = and_6 ^ and_8; xor_99 = and_28 ^ xor_97;
xor_100 = and_16 ^ and_9; xor_101 = and_17 ^ xor_98; xor_102 = not_0 ^ xor_0;
xor_103 = and_11 ^ not_4; xor_104 = and_7 ^ xor_99; xor_105 = and_13 ^ and_25;
xor_106 = and_4 ^ xor_100; xor_107 = and_1 ^ xor_101;
xor_108 = and_22 ^ and_27; xor_109 = xor_103 ^ xor_107; xor_110 = not_3 ^ x[4];
xor_111 = and_3 ^ and_26; xor_112 = and_14 ^ and_10; xor_113 = and_19 ^ and_23;
xor_114 = xor_102 ^ xor_113; xor_115 = xor_106 ^ xor_108;
xor_116 = and_2 ^ and_5; xor_117 = xor_109 ^ xor_111;
xor_118 = xor_116 ^ xor_117; xor_119 = and_29 ^ xor_104;
xor_120 = xor_110 ^ xor_119; xor_121 = and_20 ^ xor_105;
xor_122 = and_21 ^ xor_112; xor_123 = and_14 ^ xor_104;
xor_124 = xor_115 ^ xor_123; xor_125 = xor_118 ^ xor_124; not_10 = ~xor_125;
xor_126 = and_12 ^ xor_4; xor_127 = and_17 ^ xor_126;
xor_128 = xor_127 ^ xor_96; xor_129 = xor_105 ^ xor_128;
xor_130 = xor_115 ^ xor_129; xor_131 = xor_122 ^ xor_130; not_11 = ~xor_131;
xor_132 = xor_105 ^ xor_97; xor_133 = xor_114 ^ xor_132;
xor_134 = xor_118 ^ xor_133; xor_135 = not_1 ^ xor_101;
xor_136 = xor_106 ^ xor_135; xor_137 = xor_114 ^ xor_136;
xor_138 = xor_120 ^ xor_137; xor_139 = and_23 ^ xor_100;
xor_140 = xor_108 ^ xor_139; xor_141 = xor_109 ^ xor_140;
xor_142 = xor_120 ^ xor_141; xor_143 = xor_121 ^ xor_142;
xor_144 = and_1 ^ and_9; xor_145 = and_25 ^ xor_144;
xor_146 = xor_145 ^ xor_99; xor_147 = xor_102 ^ xor_146;
xor_148 = xor_110 ^ xor_147; xor_149 = xor_111 ^ xor_148;
xor_150 = xor_112 ^ xor_149; xor_151 = and_16 ^ and_3;
xor_152 = xor_151 ^ xor_98; xor_153 = xor_103 ^ xor_152;
xor_154 = xor_104 ^ xor_153; xor_155 = xor_114 ^ xor_154;
xor_156 = xor_121 ^ xor_155; xor_157 = xor_122 ^ xor_156; y[0] = not_10;
y[1] = not_11; y[2] = xor_134; y[3] = xor_138; y[4] = xor_143; y[5] = xor_150;
y[6] = xor_157;

```

C.21 YSH_{7,2}: 7-bit degree 6, $\delta = 4$, $L = 20$ (7-bit, Category B)

- **Profile:** $\delta = 4$, $L = 20$ (NL = 54), $\deg(S) = 6$, $\mathcal{D}(S) = [2^{63}, 6^{64}]$, FP = 2, SAC deviation 0.0312.
- **Cost:** 51 AND, 76 XOR, 1 NOT, AND depth $D = 3$ (unmasked NanGate 45 nm:

$176.09 \mu\text{m}^2 / 220.7 \text{ GE}$, delay 840.4 ps , logic depth 16; pipelined $d = 2$ DOM-indep:
 284 DFF_X1 [204 gadget + 80 sync], $2070.01 \mu\text{m}^2 / 2594.0 \text{ GE}$, stage delay 802.7 ps).

- **Remark:** Degree 6, the maximum for a 7-bit permutation, at $\delta = 4$, $L = 20$.
- **Inverse** ($\text{YSH}_{7,2}^{-1}$): $\deg(S^{-1}) = 6$, $\mathcal{D}(S^{-1}) = [4^{63}, 6^{64}]$; cost: 34 AND, 164 XOR, 9 NOT, AND depth $D = 3$ (Tier 2: Parallel Transposition Inversion on 29-AND $D = 2$ Gold inverse core; unmasked NanGate 45 nm: $302.71 \mu\text{m}^2 / 379.3 \text{ GE}$, delay 1263.9 ps , logic depth 21; pipelined $d = 2$ DOM-indep: 216 DFF_X1 [136 gadget + 80 sync], $1866.79 \mu\text{m}^2 / 2339.3 \text{ GE}$, stage delay 698.8 ps).

Lookup table ($\text{LUT}[0 \dots 127]$):

```
0x04, 0x5e, 0x34, 0x57, 0x73, 0x7e, 0x05, 0x53, 0x48, 0x60, 0x75, 0x64, 0x21, 0x3c, 0x5a, 0x1c
0x4c, 0x2d, 0x6f, 0x37, 0x1a, 0x4e, 0x7f, 0x12, 0x35, 0x26, 0x1b, 0x31, 0x7d, 0x5b, 0x15, 0x0a
0x47, 0x2f, 0x40, 0x11, 0x42, 0x1f, 0x03, 0x67, 0x14, 0x0e, 0x1e, 0x3d, 0x0f, 0x20, 0x43, 0x55
0x45, 0x16, 0x51, 0x3b, 0x61, 0x07, 0x33, 0x6c, 0x23, 0x02, 0x3a, 0x22, 0x19, 0x0d, 0x46, 0x6b
0x5d, 0x06, 0x30, 0x52, 0x50, 0x3e, 0x7b, 0x2c, 0x39, 0x10, 0x59, 0x49, 0x2a, 0x36, 0x0c, 0x29
0x09, 0x69, 0x77, 0x2e, 0x25, 0x70, 0x1d, 0x71, 0x58, 0x4a, 0x2b, 0x00, 0x6a, 0x4d, 0x5f, 0x41
0x68, 0x01, 0x32, 0x62, 0x17, 0x4b, 0x0b, 0x6e, 0x13, 0x08, 0x44, 0x66, 0x72, 0x5c, 0x63, 0x74
0x76, 0x24, 0x3f, 0x54, 0x28, 0x4f, 0x27, 0x79, 0x38, 0x18, 0x7c, 0x65, 0x78, 0x6d, 0x7a, 0x56
```

Netlist (51 AND, 76 XOR, 1 NOT, $D = 3$):

```
xor_0 = x[3] ^ x[4]; and_0 = x[0] & x[1]; and_1 = x[0] & x[2];
and_2 = x[1] & x[2]; and_3 = x[0] & x[3]; and_4 = x[1] & x[3];
and_5 = x[2] & x[3]; and_6 = x[0] & x[4]; and_7 = x[1] & x[4];
and_8 = x[2] & x[4]; and_9 = x[3] & x[4]; and_10 = x[0] & x[5];
and_11 = x[1] & x[5]; and_12 = x[2] & x[5]; and_13 = x[3] & x[5];
and_14 = x[4] & x[5]; and_15 = x[0] & x[6]; and_16 = x[1] & x[6];
and_17 = x[2] & x[6]; and_18 = x[3] & x[6]; and_19 = x[4] & x[6];
and_20 = x[5] & x[6]; and_21 = x[0] & and_2; and_22 = x[0] & and_5;
and_23 = x[0] & and_8; and_24 = and_0 & and_8; and_25 = and_1 & and_9;
and_26 = and_8 & x[5]; and_27 = and_9 & x[5]; and_28 = and_8 & x[6];
and_29 = and_9 & x[6]; and_30 = x[0] & and_12; and_31 = and_0 & and_12;
and_32 = and_1 & and_13; and_33 = and_1 & and_14; and_34 = and_12 & x[6];
and_35 = and_13 & x[6]; and_36 = and_14 & x[6]; and_37 = x[0] & and_17;
and_38 = and_0 & and_17; and_39 = and_1 & and_18; and_40 = and_1 & and_19;
and_41 = and_1 & and_20; and_42 = and_0 & and_26; and_43 = and_1 & and_27;
and_44 = and_0 & and_28; and_45 = and_1 & and_29; and_46 = and_0 & and_34;
and_47 = and_1 & and_35; and_48 = and_1 & and_36; and_49 = and_21 & and_36;
and_50 = and_22 & and_36; xor_1 = x[2] ^ and_11; xor_2 = and_17 ^ and_20;
xor_3 = and_12 ^ xor_2; xor_4 = and_1 ^ and_13; xor_5 = and_9 ^ xor_1;
xor_6 = x[6] ^ and_16; xor_7 = and_0 ^ and_6; xor_8 = and_3 ^ xor_3;
xor_9 = and_41 ^ and_46; xor_10 = and_39 ^ and_40; xor_11 = and_49 ^ xor_10;
xor_12 = and_38 ^ and_47; xor_13 = and_23 ^ and_24; xor_14 = xor_11 ^ xor_13;
xor_15 = and_45 ^ xor_12; xor_16 = and_21 ^ xor_15; xor_17 = and_42 ^ xor_16;
xor_18 = and_43 ^ xor_8; xor_19 = x[0] ^ and_14; xor_20 = and_30 ^ and_31;
xor_21 = and_25 ^ and_44; xor_22 = and_37 ^ and_48; xor_23 = xor_18 ^ xor_22;
xor_24 = xor_14 ^ xor_17; xor_25 = xor_9 ^ xor_24; xor_26 = and_33 ^ and_50;
xor_27 = xor_21 ^ xor_25; xor_28 = xor_5 ^ xor_7; xor_29 = and_22 ^ xor_27;
xor_30 = xor_20 ^ xor_26; xor_31 = xor_23 ^ xor_30; xor_32 = xor_29 ^ xor_31;
xor_33 = and_7 ^ xor_4; xor_34 = and_5 ^ and_19; xor_35 = and_32 ^ xor_32;
xor_36 = xor_6 ^ xor_19; xor_37 = xor_6 ^ xor_33; xor_38 = and_4 ^ xor_34;
xor_39 = x[1] ^ and_10; xor_40 = xor_28 ^ xor_37; xor_41 = and_2 ^ xor_35;
xor_42 = x[5] ^ xor_41; xor_43 = xor_36 ^ xor_0; xor_44 = x[5] ^ and_4;
xor_45 = and_8 ^ and_15; xor_46 = xor_40 ^ xor_44; xor_47 = xor_45 ^ xor_46;
xor_48 = and_5 ^ and_6; xor_49 = and_10 ^ xor_1; xor_50 = xor_19 ^ xor_33;
xor_51 = xor_48 ^ xor_49; xor_52 = xor_50 ^ xor_51; xor_53 = xor_52 ^ xor_42;
xor_54 = x[3] ^ and_2; xor_55 = and_16 ^ and_20; xor_56 = xor_4 ^ xor_5;
xor_57 = xor_38 ^ xor_54; xor_58 = xor_55 ^ xor_56; xor_59 = xor_57 ^ xor_58;
xor_60 = and_13 ^ and_17; xor_61 = and_18 ^ xor_7; xor_62 = xor_38 ^ xor_43;
xor_63 = xor_60 ^ xor_61; xor_64 = xor_62 ^ xor_63; xor_65 = x[0] ^ xor_8;
xor_66 = xor_34 ^ xor_39; xor_67 = xor_40 ^ xor_65; xor_68 = xor_66 ^ xor_67;
xor_69 = and_8 ^ and_18; xor_70 = xor_28 ^ xor_39; xor_71 = xor_69 ^ xor_70;
xor_72 = xor_71 ^ xor_35; xor_73 = x[2] ^ and_1; xor_74 = xor_43 ^ xor_73;
xor_75 = xor_74 ^ xor_42; not_0 = ~xor_59; y[0] = xor_47; y[1] = xor_53;
y[2] = not_0; y[3] = xor_64; y[4] = xor_68; y[5] = xor_72; y[6] = xor_75;
```

Inverse lookup table ($\text{LUT}_{S^{-1}}[0 \dots 127]$):

```
0x5b, 0x61, 0x39, 0x26, 0x00, 0x06, 0x41, 0x35, 0x69, 0x50, 0x1f, 0x66, 0x4e, 0x3d, 0x29, 0x2c
0x49, 0x23, 0x17, 0x68, 0x28, 0x1e, 0x31, 0x64, 0x79, 0x3c, 0x14, 0x1a, 0x0f, 0x56, 0x2a, 0x25
0x2d, 0x0c, 0x3b, 0x38, 0x71, 0x54, 0x19, 0x76, 0x74, 0x4f, 0x4c, 0x5a, 0x47, 0x11, 0x53, 0x21
0x42, 0x1b, 0x62, 0x36, 0x02, 0x18, 0x4d, 0x13, 0x78, 0x48, 0x3a, 0x33, 0x0d, 0x2b, 0x45, 0x72
```

0x22, 0x5f, 0x24, 0x2e, 0x6a, 0x30, 0x3e, 0x20, 0x08, 0x4b, 0x59, 0x65, 0x10, 0x5d, 0x15, 0x75
 0x44, 0x32, 0x43, 0x07, 0x73, 0x2f, 0x7f, 0x03, 0x58, 0x4a, 0x0e, 0x1d, 0x6d, 0x40, 0x01, 0x5e
 0x09, 0x34, 0x63, 0x6e, 0x0b, 0x7b, 0x6b, 0x27, 0x60, 0x51, 0x5c, 0x3f, 0x37, 0x7d, 0x67, 0x12
 0x55, 0x57, 0x6c, 0x04, 0x6f, 0x0a, 0x70, 0x52, 0x7c, 0x77, 0x7e, 0x46, 0x7a, 0x1c, 0x05, 0x16

Inverse netlist (34 AND, 164 XOR, 9 NOT, $D = 3$):

```
not_0 = ~x[0]; not_1 = ~x[3]; not_2 = ~x[5]; not_3 = ~x[6];
xor_0 = x[1] ^ x[2]; xor_1 = not_1 ^ x[4]; xor_2 = not_0 ^ not_3;
xor_3 = not_2 ^ xor_0; xor_4 = not_0 ^ xor_1; xor_5 = x[2] ^ xor_2;
xor_6 = x[1] ^ xor_1; xor_7 = not_3 ^ xor_0; xor_8 = not_1 ^ not_2;
xor_9 = not_2 ^ xor_5; xor_10 = xor_0 ^ xor_2; xor_11 = not_2 ^ xor_1;
xor_12 = xor_1 ^ xor_7; xor_13 = not_0 ^ xor_0; xor_14 = xor_1 ^ xor_5;
xor_15 = not_1 ^ xor_3; xor_16 = xor_3 ^ xor_4; xor_17 = x[2] ^ xor_4;
xor_18 = not_3 ^ xor_8; xor_19 = not_3 ^ xor_3; xor_20 = not_0 ^ x[4];
xor_21 = xor_20 ^ xor_3; xor_22 = xor_2 ^ xor_6; xor_23 = xor_0 ^ xor_1;
xor_24 = x[4] ^ xor_9; xor_25 = not_3 ^ x[2]; xor_26 = x[4] ^ xor_0;
xor_27 = not_3 ^ xor_6; xor_28 = x[1] ^ x[4]; xor_29 = not_2 ^ xor_28;
xor_30 = not_3 ^ xor_29; xor_31 = xor_0 ^ xor_4; and_0 = xor_10 & xor_4;
and_1 = xor_0 & xor_11; and_2 = xor_12 & xor_13; and_3 = xor_14 & xor_2;
and_4 = xor_15 & xor_16; and_5 = xor_17 & xor_7; and_6 = xor_18 & xor_19;
and_7 = x[2] & xor_21; and_8 = xor_22 & xor_8; and_9 = xor_10 & xor_9;
and_10 = xor_23 & xor_24; and_11 = x[4] & xor_25; and_12 = xor_18 & xor_25;
and_13 = xor_26 & xor_6; xor_32 = and_7 ^ xor_4; xor_33 = and_5 ^ and_9;
xor_34 = and_11 ^ and_12; xor_35 = and_3 ^ xor_13; xor_36 = and_2 ^ xor_10;
xor_37 = xor_33 ^ xor_36; xor_38 = xor_32 ^ xor_35; xor_39 = and_4 ^ xor_34;
xor_40 = and_0 ^ and_10; xor_41 = xor_37 ^ xor_38; xor_42 = and_6 ^ and_8;
xor_43 = xor_11 ^ xor_40; xor_44 = and_13 ^ xor_37; xor_45 = xor_39 ^ xor_44;
xor_46 = xor_27 ^ xor_40; xor_47 = xor_41 ^ xor_46; xor_48 = xor_34 ^ xor_41;
xor_49 = xor_42 ^ xor_48; xor_50 = and_0 ^ xor_30; xor_51 = and_1 ^ xor_50;
xor_52 = xor_38 ^ xor_51; xor_53 = xor_39 ^ xor_52; xor_54 = xor_42 ^ xor_53;
xor_55 = and_6 ^ xor_32; xor_56 = xor_33 ^ xor_55; xor_57 = xor_39 ^ xor_56;
xor_58 = xor_43 ^ xor_57; xor_59 = and_2 ^ xor_0; xor_60 = and_10 ^ xor_59;
xor_61 = and_13 ^ xor_60; xor_62 = xor_35 ^ xor_61; xor_63 = and_1 ^ xor_31;
xor_64 = and_3 ^ xor_63; xor_65 = and_4 ^ xor_64; xor_66 = and_8 ^ xor_65;
xor_67 = and_9 ^ xor_66; xor_68 = and_12 ^ xor_67; xor_69 = xor_43 ^ xor_68;
xor_70 = xor_45 ^ xor_58; xor_71 = xor_45 ^ xor_54; xor_72 = xor_49 ^ xor_58;
xor_73 = xor_62 ^ xor_69; xor_74 = xor_47 ^ xor_58; xor_75 = xor_47 ^ xor_54;
xor_76 = xor_49 ^ xor_62; xor_77 = xor_47 ^ xor_49; xor_78 = xor_58 ^ xor_62;
xor_79 = xor_54 ^ xor_62; xor_80 = xor_49 ^ xor_71; xor_81 = xor_47 ^ xor_71;
xor_82 = xor_62 ^ xor_70; xor_83 = xor_49 ^ xor_70; xor_84 = xor_69 ^ xor_72;
xor_85 = xor_54 ^ xor_58; xor_86 = xor_69 ^ xor_85; xor_87 = xor_69 ^ xor_71;
xor_88 = xor_45 ^ xor_73; xor_89 = xor_62 ^ xor_74; xor_90 = xor_49 ^ xor_69;
and_14 = xor_47 & xor_54; and_15 = xor_70 & xor_75; and_16 = xor_45 & xor_76;
and_17 = xor_77 & xor_78; and_18 = xor_72 & xor_79; and_19 = xor_49 & xor_73;
and_20 = xor_69 & xor_80; and_21 = xor_79 & xor_81; and_22 = xor_47 & xor_82;
and_23 = xor_54 & xor_83; and_24 = xor_82 & xor_84; and_25 = xor_81 & xor_86;
and_26 = xor_74 & xor_87; and_27 = xor_70 & xor_88; and_28 = xor_89 & xor_90;
xor_91 = and_6 ^ not_2; xor_92 = and_15 ^ and_7; xor_93 = and_21 ^ and_26;
xor_94 = and_27 ^ and_9; xor_95 = and_10 ^ not_0; xor_96 = and_12 ^ xor_1;
xor_97 = and_0 ^ xor_92; xor_98 = and_1 ^ xor_91; xor_99 = and_19 ^ xor_93;
xor_100 = and_28 ^ and_3; xor_101 = and_16 ^ xor_94; xor_102 = xor_95 ^ xor_96;
xor_103 = and_5 ^ x[1]; xor_104 = x[2] ^ xor_99; xor_105 = and_2 ^ and_22;
xor_106 = and_23 ^ xor_97; xor_107 = and_24 ^ xor_98;
xor_108 = and_13 ^ and_20; xor_109 = xor_100 ^ xor_104;
xor_110 = xor_101 ^ xor_108; xor_111 = and_14 ^ and_17;
xor_112 = and_25 ^ xor_102; xor_113 = and_18 ^ not_3;
xor_114 = xor_103 ^ xor_105; xor_115 = xor_106 ^ xor_107;
xor_116 = xor_93 ^ xor_98; xor_117 = xor_106 ^ xor_116;
xor_118 = xor_110 ^ xor_117; xor_119 = xor_112 ^ xor_118; not_4 = ~xor_119;
xor_120 = and_8 ^ not_2; xor_121 = and_11 ^ xor_120;
xor_122 = xor_121 ^ xor_92; xor_123 = xor_102 ^ xor_122;
xor_124 = xor_109 ^ xor_123; xor_125 = xor_111 ^ xor_124;
xor_126 = and_5 ^ not_0; xor_127 = and_15 ^ xor_126;
xor_128 = and_22 ^ xor_127; xor_129 = xor_128 ^ xor_91;
xor_130 = xor_109 ^ xor_129; xor_131 = xor_110 ^ xor_130;
xor_132 = xor_113 ^ xor_131; xor_133 = and_24 ^ xor_95;
xor_134 = xor_133 ^ xor_97; xor_135 = xor_100 ^ xor_134;
xor_136 = xor_101 ^ xor_135; xor_137 = xor_111 ^ xor_136;
xor_138 = xor_114 ^ xor_137; xor_139 = and_18 ^ and_3;
xor_140 = and_19 ^ xor_139; xor_141 = xor_140 ^ xor_94;
xor_142 = xor_141 ^ xor_96; xor_143 = xor_105 ^ xor_142;
xor_144 = xor_115 ^ xor_143; not_5 = ~xor_144; xor_145 = and_20 ^ and_27;
xor_146 = xor_145 ^ xor_99; xor_147 = xor_103 ^ xor_146;
xor_148 = xor_107 ^ xor_147; xor_149 = xor_112 ^ xor_148;
xor_150 = and_10 ^ and_4; xor_151 = and_11 ^ xor_150;
xor_152 = and_16 ^ xor_151; xor_153 = and_25 ^ xor_152;
xor_154 = and_28 ^ xor_153; xor_155 = xor_1 ^ xor_154;
xor_156 = xor_104 ^ xor_155; xor_157 = xor_113 ^ xor_156;
```

```

xor_158 = xor_114 ^ xor_157; xor_159 = xor_115 ^ xor_158; not_6 = ~xor_159;
xor_160 = x[1] ^ x[5]; not_7 = ~xor_160; xor_161 = x[5] ^ x[6];
not_8 = ~xor_161; and_29 = not_0 & x[2]; and_30 = x[3] & x[4];
and_31 = not_7 & not_8; and_32 = and_29 & and_30; and_33 = and_32 & and_31;
xor_162 = and_33 ^ xor_125; xor_163 = and_33 ^ xor_138; y[0] = not_4;
y[1] = xor_162; y[2] = xor_132; y[3] = xor_163; y[4] = not_5; y[5] = xor_149;
y[6] = not_6;

```

C.22 7-bit Gold APN x^3 (7-bit, Category A)

- **Profile:** $\delta = 2$, $L = 16$ (NL = 56), $\deg(S) = 2$, $\mathcal{D}(S) = [2^{127}]$, FP = 2, SAC deviation 0.5.
- **Cost:** 14 AND, 57 XOR, 0 NOT, AND depth $D = 1$ (unmasked NanGate 45 nm: $105.87 \mu\text{m}^2 / 132.7$ GE, delay 558.8 ps, logic depth 9; pipelined $d = 2$ DOM-indep: 72 DFF_X1 [56 gadget + 16 sync], $656.49 \mu\text{m}^2 / 822.7$ GE, stage delay 409.5 ps).
- **Remark:** The exact lookup table of x^3 over $\mathbb{F}_2[x]/(x^7 + x + 1)$, at depth $D = 1$; 11/45/1 in [BDM25] and 11/41/1 (LTP = 6) in [KBJ⁺26]: three more AND gates and 12 (resp. 16) more XOR gates.
- **Inverse (S^{-1}):** $\deg(S^{-1}) = 4$, $\mathcal{D}(S^{-1}) = [4^{127}]$; cost: 29 AND, 149 XOR, 15 NOT, AND depth $D = 2$ (Tier 3: 2-fold Gold quadratic composition $((y^5)^9)^{16}$ at $D = 2$; unmasked NanGate 45 nm: $276.64 \mu\text{m}^2 / 346.7$ GE, delay 1191.7 ps, logic depth 22; pipelined $d = 2$ DOM-indep: 166 DFF_X1 [116 gadget + 50 sync], $1542.80 \mu\text{m}^2 / 1933.3$ GE, stage delay 772.0 ps).

Lookup table (LUT[0...127]):

```

0x00, 0x01, 0x08, 0x0f, 0x40, 0x55, 0x78, 0x6b, 0x0c, 0x45, 0x27, 0x68, 0x49, 0x14, 0x52, 0x09
0x60, 0x77, 0x24, 0x35, 0x3e, 0x3d, 0x4a, 0x4f, 0x44, 0x1b, 0x23, 0x7a, 0x1f, 0x54, 0x48, 0x05
0x0a, 0x33, 0x31, 0x0e, 0x26, 0x0b, 0x2d, 0x06, 0x75, 0x04, 0x6d, 0x1a, 0x5c, 0x39, 0x74, 0x17
0x2c, 0x03, 0x5b, 0x72, 0x1e, 0x25, 0x59, 0x64, 0x7b, 0x1c, 0x2f, 0x4e, 0x4c, 0x3f, 0x28, 0x5d
0x50, 0x71, 0x1d, 0x3a, 0x0d, 0x38, 0x70, 0x43, 0x36, 0x5f, 0x58, 0x37, 0x6e, 0x13, 0x30, 0x4b
0x21, 0x16, 0x20, 0x11, 0x62, 0x41, 0x53, 0x76, 0x6f, 0x10, 0x4d, 0x34, 0x29, 0x42, 0x3b, 0x56
0x66, 0x7f, 0x18, 0x07, 0x57, 0x5a, 0x19, 0x12, 0x73, 0x22, 0x2e, 0x79, 0x47, 0x02, 0x2a, 0x69
0x51, 0x5e, 0x63, 0x6a, 0x7e, 0x65, 0x7c, 0x61, 0x6c, 0x2b, 0x7d, 0x3c, 0x46, 0x15, 0x67, 0x32

```

Netlist (14 AND, 57 XOR, 0 NOT, $D = 1$):

