Ties, Splits and Neighborhoods: New Area Records for Low-Latency Linear Layers

Mostafa Taha[1] and Arash Reyhani-Masoleh[2]

[1] Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada
[2] Department of Electrical and Computer Engineering, Western University, London, Canada

Abstract

For the 23 linear layers of the benchmark of Shi, Feng and Xu (SFX23, ToSC 2023), we search for minimum-depth circuits with the fewest 2-input XOR (XOR2) gates, and for minimum-area circuits at the same depth using both XOR2 and 3-input XOR (XOR3) gates, as in HILL by Li et al. (ToSC 2026). We combine three known techniques for the shortest linear program under a depth bound: (i) a beam search over the near-ties of the depth-bounded Boyar-Peralta heuristic; (ii) backward splitting; and (iii) a large neighborhood search. We set new records for the XOR2 count at the minimum depth for 14 of the 20 distinct problems represented by the 23 matrices up to row and column permutations, by 1 to 4 gates. After merging gates into XOR3 gates, our circuits are smaller than both the published XOR2/XOR3 circuits of HILL and the circuits of SFX23 merged in the same way for 19 of the 20 problems (UMC 180 nm unit areas). We release the code, the logs, all the circuits recorded during our runs, and the verification scripts.

Keywords: Linear Layers, Low Latency, Shortest Linear Program, Logic-Minimization Heuristics, Circuit Depth, MDS Matrices
AI designation: Substantial AI contribution (see the section "Use of AI Tools").

1 Introduction

Many block ciphers work as substitution-permutation networks, in which a layer of S-boxes is followed by a linear layer. A linear layer, e.g., the MixColumns operation of the Advanced Encryption Standard (AES) [NIS01, DR02], can be represented by a binary matrix M over $\mathbb{F}_2$ and implemented in hardware as a circuit of 2-input XOR (XOR2) gates. The number of these gates determines the area of the circuit, while the number of gates on its longest path, i.e., its depth, approximates its propagation delay. In this paper, we search for circuits with the fewest XOR2 gates at the minimum depth, as required by low-latency applications, and for minimum-area circuits at this depth when 3-input XOR (XOR3) gates are also available.

The problem of finding the smallest number of XOR2 gates that is required to implement a binary matrix is known as the Shortest Linear Program (SLP) problem. This problem is NP-hard [BMP08, BMP13]. Boyar and Peralta proposed a heuristic that adds, in each iteration, the gate that minimizes the sum of the distances from the known signals to the targets [BP10]. Several works improved on this heuristic when the depth is not bounded [TP20, Max19, XZL+20, LXZZ21, SYL25]. Under a depth bound, Boyar, Find and Peralta proposed the depth-constrained linear optimization (DCLO) heuristic [BFP19],

Emails: mostafa.taha@carleton.ca (Mostafa Taha), areyhani@uwo.ca (Arash Reyhani-Masoleh)

This work is licensed under a "CC BY 4.0" license.
Date of this document: 2026-11-21.

and Maximov and Ekdahl searched for small linear circuits with a maximum depth [ME19]. For low-latency linear layers, the depth bound is typically set to the minimum achievable depth, which is determined by the Hamming weights of the rows of M (Sec. 2.1). Li et al. extended the distances of Boyar and Peralta to this depth bound [LSL+19], and implemented AES MixColumns at depth 3 with 105 XOR2 gates. This count was reduced to 103 gates in [BFI21] and in [LWF+22], the latter with a backward search that splits each output into the XOR of two signals until the inputs are reached. Then, it was reduced to 102 gates in [LZW23], and to 99 gates by Shi, Feng and Xu [SFX23], who combined the forward heuristic with such backward splits. We refer to their work as SFX23. Table 5 of [SFX23] lists their XOR2 counts for 23 matrices at the minimum depth, and these matrices form our benchmark. NIST's collection of AES circuits lists a 97-gate circuit of depth 3, added in 2026 [NIS26].

The forward heuristic is greedy, and two or more candidate gates can give the same sum of distances. Boyar and Peralta break such ties by maximizing a norm of the distances [BP10]. Tan and Peyrin break the remaining ties at random and repeat the search. Their algorithm A1 also keeps only the candidate gates that lower the distance of an uncomputed target of minimum distance [TP20]. Maximov minimized the norm instead [Max19] (see Sec. 3.2 of [BDK+21]). Under the depth bound, SFX23 minimize the norm, apply the filter of A1, which we call the minimum-distance (MD) filter, break the remaining ties at random, and repeat the search many times [SFX23]. In contrast, Reyhani-Masoleh, Taha and Ashmawy used a delay-controlled Boyar-Peralta heuristic for the input and output transformation matrices of an AES S-box, in which the results of all the ties of one iteration are tested in the next iteration, and only the best sequence of gates is kept (the Improved-BP algorithm of [RMTA18]). Search trees were also used: Maximov explored a search tree of up to 50,000 leaves without a depth bound [Max19], and Maximov and Ekdahl used one, truncated to 400 leaves, for linear systems with multiplexers (Sec. 4.2.4 of [ME19]); for linear circuits under a maximum depth, they pick at random among a few best candidates of a cancellation-free heuristic (Sec. 3.2 of [ME19]). Stage 1 of our search is a beam search of width W, e.g., $W = 2000$: it keeps up to W sequences of gates of the depth-bounded heuristic, with the near-ties of each iteration, i.e., the candidate gates within a slack of the smallest sum of distances (Sec. 3.2). Sec. 5.4 compares it with random restarts at the same CPU time.

Gates with more than two inputs were used before for linear layers. Banik, Funabiki and Isobe observed that, with XOR3 gates, minimizing the area is no longer the SLP problem, and they converted XOR2 circuits into XOR2/XOR3 circuits [BFI19]. Baksi et al. added XOR3 gates to the heuristic of Boyar and Peralta [BDK+21], and Liu et al. derived low-area implementations with gates of three and four inputs from implementations with fewer inputs [LWS+22]; the works of Banik et al. and Baksi et al. do not bound the depth. Kurt Pehlivanoğlu and Demir bounded the depth with XOR2, XOR3 and XOR4 gates [KD24]. Recently, Li et al. targeted the area at the minimum depth with XOR2 and XOR3 gates [LWLG26]. We refer to their work as HILL. For AES MixColumns at depth 3, HILL reported 270.40 GE with the unit areas of the UMC 180 nm library, against 297 GE for the 99 XOR2 gates of SFX23 (Table 3 of [LWLG26]). For 54 of the 56 matrices of the HILL repository, the published area equals the area of a maximum merge of the XOR2 circuit that is stored next to it (Sec. 4). Hence, the area depends on which XOR2 circuit is merged. The XOR2 count and the area after merging are competing objectives: only a gate of fan-out one can be merged into the gate that uses it (Sec. 2.4), whereas the XOR2 count is reduced by sharing, i.e., by using one gate as the input of several gates. HILL takes this trade-off into account within a circuit, as its Rule 2 splits an XOR3 gate when the intermediate signal can be shared (Sec. 3.1 of [LWLG26]). We found that it also matters across circuits: for 16 of the 23 matrices, our smallest circuit after merging has more XOR2 gates than our circuits with the fewest XOR2 gates. For AES MixColumns,

our smallest merged circuit comes from a 100-gate circuit rather than from our 97-gate circuit, and the 100-gate circuit of SFX23, merged in the same way, is smaller than the HILL circuit (Sec. 5.2). Therefore, we evaluate every circuit that our searches find by both measures, and we select the circuits by their XOR2 count and by their area after merging.

Our main contributions are as follows:

- *A combination of search techniques* for the SLP problem under a depth bound (Sec. 3): beam search over the near-ties (stage 1), our reimplementation of the backward splitting of SFX23 (stage 2), beam search over the backward splits (stage 3), and a large neighborhood search (stage 4) that keeps the last gates of k random outputs of the current circuit and recomputes all the other gates from the inputs. Stage 4 can also start from another circuit, e.g., one of SFX23, and minimize the UMC area after merging instead of the XOR2 count.
- *Circuits with fewer XOR2 gates and smaller areas* (Sec. 5.2). For 16 of the 23 matrices of [SFX23], our circuits use 1 to 4 fewer XOR2 gates than the previous records, including records of permutation-equivalent matrices. These form 14 of the 20 distinct problems, 11 of them of size 32×32. The four stages set 12 of these records; stage 4 started from the circuits of SFX23 set the remaining 2, for SmallScale AES and [SKOP15] Hadamard 32×32, and lowered 2 more. For the remaining 6 problems, we reach the record. With the UMC unit areas, our merged circuits are 0.3% to 7.3% smaller than both the circuits of the HILL repository and those of SFX23 merged as ours on 19 problems, and equal to the smaller of the two on the remaining one. On 11 of these 19 problems, all of size 32×32, our merged circuit comes from an XOR2 circuit with fewer gates; on the remaining 8, from one with as many gates or more.
- *A comparison of beam search with random restarts at the same CPU time* (Sec. 5.4), with 4 seeds per matrix. Without a norm-based criterion for the ties, as in Improved-BP, beam search has a lower mean XOR2 count on 10 of the 12 problems of size 32×32. Under the tie rule and the MD filter of SFX23 (Sec. 3.1), with the slack $s = 0$, the random restarts give the lower mean for 12 of the 20 problems.

We also release the code, the matrices, the logs of all the runs, every circuit that they recorded, and the scripts that verify them, at https://github.com/Carleton-SCI/low-latency-linear-layers/tree/main.

Sec. 2 reviews the problem, the distances and the area model, Sec. 3 describes the search, Sec. 4 the benchmark, and Sec. 5 the experiments. Sec. 6 lists the limitations, and Sec. 7 concludes the paper.

2 Preliminaries

In this section, we briefly review the shortest linear program problem under a depth bound. We discuss the depth of a sum of signals, the distances that are used by the Boyar-Peralta heuristic under a depth bound, and the area model with XOR3 gates.

2.1 Linear Programs and Their Depth

Let M be an $m \times n$ binary matrix over $\mathbb{F}_2$, and let $x = [x_0, \ldots, x_{n-1}]^\top$ be the input vector. Then, the output vector is $y = [y_0, \ldots, y_{m-1}]^\top = Mx$, i.e., row j of M, denoted M_j, computes output y_j, and column c corresponds to input x_c. A circuit (or linear program) computes y using XOR2 gates, where each gate is the XOR of two earlier signals (inputs or gates). The XOR2 count of a circuit is the number of its gates. The depth of an input signal is zero, and the depth of a gate $u_i \oplus u_j$ is evaluated as $\max(\mathrm{dep}(u_i), \mathrm{dep}(u_j)) + 1$.

The depth of a circuit is the largest depth of its outputs, i.e., its propagation delay in terms of the number of XOR2 gate delays.

The SLP problem asks for the smallest number of XOR2 gates that is required to implement M [BMP08, BMP13]. In this paper, we search heuristically for small circuits that compute $y = Mx$ under a depth bound D, i.e., such that the depth of every output is at most D. Note that the minimum depth of M is $\max_j \lceil \log_2(\mathrm{HW}(M_j)) \rceil$, for $j \in [0, m-1]$, with HW denoting the Hamming weight function. Similar to [SFX23], we set D to this minimum depth for all the matrices of the benchmark.

2.2 Depth of a Sum of Signals

Li et al. showed that the minimum depth of a tree of XOR2 gates that adds the signals $u_1, \ldots, u_\ell$ of depths $d_1, \ldots, d_\ell$ is $\lceil \log_2 \sum_{i=1}^{\ell} 2^{d_i} \rceil$ [LSL+19, Lemma 1]. Equivalently, these signals can be added within a depth bound b if and only if $\sum_{i=1}^{\ell} 2^{d_i} \leq 2^b$. We call 2^b the budget of a target with depth bound b. Note that this minimum depth is the same as the one obtained by the binary tree method of [RMTA18], where, iteratively, the two signals with the least depths are added together.

2.3 Distances under a Depth Bound

Following Boyar and Peralta [BP10], the heuristic maintains a base S of known signals, which includes the inputs and the gates found so far, along with a set of targets. For a target t with depth bound $b(t)$, the distance $\delta(t)$ is the number of XOR2 gates that are required to reach t from the base S if there were no sharing of gates, without exceeding the depth bound:

$$\delta(t) = \min \Big\{ |U| - 1 \ : \ U \subseteq S, \ \bigoplus_{u \in U} u = t, \ \sum_{u \in U} 2^{\mathrm{dep}(u)} \leq 2^{b(t)} \Big\}. \tag{1}$$

If no such subset exists, $\delta(t)$ is infinite, reporting that there is no path to t under the depth bound. In the beginning, S includes only the inputs at depth zero, and $\delta(t) = \mathrm{HW}(t) - 1$ if $\mathrm{HW}(t) \leq 2^{b(t)}$, while $\delta(t)$ is infinite otherwise. A new gate $g = u \oplus v \notin S$, with $u, v \in S$, lowers $\delta(t)$ by at most one: in a set U that contains g, replacing g by u and v, and removing the pairs of equal signals, gives a set of at most $|U| + 1$ signals of S with the same XOR, and $2^{\mathrm{dep}(u)} + 2^{\mathrm{dep}(v)} \leq 2^{\mathrm{dep}(g)}$. The gate g lowers $\delta(t)$ by one if and only if $t \oplus g$ is the XOR of a set $U \subseteq S$ of $\delta(t) - 1$ signals with $\sum_{u \in U} 2^{\mathrm{dep}(u)} \leq 2^{b(t)} - 2^{\mathrm{dep}(g)}$.

Similar to [RMTA18, SFX23], we compute (1) exactly, by exhaustive search within all the signals of the base of depth smaller than D (for a target with depth bound $b(t)$, only the signals of depth smaller than $b(t)$ can be in a set U of two or more signals). To score all the candidate gates of an iteration, the search enumerates, for each target t, the subsets of $\delta(t) - 1$ such signals in increasing order of depth, pruned by the budget, and it looks up t XORed with each subset among the candidate gates; by the property above, a candidate gate that is found in this way, within the budget, lowers $\delta(t)$ by one. After a gate is added, the distances are updated in the same way. Hence, the cost of an iteration grows with the number of these subsets, summed over the targets.

2.4 XOR3 Gates and the Area Model

A gate of fan-out one that is not an output can be merged with the gate that uses it into one XOR3 gate. Each XOR3 gate absorbs at most one XOR2 gate, and an absorbed gate cannot absorb another one. Hence, the merging is a matching in the forest whose edges join each non-output gate of fan-out one to the gate that uses it, and we take a maximum matching, computed by dynamic programming over the trees. Banik et al. merge such pairs

one at a time until none is left, in an order that they do not specify (Sec. 3.1 of [BFI19]). The result is a circuit of [ν_2 XOR2 + ν_3 XOR3] gates, where $\nu_2 + 2\nu_3$ is the XOR2 count of the original circuit. Note that sharing lowers the XOR2 count, but a shared gate, i.e., a gate that is used by two or more gates, has a fan-out larger than one, so it cannot be absorbed into any gate that uses it.

The area of a circuit is the sum of the unit areas of its gates, in gate equivalents (GE), i.e., in units of the area of a 2-input NAND gate. We use the unit areas of Table 1 of [LWLG26]: XOR2 = 3.00 GE and XOR3 = 4.67 GE for the UMC 180 nm library, and XOR2 = 3.33 GE and XOR3 = 4.66 GE for the STM 130 nm library. Hence, each merge saves 1.33 GE under the UMC library and 2.00 GE under the STM library. Under the STM library, it also pays to duplicate a non-output gate of fan-out two and to absorb one copy into each of the two gates that use it: this saves $2 \times 2.00 - 3.33 = 0.67$ GE (two merges minus one more XOR2 gate), while it costs $3.00 - 2 \times 1.33 = 0.34$ GE under the UMC library, and duplicating a gate of larger fan-out pays under neither library. Hence, our STM areas allow this duplication. Liu et al. apply both rewritings in a fixed order: the gates of fan-out one first, then those of fan-out two, each in topological order (Sec. 5.1 of [LWS+22]). The best choice of merges and duplications is no longer a matching on a forest, and we compute it exactly as a 0-1 program over the gates of the given circuit and their fan-outs: one variable for each possible merge (a non-output gate of fan-out one and the gate that uses it) and for each possible duplication (a non-output gate of fan-out two and the two gates that use it), the constraint that every gate belongs to at most one selected merge or duplication, and the largest total saving as the objective. We solved it with the CP-SAT solver of OR-Tools 9.15, which proved the optimality of the solution for every circuit.

HILL counts an XOR3 gate as one gate level (Sec. 2.2 of [LWLG26]). Merging, with or without duplication, does not increase the number of gate levels on any path. Hence, our merged circuits satisfy the depth bound under both notions of depth: the XOR2 circuit before merging has depth at most D, and the merged circuit has at most D gate levels, with an XOR3 gate as one level, as in HILL. We report the depth of a merged circuit as the depth of the XOR2 circuit before merging; our repository also gives the number of gate levels of every merged circuit, which can be smaller.

3 The Search

In this section, we describe the four stages of our search. Stage 1 is a beam search over the near-ties of the forward search, i.e., its ties and the options within a slack of them. Stage 2 is our reimplementation of the backward splitting of SFX23, and stage 3 applies beam search to the backward splits. Stage 4 is a large neighborhood search (LNS) that starts from the best circuit of stages 1–3. Here, we first review the forward search under a depth bound, and we conclude this section with the selection and the verification of the circuits. Algorithm 1 summarizes the four stages. We follow one round of each of them on the 8×8 example

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_7 \end{bmatrix}.$$

Each row has weight 5; hence, the minimum depth is $D = 3$ (Sec. 2.1), each output is a target with depth bound 3, i.e., with a budget of $2^3 = 8$, and a signal of depth d costs 2^d

Algorithm 1 The four stages. A prefix is a sequence of backward splits; its targets are the outputs that are not split (depth bound D) and the parts of the splits (depth bound $D-1$), see Sec. 3.3. FORWARD($\mathcal{T}, R$) returns the best circuit of R randomized forward runs for the targets $\mathcal{T}$, with the tie rule and filter of the stage (Sec. 3.1). Circuits and prefixes are ranked by the objective of Sec. 3.5, and the best circuits by XOR2 count, UMC area and STM area are recorded.

function BEAM(targets $\mathcal{T}$, width W, slack s, tie rule, MD filter) ▷ stage 1, and in stage 4
  $\mathcal{L} \leftarrow$ {the base of the inputs, with its pre-emptive gates}
  while a sequence of $\mathcal{L}$ is not complete **do**
    $\mathcal{O} \leftarrow \{(\sigma, g) : \sigma \in \mathcal{L}$ not complete, g a candidate gate of σ (MD filter, if enabled)$\}$
    if $\mathcal{O} = \emptyset$ **then return** the circuits of the complete sequences of $\mathcal{L}$
    $\Delta(\sigma, g) \leftarrow$ the sum of distances of σ with g, before the pre-emptive gates
    keep the options with $\Delta(\sigma, g) \le \Delta_{\min} + s$, where $\Delta_{\min}$ is the minimum over all of $\mathcal{O}$
    shuffle them; sort them stably by Δ, then by the norm (*bp*: descending, *sfx*: ascending)
    $\mathcal{L}' \leftarrow$ the complete sequences of $\mathcal{L}$
    for each kept option (σ, g) in this order, while $|\mathcal{L}'| < W$ **do**
      $\sigma' \leftarrow \sigma$ with g and the pre-emptive gates that follow it
      add σ' to $\mathcal{L}'$, unless a sequence created in this iteration has the same hash
    $\mathcal{L} \leftarrow \mathcal{L}'$
  return the circuits of $\mathcal{L}$
function SPLIT(prefix P) ▷ priority (3) of SFX23
  if Rule 2 applies to P **then return** P with a split by Rule 2
  if Rule 3 applies to P **then**, with probability q, **return** P with a split by Rule 3
  return P with a split by Rule 5, if one can be drawn
function BPBS(N, R) ▷ stage 2
  $P_0 \leftarrow$ the prefix without splits, after Rule 1; evaluate it by FORWARD(targets of P_0, R)
  for $i = 1, \ldots, N$ **do**
    $P \leftarrow P_0$
    while an output of P can be split, and P has fewer splits than the maximum **do**
      $P \leftarrow$ SPLIT(P); evaluate P by FORWARD(targets of P, R)
function SPLITBEAM(N, B, K, R) ▷ stage 3
  $P_0 \leftarrow$ the prefix without splits, after Rule 1; evaluate it by FORWARD(targets of P_0, R)
  for $i = 1, \ldots, N$ **do**
    $\mathcal{P} \leftarrow \{P_0\}$
    while $\mathcal{P} \neq \emptyset$, and the prefixes of $\mathcal{P}$ have fewer splits than the maximum **do**
      $\mathcal{P}' \leftarrow$ the prefixes with distinct targets among K calls SPLIT(P) for each $P \in \mathcal{P}$
      evaluate each $P \in \mathcal{P}'$ by FORWARD(targets of P, R)
      $\mathcal{P} \leftarrow$ the B prefixes of $\mathcal{P}'$ with the best circuits
function LNS(current circuit C, number of iterations, $k_{\min}$, $k_{\max}$, W, s) ▷ stage 4
  for each iteration **do**
    draw k uniformly from $\{k_{\min}, \ldots, k_{\max}\}$
    cap k at the number of outputs computed by a gate of C
    select k such outputs at random, and take the last gate of each in C as a split $y_j = p_1 \oplus p_2$
    $\mathcal{T} \leftarrow$ the targets of this prefix of k splits (Sec. 3.3)
    $C' \leftarrow$ the best circuit of BEAM($\mathcal{T}$, W, s, *all*, no filter), completed by the k split gates
    if C' is not worse than C **then** $C \leftarrow C'$

(Sec. 2.2).