```

xor_0 = x[0] ^ x[5]; xor_1 = x[2] ^ x[3]; xor_2 = x[0] ^ x[3];
xor_3 = x[1] ^ x[4]; xor_4 = x[1] ^ xor_0; xor_5 = x[2] ^ xor_0;
xor_6 = x[1] ^ xor_2; xor_7 = x[3] ^ x[4]; xor_8 = x[0] ^ x[6];
xor_9 = x[4] ^ xor_2; xor_10 = x[3] ^ x[5]; xor_11 = x[6] ^ xor_1;
xor_12 = x[2] ^ xor_3; xor_13 = x[5] ^ xor_1; xor_14 = x[4] ^ xor_0;
xor_15 = x[6] ^ xor_0; xor_16 = x[0] ^ x[2]; xor_17 = x[1] ^ x[6];
xor_18 = x[3] ^ xor_0; xor_19 = x[6] ^ xor_3; and_0 = x[2] & x[3];
and_1 = x[4] & x[6]; and_2 = x[0] & xor_1; and_3 = xor_7 & xor_0;
and_4 = xor_3 & xor_8; and_5 = xor_9 & xor_10; and_6 = xor_6 & xor_11;
and_7 = xor_12 & xor_13; and_8 = x[1] & xor_14; and_9 = x[3] & xor_15;
and_10 = xor_16 & xor_4; and_11 = xor_5 & xor_17; and_12 = xor_6 & xor_5;
and_13 = xor_18 & xor_19; xor_20 = and_5 ^ and_9; xor_21 = and_4 ^ and_12;
xor_22 = x[2] ^ and_0; xor_23 = and_3 ^ and_10; xor_24 = and_2 ^ xor_20;
xor_25 = and_6 ^ and_7; xor_26 = and_11 ^ xor_25; xor_27 = xor_21 ^ xor_22;
xor_28 = and_1 ^ xor_27; xor_29 = x[4] ^ xor_23; xor_30 = and_8 ^ xor_28;
xor_31 = xor_1 ^ xor_24; xor_32 = and_3 ^ xor_26; xor_33 = x[3] ^ and_11;
xor_34 = and_13 ^ xor_21; xor_35 = xor_24 ^ xor_33; xor_36 = xor_34 ^ xor_35;
xor_37 = xor_0 ^ and_0; xor_38 = and_7 ^ xor_23; xor_39 = xor_31 ^ xor_37;
xor_40 = xor_38 ^ xor_39; xor_41 = x[0] ^ and_8; xor_42 = and_12 ^ xor_31;
xor_43 = xor_32 ^ xor_41; xor_44 = xor_42 ^ xor_43; xor_45 = xor_4 ^ xor_30;
xor_46 = xor_32 ^ xor_45; xor_47 = x[6] ^ and_10; xor_48 = xor_20 ^ xor_26;
xor_49 = xor_27 ^ xor_47; xor_50 = xor_48 ^ xor_49; xor_51 = x[0] ^ and_2;
xor_52 = and_13 ^ xor_29; xor_53 = xor_51 ^ xor_52; xor_54 = x[6] ^ and_9;
xor_55 = xor_29 ^ xor_30; xor_56 = xor_54 ^ xor_55; y[0] = xor_36;
y[1] = xor_40; y[2] = xor_44; y[3] = xor_46; y[4] = xor_50; y[5] = xor_53;
y[6] = xor_56;

```

Inverse lookup table (LUT $_{S^{-1}}$ [0...127]):

```

0x00, 0x01, 0x6d, 0x31, 0x29, 0x1f, 0x27, 0x63, 0x02, 0x0f, 0x20, 0x25, 0x08, 0x44, 0x23, 0x03
0x59, 0x53, 0x67, 0x4d, 0x0d, 0x7d, 0x51, 0x2f, 0x62, 0x66, 0x2b, 0x19, 0x39, 0x42, 0x34, 0x1c
0x52, 0x50, 0x69, 0x1a, 0x12, 0x35, 0x24, 0x0a, 0x3e, 0x5c, 0x6e, 0x79, 0x30, 0x26, 0x6a, 0x3a
0x4e, 0x22, 0x7f, 0x21, 0x5b, 0x13, 0x48, 0x4b, 0x45, 0x2d, 0x43, 0x5e, 0x7b, 0x15, 0x14, 0x3d
0x04, 0x55, 0x5d, 0x47, 0x18, 0x09, 0x7c, 0x6c, 0x1e, 0x0c, 0x16, 0x4f, 0x3c, 0x5a, 0x3b, 0x17
0x40, 0x70, 0x0e, 0x56, 0x1d, 0x05, 0x5f, 0x64, 0x4a, 0x36, 0x65, 0x32, 0x2c, 0x3f, 0x71, 0x49
0x10, 0x77, 0x54, 0x72, 0x37, 0x75, 0x60, 0x7e, 0x0b, 0x6f, 0x73, 0x07, 0x78, 0x2a, 0x4c, 0x58
0x46, 0x41, 0x33, 0x68, 0x2e, 0x28, 0x57, 0x11, 0x06, 0x6b, 0x1b, 0x38, 0x76, 0x7a, 0x74, 0x61

```

Inverse netlist (29 AND, 149 XOR, 15 NOT, $D = 2$):

```

not_0 = ~x[0]; not_1 = ~x[2]; not_2 = ~x[4]; not_3 = ~x[6];
xor_0 = not_2 ^ x[3]; xor_1 = not_3 ^ x[1]; xor_2 = not_1 ^ x[5];
xor_3 = xor_0 ^ not_1; xor_4 = not_0 ^ x[5]; xor_5 = xor_4 ^ not_3;
xor_6 = xor_2 ^ not_2; xor_7 = xor_0 ^ xor_1; xor_8 = not_1 ^ not_2;
xor_9 = not_2 ^ x[5]; xor_10 = xor_1 ^ x[3]; xor_11 = xor_3 ^ not_3;
xor_12 = xor_2 ^ not_3; xor_13 = not_1 ^ x[3]; xor_14 = not_1 ^ x[1];
xor_15 = xor_0 ^ xor_5; xor_16 = xor_3 ^ x[1]; xor_17 = not_0 ^ x[3];
xor_18 = xor_6 ^ not_0; xor_19 = xor_6 ^ x[1]; xor_20 = xor_7 ^ x[5];
xor_21 = x[1] ^ x[3]; xor_22 = xor_5 ^ x[3]; xor_23 = xor_0 ^ not_0;
xor_24 = xor_2 ^ xor_23; xor_25 = xor_1 ^ not_1; xor_26 = not_0 ^ not_1;
xor_27 = xor_1 ^ xor_3; xor_28 = xor_2 ^ x[3]; xor_29 = x[3] ^ x[5];
xor_30 = not_0 ^ x[1]; and_0 = xor_8 & xor_9; and_1 = xor_10 & not_1;
and_2 = xor_0 & xor_11; and_3 = xor_12 & xor_13; and_4 = xor_14 & xor_8;
and_5 = xor_15 & xor_7; and_6 = xor_16 & xor_17; and_7 = xor_17 & xor_7;
and_8 = xor_18 & xor_19; and_9 = xor_20 & xor_21; and_10 = xor_20 & xor_22;
and_11 = xor_24 & xor_25; and_12 = xor_26 & xor_27; and_13 = xor_28 & xor_5;
xor_31 = and_0 ^ not_0; xor_32 = and_8 ^ xor_31; xor_33 = xor_29 ^ xor_32;
xor_34 = and_1 ^ not_2; xor_35 = and_3 ^ and_4; xor_36 = not_1 ^ xor_33;
xor_37 = and_12 ^ xor_34; xor_38 = and_7 ^ not_3; xor_39 = and_2 ^ and_5;
xor_40 = xor_35 ^ xor_39; xor_41 = and_11 ^ xor_36; xor_42 = and_10 ^ xor_38;
xor_43 = xor_35 ^ xor_37; xor_44 = and_7 ^ xor_30; xor_45 = xor_34 ^ xor_44;
xor_46 = xor_40 ^ xor_45; xor_47 = and_1 ^ and_9; xor_48 = and_13 ^ xor_47;
xor_49 = xor_40 ^ xor_48; xor_50 = xor_41 ^ xor_49; xor_51 = xor_42 ^ xor_50;
not_4 = ~xor_51; xor_52 = and_6 ^ xor_41; xor_53 = xor_43 ^ xor_52;
xor_54 = xor_36 ^ x[1]; xor_55 = xor_37 ^ xor_54; xor_56 = xor_42 ^ xor_55;
not_5 = ~xor_56; xor_57 = and_2 ^ xor_1; xor_58 = and_6 ^ xor_57;
xor_59 = and_11 ^ xor_58; xor_60 = and_13 ^ xor_59; xor_61 = xor_33 ^ xor_60;
not_6 = ~xor_61; xor_62 = and_9 ^ xor_32; xor_63 = xor_43 ^ xor_62;
not_7 = ~xor_63; xor_64 = and_3 ^ xor_8; xor_65 = and_5 ^ xor_64;
xor_66 = and_6 ^ xor_65; xor_67 = and_8 ^ xor_66; xor_68 = xor_38 ^ xor_67;
not_8 = ~xor_68; xor_69 = not_6 ^ xor_46; xor_70 = not_5 ^ xor_46;
xor_71 = not_6 ^ xor_53; xor_72 = not_7 ^ not_8; xor_73 = not_4 ^ not_6;
xor_74 = not_4 ^ not_5; xor_75 = not_7 ^ xor_53; xor_76 = not_4 ^ xor_53;
xor_77 = not_6 ^ not_7; xor_78 = not_5 ^ not_7; xor_79 = xor_53 ^ xor_70;
xor_80 = not_4 ^ xor_70; xor_81 = not_7 ^ xor_69; xor_82 = xor_53 ^ xor_69;
xor_83 = not_8 ^ xor_71; xor_84 = not_5 ^ not_6; xor_85 = not_8 ^ xor_84;
xor_86 = not_8 ^ xor_70; xor_87 = xor_46 ^ xor_72; xor_88 = not_7 ^ xor_73;
xor_89 = not_8 ^ xor_53; and_14 = not_4 & not_5; and_15 = xor_69 & xor_74;
and_16 = xor_46 & xor_75; and_17 = xor_76 & xor_77; and_18 = xor_71 & xor_78;
and_19 = xor_53 & xor_72; and_20 = not_8 & xor_79; and_21 = xor_78 & xor_80;
and_22 = not_4 & xor_81; and_23 = not_5 & xor_82; and_24 = xor_81 & xor_83;
and_25 = xor_80 & xor_85; and_26 = xor_73 & xor_86; and_27 = xor_69 & xor_87;
and_28 = xor_88 & xor_89; xor_90 = and_21 ^ and_26; xor_91 = and_13 ^ not_0;
xor_92 = and_2 ^ and_22; xor_93 = and_25 ^ and_28; xor_94 = and_15 ^ and_8;
xor_95 = xor_94 ^ xor_64; xor_96 = and_1 ^ and_4; xor_97 = and_20 ^ xor_90;
xor_98 = xor_92 ^ xor_96; xor_99 = and_5 ^ and_7; xor_100 = and_9 ^ xor_91;
xor_101 = and_18 ^ xor_93; xor_102 = and_24 ^ xor_95;
xor_103 = xor_102 ^ xor_97; xor_104 = and_11 ^ and_12;
xor_105 = xor_29 ^ xor_100; xor_106 = xor_104 ^ xor_105;
xor_107 = and_12 ^ not_3; xor_108 = and_0 ^ and_10; xor_109 = xor_98 ^ xor_107;
xor_110 = and_24 ^ x[1]; xor_111 = xor_99 ^ not_2; xor_112 = and_14 ^ and_16;
xor_113 = and_17 ^ xor_112; xor_114 = xor_103 ^ xor_113;
xor_115 = xor_106 ^ xor_114; not_9 = ~xor_115; xor_116 = and_6 ^ not_3;
xor_117 = and_22 ^ xor_116; xor_118 = and_25 ^ xor_117;
xor_119 = xor_99 ^ xor_118; xor_120 = xor_103 ^ xor_119; not_10 = ~xor_120;
xor_121 = and_23 ^ and_27; xor_122 = xor_121 ^ xor_95;
xor_123 = xor_101 ^ xor_122; xor_124 = xor_106 ^ xor_123; not_11 = ~xor_124;
xor_125 = xor_8 ^ xor_90; xor_126 = xor_91 ^ xor_125;
xor_127 = xor_98 ^ xor_126; xor_128 = xor_101 ^ xor_127;
xor_129 = xor_108 ^ xor_128; not_12 = ~xor_129; xor_130 = and_13 ^ and_19;
xor_131 = xor_130 ^ xor_64; xor_132 = xor_131 ^ xor_93;
xor_133 = xor_109 ^ xor_132; xor_134 = xor_110 ^ xor_133; not_13 = ~xor_134;
xor_135 = and_27 ^ and_28; xor_136 = xor_135 ^ xor_97;
xor_137 = xor_108 ^ xor_136; xor_138 = xor_109 ^ xor_137;
xor_139 = xor_111 ^ xor_138; xor_140 = and_0 ^ and_3;
xor_141 = and_6 ^ xor_140; xor_142 = and_15 ^ xor_141;
xor_143 = and_16 ^ xor_142; xor_144 = xor_92 ^ xor_143;
xor_145 = xor_100 ^ xor_144; xor_146 = xor_101 ^ xor_145;

```

```
xor_147 = xor_110 ^ xor_146; xor_148 = xor_111 ^ xor_147; not_14 = ~xor_148;
y[0] = not_9; y[1] = not_10; y[2] = not_11; y[3] = not_12; y[4] = not_13;
y[5] = xor_139; y[6] = not_14;
```

C.23 7-bit Kasami APN x^{13} (7-bit, Category A)

- **Profile:** $\delta = 2$, $L = 16$ (NL = 56), $\deg(S) = 3$, $\mathcal{D}(S) = [3^{127}]$, FP = 2, SAC deviation 0.0625.
- **Cost:** 30 AND, 114 XOR, 0 NOT, AND depth $D = 2$ (unmasked NanGate 45 nm: $213.86 \mu\text{m}^2 / 268.0$ GE, delay 1007.6 ps, logic depth 16; pipelined $d = 2$ DOM-indep: 148 DFF_X1 [120 gadget + 28 sync], $1352.34 \mu\text{m}^2 / 1694.7$ GE, stage delay 644.6 ps).
- **Remark:** The exact lookup table of x^{13} over $\mathbb{F}_2[x]/(x^7 + x + 1)$; 27/115/2 for x^{11}/x^{13} in [BDM26]: three more AND gates and one fewer XOR gate.
- **Inverse (S^{-1}):** $\deg(S^{-1}) = 3$, $\mathcal{D}(S^{-1}) = [3^{127}]$; cost: 37 AND, 139 XOR, 0 NOT, AND depth $D = 2$ (Tier 3: Kasami APN inverse decomposition $y^{88} = (y^3 y^8)^8$ at $D = 2$, via the Welch coset exponent $88 \equiv 11 \times 8 \pmod{127}$; unmasked NanGate 45 nm: $261.21 \mu\text{m}^2 / 327.3$ GE, delay 1056.7 ps, logic depth 17; pipelined $d = 2$ DOM-indep: 204 DFF_X1 [148 gadget + 56 sync], $1759.86 \mu\text{m}^2 / 2205.3$ GE, stage delay 735.9 ps).

Lookup table (LUT[0...127]):

```
0x00, 0x01, 0x43, 0x1a, 0x65, 0x42, 0x39, 0x68, 0x3a, 0x0e, 0x26, 0x1e, 0x2f, 0x49, 0x67, 0x23
0x69, 0x64, 0x1b, 0x32, 0x5f, 0x38, 0x33, 0x5e, 0x08, 0x03, 0x74, 0x0f, 0x3f, 0x2a, 0x16, 0x5d
0x24, 0x13, 0x79, 0x7e, 0x7a, 0x34, 0x7d, 0x2d, 0x53, 0x75, 0x6c, 0x2e, 0x3e, 0x15, 0x10, 0x71
0x14, 0x2b, 0x46, 0x35, 0x51, 0x5b, 0x58, 0x30, 0x20, 0x3d, 0x41, 0x44, 0x27, 0x7b, 0x56, 0x3c
0x5a, 0x54, 0x6e, 0x55, 0x0c, 0x52, 0x40, 0x05, 0x4a, 0x17, 0x72, 0x4e, 0x06, 0x7f, 0x0d, 0x3b
0x4d, 0x21, 0x12, 0x37, 0x6d, 0x1d, 0x4b, 0x5c, 0x63, 0x6f, 0x61, 0x70, 0x28, 0x4c, 0x18, 0x4f
0x22, 0x76, 0x02, 0x0b, 0x2c, 0x77, 0x31, 0x19, 0x48, 0x6b, 0x59, 0x73, 0x1f, 0x47, 0x78, 0x07
0x50, 0x62, 0x25, 0x36, 0x60, 0x11, 0x29, 0x57, 0x1c, 0x6a, 0x09, 0x0a, 0x04, 0x45, 0x66, 0x7c
```

Netlist (30 AND, 114 XOR, 0 NOT, $D = 2$):

```
a_in = in_val[6:0]; split_2 = a_in[0:0]; split_3 = a_in[1:1];
split_4 = a_in[2:2]; split_5 = a_in[3:3]; split_6 = a_in[4:4];
split_7 = a_in[5:5]; split_8 = a_in[6:6]; xor_16 = split_3 ^ split_4;
xor_9 = split_2 ^ split_5; xor_19 = split_3 ^ split_5;
and_30 = split_3 & split_5; xor_10 = split_2 ^ split_6;
xor_12 = split_4 ^ split_6; xor_27 = split_5 ^ split_6;
paar_shared_xor_0_split_3_split_6 = split_3 ^ split_6;
xor_14 = split_4 ^ split_7; xor_17 = split_6 ^ split_7;
xor_21 = split_5 ^ split_7; xor_13 = split_7 ^ split_8;
xor_26 = split_4 ^ split_8; xor_20 = split_4 ^ xor_9; xor_22 = split_3 ^ xor_9;
xor_24 = split_8 ^ xor_9; xor_15 = split_7 ^ xor_10; xor_18 = split_4 ^ xor_10;
and_36 = xor_19 & xor_10; xor_29 = split_8 ^ xor_12; xor_28 = split_8 ^ xor_27;
and_31 = split_2 ^ xor_14; and_34 = xor_16 & xor_17; and_38 = xor_12 & xor_21;
xor_23 = split_2 ^ xor_13; and_32 = split_4 & xor_13; and_37 = xor_20 & split_8;
and_39 = xor_22 & xor_21; and_33 = split_3 & xor_15; and_35 = split_5 & xor_18;
and_44 = xor_15 & xor_29; and_43 = xor_22 & xor_28;
xor_25 = split_7 ^ paar_shared_xor_0_split_3_split_6;
and_41 = paar_shared_xor_0_split_3_split_6 & xor_24;
xor_67 = paar_shared_xor_0_split_3_split_6 ^ and_36; xor_54 = and_30 ^ and_34;
xor_50 = split_4 ^ and_38; and_40 = xor_10 & xor_23; xor_47 = split_2 ^ and_37;
xor_59 = and_37 ^ and_38; xor_70 = and_31 ^ and_37; xor_58 = and_32 ^ and_33;
xor_51 = and_32 ^ and_35; xor_45 = and_31 ^ and_44; and_42 = xor_25 & xor_26;
xor_48 = and_39 ^ and_41; xor_74 = and_33 ^ and_40; xor_78 = and_40 ^ and_43;
xor_52 = split_6 ^ xor_47; xor_61 = xor_58 ^ xor_59; xor_64 = split_3 ^ xor_51;
xor_72 = xor_51 ^ xor_70; xor_46 = split_7 ^ xor_45; xor_71 = and_42 ^ xor_50;
xor_75 = and_42 ^ xor_47; xor_55 = xor_48 ^ xor_54; xor_60 = and_43 ^ xor_48;
xor_68 = xor_45 ^ xor_48; xor_79 = xor_50 ^ xor_78; xor_49 = and_36 ^ xor_46;
xor_73 = xor_71 ^ xor_72; xor_76 = xor_74 ^ xor_75; xor_56 = xor_52 ^ xor_55;
xor_62 = xor_46 ^ xor_60; xor_69 = xor_67 ^ xor_68; xor_80 = xor_52 ^ xor_79;
xor_53 = split_8 ^ xor_49; xor_65 = xor_64 ^ xor_49; xor_63 = xor_61 ^ xor_62;
xor_99 = xor_69 ^ xor_73; xor_57 = xor_56 ^ xor_53; xor_77 = xor_76 ^ xor_53;
xor_84 = xor_65 ^ xor_73; xor_98 = xor_65 ^ xor_80; xor_83 = xor_63 ^ xor_73;
xor_88 = xor_63 ^ xor_65; xor_91 = xor_63 ^ xor_69; and_102 = xor_63 & xor_69;
xor_121 = xor_63 ^ xor_80; xor_100 = xor_80 ^ xor_99; xor_81 = xor_57 ^ xor_69;
xor_82 = xor_57 ^ xor_73; xor_85 = xor_77 ^ xor_80; xor_86 = xor_65 ^ xor_77;
xor_89 = xor_73 ^ xor_77; xor_93 = xor_69 ^ xor_77; xor_101 = xor_80 ^ xor_84;
```

```

xor_97 = xor_77 ^ xor_83; xor_92 = xor_65 ^ xor_81; xor_94 = xor_63 ^ xor_81;
xor_96 = xor_80 ^ xor_81; xor_87 = xor_77 ^ xor_82; xor_90 = xor_65 ^ xor_82;
and_108 = xor_91 & xor_82; xor_95 = xor_57 ^ xor_85; and_104 = xor_65 & xor_85;
and_103 = xor_57 & xor_86; and_106 = xor_88 & xor_89; and_110 = xor_84 & xor_93;
and_114 = xor_97 & xor_98; and_109 = xor_92 & xor_80; and_111 = xor_94 & xor_93;
and_115 = xor_94 & xor_100; and_113 = xor_83 & xor_96; and_105 = xor_63 & xor_87;
and_116 = xor_87 & xor_101; and_107 = xor_69 & xor_90; and_112 = xor_82 & xor_95;
xor_139 = xor_121 ^ and_104; xor_131 = and_102 ^ and_103;
xor_142 = xor_63 ^ and_103; xor_125 = xor_65 ^ and_110;
xor_119 = xor_57 ^ and_109; xor_117 = and_111 ^ and_113;
xor_122 = and_105 ^ and_114; xor_147 = and_105 ^ and_109;
xor_123 = and_108 ^ and_116; xor_126 = and_110 ^ and_116;
xor_150 = and_115 ^ xor_125; xor_124 = and_112 ^ xor_119;
xor_132 = and_106 ^ xor_119; xor_118 = xor_77 ^ xor_117;
xor_128 = and_115 ^ xor_117; xor_129 = xor_121 ^ xor_122;
xor_136 = and_107 ^ xor_122; xor_127 = xor_80 ^ xor_123;
xor_143 = xor_123 ^ xor_125; xor_140 = xor_126 ^ xor_139;
xor_120 = xor_73 ^ xor_118; xor_144 = xor_142 ^ xor_128;
xor_151 = xor_129 ^ xor_118; xor_137 = xor_126 ^ xor_136;
xor_133 = xor_131 ^ xor_127; xor_148 = xor_147 ^ xor_127;
xor_145 = xor_124 ^ xor_143; xor_130 = xor_120 ^ xor_124;
xor_134 = xor_132 ^ xor_120; xor_141 = xor_120 ^ xor_140;
xor_152 = xor_150 ^ xor_151; xor_149 = xor_128 ^ xor_148;
xor_146 = xor_144 ^ xor_145; xor_138 = xor_137 ^ xor_130;
xor_153 = xor_129 ^ xor_130; xor_135 = xor_133 ^ xor_134;
concat_154 = { xor_153, xor_152, xor_149, xor_146, xor_141, xor_138, xor_135 };
out_val = concat_154;

```

Inverse lookup table ($\text{LUT}_{S^{-1}}[0 \dots 127]$):

```

0x00, 0x01, 0x62, 0x19, 0x7c, 0x47, 0x4c, 0x6f, 0x18, 0x7a, 0x7b, 0x63, 0x44, 0x4e, 0x09, 0x1b
0x2e, 0x75, 0x52, 0x21, 0x30, 0x2d, 0x1e, 0x49, 0x5e, 0x67, 0x03, 0x12, 0x78, 0x55, 0x0b, 0x6c
0x38, 0x51, 0x60, 0x0f, 0x20, 0x72, 0x0a, 0x3c, 0x76, 0x1d, 0x31, 0x64, 0x27, 0x2b, 0x0c
0x37, 0x66, 0x13, 0x16, 0x25, 0x33, 0x73, 0x53, 0x15, 0x06, 0x08, 0x4f, 0x3f, 0x39, 0x2c, 0x1c
0x46, 0x3a, 0x05, 0x02, 0x3b, 0x7d, 0x32, 0x6d, 0x68, 0x0d, 0x48, 0x56, 0x5d, 0x50, 0x4b, 0x5f
0x70, 0x34, 0x45, 0x28, 0x41, 0x43, 0x3e, 0x77, 0x36, 0x6a, 0x40, 0x35, 0x57, 0x1f, 0x17, 0x14
0x74, 0x5a, 0x71, 0x58, 0x11, 0x04, 0x7e, 0x0e, 0x07, 0x10, 0x79, 0x69, 0x2a, 0x54, 0x42, 0x59
0x5b, 0x2f, 0x4a, 0x6b, 0x1a, 0x29, 0x61, 0x65, 0x6e, 0x22, 0x24, 0x3d, 0x7f, 0x26, 0x23, 0x4d

```

Inverse netlist (37 AND, 139 XOR, 0 NOT, $D = 2$):

```

xor_0 = x[1] ^ x[4]; xor_1 = x[0] ^ x[5]; xor_2 = x[2] ^ x[6];
xor_3 = x[3] ^ x[5]; xor_4 = xor_2 ^ x[3]; xor_5 = xor_0 ^ x[6];
xor_6 = xor_0 ^ x[2]; xor_7 = x[3] ^ x[4]; xor_8 = xor_0 ^ xor_2;
xor_9 = x[2] ^ x[4]; xor_10 = xor_2 ^ x[4]; xor_11 = xor_1 ^ x[4];
xor_12 = x[0] ^ x[1]; xor_13 = xor_1 ^ x[1]; xor_14 = x[0] ^ x[6];
xor_15 = x[2] ^ x[3]; xor_16 = xor_7 ^ x[0]; xor_17 = xor_12 ^ x[3];
xor_18 = xor_3 ^ x[2]; xor_19 = xor_1 ^ x[6]; xor_20 = x[0] ^ x[2];
xor_21 = xor_1 ^ x[2]; xor_22 = x[1] ^ x[6]; xor_23 = xor_1 ^ x[3];
xor_24 = x[1] ^ x[5]; xor_25 = xor_5 ^ x[0]; xor_26 = xor_8 ^ x[5];
xor_27 = x[3] ^ x[6]; xor_28 = xor_3 ^ xor_6; xor_29 = xor_1 ^ xor_8;
xor_30 = xor_0 ^ x[3]; xor_31 = xor_1 ^ xor_4; xor_32 = xor_10 ^ xor_3;
xor_33 = xor_3 ^ x[6]; xor_34 = xor_11 ^ xor_4; xor_35 = xor_0 ^ xor_3;
xor_36 = xor_12 ^ xor_2; xor_37 = xor_0 ^ xor_4; xor_38 = xor_5 ^ x[5];
xor_39 = xor_14 ^ x[4]; and_0 = x[2] & x[3]; and_1 = x[4] & x[6];
and_2 = xor_15 & x[0]; and_3 = xor_1 & xor_7; and_4 = xor_0 & xor_14;
and_5 = xor_16 & xor_3; and_6 = xor_17 & xor_4; and_7 = xor_18 & xor_6;
and_8 = xor_11 & x[1]; and_9 = xor_19 & x[3]; and_10 = xor_13 & xor_20;
and_11 = xor_21 & xor_22; and_12 = xor_17 & xor_21; and_13 = xor_23 & xor_5;
xor_40 = and_7 ^ x[2]; xor_41 = and_5 ^ and_9; xor_42 = and_11 ^ and_12;
xor_43 = and_3 ^ x[0]; xor_44 = and_2 ^ x[3]; xor_45 = xor_41 ^ xor_44;
xor_46 = xor_40 ^ xor_43; xor_47 = and_4 ^ xor_42; xor_48 = and_0 ^ and_10;
xor_49 = xor_45 ^ xor_46; xor_50 = and_6 ^ and_8; xor_51 = xor_48 ^ x[6];
xor_52 = and_13 ^ xor_45; xor_53 = xor_47 ^ xor_52; xor_54 = xor_48 ^ x[5];
xor_55 = xor_49 ^ xor_54; xor_56 = xor_42 ^ xor_49; xor_57 = xor_50 ^ xor_56;
xor_58 = and_0 ^ xor_24; xor_59 = and_1 ^ xor_58; xor_60 = xor_46 ^ xor_59;
xor_61 = xor_47 ^ xor_60; xor_62 = xor_50 ^ xor_61; xor_63 = and_6 ^ xor_40;
xor_64 = xor_41 ^ xor_63; xor_65 = xor_47 ^ xor_64; xor_66 = xor_51 ^ xor_65;
xor_67 = and_2 ^ x[4]; xor_68 = and_10 ^ xor_67; xor_69 = and_13 ^ xor_68;
xor_70 = xor_43 ^ xor_69; xor_71 = and_1 ^ xor_9; xor_72 = and_3 ^ xor_71;
xor_73 = and_4 ^ xor_72; xor_74 = and_8 ^ xor_73; xor_75 = and_9 ^ xor_74;
xor_76 = and_12 ^ xor_75; xor_77 = xor_51 ^ xor_76; and_14 = xor_53 & x[0];
and_15 = xor_5 & xor_55; xor_78 = xor_53 ^ xor_55; and_16 = xor_25 & xor_78;
and_17 = xor_26 & xor_57; and_18 = xor_27 & xor_62; xor_79 = xor_57 ^ xor_62;
and_19 = xor_28 & xor_79; xor_80 = xor_53 ^ xor_57; xor_81 = xor_55 ^ xor_62;
and_20 = xor_29 & xor_80; and_21 = xor_30 & xor_81; xor_82 = xor_80 ^ xor_81;
and_22 = xor_31 & xor_82; and_23 = xor_32 & xor_66; and_24 = xor_33 & xor_70;
and_25 = xor_3 & xor_77; xor_83 = xor_66 ^ xor_70; and_26 = xor_9 & xor_83;
xor_84 = xor_66 ^ xor_77; and_27 = xor_10 & xor_84; xor_85 = xor_70 ^ xor_77;

```

```

and_28 = xor_85 & x[6]; xor_86 = xor_53 ^ xor_66; xor_87 = xor_55 ^ xor_70;
xor_88 = xor_57 ^ xor_77; and_29 = xor_34 & xor_86; and_30 = xor_35 & xor_87;
xor_89 = xor_86 ^ xor_87; and_31 = xor_36 & xor_89; and_32 = xor_37 & xor_88;
xor_90 = xor_88 ^ xor_62; and_33 = xor_6 & xor_90; xor_91 = xor_86 ^ xor_88;
xor_92 = xor_87 ^ xor_62; and_34 = xor_13 & xor_91; and_35 = xor_38 & xor_92;
xor_93 = xor_91 ^ xor_92; and_36 = xor_39 & xor_93; xor_94 = and_21 ^ and_22;
xor_95 = and_29 ^ and_30; xor_96 = and_31 ^ xor_94; xor_97 = and_24 ^ and_25;
xor_98 = and_19 ^ and_28; xor_99 = and_33 ^ xor_95; xor_100 = and_32 ^ and_36;
xor_101 = and_16 ^ xor_97; xor_102 = and_20 ^ and_35;
xor_103 = and_15 ^ xor_96; xor_104 = and_17 ^ xor_99;
xor_105 = xor_100 ^ xor_103; xor_106 = xor_98 ^ xor_101;
xor_107 = xor_106 ^ xor_102; xor_108 = and_34 ^ xor_104;
xor_109 = and_14 ^ and_34; xor_110 = and_27 ^ xor_96;
xor_111 = xor_105 ^ xor_107; xor_112 = xor_108 ^ xor_111;
xor_113 = and_19 ^ xor_99; xor_114 = xor_105 ^ xor_113;
xor_115 = and_18 ^ and_22; xor_116 = and_23 ^ xor_115;
xor_117 = and_30 ^ xor_116; xor_118 = xor_117 ^ xor_100;
xor_119 = xor_107 ^ xor_118; xor_120 = and_23 ^ and_26;
xor_121 = xor_94 ^ xor_120; xor_122 = xor_121 ^ xor_101;
xor_123 = xor_108 ^ xor_122; xor_124 = and_26 ^ xor_95;
xor_125 = xor_124 ^ xor_102; xor_126 = xor_109 ^ xor_125;
xor_127 = xor_110 ^ xor_126; xor_128 = and_18 ^ and_33;
xor_129 = xor_97 ^ xor_128; xor_130 = xor_129 ^ xor_98;
xor_131 = xor_130 ^ xor_103; xor_132 = xor_109 ^ xor_131;
xor_133 = and_20 ^ and_23; xor_134 = and_25 ^ xor_133;
xor_135 = and_32 ^ xor_134; xor_136 = xor_98 ^ xor_135;
xor_137 = xor_104 ^ xor_136; xor_138 = xor_110 ^ xor_137; y[0] = xor_112;
y[1] = xor_114; y[2] = xor_119; y[3] = xor_123; y[4] = xor_127; y[5] = xor_132;
y[6] = xor_138;

```

C.24 7-bit inversion class (7-bit, Category A)

- **Profile:** $\delta = 2$, $L = 20$ (NL = 54), $\deg(S) = 6$, $\mathcal{D}(S) = [6^{127}]$, FP = 0, SAC deviation 0.0938.
- **Cost:** 42 AND, 182 XOR, 6 NOT, AND depth $D = 3$ (unmasked NanGate 45 nm: $338.35 \mu\text{m}^2 / 424.0 \text{ GE}$, delay 2043.5 ps, logic depth 33; pipelined $d = 2$ DOM-indep: 212 DFF_X1 [168 gadget + 44 sync], $1989.68 \mu\text{m}^2 / 2493.3 \text{ GE}$, stage delay 718.5 ps).
- **Remark:** Certified affine equivalent to the inversion $x^{63} \sim_{\text{aff}} x^{126}$ (with $S_{\text{ref}}(u) = u^{63} \bmod (x^7 + x + 1)$, $A = [0x03, 0x62, 0x39, 0x6f, 0x0c, 0x10, 0x15]$, $a = 0x01$, $B = [0x2e, 0x5f, 0x02, 0x46, 0x38, 0x3f, 0x7f]$, $b = 0x16$); degree 6 at depth 3.
- **Inverse (S^{-1}):** $\deg(S^{-1}) = 6$, $\mathcal{D}(S^{-1}) = [6^{127}]$; cost: 42 AND, 229 XOR, 6 NOT, AND depth $D = 3$ (Tier 1: 3-stage affine-folded core reuse; unmasked NanGate 45 nm: $413.36 \mu\text{m}^2 / 518.0 \text{ GE}$, delay 2118.5 ps, logic depth 33; pipelined $d = 2$ DOM-indep: 306 DFF_X1 [168 gadget + 138 sync], $2564.77 \mu\text{m}^2 / 3214.0 \text{ GE}$, stage delay 716.5 ps).

Lookup table (LUT[0...127]):

```

0x38, 0x7e, 0x6d, 0x5b, 0x17, 0x43, 0x09, 0x1d, 0x3f, 0x23, 0x57, 0x7f, 0x1a, 0x19, 0x7c, 0x00
0x7b, 0x60, 0x22, 0x37, 0x64, 0x28, 0x2d, 0x25, 0x77, 0x49, 0x16, 0x6e, 0x01, 0x54, 0x4f, 0x0e
0x30, 0x05, 0x42, 0x5d, 0x36, 0x69, 0x62, 0x59, 0x5f, 0x2f, 0x32, 0x6c, 0x58, 0x02, 0x2c, 0x10
0x46, 0x0c, 0x50, 0x08, 0x21, 0x61, 0x1c, 0x70, 0x11, 0x2b, 0x78, 0x6b, 0x4b, 0x20, 0x7d, 0x67
0x24, 0x0f, 0x18, 0x44, 0x1f, 0x48, 0x5a, 0x76, 0x63, 0x1e, 0x66, 0x51, 0x0b, 0x71, 0x27, 0x07
0x04, 0x13, 0x35, 0x14, 0x26, 0x2a, 0x0d, 0x40, 0x3c, 0x03, 0x4c, 0x5e, 0x79, 0x68, 0x1b, 0x29
0x47, 0x15, 0x4e, 0x53, 0x39, 0x3b, 0x55, 0x3a, 0x41, 0x6f, 0x73, 0x65, 0x31, 0x5c, 0x6a, 0x72
0x45, 0x0a, 0x7a, 0x33, 0x34, 0x52, 0x4d, 0x4a, 0x3d, 0x12, 0x75, 0x06, 0x74, 0x56, 0x3e, 0x2e

```

Netlist (42 AND, 182 XOR, 6 NOT, $D = 3$):

```

a_in = in_val[6:0]; split_1 = a_in[0:0]; split_2 = a_in[1:1];
split_3 = a_in[2:2]; split_4 = a_in[3:3]; split_5 = a_in[4:4];
split_6 = a_in[5:5]; split_7 = a_in[6:6];
sync_buf_split_1_to_xor_61_stg0 = split_1;
sync_buf_split_3_to_xor_65_stg0 = split_3; xor_8 = split_1 ^ split_4;
xor_13 = split_5 ^ split_6; sync_buf_split_7_to_xor_66_stg0 = split_7;
sync_buf_split_7_to_xor_54_stg0 = split_7; xor_9 = split_6 ^ xor_8;
not_15 = ~xor_8; not_14 = ~xor_13; xor_10 = split_2 ^ xor_9;
xor_11 = split_7 ^ xor_9; xor_19 = split_1 ^ not_15; xor_28 = not_15 ^ split_7;
sync_buf_not_15_to_xor_75_stg0 = not_15; xor_17 = not_14 ^ split_7;

```

```

sync_buf_not_14_to_xor_62_stg0 = not_14; sync_buf_not_14_to_xor_84_stg0 = not_14;
xor_18 = split_1 ^ xor_10; xor_20 = xor_10 ^ not_14; xor_22 = xor_10 ^ not_15;
sync_buf_xor_10_to_xor_81_stg0 = xor_10; not_12 = ~xor_11;
xor_24 = split_3 ^ xor_19; xor_30 = xor_10 ^ xor_17; xor_32 = not_15 ^ xor_17;
xor_35 = split_1 ^ xor_17; xor_37 = split_3 ^ xor_17; xor_36 = split_7 ^ xor_18;
xor_27 = not_15 ^ xor_20; xor_16 = split_1 ^ not_12; xor_21 = split_3 ^ not_12;
xor_23 = not_12 ^ split_7; xor_33 = not_12 ^ not_14; and_38 = xor_10 & not_12;
sync_buf_not_12_to_xor_56_stg0 = not_12; and_43 = xor_18 & xor_24;
and_51 = xor_36 & xor_37; and_46 = xor_27 & xor_28; xor_25 = split_7 ^ xor_16;
xor_26 = not_15 ^ xor_16; xor_29 = split_3 ^ xor_16; and_41 = xor_16 & xor_22;
xor_31 = xor_10 ^ xor_21; and_39 = xor_21 & not_14; and_40 = xor_18 & xor_21;
and_42 = xor_20 & xor_23; xor_34 = not_15 ^ xor_33;
xor_61 = sync_buf_split_1_to_xor_61_stg0 ^ and_51; and_44 = xor_19 & xor_25;
and_45 = xor_26 & xor_23; and_47 = xor_29 & xor_30; and_48 = xor_31 & xor_25;
and_49 = xor_31 & xor_32; xor_57 = and_40 ^ and_51;
xor_62 = sync_buf_not_14_to_xor_62_stg0 ^ and_42;
xor_66 = sync_buf_split_7_to_xor_66_stg0 ^ and_42; and_50 = xor_34 & xor_35;
xor_54 = sync_buf_split_7_to_xor_54_stg0 ^ and_44; xor_52 = and_46 ^ and_47;
xor_58 = and_43 ^ and_48; xor_78 = and_48 ^ xor_61; xor_55 = and_38 ^ and_49;
xor_84 = sync_buf_not_14_to_xor_84_stg0 ^ and_49;
xor_75 = sync_buf_not_15_to_xor_75_stg0 ^ and_50; xor_59 = and_39 ^ xor_54;
xor_81 = sync_buf_xor_10_to_xor_81_stg0 ^ xor_54; xor_53 = and_41 ^ xor_52;
xor_69 = xor_52 ^ xor_61; xor_63 = xor_57 ^ xor_58; xor_72 = and_50 ^ xor_55;
xor_85 = xor_57 ^ xor_84; xor_76 = xor_62 ^ xor_75; xor_64 = and_45 ^ xor_59;
xor_79 = xor_59 ^ xor_78; xor_56 = sync_buf_not_12_to_xor_56_stg0 ^ xor_53;
xor_70 = xor_62 ^ xor_69; xor_67 = xor_66 ^ xor_63; xor_82 = xor_63 ^ xor_81;
xor_73 = xor_58 ^ xor_72; xor_86 = xor_53 ^ xor_85; xor_60 = xor_55 ^ xor_56;
xor_71 = xor_64 ^ xor_70; xor_83 = xor_56 ^ xor_82; xor_74 = xor_64 ^ xor_73;
sync_buf_xor_86_to_xor_137_stg0 = xor_86;
sync_buf_xor_86_to_xor_125_stg0 = xor_86;
xor_65 = sync_buf_split_3_to_xor_65_stg0 ^ xor_60; xor_68 = xor_67 ^ xor_60;
sync_buf_xor_71_to_xor_152_stg0 = xor_71; xor_93 = xor_71 ^ xor_83;
xor_99 = xor_83 ^ xor_86; sync_buf_xor_83_to_xor_146_stg0 = xor_83;
sync_buf_xor_74_to_xor_136_stg0 = xor_74; xor_77 = xor_76 ^ xor_65;
xor_80 = xor_79 ^ xor_65; xor_89 = xor_68 ^ xor_71; xor_90 = xor_68 ^ xor_83;
sync_buf_xor_68_to_xor_132_stg0 = xor_68; xor_87 = xor_68 ^ xor_77;
xor_92 = xor_74 ^ xor_77; xor_94 = xor_77 ^ xor_86; and_109 = xor_71 & xor_77;
sync_buf_xor_77_to_xor_127_stg0 = xor_77; xor_88 = xor_80 ^ xor_86;
xor_91 = xor_71 ^ xor_80; xor_104 = xor_77 ^ xor_80;
sync_buf_xor_80_to_xor_133_stg0 = xor_80;
sync_buf_xor_80_to_xor_155_stg0 = xor_80; xor_107 = xor_86 ^ xor_89;
xor_95 = xor_74 ^ xor_90; xor_96 = xor_86 ^ xor_87; xor_97 = xor_83 ^ xor_87;
xor_100 = xor_74 ^ xor_87; and_112 = xor_87 & xor_93; xor_102 = xor_71 ^ xor_92;
and_110 = xor_92 & xor_80; and_111 = xor_89 & xor_92; xor_101 = xor_71 ^ xor_88;
xor_103 = xor_83 ^ xor_88; xor_106 = xor_68 ^ xor_88; xor_108 = xor_74 ^ xor_88;
xor_98 = xor_83 ^ xor_91; and_113 = xor_91 & xor_94; xor_105 = xor_83 ^ xor_104;
and_114 = xor_89 & xor_95; and_115 = xor_90 & xor_96; and_116 = xor_97 & xor_94;
and_119 = xor_102 & xor_96; and_118 = xor_100 & xor_101;
and_120 = xor_102 & xor_103; and_122 = xor_107 & xor_108;
and_117 = xor_98 & xor_99; xor_133 = sync_buf_xor_80_to_xor_133_stg0 ^ and_113;
xor_137 = sync_buf_xor_86_to_xor_137_stg0 ^ and_113; and_121 = xor_105 & xor_106;
xor_125 = sync_buf_xor_86_to_xor_125_stg0 ^ and_115; xor_129 = and_114 ^ and_119;
xor_126 = and_109 ^ and_120; xor_155 = sync_buf_xor_80_to_xor_155_stg0 ^ and_120;
xor_128 = and_111 ^ and_122; xor_132 = sync_buf_xor_68_to_xor_132_stg0 ^ and_122;
xor_123 = and_117 ^ and_118; xor_146 = sync_buf_xor_83_to_xor_146_stg0 ^ and_121;
xor_130 = and_110 ^ xor_125; xor_152 = sync_buf_xor_71_to_xor_152_stg0 ^ xor_125;
xor_143 = and_121 ^ xor_126; xor_134 = xor_128 ^ xor_129;
xor_156 = xor_128 ^ xor_155; xor_149 = and_119 ^ xor_132;
xor_124 = and_112 ^ xor_123; xor_140 = xor_123 ^ xor_132;
xor_147 = xor_133 ^ xor_146; xor_135 = and_116 ^ xor_130;
xor_144 = xor_129 ^ xor_143; xor_138 = xor_137 ^ xor_134;
xor_153 = xor_134 ^ xor_152; xor_150 = xor_130 ^ xor_149;
xor_127 = sync_buf_xor_77_to_xor_127_stg0 ^ xor_124; xor_157 = xor_124 ^ xor_156;
xor_141 = xor_133 ^ xor_140; xor_145 = xor_135 ^ xor_144;
xor_131 = xor_126 ^ xor_127; xor_154 = xor_127 ^ xor_153;
sync_buf_xor_157_to_xor_209_stg0 = xor_157;
sync_buf_xor_157_to_xor_196_stg0 = xor_157; xor_142 = xor_135 ^ xor_141;
sync_buf_xor_145_to_xor_223_stg0 = xor_145;
xor_136 = sync_buf_xor_74_to_xor_136_stg0 ^ xor_131; xor_139 = xor_138 ^ xor_131;
xor_170 = xor_154 ^ xor_157; sync_buf_xor_154_to_xor_203_stg0 = xor_154;
xor_164 = xor_142 ^ xor_154; sync_buf_xor_142_to_xor_204_stg0 = xor_142;
xor_148 = xor_147 ^ xor_136; xor_151 = xor_150 ^ xor_136;
xor_160 = xor_139 ^ xor_142; xor_161 = xor_139 ^ xor_154;
sync_buf_xor_139_to_xor_201_stg0 = xor_139; xor_158 = xor_139 ^ xor_148;
xor_163 = xor_145 ^ xor_148; xor_165 = xor_148 ^ xor_157;
and_180 = xor_142 & xor_148; sync_buf_xor_148_to_xor_199_stg0 = xor_148;
xor_159 = xor_151 ^ xor_157; xor_162 = xor_142 ^ xor_151;
paar_shared_xor_0_xor_148_xor_151 = xor_148 ^ xor_151;

```

```

sync_buf_xor_151_to_xor_195_stg0 = xor_151; xor_178 = xor_157 ^ xor_160;
xor_166 = xor_145 ^ xor_161; xor_167 = xor_157 ^ xor_158;
xor_168 = xor_154 ^ xor_158; xor_171 = xor_145 ^ xor_158;
and_183 = xor_158 & xor_164; xor_173 = xor_142 ^ xor_163;
and_181 = xor_163 & xor_151; and_182 = xor_160 & xor_163;
xor_172 = xor_142 ^ xor_159; xor_174 = xor_154 ^ xor_159;
xor_177 = xor_139 ^ xor_159; xor_179 = xor_145 ^ xor_159;
xor_169 = xor_154 ^ xor_162; and_184 = xor_162 & xor_165;
and_185 = xor_160 & xor_166; and_186 = xor_161 & xor_167;
and_187 = xor_168 & xor_165;
xor_199 = sync_buf_xor_148_to_xor_199_stg0 ^ and_183;
and_190 = xor_173 & xor_167; and_189 = xor_171 & xor_172;
and_191 = xor_173 & xor_174; and_193 = xor_178 & xor_179;
and_188 = xor_169 & xor_170;
xor_195 = sync_buf_xor_151_to_xor_195_stg0 ^ and_184;
xor_209 = sync_buf_xor_157_to_xor_209_stg0 ^ and_184;
xor_176 = xor_154 ^ paar_shared_xor_0_xor_148_xor_151;
sync_buf_xor_175_to_xor_218_stg0 = paar_shared_xor_0_xor_148_xor_151;
xor_196 = sync_buf_xor_157_to_xor_196_stg0 ^ and_186;
xor_206 = and_182 ^ xor_199; xor_197 = and_185 ^ and_190;
xor_200 = and_180 ^ and_191; xor_214 = and_190 ^ and_193;
xor_194 = and_188 ^ and_189;
xor_201 = sync_buf_xor_139_to_xor_201_stg0 ^ xor_195;
xor_224 = xor_195 ^ xor_199; and_192 = xor_176 & xor_177;
xor_218 = sync_buf_xor_175_to_xor_218_stg0 ^ and_191;
xor_204 = sync_buf_xor_142_to_xor_204_stg0 ^ xor_196;
xor_205 = and_181 ^ xor_196; xor_210 = xor_209 ^ xor_206;
xor_198 = and_193 ^ xor_194;
xor_223 = sync_buf_xor_145_to_xor_223_stg0 ^ xor_194;
xor_203 = sync_buf_xor_154_to_xor_203_stg0 ^ and_192;
xor_212 = and_192 ^ xor_201; xor_219 = xor_197 ^ xor_204;
xor_228 = xor_206 ^ xor_204; xor_207 = xor_201 ^ xor_205;
xor_202 = xor_197 ^ xor_198; xor_221 = and_187 ^ xor_198;
xor_226 = xor_223 ^ xor_224; xor_215 = xor_203 ^ xor_214;
xor_225 = xor_200 ^ xor_203; xor_220 = xor_218 ^ xor_219;
xor_208 = xor_200 ^ xor_202; xor_229 = xor_202 ^ xor_228;
xor_222 = xor_207 ^ xor_221; xor_216 = xor_215 ^ xor_207;
xor_227 = xor_225 ^ xor_226; xor_211 = xor_210 ^ xor_208;
xor_213 = xor_212 ^ xor_208; sync_buf_xor_229_to_concat_240_stg0 = xor_229;
not_231 = ~xor_216; xor_233 = xor_216 ^ xor_222; xor_232 = xor_220 ^ xor_227;
xor_236 = xor_211 ^ xor_227; sync_buf_xor_211_to_xor_239_stg0 = xor_211;
not_230 = ~xor_213; sync_buf_not_231_to_concat_240_stg0 = not_231;
xor_234 = xor_227 ^ xor_233; sync_buf_xor_232_to_concat_240_stg0 = xor_232;
sync_buf_xor_236_to_concat_240_stg0 = xor_236; xor_238 = not_230 ^ not_231;
sync_buf_not_230_to_concat_240_stg0 = not_230; not_235 = ~xor_234;
xor_239 = sync_buf_xor_211_to_xor_239_stg0 ^ xor_238;
sync_buf_not_235_to_concat_240_stg0 = not_235;
concat_240 = { sync_buf_xor_229_to_concat_240_stg0,
  sync_buf_xor_236_to_concat_240_stg0, sync_buf_not_235_to_concat_240_stg0,
  sync_buf_xor_232_to_concat_240_stg0, sync_buf_not_231_to_concat_240_stg0,
  sync_buf_not_230_to_concat_240_stg0, xor_239 };
out_val = concat_240;

```

Inverse lookup table ($LUT_{S^{-1}}[0 \dots 127]$):

```

0x0f, 0x1c, 0x2d, 0x59, 0x50, 0x21, 0x7b, 0x4f, 0x33, 0x06, 0x71, 0x4c, 0x31, 0x56, 0x1f, 0x41
0x2f, 0x38, 0x79, 0x51, 0x53, 0x61, 0x1a, 0x04, 0x42, 0x0d, 0x0c, 0x5e, 0x36, 0x07, 0x49, 0x44
0x3d, 0x34, 0x12, 0x09, 0x40, 0x17, 0x54, 0x4e, 0x15, 0x5f, 0x55, 0x39, 0x2e, 0x16, 0x7f, 0x29
0x20, 0x6c, 0x2a, 0x73, 0x74, 0x52, 0x24, 0x13, 0x00, 0x64, 0x67, 0x65, 0x58, 0x78, 0x7e, 0x08
0x57, 0x68, 0x22, 0x05, 0x43, 0x70, 0x30, 0x60, 0x45, 0x19, 0x77, 0x3c, 0x5a, 0x76, 0x62, 0x1e
0x32, 0x4b, 0x75, 0x63, 0x1d, 0x66, 0x7d, 0x0a, 0x2c, 0x27, 0x46, 0x03, 0x6d, 0x23, 0x5b, 0x28
0x11, 0x35, 0x26, 0x48, 0x14, 0x6b, 0x4a, 0x3f, 0x5d, 0x25, 0x6e, 0x3b, 0x2b, 0x02, 0x1b, 0x69
0x37, 0x4d, 0x6f, 0x6a, 0x7c, 0x7a, 0x47, 0x18, 0x3a, 0x5c, 0x72, 0x10, 0x0e, 0x3e, 0x01, 0x0b

```

Inverse netlist (42 AND, 229 XOR, 6 NOT, $D = 3$):

```

not_0 = ~x[1]; not_1 = ~x[2]; not_2 = ~x[4]; xor_0 = not_0 ^ x[5];
xor_1 = not_2 ^ x[6]; xor_2 = not_1 ^ x[0]; xor_3 = xor_0 ^ x[3];
xor_4 = xor_1 ^ x[0]; xor_5 = not_1 ^ not_2; xor_6 = xor_3 ^ not_1;
xor_7 = not_0 ^ x[6]; xor_8 = xor_2 ^ xor_3; xor_9 = xor_1 ^ x[5];
xor_10 = xor_3 ^ x[0]; xor_11 = xor_2 ^ xor_7; xor_12 = xor_6 ^ x[6];
xor_13 = xor_5 ^ x[3]; xor_14 = xor_0 ^ xor_4; xor_15 = not_2 ^ x[3];
xor_16 = xor_4 ^ not_0; xor_17 = xor_2 ^ not_0; xor_18 = xor_0 ^ xor_5;
xor_19 = xor_4 ^ x[5]; xor_20 = not_0 ^ not_1; xor_21 = xor_1 ^ xor_20;
xor_22 = xor_8 ^ x[6]; xor_23 = xor_9 ^ x[3]; xor_24 = xor_0 ^ xor_1;
xor_25 = xor_2 ^ xor_24; xor_26 = x[0] ^ x[6]; xor_27 = not_2 ^ x[0];
xor_28 = xor_0 ^ xor_27; and_0 = xor_10 & xor_11; and_1 = xor_0 & xor_6;
and_2 = xor_1 & xor_12; and_3 = xor_13 & xor_14; and_4 = xor_15 & xor_7;
and_5 = xor_16 & not_1; and_6 = xor_16 & xor_6; and_7 = xor_17 & xor_8;

```