3.1 Forward Search under a Depth Bound

The forward search follows the Boyar-Peralta heuristic [BP10], with the distances defined in Sec. 2.3. In the beginning, the base S includes only the input signals at depth zero, and each output is a target with depth bound D. Then, in each iteration, the algorithm prepares the candidate next gates by XORing every two signals of the base. It keeps

Table 1: Distances δ of the outputs of the example in iterations 1 and 2 of the forward search, with the base alone and with four or three candidate gates, and their sum Δ and norm $\sum \delta^2$. In iteration 1, the base holds the inputs, and the other 24 candidate gates are rotations of the four shown. In iteration 2, the base also holds $x_0 \oplus x_1$; $x_5 \oplus x_6$ and $x_6 \oplus x_7$, the other two ties, are the mirror images of $x_3 \oplus x_4$ and $x_2 \oplus x_3$. MD: whether the MD filter keeps the gate.

		y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7	Δ	$\sum \delta^2$	MD
Iteration 1	base	4	4	4	4	4	4	4	4	32	128	
	$x_0 \oplus x_1$	3	4	4	4	4	3	3	3	28	100	
	$x_0 \oplus x_2$	3	4	4	4	4	4	3	3	29	107	
	$x_0 \oplus x_3$	3	4	4	4	4	4	4	3	30	114	
	$x_0 \oplus x_4$	3	4	4	4	3	4	4	4	30	114	
Iteration 2	base	3	4	4	4	4	3	3	3	28	100	
	$x_2 \oplus x_3$	2	3	3	4	4	3	3	2	24	76	kept
	$x_3 \oplus x_4$	2	3	3	3	4	3	3	3	24	74	kept
	$x_4 \oplus x_5$	3	3	3	3	3	3	3	3	24	72	removed

only the gates of depth at most $D-1$ whose values are not in the base already, and that lower the distance of at least one target; a gate of depth D can only be an output, and outputs are added as pre-emptive gates (see below). One of these gates is selected such that if it were added to the base, the new base would minimize the sum of the distances, $\Delta = \sum_t \delta(t)$. In the example, each output requires 4 gates at first, i.e., $\delta(y_j) = 4$, and $\Delta = 32$. The candidate gates of iteration 1 are the 28 XORs of two inputs, and Table 1 gives the distances with four of them. For example, $x_0 \oplus x_1$ appears in y_0, y_5, y_6 and y_7, and it lowers their distances to 3; hence, $\Delta = 28$. The 8 gates $x_c \oplus x_{c+1}$, with $c+1$ taken modulo 8, tie at $\Delta = 28$.

If two or more candidate gates tie, one of three tie rules applies:

- *bp*: select the gate that maximizes the squared Euclidean norm of the distances, $\sum_t \delta(t)^2$, as in [BP10];
- *sfx*: select the gate that minimizes this norm, as in [Max19, SFX23];
- *all*: keep every gate of minimum Δ in the set of ties, without any norm, as in the Improved-BP algorithm of [RMTA18]; a forward run with this rule breaks the ties as algorithm A2 of [TP20], without its filter.

If there is a further tie, a gate is selected uniformly at random among the ties. In iteration 1 of the example, the 8 ties also have the same norm, $\sum \delta^2 = 100$, so that each rule selects one of them at random, say $x_0 \oplus x_1$. In iteration 2, the 29 candidate gates include five ties at $\Delta = 24$, with different norms (Table 1): the *bp* rule selects a tie of maximum norm, $x_2 \oplus x_3$ or $x_6 \oplus x_7$, the *sfx* rule selects the tie of minimum norm, $x_4 \oplus x_5$, and the *all* rule keeps the five ties and selects one of them at random.

Whenever a target can be computed by one gate from the base, i.e., $\delta(t) = 1$, this 'pre-emptive' gate is added directly, at the smallest possible depth, and the distances are updated. For example, once the base holds $x_0 \oplus x_1$, $x_2 \oplus x_3$ and their XOR, of depth 2 and cost 4, the outputs $y_0 = (x_0 \oplus x_1 \oplus x_2 \oplus x_3) \oplus x_4$ and $y_7 = (x_0 \oplus x_1 \oplus x_2 \oplus x_3) \oplus x_7$ are one gate away within their budget ($4 + 1 \leq 8$), and both are added. Note that Δ is evaluated for a candidate gate before the pre-emptive gates that follow it are added. If the depth of any signal in the base reaches D, this signal is no longer used to build any further target.

In addition, a filter can be enabled that keeps only the candidate gates that lower the distance of at least one uncomputed target of minimum distance. This is the filtering step of

algorithm A1 of Tan and Peyrin [TP20], which SFX23 apply under a depth bound [SFX23]; we call it the minimum-distance (MD) filter. In iteration 2 of the example, the targets of minimum distance are y_0, y_5, y_6 and y_7, with $\delta = 3$, and the MD filter removes 14 of the 29 candidate gates, among them $x_4 \oplus x_5$, which lowers only distances of 4; hence, with the *sfx* rule and the MD filter, as in SFX23, the forward search selects $x_3 \oplus x_4$ or $x_5 \oplus x_6$ instead. A forward search is complete when all its targets are computed, and we call one complete forward search a forward run.

3.2 Beam Search over the Near-Ties

Reyhani-Masoleh et al. test the results of all the ties of one iteration in the next iteration, and keep only the best sequence of gates [RMTA18]. We extend this idea into a beam search that keeps up to W sequences of gates, i.e., partial circuits of the forward search, with W being the width of the search. In each iteration, every kept sequence that is not complete proposes its candidate gates, and each option (a sequence and one of its candidate gates) is scored by the sum of distances Δ of the sequence after adding the gate, before the pre-emptive gates. We keep the options whose Δ is at most $\Delta_{\min} + s$, where $\Delta_{\min}$ is the smallest sum over the options of all the sequences, and $s \geq 0$ is the slack. We call these options the near-ties; with $s = 0$, each of them is a tie of its sequence. Since $\Delta_{\min}$ is taken over all the sequences, all the options of a sequence whose smallest Δ is above $\Delta_{\min} + s$ are dropped. The near-ties are shuffled, and sorted stably by Δ, and then by the norm for the *bp* and *sfx* rules, and we create the new sequences in this order, until W sequences are kept; hence, the norm decides which near-ties are kept only when there are more than W of them. With $W = 1$ and $s = 0$, the choices are distributed as those of a randomized forward run with the same tie rule and filter; Sec. 5.4 compares the beam search with such independent runs at the same CPU time.

The completed circuits of the previous iteration are kept first, and they count among the W sequences. A new sequence whose set of gate values has the same order-independent 64-bit hash as a sequence created before in the same iteration is skipped; the operands and the depths of the gates are not compared. The ranking uses Δ only, although two sequences of one iteration can contain different numbers of pre-emptive gates. At the end, every completed circuit is pruned, i.e., its unused gates are removed, and then it is verified and evaluated (Sec. 3.5).

In the example, with $W = 4$ and $s = 0$, iteration 1 has one sequence and 28 options, and its near-ties are the 8 ties of Table 1. They are shuffled, and the first 4 of them form the beam, e.g., $\{x_0 \oplus x_1\}$, $\{x_2 \oplus x_3\}$, $\{x_4 \oplus x_5\}$ and $\{x_1 \oplus x_2\}$. In iteration 2, each of these sequences has 29 options, 5 of which tie at $\Delta = 24$; hence, $\Delta_{\min} = 24$, and 20 of the 116 options are near-ties. They are shuffled, and the first 4 of them form the new beam, except that a sequence whose set of gate values was already created in this iteration is skipped; e.g., $\{x_0 \oplus x_1, x_2 \oplus x_3\}$ is reached from $\{x_0 \oplus x_1\}$ and from $\{x_2 \oplus x_3\}$. With $s = 1$, the options of $\Delta = \Delta_{\min} + 1$ are also near-ties: the 8 gates $x_c \oplus x_{c+2}$ in iteration 1, and 6 more options of the sequence $\{x_0 \oplus x_1\}$ in iteration 2. Hence, W bounds the number of partial circuits, and s bounds how far above the best option a kept option can be. Stage 1 with $W = 4$ and $s = 0$ ends with the 16-gate circuit of Sec. 3.5.

Compared with the searches of Sec. 1, Improved-BP keeps one sequence and tests all its ties one iteration ahead [RMTA18], the search tree of Maximov has no depth bound [Max19], and Maximov and Ekdahl keep, for each node of their tree, a number of its best children [ME19, Sec. 4.2.4]. Our beam search differs from these searches in that it compares the options of all its sequences against one threshold, $\Delta_{\min} + s$, under the depth bound, and removes the sequences with equal sets of gate values; its distances are those of [RMTA18, SFX23] (Sec. 2.3).

3.3 Backward Splitting and Beam Search over the Splits

The backward search of [LWF+22] splits an output into the XOR of two signals of smaller depth, and SFX23 combined such splits with the forward search [SFX23]. A split replaces an output w by two targets p_1 and p_2 with depth bound $D-1$ each, where $w = p_1 \oplus p_2$, and the gate $p_1 \oplus p_2$ is added after the forward search computes p_1 and p_2. Hence, a sequence (prefix) of splits defines a new set of targets for the forward search: the outputs that are not split, with depth bound D, and the parts of the splits (the predecessors), with depth bound $D-1$, also when a part is an output. The rules of the splits, numbered as in SFX23, are:

- Rule 1: an output whose minimum depth is smaller than D is not split; it stays a target.
- Rule 2: if an output is the XOR of two targets that can be built within depth $D-1$, it is split into these two targets.
- Rule 3: an output w is split into a target p and $w \oplus p$ if both can be built within depth $D-1$.
- Rule 4 (not used): if two outputs are $w_1 = p_1 \oplus p_2$ and $w_2 = p_2 \oplus p_3$, where p_1, p_2 and p_3 can be built within depth $D-1$, both are split at once.
- Rule 5: a uniformly chosen output is split into a random subset of its input terms and the rest, with a size drawn uniformly among the sizes that let both parts be built within depth $D-1$; hence, our random splits are cancellation-free, while the definition of Rule 5 in SFX23 does not require it.

Rules 2 and 3 choose uniformly among the pairs of an output and a target to which they apply. We follow the priority (3) of SFX23 (Sec. 4.2 of [SFX23]), i.e., Rule 1 > Rule 2 > Rule 3, Rule 5, where Rules 3 and 5 have the same priority: Rule 1 is applied before any split; then, if Rule 2 does not apply and Rule 3 applies, we apply Rule 3 with probability q; otherwise, i.e., with probability $1-q$ or when Rule 3 does not apply, we apply Rule 5 (Split in Algorithm 1). The parameter q is ours, and with $q = 0.5$, Rules 3 and 5 are equally likely when both apply; SFX23 choose at random among the rules of the highest priority, and they leave their Rule 4 out of this priority, since its cases are covered by Rules 3 and 5 (Sec. 4.2 of [SFX23]). Each prefix of splits is evaluated by R randomized forward runs, and the best circuit is kept.

In the example, Rule 1 applies to no output, since every output has minimum depth 3, and the parts of a split have depth bound 2, i.e., a budget of 4. In the beginning, no target can be built within depth 2; hence, Rules 2 and 3 do not apply, and the first split is drawn by Rule 5, e.g., y_1 into x_5 and $p = x_1 \oplus x_2 \oplus x_3 \oplus x_4$ (here, Rule 5 gives each part 1 to 4 of the 5 terms). Then, Rule 2 applies to no output, since $x_5 \oplus p = y_1$ is already split, and Rule 3 applies to 7 pairs of an output and a part, e.g., to y_0 and p, since $y_0 \oplus p = x_0$ can be built within depth 2. With probability q, Rule 3 is applied to one of these pairs, drawn at random; e.g., it splits y_0 into p and x_0. After these two splits, the targets are $y_2, \ldots, y_7$ and the parts p, x_5 and x_0, with $\Delta = 27$ (Table 2), and the split gates $y_1 = x_5 \oplus p$ and $y_0 = x_0 \oplus p$ complete the circuit.

Stage 2, backward splitting (BPBS): This stage is our reimplementation of the framework of SFX23 that combines their improved BP heuristic with backward search. It runs N independent random trajectories. Each trajectory starts from the initial prefix and adds one split at a time, until no split can be drawn or the maximum number of splits is reached, and every prefix along the way is evaluated. In the example, a trajectory that starts with the two splits above draws a third one: Rule 2 still applies to no output, and

Table 2: Targets of the forward search in the example and their distances δ: without splits, after Rule 5 splits y_1 into x_5 and $p = x_1 \oplus x_2 \oplus x_3 \oplus x_4$, and after Rule 3 splits y_0 into p and x_0. The outputs have depth bound 3 and p has depth bound 2; –: not a target. The parts x_5 and x_0 are inputs ($\delta = 0$).

	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7	p	Δ
No split	4	4	4	4	4	4	4	4	–	32
Rule 5, y_1	4	–	4	4	4	4	4	4	3	31
Rule 3, y_0	–	–	4	4	4	4	4	4	3	27

Rule 3 applies again, but Rule 5 is used, with probability $1 - q$: it splits y_5 into x_1 and $x_0 \oplus x_5 \oplus x_6 \oplus x_7$. The forward runs with the *sfx* rule and the MD filter reach 17 gates after each of the first two splits, and 16 gates after the third.

Stage 3, beam search over the splits: In each step, each of the B kept prefixes draws K split options, and each new prefix is evaluated by R randomized forward runs. Then, the B new prefixes with the best circuits are kept for the next step, with ties broken arbitrarily; two new prefixes with the same targets and depth bounds count as one. This search is repeated N times from the initial prefix, until no split can be drawn or the maximum number of splits is reached. For example, with $B = 4$ and $K = 8$, the initial prefix draws 8 splits, all by Rule 5, and the 4 new prefixes with the best circuits are kept; in the next step, each of them draws 8 splits, and the 4 best of the up to 32 new prefixes are kept. Note that our implementation of the backward splitting assumes that D equals the minimum depth, as in SFX23.

3.4 Large Neighborhood Search

The large neighborhood search starts from the best circuit of all the runs of stages 1–3, which we denote as the current circuit; all the seeds of stage 4 start from this circuit. In each iteration, we draw k uniformly from $\{k_{\min}, \ldots, k_{\max}\}$, cap it at the number of outputs that are computed by a gate of the current circuit, and select k such outputs at random. For each selected output y_j, we keep its last gate, i.e., the split $y_j = p_1 \oplus p_2$, where p_1 and p_2 are the values of the operands of this gate in the current circuit. Hence, p_1 and p_2 become targets with depth bound $D - 1$, while all the other outputs remain targets with depth bound D. Then, we re-solve the whole circuit from the inputs with the beam search of Sec. 3.2. The new circuit replaces the current circuit if its objective (Sec. 3.5) is not worse. Each iteration evaluates a prefix of k backward splits (Sec. 3.3) that is taken from the current circuit, by beam search instead of randomized forward runs, and the neighborhood of the current circuit is the set of circuits in which each of the k selected outputs is the XOR of the same two values as in the current circuit. Note that only the values p_1 and p_2 are kept, not the gates that compute them.

In the example, we start stage 4 from an 18-gate circuit of a forward run with the *sfx* rule and the MD filter. In its second iteration, with $k = 2$, it keeps $y_0 = (x_0 \oplus x_1) \oplus (x_2 \oplus x_3 \oplus x_4)$ and $y_3 = (x_3 \oplus x_4) \oplus (x_5 \oplus x_6 \oplus x_7)$. The targets are then the other six outputs, with depth bound 3 and $\delta = 4$, and the four parts, with depth bound 2: $x_0 \oplus x_1$ and $x_3 \oplus x_4$, with $\delta = 1$, which are added at once as pre-emptive gates, and $x_2 \oplus x_3 \oplus x_4$ and $x_5 \oplus x_6 \oplus x_7$, with $\delta = 2$. The beam search solves these targets, and the two kept gates complete a circuit of 17 gates, which is not worse than the current one; hence, it replaces it. A later iteration, which keeps only $y_7 = x_3 \oplus (x_0 \oplus x_1 \oplus x_2 \oplus x_7)$, gives 16 gates.

Other works also improve an existing circuit. Tan and Peyrin swap the order of XORs in small patterns of gates and search exhaustively for a smaller binary tree with the same root and at most five leaves (Sec. 4.3 of [TP20]). Xiang et al. optimize the decompositions

of an invertible matrix, i.e., its implementations under the sequential-XOR metric, with rules of matrix multiplication over $\mathbb{F}_2$ [XZL+20]. Liu et al. divide a circuit into a base part, a target part and an unrelated part, re-search the target part from the base part, and keep the rest of the circuit (Sec. 4 of [LZW23]). Sun et al. remove random gates of fan-out one, keep the other gates that the inputs can still compute, and solve a smaller SLP problem for the gates that used the removed ones (Sec. 4.2 of [SYL25]). In contrast, our neighborhood search keeps only the last gates of k outputs, and it recomputes all the other gates from the inputs.

3.5 Selection and Verification of the Circuits

Every completed circuit is pruned, verified and merged (Sec. 2.4). The objective is either the XOR2 count, with the UMC area as a tie-breaker, or the area (UMC or STM), with the XOR2 count as a tie-breaker. Whatever the objective, each run records the best circuit by each of the three measures, over all the circuits that it completes: the XOR2 count, the UMC area and the STM area, the latter with merges only. Since the XOR2 count and the area after merging are competing objectives (Sec. 2.4), the smallest circuit after merging need not have the fewest XOR2 gates. All the runs in Sec. 5 used the XOR2 count as the objective, except the runs of stage 4 with the UMC area as the objective (Sec. 5.1). The UMC areas reported there are those of the smallest recorded circuits, and the STM areas come from the exact merge with duplication of Sec. 2.4, applied to all the recorded circuits. Before a circuit is recorded, it is evaluated on the unit vectors against the matrix, and the depth of every output is checked.

In the example, each of the 8 non-output gates of the 16-gate circuit of stage 1, listed below (+ denotes XOR), is used by two gates; hence, no gate can be merged, and its UMC area is $16 \times 3.00 = 48.00$ GE. Stage 1 with $s = 1$ ends with a 17-gate circuit, also listed below, whose non-output gates $x_1 \oplus x_6$, $x_0 \oplus x_4$ and $x_3 \oplus x_7$ have fan-out one. Merging them gives [11 XOR2 + 3 XOR3] gates, i.e., 47.01 GE. Hence, the selection by XOR2 count keeps the 16-gate circuit, and the selection by area keeps the 17-gate circuit, as for 16 of the 23 benchmark matrices (Sec. 5.2). With the STM unit areas, the exact merge with duplication gives the 17-gate circuit 3 merges and 2 duplications, i.e., [5 XOR2 + 7 XOR3] gates and 49.27 GE, and the 16-gate circuit 4 duplications, i.e., [4 XOR2 + 8 XOR3] gates and 50.60 GE.