```

xor_29 = and_2 ^ xor_2; and_8 = xor_18 & not_0; and_9 = xor_17 & xor_19;
and_10 = xor_21 & xor_22; and_11 = xor_23 & not_0; and_12 = xor_23 & xor_25;
xor_30 = and_6 ^ and_2; xor_31 = and_7 ^ not_1; xor_32 = and_7 ^ x[6];
and_13 = xor_26 & xor_9; xor_33 = and_8 ^ x[6]; xor_34 = and_3 ^ and_10;
xor_35 = and_1 ^ and_11; xor_36 = and_11 ^ xor_29; xor_37 = and_0 ^ and_12;
xor_38 = and_12 ^ not_1; xor_39 = and_13 ^ xor_28; xor_40 = and_5 ^ xor_33;
xor_41 = xor_10 ^ xor_33; xor_42 = and_4 ^ xor_34; xor_43 = xor_34 ^ xor_29;
xor_44 = xor_30 ^ xor_35; xor_45 = and_13 ^ xor_37; xor_46 = xor_30 ^ xor_38;
xor_47 = xor_31 ^ xor_39; xor_48 = and_9 ^ xor_40; xor_49 = xor_40 ^ xor_36;
xor_50 = xor_11 ^ xor_42; xor_51 = xor_31 ^ xor_43; xor_52 = xor_44 ^ xor_32;
xor_53 = xor_44 ^ xor_41; xor_54 = xor_35 ^ xor_45; xor_55 = xor_42 ^ xor_46;
xor_56 = xor_37 ^ xor_50; xor_57 = xor_48 ^ xor_51; xor_58 = xor_50 ^ xor_53;
xor_59 = xor_48 ^ xor_54; xor_60 = xor_5 ^ xor_56; xor_61 = xor_56 ^ xor_52;
xor_62 = xor_57 ^ xor_58; xor_63 = xor_58 ^ xor_55; xor_64 = xor_60 ^ xor_47;
xor_65 = xor_60 ^ xor_49; xor_66 = xor_61 ^ xor_57; xor_67 = xor_61 ^ xor_58;
xor_68 = xor_61 ^ xor_64; xor_69 = xor_59 ^ xor_64; xor_70 = xor_64 ^ xor_55;
and_14 = xor_57 & xor_64; xor_71 = xor_65 ^ xor_55; xor_72 = xor_57 ^ xor_65;
xor_73 = xor_64 ^ xor_65; xor_74 = xor_55 ^ xor_66; xor_75 = xor_59 ^ xor_67;
xor_76 = xor_55 ^ xor_68; xor_77 = xor_58 ^ xor_68; xor_78 = xor_59 ^ xor_68;
and_15 = xor_68 & xor_62; xor_79 = xor_57 ^ xor_69; and_16 = xor_65 & xor_69;
and_17 = xor_66 & xor_69; xor_80 = xor_57 ^ xor_71; xor_81 = xor_58 ^ xor_71;
xor_82 = xor_61 ^ xor_71; xor_83 = xor_59 ^ xor_71; xor_84 = xor_58 ^ xor_72;
and_18 = xor_72 & xor_70; xor_85 = xor_73 ^ xor_58; and_19 = xor_66 & xor_75;
and_20 = xor_67 & xor_76; and_21 = xor_70 & xor_77; and_22 = xor_79 & xor_76;
and_23 = xor_78 & xor_80; and_24 = xor_79 & xor_81; and_25 = xor_74 & xor_83;
and_26 = xor_84 & xor_63; xor_86 = and_18 ^ xor_65; xor_87 = and_18 ^ xor_55;
and_27 = xor_85 & xor_82; xor_88 = and_20 ^ xor_55; xor_89 = and_19 ^ and_22;
xor_90 = and_14 ^ and_24; xor_91 = and_24 ^ xor_65; xor_92 = and_17 ^ and_25;
xor_93 = and_25 ^ xor_61; xor_94 = and_26 ^ and_23; xor_95 = and_27 ^ xor_58;
xor_96 = and_16 ^ xor_88; xor_97 = xor_88 ^ xor_57; xor_98 = and_27 ^ xor_90;
xor_99 = xor_92 ^ xor_89; xor_100 = xor_92 ^ xor_91; xor_101 = and_22 ^ xor_93;
xor_102 = and_15 ^ xor_94; xor_103 = xor_94 ^ xor_93;
xor_104 = xor_86 ^ xor_95; xor_105 = and_21 ^ xor_96;
xor_106 = xor_89 ^ xor_98; xor_107 = xor_99 ^ xor_87;
xor_108 = xor_99 ^ xor_97; xor_109 = xor_96 ^ xor_101;
xor_110 = xor_102 ^ xor_64; xor_111 = xor_102 ^ xor_100;
xor_112 = xor_86 ^ xor_103; xor_113 = xor_105 ^ xor_106;
xor_114 = xor_90 ^ xor_110; xor_115 = xor_110 ^ xor_108;
xor_116 = xor_105 ^ xor_112; xor_117 = xor_114 ^ xor_59;
xor_118 = xor_114 ^ xor_107; xor_119 = xor_115 ^ xor_111;
xor_120 = xor_116 ^ xor_115; xor_121 = xor_117 ^ xor_104;
xor_122 = xor_117 ^ xor_109; xor_123 = xor_118 ^ xor_116;
xor_124 = xor_118 ^ xor_115; xor_125 = xor_118 ^ xor_121;
xor_126 = xor_113 ^ xor_121; xor_127 = xor_121 ^ xor_111;
and_28 = xor_116 & xor_121; xor_128 = xor_122 ^ xor_111;
xor_129 = xor_116 ^ xor_122; xor_130 = xor_121 ^ xor_122;
xor_131 = xor_111 ^ xor_123; xor_132 = xor_113 ^ xor_124;
xor_133 = xor_111 ^ xor_125; xor_134 = xor_115 ^ xor_125;
xor_135 = xor_113 ^ xor_125; and_29 = xor_125 & xor_120;
xor_136 = xor_116 ^ xor_126; and_30 = xor_122 & xor_126;
and_31 = xor_123 & xor_126; xor_137 = xor_116 ^ xor_128;
xor_138 = xor_115 ^ xor_128; xor_139 = xor_118 ^ xor_128;
xor_140 = xor_113 ^ xor_128; xor_141 = xor_115 ^ xor_129;
and_32 = xor_129 & xor_127; and_33 = xor_123 & xor_132;
and_34 = xor_124 & xor_133; and_35 = xor_127 & xor_134;
and_36 = xor_133 & xor_136; and_37 = xor_135 & xor_137;
and_38 = xor_136 & xor_138; and_39 = xor_131 & xor_140;
and_40 = xor_141 & xor_119; xor_142 = xor_130 ^ xor_115;
and_41 = xor_142 & xor_139; xor_143 = and_13 ^ not_2;
xor_144 = and_24 ^ and_38; xor_145 = and_28 ^ and_2; xor_146 = and_18 ^ and_7;
xor_147 = and_32 ^ xor_146; xor_148 = and_11 ^ xor_143;
xor_149 = xor_145 ^ xor_148; xor_150 = and_36 ^ x[6];
xor_151 = and_20 ^ xor_144; xor_152 = and_17 ^ x[3]; xor_153 = and_27 ^ and_5;
xor_154 = and_37 ^ and_9; xor_155 = and_14 ^ and_29;
xor_156 = and_40 ^ xor_154; xor_157 = xor_149 ^ xor_151;
xor_158 = and_35 ^ xor_150; xor_159 = and_4 ^ not_1; xor_160 = and_16 ^ and_31;
xor_161 = and_22 ^ xor_159; xor_162 = and_8 ^ not_0; xor_163 = and_30 ^ and_12;
xor_164 = and_21 ^ xor_147; xor_165 = and_33 ^ and_34;
xor_166 = xor_163 ^ xor_165; xor_167 = and_39 ^ xor_156;
xor_168 = xor_155 ^ xor_152; xor_169 = xor_153 ^ xor_157;
xor_170 = xor_158 ^ xor_160; xor_171 = xor_169 ^ xor_170;
xor_172 = and_15 ^ and_0; xor_173 = and_41 ^ xor_164;
xor_174 = xor_168 ^ xor_94; xor_175 = xor_172 ^ xor_174;
xor_176 = xor_162 ^ x[5]; xor_177 = and_19 ^ xor_166;
xor_178 = and_6 ^ xor_161; xor_179 = and_25 ^ xor_167;
xor_180 = xor_171 ^ x[0]; xor_181 = xor_175 ^ xor_180;
xor_182 = xor_177 ^ xor_181; xor_183 = xor_171 ^ xor_173;
xor_184 = xor_176 ^ xor_183; not_3 = ~xor_184; xor_185 = and_8 ^ xor_156;

```

```

xor_186 = xor_157 ^ xor_185; xor_187 = xor_173 ^ xor_186;
xor_188 = xor_175 ^ xor_187; xor_189 = xor_178 ^ xor_188;
xor_190 = and_27 ^ xor_149; xor_191 = xor_190 ^ xor_152;
xor_192 = xor_158 ^ xor_191; xor_193 = xor_161 ^ xor_192;
xor_194 = xor_176 ^ xor_193; xor_195 = xor_177 ^ xor_194;
xor_196 = xor_195 ^ xor_34; xor_197 = xor_179 ^ xor_196; not_4 = ~xor_197;
xor_198 = not_0 ^ x[3]; xor_199 = and_1 ^ xor_198; xor_200 = and_5 ^ xor_199;
xor_201 = and_30 ^ xor_200; xor_202 = xor_143 ^ xor_201;
xor_203 = xor_147 ^ xor_202; xor_204 = xor_203 ^ xor_150;
xor_205 = xor_204 ^ xor_151; xor_206 = xor_160 ^ xor_205;
xor_207 = xor_206 ^ xor_94; xor_208 = xor_207 ^ xor_34;
xor_209 = xor_178 ^ xor_208; not_5 = ~xor_209; xor_210 = and_13 ^ xor_26;
xor_211 = and_17 ^ xor_210; xor_212 = and_22 ^ xor_211;
xor_213 = xor_144 ^ xor_212; xor_214 = xor_153 ^ xor_213;
xor_215 = xor_155 ^ xor_214; xor_216 = xor_162 ^ xor_215;
xor_217 = xor_164 ^ xor_216; xor_218 = xor_166 ^ xor_217;
xor_219 = xor_167 ^ xor_218; xor_220 = and_31 ^ and_6;
xor_221 = and_33 ^ xor_220; xor_222 = and_36 ^ xor_221;
xor_223 = and_38 ^ xor_222; xor_224 = xor_145 ^ xor_223;
xor_225 = xor_147 ^ xor_224; xor_226 = xor_153 ^ xor_225;
xor_227 = xor_168 ^ xor_226; xor_228 = xor_179 ^ xor_227; y[0] = xor_182;
y[1] = not_3; y[2] = xor_189; y[3] = not_4; y[4] = not_5; y[5] = xor_219;
y[6] = xor_228;

```

C.25 7-bit power map x^7 (7-bit, Category A)

- **Profile:** $\delta = 6$, $L = 40$ (NL = 44), $\deg(S) = 3$, $\mathcal{D}(S) = [3^{127}]$, FP = 2, SAC deviation 0.125.
- **Cost:** 56 AND, 98 XOR, 0 NOT, AND depth $D = 2$ (unmasked NanGate 45 nm: $215.99 \mu\text{m}^2$ / 270.7 GE, delay 772.4 ps, logic depth 14; pipelined $d = 2$ DOM-indep: 280 DFF_X1 [224 gadget + 56 sync], $2174.82 \mu\text{m}^2$ / 2725.3 GE, stage delay 765.9 ps).
- **Remark:** The exact lookup table of x^7 over $\mathbb{F}_2[x]/(x^7+x+1)$; 30/140/2 in [BDM26]: 26 more AND gates and 42 fewer XOR gates.
- **Inverse (S^{-1}):** $\deg(S^{-1}) = 5$, $\mathcal{D}(S^{-1}) = [5^{127}]$; cost: 51 AND, 220 XOR, 15 NOT, AND depth $D = 3$ (Tier 3: 3-stage power-map factorization $y^{109} = ((y^5)^5)^{64}(y^5)^{32}$ at $D = 3$; unmasked NanGate 45 nm: $413.36 \mu\text{m}^2$ / 518.0 GE, delay 1904.0 ps, logic depth 33; pipelined $d = 2$ DOM-indep: 284 DFF_X1 [204 gadget + 80 sync], $2537.11 \mu\text{m}^2$ / 3179.3 GE, stage delay 976.9 ps).

Lookup table (LUT[0...127]):

```

0x00, 0x01, 0x03, 0x7c, 0x05, 0x2e, 0x07, 0x30, 0x0f, 0x70, 0x72, 0x0d, 0x09, 0x7d, 0x50, 0x46
0x11, 0x4c, 0x13, 0x62, 0x15, 0x1e, 0x17, 0x52, 0x1b, 0x7f, 0x04, 0x32, 0x73, 0x60, 0x49, 0x6a
0x33, 0x71, 0x57, 0x40, 0x35, 0x44, 0x25, 0x63, 0x3f, 0x1f, 0x22, 0x29, 0x39, 0x0b, 0x75, 0x0e
0x2d, 0x76, 0x02, 0x5e, 0x0c, 0x18, 0x56, 0x27, 0x16, 0x68, 0x23, 0x24, 0x58, 0x48, 0x3d, 0x36
0x55, 0x7e, 0x10, 0x1a, 0x7a, 0x51, 0x43, 0x2b, 0x5f, 0x45, 0x4f, 0x0a, 0x6f, 0x54, 0x26, 0x20
0x41, 0x2f, 0x21, 0x3c, 0x66, 0x74, 0x7b, 0x78, 0x4b, 0x53, 0x1d, 0x08, 0x1c, 0x59, 0x12, 0x38
0x77, 0x2a, 0x19, 0x4e, 0x06, 0x42, 0x61, 0x4d, 0x14, 0x64, 0x28, 0x2c, 0x79, 0x31, 0x69, 0x37
0x3a, 0x67, 0x3b, 0x3e, 0x65, 0x5d, 0x6c, 0x6e, 0x6b, 0x5c, 0x5b, 0x4a, 0x47, 0x34, 0x5a, 0x6d

```

Netlist (56 AND, 98 XOR, 0 NOT, $D = 2$):

```

xor_0 = x[0] ^ x[5]; and_0 = x[0] & x[1]; and_1 = x[0] & x[2];
and_2 = x[1] & x[2]; and_3 = x[0] & x[3]; and_4 = x[1] & x[3];
and_5 = x[2] & x[3]; and_6 = x[0] & x[4]; and_7 = x[1] & x[4];
and_8 = x[2] & x[4]; and_9 = x[3] & x[4]; and_10 = x[0] & x[5];
and_11 = x[1] & x[5]; and_12 = x[2] & x[5]; and_13 = x[3] & x[5];
and_14 = x[4] & x[5]; and_15 = x[0] & x[6]; and_16 = x[1] & x[6];
and_17 = x[2] & x[6]; and_18 = x[3] & x[6]; and_19 = x[4] & x[6];
and_20 = x[5] & x[6]; and_21 = x[0] & and_2; and_22 = x[0] & and_4;
and_23 = x[0] & and_5; and_24 = x[1] & and_5; and_25 = x[0] & and_7;
and_26 = x[0] & and_8; and_27 = x[1] & and_8; and_28 = x[0] & and_9;
and_29 = x[1] & and_9; and_30 = x[2] & and_9; and_31 = x[0] & and_11;
and_32 = x[0] & and_12; and_33 = x[1] & and_12; and_34 = x[0] & and_13;
and_35 = x[1] & and_13; and_36 = x[2] & and_13; and_37 = x[0] & and_14;
and_38 = x[1] & and_14; and_39 = x[2] & and_14; and_40 = x[3] & and_14;
and_41 = x[0] & and_16; and_42 = x[0] & and_17; and_43 = x[1] & and_17;
and_44 = x[0] & and_18; and_45 = x[1] & and_18; and_46 = x[2] & and_18;

```

```

and_47 = x[0] & and_19; and_48 = x[1] & and_19; and_49 = x[2] & and_19;
and_50 = x[3] & and_19; and_51 = x[0] & and_20; and_52 = x[1] & and_20;
and_53 = x[2] & and_20; and_54 = x[3] & and_20; and_55 = x[4] & and_20;
xor_1 = and_0 ^ and_4; xor_2 = and_3 ^ and_22; xor_3 = xor_1 ^ xor_2;
xor_4 = and_43 ^ xor_3; xor_5 = and_41 ^ and_53; xor_6 = and_26 ^ xor_4;
xor_7 = and_30 ^ and_44; xor_8 = and_31 ^ and_45; xor_9 = and_6 ^ xor_6;
xor_10 = and_11 ^ xor_7; xor_11 = xor_5 ^ xor_9; xor_12 = x[6] ^ and_33;
xor_13 = and_28 ^ xor_10; xor_14 = x[3] ^ and_14; xor_15 = and_46 ^ xor_11;
xor_16 = and_42 ^ and_54; xor_17 = and_1 ^ xor_8; xor_18 = xor_16 ^ xor_17;
xor_19 = and_37 ^ xor_12; xor_20 = and_32 ^ and_47; xor_21 = and_10 ^ and_29;
xor_22 = and_25 ^ and_55; xor_23 = xor_13 ^ xor_19; xor_24 = and_35 ^ xor_14;
xor_25 = and_15 ^ and_17; xor_26 = xor_18 ^ xor_25; xor_27 = and_24 ^ and_51;
xor_28 = x[5] ^ and_40; xor_29 = and_34 ^ xor_15; xor_30 = and_39 ^ xor_26;
xor_31 = and_12 ^ xor_24; xor_32 = and_48 ^ xor_27; xor_33 = x[2] ^ and_9;
xor_34 = xor_23 ^ xor_33; xor_35 = and_13 ^ xor_21; xor_36 = and_36 ^ xor_35;
xor_37 = and_21 ^ xor_30; xor_38 = and_5 ^ xor_31; xor_39 = x[1] ^ xor_38;
xor_40 = and_16 ^ xor_22; xor_41 = and_52 ^ xor_29; xor_42 = and_50 ^ xor_34;
xor_43 = and_19 ^ xor_42; xor_44 = and_18 ^ xor_43; xor_45 = x[4] ^ and_20;
xor_46 = xor_36 ^ xor_39; xor_47 = xor_20 ^ xor_45; xor_48 = and_38 ^ xor_40;
xor_49 = xor_28 ^ xor_37; xor_50 = xor_0 ^ and_2; xor_51 = and_7 ^ and_8;
xor_52 = xor_50 ^ and_17; xor_53 = xor_52 ^ and_23; xor_54 = and_27 ^ and_43;
xor_55 = and_49 ^ xor_5; xor_56 = xor_8 ^ xor_27; xor_57 = xor_44 ^ xor_46;
xor_58 = xor_47 ^ xor_53; xor_59 = xor_54 ^ xor_55; xor_60 = xor_56 ^ xor_57;
xor_61 = xor_58 ^ xor_59; xor_62 = xor_60 ^ xor_61; xor_63 = and_41 ^ xor_3;
xor_64 = xor_13 ^ xor_46; xor_65 = xor_48 ^ xor_49; xor_66 = xor_63 ^ xor_64;
xor_67 = xor_65 ^ xor_66; xor_68 = and_16 ^ and_39; xor_69 = and_55 ^ xor_24;
xor_70 = xor_32 ^ xor_41; xor_71 = xor_44 ^ xor_68; xor_72 = xor_69 ^ xor_70;
xor_73 = xor_71 ^ xor_72; xor_74 = xor_14 ^ and_38; xor_75 = and_49 ^ xor_7;
xor_76 = xor_20 ^ xor_26; xor_77 = xor_41 ^ xor_74; xor_78 = xor_75 ^ xor_76;
xor_79 = xor_77 ^ xor_78; xor_80 = and_51 ^ xor_12; xor_81 = xor_22 ^ xor_28;
xor_82 = xor_29 ^ xor_47; xor_83 = xor_80 ^ xor_81; xor_84 = xor_82 ^ xor_83;
xor_85 = and_11 ^ and_23; xor_86 = and_29 ^ and_30; xor_87 = and_33 ^ xor_6;
xor_88 = xor_32 ^ xor_49; xor_89 = xor_85 ^ xor_86; xor_90 = xor_87 ^ xor_88;
xor_91 = xor_89 ^ xor_90; xor_92 = and_21 ^ and_52; xor_93 = and_54 ^ xor_11;
xor_94 = xor_21 ^ xor_23; xor_95 = xor_48 ^ xor_92; xor_96 = xor_93 ^ xor_94;
xor_97 = xor_95 ^ xor_96; y[0] = xor_62; y[1] = xor_67; y[2] = xor_73;
y[3] = xor_79; y[4] = xor_84; y[5] = xor_91; y[6] = xor_97;

```

Inverse lookup table ($\text{LUT}_{S^{-1}}[0 \dots 127]$):

```

0x00, 0x01, 0x32, 0x02, 0x1a, 0x04, 0x64, 0x06, 0x5b, 0x0c, 0x4b, 0x2d, 0x34, 0x0b, 0x2f, 0x08
0x42, 0x10, 0x5e, 0x12, 0x68, 0x14, 0x38, 0x16, 0x35, 0x62, 0x43, 0x18, 0x5c, 0x5a, 0x15, 0x29
0x4f, 0x52, 0x2a, 0x3a, 0x3b, 0x26, 0x4e, 0x37, 0x6a, 0x2b, 0x61, 0x47, 0x6b, 0x30, 0x05, 0x51
0x07, 0x6d, 0x1b, 0x20, 0x7d, 0x24, 0x3f, 0x6f, 0x5f, 0x2c, 0x70, 0x72, 0x53, 0x3e, 0x73, 0x28
0x23, 0x50, 0x65, 0x46, 0x25, 0x49, 0x0f, 0x7c, 0x3d, 0x1e, 0x7b, 0x58, 0x11, 0x67, 0x63, 0x4a
0x0e, 0x45, 0x17, 0x59, 0x4d, 0x40, 0x36, 0x22, 0x3c, 0x5d, 0x7e, 0x7a, 0x79, 0x75, 0x33, 0x48
0x1d, 0x66, 0x13, 0x27, 0x69, 0x74, 0x54, 0x71, 0x39, 0x6e, 0x1f, 0x78, 0x76, 0x7f, 0x77, 0x4c
0x09, 0x21, 0x0a, 0x1c, 0x55, 0x2e, 0x31, 0x60, 0x57, 0x6c, 0x44, 0x56, 0x03, 0x0d, 0x41, 0x19

```

Inverse netlist (51 AND, 220 XOR, 15 NOT, $D = 3$):

```

not_0 = ~x[0]; not_1 = ~x[2]; not_2 = ~x[4]; not_3 = ~x[6];
xor_0 = not_2 ^ x[3]; xor_1 = not_3 ^ x[1]; xor_2 = not_1 ^ x[5];
xor_3 = xor_0 ^ not_1; xor_4 = not_0 ^ x[5]; xor_5 = xor_4 ^ not_3;
xor_6 = xor_2 ^ not_2; xor_7 = xor_0 ^ xor_1; xor_8 = not_1 ^ not_2;
xor_9 = not_2 ^ x[5]; xor_10 = xor_1 ^ x[3]; xor_11 = xor_3 ^ not_3;
xor_12 = xor_2 ^ not_3; xor_13 = not_1 ^ x[3]; xor_14 = not_1 ^ x[1];
xor_15 = xor_0 ^ xor_5; xor_16 = xor_3 ^ x[1]; xor_17 = not_0 ^ x[3];
xor_18 = xor_6 ^ not_0; xor_19 = xor_6 ^ x[1]; xor_20 = xor_7 ^ x[5];
xor_21 = x[1] ^ x[3]; xor_22 = xor_5 ^ x[3]; xor_23 = xor_0 ^ not_0;
xor_24 = xor_2 ^ xor_23; xor_25 = xor_1 ^ not_1; xor_26 = not_0 ^ not_1;
xor_27 = xor_1 ^ xor_3; xor_28 = xor_2 ^ x[3]; xor_29 = x[3] ^ x[5];
xor_30 = not_0 ^ x[1]; and_0 = xor_8 & xor_9; and_1 = xor_10 & not_1;
and_2 = xor_0 & xor_11; and_3 = xor_12 & xor_13; and_4 = xor_14 & xor_8;
and_5 = xor_15 & xor_7; and_6 = xor_16 & xor_17; and_7 = xor_17 & xor_7;
and_8 = xor_18 & xor_19; and_9 = xor_20 & xor_21; and_10 = xor_20 & xor_22;
and_11 = xor_24 & xor_25; and_12 = xor_26 & xor_27; and_13 = xor_28 & xor_5;
xor_31 = and_0 ^ not_0; xor_32 = and_8 ^ xor_31; xor_33 = xor_29 ^ xor_32;
xor_34 = and_1 ^ not_2; xor_35 = and_3 ^ and_4; xor_36 = not_1 ^ xor_33;
xor_37 = and_12 ^ xor_34; xor_38 = and_7 ^ not_3; xor_39 = and_2 ^ and_5;
xor_40 = xor_35 ^ xor_39; xor_41 = and_11 ^ xor_36; xor_42 = and_10 ^ xor_38;
xor_43 = xor_35 ^ xor_37; xor_44 = and_7 ^ xor_30; xor_45 = xor_34 ^ xor_44;
xor_46 = xor_40 ^ xor_45; xor_47 = and_1 ^ and_9; xor_48 = and_13 ^ xor_47;
xor_49 = xor_40 ^ xor_48; xor_50 = xor_41 ^ xor_49; xor_51 = xor_42 ^ xor_50;
not_4 = ~xor_51; xor_52 = and_6 ^ xor_41; xor_53 = xor_43 ^ xor_52;
xor_54 = xor_36 ^ x[1]; xor_55 = xor_37 ^ xor_54; xor_56 = xor_42 ^ xor_55;
not_5 = ~xor_56; xor_57 = and_2 ^ xor_1; xor_58 = and_6 ^ xor_57;
xor_59 = and_11 ^ xor_58; xor_60 = and_13 ^ xor_59; xor_61 = xor_33 ^ xor_60;
xor_62 = and_9 ^ xor_32; xor_63 = xor_43 ^ xor_62; not_6 = ~xor_63;

```

```

xor_64 = and_3 ^ xor_8; xor_65 = and_5 ^ xor_64; xor_66 = and_6 ^ xor_65;
xor_67 = and_8 ^ xor_66; xor_68 = xor_38 ^ xor_67; not_7 = ~xor_68;
not_8 = ~xor_46; not_9 = ~xor_53; xor_69 = not_5 ^ xor_61;
xor_70 = not_9 ^ not_4; xor_71 = xor_68 ^ xor_69; xor_72 = not_6 ^ not_8;
xor_73 = not_9 ^ xor_61; xor_74 = not_6 ^ xor_61; xor_75 = not_4 ^ xor_71;
xor_76 = not_4 ^ not_5; xor_77 = not_9 ^ not_5; xor_78 = xor_68 ^ xor_72;
xor_79 = xor_68 ^ xor_76; xor_80 = not_9 ^ xor_71; xor_81 = not_9 ^ not_6;
xor_82 = xor_68 ^ xor_81; xor_83 = xor_71 ^ xor_72; xor_84 = xor_69 ^ xor_70;
xor_85 = not_5 ^ not_8; xor_86 = xor_72 ^ xor_73; xor_87 = xor_70 ^ xor_74;
xor_88 = not_6 ^ xor_75; xor_89 = not_5 ^ xor_78; xor_90 = not_9 ^ xor_69;
xor_91 = xor_72 ^ xor_90; xor_92 = xor_68 ^ xor_70; xor_93 = not_9 ^ not_8;
xor_94 = xor_70 ^ xor_71; xor_95 = not_6 ^ xor_77; and_14 = xor_73 & xor_74;
and_15 = not_9 & xor_79; and_16 = xor_69 & xor_80; and_17 = xor_77 & xor_82;
and_18 = xor_70 & xor_73; and_19 = xor_75 & xor_83; and_20 = xor_84 & xor_85;
and_21 = xor_75 & xor_85; and_22 = xor_86 & xor_87; and_23 = xor_76 & xor_88;
and_24 = xor_88 & xor_89; and_25 = xor_91 & xor_92; and_26 = xor_93 & xor_94;
and_27 = xor_78 & xor_95; xor_96 = and_14 ^ not_8; xor_97 = and_22 ^ xor_96;
xor_98 = not_5 ^ not_6; xor_99 = xor_97 ^ xor_98; xor_100 = and_15 ^ xor_61;
xor_101 = and_17 ^ and_18; xor_102 = not_9 ^ xor_99;
xor_103 = and_26 ^ xor_100; xor_104 = and_21 ^ xor_68;
xor_105 = and_16 ^ and_19; xor_106 = xor_101 ^ xor_105;
xor_107 = and_25 ^ xor_102; xor_108 = and_24 ^ xor_104;
xor_109 = xor_101 ^ xor_103; xor_110 = not_4 ^ not_8;
xor_111 = and_21 ^ xor_110; xor_112 = xor_111 ^ xor_100;
xor_113 = xor_106 ^ xor_112; xor_114 = and_15 ^ and_23;
xor_115 = and_27 ^ xor_114; xor_116 = xor_106 ^ xor_115;
xor_117 = xor_107 ^ xor_116; xor_118 = xor_108 ^ xor_117; not_10 = ~xor_118;
xor_119 = and_20 ^ xor_107; xor_120 = xor_109 ^ xor_119;
xor_121 = not_4 ^ xor_102; xor_122 = xor_103 ^ xor_121;
xor_123 = xor_108 ^ xor_122; not_11 = ~xor_123; xor_124 = not_4 ^ xor_68;
xor_125 = and_16 ^ xor_124; xor_126 = and_20 ^ xor_125;
xor_127 = and_25 ^ xor_126; xor_128 = and_27 ^ xor_127;
xor_129 = xor_128 ^ xor_99; not_12 = ~xor_129; xor_130 = and_23 ^ xor_97;
xor_131 = xor_109 ^ xor_130; not_13 = ~xor_131; xor_132 = and_17 ^ xor_73;
xor_133 = and_19 ^ xor_132; xor_134 = and_20 ^ xor_133;
xor_135 = and_22 ^ xor_134; xor_136 = xor_104 ^ xor_135; not_14 = ~xor_136;
xor_137 = not_4 ^ xor_53; xor_138 = not_6 ^ not_7; xor_139 = not_4 ^ xor_138;
xor_140 = not_10 ^ xor_120; xor_141 = not_11 ^ not_12;
xor_142 = not_13 ^ not_14; and_28 = xor_113 & xor_46;
and_29 = xor_84 & xor_140; xor_143 = xor_113 ^ xor_140;
xor_144 = xor_84 ^ xor_46; and_30 = xor_143 & xor_144;
and_31 = xor_139 & xor_141; and_32 = xor_142 & xor_98;
xor_145 = xor_141 ^ xor_142; xor_146 = xor_139 ^ xor_98;
and_33 = xor_145 & xor_146; xor_147 = xor_113 ^ xor_141;
xor_148 = xor_140 ^ xor_142; xor_149 = xor_139 ^ xor_46;
and_34 = xor_147 & xor_149; and_35 = xor_148 & xor_87;
xor_150 = xor_147 ^ xor_148; xor_151 = xor_149 ^ xor_87;
and_36 = xor_150 & xor_151; and_37 = not_10 & xor_137;
and_38 = not_11 & xor_138; and_39 = not_13 & not_5; xor_152 = not_10 ^ not_11;
xor_153 = xor_137 ^ xor_138; and_40 = xor_152 & xor_153;
xor_154 = not_10 ^ not_13; xor_155 = not_5 ^ xor_137;
and_41 = xor_154 & xor_155; xor_156 = not_11 ^ not_13;
xor_157 = not_5 ^ xor_138; and_42 = xor_156 & xor_157;
xor_158 = not_10 ^ xor_113; xor_159 = not_11 ^ xor_140;
xor_160 = not_13 ^ xor_141; xor_161 = xor_137 ^ xor_46;
xor_162 = xor_138 ^ xor_84; xor_163 = not_5 ^ xor_139;
and_43 = xor_158 & xor_161; and_44 = xor_159 & xor_162;
xor_164 = xor_158 ^ xor_159; xor_165 = xor_161 ^ xor_162;
and_45 = xor_164 & xor_165; and_46 = xor_160 & xor_163;
xor_166 = xor_142 ^ xor_160; xor_167 = xor_163 ^ xor_98;
and_47 = xor_166 & xor_167; xor_168 = xor_158 ^ xor_160;
xor_169 = xor_142 ^ xor_159; xor_170 = xor_161 ^ xor_163;
xor_171 = xor_162 ^ xor_98; and_48 = xor_168 & xor_170;
and_49 = xor_169 & xor_171; xor_172 = xor_168 ^ xor_169;
xor_173 = xor_170 ^ xor_171; and_50 = xor_172 & xor_173;
xor_174 = and_29 ^ and_43; xor_175 = and_34 ^ and_47;
xor_176 = and_42 ^ xor_174; xor_177 = and_28 ^ and_31;
xor_178 = and_33 ^ and_38; xor_179 = xor_175 ^ xor_178;
xor_180 = and_35 ^ and_39; xor_181 = and_45 ^ xor_176;
xor_182 = and_32 ^ and_41; xor_183 = and_44 ^ xor_177;
xor_184 = and_36 ^ and_48; xor_185 = and_50 ^ xor_179;
xor_186 = xor_184 ^ xor_181; xor_187 = xor_185 ^ xor_186;
xor_188 = and_49 ^ xor_180; xor_189 = and_30 ^ and_46;
xor_190 = and_40 ^ xor_183; xor_191 = and_31 ^ and_44;
xor_192 = xor_187 ^ xor_191; xor_193 = xor_188 ^ xor_192;
xor_194 = xor_189 ^ xor_193; xor_195 = and_37 ^ xor_187;
xor_196 = xor_188 ^ xor_179; xor_197 = xor_190 ^ xor_196;
xor_198 = and_28 ^ and_29; xor_199 = and_36 ^ xor_198;

```

```

xor_200 = and_40 ^ xor_199; xor_201 = xor_175 ^ xor_200;
xor_202 = xor_201 ^ xor_180; xor_203 = xor_202 ^ xor_182;
xor_204 = xor_189 ^ xor_203; xor_205 = and_35 ^ xor_176;
xor_206 = xor_177 ^ xor_205; xor_207 = xor_206 ^ xor_182;
xor_208 = and_30 ^ and_32; xor_209 = and_33 ^ xor_208;
xor_210 = and_37 ^ xor_209; xor_211 = xor_210 ^ xor_181;
xor_212 = xor_190 ^ xor_211; xor_213 = and_34 ^ and_37;
xor_214 = and_38 ^ xor_213; xor_215 = and_46 ^ xor_214;
xor_216 = and_48 ^ xor_215; xor_217 = xor_174 ^ xor_216;
xor_218 = xor_217 ^ xor_182; xor_219 = xor_218 ^ xor_183; y[0] = xor_194;
y[1] = xor_195; y[2] = xor_197; y[3] = xor_204; y[4] = xor_207; y[5] = xor_212;
y[6] = xor_219;

```

C.26 YSH_{8,1}: 8-bit degree 7, $\delta = 6$, $L = 56$ (8-bit, Category B)

- **Profile:** $\delta = 6$, $L = 56$ (NL = 100), $\deg(S) = 7$, $\mathcal{D}(S) = [6^1, 7^{254}]$, FP = 1, SAC deviation 0.0781.
- **Cost:** 43 AND, 107 XOR, 7 NOT, AND depth $D = 4$ (unmasked NanGate 45 nm: $220.25 \mu\text{m}^2$ / 276.0 GE, delay 1529.9 ps, logic depth 25; pipelined $d = 2$ DOM-indep: 236 DFF_X1 [172 gadget + 64 sync], $1869.98 \mu\text{m}^2$ / 2343.3 GE, stage delay 574.7 ps).
- **Remark:** Degree 7, the maximum for an 8-bit permutation, at $\delta = 6$, $L = 56$. The AES circuits of Table 4 are better in every metric.
- **Inverse** (YSH_{8,1}⁻¹): $\deg(S^{-1}) = 7$, $\mathcal{D}(S^{-1}) = [6^1, 7^{254}]$; cost: 43 AND, 109 XOR, 10 NOT, AND depth $D = 4$ (Tier 2: Output shear unwinding + 42-AND $D = 3$ $\mathbb{F}_2^8$ inversion core; unmasked NanGate 45 nm: $225.04 \mu\text{m}^2$ / 282.0 GE, delay 1380.5 ps, logic depth 24; pipelined $d = 2$ DOM-indep: 264 DFF_X1 [172 gadget + 92 sync], $2004.58 \mu\text{m}^2$ / 2512.0 GE, stage delay 578.1 ps).

Lookup table (LUT[0...255]):

```

0xc1, 0x7c, 0x5c, 0xb9, 0x2c, 0x83, 0x98, 0x02, 0xea, 0xe9, 0x88, 0x15, 0x10, 0xf9, 0xe8, 0xc9
0x84, 0x12, 0xae, 0xd3, 0x86, 0x85, 0x9c, 0x1a, 0x60, 0x1b, 0x74, 0x3e, 0x1d, 0x5b, 0x8e, 0x54
0x3f, 0x48, 0xcb, 0x16, 0x29, 0xff, 0xe0, 0xf8, 0xb5, 0xc4, 0xca, 0x69, 0xfa, 0xcf, 0xbe, 0xee
0xec, 0x64, 0x68, 0xe1, 0x06, 0xb0, 0x0c, 0xd1, 0x05, 0x20, 0x50, 0xe4, 0xa2, 0x49, 0x0f, 0xa3
0x7e, 0xfd, 0x4b, 0xed, 0x5a, 0x45, 0x1c, 0x18, 0x38, 0x63, 0x04, 0xf2, 0x35, 0x95, 0xb7, 0xde
0xab, 0x6e, 0x0d, 0x9d, 0xc5, 0xdf, 0xe6, 0x52, 0x81, 0xc2, 0x8c, 0x37, 0x4e, 0x9e, 0x32, 0x4a
0x1f, 0xf4, 0x72, 0x4d, 0xf6, 0xdc, 0x3c, 0xb1, 0x56, 0x41, 0x5d, 0x3a, 0x17, 0xa9, 0x0b, 0xd8
0x44, 0x5f, 0x3d, 0x1e, 0x4c, 0x80, 0x27, 0x26, 0x59, 0x2a, 0x6b, 0x2b, 0x46, 0x36, 0xdb, 0x33
0xba, 0x90, 0x11, 0x53, 0x76, 0xb4, 0x2f, 0x8a, 0xe3, 0x55, 0x30, 0x3b, 0xf1, 0x40, 0x89, 0xad
0xd5, 0xd2, 0x42, 0x14, 0x61, 0xb6, 0xd9, 0x22, 0x19, 0xcc, 0xa4, 0xf7, 0xef, 0x96, 0x67, 0x6a
0xc8, 0x24, 0xc0, 0x9f, 0x62, 0xa0, 0x75, 0xc6, 0x8f, 0x7a, 0xe7, 0x94, 0xf0, 0x9a, 0xda, 0x43
0x66, 0x0e, 0xfc, 0xb8, 0x47, 0x78, 0xc7, 0xd6, 0xfe, 0x21, 0x71, 0xf5, 0xaf, 0x51, 0x77, 0x79
0x2e, 0x07, 0x93, 0x58, 0xb2, 0xd4, 0x13, 0x82, 0x28, 0x57, 0x73, 0x6d, 0xd7, 0x25, 0xa5, 0x5e
0xa7, 0x97, 0x2d, 0x65, 0x6c, 0x99, 0xbd, 0x8b, 0xf3, 0xbf, 0xeb, 0x7f, 0x00, 0x39, 0x87, 0x08
0x6f, 0x92, 0x8d, 0xe2, 0xc3, 0x0a, 0x70, 0xaa, 0x34, 0x7d, 0xce, 0x09, 0x91, 0xdd, 0x23, 0xfb
0x9b, 0xac, 0xbc, 0xd0, 0xbb, 0xa6, 0x31, 0x7b, 0xe5, 0xa8, 0x4f, 0x01, 0xb3, 0xa1, 0xcd, 0x03

```

Netlist (43 AND, 107 XOR, 7 NOT, $D = 4$):

```

a_in = in_val[7:0];
split_1 = a_in[0:0]; split_2 = a_in[1:1]; split_3 = a_in[2:2];
split_4 = a_in[3:3]; split_5 = a_in[4:4]; split_6 = a_in[5:5];
split_7 = a_in[6:6]; split_8 = a_in[7:7]; xor_15 = split_1 ^ split_3;
xor_11 = split_4 ^ split_5; xor_9 = split_1 ^ split_6;
xor_12 = split_5 ^ split_6; xor_19 = split_2 ^ split_7;
xor_16 = split_8 ^ xor_15; xor_21 = split_2 ^ xor_11; xor_10 = split_7 ^ xor_9;
xor_13 = split_8 ^ xor_12; xor_23 = split_3 ^ xor_12; xor_20 = split_8 ^ xor_19;
not_17 = ~xor_16; xor_18 = split_3 ^ xor_10; xor_22 = xor_11 ^ xor_10;
not_14 = ~xor_13; paar_shared_xor_10_xor_21_xor_23 = xor_21 ^ xor_23;
and_31 = xor_20 & xor_10; xor_54 = xor_20 ^ xor_10;
paar_shared_xor_7_xor_20_xor_21 = xor_20 ^ xor_21;
paar_shared_xor_11_not_17_xor_10 = xor_10 ^ not_17; and_25 = xor_23 & xor_18;
xor_57 = xor_23 ^ xor_18; paar_shared_xor_6_not_17_xor_18 = not_17 ^ xor_18;
and_24 = xor_22 & not_17; xor_56 = xor_22 ^ not_17;
paar_shared_xor_5_xor_22_xor_23 = xor_22 ^ xor_23;
paar_shared_xor_9_xor_20_xor_22 = xor_20 ^ xor_22; and_32 = xor_21 & not_14;
xor_55 = xor_21 ^ not_14; paar_shared_xor_8_not_14_xor_10 = xor_10 ^ not_14;
paar_shared_xor_12_not_14_xor_18 = not_14 ^ xor_18; xor_58 = xor_54 ^ xor_55;

```

```

xor_59 = xor_55 ^ xor_56;
and_28 = paar_shared_xor_5_xor_22_xor_23 & paar_shared_xor_6_not_17_xor_18;
and_42 = paar_shared_xor_9_xor_20_xor_22 & paar_shared_xor_11_not_17_xor_10;
xor_44 = paar_shared_xor_9_xor_20_xor_22 ^ paar_shared_xor_10_xor_21_xor_23;
xor_60 = xor_58 ^ xor_56; xor_61 = xor_59 ^ xor_57;
and_35 = paar_shared_xor_7_xor_20_xor_21 & paar_shared_xor_8_not_14_xor_10;
and_43 = paar_shared_xor_10_xor_21_xor_23 & paar_shared_xor_12_not_14_xor_18;
xor_45 = paar_shared_xor_11_not_17_xor_10 ^ paar_shared_xor_12_not_14_xor_18;
xor_29 = and_28 ^ and_24; xor_30 = and_28 ^ and_25; xor_36 = and_35 ^ and_31;
xor_37 = and_35 ^ and_32; and_46 = xor_44 & xor_45; xor_47 = and_46 ^ and_42;
xor_48 = and_46 ^ and_43; xor_51 = xor_37 ^ xor_47; xor_53 = xor_30 ^ xor_47;
xor_49 = xor_47 ^ xor_48; xor_63 = xor_51 ^ xor_61; xor_65 = xor_53 ^ xor_55;
xor_50 = xor_36 ^ xor_49; xor_52 = xor_29 ^ xor_49; and_79 = xor_18 & xor_63;
and_109 = xor_23 & xor_63; and_67 = xor_65 & xor_63; and_86 = not_14 & xor_65;
and_116 = xor_21 & xor_65; paar_shared_xor_4_xor_63_xor_65 = xor_63 ^ xor_65;
xor_62 = xor_50 ^ xor_60; xor_64 = xor_52 ^ xor_58; and_78 = not_17 & xor_62;
and_108 = xor_22 & xor_62; paar_shared_xor_2_xor_62_xor_63 = xor_62 ^ xor_63;
and_66 = xor_64 & xor_62; and_85 = xor_10 & xor_64; and_115 = xor_20 & xor_64;
paar_shared_xor_1_xor_64_xor_65 = xor_64 ^ xor_65;
paar_shared_xor_3_xor_62_xor_64 = xor_62 ^ xor_64;
and_97 = paar_shared_xor_12_not_14_xor_18 & paar_shared_xor_4_xor_63_xor_65;
and_127 = paar_shared_xor_10_xor_21_xor_23 & paar_shared_xor_4_xor_63_xor_65;
and_82 = paar_shared_xor_6_not_17_xor_18 & paar_shared_xor_2_xor_62_xor_63;
and_112 = paar_shared_xor_5_xor_22_xor_23 & paar_shared_xor_2_xor_62_xor_63;
and_70 = paar_shared_xor_1_xor_64_xor_65 & paar_shared_xor_2_xor_62_xor_63;
and_89 = paar_shared_xor_8_not_14_xor_10 & paar_shared_xor_1_xor_64_xor_65;
and_119 = paar_shared_xor_7_xor_20_xor_21 & paar_shared_xor_1_xor_64_xor_65;
xor_75 = paar_shared_xor_3_xor_62_xor_64 ^ paar_shared_xor_4_xor_63_xor_65;
and_96 = paar_shared_xor_11_not_17_xor_10 & paar_shared_xor_3_xor_62_xor_64;
and_126 = paar_shared_xor_9_xor_20_xor_22 & paar_shared_xor_3_xor_62_xor_64;
xor_83 = and_82 ^ and_78; xor_84 = and_82 ^ and_79; xor_113 = and_112 ^ and_108;
xor_114 = and_112 ^ and_109; xor_71 = and_70 ^ and_66; xor_72 = and_70 ^ and_67;
xor_90 = and_89 ^ and_85; xor_91 = and_89 ^ and_86; xor_120 = and_119 ^ and_115;
xor_121 = and_119 ^ and_116; and_100 = xor_45 & xor_75;
and_130 = xor_44 & xor_75; xor_76 = xor_71 ^ xor_75;
xor_77 = xor_72 ^ paar_shared_xor_4_xor_63_xor_65; xor_101 = and_100 ^ and_96;
xor_102 = and_100 ^ and_97; xor_131 = and_130 ^ and_126;
xor_132 = and_130 ^ and_127; paar_shared_xor_0_xor_76_xor_77 = xor_76 ^ xor_77;
xor_105 = xor_91 ^ xor_101; xor_107 = xor_84 ^ xor_101;
xor_103 = xor_101 ^ xor_102; xor_135 = xor_121 ^ xor_131;
xor_137 = xor_114 ^ xor_131; xor_133 = xor_131 ^ xor_132;
and_139 = xor_105 & xor_76; and_146 = xor_107 & xor_76;
xor_104 = xor_90 ^ xor_103; xor_106 = xor_83 ^ xor_103;
and_153 = xor_135 & xor_76; and_160 = xor_137 & xor_76;
xor_134 = xor_120 ^ xor_133; xor_136 = xor_113 ^ xor_133;
and_138 = xor_104 & xor_77; xor_140 = xor_104 ^ xor_105;
and_145 = xor_106 & xor_77; xor_147 = xor_106 ^ xor_107;
and_152 = xor_134 & xor_77; xor_154 = xor_134 ^ xor_135;
and_159 = xor_136 & xor_77; xor_161 = xor_136 ^ xor_137;
and_142 = xor_140 & paar_shared_xor_0_xor_76_xor_77;
and_149 = xor_147 & paar_shared_xor_0_xor_76_xor_77;
and_156 = xor_154 & paar_shared_xor_0_xor_76_xor_77;
and_163 = xor_161 & paar_shared_xor_0_xor_76_xor_77; xor_143 = and_142 ^ and_138;
xor_144 = and_142 ^ and_139; xor_150 = and_149 ^ and_145;
xor_151 = and_149 ^ and_146; xor_157 = and_156 ^ and_152;
xor_158 = and_156 ^ and_153; xor_164 = and_163 ^ and_159;
xor_165 = and_163 ^ and_160; xor_168 = xor_143 ^ xor_144;
xor_174 = xor_143 ^ xor_150; xor_178 = xor_150 ^ xor_151;
xor_167 = xor_158 ^ xor_150; xor_177 = xor_157 ^ xor_164;
xor_166 = xor_165 ^ xor_151; xor_172 = xor_157 ^ xor_168;
xor_170 = xor_167 ^ xor_168; xor_179 = xor_177 ^ xor_178;
xor_169 = xor_157 ^ xor_166; xor_180 = xor_166 ^ xor_167; not_173 = ~xor_172;
xor_171 = xor_164 ^ xor_170; xor_183 = xor_166 ^ xor_170;
xor_175 = xor_174 ^ xor_169; xor_181 = xor_158 ^ xor_169; not_176 = ~xor_171;
xor_185 = xor_169 ^ xor_171; not_184 = ~xor_183; not_182 = ~xor_181;
and_188 = xor_175 & not_176; not_186 = ~xor_185; xor_189 = not_173 ^ and_188;
concat_190 = { not_186, not_184, not_182, xor_180, xor_179, not_176, xor_175, xor_189 };
out_val = concat_190;

```

Inverse lookup table ($\text{LUT}_{S^{-1}}[0 \dots 255]$):

```

0xdc, 0xfb, 0x07, 0xff, 0x4a, 0x38, 0x34, 0xc1, 0xdf, 0xeb, 0xe5, 0x6e, 0x36, 0x52, 0xb1, 0x3e
0x0c, 0x82, 0x11, 0xc6, 0x93, 0x0b, 0x23, 0x6c, 0x47, 0x98, 0x17, 0x19, 0x46, 0x1c, 0x73, 0x60
0x39, 0xb9, 0x97, 0xee, 0xa1, 0xcd, 0x77, 0x76, 0xc8, 0x24, 0x79, 0x7b, 0x04, 0xd2, 0xc0, 0x86
0x8a, 0xf6, 0x5e, 0x7f, 0xe8, 0x4c, 0x7d, 0x5b, 0x48, 0xdd, 0x6b, 0x8b, 0x66, 0x72, 0x1b, 0x20
0x8d, 0x69, 0x92, 0xaf, 0x70, 0x45, 0x7c, 0xb4, 0x21, 0x3d, 0x5f, 0x42, 0x74, 0x63, 0x5c, 0xfa
0x3a, 0xbd, 0x57, 0x83, 0x1f, 0x89, 0x68, 0xc9, 0xc3, 0x78, 0x44, 0x1d, 0x02, 0x6a, 0xcf, 0x71
0x18, 0x94, 0xa4, 0x49, 0x31, 0xd3, 0xb0, 0x9e, 0x32, 0x2b, 0x9f, 0x7a, 0xd4, 0xcb, 0x51, 0xe0
0xe6, 0xba, 0x62, 0xca, 0x1a, 0xa6, 0x84, 0xbe, 0xb5, 0xbf, 0xa9, 0xf7, 0x01, 0xe9, 0x40, 0xdb

```

```

0x75, 0x58, 0xc7, 0x05, 0x10, 0x15, 0x14, 0xde, 0x0a, 0x8e, 0x87, 0xd7, 0x5a, 0xe2, 0x1e, 0xa8
0x81, 0xec, 0xe1, 0xc2, 0xab, 0x4d, 0x9d, 0xd1, 0x06, 0xd5, 0xad, 0xf0, 0x16, 0x53, 0x5d, 0xa3
0xa5, 0xfd, 0x3c, 0x3f, 0x9a, 0xce, 0xf5, 0xd0, 0xf9, 0x6d, 0xe7, 0x50, 0xf1, 0x8f, 0x12, 0xbc
0x35, 0x67, 0xc4, 0xfc, 0x85, 0x28, 0x95, 0x4e, 0xb3, 0x03, 0x80, 0xf4, 0xf2, 0xd6, 0x2e, 0xd9
0xa2, 0x00, 0x59, 0xe4, 0x29, 0x54, 0xa7, 0xb6, 0xa0, 0x0f, 0x2a, 0x22, 0x99, 0xfe, 0xea, 0x2d
0xf3, 0x37, 0x91, 0x13, 0xc5, 0x90, 0xb7, 0xcc, 0x6f, 0x96, 0xae, 0x7e, 0x65, 0xed, 0x4f, 0x55
0x26, 0x33, 0xe3, 0x88, 0x3b, 0xf8, 0x56, 0xaa, 0x0e, 0x09, 0x08, 0xda, 0x30, 0x43, 0x2f, 0x9c
0xac, 0x8c, 0x4b, 0xd8, 0x61, 0xbb, 0x64, 0x9b, 0x27, 0xd0, 0x2c, 0xef, 0xb2, 0x41, 0xb8, 0x25

```

Inverse netlist (43 AND, 109 XOR, 10 NOT, $D = 4$):