The 16-gate circuit of stage 1 (Sec. 3.2). Each of its 8 non-output gates, $g_1, \ldots, g_8$, is used by two gates.

```
g1 = x4 + x5          g5 = g2 + g4          y3 = x3 + g6          y2 = x6 + g7
g2 = x6 + x7          y5 = x5 + g5          y4 = x0 + g6          g8 = g3 + g4
g3 = x2 + x3          y6 = x2 + g5          g7 = g1 + g3          y0 = x4 + g8
g4 = x0 + x1          g6 = g1 + g2          y1 = x1 + g7          y7 = x7 + g8
```

The 17-gate circuit of stage 1 with $s = 1$, merged: [11 XOR2 + 3 XOR3] gates, 47.01 GE. Its XOR3 gates are written $a + (b + c)$, where $b + c$ is the absorbed gate.

```
g1 = x4 + x5          y1 = g1 + g4          y4 = g3 + g5          y0 = g4 + (x0 + x4)
g2 = x2 + x3          y7 = g3 + g4          g6 = g3 + (x1 + x6)   y3 = g5 + (x3 + x7)
g3 = x0 + x7          g5 = x6 + g1          y5 = x5 + g6
g4 = x1 + g2          y2 = g2 + g5          y6 = x2 + g6
```

4 The Benchmark

We use the 23 matrices of Table 5 of SFX23 [SFX23], with the XOR2 counts reported there (this table does not include the [BKL16] circulant 16×16 and [JPST17] involutory 32×32 matrices of Table 1 of [LZW23]), and the XOR2/XOR3 circuits of the HILL repository [Li26]. These are the matrices of AES MixColumns [DR02], SmallScale AES [CMR05] and Joltik [JNP15], along with MDS matrices from [SKOP15, LS16, LW16, BKL16, SS16, JPST17, KLSW17, LSL+19], and our tables label each matrix by its source. The code

repository of SFX23 provides 42 circuits, which reproduce every XOR2 count of their Table 5, and the HILL repository provides a result circuit for each of 56 matrices, along with the XOR2 circuit that it started from. We derived the matrix of every circuit by evaluating it on the unit vectors. As in Sec. 2.1, row j of a matrix computes output y_j, and column c corresponds to input x_c; the orientation matters, since five of the matrices are not permutation-equivalent to their transposes.

Equivalent matrices: The XOR2 count and the depth are invariant under row and column permutations. Joltik and [SKOP15] involutory 16×16 are the same matrix, and [LW16] involutory 16×16 is a row and column permutation of it; [LS16] and [LW16] circulant 16×16 are row and column permutations of each other. Therefore, the 23 matrices form 20 distinct problems. In addition, [KLSW17] is a row and column permutation of [LSL+19] HW160 #93 (named R in the HILL repository); hence, the 84-gate circuit of SFX23 and the 230.72-GE circuit of HILL for #93 apply to [KLSW17], for which Table 5 of SFX23 lists 86 gates and HILL lists 231.40 GE. All the matrices of Table 3 are invertible and MDS.

Sources: The two repositories use the same matrix, entry by entry and in the same orientation, for 22 of the 23 matrices. For [LS16] circulant 32×32, the circuits of the HILL repository compute a matrix that differs from that of SFX23 in two entries; we follow SFX23 in all tables, and for the other matrix, the four stages give the same values, 107 XOR2 gates and 274.79 GE at depth 3. Where Table 3 of HILL and the HILL repository differ, for four matrices, we follow the repository.

HILL areas: For 54 of the 56 matrices of the HILL repository, the area of the published XOR2/XOR3 circuit equals the area of a maximum merge of the gates of fan-out one (Sec. 2.4) of the XOR2 circuit stored next to it; for AES MixColumns, this XOR2 circuit is the 99-gate circuit of SFX23. Against the published circuits of SFX23 after the same merge (Table 3), the HILL area is the same for 12 of the 19 problems of the repository, smaller for 4 and larger for 3.

5 Experimental Results and Comparison

In this section, we compare the circuits found by our search with the best published circuits for the benchmark of Sec. 4. Thereafter, we report the results of each stage, we compare beam search over the near-ties with random restarts at the same CPU time, and we report the computation time.

5.1 Setup

The search program is written in C++17, and it was compiled using g++ 11.4.0 with the flags `-O3 -march=native -std=c++17` under Ubuntu 20.04.6 LTS. Every run is single-threaded and deterministic for a given seed with the C++ standard library libstdc++: its random generator is `std::mt19937_64`, seeded from the seed of the run, and its shuffles, sorts and uniform draws are those of libstdc++. All the runs of the search were done on a server with two Intel Xeon E5-2670 v3 processors (24 cores, 48 hardware threads) equipped with 128 GB of RAM, with up to 48 runs at a time. The run times below are wall-clock times of single runs under this load.

We applied the four stages of Algorithm 1 to each matrix as follows, where stages 2 to 4 used seeds 1 to 8.

Stage 1, beam search over the near-ties: 56 runs: tie rule *all* with width $W = 2000$ and slack $s = 1$ (seeds 1–32), *bp* with $W = 2000$ and $s = 1$ (seeds 1–8), *sfx* with $W = 2000$ and $s = 0$ (seeds 1–8), *all* with $W = 8000$ and $s = 1$ (seeds 1–4), and *all* with $W = 2000$ and $s = 2$ (seeds 1–4). None of these runs used the MD filter.

Stage 2, backward splitting (BPBS): Our reimplementation of BPBS [SFX23]. Per seed, 2000 random trajectories of at most 16 splits (16×16), or 300 trajectories of at most 12 splits (32×32), with $q = 0.5$. Every prefix is evaluated by $R = 4$ randomized forward runs with the *sfx* rule and the MD filter.

Stage 3, beam search over the splits: Per seed, $N = 40$ (16×16) or $N = 6$ (32×32) repetitions, with $B = 8$ prefixes kept, $K = 16$ split options drawn per prefix, $R = 2$ forward runs per new prefix, at most 14 splits and $q = 0.5$; *sfx* rule with the MD filter.

Stage 4, large neighborhood search: Per seed, starting from the best XOR2 circuit of stages 1–3, 300 iterations with $W = 1000$ (16×16), or 150 iterations with $W = 200$ (32×32); k uniform in $\{4, \ldots, 28\}$, capped at the number of outputs computed by a gate; $s = 1$; tie rule *all*. A new circuit of equal cost replaces the current circuit. For the 16×16 matrices, $k \geq 16$ in 13 of the 25 values of k, so about half of the iterations keep the last gates of all the outputs.

Our run script applied stages 2–4 only to the matrices of minimum depth 3, i.e., to the 22 matrices of Table 3 other than [SS16] involutory 32×32, and to the 32×32 [LS16] matrix of the HILL repository (Sec. 4). For [SS16] involutory 32×32, of minimum depth 4, we ran stages 2–4 separately, with the same settings and seeds.

Additional runs of stage 4: With the settings above and seeds 1–8, we also started stage 4 from the published circuit of SFX23 with the fewest XOR2 gates for each of the 22 matrices of minimum depth 3, mapped onto the matrix by its row and column permutations, and we ran it with the UMC area after merging (Sec. 2.4) as its objective, from our circuit of the smallest UMC area of the four stages (of stage 1 for [SS16] involutory 32×32), for each of the 23 matrices. These runs and the other runs outside the four stages and Table 6 ran on the same server, with up to 51 runs at a time.

We verified every best circuit three times. First, the program evaluates each circuit that it writes on the unit vectors, against the matrix and the depth bound. Second, a Python script that shares no code with the program re-evaluates every best circuit against the matrix file, before and after merging. Third, every best circuit computes the same linear map on all the unit vectors as the published circuit of HILL for the same matrix, i.e., for the 22 benchmark matrices of the HILL repository and for its [LS16] variant, and as the circuit of SFX23 for the [LS16] matrix of SFX23. In addition, the AES MixColumns circuits were evaluated as MixColumns over $\mathbb{F}_{2^8}$ on the 32 unit vectors. The script that computes the areas (Sec. 2.4) also verifies each of the 9,060 circuits that the 3,020 runs at the minimum depth recorded, three per run, and each of its merged circuits. The areas are sums of the unit areas of Sec. 2.4. The program and these scripts were written with an AI tool (see the section on the use of AI tools); the circuits listed in Appendix A can be checked by hand.

5.2 Results

Table 3 compares the XOR2 counts of our circuits at the minimum depth with Table 5 of SFX23, and their areas after merging with those of the circuits of HILL and of the published circuits of SFX23 after the same merge.

Table 3: XOR2 count and area (GE) at the minimum depth D. SFX23: Table 5 of [SFX23], and their published circuits after our merge; HILL: the circuits of its repository [Li26], which has no STM areas. The areas and our counts are the best for each problem over all our runs; in parentheses, the XOR2 count before merging. †: from another matrix of the problem (Sec. 4); ‡: from stage 4 started from the circuits of SFX23; §: from stage 4 with the UMC area as the objective (Sec. 5.1). Bold: below the previous record (Sec. 5.2), below SFX23 and HILL (UMC), or below SFX23 (STM).

		XOR2 count		UMC 180 nm area			STM 130 nm area	
Matrix	D	SFX23	ours	SFX23	HILL	ours	SFX23	ours
16×16 *matrices*								
SmallScale AES[a]	3	46	**45**‡	128.69 (46)	127.70 (47)	**118.39** (47)	135.83 (46)	**122.51** (47)
Joltik[b]	3	47	**46**	131.69 (47)†	125.38 (48)†	**122.72** (48)§	139.16 (47)†	**127.17** (48)§
[SKOP15] Hadamard	3	47	47	125.04 (47)	125.04 (47)	**122.38** (47)	131.17 (47)	**127.17** (49)
[LS16] circulant	3	43	43	118.36 (43)†	116.04 (44)†	116.04 (44)	124.51 (43)†	**120.51** (46)
[LW16] circulant	3	43	43	118.36 (43)	116.04 (44)	116.04 (44)	124.51 (43)	**120.51** (46)†
[SS16] Toeplitz	3	42	42	114.03 (42)	114.03 (42)	**111.37** (42)	119.85 (42)	**115.85** (45)
[JPST17]	3	43	43	113.04 (43)	113.04 (43)	**111.71** (43)	117.85 (43)	**115.85** (43)
[SKOP15] involutory	3	47	**46**	131.69 (47)†	125.38 (48)	**122.72** (48)§	139.16 (47)†	**127.17** (48)§
[LW16] involutory	3	47	**46**†	131.69 (47)	125.38 (48)	**122.72** (48)§	139.16 (47)	**127.17** (48)§
[SS16] involutory	3	42	**40**	110.04 (42)	110.04 (42)	**109.05** (43)	114.52 (42)	**113.19** (43)
[JPST17] involutory	3	46	46	123.37 (46)	123.37 (46)	**121.05** (47)	129.17 (46)	**125.84** (47)
32×32 *matrices*								
AES MixColumns[c]	3	99	97	264.09 (100)	270.40 (99)	**258.77** (100)§	274.98 (100)	**268.99** (101)§
[SKOP15] Hadamard	3	96	**94**‡	245.44 (96)	245.44 (96)	**244.76** (94)‡	255.68 (96)	**255.01** (97)
[LS16] circulant[d]	3	110	**106**‡	283.45 (110)	–	**274.79** (108)	294.96 (110)	**283.63** (108)
[LW16]	3	101	**98**	261.77 (101)	261.77 (101)	**247.45** (98)‡§	273.66 (101)	**256.34** (98)‡§
[BKL16] circulant	3	107	**104**	278.44 (107)	278.44 (107)	**264.12** (104)	288.96 (107)	**274.32** (104)
[SS16] Toeplitz	3	105	**101**‡	269.78 (105)	272.44 (105)	**259.45** (102)‡	280.31 (105)	**268.99** (102)‡
[JPST17] subfield	3	89	**86**	240.40 (89)	240.40 (89)	**223.42** (86)§	250.34 (89)	**231.70** (86)§
[SKOP15] involutory[e]	3	98	**94**	256.76 (98)	256.76 (98)	**250.42** (95)	270.34 (98)	**260.33** (97)§
[LW16] involutory	3	89	**87**	255.03 (89)	247.42 (94)	**243.06** (89)‡§	269.66 (89)	**253.67** (89)‡§
[SS16] involutory	4	98	**94**	263.41 (98)	266.07 (98)	**244.11** (96)§	278.33 (98)	**253.01** (96)§
[KLSW17][f]	3	86	**80**	230.72 (84)†	230.72 (84)†	**219.40** (82)‡	243.70 (86)	**230.38** (83)‡
[LSL+19] HW168 #130	3	84	**82**	229.39 (84)	229.39 (84)	**221.07** (83)‡§	241.70 (84)	**232.38** (84)

[a] The HILL circuit has [27 XOR2 + 10 XOR3] gates; in parentheses, $27 + 2 \times 10$.
[b] The GitHub repository [Joe26] lists a 46-gate circuit of depth 3 for Joltik, dated October 30, 2026, which we verified.
[c] The NIST collection of AES circuits [NIS26] lists a 97-gate circuit of depth 3, added in 2026.
[d] The circuits of the HILL repository compute a variant of this matrix (Sec. 4); we use the matrix of SFX23.
[e] Table 3 of HILL lists 240.09 GE; we follow its repository (Sec. 4).
[f] SFX23 published 84 gates for [LSL+19] HW160 #93, a row and column permutation of [KLSW17]. Daggers: the circuits of SFX23 and HILL for #93; HILL lists 231.40 GE for [KLSW17].

XOR2 counts: We take as the previous record of a problem the smallest XOR2 count at the minimum depth in Table 5 of SFX23, in the circuits of SFX23 for permutation-equivalent matrices (Sec. 4), and in NIST's collection [NIS26]. None of the XOR2 circuits stored in the HILL repository, and none of the XOR2 counts at the minimum depth in Table 1 of [LZW23] and Table 3 of [KD24], is below these records. With the transfers between permutation-equivalent matrices, our circuits have fewer XOR2 gates than the previous records for 16 of the 23 matrices. These matrices represent 14 of the 20 distinct problems, i.e., 11 of the 12 problems of size 32×32 and 3 of the 8 problems of size 16×16, and the improvements are 1 gate for 2 problems, 2 gates for 4, 3 gates for 3 and 4 gates for 5. For the remaining 6 problems, our XOR2 counts equal the records. None of our counts is above Table 5 of SFX23. The values marked ‡ and § come from the additional runs of stage 4 (Sec. 5.1), and all the others from the four stages, which set 12 of the 14 records. The analyses below cover all our runs at the minimum depth. Appendix A lists the circuit of each new record and of each of our UMC areas below the baseline.

Areas: For each problem, we compare our smallest UMC area with a baseline: the smallest UMC area of the circuits of the HILL repository and of the published circuits of SFX23 after a maximum merge, as ours. Our circuits are smaller than the baseline for 19 of the 20 problems, by 0.3% to 7.3%, and equal to it for [LS16]/[LW16] circulant 16×16 (116.04 GE). Since our UMC merge is a maximum merge, as are the HILL areas for 54 of the 56 matrices of its repository (Sec. 4), these gains come from the XOR2 circuits that are merged; the exception is SmallScale AES, whose HILL circuit is not a merge of the stored XOR2 circuit. For 11 of the 19 problems, i.e., those of size 32×32 other than AES MixColumns, our circuit comes from an XOR2 circuit with fewer gates than the circuit of the baseline. For the remaining 8 problems, i.e., the 7 problems of size 16×16 among the 19 and AES MixColumns, it comes from one with as many gates (6 problems) or more (2 problems), and more of its gates are merged.

With the STM unit areas, our circuits are smaller than the merged circuits of SFX23, which were not optimized for these areas, for all the 20 problems. HILL gives an STM area only for AES MixColumns, 289 GE from synthesis (Table 13 of [LWLG26]); its [59 XOR2 + 20 XOR3] gates sum to 289.67 GE with the STM unit areas, against 268.99 GE for our circuit. The duplication of gates of fan-out two (Sec. 2.4) lowers our smallest STM area for 15 of the 23 matrices, each with its own circuits, by 0.67 to 5.36 GE, against our smallest STM area without duplication, which can come from another circuit.

XOR2 count against area: For 16 of the 23 matrices, our smallest circuit after merging has more XOR2 gates than our circuits with the fewest XOR2 gates, and the smallest merged area among the latter exceeds our smallest merged area by 0.31 to 9.96 GE (UMC). Had we merged only the recorded circuits with the fewest XOR2 gates, our areas would be below the baseline for 16 of the 20 problems and above it for 4 problems, i.e., those of AES MixColumns, SmallScale AES, Joltik and [LS16] circulant 16×16. Only the runs of stage 4 with the UMC objective optimized the area directly; all the other runs used the XOR2 count as the objective.

AES MixColumns: Our best circuit requires 97 XOR2 gates at depth 3, against 99 gates in SFX23. However, NIST's collection of AES circuits [NIS26] also contains a 97-gate circuit of depth 3. Hence, we do not count AES MixColumns among the improvements. Our circuit has 28, 37 and 32 gates at depths 1, 2 and 3, respectively, and a maximum fan-out of 5. After merging, it has [59 XOR2 + 19 XOR3] gates (265.73 GE), while a 100-gate circuit of stage 4 with the UMC objective merges to [38 XOR2 + 31 XOR3] gates (258.77 GE): its 3 additional XOR2 gates (9.00 GE) allow 12 additional merges, which save 15.96 GE. The 100-gate circuit of SFX23 merges to 264.09 GE, the 99-gate circuit of SFX23 that HILL merged gives 270.40 GE, and the 97-gate circuit of NIST merges to [67 XOR2 + 15 XOR3] gates (271.05 GE).

5.3 Results of Each Stage

Table 4 shows the best values of each stage with the settings of Sec. 5.1. The stages used different budgets (Table 7), and stage 4 starts from the best XOR2 circuit of stages 1–3; hence, the table is not a comparison at the same cost. Of the 13 matrices for which the four stages set a new record, stage 1 is below the previous record for 8, stage 2 for 4 and stage 3 for 12, and the final XOR2 count was first reached at stage 4 for 9 of them, at stage 3 for 2 and at stage 1 for 2. Stage 4 lowered the XOR2 count for 10 of the 23 matrices. Our stage 2 reproduces the published count of SFX23 for 14 of the 23 matrices, and it is below it for 5 and above it for 4; since our implementation and budget differ from those of SFX23, these differences cannot be attributed to the method alone. Stage 3 reached fewer XOR2 gates than stage 2 for 10 of the 11 matrices of size 32×32 and depth 3, with a

Table 4: Best XOR2 count and UMC area (GE) of each stage (Sec. 5.1); stage 4 starts from the best XOR2 circuit of stages 1–3, which its values include. Underlined: the best of the four stages; bold: as in Table 3.