```

and_0 = x[1] & x[2]; xor_0 = and_0 ^ x[0]; not_0 = ~xor_0; not_1 = ~x[2];
not_2 = ~x[5]; not_3 = ~x[6]; not_4 = ~x[7]; xor_1 = not_2 ^ x[4];
xor_2 = not_3 ^ x[1]; xor_3 = not_0 ^ not_4; xor_4 = x[3] ^ xor_2;
xor_5 = not_1 ^ xor_1; xor_6 = not_4 ^ xor_4; xor_7 = xor_1 ^ xor_3;
xor_8 = not_0 ^ not_1; xor_9 = not_1 ^ xor_3; xor_10 = not_0 ^ x[1];
xor_11 = x[1] ^ x[3]; xor_12 = not_1 ^ not_2; xor_13 = not_1 ^ x[1];
xor_14 = x[4] ^ xor_13; xor_15 = x[3] ^ xor_7; xor_16 = xor_5 ^ xor_10;
xor_17 = xor_2 ^ xor_8; xor_18 = not_3 ^ x[3]; xor_19 = xor_18 ^ xor_8;
xor_20 = not_2 ^ x[1]; xor_21 = xor_20 ^ xor_3; xor_22 = not_0 ^ not_2;
xor_23 = xor_5 ^ xor_6; xor_24 = xor_1 ^ xor_4; xor_25 = not_4 ^ xor_2;
xor_26 = xor_12 ^ xor_6; xor_27 = xor_3 ^ xor_5; xor_28 = xor_4 ^ xor_9;
xor_29 = not_2 ^ xor_10; xor_30 = xor_11 ^ xor_3; xor_31 = x[3] ^ xor_12;
and_1 = xor_14 & xor_7; and_2 = xor_15 & xor_16; and_3 = xor_17 & xor_19;
and_4 = xor_21 & xor_9; and_5 = x[3] & xor_22; and_6 = xor_23 & xor_24;
xor_32 = and_1 ^ and_5; xor_33 = and_2 ^ and_5; and_7 = xor_25 & xor_26;
and_8 = xor_27 & xor_5; xor_34 = and_4 ^ and_7; xor_35 = and_3 ^ and_7;
and_9 = xor_28 & xor_6; xor_36 = and_8 ^ and_9; xor_37 = and_6 ^ and_9;
xor_38 = xor_35 ^ xor_36; xor_39 = xor_33 ^ xor_36; xor_40 = xor_36 ^ xor_37;
xor_41 = xor_29 ^ xor_38; xor_42 = xor_11 ^ xor_39; xor_43 = xor_34 ^ xor_40;
xor_44 = xor_32 ^ xor_40; and_10 = xor_16 & xor_41; and_11 = xor_15 & xor_41;
and_12 = xor_41 & xor_42; and_13 = xor_19 & xor_42; and_14 = xor_17 & xor_42;
xor_45 = xor_41 ^ xor_42; xor_46 = xor_30 ^ xor_43; xor_47 = xor_31 ^ xor_44;
and_15 = xor_14 & xor_46; and_16 = xor_46 & xor_7; xor_48 = xor_41 ^ xor_46;
and_17 = xor_46 & xor_47; and_18 = xor_21 & xor_47; and_19 = xor_47 & xor_9;
xor_49 = xor_42 ^ xor_47; xor_50 = xor_46 ^ xor_47; and_20 = xor_24 & xor_45;
and_21 = xor_23 & xor_45; and_22 = xor_22 & xor_48; and_23 = x[3] & xor_48;
and_24 = xor_48 & xor_49; and_25 = xor_26 & xor_49; and_26 = xor_25 & xor_49;
xor_51 = xor_45 ^ xor_50; and_27 = xor_27 & xor_50; and_28 = xor_5 & xor_50;
xor_52 = and_15 ^ and_22; xor_53 = and_22 ^ and_10; xor_54 = and_16 ^ and_23;
xor_55 = and_11 ^ and_23; xor_56 = and_17 ^ and_24; xor_57 = and_12 ^ and_24;
xor_58 = and_18 ^ and_25; xor_59 = and_13 ^ and_25; xor_60 = and_19 ^ and_26;
xor_61 = and_14 ^ and_26; and_29 = xor_28 & xor_51; and_30 = xor_6 & xor_51;
xor_62 = xor_51 ^ xor_56; xor_63 = xor_45 ^ xor_57; xor_64 = and_27 ^ and_29;
xor_65 = and_20 ^ and_29; xor_66 = and_28 ^ and_30; xor_67 = and_21 ^ and_30;
xor_68 = xor_62 ^ xor_63; xor_69 = xor_59 ^ xor_64; xor_70 = xor_53 ^ xor_64;
xor_71 = xor_64 ^ xor_65; xor_72 = xor_61 ^ xor_66; xor_73 = xor_55 ^ xor_66;
xor_74 = xor_66 ^ xor_67; and_31 = xor_62 & xor_69; and_32 = xor_62 & xor_70;
xor_75 = xor_58 ^ xor_71; xor_76 = xor_52 ^ xor_71; and_33 = xor_62 & xor_72;
and_34 = xor_62 & xor_73; xor_77 = xor_60 ^ xor_74; xor_78 = xor_54 ^ xor_74;
and_35 = xor_63 & xor_75; xor_79 = xor_69 ^ xor_75; and_36 = xor_63 & xor_76;
xor_80 = xor_70 ^ xor_76; and_37 = xor_63 & xor_77; xor_81 = xor_72 ^ xor_77;
and_38 = xor_63 & xor_78; xor_82 = xor_73 ^ xor_78; and_39 = xor_68 & xor_79;
and_40 = xor_68 & xor_80; and_41 = xor_68 & xor_81; and_42 = xor_68 & xor_82;
xor_83 = and_35 ^ and_38; xor_84 = and_37 ^ and_42; xor_85 = and_32 ^ xor_83;
xor_86 = and_40 ^ xor_84; xor_87 = and_34 ^ and_36; xor_88 = and_39 ^ xor_85;
xor_89 = and_31 ^ xor_85; xor_90 = and_33 ^ xor_86; xor_91 = and_41 ^ xor_86;
xor_92 = xor_87 ^ xor_89; xor_93 = and_36 ^ and_42; xor_94 = xor_88 ^ xor_93;
xor_95 = xor_88 ^ xor_90; xor_96 = and_36 ^ and_39; xor_97 = xor_83 ^ xor_96;
xor_98 = xor_91 ^ xor_97; not_5 = ~xor_98; xor_99 = xor_89 ^ xor_90;
not_6 = ~xor_99; xor_100 = and_31 ^ and_35; xor_101 = xor_87 ^ xor_100;
xor_102 = xor_101 ^ xor_91; not_7 = ~xor_102; xor_103 = and_32 ^ and_41;
xor_104 = xor_103 ^ xor_84; xor_105 = xor_104 ^ xor_87; not_8 = ~xor_105;
xor_106 = and_33 ^ and_38; xor_107 = and_41 ^ xor_106;
xor_108 = and_42 ^ xor_107; not_9 = ~xor_108; y[0] = xor_92; y[1] = xor_94;
y[2] = xor_95; y[3] = not_5; y[4] = not_6; y[5] = not_7; y[6] = not_8;
y[7] = not_9;

```

C.27 YSH_{8,2}: 8-bit degree 7, $\delta = 10$, $L = 72$, 28 AND (8-bit, Category B)

- **Profile:** $\delta = 10$, $L = 72$ (NL = 92), $\deg(S) = 7$, $\mathcal{D}(S) = [6^3, 7^{252}]$, FP = 0, SAC deviation 0.1094.
- **Cost:** 28 AND, 61 XOR, 1 NOT, AND depth $D = 6$ (unmasked NanGate 45 nm:

$127.68 \mu\text{m}^2 / 160.0 \text{ GE}$, delay 969.0 ps , logic depth 17; pipelined $d = 2$ DOM-indep:
 208 DFF_X1 [112 gadget + 96 sync], $1433.74 \mu\text{m}^2 / 1796.7 \text{ GE}$, stage delay 275.1 ps).

- **Remark:** A six-round generalized Feistel network over single-bit branches (with 4, 6, 4, 4, 4, 6 AND gates per level) that Stage 2 evolved from the Canright tower-field AES seed (36/136/4 [Can05]; Section 6.5): 8 fewer AND and 75 fewer XOR gates than the seed at two more AND levels and the relaxed profile $(\delta, L) = (10, 72)$. It realizes the 8-bit $(10, 72, 7)$ profile with 28 AND gates where the ANF-synthesized archive representatives of the same profile need 239 (Table 11), with $\beta = 22$ and no fixed point; $\text{YSH}_{8,1}$ and the AES circuits remain better in δ and L . Stage-2 AND erasure applied to this circuit produces the AND-versus- δ family of Table 8, whose 18-AND member is $\text{YSH}_{8,3}$ (SM C.28).
- **Inverse** ($\text{YSH}_{8,2}^{-1}$): $\deg(S^{-1}) = 7$, $\mathcal{D}(S^{-1}) = [6^3, 7^{252}]$; cost: 28 AND, 84 XOR, 2 NOT, AND depth $D = 6$ (Tier 2: Exact 6-round generalized Feistel layer reversal; unmasked NanGate 45 nm: $164.92 \mu\text{m}^2 / 206.7 \text{ GE}$, delay 1524.5 ps , logic depth 28; pipelined $d = 2$ DOM-indep: 296 DFF_X1 [112 gadget + 184 sync], $1905.62 \mu\text{m}^2 / 2388.0 \text{ GE}$, stage delay 431.9 ps).

Lookup table (LUT[0...255]):

```
0x32, 0x24, 0x45, 0xdc, 0x07, 0xb7, 0xde, 0x62, 0xd1, 0xc4, 0xcb, 0x0c, 0xbb, 0x2e, 0x3f, 0x8f
0x03, 0x00, 0xd4, 0xf4, 0x15, 0xd6, 0xee, 0x67, 0x70, 0x83, 0x63, 0x89, 0xa1, 0xb5, 0x81, 0xc9
0x51, 0xbc, 0x55, 0xe0, 0xf7, 0x2f, 0x64, 0xa4, 0x22, 0xf9, 0x97, 0x9a, 0x6f, 0xeb, 0xae, 0x93
0x86, 0x92, 0xab, 0x23, 0xf1, 0x05, 0x3b, 0x1b, 0x88, 0xb3, 0xd3, 0xb4, 0xda, 0xb1, 0x38, 0x1c
0x0a, 0xea, 0x9c, 0x04, 0xa5, 0xd8, 0x3a, 0xac, 0x95, 0x71, 0x19, 0x9e, 0xdd, 0x0b, 0xc7, 0xf6
0x6b, 0xba, 0x6a, 0x8c, 0x7b, 0x08, 0xd0, 0xe9, 0xb8, 0x7e, 0x76, 0xcf, 0xfe, 0x9d, 0x2c, 0x17
0x1a, 0xe6, 0x2d, 0xbd, 0xe1, 0x96, 0x12, 0xc6, 0x31, 0x11, 0x21, 0x50, 0x99, 0x4b, 0x8b, 0x9b
0xfc, 0xff, 0x20, 0x42, 0x28, 0x61, 0xad, 0xb0, 0x1e, 0xc0, 0x8a, 0x5e, 0xd9, 0x34, 0x7f, 0xe4
0x10, 0x4f, 0x98, 0xa7, 0x2a, 0x0e, 0xd2, 0xa8, 0xc1, 0xfd, 0x36, 0x06, 0x8e, 0x7d, 0x8d, 0x14
0x6d, 0xa6, 0xf5, 0xb6, 0xa9, 0x84, 0x13, 0x0f, 0xaf, 0x5f, 0xe5, 0x41, 0xbf, 0x39, 0xd7, 0x57
0x2b, 0xb9, 0x56, 0x47, 0x82, 0x1d, 0x53, 0xf2, 0x66, 0x27, 0x4d, 0xfb, 0x02, 0x26, 0xa2, 0xca
0xe8, 0xe3, 0x79, 0x65, 0x59, 0xf3, 0x44, 0xfa, 0x6e, 0x0d, 0xf8, 0x75, 0x80, 0xed, 0x18, 0x01
0x54, 0x5b, 0x74, 0x40, 0xdb, 0x91, 0x30, 0x4a, 0xce, 0x52, 0x09, 0xef, 0x87, 0x7a, 0xec, 0xc5
0x68, 0x7c, 0x6c, 0x5c, 0xe2, 0xaa, 0x77, 0x9f, 0x3d, 0xc2, 0x72, 0x69, 0xc3, 0x43, 0x5a, 0xcd
0xbe, 0xd5, 0xb2, 0x60, 0x25, 0x33, 0x4e, 0xdf, 0x16, 0x58, 0x49, 0xa3, 0x35, 0x78, 0x37, 0x73
0x94, 0x48, 0x90, 0x1f, 0x46, 0x3c, 0xf0, 0x4c, 0x3e, 0x5d, 0x29, 0xcc, 0x85, 0xa0, 0xe7, 0xc8
```

Netlist (28 AND, 61 XOR, 1 NOT, $D = 6$):

```
xor_0 = x[0] ^ x[4]; and_0 = x[4] & x[5]; xor_1 = x[3] ^ x[5];
and_1 = x[4] & x[6]; and_2 = x[5] & x[6]; xor_2 = x[2] ^ x[6];
and_3 = x[6] & x[7]; xor_3 = x[1] ^ x[7]; xor_4 = xor_0 ^ and_0;
xor_5 = xor_1 ^ and_2; xor_6 = xor_2 ^ and_3; xor_7 = xor_3 ^ and_1;
xor_8 = x[7] ^ xor_4; and_4 = xor_4 & xor_5; xor_9 = x[4] ^ xor_5;
xor_10 = xor_5 ^ xor_4; and_5 = xor_4 & xor_6; and_6 = xor_6 & xor_5;
xor_11 = x[6] ^ xor_6; and_7 = xor_4 & xor_7; and_8 = xor_7 & xor_6;
and_9 = xor_7 & xor_5; xor_12 = x[5] ^ xor_7; xor_13 = and_4 ^ and_6;
xor_14 = and_7 ^ and_5; xor_15 = and_5 ^ and_8; xor_16 = and_4 ^ and_9;
xor_17 = xor_12 ^ xor_13; xor_18 = xor_11 ^ xor_14; xor_19 = xor_8 ^ xor_15;
xor_20 = xor_9 ^ xor_16; xor_21 = xor_6 ^ xor_17; and_10 = xor_18 & xor_4;
xor_22 = xor_7 ^ xor_18; and_11 = xor_17 & xor_18; and_12 = xor_18 & xor_19;
and_13 = xor_17 & xor_19; not_0 = ~xor_20; xor_23 = xor_10 ^ and_10;
xor_24 = xor_21 ^ and_11; xor_25 = xor_22 ^ and_13; xor_26 = not_0 ^ xor_19;
xor_27 = xor_19 ^ xor_23; xor_28 = xor_17 ^ xor_24; and_14 = xor_25 & xor_23;
xor_29 = xor_4 ^ xor_25; and_15 = xor_25 & xor_24; xor_30 = xor_26 ^ and_12;
and_16 = xor_24 & xor_30; and_17 = xor_23 & xor_30; xor_31 = xor_18 ^ xor_30;
xor_32 = xor_30 ^ xor_25; xor_33 = and_14 ^ and_16; xor_34 = and_15 ^ and_16;
xor_35 = and_15 ^ and_17; xor_36 = and_14 ^ and_17; xor_37 = xor_28 ^ xor_33;
xor_38 = xor_27 ^ xor_34; xor_39 = xor_31 ^ xor_35; xor_40 = xor_29 ^ xor_36;
xor_41 = xor_24 ^ xor_38; and_18 = xor_39 & xor_38; and_19 = xor_39 & xor_25;
xor_42 = xor_37 ^ xor_39; and_20 = xor_38 & xor_40; and_21 = xor_40 & xor_25;
xor_43 = xor_23 ^ xor_40; xor_44 = xor_32 ^ and_19; xor_45 = xor_42 ^ and_18;
xor_46 = xor_41 ^ and_21; xor_47 = xor_43 ^ and_20; xor_48 = xor_38 ^ xor_44;
and_22 = xor_44 & xor_45; xor_49 = xor_40 ^ xor_45; and_23 = xor_46 & xor_44;
xor_50 = xor_39 ^ xor_46; and_24 = xor_46 & xor_45; and_25 = xor_46 & xor_47;
and_26 = xor_47 & xor_44; xor_51 = xor_25 ^ xor_47; and_27 = xor_47 & xor_45;
xor_52 = and_23 ^ and_24; xor_53 = and_25 ^ and_22; xor_54 = and_23 ^ and_26;
xor_55 = and_26 ^ and_27; xor_56 = xor_49 ^ xor_52; xor_57 = xor_50 ^ xor_53;
xor_58 = xor_48 ^ xor_54; xor_59 = xor_51 ^ xor_55; xor_60 = xor_45 ^ xor_56;
y[0] = xor_47; y[1] = xor_44; y[2] = xor_60; y[3] = xor_57; y[4] = xor_58;
y[5] = xor_56; y[6] = xor_59; y[7] = xor_46;
```

Inverse lookup table ($\text{LUT}_{S^{-1}}[0 \dots 255]$):

```

0x11, 0xbf, 0xac, 0x10, 0x43, 0x35, 0x8b, 0x04, 0x55, 0xca, 0x40, 0x4d, 0x0b, 0xb9, 0x85, 0x97
0x80, 0x69, 0x66, 0x96, 0x8f, 0x14, 0xe8, 0x5f, 0xbe, 0x4a, 0x60, 0x37, 0x3f, 0xa5, 0x78, 0xf3
0x72, 0x6a, 0x28, 0x33, 0x01, 0xe4, 0xad, 0xa9, 0x74, 0xfa, 0x84, 0xa0, 0x5e, 0x62, 0x0d, 0x25
0xc6, 0x68, 0x00, 0xe5, 0x7d, 0xec, 0x8a, 0xee, 0x3e, 0x9d, 0x46, 0x36, 0xf5, 0xd8, 0xf8, 0x0e
0xc3, 0x9b, 0x73, 0xdd, 0xb6, 0x02, 0xf4, 0xa3, 0xf1, 0xea, 0xc7, 0x6d, 0xf7, 0xaa, 0xe6, 0x81
0x6b, 0x20, 0xc9, 0xa6, 0xc0, 0x22, 0xa2, 0x9f, 0xe9, 0xb4, 0xde, 0xc1, 0xd3, 0xf9, 0x7b, 0x99
0xe3, 0x75, 0x07, 0x1a, 0x26, 0xb3, 0xa8, 0x17, 0xd0, 0xdb, 0x52, 0x50, 0xd2, 0x90, 0xb8, 0x2c
0x18, 0x49, 0xda, 0xef, 0xc2, 0xbb, 0x5a, 0xd6, 0xed, 0xb2, 0xcd, 0x54, 0xd1, 0x8d, 0x59, 0x7e
0xbc, 0x1e, 0xa4, 0x19, 0x95, 0xfc, 0x30, 0xcc, 0x38, 0x1b, 0x7a, 0x6e, 0x53, 0x8e, 0x8c, 0x0f
0xf2, 0xc5, 0x31, 0x2f, 0xf0, 0x48, 0x65, 0x2a, 0x82, 0x6c, 0x2b, 0x6f, 0x42, 0x5d, 0x4b, 0xd7
0xfd, 0x1c, 0xae, 0xeb, 0x27, 0x44, 0x91, 0x83, 0x87, 0x94, 0xd5, 0x32, 0x47, 0x76, 0x2e, 0x98
0x77, 0x3d, 0xe2, 0x39, 0x3b, 0x1d, 0x93, 0x05, 0x58, 0xa1, 0x51, 0x0c, 0x21, 0x63, 0xe0, 0x9c
0x79, 0x88, 0xd9, 0xdc, 0x09, 0xcf, 0x67, 0x4e, 0xff, 0x1f, 0xaf, 0x0a, 0xfb, 0xdf, 0xc8, 0x5b
0x56, 0x08, 0x86, 0x3a, 0x12, 0xe1, 0x15, 0x9e, 0x45, 0x7c, 0x3c, 0xc4, 0x03, 0x4c, 0x06, 0xe7
0x23, 0x64, 0xd4, 0xb1, 0x7f, 0x9a, 0x61, 0xfe, 0xb0, 0x57, 0x41, 0x2d, 0xce, 0xbd, 0x16, 0xcb
0xf6, 0x34, 0xa7, 0xb5, 0x13, 0x92, 0x4f, 0x24, 0xba, 0x29, 0xb7, 0xab, 0x70, 0x89, 0x5c, 0x71

```

Inverse netlist (28 AND, 84 XOR, 2 NOT, $D = 6$):

```

xor_0 = x[2] ^ x[5]; xor_1 = x[1] ^ x[7]; and_0 = x[1] & xor_0;
and_1 = x[7] & x[1]; and_2 = x[7] & xor_0; and_3 = x[7] & x[0];
and_4 = x[0] & x[1]; and_5 = x[0] & xor_0; xor_2 = x[6] ^ and_5;
xor_3 = and_4 ^ xor_2; xor_4 = x[0] ^ xor_3; xor_5 = x[3] ^ and_3;
xor_6 = and_0 ^ xor_5; xor_7 = x[4] ^ and_1; xor_8 = x[7] ^ xor_6;
xor_9 = and_4 ^ xor_7; xor_10 = x[2] ^ and_1; xor_11 = and_2 ^ xor_10;
xor_12 = xor_2 ^ xor_7; xor_13 = x[1] ^ xor_9; xor_14 = x[0] ^ xor_12;
and_6 = xor_8 & xor_13; and_7 = xor_8 & xor_4; and_8 = xor_13 & xor_11;
and_9 = xor_11 & xor_4; xor_15 = x[5] ^ and_6; xor_16 = and_9 ^ xor_9;
xor_17 = and_8 ^ xor_12; xor_18 = x[0] ^ and_8; xor_19 = xor_11 ^ xor_18;
xor_20 = and_7 ^ xor_4; xor_21 = xor_16 ^ xor_1; xor_22 = x[1] ^ xor_20;
xor_23 = and_9 ^ xor_17; and_10 = xor_4 & xor_19; and_11 = xor_4 & xor_21;
and_12 = xor_21 & xor_22; and_13 = xor_19 & xor_22; xor_24 = and_10 ^ and_13;
xor_25 = and_7 ^ and_11; xor_26 = x[2] ^ and_12; xor_27 = xor_26 ^ xor_15;
xor_28 = x[1] ^ xor_25; xor_29 = xor_24 ^ xor_4; xor_30 = xor_8 ^ xor_28;
xor_31 = and_13 ^ xor_30; xor_32 = and_2 ^ xor_26; xor_33 = xor_27 ^ xor_16;
xor_34 = and_4 ^ xor_32; xor_35 = x[4] ^ xor_34; xor_36 = and_10 ^ xor_8;
xor_37 = xor_29 ^ xor_11; xor_38 = x[1] ^ and_11; xor_39 = xor_4 ^ xor_31;
xor_40 = and_12 ^ xor_17; xor_41 = xor_38 ^ xor_18; xor_42 = and_10 ^ xor_6;
xor_43 = x[1] ^ xor_33; xor_44 = xor_35 ^ xor_41; xor_45 = xor_42 ^ xor_43;
xor_46 = and_7 ^ xor_36; and_14 = xor_39 & xor_37; and_15 = xor_45 & xor_39;
and_16 = xor_39 & xor_44; and_17 = xor_45 & xor_44; xor_47 = and_15 ^ xor_27;
xor_48 = and_14 ^ xor_24; xor_49 = x[7] ^ and_17; xor_50 = xor_29 ^ xor_47;
xor_51 = xor_3 ^ xor_48; xor_52 = and_17 ^ xor_31; xor_53 = xor_47 ^ xor_36;
xor_54 = and_8 ^ xor_51; xor_55 = and_12 ^ and_14; xor_56 = xor_14 ^ xor_55;
xor_57 = xor_27 ^ and_14; xor_58 = xor_23 ^ xor_57; and_18 = xor_37 & xor_54;
and_19 = xor_37 & xor_53; and_20 = xor_53 & xor_54; and_21 = xor_37 & xor_52;
and_22 = xor_52 & xor_53; and_23 = xor_52 & xor_54; xor_59 = and_18 ^ xor_25;
xor_60 = and_19 ^ and_21; xor_61 = and_16 ^ and_23; xor_62 = xor_59 ^ xor_49;
xor_63 = xor_59 ^ xor_61; xor_64 = xor_28 ^ xor_60; xor_65 = and_20 ^ xor_62;
xor_66 = xor_50 ^ xor_64; xor_67 = and_22 ^ xor_40; xor_68 = xor_35 ^ xor_63;
xor_69 = xor_24 ^ xor_65; xor_70 = and_19 ^ xor_67; xor_71 = xor_33 ^ xor_69;
xor_72 = xor_48 ^ xor_68; xor_73 = xor_38 ^ xor_70; xor_74 = xor_24 ^ xor_73;
xor_75 = and_17 ^ xor_70; xor_76 = xor_39 ^ xor_60; not_0 = ~xor_72;
and_24 = not_0 & xor_71; and_25 = not_0 & xor_66; and_26 = xor_71 & xor_66;
and_27 = xor_66 & xor_74; xor_77 = xor_63 ^ and_24; xor_78 = xor_77 ^ xor_56;
xor_79 = and_25 ^ xor_46; xor_80 = xor_75 ^ xor_79; xor_81 = and_27 ^ xor_76;
xor_82 = xor_65 ^ and_26; xor_83 = xor_82 ^ xor_58; not_1 = ~xor_78;
y[0] = not_1; y[1] = xor_80; y[2] = xor_81; y[3] = xor_83; y[4] = not_0;
y[5] = xor_71; y[6] = xor_66; y[7] = xor_74;

```

C.28 YSH_{8,3}: 8-bit degree 7, $\delta = 14$, $L = 72$, 18 AND (8-bit, Category B)

- **Profile:** $\delta = 14$, $L = 72$ (NL = 92), $\deg(S) = 7$, $\mathcal{D}(S) = [4^1, 5^2, 6^{60}, 7^{192}]$, FP = 0, SAC deviation 0.1094.
- **Cost:** 18 AND, 71 XOR, 2 NOT, AND depth $D = 5$ (unmasked NanGate 45 nm: $133.53 \mu\text{m}^2$ / 167.3 GE, delay 1084.7 ps, logic depth 18; pipelined $d = 2$ DOM-indep: 160 DFF_X1 [72 gadget + 88 sync], $1142.74 \mu\text{m}^2$ / 1432.0 GE, stage delay 431.6 ps).
- **Remark:** The 18-AND member of the Stage-2 AND-erasure family of YSH_{8,2} (Table 8; Section 6.5): seeded with YSH_{8,2} itself as an archive elite (28/62, $D = 6$),

AND erasure removed ten AND gates and one level (3, 5, 3, 4, 3 AND gates per level) and moved the profile from $(\delta, L) = (10, 72)$ to $(14, 72)$ while keeping degree 7, $\beta = 38$, and no fixed point. The 63 components of degree at most 6 form a 6-dimensional subspace ($k = 6$ split $[4^1, 5^2, 6^{60}, 7^{192}]$, nested over the 2-dimensional subspace of the three components of degree at most 5); the inverse has $\mathcal{D}(S^{-1}) = [5^1, 6^{62}, 7^{192}]$. As a pipelined $d = 2$ DOM-indep module it is the smallest 8-bit circuit of the benchmark set (1, 142.74 μm^2 , 72 gadget flip-flops), although its unmasked area (133.53 μm^2) exceeds that of YSH_{8,2} because of its ten additional XOR gates. Stage-4 post-processing reduced the archived 18/73/2 netlist to 18/71/2; the inverse (Tier 2 backward unrolling) is one level deeper than S .

- **Inverse** (YSH_{8,3}⁻¹): $\deg(S^{-1}) = 7$, $\mathcal{D}(S^{-1}) = [5^1, 6^{62}, 7^{192}]$; cost: 18 AND, 77 XOR, 8 NOT, AND depth $D = 6$ (Tier 2: Structural backward unrolling with linear-span closure (one level deeper than the forward circuit); unmasked NanGate 45 nm: 146.30 μm^2 / 183.3 GE, delay 1389.2 ps, logic depth 28; pipelined $d = 2$ DOM-indep: 274 DFF_X1 [72 gadget + 202 sync], 1680.59 μm^2 / 2106.0 GE, stage delay 438.1 ps).

Lookup table (LUT[0...255]):