	XOR2 count, stage				UMC area (GE), stage			
Matrix	1	2	3	4	1	2	3	4
16×16 *matrices*								
SmallScale AES	47	46	46	46	**125.38**	**122.38**	**118.39**	**121.05**
Joltik	48	48	**46**	**46**	129.37	125.38	**124.05**	125.38
[SKOP15] Hadamard	48	47	47	47	127.05	**122.72**	**122.72**	**122.38**
[LS16] circulant	44	43	43	43	116.04	116.04	116.04	116.04
[LW16] circulant	44	43	43	43	116.04	116.04	116.04	116.04
[SS16] Toeplitz	42	42	42	42	**112.70**	**112.05**	**112.05**	**111.37**
[JPST17]	43	43	43	43	114.71	113.04	113.04	**111.71**
[SKOP15] involutory	48	48	**46**	**46**	129.37	125.38	**124.05**	125.38
[LW16] involutory	48	47	47	47	129.37	125.38	**124.05**	**125.04**
[SS16] involutory	**41**	42	**41**	**40**	110.04	110.04	**109.05**	**109.70**
[JPST17] involutory	46	46	46	46	**122.38**	**122.38**	**121.05**	**121.05**
32×32 *matrices*								
AES MixColumns	98	100	99	97	264.74	**263.10**	**260.10**	**262.42**
[SKOP15] Hadamard	96	97	96	96	245.44	251.10	246.77	245.44
[LS16] circulant	**108**	110	**109**	**107**	**276.12**	**279.46**	**277.79**	**274.79**
[LW16]	**98**	101	**99**	**98**	**251.44**	**256.45**	**254.44**	**248.78**
[BKL16] circulant	108	107	**106**	**104**	281.44	**277.45**	**272.78**	**264.12**
[SS16] Toeplitz	**103**	105	**103**	**102**	**262.45**	**264.80**	**263.13**	**262.11**
[JPST17] subfield	90	**88**	**87**	**86**	**235.42**	**230.75**	**226.42**	**224.75**
[SKOP15] involutory	**96**	**97**	**96**	**94**	257.10	258.09	**250.76**	**250.42**
[LW16] involutory	**88**	89	**88**	**87**	**246.06**	**246.74**	**246.06**	**245.04**
[SS16] involutory	**96**	**97**	99	**94**	**251.10**	**257.75**	**259.76**	**246.77**
[KLSW17]	84	85	**83**	**80**	**226.73**	**227.41**	**222.74**	**219.74**
[LSL+19] HW168 #130	**82**	**82**	**82**	**82**	**224.72**	**224.72**	**221.41**	**222.06**

similar number of forward runs (1.33 against 1.27 million), but the two stages also differ in their maximum number of splits and in R (Sec. 5.1), so this comparison is not controlled. Started from the circuits of SFX23, stage 4 reached fewer XOR2 gates than the four stages for 4 of the 22 matrices of depth 3: 45 for SmallScale AES, with 6 of its 8 seeds, and 94, 106 and 101 for [SKOP15] Hadamard, [LS16] circulant and [SS16] Toeplitz 32×32, against 46, 96, 107 and 102. For SmallScale AES, the four stages had only matched the previous record, 46, and our stage 4 had also started from a 46-gate circuit; hence, its result depends on the starting circuit, not only on its XOR2 count. With the UMC area as its objective, stage 4 lowered the smallest UMC area of the four stages for 10 of the 23 matrices, by 0.34 to 2.66 GE. In Table 5, the best value of four configurations of the forward beam search, with 8 seeds each, is reached, alone or tied, by the *all* rule for 19 of the 23 matrices, by the *bp* rule for 16, and by the *sfx* rule for 10 with $s = 0$ and 16 with $s = 1$; no rule is the best for all the matrices.

5.4 Tie Exploration against Random Restarts

For every matrix, tie rule and seed from 1 to 4, we ran one beam search with $W = 2000$, and then independent randomized forward runs with the same tie rule, i.e., random restarts without backward splitting, until they used the same CPU time (user plus system time). These runs are separate from those of Table 3; they ran on the server of Sec. 5.1, with up to 48 runs at a time. Table 6 gives the mean XOR2 counts. With the *all* rule, i.e.,

Table 5: Forward beam search by configuration rule/W/s (tie rule, width, slack; *sfx* without the MD filter), with the number of seeds below: best XOR2 count (mean). The first column uses seeds 1–8 of the *all*/2000/1 runs of stage 1, the last one all 32; *sfx*/2000/1 is not part of stage 1 (Sec. 5.1). Bold: the best of the first four columns.

Matrix	all/2000/1 (8)	bp/2000/1 (8)	sfx/2000/0 (8)	sfx/2000/1 (8)	all/8000/1 (4)	all/2000/2 (4)	all/2000/1 (32)
16×16 *matrices*							
SmallScale AES	**47** (47.25)	**47** (47.38)	48 (48.00)	**47** (47.00)	47 (47.00)	47 (47.50)	47 (47.12)
Joltik	**48** (48.00)	**48** (48.00)	**48** (48.00)	**48** (48.00)	48 (48.00)	48 (48.00)	48 (48.00)
[SKOP15] Hadamard	**48** (48.00)	**48** (48.00)	**48** (48.88)	**48** (48.25)	48 (48.00)	48 (48.00)	48 (48.00)
[LS16] circulant	44 (44.00)	44 (44.75)	44 (44.75)	**43** (43.62)	44 (44.00)	44 (44.00)	44 (44.00)
[LW16] circulant	44 (44.00)	44 (44.00)	44 (44.62)	**43** (43.88)	44 (44.00)	44 (44.00)	44 (44.00)
[SS16] Toeplitz	**43** (43.00)	**43** (43.00)	44 (44.00)	**43** (43.00)	42 (42.75)	43 (43.00)	42 (43.19)
[JPST17]	**45** (45.00)	**45** (45.00)	**45** (45.00)	**45** (45.00)	44 (44.75)	45 (45.00)	43 (44.78)
[SKOP15] involutory	**48** (48.00)	**48** (48.00)	**48** (48.00)	**48** (48.00)	48 (48.00)	48 (48.00)	48 (48.00)
[LW16] involutory	**48** (48.00)	**48** (48.00)	**48** (48.00)	**48** (48.00)	48 (48.00)	48 (48.00)	48 (48.00)
[SS16] involutory	**43** (43.00)	**43** (43.00)	44 (44.00)	**43** (43.00)	43 (43.00)	41 (41.25)	43 (43.00)
[JPST17] involutory	**46** (46.00)	**46** (46.00)	47 (47.00)	**46** (46.00)	46 (46.00)	46 (46.00)	46 (46.00)
32×32 *matrices*							
AES MixColumns	**99** (99.62)	**99** (99.62)	100 (101.00)	100 (100.62)	99 (99.25)	99 (99.75)	98 (99.31)
[SKOP15] Hadamard	97 (97.38)	99 (99.12)	**96** (96.38)	**96** (96.12)	96 (96.50)	96 (96.50)	96 (97.22)
[LS16] circulant	**108** (108.88)	**108** (108.75)	**108** (108.62)	**108** (108.75)	108 (108.25)	108 (109.00)	108 (108.91)
[LW16]	**98** (99.12)	**98** (99.12)	99 (99.12)	**98** (98.88)	98 (98.50)	98 (98.75)	98 (99.31)
[BKL16] circulant	**108** (108.00)	**108** (108.50)	**108** (108.00)	**108** (108.00)	108 (108.00)	108 (108.00)	108 (108.03)
[SS16] Toeplitz	**103** (104.12)	**103** (103.50)	**103** (103.38)	104 (105.12)	103 (103.50)	103 (103.75)	103 (103.94)
[JPST17] subfield	90 (90.00)	90 (91.50)	90 (90.38)	**89** (89.88)	90 (90.00)	90 (90.00)	90 (90.00)
[SKOP15] involutory	**96** (96.50)	**96** (96.25)	98 (98.62)	97 (97.62)	96 (96.00)	96 (96.50)	96 (96.47)
[LW16] involutory	**89** (89.62)	**89** (89.00)	**89** (90.62)	90 (90.38)	88 (88.00)	89 (90.50)	89 (89.78)
[SS16] involutory	**97** (98.38)	99 (99.00)	100 (101.25)	99 (99.25)	98 (98.50)	96 (97.00)	97 (98.41)
[KLSW17]	**85** (86.38)	89 (89.00)	88 (88.00)	86 (86.00)	84 (84.75)	85 (86.50)	84 (86.12)
[LSL+19] HW168 #130	**82** (83.50)	83 (83.00)	86 (86.00)	84 (84.00)	82 (82.00)	82 (82.00)	82 (83.06)

without a norm-based criterion for the ties, as in Improved-BP, beam search has the lower mean for 10 of the 12 problems of size 32×32, both with $s = 0$ and with $s = 1$; with $s = 0$, for 6 of the 12, no restart reached the median result of beam search, even with the restarts of the 4 seeds pooled. The random restarts give the lower mean for [LSL+19] HW168 #130 ($s = 0$) or [SKOP15] Hadamard ($s = 1$), and for 3 of the 8 problems of size 16×16 with $s = 0$ ([SS16] Toeplitz, [JPST17] and [SS16] involutory). With the *sfx* rule and the MD filter of SFX23 ($s = 0$), the random restarts give the lower mean for 12 of the 20 problems, by up to 8.50 gates for [LW16] involutory 32×32. With this filter, the near-ties of an iteration come from few sequences (for the 16×16 matrices, a median of 3 to 135, against 12.5 to 1,248 without it), and the beam search does not improve with its width: for [LW16] involutory 32×32, $W = 1$, 10, 100 and 2000 give means of 97.25, 98.00, 98.75 and 98.00 gates.

5.5 Computation Time

The run time of the four stages is shown in Table 7. Stage 1 took 437.2 h in total over 1,344 runs, i.e., 56 runs for each of the 24 matrices, including the two 32×32 [LS16] matrices. Its longest run took 6.3 h for [SS16] involutory 32×32 at depth 4, while the longest run for a matrix of depth 3 took 3.5 h ([SS16] Toeplitz 32×32). Stages 2 and 3 took the most time, because every prefix of splits is evaluated by complete forward runs; e.g., a stage-2 run for a 32×32 matrix evaluates up to 3,600 prefixes by 4 forward runs each, and the stage-2 runs of AES MixColumns took 1,005 to 1,046 minutes each. In total, a 32×32 matrix of depth 3 used 150.0 h of single-thread run time on average, under the load of the server. The additional runs of stage 4 took 123.0 h of CPU time from the circuits of SFX23 (176 runs) and 216.1 h with the UMC objective (184 runs). SFX23 repeated each method tens of thousands of times per matrix, and reported that, with 360 threads, finding their best result took less than two hours for each 32×32 matrix (Sec. 5 of [SFX23]), i.e., up to about 720 thread-hours to their best result. They do not report their total computation time or their hardware, so the two budgets cannot be compared.

Table 6: Mean XOR2 count over 4 seeds at the minimum depth D: one beam search ($W = 2000$) per seed against random restarts, i.e., forward runs with the same tie rule (and no slack), repeated until they used the CPU time of the beam search. Bold: the lower mean. Last rows: the problems for which the beam mean is lower, equal or higher (equivalent matrices count once, Sec. 4).

			all, $s = 0$		*all*, $s = 1$		*sfx* + MD, $s = 0$	
Matrix	size	D	beam	restarts	beam	restarts	beam	restarts
SmallScale AES	16×16	3	48.00	48.00	**47.25**	48.00	48.00	48.00
Joltik	16×16	3	**48.00**	49.25	**48.00**	48.25	**49.00**	49.50
[SKOP15] Hadamard	16×16	3	48.00	48.00	48.00	48.00	50.00	**49.00**
[LS16] circulant	16×16	3	**44.00**	44.50	44.00	44.00	46.00	**44.00**
[LW16] circulant	16×16	3	**44.00**	44.25	44.00	44.00	46.00	**44.00**
[SS16] Toeplitz	16×16	3	44.00	**43.00**	43.00	43.00	44.00	44.00
[JPST17]	16×16	3	45.00	**44.25**	45.00	**44.00**	45.00	**44.50**
[SKOP15] involutory	16×16	3	**48.00**	49.25	**48.00**	48.25	**49.00**	49.50
[LW16] involutory	16×16	3	**48.00**	49.00	**48.00**	49.00	**49.00**	49.25
[SS16] involutory	16×16	3	44.00	**42.75**	43.00	**41.75**	46.00	**44.00**
[JPST17] involutory	16×16	3	47.00	47.00	**46.00**	47.00	48.00	**47.00**
AES MixColumns	32×32	3	**100.00**	102.25	**100.00**	102.25	**99.00**	101.25
[SKOP15] Hadamard	32×32	3	**97.00**	97.50	97.50	**97.00**	100.00	**98.00**
[LS16] circulant	32×32	3	**109.00**	110.00	**108.50**	109.75	110.00	110.00
[LW16]	32×32	3	**99.50**	102.00	**99.00**	102.00	**99.25**	101.75
[BKL16] circulant	32×32	3	**108.00**	108.50	**108.00**	108.50	110.25	**108.75**
[SS16] Toeplitz	32×32	3	**104.50**	105.75	**104.00**	105.25	104.25	**104.00**
[JPST17] subfield	32×32	3	90.00	90.00	90.00	90.00	90.00	**89.75**
[SKOP15] involutory	32×32	3	**97.50**	99.75	**96.25**	99.75	**98.00**	99.00
[LW16] involutory	32×32	3	**90.75**	92.50	**90.00**	91.50	98.00	**89.50**
[SS16] involutory	32×32	4	**99.75**	101.00	**98.25**	100.50	99.00	**98.00**
[KLSW17]	32×32	3	**88.00**	89.75	**85.75**	88.50	92.00	**89.75**
[LSL+19] HW168 #130	32×32	3	89.00	**87.75**	**83.75**	85.50	**86.00**	87.25
lower / equal / higher, 16×16 (8)			2 / 3 / 3		3 / 3 / 2		1 / 2 / 5	
lower / equal / higher, 32×32 (12)			10 / 1 / 1		10 / 1 / 1		4 / 1 / 7	

Table 7: Wall-clock time of the single-threaded runs of each stage on the server (Sec. 5.1), up to 48 at a time; for stages 2–4, from the creation of the log to its last write. The 32×32 columns include the [LS16] matrix of the HILL repository. Not included: [SS16] involutory 32×32, with medians of 152.4, 517, 212 and 593 min per run for stages 1–4.

	16×16 (11 matrices)			32×32, depth 3 (12 matrices)		
Stage	runs	median	total	runs	median	total
1, beam search over the near-ties	616	26 s	4.8 h	672	20.6 min	286.8 h
2, backward splitting (BPBS)	88	38 min	50.9 h	96	388 min	745.3 h
3, beam search over the splits	88	26 min	34.8 h	96	347 min	626.2 h
4, large neighborhood search	88	2 min	3.4 h	96	78 min	141.5 h
Total	880		93.9 h	960		1799.8 h

Note that the run time depends on the load of the server. The stage-2 runs of the 32×32 [LS16] matrix of SFX23, which ran with 8 runs at a time, took 346 to 355 minutes each, against 559 to 574 minutes for the [LS16] matrix of the HILL repository, which ran with up to 48 runs at a time.

6 Limitations

Scaling with the row weight: The distances are computed by exhaustive search (Sec. 2.3), whose cost grows quickly with the distances. In the first iteration, the search enumerates, for a target at distance δ of a 32×32 matrix, all the $\binom{32}{\delta-1}$ subsets of $\delta - 1$ inputs: at most $\binom{32}{5} \approx 2.0 \times 10^5$ for AES MixColumns (depth 3), against $\binom{32}{17} \approx 5.7 \times 10^8$ for each row of weight 19 of AES InvMixColumns (depth 5). One forward run of our program on InvMixColumns at depth 5 took 5.8 to 11.4 h of CPU time and gave 154 to 163 XOR2 gates, depending on the tie rule, while NIST lists 119 gates [NIS26]; Huang et al. computed these distances faster with a meet-in-the-middle approach on GPUs, and report 157 gates at depth 5 with the heuristic of [LSL+19] [HSWC26]. We did not run our stages on InvMixColumns.

Larger depth bounds: Our search targets the minimum depth, which stages 2 and 3 assume. For AES MixColumns, a beam search with $W = 500$ gives 97, 95 and 95 XOR2 gates under the depth bounds 4, 5 and 8, which stage 4 does not lower, while NIST lists 90 and 88 gates at depths 4 and 5 [NIS26], and the 88-gate circuit of Jean [Jea26] has depth 7 (by our evaluation of its listing).

Optimality and scope: We give no lower bounds, and we evaluated only the 23 matrices of the benchmark at their minimum depth; HILL also evaluated the 4,254 MDS matrices of [LSL+19].

Choices in the beam search: The ranking by Δ alone (Sec. 3.2) ignores the number of targets already computed. Ranking by the number of gates so far plus Δ gave fewer, as many and more XOR2 gates in 27, 53 and 8 of the 88 runs of the main configuration of stage 1 (seeds 1–8) on the matrices of size 16×16, and in 35, 32 and 29 of the 96 runs on those of size 32×32. We have not tested this ranking in stage 4, nor two other choices: the completed circuits count among the W sequences, and sequences are compared by a hash of their gate values.

Area model: Our areas are sums of unit areas, not synthesis results, and our depth bound counts gate levels; it reflects neither that the delay of a gate grows with its fan-out, nor that an XOR3 gate is slower than an XOR2 gate. We merge the three circuits recorded per run, not all the completed circuits, and we use no other rewriting, e.g., the depth-reducing Rules 3 and 4 of HILL [LWLG26] or gates with more inputs [LWS+22, KD24]. Without XOR3 gates, only the XOR2 results apply.

Computation budget: Our budget (Sec. 5.5) differs from that of SFX23, and we did not run their code, or our stage 2, with it. Hence, Sec. 5.3 shows which stages reach the records with our settings, not what the method of SFX23 would find with our budget, and the comparison of Sec. 5.4 fixes the CPU time for the forward search only.

7 Conclusions and Future Work

We have combined beam search over the near-ties of the forward heuristic and over the backward splits of SFX23 with a large neighborhood search, for depth-bounded SLP. With this combination, we have lowered the records for 14 of the 20 distinct problems in the benchmark of [SFX23], and after merging, our circuits have smaller areas than both the HILL circuits [LWLG26] and the merged SFX23 circuits for 19 of them. At equal CPU time, beam search over the near-ties was better than random restarts on most 32×32 matrices when the ties are explored without a norm-based criterion, but not with the rule and the filter of SFX23. When the area is the goal, the circuits should be selected by their area after merging: merging only the circuits of fewest XOR2 gates would have missed our smallest area for 16 matrices. As future work, we will investigate faster depth-aware distances for heavy rows, e.g., for AES InvMixColumns and merged round functions.

Use of AI Tools

In accordance with the IACR policy on the use of AI tools by authors, we disclose that we used an AI assistant (Claude Opus 5.5, by Anthropic) under our instructions for the scripts that run, summarize and verify the experiments; the tables and gate counts; checking against published values; and drafting of all the sections, which we revised with it over several review cycles. All the search runs were done by the authors, who take full responsibility for the content of this paper, including the text, tables and code produced with the tool.

Acknowledgments

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) through two Discovery Grants awarded to Mostafa Taha and Arash Reyhani-Masoleh.

References

[BDK+21] Anubhab Baksi, Vishnu Asutosh Dasu, Banashri Karmakar, Anupam Chattopadhyay, and Takanori Isobe. Three input exclusive-OR gate support for Boyar-Peralta's algorithm. In *Progress in Cryptology – INDOCRYPT 2021*, volume 13143 of *Lecture Notes in Computer Science*, pages 141–158. Springer, 2021. doi:10.1007/978-3-030-92518-5_7.

[BFI19] Subhadeep Banik, Yuki Funabiki, and Takanori Isobe. More results on shortest linear programs. In *Advances in Information and Computer Security – IWSEC 2019*, volume 11689 of *Lecture Notes in Computer Science*, pages 109–128. Springer, 2019. doi:10.1007/978-3-030-26834-3_7.

[BFI21] Subhadeep Banik, Yuki Funabiki, and Takanori Isobe. Further results on efficient implementations of block cipher linear layers. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 104-A(1):213–225, 2021.

[BFP19] Joan Boyar, Magnus Gausdal Find, and René Peralta. Small low-depth circuits for cryptographic applications. *Cryptography and Communications*, 11(1):109–127, 2019. doi:10.1007/s12095-018-0296-3.

[BKL16] Christof Beierle, Thorsten Kranz, and Gregor Leander. Lightweight multiplication in GF(2^n) with applications to MDS matrices. In Matthew Robshaw

and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 625–653. Springer, 2016.

[BMP08] Joan Boyar, Philip Matthews, and René Peralta. On the shortest linear straight-line program for computing linear forms. In Edward Ochmanski and Jerzy Tyszkiewicz, editors, *Mathematical Foundations of Computer Science 2008, 33rd International Symposium, MFCS 2008*, volume 5162 of *Lecture Notes in Computer Science*, pages 168–179. Springer, 2008.

[BMP13] Joan Boyar, Philip Matthews, and René Peralta. Logic minimization techniques with applications to cryptology. *J. Cryptol.*, 26(2):280–312, 2013.

[BP10] Joan Boyar and René Peralta. A new combinational logic minimization technique with applications to cryptology. In Paola Festa, editor, *Experimental Algorithms, 9th International Symposium, SEA 2010*, volume 6049 of *Lecture Notes in Computer Science*, pages 178–189. Springer, 2010.

[CMR05] Carlos Cid, Sean Murphy, and Matthew J. B. Robshaw. Small scale variants of the AES. In Henri Gilbert and Helena Handschuh, editors, *Fast Software Encryption: 12th International Workshop, FSE 2005*, volume 3557 of *Lecture Notes in Computer Science*, pages 145–162. Springer, 2005.

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES – The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002. doi:10.1007/978-3-662-04722-4.

[HSWC26] Xiao Huang, Chao Sun, Ge Wu, and Liquan Chen. GPU acceleration for Boyar-Peralta algorithm in high dimensional linear layers. *IACR Trans. Symmetric Cryptol.*, 2026(3), 2026. URL: https://tosc.iacr.org/index.php/ToSC/article/view/13387, doi:10.46586/tosc.anjbhe8nm2-x.

[Jea26] Jérémy Jean. 88-XOR implementation of the AES MixColumns matrix. Cryptology ePrint Archive, Paper 2026/1481, 2026. URL: https://eprint.iacr.org/2026/1481.

[JNP15] Jérémy Jean, Ivica Nikolić, and Thomas Peyrin. Joltik v1.3. Submission to the CAESAR competition, https://competitions.cr.yp.to/round2/joltikv13.pdf, 2015.

[Joe26] Joe-b-20. xor-circuit-research: small, shallow XOR circuits for cipher linear maps. GitHub repository, https://github.com/Joe-b-20/xor-circuit-research, 2026. Accessed November 10, 2026.

[JPST17] Jérémy Jean, Thomas Peyrin, Siang Meng Sim, and Jade Tourteaux. Optimizing implementations of lightweight building blocks. *IACR Trans. Symmetric Cryptol.*, 2017(4):130–168, 2017.

[KD24] Meltem Kurt Pehlivanoğlu and Mehmet Ali Demir. Optimizing implementations of linear layers using two and higher input XOR gates. *PeerJ Computer Science*, 10:e1820, 2024. doi:10.7717/peerj-cs.1820.

[KLSW17] Thorsten Kranz, Gregor Leander, Ko Stoffelen, and Friedrich Wiemer. Shorter linear straight-line programs for MDS matrices. *IACR Trans. Symmetric Cryptol.*, 2017(4):188–211, 2017.

[Li26] Yongchao Li. Linear_Layers_by_YongchaoLi: circuits of the paper "HILL: Efficient low-latency implementations of linear layers via heuristic search". GitHub repository, https://github.com/13est-Wishes/Linear_Layers_by_YongchaoLi, 2026. Accessed October 2026.

[LS16] Meicheng Liu and Siang Meng Sim. Lightweight MDS generalized circulant matrices. In Thomas Peyrin, editor, *Fast Software Encryption – 23rd International Conference, FSE 2016*, volume 9783 of *Lecture Notes in Computer Science*, pages 101–120. Springer, 2016.

[LSL+19] Shun Li, Siwei Sun, Chaoyun Li, Zihao Wei, and Lei Hu. Constructing low-latency involutory MDS matrices with lightweight circuits. *IACR Trans. Symmetric Cryptol.*, 2019(1):84–117, 2019.

[LW16] Yongqiang Li and Mingsheng Wang. On the construction of lightweight circulant involutory MDS matrices. In Thomas Peyrin, editor, *Fast Software Encryption – 23rd International Conference, FSE 2016*, volume 9783 of *Lecture Notes in Computer Science*, pages 121–139. Springer, 2016.

[LWF+22] Qun Liu, Weijia Wang, Yanhong Fan, Lixuan Wu, Ling Sun, and Meiqin Wang. Towards low-latency implementation of linear layers. *IACR Trans. Symmetric Cryptol.*, 2022(1):158–182, 2022. doi:10.46586/tosc.v2022.i1.158-182.

[LWLG26] Yongchao Li, Yongzhuang Wei, Lang Li, and Jian Guo. HILL: Efficient low-latency implementations of linear layers via heuristic search. *IACR Trans. Symmetric Cryptol.*, 2026(1):506–526, 2026. doi:10.46586/tosc.v2026.i1.506-526.

[LWS+22] Qun Liu, Weijia Wang, Ling Sun, Yanhong Fan, Lixuan Wu, and Meiqin Wang. More inputs makes difference: Implementations of linear layers using gates with more than two inputs. *IACR Trans. Symmetric Cryptol.*, 2022(2):351–378, 2022. doi:10.46586/tosc.v2022.i2.351-378.

[LXZZ21] Da Lin, Zejun Xiang, Xiangyong Zeng, and Shasha Zhang. A framework to optimize implementations of matrices. In Kenneth G. Paterson, editor, *Topics in Cryptology – CT-RSA 2021*, volume 12704 of *Lecture Notes in Computer Science*, pages 609–632. Springer, 2021.

[LZW23] Qun Liu, Zheng Zhao, and Meiqin Wang. Improved heuristics for low-latency implementations of linear layers. In Mike Rosulek, editor, *Topics in Cryptology – CT-RSA 2023*, volume 13871 of *Lecture Notes in Computer Science*, pages 524–550. Springer, 2023.

[Max19] Alexander Maximov. AES MixColumn with 92 XOR gates. Cryptology ePrint Archive, Paper 2019/833, 2019. URL: https://eprint.iacr.org/2019/833.

[ME19] Alexander Maximov and Patrik Ekdahl. New circuit minimization techniques for smaller and faster AES SBoxes. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(4):91–125, 2019. doi:10.13154/tches.v2019.i4.91-125.

[NIS01] NIST. Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197, 2001. Updated May 9, 2023. doi:10.6028/NIST.FIPS.197-upd1.

[NIS26] NIST. Circuit Complexity Project: circuits for AES (MixColumns, InvMixColumns and S-box). https://github.com/usnistgov/Circuits/tree/master/data/slp/aes, 2026. Accessed October 29, 2026, commit 0876577.

[RMTA18] Arash Reyhani-Masoleh, Mostafa Taha, and Doaa Ashmawy. Smashing the implementation records of AES S-box. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(2):298–336, 2018. doi:10.13154/tches.v2018.i2.298-336.

[SFX23] Haotian Shi, Xiutao Feng, and Shengyuan Xu. A framework with improved heuristics to optimize low-latency implementations of linear layers. *IACR Trans. Symmetric Cryptol.*, 2023(4):489–510, 2023. Source code and results: https://gitee.com/Haotian-Shi/A_framework_with_improved_heuristics_to_optimize_low-latency_implementations_of_linear_layers. doi:10.46586/tosc.v2023.i4.489-510.

[SKOP15] Siang Meng Sim, Khoongming Khoo, Frédérique E. Oggier, and Thomas Peyrin. Lightweight MDS involution matrices. In Gregor Leander, editor, *Fast Software Encryption – 22nd International Workshop, FSE 2015*, volume 9054 of *Lecture Notes in Computer Science*, pages 471–493. Springer, 2015.

[SS16] Sumanta Sarkar and Habeeb Syed. Lightweight diffusion layer: Importance of Toeplitz matrices. *IACR Trans. Symmetric Cryptol.*, 2016(1):95–113, 2016.

[SYL25] Yao Sun, Runhe Yang, and Ting Li. Revisit the Boyar-Peralta algorithm to solve the shortest linear program problem. Cryptology ePrint Archive, Paper 2025/1493, 2025. URL: https://eprint.iacr.org/2025/1493.

[TP20] Quan Quan Tan and Thomas Peyrin. Improved heuristics for short linear programs. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(1):203–230, 2020. doi:10.13154/tches.v2020.i1.203-230.

[XZL+20] Zejun Xiang, Xiangyong Zeng, Da Lin, Zhenzhen Bao, and Shasha Zhang. Optimizing implementations of linear layers. *IACR Trans. Symmetric Cryptol.*, 2020(2):120–145, 2020. doi:10.13154/tosc.v2020.i2.120-145.

A Supplementary Material: Circuits

This appendix lists the XOR2 circuit of each of the 14 problems with a new record in Table 3 (Appendix A.1), and our circuit of each of the 19 UMC areas of Table 3 that are below the baseline (Appendix A.2). For comparison, each entry gives the previous record (Appendix A.1) or the baseline (Appendix A.2) of its problem (Sec. 5.2); a baseline comes from HILL, from SFX23 (its published circuits after our merge), or from both. The circuits of Appendix A.1 are selected by their XOR2 count, those of Appendix A.2 by their UMC area after merging, and the two measures are competing objectives (Sec. 2.4). Hence, for 10 of the 14 matrices of Appendix A.1, our circuit of Appendix A.2 comes from an XOR2 circuit with one to three more gates (in parentheses in Table 3), and only for [SKOP15] Hadamard and [BKL16] circulant 32×32 is it the merge of the circuit of Appendix A.1. For example, only 5 of the 29 non-output gates of the 45-gate circuit of SmallScale AES have fan-out one: merging them gives [35 XOR2 + 5 XOR3] gates, i.e., 128.35 GE instead of 135.00 GE. The 47-gate circuit of Appendix A.2 has 17 such gates: merging them gives [13 XOR2 + 17 XOR3] gates, i.e., 118.39 GE instead of 141.00 GE (see also "XOR2 count against area" in Sec. 5.2). Each entry of Appendix A.2 also gives the area after merging of our circuit with the fewest XOR2 gates: the circuit of Appendix A.1 or, for the five problems of Appendix A.2 without a new record, the one of our repository. Among our circuits with the record count, Appendix A.1 lists one of the four stages whenever they reach it, as Table 3 credits them; for [SS16] involutory 16×16, [LW16] and [JPST17] subfield 32×32, another of these circuits merges into a smaller area, and Sec. 5.2 uses the smallest one.

Our repository holds these circuits and all the other circuits of the paper. Each matrix is printed row by row: the row of y_j lists the coefficients of $x_0, x_1, \ldots, x_{n-1}$ from left to right (column c of M is x_c, Sec. 2.1), in groups of four bits for $n = 16$ and eight bits for $n = 32$. For example, the row of y_0 of SmallScale AES, `0001 1001 1000 1000`, gives $y_0 = x_3 \oplus x_4 \oplus x_7 \oplus x_8 \oplus x_{12}$. The circuit files of our repository number the inputs from 1. In the listings, $a + b$ denotes the XOR of a and b, and an XOR3 gate is written $a + (b + c)$, where $b + c$ is the XOR2 gate that it absorbed; hence, the XOR2 circuit before merging, and its depth, can be read from the listing. The inputs are $x_0, \ldots, x_{n-1}$, the outputs are $y_0, \ldots, y_{m-1}$, and the other gates are $g_1, g_2, \ldots$, in the order of the circuit. The script that writes these listings checks each circuit against its matrix, its gate counts and area, and its depth, as do the checks of Sec. 5.1.

A.1 XOR2 Circuits of the New Records

SmallScale AES (16×16), $D = 3$: 45 XOR2 gates, previous record 46. From stage 4 started from the circuit of SFX23 (Sec. 5.1). Matrix rows $y_0, \ldots, y_{15}$:

```
y0   0001 1001 1000 1000   y4   1000 0001 1001 1000   y8   1000 1000 0001 1001   y12  1001 1000 1000 0001
y1   1001 1101 0100 0100   y5   0100 1001 1101 0100   y9   0100 0100 1001 1101   y13  1101 0100 0100 1001
y2   0100 0110 0010 0010   y6   0010 0100 0110 0010   y10  0010 0010 0100 0110   y14  0110 0010 0010 0100
y3   0010 0011 0001 0001   y7   0001 0010 0011 0001   y11  0001 0001 0010 0011   y15  0011 0001 0001 0010
```

```
g1 = x6 + x2          g9 = x2 + g8          g16 = g14 + g15       g23 = g19 + g21       y15 = g4 + g18
g2 = x11 + x7         g10 = x15 + x11       y8 = g11 + g16        g24 = x13 + x1        y13 = g6 + g20
g3 = x8 + g2          g11 = x8 + g10        y12 = g15 + g6        g25 = x10 + g24       y2 = g25 + g22
g4 = g1 + g2          g12 = x14 + x10       g17 = x14 + x6        g26 = x9 + x4         y14 = g1 + g25
y3 = x15 + g4         g13 = g10 + g12       g18 = x3 + g17        g27 = g24 + g26       y7 = g18 + g13
g5 = x3 + x0          y6 = g12 + g9         g19 = x12 + x1        g28 = g21 + g26       y10 = g22 + g9
g6 = x15 + g5         g14 = x12 + x0        g20 = g8 + g19        g29 = x3 + g2         y9 = g11 + g23
g7 = x7 + g5          y4 = g14 + g3         g21 = x13 + x5        y11 = g13 + g29       y1 = g7 + g28
g8 = x9 + x5          g15 = x8 + x4         g22 = g17 + g21       y5 = g3 + g27         y0 = g7 + g16
```

Joltik (16×16), $D = 3$: 46 XOR2 gates, previous record 47. [SKOP15] involutory 16×16 is the same matrix, and [LW16] involutory 16×16 is a row and column permutation of it (Sec. 4). As noted in Table 3, the GitHub repository [Joe26] also lists a 46-gate circuit of depth 3. Matrix rows $y_0, \ldots, y_{15}$:

```
y0   1000 0010 1100 1110   y4   0010 1000 1110 1100   y8   1100 1110 1000 0010   y12  1110 1100 0010 1000
y1   0100 0011 0010 0001   y5   0011 0100 0001 0010   y9   0010 0001 0100 0011   y13  0001 0010 0011 0100
y2   0010 1001 0001 1000   y6   1001 0010 1000 0001   y10  0001 1000 0010 1001   y14  1000 0001 1001 0010
y3   0001 0100 1000 1100   y7   0100 0001 1100 1000   y11  1000 1100 0001 0100   y15  1100 1000 0100 0001
```

```
g1 = x10 + x3         g11 = g4 + g10        y5 = g18 + g19        g26 = x9 + g13        y15 = g7 + g17
g2 = x13 + x11        g12 = x14 + x11       g20 = x13 + x8        g27 = g10 + g24       y7 = g17 + g5
g3 = x6 + g1          y14 = g12 + g5        g21 = x12 + g20       y9 = g26 + g27        y11 = g2 + g9
g4 = x8 + x0          g13 = x10 + x2        y3 = g18 + g21        g28 = x1 + g22        y12 = g9 + g15
g5 = x7 + g4          g14 = x12 + x1        g22 = x14 + x6        y1 = g27 + g28        y13 = g2 + g3
g6 = x12 + x4         g15 = g13 + g14       g23 = g16 + g22       g29 = g6 + g20        y6 = g3 + g11
g7 = x15 + g6         g16 = x9 + x0         y0 = g21 + g23        y4 = g26 + g29
g8 = x5 + x4          g17 = g14 + g16       g24 = x14 + x7        g30 = g4 + g8
g9 = x0 + g8          g18 = x5 + x3         g25 = g6 + g24        y8 = g28 + g30
g10 = x15 + x10       g19 = x2 + g12        y2 = g19 + g25        y10 = g1 + g7
```

[SS16] involutory (16×16), $D = 3$: 40 XOR2 gates, previous record 42. Matrix rows $y_0, \ldots, y_{15}$:

```
y0   1000 0001 0010 1000   y4   0001 1000 1000 0010   y8   0100 0001 1000 0001   y12  0001 0100 0001 1000
y1   0100 1001 0011 0100   y5   1001 0100 0100 0011   y9   0110 1001 0100 1001   y13  1001 0110 1001 0100
y2   0010 0100 1001 0010   y6   0100 0010 0010 1001   y10  0011 0100 0010 0100   y14  0100 0011 0100 0010
y3   0001 0010 0100 0001   y7   0010 0001 0001 0100   y11  1001 0010 0001 0010   y15  0010 1001 0010 0001
```

```
g1 = x8 + x6          g9 = x7 + x1          g16 = g14 + g15       y2 = g6 + g20         g24 = g1 + g14
g2 = x15 + x9         g10 = x15 + x6        y1 = g13 + g16        y4 = g12 + g20        y13 = g23 + g24
g3 = x12 + g2         g11 = g9 + g10        y7 = g7 + g16         g21 = x11 + x0        y0 = x0 + g8
g4 = x14 + g2         y6 = g8 + g11         g17 = x14 + x3        y11 = g21 + g18       y8 = g1 + g11
g5 = x5 + x3          g12 = x4 + x2         g18 = x6 + g17        g22 = x15 + g12       y10 = g6 + g16
g6 = x11 + g5         g13 = g9 + g12        y3 = g4 + g18         y15 = g7 + g22        y9 = g13 + g3
g7 = x10 + x7         g14 = x13 + x11       g19 = x8 + x2         g23 = x0 + g5         y14 = g11 + g4
g8 = x12 + g7         g15 = x10 + x2        g20 = g17 + g19       y5 = g4 + g23         y12 = x12 + g6
```

[SKOP15] Hadamard (32×32), $D = 3$: 94 XOR2 gates, previous record 96. From stage 4 started from the circuit of SFX23 (Sec. 5.1). Matrix rows $y_0, \ldots, y_{31}$:

```
y0   10000000 00010000 01000000 11000000        y16  01000000 11000000 10000000 00010000
y1   01000000 10010000 01100000 00100000        y17  01100000 00100000 01000000 10010000
y2   00100000 01000000 00110000 00010000        y18  00110000 00010000 00100000 01000000
y3   00010000 00100000 10010000 10000000        y19  10010000 10000000 00010000 00100000
y4   00001000 00000001 00000100 00001100        y20  00000100 00001100 00001000 00000001
y5   00000100 00001001 00000110 00000010        y21  00000110 00000010 00000100 00001001
y6   00000010 00000100 00000011 00000001        y22  00000011 00000001 00000010 00000100
y7   00000001 00000010 00001001 00001000        y23  00001001 00001000 00000001 00000010
y8   00010000 10000000 11000000 01000000        y24  11000000 01000000 00010000 10000000
y9   10010000 01000000 00100000 01100000        y25  00100000 01100000 10010000 01000000
y10  01000000 00100000 00010000 00110000        y26  00010000 00110000 01000000 00100000
y11  00100000 00010000 10000000 10010000        y27  10000000 10010000 00100000 00010000
y12  00000001 00001000 00001100 00000100        y28  00001100 00000100 00000001 00001000
y13  00001001 00000100 00000010 00000110        y29  00000010 00000110 00001001 00000100
y14  00000100 00000010 00000001 00000011        y30  00000001 00000011 00000100 00000010
y15  00000010 00000001 00001000 00001001        y31  00001000 00001001 00000010 00000001
```

```
g1 = x20 + x12        g20 = g8 + g17        g39 = g5 + g38        g57 = x24 + x10       y0 = g6 + g33
g2 = x21 + x15        g21 = x20 + x14       g40 = x6 + g38        g58 = g11 + g57       y2 = g7 + g48
g3 = x19 + x3         g22 = x31 + g21       g41 = x29 + x7        g59 = x30 + x12       y29 = g40 + g23
g4 = x15 + x6         g23 = x29 + g21       g42 = x21 + g41       g60 = g9 + g59        y14 = g53 + g20
g5 = x28 + x4         g24 = x28 + g21       g43 = x22 + g41       y5 = g19 + g60        y7 = g8 + g24
g6 = x24 + x0         g25 = x16 + x10       g44 = x30 + x4        g61 = g12 + g32       y24 = x1 + g49
g7 = x27 + x18        g26 = x24 + g25       g45 = x13 + g44       y26 = x3 + g61        y17 = g34 + g58
g8 = x23 + x7         g27 = x25 + g25       g46 = x12 + g44       g62 = x15 + g44       y25 = g48 + g27
g9 = x31 + x22        g28 = x27 + g25       g47 = x19 + x9        y31 = g60 + g62       y6 = g9 + g40
g10 = x16 + x8        g29 = x27 + x1        g48 = x2 + g47        y12 = g1 + g42        y19 = g3 + g36
g11 = x11 + x2        g30 = x9 + g29        g49 = g6 + g47        y20 = x13 + g18       y16 = g10 + g30
g12 = x26 + x10       g31 = x19 + g29       g50 = x28 + x14       y30 = g2 + g53        y1 = g34 + g55
g13 = x25 + x3        g32 = x17 + x11       g51 = g4 + g50        y21 = g19 + g51       y23 = g8 + g46
g14 = x29 + g2        g33 = x25 + g32       g52 = x30 + x14       y3 = g3 + g26         y18 = g11 + g15
g15 = x18 + g13       g34 = g29 + g32       g53 = x7 + g52        y22 = g4 + g43        y11 = g58 + g28
g16 = x17 + g10       g35 = x26 + x0        g54 = x26 + x8        y15 = g51 + g22       y27 = g55 + g37
g17 = x31 + x5        g36 = x8 + g35        g55 = g7 + g54        y28 = x5 + g39        y10 = g12 + g31
g18 = g1 + g17        g37 = x11 + g35       g56 = x9 + g35        y4 = g5 + g14         y8 = g13 + g16
g19 = g2 + g17        g38 = x23 + x13       y9 = g15 + g56        y13 = g43 + g45
```