```
0xcb, 0xa8, 0xb4, 0x5a, 0x79, 0x77, 0xc2, 0xe6, 0x67, 0x90, 0xfd, 0x6b, 0xf8, 0xb9, 0x8c, 0x5c
0x20, 0x08, 0x24, 0x3b, 0x59, 0x3e, 0xdc, 0x75, 0xee, 0x21, 0x8b, 0xff, 0x4a, 0x33, 0x22, 0x89
0xf2, 0x66, 0x74, 0xc5, 0xf5, 0x48, 0x2f, 0x46, 0x01, 0x99, 0xcd, 0xc4, 0xc0, 0x94, 0xdb, 0xb0
0xa7, 0x54, 0x11, 0xea, 0xbb, 0xa0, 0xbd, 0x09, 0x02, 0xd6, 0x8a, 0xd2, 0x81, 0x56, 0x78, 0x38
0xe2, 0xb8, 0x7d, 0xbe, 0x5f, 0xc7, 0x62, 0x1e, 0x53, 0x64, 0x5d, 0x40, 0xde, 0x44, 0x91, 0x93
0xf9, 0xa2, 0x50, 0xad, 0x2c, 0x37, 0x03, 0x2e, 0xec, 0x7c, 0xae, 0xf0, 0x73, 0xb6, 0x16, 0xf1
0xb3, 0x17, 0x69, 0x86, 0x00, 0xba, 0xc3, 0x35, 0x55, 0xd1, 0x10, 0x34, 0x6d, 0x82, 0x07, 0x88
0xf3, 0xd4, 0x9d, 0x51, 0x60, 0x36, 0x8f, 0x80, 0xd3, 0x2a, 0x14, 0x42, 0x29, 0xda, 0x6f, 0xa5
0xfa, 0x25, 0x1c, 0xef, 0x72, 0xc6, 0x31, 0x1b, 0xa6, 0xc9, 0xbf, 0x4f, 0x5b, 0x30, 0xe9, 0x7a
0xfb, 0xe3, 0xa1, 0x19, 0x04, 0x3d, 0xdf, 0xaa, 0xe7, 0x0c, 0xb5, 0x2d, 0xfc, 0xbc, 0x71, 0xc8
0xac, 0xfe, 0x9f, 0x9b, 0xf7, 0x0e, 0xdd, 0xb2, 0x85, 0x83, 0xca, 0x05, 0xe1, 0x15, 0x18, 0x4c
0x23, 0xd0, 0x70, 0x13, 0x95, 0xb1, 0x47, 0xcc, 0x3a, 0x0b, 0x39, 0x9e, 0x84, 0x7f, 0x97, 0x52
0x2b, 0x1a, 0xa4, 0xab, 0x4e, 0xf6, 0x76, 0x6e, 0x7e, 0x1d, 0xed, 0xb7, 0x68, 0xc1, 0xaf, 0x98
0xa9, 0x43, 0xd0, 0xe0, 0x3c, 0xe5, 0x41, 0x45, 0x61, 0x32, 0x1f, 0x8d, 0x8e, 0x92, 0xa3, 0x87
0x3f, 0x06, 0x6a, 0x63, 0x65, 0xd5, 0x6c, 0x58, 0xd8, 0xe4, 0x12, 0x49, 0xeb, 0x7b, 0x4b, 0x4d
0x57, 0xcf, 0x26, 0xd9, 0xce, 0xf4, 0x0f, 0x27, 0x9c, 0x96, 0x5e, 0x28, 0xe8, 0xa0, 0x9a, 0xd7
```

Netlist (18 AND, 71 XOR, 2 NOT, $D = 5$):

```
xor_0 = x[0] ^ x[4]; and_0 = x[4] & x[5]; xor_1 = x[3] ^ x[5];
and_1 = x[4] & x[6]; xor_2 = x[5] ^ x[6]; xor_3 = x[2] ^ x[6];
and_2 = x[6] & x[7]; xor_4 = x[1] ^ x[7]; xor_5 = xor_0 ^ and_0;
xor_6 = xor_1 ^ xor_2; xor_7 = xor_3 ^ and_2; xor_8 = xor_4 ^ and_1;
xor_9 = x[7] ^ xor_5; and_3 = xor_5 & xor_6; xor_10 = x[4] ^ xor_6;
xor_11 = xor_6 ^ xor_5; and_4 = xor_5 & xor_7; and_5 = xor_7 & xor_6;
xor_12 = x[6] ^ xor_7; and_6 = xor_5 & xor_8; and_7 = xor_8 & xor_7;
xor_13 = xor_8 ^ xor_6; xor_14 = x[5] ^ xor_8; xor_15 = and_3 ^ and_5;
xor_16 = and_6 ^ and_4; xor_17 = and_4 ^ and_7; xor_18 = and_3 ^ xor_13;
xor_19 = xor_12 ^ xor_16; xor_20 = xor_9 ^ xor_17; xor_21 = xor_10 ^ xor_18;
xor_22 = xor_14 ^ xor_15; xor_23 = xor_19 ^ xor_5; xor_24 = xor_8 ^ xor_19;
xor_25 = xor_19 ^ xor_20; not_0 = ~xor_21; and_8 = xor_22 & xor_19;
xor_26 = xor_22 ^ xor_20; xor_27 = xor_7 ^ xor_22; xor_28 = xor_11 ^ xor_23;
xor_29 = not_0 ^ xor_20; xor_30 = xor_20 ^ xor_28; xor_31 = xor_24 ^ xor_26;
xor_32 = xor_27 ^ and_8; and_9 = xor_31 & xor_28; xor_33 = xor_5 ^ xor_31;
xor_34 = xor_29 ^ xor_25; xor_35 = xor_31 ^ xor_32; xor_36 = xor_22 ^ xor_32;
xor_37 = xor_32 ^ xor_34; and_10 = xor_28 & xor_34; xor_38 = xor_34 ^ xor_31;
xor_39 = and_9 ^ xor_37; xor_40 = xor_35 ^ and_10; xor_41 = and_9 ^ and_10;
xor_42 = xor_36 ^ xor_39; xor_43 = xor_30 ^ xor_38; xor_44 = xor_33 ^ xor_41;
xor_45 = not_0 ^ xor_40; xor_46 = xor_32 ^ xor_43; and_11 = xor_43 & xor_44;
and_12 = xor_44 & xor_31; xor_47 = xor_28 ^ xor_44; and_13 = xor_45 & xor_43;
and_14 = xor_45 & xor_31; xor_48 = xor_42 ^ xor_45; xor_49 = xor_38 ^ and_14;
xor_50 = xor_46 ^ and_12; xor_51 = xor_47 ^ and_11; xor_52 = xor_43 ^ xor_49;
xor_53 = xor_48 ^ and_13; xor_54 = xor_45 ^ xor_50; xor_55 = xor_50 ^ xor_49;
xor_56 = xor_50 ^ xor_51; xor_57 = xor_31 ^ xor_51; and_15 = xor_51 & xor_49;
xor_58 = xor_51 ^ xor_49; xor_59 = xor_44 ^ xor_53; and_16 = xor_50 & xor_53;
xor_60 = xor_51 ^ xor_53; and_17 = xor_49 & xor_53; xor_61 = xor_55 ^ and_15;
xor_62 = xor_55 ^ and_16; xor_63 = and_15 ^ xor_60; xor_64 = xor_56 ^ and_17;
xor_65 = xor_52 ^ xor_61; xor_66 = xor_59 ^ xor_62; xor_67 = xor_57 ^ xor_63;
xor_68 = xor_54 ^ xor_64; xor_69 = xor_53 ^ xor_66; not_1 = ~xor_68;
xor_70 = xor_69 ^ xor_58; y[0] = xor_58; y[1] = xor_49; y[2] = xor_70;
y[3] = not_1; y[4] = xor_65; y[5] = xor_66; y[6] = xor_67; y[7] = xor_56;
```

Inverse lookup table (LUT_{S⁻¹}[0...255]):

```

0x64, 0x28, 0x38, 0x56, 0x94, 0xab, 0xe1, 0x6e, 0x11, 0x37, 0xfd, 0xb9, 0x99, 0xb1, 0xa5, 0xf6
0x6a, 0x32, 0xea, 0xb3, 0x7a, 0xad, 0x5e, 0x61, 0xae, 0x93, 0xc1, 0x87, 0x82, 0xc9, 0x47, 0xda
0x10, 0x19, 0x1e, 0xb0, 0x12, 0x81, 0xf2, 0xf7, 0xfb, 0x7c, 0x79, 0xc0, 0x54, 0x9b, 0x57, 0x26
0x8d, 0x86, 0xd9, 0x1d, 0x6b, 0x67, 0x75, 0x55, 0x3f, 0xba, 0xb8, 0x13, 0xd4, 0x95, 0x15, 0xe0
0x4b, 0xd6, 0x7b, 0xd1, 0x4d, 0xd7, 0x27, 0xb6, 0x25, 0xeb, 0x1c, 0xee, 0xaf, 0xef, 0xc4, 0x8b
0x52, 0x73, 0xbf, 0x48, 0x31, 0x68, 0x3d, 0xf0, 0xe7, 0x14, 0x03, 0x8c, 0x0f, 0x4a, 0xfa, 0x44
0x74, 0xd8, 0x46, 0xe3, 0x49, 0xe4, 0x21, 0x08, 0xcc, 0x62, 0xe2, 0x0b, 0xe6, 0x6c, 0xc7, 0x7e
0xb2, 0x9e, 0x84, 0x5c, 0x22, 0x17, 0xc6, 0x05, 0x3e, 0x04, 0x8f, 0xed, 0x59, 0x42, 0xc8, 0xbd
0x77, 0x3c, 0x6d, 0xa9, 0xbc, 0xa8, 0x63, 0xdf, 0x6f, 0x1f, 0x3a, 0x1a, 0x0e, 0xdb, 0xdc, 0x76
0x09, 0x4e, 0xdd, 0x4f, 0x2d, 0xb4, 0xf9, 0xbe, 0xcf, 0x29, 0xfe, 0xa3, 0xf8, 0x72, 0xbb, 0xa2
0x35, 0x92, 0x51, 0xde, 0xc2, 0x7f, 0x88, 0x30, 0x01, 0xd0, 0x97, 0xc3, 0xa0, 0x53, 0x5a, 0xce
0x2f, 0xb5, 0xa7, 0x60, 0x02, 0x9a, 0x5d, 0xcb, 0x41, 0x0d, 0x65, 0x34, 0x9d, 0x36, 0x43, 0x8a
0x2c, 0xcd, 0x06, 0x66, 0x2b, 0x23, 0x85, 0x45, 0x9f, 0x89, 0xaa, 0x00, 0xb7, 0x2a, 0xf4, 0xf1
0xd2, 0x69, 0x3b, 0x78, 0x71, 0xe5, 0x39, 0xff, 0xe8, 0xf3, 0x7d, 0x2e, 0x16, 0xa6, 0x4c, 0x96
0xd3, 0xac, 0x40, 0x91, 0xe9, 0xd5, 0x07, 0x98, 0xfc, 0x8e, 0x33, 0xec, 0x58, 0xca, 0x18, 0x83
0x5b, 0x5f, 0x20, 0x70, 0xf5, 0x24, 0xc5, 0xa4, 0x0c, 0x50, 0x80, 0x90, 0x9c, 0x0a, 0xa1, 0x1b

```

Inverse netlist (18 AND, 77 XOR, 8 NOT, $D = 6$):

```

xor_0 = x[0] ^ x[5]; xor_1 = x[2] ^ x[7]; xor_2 = x[0] ^ x[1];
xor_3 = x[2] ^ xor_0; xor_4 = x[6] ^ xor_3; xor_5 = x[7] ^ xor_2;
xor_6 = x[4] ^ xor_5; xor_7 = x[4] ^ x[6]; and_0 = xor_2 & x[1];
and_1 = xor_5 & xor_3; and_2 = x[1] & xor_3; xor_8 = x[3] ^ and_2;
xor_9 = and_0 ^ xor_4; xor_10 = and_1 ^ xor_1; xor_11 = xor_8 ^ xor_2;
xor_12 = and_0 ^ xor_6; xor_13 = x[1] ^ xor_9; xor_14 = and_0 ^ xor_8;
not_0 = ~xor_11; and_3 = xor_12 & xor_10; and_4 = xor_10 & xor_9;
and_5 = not_0 & xor_12; and_6 = not_0 & xor_9; xor_15 = x[4] ^ and_6;
xor_16 = xor_8 ^ xor_15; xor_17 = and_1 ^ and_3; xor_18 = x[1] ^ xor_17;
xor_19 = and_4 ^ xor_16; xor_20 = xor_1 ^ xor_18; xor_21 = x[0] ^ xor_19;
xor_22 = and_4 ^ xor_18; xor_23 = and_3 ^ xor_1; xor_24 = xor_19 ^ xor_20;
xor_25 = x[1] ^ xor_23; xor_26 = x[0] ^ xor_20; xor_27 = and_6 ^ xor_13;
xor_28 = and_5 ^ xor_23; xor_29 = xor_10 ^ and_4; and_7 = xor_9 & xor_26;
and_8 = xor_26 & xor_27; xor_30 = x[6] ^ and_7; xor_31 = and_0 ^ and_8;
xor_32 = and_5 ^ xor_30; xor_33 = xor_30 ^ xor_0; xor_34 = x[0] ^ xor_32;
xor_35 = xor_17 ^ xor_34; xor_36 = x[7] ^ xor_33; xor_37 = and_1 ^ xor_31;
xor_38 = xor_36 ^ xor_37; xor_39 = x[5] ^ xor_31; xor_40 = and_7 ^ xor_22;
xor_41 = and_6 ^ and_8; xor_42 = x[6] ^ xor_39; xor_43 = and_8 ^ xor_15;
xor_44 = xor_22 ^ xor_42; xor_45 = xor_31 ^ xor_21; xor_46 = xor_40 ^ xor_41;
xor_47 = xor_31 ^ xor_24; xor_48 = xor_0 ^ xor_46; xor_49 = and_5 ^ xor_48;
xor_50 = and_6 ^ xor_34; xor_51 = xor_14 ^ xor_50; not_1 = ~xor_47;
not_2 = ~xor_51; not_3 = ~xor_45; and_9 = xor_38 & not_1;
and_10 = xor_38 & xor_49; and_11 = not_2 & not_3; xor_52 = and_11 ^ xor_32;
xor_53 = and_9 ^ xor_16; xor_54 = x[2] ^ xor_53; xor_55 = xor_52 ^ xor_21;
xor_56 = xor_35 ^ xor_54; xor_57 = xor_39 ^ xor_53; not_4 = ~xor_55;
and_12 = xor_38 & not_4; and_13 = not_4 & not_1; and_14 = xor_49 & not_4;
xor_58 = and_10 ^ and_12; xor_59 = xor_52 ^ xor_58; xor_60 = and_12 ^ and_14;
xor_61 = xor_31 ^ xor_59; xor_62 = xor_33 ^ xor_60; xor_63 = and_13 ^ xor_44;
xor_64 = xor_43 ^ xor_62; xor_65 = and_9 ^ xor_63; xor_66 = xor_8 ^ xor_65;
xor_67 = xor_25 ^ xor_64; xor_68 = and_5 ^ xor_60; xor_69 = xor_29 ^ xor_68;
xor_70 = xor_45 ^ xor_58; xor_71 = xor_24 ^ xor_59; not_5 = ~xor_66;
not_6 = ~xor_71; and_15 = xor_56 & not_5; and_16 = xor_56 & xor_61;
and_17 = xor_61 & xor_67; xor_72 = and_15 ^ xor_28; xor_73 = xor_57 ^ xor_72;
xor_74 = and_16 ^ xor_7; xor_75 = xor_74 ^ xor_69; xor_76 = and_17 ^ xor_70;
not_7 = ~xor_76; y[0] = xor_73; y[1] = xor_75; y[2] = not_7; y[3] = not_6;
y[4] = xor_56; y[5] = not_5; y[6] = xor_61; y[7] = xor_67;

```

C.29 8-bit AES class at $D = 3$ (8-bit, Category A)

- **Profile:** $\delta = 4$, $L = 32$ (NL = 112), $\deg(S) = 7$, $\mathcal{D}(S) = [7^{255}]$, FP = 3, SAC deviation 0.0625.
- **Cost:** 42 AND, 101 XOR, 4 NOT, AND depth $D = 3$ (unmasked NanGate 45 nm: $208.01 \mu\text{m}^2 / 260.7 \text{ GE}$, delay 1516.4 ps, logic depth 24; pipelined $d = 2$ DOM-indep: 216 DFF_X1 [168 gadget + 48 sync], $1748.15 \mu\text{m}^2 / 2190.7 \text{ GE}$, stage delay 574.7 ps).
- **Remark:** Certified affine equivalent to the FIPS-197 AES S-box ($\sim_{\text{aff}}$ AES, with $A = I_8$, $a = 0x00$, $B = [0x08, 0x04, 0x02, 0x01, 0x80, 0x40, 0x20, 0x10]$, $b = 0x0f$); replacing the output assignments by $y[7..0] = (\text{xor_94}, \text{not_1}, \text{not_0}, \text{xor_100}, \text{xor_98}, \text{xor_95}, \sim\text{xor_99}, \sim\text{xor_96})$ yields the exact FIPS-197 AES lookup table at the identical gate cost (42 AND, 101 XOR, 4 NOT, $D = 3$). Depth 3 is the minimum for degree 7 (Section 3). The depth-4 circuit 33/84/4 of [BDM26] uses nine fewer AND gates and 17 fewer XOR gates.

- **Inverse** (S^{-1}): $\deg(S^{-1}) = 7$, $\mathcal{D}(S^{-1}) = [7^{255}]$; cost: 42 AND, 106 XOR, 4 NOT, AND depth $D = 3$ (Tier 1: 3-stage affine-folded core reuse; unmasked NanGate 45 nm: $215.99 \mu\text{m}^2$ / 270.7 GE, delay 1337.5 ps, logic depth 22; pipelined $d = 2$ DOM-indep: 216 DFF_X1 [168 gadget + 48 sync], $1764.11 \mu\text{m}^2$ / 2210.7 GE, stage delay 576.4 ps).

Lookup table (LUT[0...255]):

```
0x63, 0xec, 0xe1, 0xe2, 0xfb, 0x62, 0x60, 0x35, 0xcf, 0x07, 0x61, 0x42, 0xf8, 0xb1, 0x52, 0xe9
0x3a, 0x1b, 0x36, 0xe4, 0xfa, 0xa6, 0x21, 0xff, 0x54, 0xbd, 0x5b, 0x50, 0x9c, 0x5d, 0xeb, 0x3f
0xd1, 0xf4, 0x93, 0x49, 0xc9, 0xc0, 0xf1, 0x3c, 0xcd, 0x55, 0x75, 0xf7, 0xe7, 0xbe, 0xc7, 0x85
0x0d, 0x31, 0x43, 0x33, 0x8e, 0x99, 0x05, 0x9a, 0x01, 0x8b, 0x1f, 0x7b, 0x72, 0x41, 0xdb, 0xe5
0x06, 0x13, 0x4c, 0x8a, 0x82, 0x68, 0xaa, 0x5f, 0xab, 0xc2, 0xb9, 0xd3, 0x46, 0x73, 0x40, 0x1d
0xa3, 0xb7, 0x0f, 0x74, 0x4f, 0xfc, 0xd7, 0xa2, 0x6a, 0x32, 0xd8, 0xc6, 0x2a, 0x2c, 0xae, 0x30
0xbf, 0x70, 0x5a, 0xf2, 0x23, 0x24, 0xc3, 0x15, 0x25, 0xf6, 0x0b, 0xe0, 0xaf, 0xcc, 0x90, 0x5e
0xa7, 0x53, 0x2f, 0x10, 0x9b, 0x94, 0xce, 0xf5, 0xdc, 0xd9, 0xba, 0x47, 0x8f, 0xf0, 0xf3, 0xbb
0x34, 0x0c, 0x83, 0x7c, 0xa0, 0x91, 0x2d, 0x81, 0x3d, 0x51, 0xe8, 0xc4, 0x6d, 0xa4, 0x86, 0xe3
0x6f, 0x17, 0x20, 0xbc, 0x4b, 0x4a, 0x9f, 0x1e, 0x29, 0x78, 0xde, 0x8d, 0xb8, 0xa8, 0x02, 0xb2
0x7f, 0xcb, 0xca, 0x0a, 0x26, 0x09, 0x4d, 0xac, 0x3b, 0xb3, 0x5c, 0x6b, 0x97, 0x95, 0x7d, 0xe6
0x71, 0x3e, 0xc1, 0x64, 0x14, 0xb5, 0x28, 0x56, 0x6c, 0xa9, 0xfd, 0x7a, 0x65, 0xea, 0x58, 0x0e
0xda, 0xee, 0x45, 0x48, 0x8c, 0x59, 0xdd, 0x39, 0x7e, 0xb4, 0xed, 0x80, 0x22, 0xd4, 0x12, 0x1a
0xef, 0xc8, 0xd5, 0x69, 0x2e, 0x03, 0xf9, 0x08, 0x67, 0xc5, 0xa1, 0xd6, 0x19, 0x37, 0x84, 0x98
0x77, 0xfe, 0x9e, 0x87, 0x66, 0xb6, 0x18, 0x9d, 0x92, 0x88, 0x11, 0x76, 0x38, 0xa5, 0x4e, 0xb0
0x1c, 0x57, 0x16, 0x04, 0xd0, 0x79, 0x2b, 0x6e, 0x27, 0x96, 0x44, 0x00, 0xdf, 0xad, 0xd2, 0x89
```

Netlist (42 AND, 101 XOR, 4 NOT, $D = 3$):

```
xor_0 = x[0] ^ x[5]; xor_1 = x[0] ^ x[6]; xor_2 = x[2] ^ x[7];
xor_3 = x[4] ^ x[7]; xor_4 = x[3] ^ xor_0; xor_5 = x[7] ^ xor_0;
xor_6 = x[2] ^ xor_1; xor_7 = x[1] ^ xor_2; xor_8 = xor_1 ^ xor_3;
xor_9 = x[4] ^ xor_4; xor_10 = x[6] ^ xor_4; xor_11 = xor_1 ^ xor_5;
xor_12 = xor_5 ^ xor_6; and_0 = xor_8 & xor_5; xor_13 = xor_8 ^ xor_5;
xor_14 = xor_2 ^ xor_9; xor_15 = xor_7 ^ xor_9; and_1 = xor_9 & xor_6;
xor_16 = xor_9 ^ xor_6; xor_17 = xor_9 ^ xor_8; xor_18 = xor_7 ^ xor_10;
xor_19 = xor_1 ^ xor_14; xor_20 = xor_14 ^ xor_6; and_2 = xor_15 & xor_14;
and_3 = xor_18 & xor_1; xor_21 = xor_18 ^ xor_1; xor_22 = xor_18 ^ xor_15;
xor_23 = xor_18 ^ xor_8; xor_24 = x[1] ^ xor_13; and_4 = xor_17 & xor_12;
xor_25 = xor_21 ^ x[1]; xor_26 = xor_11 ^ xor_20; xor_27 = xor_24 ^ xor_16;
and_5 = xor_7 & xor_20; xor_28 = and_4 ^ and_0; xor_29 = and_4 ^ and_1;
xor_30 = xor_25 ^ xor_13; and_6 = xor_22 & xor_19; and_7 = xor_23 & xor_11;
xor_31 = xor_23 ^ xor_7; xor_32 = and_6 ^ and_3; xor_33 = and_6 ^ and_2;
and_8 = xor_31 & xor_26; xor_34 = and_8 ^ and_7; xor_35 = and_8 ^ and_5;
xor_36 = xor_33 ^ xor_34; xor_37 = xor_29 ^ xor_34; xor_38 = xor_34 ^ xor_35;
xor_39 = xor_36 ^ xor_27; xor_40 = xor_37 ^ x[1]; xor_41 = xor_32 ^ xor_38;
xor_42 = xor_28 ^ xor_38; and_9 = xor_6 & xor_39; and_10 = xor_9 & xor_39;
and_11 = xor_40 & xor_39; and_12 = xor_14 & xor_40; and_13 = xor_15 & xor_40;
xor_43 = xor_39 ^ xor_40; xor_44 = xor_41 ^ xor_30; xor_45 = xor_42 ^ xor_25;
and_14 = xor_5 & xor_44; and_15 = xor_8 & xor_44; xor_46 = xor_44 ^ xor_39;
and_16 = xor_45 & xor_44; and_17 = xor_1 & xor_45; and_18 = xor_18 & xor_45;
xor_47 = xor_45 ^ xor_40; xor_48 = xor_44 ^ xor_45; and_19 = xor_20 & xor_43;
and_20 = xor_7 & xor_43; and_21 = xor_12 & xor_46; and_22 = xor_17 & xor_46;
and_23 = xor_47 & xor_46; and_24 = xor_19 & xor_47; and_25 = xor_22 & xor_47;
xor_49 = xor_48 ^ xor_43; and_26 = xor_11 & xor_48; and_27 = xor_23 & xor_48;
xor_50 = and_21 ^ and_14; xor_51 = and_21 ^ and_9; xor_52 = and_22 ^ and_15;
xor_53 = and_22 ^ and_10; xor_54 = and_23 ^ and_16; xor_55 = and_23 ^ and_11;
xor_56 = and_24 ^ and_17; xor_57 = and_24 ^ and_12; xor_58 = and_25 ^ and_18;
xor_59 = and_25 ^ and_13; and_28 = xor_26 & xor_49; and_29 = xor_31 & xor_49;
xor_60 = xor_54 ^ xor_49; xor_61 = xor_55 ^ xor_43; xor_62 = and_28 ^ and_26;
xor_63 = and_28 ^ and_19; xor_64 = and_29 ^ and_27; xor_65 = and_29 ^ and_20;
xor_66 = xor_60 ^ xor_61; xor_67 = xor_57 ^ xor_62; xor_68 = xor_51 ^ xor_62;
xor_69 = xor_62 ^ xor_63; xor_70 = xor_59 ^ xor_64; xor_71 = xor_53 ^ xor_64;
xor_72 = xor_64 ^ xor_65; and_30 = xor_67 & xor_60; and_31 = xor_68 & xor_60;
xor_73 = xor_56 ^ xor_69; xor_74 = xor_50 ^ xor_69; and_32 = xor_70 & xor_60;
and_33 = xor_71 & xor_60; xor_75 = xor_58 ^ xor_72; xor_76 = xor_52 ^ xor_72;
and_34 = xor_73 & xor_61; xor_77 = xor_73 ^ xor_67; and_35 = xor_74 & xor_61;
xor_78 = xor_74 ^ xor_68; and_36 = xor_75 & xor_61; xor_79 = xor_75 ^ xor_70;
and_37 = xor_76 & xor_61; xor_80 = xor_76 ^ xor_71; and_38 = xor_77 & xor_66;
and_39 = xor_78 & xor_66; and_40 = xor_79 & xor_66; and_41 = xor_80 & xor_66;
xor_81 = and_38 ^ and_34; xor_82 = and_38 ^ and_30; xor_83 = and_39 ^ and_35;
xor_84 = and_39 ^ and_31; xor_85 = and_40 ^ and_36; xor_86 = and_40 ^ and_32;
xor_87 = and_41 ^ and_37; xor_88 = and_41 ^ and_33; xor_89 = xor_81 ^ xor_83;
xor_90 = xor_85 ^ xor_82; xor_91 = xor_85 ^ xor_83; xor_92 = xor_87 ^ xor_81;
xor_93 = xor_88 ^ xor_84; xor_94 = xor_86 ^ xor_88; not_0 = ~xor_89;
xor_95 = xor_84 ^ xor_90; xor_96 = xor_82 ^ xor_92; xor_97 = xor_92 ^ xor_93;
not_1 = ~xor_93; not_2 = ~xor_95; xor_98 = xor_86 ^ xor_97;
xor_99 = xor_83 ^ xor_97; not_3 = ~xor_98; xor_100 = xor_91 ^ xor_98;
y[0] = not_3; y[1] = not_2; y[2] = xor_99; y[3] = xor_96; y[4] = xor_94;
y[5] = not_1; y[6] = not_0; y[7] = xor_100;
```

Inverse lookup table ($\text{LUT}_{S^{-1}}[0 \dots 255]$):

```

0xfb, 0x38, 0x9e, 0xd5, 0xf3, 0x36, 0x40, 0x09, 0xd7, 0xa5, 0xa3, 0x6a, 0x81, 0x30, 0xbf, 0x52
0x73, 0xea, 0xce, 0x41, 0xb4, 0x67, 0xf2, 0x91, 0xe6, 0xdc, 0xcf, 0x11, 0xf0, 0x4f, 0x97, 0x3a
0x92, 0x16, 0xcc, 0x64, 0x65, 0x68, 0xa4, 0xf8, 0xb6, 0x98, 0x5c, 0xf6, 0x5d, 0x86, 0xd4, 0x72
0x5f, 0x31, 0x59, 0x33, 0x80, 0x07, 0x12, 0xdd, 0xec, 0xc7, 0x10, 0xa8, 0x27, 0x88, 0xb1, 0x1f
0x4e, 0x3d, 0x0b, 0x32, 0xfa, 0xc2, 0x4c, 0x7b, 0xc3, 0x23, 0x95, 0x94, 0x42, 0xa6, 0xee, 0x54
0x1b, 0x89, 0x0e, 0x71, 0x18, 0x29, 0xb7, 0xf1, 0xbe, 0xc5, 0x62, 0x1a, 0xaa, 0x1d, 0x6f, 0x47
0x06, 0x0a, 0x05, 0x00, 0xb3, 0xbc, 0xe4, 0xd8, 0x45, 0xd3, 0x58, 0xab, 0xb8, 0x8c, 0xf7, 0x90
0x61, 0xb0, 0x3c, 0x4d, 0x53, 0x2a, 0xeb, 0xe0, 0x99, 0xf5, 0xbb, 0x3b, 0x83, 0xae, 0xc8, 0xa0
0xcb, 0x87, 0x44, 0x82, 0xde, 0x2f, 0x8e, 0xe3, 0xe9, 0xff, 0x43, 0x39, 0xc4, 0x9b, 0x34, 0x7c
0x6e, 0x85, 0xe8, 0x22, 0x75, 0xad, 0xf9, 0xac, 0xdf, 0x35, 0x37, 0x74, 0x1c, 0xe7, 0xe2, 0x96
0x84, 0xda, 0x57, 0x50, 0x8d, 0xed, 0x15, 0x70, 0x9d, 0xb9, 0x46, 0x48, 0xa7, 0xfd, 0x5e, 0x6c
0xef, 0x0d, 0x9f, 0xa9, 0xc9, 0xb5, 0xe5, 0x51, 0x9c, 0x4a, 0x7a, 0x7f, 0x93, 0x19, 0x2d, 0x60
0x25, 0xb2, 0x49, 0x66, 0x8b, 0xd9, 0x5b, 0x2e, 0xd1, 0x24, 0xa2, 0xa1, 0x6d, 0x28, 0x76, 0x08
0xf4, 0x20, 0xfe, 0x4b, 0xcd, 0xdb, 0x56, 0x5a, 0x79, 0xc0, 0x3e, 0x78, 0xc6, 0x9a, 0xfc
0x6b, 0x02, 0x03, 0x8f, 0x13, 0x3f, 0xaf, 0x2c, 0x8a, 0x0f, 0xbd, 0x1e, 0x01, 0xca, 0xc1, 0xd0
0x7d, 0x26, 0x63, 0x7e, 0x21, 0x77, 0x69, 0x2b, 0x0c, 0xd6, 0x14, 0x04, 0x55, 0xba, 0xe1, 0x17

```

Inverse netlist (42 AND, 106 XOR, 4 NOT, $D = 3$):

```

not_0 = ~x[0]; not_1 = ~x[1]; not_2 = ~x[5]; not_3 = ~x[6];
xor_0 = not_1 ^ not_2; xor_1 = x[2] ^ x[4]; xor_2 = xor_0 ^ not_3;
xor_3 = not_0 ^ x[3]; xor_4 = not_0 ^ x[2]; xor_5 = xor_1 ^ x[7];
xor_6 = not_2 ^ not_3; xor_7 = xor_2 ^ not_0; xor_8 = not_3 ^ x[4];
xor_9 = not_2 ^ x[4]; xor_10 = xor_3 ^ not_1; xor_11 = xor_3 ^ x[7];
xor_12 = xor_4 ^ xor_6; xor_13 = xor_1 ^ xor_2; xor_14 = xor_2 ^ x[7];
xor_15 = xor_9 ^ x[7]; xor_16 = x[2] ^ x[3]; xor_17 = xor_16 ^ x[7];
xor_18 = xor_17 ^ xor_6; xor_19 = xor_1 ^ xor_7; xor_20 = not_1 ^ x[7];
xor_21 = xor_20 ^ xor_4; xor_22 = xor_0 ^ x[2]; xor_23 = xor_8 ^ not_1;
xor_24 = xor_0 ^ x[4]; xor_25 = xor_3 ^ xor_9; xor_26 = xor_4 ^ not_3;
xor_27 = xor_0 ^ x[3]; xor_28 = xor_27 ^ xor_5; xor_29 = xor_10 ^ x[4];
xor_30 = xor_10 ^ xor_5; xor_31 = xor_1 ^ xor_6; and_0 = xor_11 & xor_2;
and_1 = xor_12 & not_0; and_2 = xor_13 & xor_14; and_3 = xor_15 & xor_8;
and_4 = xor_18 & xor_7; and_5 = xor_19 & xor_21; xor_32 = and_0 ^ and_4;
xor_33 = and_1 ^ and_4; and_6 = xor_22 & xor_23; and_7 = xor_24 & xor_25;
xor_34 = and_3 ^ and_6; xor_35 = and_2 ^ and_6; and_8 = xor_26 & xor_28;
xor_36 = and_7 ^ and_8; xor_37 = and_5 ^ and_8; xor_38 = xor_35 ^ xor_36;
xor_39 = xor_33 ^ xor_36; xor_40 = xor_36 ^ xor_37; xor_41 = xor_29 ^ xor_38;
xor_42 = xor_5 ^ xor_39; xor_43 = xor_34 ^ xor_40; xor_44 = xor_32 ^ xor_40;
and_9 = xor_12 & xor_41; and_10 = not_0 & xor_41; and_11 = xor_41 & xor_42;
and_12 = xor_14 & xor_42; and_13 = xor_13 & xor_42; xor_45 = xor_41 ^ xor_42;
xor_46 = xor_30 ^ xor_43; xor_47 = xor_31 ^ xor_44; and_14 = xor_11 & xor_46;
and_15 = xor_2 & xor_46; xor_48 = xor_46 ^ xor_41; and_16 = xor_46 & xor_47;
and_17 = xor_15 & xor_47; and_18 = xor_8 & xor_47; xor_49 = xor_47 ^ xor_42;
xor_50 = xor_46 ^ xor_47; and_19 = xor_21 & xor_45; and_20 = xor_19 & xor_45;
and_21 = xor_18 & xor_48; and_22 = xor_7 & xor_48; and_23 = xor_49 & xor_48;
and_24 = xor_23 & xor_49; and_25 = xor_22 & xor_49; xor_51 = xor_50 ^ xor_45;
and_26 = xor_25 & xor_50; and_27 = xor_24 & xor_50; xor_52 = and_14 ^ and_21;
xor_53 = and_9 ^ and_21; xor_54 = and_15 ^ and_22; xor_55 = and_10 ^ and_22;
xor_56 = and_16 ^ and_23; xor_57 = and_11 ^ and_23; xor_58 = and_17 ^ and_24;
xor_59 = and_12 ^ and_24; xor_60 = and_18 ^ and_25; xor_61 = and_13 ^ and_25;
and_28 = xor_28 & xor_51; and_29 = xor_26 & xor_51; xor_62 = xor_56 ^ xor_51;
xor_63 = xor_45 ^ xor_57; xor_64 = and_26 ^ and_28; xor_65 = and_19 ^ and_28;
xor_66 = and_27 ^ and_29; xor_67 = and_20 ^ and_29; xor_68 = xor_62 ^ xor_63;
xor_69 = xor_64 ^ xor_59; xor_70 = xor_64 ^ xor_53; xor_71 = xor_64 ^ xor_65;
xor_72 = xor_61 ^ xor_66; xor_73 = xor_55 ^ xor_66; xor_74 = xor_66 ^ xor_67;
and_30 = xor_69 & xor_62; and_31 = xor_70 & xor_62; xor_75 = xor_71 ^ xor_58;
xor_76 = xor_71 ^ xor_52; and_32 = xor_72 & xor_62; and_33 = xor_73 & xor_62;
xor_77 = xor_60 ^ xor_74; xor_78 = xor_54 ^ xor_74; and_34 = xor_75 & xor_63;
xor_79 = xor_75 ^ xor_69; and_35 = xor_76 & xor_63; xor_80 = xor_76 ^ xor_70;
and_36 = xor_77 & xor_63; xor_81 = xor_77 ^ xor_72; and_37 = xor_78 & xor_63;
xor_82 = xor_78 ^ xor_73; and_38 = xor_68 & xor_79; and_39 = xor_68 & xor_80;
and_40 = xor_68 & xor_81; and_41 = xor_68 & xor_82; xor_83 = and_37 ^ and_33;
xor_84 = and_34 ^ xor_83; xor_85 = and_31 ^ xor_84; xor_86 = and_38 ^ and_35;
xor_87 = and_30 ^ and_40; xor_88 = and_39 ^ xor_85; xor_89 = and_36 ^ and_41;
xor_90 = and_39 ^ and_32; xor_91 = xor_86 ^ xor_90; xor_92 = xor_87 ^ xor_91;
xor_93 = xor_85 ^ xor_86; xor_94 = and_38 ^ xor_88; xor_95 = and_36 ^ xor_84;
xor_96 = xor_95 ^ xor_87; xor_97 = and_30 ^ xor_88; xor_98 = and_35 ^ and_33;
xor_99 = and_39 ^ xor_98; xor_100 = and_40 ^ xor_99;
xor_101 = xor_100 ^ xor_89; xor_102 = and_32 ^ and_37;
xor_103 = xor_102 ^ xor_89; xor_104 = and_31 ^ and_39;
xor_105 = xor_83 ^ xor_104; y[0] = xor_92; y[1] = xor_93; y[2] = xor_94;
y[3] = xor_96; y[4] = xor_97; y[5] = xor_101; y[6] = xor_103; y[7] = xor_105;

```

C.30 8-bit AES S-box (FIPS-197) (8-bit, Category A)

- **Profile:** $\delta = 4$, $L = 32$ (NL = 112), $\deg(S) = 7$, $\mathcal{D}(S) = [7^{255}]$, FP = 0, SAC deviation 0.0625.
- **Cost:** 36 AND, 99 XOR, 4 NOT, AND depth $D = 4$ (unmasked NanGate 45 nm: $198.44 \mu\text{m}^2$ / 248.7 GE, delay 1148.6 ps, logic depth 21; pipelined $d = 2$ DOM-indep: 282 DFF_X1 [144 gadget + 138 sync], $1976.38 \mu\text{m}^2$ / 2476.7 GE, stage delay 371.0 ps).
- **Remark:** The exact FIPS-197 lookup table [Nat23]; 33/84/4 in [BDM26]: our circuit has three more AND and 15 more XOR gates.
- **Inverse** (S^{-1}): $\deg(S^{-1}) = 7$, $\mathcal{D}(S^{-1}) = [7^{255}]$; cost: 36 AND, 95 XOR, 4 NOT, AND depth $D = 4$ (Tier 1: 3-stage affine-folded core reuse; unmasked NanGate 45 nm: $192.05 \mu\text{m}^2$ / 240.7 GE, delay 1281.9 ps, logic depth 23; pipelined $d = 2$ DOM-indep: 276 DFF_X1 [144 gadget + 132 sync], $1936.48 \mu\text{m}^2$ / 2426.7 GE, stage delay 462.5 ps).

Lookup table (LUT[0...255]):

```
0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76
0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0
0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15
0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75
0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84
0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf
0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8
0x51, 0xa3, 0x40, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2
0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73
0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb
0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79
0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08
0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a
0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e
0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf
0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16
```

Netlist (36 AND, 99 XOR, 4 NOT, $D = 4$):

```
xor_0 = x[5] ^ x[7]; xor_1 = x[3] ^ x[4]; xor_2 = x[6] ^ xor_0;
xor_3 = x[0] ^ x[2]; xor_4 = xor_0 ^ xor_1; xor_5 = x[5] ^ xor_1;
xor_6 = x[0] ^ x[6]; xor_7 = x[3] ^ xor_2; xor_8 = x[2] ^ xor_2;
xor_9 = x[1] ^ xor_3; xor_10 = x[1] ^ x[2]; xor_11 = x[0] ^ xor_5;
xor_12 = x[6] ^ xor_3; xor_13 = x[4] ^ x[7]; xor_14 = xor_6 ^ xor_13;
xor_15 = x[0] ^ xor_0; xor_16 = xor_7 ^ xor_9; xor_17 = xor_4 ^ xor_9;
xor_18 = xor_3 ^ xor_4; xor_19 = xor_5 ^ xor_10; xor_20 = x[4] ^ x[6];
xor_21 = xor_1 ^ xor_8; xor_22 = x[7] ^ xor_10; xor_23 = xor_1 ^ xor_2;
and_0 = xor_11 & xor_12; and_1 = xor_14 & xor_15; and_2 = xor_7 & xor_8;
and_3 = xor_16 & xor_6; and_4 = xor_17 & xor_18; and_5 = xor_19 & xor_2;
and_6 = xor_20 & xor_21; and_7 = xor_22 & xor_23; and_8 = xor_4 & xor_1;
xor_24 = x[1] ^ and_8; xor_25 = x[2] ^ x[3]; xor_26 = x[7] ^ and_7;
xor_27 = and_5 ^ xor_25; xor_28 = and_0 ^ xor_24; xor_29 = and_4 ^ xor_24;
xor_30 = and_6 ^ xor_27; xor_31 = x[5] ^ and_1; xor_32 = xor_26 ^ xor_31;
xor_33 = and_3 ^ xor_26; xor_34 = xor_20 ^ xor_33; xor_35 = and_2 ^ and_5;
xor_36 = xor_28 ^ xor_35; xor_37 = xor_29 ^ xor_30; xor_38 = and_2 ^ xor_27;
xor_39 = xor_32 ^ xor_38; xor_40 = xor_30 ^ xor_34; xor_41 = xor_25 ^ xor_28;
xor_42 = xor_32 ^ xor_41; xor_43 = xor_29 ^ xor_34; and_9 = xor_36 & xor_37;
and_10 = xor_39 & xor_40; and_11 = xor_42 & xor_43; xor_44 = and_3 ^ xor_20;
xor_45 = and_4 ^ xor_25; xor_46 = and_0 ^ and_11; xor_47 = and_10 ^ xor_31;
xor_48 = xor_44 ^ xor_47; xor_49 = xor_45 ^ xor_46; xor_50 = and_2 ^ and_6;
xor_51 = and_9 ^ xor_50; xor_52 = xor_48 ^ xor_49; xor_53 = xor_49 ^ xor_51;
xor_54 = xor_48 ^ xor_51; and_12 = xor_37 & xor_52; and_13 = xor_36 & xor_52;
and_14 = xor_40 & xor_53; and_15 = xor_39 & xor_53; and_16 = xor_43 & xor_54;
and_17 = xor_42 & xor_54; xor_55 = and_14 ^ and_16; xor_56 = and_12 ^ and_16;
xor_57 = and_15 ^ and_17; xor_58 = and_13 ^ and_17; xor_59 = and_12 ^ and_14;
xor_60 = and_13 ^ and_15; xor_61 = xor_55 ^ xor_57; xor_62 = xor_56 ^ xor_58;
xor_63 = xor_59 ^ xor_60; and_18 = xor_15 & xor_55; and_19 = xor_14 & xor_55;
and_20 = xor_12 & xor_56; and_21 = xor_11 & xor_56; and_22 = xor_6 & xor_57;
and_23 = xor_16 & xor_57; and_24 = xor_18 & xor_58; and_25 = xor_17 & xor_58;
and_26 = xor_8 & xor_59; and_27 = xor_7 & xor_59; and_28 = xor_2 & xor_61;
and_29 = xor_19 & xor_61; and_30 = xor_21 & xor_60; and_31 = xor_20 & xor_60;
and_32 = xor_23 & xor_62; and_33 = xor_22 & xor_62; and_34 = xor_1 & xor_63;
and_35 = xor_4 & xor_63; xor_64 = and_19 ^ and_22; xor_65 = and_34 ^ xor_64;
xor_66 = and_20 ^ and_21; xor_67 = and_29 ^ and_33; xor_68 = and_18 ^ and_30;
```

```

xor_69 = and_28 ^ xor_66; xor_70 = and_24 ^ xor_67; xor_71 = and_32 ^ xor_65;
xor_72 = and_35 ^ xor_69; xor_73 = xor_65 ^ xor_68; xor_74 = and_26 ^ and_30;
xor_75 = and_31 ^ xor_74; xor_76 = and_27 ^ xor_70; xor_77 = xor_71 ^ xor_76;
not_0 = ~xor_77; xor_78 = and_33 ^ xor_72; xor_79 = xor_73 ^ xor_78;
not_1 = ~xor_79; xor_80 = and_20 ^ and_23; xor_81 = xor_70 ^ xor_75;
xor_82 = xor_80 ^ xor_81; xor_83 = and_25 ^ xor_66; xor_84 = xor_67 ^ xor_71;
xor_85 = xor_75 ^ xor_83; xor_86 = xor_84 ^ xor_85; xor_87 = and_23 ^ and_25;
xor_88 = xor_69 ^ xor_73; xor_89 = xor_87 ^ xor_88; xor_90 = and_22 ^ and_26;
xor_91 = xor_68 ^ xor_90; not_2 = ~xor_91; xor_92 = and_26 ^ and_27;
xor_93 = and_29 ^ and_34; xor_94 = xor_72 ^ xor_92; xor_95 = xor_93 ^ xor_94;
not_3 = ~xor_95; xor_96 = and_21 ^ and_25; xor_97 = and_27 ^ and_31;
xor_98 = xor_96 ^ xor_97; y[0] = not_0; y[1] = not_1; y[2] = xor_82;
y[3] = xor_86; y[4] = xor_89; y[5] = not_2; y[6] = not_3; y[7] = xor_98;

```

Inverse lookup table ($LUT_{S^{-1}}[0 \dots 255]$):

```

0x52, 0x09, 0x6a, 0xd5, 0x30, 0x36, 0xa5, 0x38, 0xbf, 0x40, 0xa3, 0x9e, 0x81, 0xf3, 0xd7, 0xfb
0x7c, 0xe3, 0x39, 0x82, 0x9b, 0x2f, 0xff, 0x87, 0x34, 0x8e, 0x43, 0x44, 0xc4, 0xde, 0xe9, 0xcb
0x54, 0x7b, 0x94, 0x32, 0xa6, 0xc2, 0x23, 0x3d, 0xee, 0x4c, 0x95, 0x0b, 0x42, 0xfa, 0xc3, 0x4e
0x08, 0x2e, 0xa1, 0x66, 0x28, 0xd9, 0x24, 0xb2, 0x76, 0x5b, 0xa2, 0x49, 0x6d, 0x8b, 0xd1, 0x25
0x72, 0xf8, 0xf6, 0x64, 0x86, 0x68, 0x98, 0x16, 0xd4, 0xa4, 0x5c, 0xcc, 0x5d, 0x65, 0xb6, 0x92
0x6c, 0x70, 0x48, 0x50, 0xfd, 0xed, 0xb9, 0xda, 0x5e, 0x15, 0x46, 0x57, 0xa7, 0x8d, 0x9d, 0x84
0x90, 0xd8, 0xab, 0x00, 0x8c, 0xbc, 0xd3, 0x0a, 0xf7, 0xe4, 0x58, 0x05, 0xb8, 0xb3, 0x45, 0x06
0xd0, 0x2c, 0x1e, 0x8f, 0xca, 0x3f, 0x0f, 0x02, 0xc1, 0xaf, 0xbd, 0x03, 0x01, 0x13, 0x8a, 0x6b
0x3a, 0x91, 0x11, 0x41, 0x4f, 0x67, 0xdc, 0xea, 0x97, 0xf2, 0xcf, 0xce, 0xf0, 0xb4, 0xe6, 0x73
0x96, 0xac, 0x74, 0x22, 0xe7, 0xad, 0x35, 0x85, 0xe2, 0xf9, 0x37, 0xe8, 0x1c, 0x75, 0xdf, 0x6e
0x47, 0xf1, 0x1a, 0x71, 0x1d, 0x29, 0xc5, 0x89, 0x6f, 0xb7, 0x62, 0x0e, 0xaa, 0x18, 0xbe, 0x1b
0xfc, 0x56, 0x3e, 0x4b, 0xc6, 0xd2, 0x79, 0x20, 0x9a, 0xdb, 0xc0, 0xfe, 0x78, 0xcd, 0x5a, 0xf4
0x1f, 0xdd, 0xa8, 0x33, 0x88, 0x07, 0xc7, 0x31, 0xb1, 0x12, 0x10, 0x59, 0x27, 0x80, 0xec, 0x5f
0x60, 0x51, 0x7f, 0xa9, 0x19, 0xb5, 0x4a, 0x0d, 0x2d, 0xe5, 0x7a, 0x9f, 0x93, 0xc9, 0x9c, 0xef
0xa0, 0xe0, 0x3b, 0x4d, 0xae, 0x2a, 0xf5, 0xb0, 0xc8, 0xeb, 0xbb, 0x3c, 0x83, 0x53, 0x99, 0x61
0x17, 0x2b, 0x04, 0x7e, 0xba, 0x77, 0xd6, 0x26, 0xe1, 0x69, 0x14, 0x63, 0x55, 0x21, 0x0c, 0x7d

```

Inverse netlist (36 AND, 95 XOR, 4 NOT, $D = 4$):

```

not_0 = ~x[0]; not_1 = ~x[1]; not_2 = ~x[5]; not_3 = ~x[6];
xor_0 = not_1 ^ not_2; xor_1 = not_3 ^ x[2]; xor_2 = x[2] ^ x[4];
xor_3 = x[3] ^ x[7]; xor_4 = xor_0 ^ not_0; xor_5 = xor_1 ^ not_2;
xor_6 = xor_0 ^ x[7]; xor_7 = not_0 ^ not_1; xor_8 = xor_3 ^ not_2;
xor_9 = not_0 ^ x[4]; xor_10 = xor_3 ^ not_1; xor_11 = not_1 ^ x[2];
xor_12 = xor_2 ^ xor_7; xor_13 = xor_1 ^ xor_6; xor_14 = not_2 ^ x[7];
xor_15 = xor_5 ^ x[3]; xor_16 = xor_1 ^ not_1; xor_17 = not_0 ^ not_2;
xor_18 = xor_0 ^ x[4]; xor_19 = xor_0 ^ xor_2; xor_20 = xor_2 ^ xor_6;
xor_21 = xor_7 ^ x[7]; xor_22 = xor_3 ^ not_3; xor_23 = xor_22 ^ xor_4;
xor_24 = xor_1 ^ xor_4; and_0 = xor_8 & xor_9; and_1 = xor_10 & xor_11;
and_2 = xor_0 & xor_12; xor_25 = and_1 ^ and_2; xor_26 = and_0 ^ and_2;
and_3 = xor_13 & not_1; and_4 = xor_14 & xor_4; and_5 = xor_15 & x[2];
and_6 = xor_16 & xor_17; and_7 = xor_18 & x[3]; xor_27 = and_3 ^ and_6;
xor_28 = and_4 ^ and_6; and_8 = xor_19 & xor_5; xor_29 = and_5 ^ and_8;
xor_30 = and_7 ^ and_8; xor_31 = xor_28 ^ xor_29; xor_32 = xor_26 ^ xor_29;
xor_33 = xor_29 ^ xor_30; xor_34 = xor_20 ^ xor_31; xor_35 = xor_21 ^ xor_32;
xor_36 = xor_27 ^ xor_33; xor_37 = xor_25 ^ xor_33; and_9 = xor_34 & xor_35;
xor_38 = xor_34 ^ xor_35; xor_39 = xor_23 ^ xor_36; xor_40 = xor_24 ^ xor_37;
xor_41 = xor_39 ^ xor_34; and_10 = xor_39 & xor_40; xor_42 = xor_40 ^ xor_35;
xor_43 = xor_39 ^ xor_40; and_11 = xor_42 & xor_41; xor_44 = xor_43 ^ xor_38;
xor_45 = and_10 ^ and_11; xor_46 = and_9 ^ and_11; xor_47 = xor_45 ^ xor_44;
xor_48 = xor_46 ^ xor_38; and_12 = xor_34 & xor_47; and_13 = xor_35 & xor_47;
and_14 = xor_39 & xor_48; xor_49 = xor_47 ^ xor_48; and_15 = xor_40 & xor_48;
and_16 = xor_41 & xor_49; and_17 = xor_42 & xor_49; xor_50 = and_14 ^ and_16;
xor_51 = and_12 ^ and_16; xor_52 = and_17 ^ and_15; xor_53 = and_17 ^ and_13;
and_18 = xor_11 & xor_50; and_19 = xor_10 & xor_50; and_20 = xor_9 & xor_51;
xor_54 = xor_50 ^ xor_51; and_21 = xor_8 & xor_51; and_22 = not_1 & xor_52;
xor_55 = xor_52 ^ xor_50; and_23 = xor_13 & xor_52; and_24 = xor_4 & xor_53;
xor_56 = xor_52 ^ xor_53; xor_57 = xor_53 ^ xor_51; and_25 = xor_14 & xor_53;
and_26 = xor_12 & xor_54; and_27 = xor_0 & xor_54; and_28 = xor_55 & x[2];
and_29 = xor_15 & xor_55; and_30 = xor_17 & xor_56; and_31 = xor_16 & xor_56;
and_32 = xor_18 & xor_57; xor_58 = xor_55 ^ xor_57; and_33 = xor_57 & x[3];
and_34 = xor_19 & xor_58; and_35 = xor_5 & xor_58; xor_59 = and_32 ^ and_33;
xor_60 = and_24 ^ and_25; xor_61 = and_18 ^ and_23; xor_62 = and_29 ^ xor_59;
xor_63 = and_19 ^ and_27; xor_64 = and_30 ^ and_28; xor_65 = xor_63 ^ xor_64;
xor_66 = and_34 ^ and_35; xor_67 = xor_60 ^ xor_66; xor_68 = and_22 ^ xor_62;
xor_69 = and_26 ^ xor_61; xor_70 = xor_65 ^ xor_68; xor_71 = and_31 ^ and_33;
xor_72 = xor_71 ^ xor_65; xor_73 = xor_72 ^ xor_67; xor_74 = and_30 ^ xor_59;
xor_75 = xor_74 ^ xor_67; xor_76 = xor_69 ^ xor_75; xor_77 = and_20 ^ and_19;
xor_78 = and_21 ^ xor_77; xor_79 = and_22 ^ xor_78; xor_80 = xor_60 ^ xor_79;
xor_81 = xor_61 ^ xor_80; xor_82 = and_24 ^ and_23; xor_83 = and_31 ^ xor_82;
xor_84 = and_34 ^ xor_83; xor_85 = xor_84 ^ xor_68; xor_86 = and_28 ^ and_31;
xor_87 = xor_86 ^ xor_62; xor_88 = xor_69 ^ xor_87; xor_89 = and_21 ^ and_27;
xor_90 = and_29 ^ xor_89; xor_91 = and_35 ^ xor_90; xor_92 = and_18 ^ and_20;

```

```
xor_93 = and_32 ^ xor_92; xor_94 = and_34 ^ xor_93; y[0] = xor_70;  
y[1] = xor_73; y[2] = xor_76; y[3] = xor_81; y[4] = xor_85; y[5] = xor_88;  
y[6] = xor_91; y[7] = xor_94;
```