[LS16] circulant (32×32), $D = 3$: 106 XOR2 gates, previous record 110. The matrix of SFX23 (Sec. 4). From stage 4 started from the circuit of SFX23 (Sec. 5.1). Matrix rows $y_0, \ldots, y_{31}$:

```
y0   10000000 10000000 00000001 11100000        y9   00010000 01000000 01000000 10000001
y1   01000000 01000000 10000001 00010000        y10  00001000 00100000 00100000 01000000
y2   00100000 00100000 01000000 00001000        y11  00000100 00010000 00010000 00100000
y3   00010000 00010000 00100000 00000100        y12  10000010 00001000 00001000 00010000
y4   00001000 00001000 00010000 10000010        y13  01000001 00000100 00000100 00001000
y5   00000100 00000100 00001000 01000001        y14  01000000 00000010 00000010 00000101
y6   00000010 00000010 00000101 01000000        y15  11000000 00000001 00000001 00000011
y7   00000001 00000001 00000011 11000000        y16  00000001 11100000 10000000 10000000
y8   11100000 10000000 10000000 00000001        y17  10000001 00010000 01000000 01000000
```

```
y18  01000000 00001000 00100000 00100000
y19  00100000 00000100 00010000 00010000
y20  00010000 10000010 00001000 00001000
y21  00001000 01000001 00000100 00000100
y22  00000101 01000000 00000010 00000010
y23  00000011 11000000 00000001 00000001
y24  10000000 00000001 11100000 10000000
y25  01000000 10000001 00010000 01000000
y26  00100000 01000000 00001000 00100000
y27  00010000 00100000 00000100 00010000
y28  00001000 00010000 10000010 00001000
y29  00000100 00001000 01000001 00000100
y30  00000010 00000101 01000000 00000010
y31  00000001 00000011 11000000 00000001
```

```
g1 = x30 + x24
g2 = x21 + x9
g3 = x19 + x12
g4 = x4 + g3
g5 = x26 + g3
g6 = x24 + x7
g7 = x9 + g6
g8 = x31 + x7
g9 = x6 + g8
g10 = x23 + x8
g11 = x9 + g10
g12 = x16 + x8
g13 = x10 + g12
g14 = x19 + x1
g15 = x18 + g14
g16 = x17 + x15
g17 = x31 + x1
g18 = x17 + x3
g19 = g8 + g18
g20 = x12 + x5
g21 = x24 + x0
g22 = x25 + g6
g23 = x23 + x15
g24 = x16 + g16
g25 = x25 + g10
g26 = x0 + g17
g27 = x2 + g12
y8 = g26 + g27
g28 = g14 + g23
y25 = g25 + g28
g29 = x26 + g21
y0 = g25 + g29
g30 = x14 + x8
g31 = x22 + g23
y7 = g22 + g31
g32 = x22 + x5
g33 = g1 + g32
g34 = x18 + x4
g35 = x25 + g34
g36 = x30 + g23
y15 = g26 + g36
g37 = x20 + x2
g38 = x26 + g37
g39 = x14 + g8
y31 = g24 + g39
g40 = x11 + g20
y11 = g5 + g40
g41 = x29 + x4
g42 = x15 + g41
g43 = x18 + g21
y24 = g24 + g43
g44 = x11 + g41
g45 = x29 + x22
g46 = x3 + g34
y3 = g44 + g46
g47 = x14 + g45
y14 = g17 + g47
g48 = x30 + g16
g49 = x23 + g20
g50 = x13 + x6
y30 = g50 + g48
g51 = x29 + x17
y29 = g51 + g49
g52 = x20 + x13
g53 = x27 + g52
g54 = x19 + g37
y19 = g53 + g54
g55 = x28 + x3
g56 = x10 + g55
g57 = x20 + g30
y20 = g55 + g57
g58 = x28 + x21
g59 = x27 + g58
y27 = g56 + g59
g60 = x2 + g18
y2 = g56 + g60
g61 = x28 + x16
g62 = g45 + g61
y28 = g44 + g62
g63 = x12 + x0
g64 = g50 + g63
y12 = g53 + g64
g65 = x5 + g52
g66 = x31 + x25
y5 = g66 + g65
g67 = x1 + g58
g68 = x13 + x7
y13 = g68 + g67
g69 = x27 + x1
g70 = g12 + g69
y1 = g11 + g70
g71 = x17 + x11
g72 = g21 + g71
y17 = g22 + g72
g73 = x21 + x6
g74 = g30 + g73
y6 = g25 + g74
y9 = g7 + g19
y4 = g1 + g4
y10 = x10 + g35
y26 = x9 + g38
y16 = g7 + g13
y18 = g5 + g15
y21 = g2 + g42
y22 = g7 + g33
y23 = g11 + g9
```

[LW16] (32×32), $D = 3$: 98 XOR2 gates, previous record 101. Matrix rows $y_0, \ldots, y_{31}$:

```
y0   10000000 10000000 01000000 10000010
y1   01000000 01000000 00100000 01000001
y2   00100000 00100000 00010000 10000000
y3   00010000 00010000 00001000 01000000
y4   00001000 00001000 00000100 00100000
y5   00000100 00000100 00000010 00010000
y6   00000010 00000010 00000001 00001000
y7   00000001 00000001 10100000 00000100
y8   10000010 10000000 10000000 01000000
y9   01000001 01000000 01000000 00100000
y10  10000000 00100000 00100000 00010000
y11  01000000 00010000 00010000 00001000
y12  00100000 00001000 00001000 00000100
y13  00010000 00000100 00000100 00000010
y14  00001000 00000010 00000010 00000001
y15  00000100 00000001 00000001 10100000
y16  01000000 10000010 10000000 10000000
y17  00100000 01000001 01000000 01000000
y18  00010000 10000000 00100000 00100000
y19  00001000 01000000 00010000 00010000
y20  00000100 00100000 00001000 00001000
y21  00000010 00010000 00000100 00000100
y22  00000001 00001000 00000010 00000010
y23  10100000 00000100 00000001 00000001
y24  10000000 01000000 10000010 10000000
y25  01000000 00100000 01000001 01000000
y26  00100000 00010000 10000000 00100000
y27  00010000 00001000 01000000 00010000
y28  00001000 00000100 00100000 00001000
y29  00000100 00000010 00010000 00000100
y30  00000010 00000001 00001000 00000010
y31  00000001 10100000 00000100 00000001
```

```
g1 = x20 + x11
g2 = x25 + x3
g3 = x24 + x15
g4 = x9 + x4
g5 = x31 + g4
g6 = x27 + g4
g7 = x29 + x20
g8 = x6 + g7
g9 = x28 + x5
g10 = x29 + g9
g11 = x24 + x0
g12 = x19 + g11
g13 = x28 + x14
g14 = x19 + g13
g15 = x10 + x0
g16 = x2 + g15
g17 = x22 + x9
g18 = x14 + g17
g19 = x29 + x15
g20 = x30 + g19
g21 = x16 + g17
y24 = g11 + g21
g22 = x16 + x8
g23 = x25 + x1
g24 = x14 + x1
g25 = x17 + x9
g26 = x26 + x23
g27 = x5 + g26
g28 = x23 + x10
g29 = g23 + g28
y25 = x17 + g29
g30 = x18 + x4
g31 = x11 + g24
y11 = g14 + g31
g32 = x18 + g15
y10 = x27 + g32
g33 = x21 + g1
y21 = g8 + g33
g34 = g22 + g24
y16 = x24 + g34
g35 = x2 + g7
y12 = x12 + g35
g36 = x23 + x6
y6 = g13 + g36
g37 = x13 + g30
y28 = x28 + g37
g38 = x20 + g9
y20 = x10 + g38
g39 = g23 + g30
y1 = g5 + g39
g40 = x26 + x16
g41 = x2 + g40
y26 = x11 + g41
g42 = x26 + x7
g43 = x18 + g40
g44 = x1 + g42
y9 = g25 + g44
g45 = x3 + g22
y18 = g43 + g45
g46 = g19 + g42
y7 = g43 + g46
g47 = x30 + x12
g48 = x22 + g47
y22 = x7 + g48
g49 = x22 + x13
g50 = x27 + g49
y5 = x5 + g50
g51 = x30 + x17
g52 = x8 + g51
y0 = g11 + g52
g53 = x21 + x12
g54 = x3 + g47
g55 = x27 + g51
y27 = g54 + g55
g56 = x13 + g53
y13 = g54 + g56
g57 = x4 + g53
y4 = x26 + g57
g58 = x25 + x2
g59 = g25 + g58
y17 = x15 + g59
g60 = x31 + x13
g61 = g28 + g60
y23 = g16 + g61
g62 = x31 + x8
g63 = x0 + g22
g64 = x25 + x6
y8 = g64 + g63
g65 = x10 + x7
g66 = x21 + g65
y31 = g62 + g66
y3 = g1 + g2
y29 = g14 + g10
y2 = g16 + g12
y15 = g3 + g27
y30 = g8 + g20
y14 = g5 + g18
y19 = x19 + g6
```

[BKL16] circulant (32×32), $D = 3$: 104 XOR2 gates, previous record 107. Matrix rows $y_0, \ldots, y_{31}$:

```
y0   10000000 10000000 00000001 11100000
y1   01000000 01000000 10000001 00010000
y2   00100000 00100000 01000000 00001000
y3   00010000 00010000 00100000 10000100
y4   00001000 00001000 00010000 01000010
y5   00000100 00000100 00001001 11000001
y6   00000010 00000010 00000101 10000000
y7   00000001 00000001 00000010 11000000
y8   11100000 10000000 10000000 00000001
y9   00010000 01000000 01000000 10000001
y10  00001000 00100000 00100000 01000000
y11  10000100 00010000 00010000 00100000
y12  01000010 00001000 00001000 00010000
y13  11000001 00000100 00000100 00001001
y14  10000000 00000010 00000010 00000101
y15  11000000 00000001 00000001 00000010
y16  00000001 11100000 10000000 10000000
y17  10000001 00010000 01000000 01000000
y18  01000000 00001000 00100000 00100000
y19  00100000 10000100 00010000 00010000
y20  00010000 01000010 00001000 00001000
y21  00001001 11000001 00000100 00000100
y22  00000101 10000000 00000010 00000010
y23  00000010 11000000 00000001 00000001
y24  10000000 00000001 11100000 10000000
y25  01000000 10000001 00010000 01000000
y26  00100000 01000000 00001000 00100000
y27  00010000 00100000 10000100 00010000
y28  00001000 00010000 01000010 00001000
y29  00000100 00001001 11000001 00000100
y30  00000010 00000101 10000000 00000010
y31  00000001 00000010 11000000 00000001
```

```
g1 = x23 + x6
g2 = x20 + x9
g3 = x31 + x9
g4 = x27 + x6
g5 = x30 + x15
g6 = x23 + x1
g7 = x11 + x0
g8 = x28 + x17
g9 = x10 + x2
g10 = x26 + x2
g11 = x0 + g5
g12 = g1 + g3
g13 = x3 + g7
g14 = x24 + x3
g15 = x17 + g14
g16 = x27 + g14
g17 = x27 + x16
g18 = x19 + g17
g19 = x9 + g17
g20 = x21 + x7
g21 = x9 + x8
g22 = g20 + g21
g23 = x16 + x13
g24 = x6 + g23
g25 = x8 + x2
g26 = g23 + g25
g27 = x31 + x13
g28 = x20 + g27
g29 = x19 + x8
g30 = x11 + g29
g31 = x17 + x7
g32 = x25 + g31
g33 = x23 + x5
g34 = x12 + g33
g35 = x25 + x15
g36 = g29 + g35
g37 = x24 + x21
g38 = x14 + g37
g39 = x25 + x4
g40 = x18 + g39
g41 = x12 + x1
g42 = x20 + g41
g43 = x31 + x14
g44 = x29 + x0
g45 = g43 + g44
g46 = x1 + x0
g47 = g27 + g46
y8 = g26 + g47
g48 = x26 + g41
y18 = x18 + g48
g49 = x8 + x5
g50 = x22 + x7
g51 = g49 + g50
g52 = x16 + x10
g53 = g37 + g52
g54 = x25 + x24
g55 = g33 + g54
g56 = x24 + g50
y7 = g35 + g56
g57 = x22 + x4
g58 = g8 + g57
g59 = x29 + x15
g60 = x4 + g59
y21 = g22 + g60
g61 = x24 + x18
g62 = g44 + g61
g63 = x26 + x0
g64 = g49 + g63
g65 = x28 + g20
y13 = g47 + g65
g66 = x28 + x14
g67 = x3 + g66
g68 = x16 + g43
y31 = g31 + g68
g69 = x17 + x16
g70 = g59 + g69
g71 = x30 + x12
g72 = x19 + g39
y4 = g71 + g72
y6 = g1 + g38
y14 = x22 + g45
y5 = g55 + g28
y29 = g70 + g34
y20 = g2 + g67
y9 = g3 + g15
y0 = g64 + g55
y22 = x30 + g51
y12 = g4 + g42
y11 = g64 + g30
y16 = g53 + g22
y24 = g62 + g70
y30 = g5 + g24
y15 = g6 + g11
y17 = g7 + g32
y25 = x1 + g36
y27 = g53 + g16
y2 = g8 + g9
y28 = x11 + g58
y26 = g2 + g10
y23 = x8 + g12
y19 = g26 + g18
y1 = g6 + g19
y3 = g62 + g13
y10 = x10 + g40
```

[SS16] Toeplitz (32×32), $D = 3$: 101 XOR2 gates, previous record 105. From stage 4 started from the circuit of SFX23 (Sec. 5.1). Matrix rows $y_0, \ldots, y_{31}$:

```
y0   10000000 10000000 00000001 11000000
y1   01000000 01000000 10000001 00100000
y2   00100000 00100000 01000000 00010000
y3   00010000 00010000 00100000 00001000
y4   00001000 00001000 00010000 00000100
y5   00000100 00000100 00001000 10000010
y6   00000010 00000010 00000101 10000001
y7   00000001 00000001 00000011 10000000
y8   11100000 10000000 10000000 00000001
y9   00010000 01000000 01000000 10000001
y10  00001000 00100000 00100000 01000000
y11  00000100 00010000 00010000 00100000
y12  10000010 00001000 00001000 00010000
y13  01000001 00000100 00000100 00001000
y14  01000000 00000010 00000010 00000101
y15  11000000 00000001 00000001 00000011
y16  10000000 11100000 10000000 10000000
y17  01000000 00010000 01000000 01000000
y18  00100000 00001000 00100000 00100000
y19  00010000 00000100 00010000 00010000
y20  00001000 10000010 00001000 00001000
y21  00000100 01000001 00000100 00000100
y22  00000010 01000000 00000010 00000010
y23  00000001 11000000 00000001 00000001
y24  11000000 10000000 11100000 10000000
y25  00100000 01000000 00010000 01000000
y26  00010000 00100000 00001000 00100000
y27  00001000 00010000 00000100 00010000
y28  00000100 00001000 10000010 00001000
y29  10000010 00000100 01000001 00000100
y30  10000001 00000010 01000000 00000010
y31  10000000 00000001 11000000 00000001
```

```
g1 = x21 + x9
g2 = x10 + x9
g3 = x16 + x0
g4 = x11 + x4
g5 = x27 + x21
g6 = x28 + x1
g7 = x3 + g2
g8 = x24 + x8
g9 = x9 + g8
g10 = g3 + g8
g11 = x23 + x0
g12 = x1 + g11
g13 = x17 + g11
g14 = x26 + x9
g15 = g3 + g14
g16 = x25 + g14
g17 = x31 + x24
g18 = x13 + x6
g19 = x29 + g18
g20 = x20 + g18
g21 = x23 + x7
g22 = g17 + g21
g23 = x31 + x15
g24 = x30 + g23
g25 = x17 + g23
g26 = x22 + g23
y7 = g22 + g26
g27 = x26 + x19
g28 = x2 + g27
g29 = x21 + x7
g30 = x13 + g29
g31 = x17 + x10
g32 = g17 + g31
g33 = x29 + x15
g34 = x5 + g33
g35 = x5 + g27
y11 = x11 + g35
g36 = x17 + x1
g37 = x2 + x1
g38 = g17 + g37
g39 = x27 + x13
g40 = x25 + g8
y0 = g11 + g40
g41 = x20 + g14
y26 = g7 + g41
```

```
g42 = x19 + x12
g43 = x19 + x3
y19 = g39 + g43
g44 = x30 + x6
g45 = x29 + g42
y4 = x4 + g45
g46 = x18 + g36
y24 = g10 + g46
g47 = x9 + g44
y22 = x22 + g47
g48 = x27 + g31
y2 = x2 + g48
g49 = x18 + x11
g50 = x25 + g49
y17 = g46 + g50
g51 = x10 + g4
y10 = g50 + g51
g52 = x28 + x3
y3 = g49 + g52
g53 = x18 + g42
y18 = g28 + g53
g54 = x24 + x5
g55 = g44 + g54
g56 = x30 + x14
g57 = g21 + g56
y30 = g13 + g57
g58 = x14 + x6
g59 = g29 + g58
y6 = g22 + g59
g60 = x12 + x0
g61 = g39 + g60
y12 = g20 + g61
g62 = x22 + x16
g63 = x14 + x1
g64 = g33 + g63
y14 = g26 + g64
g65 = x5 + g62
g66 = x28 + x12
y28 = g66 + g65
g67 = x28 + x8
g68 = x20 + x14
g69 = x4 + g67
y20 = g68 + g69
y21 = g1 + g34
y15 = g12 + g24
y1 = g12 + g15
y23 = g22 + g9
y16 = g2 + g10
y31 = g3 + g25
y9 = g7 + g32
y29 = g13 + g19
y5 = g20 + g55
y8 = g10 + g38
y27 = g4 + g5
y25 = g28 + g16
y13 = g6 + g30
```

[JPST17] subfield (32×32), $D = 3$: 86 XOR2 gates, previous record 89. Matrix rows $y_0, \ldots, y_{31}$:

```
y0   10000000 10000000 10000000 00010000
y1   01000000 01000000 01000000 10010000
y2   00100000 00100000 00100000 01000000
y3   00010000 00010000 00010000 00100000
y4   00001000 00001000 00001000 00000001
y5   00000100 00000100 00000100 00001001
y6   00000010 00000010 00000010 00000100
y7   00000001 00000001 00000001 00000010
y8   10000000 00010000 11100000 10000000
y9   01000000 10010000 00010000 01000000
y10  00100000 01000000 10000000 00100000
y11  00010000 00100000 11000000 00010000
y12  00001000 00000001 00001110 00001000
y13  00000100 00001001 00000001 00000100
y14  00000010 00000100 00001000 00000010
y15  00000001 00000010 00001100 00000001
y16  00010000 11100000 10000000 10000000
y17  10010000 00010000 01000000 01000000
y18  01000000 10000000 00100000 00100000
y19  00100000 11000000 00010000 00010000
y20  00000001 00001110 00001000 00001000
y21  00001001 00000001 00000100 00000100
y22  00000100 00001000 00000010 00000010
y23  00000010 00001100 00000001 00000001
y24  11100000 10000000 00010000 10000000
y25  00010000 01000000 10010000 01000000
y26  10000000 00100000 01000000 00100000
y27  11000000 00010000 00100000 00010000
y28  00001110 00001000 00000001 00001000
y29  00000001 00000100 00001001 00000100
y30  00001000 00000010 00000100 00000010
y31  00001100 00000001 00000010 00000001
```

```
g1 = x30 + x23
g2 = x21 + x4
g3 = x30 + x14
g4 = x27 + x16
g5 = x28 + x20
g6 = x12 + x5
g7 = x30 + x22
g8 = x27 + x1
g9 = x9 + g8
g10 = x10 + x3
g11 = x17 + g10
g12 = x25 + x10
g13 = x19 + g12
g14 = x2 + g12
g15 = x16 + x9
g16 = g10 + g15
g17 = x8 + x1
g18 = x19 + x2
g19 = g17 + g18
g20 = x24 + x17
g21 = x16 + g20
y1 = g20 + g9
g22 = x22 + x15
g23 = g2 + g22
g24 = x11 + x0
g25 = x18 + g24
g26 = x31 + x21
g27 = x5 + g26
y31 = g23 + g27
g28 = x26 + x3
g29 = x19 + x11
y3 = g28 + g29
g30 = x15 + x7
y7 = g1 + g30
g31 = x20 + x7
g32 = x28 + x13
y5 = g32 + g27
g33 = x14 + g31
y15 = g26 + g33
g34 = x8 + g4
y0 = x0 + g34
g35 = x29 + g30
y21 = g2 + g35
g36 = x23 + x6
g37 = x20 + x13
g38 = g36 + g37
g39 = x26 + g17
y18 = x18 + g39
g40 = x29 + x6
g41 = x22 + x14
y6 = g40 + g41
g42 = x31 + x20
g43 = x12 + g42
g44 = x7 + g40
y29 = g38 + g44
g45 = x25 + g17
y9 = g29 + g45
g46 = x0 + g28
y26 = g11 + g46
g47 = g6 + g36
g48 = g12 + g24
y17 = g11 + g48
g49 = x24 + x0
y24 = g49 + g19
g50 = x26 + g15
y10 = x2 + g50
g51 = x12 + g32
y20 = g33 + g51
g52 = x24 + x8
y16 = g52 + g16
g53 = x28 + x4
y28 = g53 + g47
g54 = x15 + g40
y13 = g47 + g54
y14 = g1 + g38
y30 = g2 + g3
y8 = g25 + g21
y25 = g16 + g13
y4 = x4 + g43
y2 = x18 + g14
y23 = g43 + g38
y11 = g4 + g11
y12 = g5 + g23
y22 = g6 + g7
y27 = g8 + g25
y19 = g9 + g19
```

[SKOP15] involutory (32×32), $D = 3$: 94 XOR2 gates, previous record 98. Matrix rows $y_0, \ldots, y_{31}$:

```
y0   10000000 00100000 11000000 11100000
y1   01000000 00110000 00100000 00010000
y2   00100000 10010000 00010000 10000000
y3   00010000 01000000 10000000 11000000
y4   00001000 00000010 00001100 00001110
y5   00000100 00000011 00000010 00000001
y6   00000010 00001001 00000001 00001000
y7   00000001 00000100 00001000 00001100
y8   00100000 10000000 11100000 11000000
y9   00110000 01000000 00010000 00100000
y10  10010000 00100000 10000000 00010000
y11  01000000 00010000 11000000 10000000
y12  00000010 00001000 00001110 00001100
y13  00000011 00000100 00000001 00000010
y14  00001001 00000010 00001000 00000001
y15  00000100 00000001 00001100 00001000
y16  11000000 11100000 10000000 00100000
y17  00100000 00010000 01000000 00110000
y18  00010000 10000000 00100000 10010000
y19  10000000 11000000 00010000 01000000
y20  00001100 00001110 00001000 00000010
y21  00000010 00000001 00000100 00000011
y22  00000001 00001000 00000010 00001001
y23  00001000 00001100 00000001 00000100
y24  11100000 11000000 00100000 10000000
y25  00010000 00100000 00110000 01000000
y26  10000000 00010000 10010000 00100000
y27  11000000 10000000 01000000 00010000
```

```
y28  00001110 00001100 00000010 00001000        y30  00001000 00000001 00001001 00000010
y29  00000001 00000010 00000011 00000100        y31  00001100 00001000 00000100 00000001
```

```
g1 = x30 + x23      g20 = x16 + x0      g38 = x27 + g20      g52 = x1 + g39       y17 = g22 + g40
g2 = x20 + x4       g21 = x26 + x11     y27 = g34 + g38      y24 = g51 + g52      y3 = g27 + g17
g3 = x24 + x8       g22 = g18 + g21     g39 = x18 + x2       g53 = g41 + g47      y15 = g57 + g4
g4 = x15 + g2       g23 = g20 + g21     g40 = x17 + g39      g54 = x17 + g46      y23 = g29 + g36
g5 = x11 + g3       g24 = x22 + x7      g41 = x10 + x3       y0 = g27 + g54       y7 = g37 + g30
g6 = x30 + x14      g25 = g1 + g24      y10 = g41 + g38      g55 = g35 + g49      y21 = g45 + g7
g7 = x21 + g6       y22 = g24 + g15     g42 = x31 + x14      y4 = g7 + g55        y30 = g1 + g4
g8 = x5 + g6        g26 = x25 + x24     g43 = x7 + g42       g56 = x21 + x5       y31 = g57 + g15
g9 = x29 + g6       g27 = g20 + g26     g44 = x15 + x6       g57 = g49 + g56      y26 = x19 + g23
g10 = x22 + x6      g28 = x13 + x12     g45 = g42 + g44      g58 = g16 + g31      y20 = g29 + g8
g11 = x21 + g10     g29 = g2 + g28      g46 = x26 + x10      y16 = g48 + g58      y8 = g40 + g32
g12 = x13 + g10     g30 = x7 + g28      g47 = x19 + x2       g59 = x25 + g39      y12 = g37 + g11
g13 = x5 + g10      g31 = x16 + x8      y2 = g47 + g5        y25 = g53 + g59      y13 = g12 + g25
g14 = x28 + x12     g32 = g26 + g31     g48 = x1 + g46       g60 = x23 + g44      y29 = g25 + g9
g15 = x31 + g14     g33 = x17 + x1      y1 = g22 + g48       y6 = g14 + g60       y14 = g2 + g43
g16 = x9 + x0       g34 = g31 + g33     g49 = x28 + x4       g61 = x9 + g46       y11 = g5 + g34
g17 = x3 + g16      g35 = x29 + x20     g50 = g28 + g49      y9 = g53 + g61       y18 = g3 + g19
g18 = x27 + x18     g36 = x23 + g35     y28 = g13 + g50      g62 = x19 + g26      y5 = g45 + g13
g19 = x3 + g18      g37 = g14 + g35     g51 = g3 + g16       y19 = g51 + g62
```

[LW16] involutory (32×32), $D = 3$: 87 XOR2 gates, previous record 89. Matrix rows $y_0, \ldots, y_{31}$:

```
y0   10000000 01000000 01000001 00000001        y16  01000001 00000001 10000000 01000000
y1   01000000 00100000 10000000 10100000        y17  10000000 10100000 01000000 00100000
y2   00100000 00010000 01000000 01010000        y18  01000000 01010000 00100000 00010000
y3   00010000 00001000 00100000 00101000        y19  00100000 00101000 00010000 00001000
y4   00001000 00000100 00010000 00010100        y20  00010000 00010100 00001000 00000100
y5   00000100 00000010 00001000 00001010        y21  00001000 00001010 00000100 00000010
y6   00000010 00000001 00000100 00000101        y22  00000100 00000101 00000010 00000001
y7   00000001 10100000 00000010 10100010        y23  00000010 10100010 00000001 10100000
y8   01000000 10000000 00000001 01000001        y24  00000001 01000001 01000000 10000000
y9   00100000 01000000 10100000 10000000        y25  10100000 10000000 00100000 01000000
y10  00010000 00100000 01010000 01000000        y26  01010000 01000000 00010000 00100000
y11  00001000 00010000 00101000 00100000        y27  00101000 00100000 00001000 00010000
y12  00000100 00001000 00010100 00010000        y28  00010100 00010000 00000100 00001000
y13  00000010 00000100 00001010 00001000        y29  00001010 00001000 00000010 00000100
y14  00000001 00000010 00000101 00000100        y30  00000101 00000100 00000001 00000010
y15  10100000 00000001 10100010 00000010        y31  10100010 00000010 10100000 00000001
```

```
g1 = x20 + x4       g19 = x20 + x13     y19 = g29 + g33      g49 = x31 + x22      y14 = g6 + g54
g2 = x24 + x9       g20 = x29 + g19     y27 = g14 + g33      g50 = x15 + g49      y1 = g13 + g16
g3 = x31 + x23      g21 = x28 + g19     g34 = x14 + x6       g51 = x24 + x8       y18 = g7 + g24
g4 = x26 + x10      g22 = x18 + x11     g35 = x23 + g34      g52 = g4 + g51       y15 = g46 + g38
g5 = x25 + x1       g23 = x26 + g22     g36 = x31 + g34      g53 = x14 + x7       y10 = g8 + g32
g6 = x29 + x21      g24 = x27 + g22     g37 = x30 + x22      g54 = x23 + g53      y21 = g9 + g26
g7 = x9 + x1        y2 = g18 + g24      g38 = x15 + g37      g55 = x16 + g5       y31 = g46 + g36
g8 = x25 + x17      g25 = x21 + x12     g39 = x7 + g37       y16 = g47 + g55      y30 = g54 + g17
g9 = x30 + x14      g26 = x4 + g25      g40 = x22 + x6       y3 = g23 + g30       y25 = g44 + g18
g10 = x18 + x2      g27 = x5 + g25      g41 = g6 + g40       y26 = g13 + g32      y7 = g52 + g39
g11 = x27 + x19     g28 = x11 + x3      y13 = g40 + g21      y11 = g1 + g23       y9 = g10 + g16
g12 = x13 + x5      y20 = g28 + g20     g42 = x17 + x0       y29 = g41 + g26      y12 = g11 + g27
g13 = g4 + g7       g29 = x28 + x12     g43 = x9 + g42       y24 = g2 + g48       y6 = g41 + g50
g14 = g1 + g11      g30 = g28 + g29     g44 = x8 + g42       y4 = g20 + g14       y5 = g17 + g21
g15 = x8 + g3       y28 = g27 + g30     g45 = x16 + x0       y23 = g52 + g35      y22 = g12 + g50
g16 = x16 + g2      g31 = x19 + x10     g46 = g10 + g45      y0 = g3 + g43
g17 = g9 + g12      g32 = x3 + g31      g47 = x15 + x7       y17 = g4 + g44
g18 = g8 + g10      g33 = x2 + g31      g48 = x17 + g47      y8 = g5 + g15
```

[SS16] involutory (32×32), $D = 4$: 94 XOR2 gates, previous record 98. Matrix rows $y_0, \ldots, y_{31}$:

```
y0   10000000 00000001 10000000 10000010
y1   01000000 10000000 01000000 01000001
y2   00100000 01000001 00100000 10100010
y3   00010000 00100000 00010000 01010001
y4   00001000 00010000 00001000 00101000
y5   00000100 00001001 00000100 00010110
y6   00000010 00000101 00000010 00001001
y7   00000001 00000010 00000001 00000100
y8   00000001 10000000 10000010 10000000
y9   10000000 01000000 01000001 01000000
y10  01000001 00100000 10100010 00100000
y11  00100000 00010000 01010001 00010000
y12  00010000 00001000 00101000 00001000
y13  00001001 00000100 00010110 00000100
y14  00000101 00000010 00001001 00000010
y15  00000010 00000001 00000100 00000001
```

```
y16  10100000 00100000 10000000 00000001
y17  01010000 00010000 01000000 10000000
y18  00001000 00101000 00100000 01000001
y19  10000100 10010100 00010000 00100000
y20  11000010 11001010 00001000 00010000
y21  01000001 01000101 00000100 00001001
y22  10000000 10000010 00000010 00000101
y23  01000000 01000001 00000001 00000010
y24  00100000 10100000 00000001 10000000
y25  00010000 01010000 10000000 01000000
y26  00101000 00001000 01000001 00100000
y27  10010100 10000100 00100000 00010000
y28  11001010 11000010 00010000 00001000
y29  01000101 01000001 00001001 00000100
y30  10000010 10000000 00000101 00000010
y31  01000001 01000000 00000010 00000001
```

```
g1 = x24 + x8
g2 = x18 + x3
g3 = x22 + x7
g4 = x9 + x1
g5 = x30 + x15
g6 = x16 + x9
g7 = x31 + x10
g8 = x16 + g1
g9 = x31 + g3
g10 = x23 + g4
y23 = g5 + g10
y31 = g4 + g9
g11 = x8 + x0
g12 = g5 + g11
y0 = g8 + g12
g13 = x21 + x15
g14 = x6 + g13
y15 = x31 + g14
g15 = x25 + x3
```

```
g16 = x11 + g15
g17 = x24 + g16
g18 = x17 + x1
g19 = x25 + g18
g20 = x29 + x7
g21 = x23 + g20
y7 = x14 + g21
g22 = x28 + x4
g23 = x20 + g22
g24 = x12 + x4
g25 = g2 + g24
g26 = x23 + x2
g27 = x10 + g26
g28 = x26 + x17
g29 = g26 + g28
y26 = g24 + g29
g30 = x14 + g11
g31 = g20 + g30
g32 = x6 + g4
```

```
g33 = g30 + g32
g34 = x30 + x5
g35 = x20 + g34
g36 = g20 + g35
g37 = x29 + g35
g38 = x27 + x12
g39 = x20 + g38
g40 = x22 + x13
g41 = x28 + g40
g42 = x21 + g41
g43 = g7 + g15
y18 = g25 + g43
g44 = x27 + x19
y3 = g44 + g43
g45 = x26 + x11
g46 = g44 + g45
y4 = g45 + g23
g47 = x2 + x0
g48 = x16 + g47
```

```
g49 = x26 + x18
g50 = g6 + g49
g51 = g47 + g50
g52 = g7 + g50
g53 = x19 + g22
y28 = g53 + g33
g54 = g20 + g53
y13 = g42 + g54
g55 = g13 + g41
y21 = y31 + g55
g56 = x0 + g19
y9 = g10 + g56
g57 = x8 + g19
y1 = x31 + g57
g58 = g12 + g14
y30 = x23 + g58
g59 = g13 + g39
y5 = g37 + g59
g60 = x27 + x5
```

```
g61 = x13 + g11
g62 = g60 + g61
y24 = g1 + g27
y27 = g2 + g62
y19 = g46 + g62
y2 = g51 + y0
y12 = g25 + g23
y8 = g3 + g8
y6 = g42 + y15
y29 = g36 + y23
y17 = g19 + g17
y11 = g46 + g29
y10 = y31 + g52
y14 = g37 + y7
y20 = g33 + g39
y22 = g9 + g31
y25 = g6 + g16
y16 = g7 + g48
```

[KLSW17] (32×32), $D = 3$: 80 XOR2 gates, previous record 84. The previous record comes from the circuit of SFX23 for [LSL+19] HW160 #93 (Sec. 4). Matrix rows $y_0, \ldots, y_{31}$:

```
y0   10000000 00010000 00100000 10000000
y1   01000000 10010000 00110000 01000000
y2   00100000 01000000 10010000 00100000
y3   00010000 00100000 01000000 00010000
y4   00001000 00000001 00000010 00001000
y5   00000100 00001001 00000011 00000100
y6   00000010 00000100 00001001 00000010
y7   00000001 00000010 00000100 00000001
y8   00010000 10000000 10000000 00100000
y9   10010000 01000000 01000000 00110000
y10  01000000 00100000 00100000 10010000
y11  00100000 00010000 00010000 01000000
y12  00000001 00001000 00001000 00000010
y13  00001001 00000100 00000100 00000011
y14  00000100 00000010 00000010 00001001
y15  00000010 00000001 00000001 00000100
```

```
y16  01000000 00010000 10000000 00010000
y17  01100000 10010000 01000000 10010000
y18  00110000 01000000 00100000 01000000
y19  10010000 00100000 00010000 00100000
y20  00000100 00000001 00001000 00000001
y21  00000110 00001001 00000100 00001001
y22  00000011 00000100 00000010 00000100
y23  00001001 00000010 00000001 00000010
y24  00010000 01000000 00010000 10000000
y25  10010000 01100000 10010000 01000000
y26  01000000 00110000 01000000 00100000
y27  00100000 10010000 00100000 00010000
y28  00000001 00000100 00000001 00001000
y29  00001001 00000110 00001001 00000100
y30  00000100 00000011 00000100 00000010
y31  00000010 00001001 00000010 00000001
```

```
g1 = x30 + x7
g2 = x8 + x2
g3 = x18 + x11
g4 = x19 + x0
g5 = x20 + g1
g6 = x27 + g2
y12 = x12 + g5
g7 = x15 + x5
g8 = x12 + g7
g9 = x27 + x17
g10 = x24 + g9
g11 = x26 + g9
g12 = x9 + x3
g13 = x0 + g12
```

```
g14 = x26 + x3
g15 = x10 + g14
y3 = g11 + g15
g16 = x16 + x2
g17 = g14 + g16
g18 = x19 + g12
y2 = g17 + g18
y24 = x24 + g18
g19 = x11 + x1
g20 = g2 + g19
g21 = x29 + x23
g22 = x22 + g21
g23 = x18 + x2
g24 = x25 + x19
```

```
g25 = g23 + g24
y18 = g18 + g25
y11 = g3 + g25
g26 = x30 + x14
g27 = x31 + x21
g28 = g26 + g27
g29 = x31 + g7
y30 = g28 + g29
y20 = x20 + g29
g30 = x24 + g3
y0 = x0 + g30
g31 = x22 + x15
g32 = x6 + g31
y15 = g22 + g32
```

```
g33 = x13 + x7
g34 = x27 + x10
g35 = g19 + g34
y10 = g30 + g35
y26 = g11 + g35
g36 = x16 + x10
y16 = g36 + g35
g37 = x14 + x4
g38 = g33 + g37
y13 = g28 + g38
g39 = x28 + x14
g40 = g31 + g39
y14 = g29 + g40
y4 = g37 + g40
```

```
g41 = x23 + x6
g42 = g33 + g41
g43 = x20 + g21
y29 = g38 + g43
g44 = x28 + x6
y28 = g44 + g42
g45 = g27 + g44
y21 = g8 + g45
g46 = x23 + g1
y23 = g37 + g46
g47 = x31 + x12
y31 = g47 + g32
g48 = g24 + g36
y25 = g13 + g48
```

```
y1 = g25 + g20      y17 = g20 + g10     y9 = g11 + g13      y27 = g3 + g6       y5 = g22 + g8
y7 = g1 + g28       y6 = g42 + g5       y8 = g2 + g17       y19 = g4 + g15      y22 = g22 + g42
```

[LSL+19] HW168 #130 (32×32), $D = 3$: 82 XOR2 gates, previous record 84. Matrix rows $y_0, \ldots, y_{31}$:

```
y0   10000000 00001000 00001000 00001010      y16  10000000 00100000 10000000 00001000
y1   01000000 00000100 00000100 00000101      y17  01000000 00010000 01000000 00000100
y2   00100000 00001010 00001010 10001000      y18  00100000 00101000 00100000 00001010
y3   00010000 00000101 00000101 01000100      y19  00010000 00010100 00010000 00000101
y4   00001000 10000010 10000010 10100010      y20  00001000 00001010 00001000 10000010
y5   00000100 01000001 01000001 01010001      y21  00000100 00000101 00000100 01000001
y6   00000010 00100000 00100000 00101000      y22  00000010 10000010 00000010 00100000
y7   00000001 00010000 00010000 00010100      y23  00000001 01000001 00000001 00010000
y8   10100000 10000000 00001000 00001000      y24  10000000 10000000 10100000 10000000
y9   01010000 01000000 00000100 00000100      y25  01000000 01000000 01010000 01000000
y10  00001000 00100000 00001010 00001010      y26  00100000 00100000 00001000 00100000
y11  00000100 00010000 00000101 00000101      y27  00010000 00010000 00000100 00010000
y12  00000010 00001000 10000010 10000010      y28  00001000 00001000 00000010 00001000
y13  00000001 00000100 01000001 01000001      y29  00000100 00000100 00000001 00000100
y14  10000000 00000010 00100000 00100000      y30  00000010 00000010 10000000 00000010
y15  01000000 00000001 00010000 00010000      y31  00000001 00000001 01000000 00000001
```

```
g1 = x25 + x17      g16 = x18 + g10     g30 = g27 + g29     y6 = g45 + g15      y16 = g3 + g4
g2 = x29 + x11      g17 = x28 + x20     y20 = g25 + g30     g46 = x19 + g2      y13 = g38 + y31
g3 = x28 + x10      g18 = x2 + g17      g31 = x27 + x9      g47 = x3 + g39      y4 = g28 + g23
g4 = x16 + x0       g19 = x30 + x12     g32 = x15 + g31     y19 = g46 + g47     y5 = g34 + g42
g5 = x31 + x15      g20 = g17 + g19     g33 = x23 + x15     g48 = x27 + x7      y17 = x1 + g13
g6 = x17 + x7       g21 = x2 + g19      g34 = g31 + g33     y7 = g48 + g46      y15 = g12 + g32
g7 = x30 + x14      y0 = x0 + g20       g35 = x29 + x21     g49 = x26 + g3      y2 = g18 + g30
g8 = x16 + x6       g22 = x30 + x4      g36 = x3 + g35      y26 = g18 + g49     y9 = g9 + g36
g9 = x9 + x1        g23 = g11 + g22     g37 = x25 + x13     g50 = x14 + g26     y27 = g14 + g36
g10 = x8 + x0       g24 = x22 + x20     g38 = g33 + g37     y14 = g16 + g50     y29 = g44 + g40
g11 = x24 + x16     g25 = g22 + g24     g39 = x31 + x13     y21 = g44 + g38     y18 = g15 + g21
g12 = x19 + g9      y10 = g3 + g25      g40 = g35 + g39     y23 = x7 + g34      y1 = x1 + g40
y30 = g7 + g8       g26 = x26 + x8      g41 = x31 + x5      y28 = g25 + g20     y8 = g10 + g18
y31 = g5 + g6       g27 = x22 + x14     g42 = g1 + g41      y3 = g36 + g38      y24 = g11 + g16
g13 = x17 + g2      g28 = g26 + g27     g43 = x23 + x21     y25 = g1 + g12
g14 = x27 + g2      y22 = x6 + g28      g44 = g41 + g43     y11 = g2 + g44
g15 = x18 + g3      g29 = x24 + x12     g45 = x26 + x6      y12 = g30 + y30
```

A.2 Circuits of Our UMC Areas

SmallScale AES (16×16), $D = 3$: [13 XOR2 + 17 XOR3] gates, 118.39 GE, from 47 XOR2 gates; baseline 127.70 GE (HILL). The circuit of Appendix A.1, with 45 XOR2 gates, gives [35 XOR2 + 5 XOR3] gates, i.e., 128.35 GE. Matrix: as in Appendix A.1.

```
g1 = x15 + x12              g9 = g3 + (x13 + x1)        y0 = g8 + x3            y11 = g2 + (x7 + g4)
g2 = x15 + x3               g10 = g2 + (x11 + x6)       y7 = x10 + g10          y12 = g11 + (g2 + g3)
g3 = x11 + x8               y9 = g9 + (x5 + g1)         y3 = g10 + g12          y15 = g12 + (x14 + x11)
g4 = x14 + x10              g11 = x4 + (x11 + x0)       y4 = g8 + g11           y14 = g13 + g6
g5 = x7 + x4                g12 = x2 + (x7 + x3)        y2 = g13 + x5           y13 = g14 + (x1 + g1)
g6 = x2 + (x14 + x13)       g13 = g4 + (x6 + x1)        y8 = g11 + g1           y5 = g9 + (x9 + g5)
g7 = x9 + x5                y10 = g6 + (x9 + x6)        y1 = g14 + (x13 + g5)
g8 = g5 + (x12 + x8)        g14 = g7 + (x3 + x0)        y6 = x2 + (g4 + g7)
```

Joltik (16×16), $D = 3$: [16 XOR2 + 16 XOR3] gates, 122.72 GE, from 48 XOR2 gates; baseline 125.38 GE (HILL). The same area for [SKOP15] and [LW16] involutory 16×16 (Table 3). From stage 4 with the UMC area as the objective (Sec. 5.1). The circuit of Appendix A.1, with 46 XOR2 gates, gives [34 XOR2 + 6 XOR3] gates, i.e., 130.02 GE. Matrix: as in Appendix A.1.

```
g1 = x15 + x6
g2 = x7 + x2
g3 = x5 + x0
g4 = x13 + x12
g5 = g3 + g4
g6 = g4 + (x9 + x8)
g7 = x11 + (x12 + x4)
y11 = g5 + g7
g8 = x10 + x2
g9 = g3 + (x4 + x1)
g10 = x3 + (x8 + x0)
g11 = x15 + x9
g12 = x14 + x6
g13 = x11 + x3
g14 = x7 + x1
y8 = g9 + (x8 + g12)
y1 = g1 + (x10 + g14)
y7 = g6 + (x13 + g14)
g15 = g13 + (x14 + x7)
y14 = g10 + g15
y15 = g9 + (x5 + g11)
g16 = g13 + (x15 + x10)
y0 = g6 + (x0 + g12)
y5 = g15 + (x5 + g2)
y3 = g10 + g5
y12 = g9 + (x12 + g8)
y10 = g7 + g16
y13 = g16 + (x13 + g1)
y6 = g1 + g10
y4 = g6 + (x4 + g8)
y9 = g2 + (x14 + g11)
y2 = g2 + g7
```

[SKOP15] Hadamard (16×16), $D = 3$: [19 XOR2 + 14 XOR3] gates, 122.38 GE, from 47 XOR2 gates; baseline 125.04 GE (SFX23, HILL). It is our circuit with the fewest XOR2 gates, merged. Matrix rows $y_0, \ldots, y_{15}$:

```
y0   1000 0001 0100 1100
y1   0100 1001 0110 0010
y2   0010 0100 0011 0001
y3   0001 0010 1001 1000
y4   0001 1000 1100 0100
y5   1001 0100 0010 0110
y6   0100 0010 0001 0011
y7   0010 0001 1000 1001
y8   0100 1100 1000 0001
y9   0110 0010 0100 1001
y10  0011 0001 0010 0100
y11  1001 1000 0001 0010
y12  1100 0100 0001 1000
y13  0010 0110 1001 0100
y14  0001 0011 0100 0010
y15  1000 1001 0010 0001
```

```
g1 = x15 + x10
g2 = x9 + x7
g3 = x7 + x2
g4 = x14 + x0
g5 = x8 + x6
g6 = x11 + x5
g7 = x2 + g6
g8 = x15 + x1
g9 = g2 + g8
g10 = x13 + x3
g11 = x10 + g10
g12 = x14 + x6
g13 = x12 + x0
g14 = g1 + (x14 + x4)
y12 = g6 + (x1 + g13)
y6 = g8 + (x11 + g12)
g15 = x8 + x4
y4 = g15 + (x9 + g10)
g16 = x11 + x3
y11 = x4 + (g4 + g16)
g17 = g3 + (x12 + x6)
y3 = g16 + (x12 + g5)
y8 = g8 + (x5 + g15)
y5 = g11 + (x5 + g4)
y15 = g14 + (x7 + g4)
y13 = g7 + (x13 + g5)
y2 = g1 + g7
y14 = g2 + (x3 + g12)
y7 = g17 + (x15 + g5)
y10 = g3 + g11
y9 = g9 + g17
y1 = g9 + g14
y0 = g2 + (x13 + g13)
```

[SS16] Toeplitz (16×16), $D = 3$: [20 XOR2 + 11 XOR3] gates, 111.37 GE, from 42 XOR2 gates; baseline 114.03 GE (SFX23, HILL). It is our circuit with the fewest XOR2 gates, merged. Matrix rows $y_0, \ldots, y_{15}$:

```
y0   1000 1000 0001 0100
y1   0100 0100 1000 0010
y2   0010 0010 0100 1001
y3   0001 0001 0011 1000
y4   0010 1000 1000 0001
y5   1001 0100 0100 1000
y6   1100 0010 0010 0100
y7   0100 0001 0001 0011
y8   1000 0010 1000 1000
y9   0100 1001 0100 0100
y10  0010 1100 0010 0010
y11  0001 0100 0001 0001
y12  0100 1000 0010 1000
y13  0010 0100 1001 0100
y14  1001 0010 1100 0010
y15  1000 0001 0100 0001
```

```
g1 = x13 + x4
g2 = x11 + g1
y0 = x0 + g2
g3 = x12 + (x6 + x0)
y8 = x8 + g3
g4 = x9 + x0
g5 = x3 + g4
g6 = x15 + x2
g7 = x8 + x4
y4 = g6 + g7
g8 = g7 + (x5 + x2)
g9 = x14 + x8
g10 = x12 + x10
g11 = g10 + (x13 + x1)
y6 = g3 + g11
y10 = g8 + (x10 + g9)
g12 = x15 + x7
y15 = g4 + g12
y5 = g5 + (x12 + x5)
y1 = x5 + (x1 + g9)
g13 = x15 + x11
g14 = x3 + g13
y11 = x5 + g14
g15 = x7 + x1
y7 = x14 + (g13 + g15)
y9 = g1 + (x9 + g15)
y3 = g14 + (g10 + g12)
y12 = g1 + g11
y2 = g3 + (g4 + g6)
y13 = g8 + g2
y14 = g5 + (x6 + g9)
```

[JPST17] (16×16), $D = 3$: [17 XOR2 + 13 XOR3] gates, 111.71 GE, from 43 XOR2 gates; baseline 113.04 GE (SFX23, HILL). It is our circuit with the fewest XOR2 gates, merged. Matrix rows $y_0, \ldots, y_{15}$:

```
y0   1000 1000 1000 0001   y4   1000 0001 1110 1000   y8   0001 1110 1000 1000   y12  1110 1000 0001 1000
y1   0100 0100 0100 1001   y5   0100 1001 0001 0100   y9   1001 0001 0100 0100   y13  0001 0100 1001 0100
y2   0010 0010 0010 0100   y6   0010 0100 1000 0010   y10  0100 1000 0010 0010   y14  1000 0010 0100 0010
y3   0001 0001 0001 0010   y7   0001 0010 1100 0001   y11  0010 1100 0001 0001   y15  1100 0001 0010 0001
```

```
g1 = x14 + x3              y7 = g4 + g8               y10 = x10 + (x14 + g11)    y4 = g14 + (x8 + g3)
g2 = x11 + x7              y14 = g1 + g8              y9 = g8 + (x7 + g13)       y2 = x2 + (x10 + g13)
g3 = x12 + x9              g9 = x15 + x1              y5 = x13 + (g2 + g11)      y0 = x4 + g4
g4 = x15 + (x8 + x0)       g10 = x5 + g9              y6 = x2 + (x14 + g5)       y3 = g1 + g2
g5 = x8 + x5               g11 = x4 + x1              y13 = g7 + (x11 + g13)     y8 = g7 + (x12 + x4)
g6 = x6 + x3               g12 = g11 + (x11 + x2)     g14 = x0 + (x10 + x7)      y1 = g3 + g10
g7 = g5 + g6               y11 = g10 + g12            y15 = g9 + g14
g8 = g6 + (x9 + x0)        g13 = x13 + x6             y12 = g12 + (x12 + x0)
```

[SS16] involutory (16×16), $D = 3$: [13 XOR2 + 15 XOR3] gates, 109.05 GE, from 43 XOR2 gates; baseline 110.04 GE (SFX23, HILL). The circuit of Appendix A.1, with 40 XOR2 gates, gives [26 XOR2 + 7 XOR3] gates, i.e., 110.69 GE. Matrix: as in Appendix A.1.

```
g1 = x9 + x6               y6 = g4 + (x6 + g6)        y7 = g6 + g9               y5 = y3 + g11
g2 = x7 + x1               y0 = x0 + (x12 + g6)       y4 = x3 + (x4 + g8)        y15 = x15 + (g5 + g6)
g3 = x5 + (x11 + x3)       y8 = x8 + (x15 + g2)       y1 = g9 + (g2 + g5)        y11 = g11 + g3
y12 = x12 + g3             y3 = g1 + (x15 + x3)       g10 = x6 + x0              y10 = g3 + g9
g4 = g2 + (x15 + x12)      g7 = x11 + x2              g11 = g10 + (x14 + x5)     y14 = g1 + (x14 + g2)
g5 = x4 + x2               g8 = x14 + x8              g12 = g10 + (x13 + x8)     y9 = g4 + (x9 + g5)
g6 = x10 + x7              g9 = g7 + (x13 + x10)      y13 = g3 + g12             y2 = x5 + (g7 + g8)
```

[JPST17] involutory (16×16), $D = 3$: [17 XOR2 + 15 XOR3] gates, 121.05 GE, from 47 XOR2 gates; baseline 123.37 GE (SFX23, HILL). Our circuit with the fewest XOR2 gates, 46, gives [22 XOR2 + 12 XOR3] gates, i.e., 122.04 GE. Matrix rows $y_0, \ldots, y_{15}$:

```
y0   0001 1000 1000 1100   y4   1000 0010 1111 1000   y8   1110 1100 0010 1000   y12  1000 1110 1000 0001
y1   1001 0100 0100 0010   y5   0100 0011 1000 0100   y9   0001 0010 0011 0100   y13  0100 0001 0100 1001
y2   0100 0010 0010 0001   y6   0010 1001 1100 0010   y10  1000 0001 1001 0010   y14  0010 1000 0010 0100
y3   0010 0001 0001 1000   y7   0001 0100 1110 0001   y11  1100 1000 0100 0001   y15  0001 1100 0001 0010
```

```
g1 = x8 + x7               g8 = x15 + (x8 + x5)       g14 = g4 + (x5 + x3)       y1 = g14 + g15
g2 = x12 + x7              g9 = x10 + x2              y7 = g8 + (x9 + g3)        y5 = x1 + (g1 + g12)
g3 = x10 + x3              g10 = x13 + x4             g15 = x14 + (x9 + x4)      y13 = g5 + g2
g4 = x4 + x0               y14 = g9 + g10             g16 = x6 + (x10 + x9)      y6 = g15 + (x2 + g1)
g5 = x1 + (x15 + x9)       g11 = x12 + x8             y2 = g5 + g16              y0 = x3 + (g10 + g11)
y11 = g4 + g5              g12 = x13 + x6             y4 = g16 + (g6 + g11)      y12 = g8 + (x6 + g4)
g6 = x11 + x0              g13 = x5 + (x12 + x1)      y10 = g7 + g1              y9 = g3 + (x11 + g12)
g7 = x14 + g6              y3 = x11 + (x2 + g2)       y15 = g7 + g14             y8 = g13 + (g4 + g9)
```

AES MixColumns (32×32), $D = 3$: [38 XOR2 + 31 XOR3] gates, 258.77 GE, from 100 XOR2 gates; baseline 264.09 GE (SFX23). Column $8i + k$ (row $8i + k$) is bit k of input (output) byte i, bit 0 being the least significant one. From stage 4 with the UMC area as the objective (Sec. 5.1). Our circuit with the fewest XOR2 gates, 97, gives [59 XOR2 + 19 XOR3] gates, i.e., 265.73 GE. Matrix rows $y_0, \ldots, y_{31}$:

```
y0   00000001 10000001 10000000 10000000   y10  00100000 01000000 01100000 00100000
y1   10000001 11000001 01000000 01000000   y11  00010000 00100001 00110001 00010000
y2   01000000 01100000 00100000 00100000   y12  00001000 00010001 00011001 00001000
y3   00100001 00110001 00010000 00010000   y13  00000100 00001000 00001100 00000100
y4   00010001 00011001 00001000 00001000   y14  00000010 00000100 00000110 00000010
y5   00001000 00001100 00000100 00000100   y15  00000001 00000010 00000011 00000001
y6   00000100 00000110 00000010 00000010   y16  10000000 10000000 00000001 10000001
y7   00000010 00000011 00000001 00000001   y17  01000000 01000000 10000001 11000001
y8   10000000 00000001 10000001 10000000   y18  00100000 00100000 01000000 01100000
y9   01000000 10000001 11000001 01000000   y19  00010000 00010000 00100001 00110001
```

```
y20  00001000 00001000 00010001 00011001
y21  00000100 00000100 00001000 00001100
y22  00000010 00000010 00000100 00000110
y23  00000001 00000001 00000010 00000011
y24  10000001 10000000 10000000 00000001
y25  11000001 01000000 01000000 10000001
y26  01100000 00100000 00100000 01000000
y27  00110001 00010000 00010000 00100001
y28  00011001 00001000 00001000 00010001
y29  00001100 00000100 00000100 00001000
y30  00000110 00000010 00000010 00000100
y31  00000011 00000001 00000001 00000010
```

```
g1 = x24 + x16
g2 = x29 + x5
g3 = x23 + x15
g4 = x31 + x7
g5 = x22 + x14
g6 = x10 + (x27 + x19)
g7 = x15 + x7
g8 = x8 + g7
g9 = x31 + x23
g10 = x30 + x22
g11 = g10 + (x21 + x6)
y14 = x13 + g11
g12 = g7 + (x11 + x2)
g13 = x28 + x20
g14 = x12 + x3
g15 = x25 + x17
g16 = x9 + x1
g17 = x13 + x5
g18 = g17 + (x21 + x4)
g19 = x12 + g2
y5 = g18 + g19
g20 = x16 + x8
g21 = x0 + g20
y24 = g4 + g21
g22 = g1 + g9
y16 = g21 + g22
y6 = x14 + (g10 + g17)
g23 = x9 + x0
g24 = x6 + g3
y7 = g24 + (x31 + x14)
g25 = x11 + g13
y23 = x31 + (g7 + g10)
g26 = g15 + (x10 + x2)
y18 = x26 + g26
g27 = x17 + x1
g28 = g16 + (x26 + x18)
y2 = x10 + g28
y15 = g5 + (x23 + g4)
y17 = g22 + (x25 + g16)
y21 = x13 + (g2 + g13)
y4 = g25 + (g7 + g14)
y10 = g28 + (x2 + g27)
y31 = g24 + (x30 + x7)
y13 = g19 + (x21 + x20)
g29 = x4 + (x27 + x20)
g30 = x19 + x4
g31 = x27 + g9
y12 = g25 + (g3 + g30)
g32 = g4 + g27
y25 = g32 + (x24 + g23)
y26 = g26 + (x18 + g27)
g33 = x18 + x3
y11 = g6 + (g3 + g33)
g34 = g3 + g15
y9 = g34 + (x1 + g20)
g35 = x26 + x11
g36 = g30 + (x28 + x12)
y20 = g31 + g36
y19 = g31 + (g33 + g35)
g37 = g4 + (x3 + x2)
y27 = g37 + (x19 + g35)
y28 = g29 + (g4 + g14)
y0 = g1 + g8
y22 = g11 + (x29 + g5)
y30 = g2 + (x6 + g5)
y3 = g6 + g12
y8 = g1 + (x0 + g3)
y1 = g8 + (g15 + g23)
y29 = x28 + g18
```

[SKOP15] Hadamard (32×32), $D = 3$: [38 XOR2 + 28 XOR3] gates, 244.76 GE, from 94 XOR2 gates; baseline 245.44 GE (SFX23, HILL). From stage 4 started from the circuit of SFX23 (Sec. 5.1). It is the circuit of Appendix A.1, merged. Matrix: as in Appendix A.1.

```
g1 = x20 + x12
g2 = x21 + x15
g3 = x19 + x3
g4 = x15 + x6
g5 = x28 + x4
g6 = x24 + x0
g7 = x27 + x18
g8 = x23 + x7
g9 = x31 + x22
g10 = x16 + x8
g11 = x11 + x2
g12 = x26 + x10
g13 = x25 + x3
g14 = x18 + g13
g15 = x31 + x5
g16 = g2 + g15
g17 = x20 + x14
g18 = x16 + x10
g19 = x27 + x1
g20 = x17 + x11
g21 = g19 + g20
g22 = x26 + x0
g23 = x23 + x13
g24 = x6 + g23
g25 = x29 + x7
g26 = x22 + g25
g27 = x30 + x4
g28 = x19 + x9
g29 = x2 + g28
g30 = g4 + (x28 + x14)
g31 = x7 + (x30 + x14)
g32 = g7 + (x26 + x8)
y9 = g14 + (x9 + g22)
g33 = g11 + (x24 + x10)
g34 = g9 + (x30 + x12)
y5 = g16 + g34
y26 = x3 + (g12 + g20)
y31 = g34 + (x15 + g27)
y12 = g1 + (x21 + g25)
y20 = x13 + (g1 + g15)
y30 = g2 + g31
y21 = g16 + g30
y3 = g3 + (x24 + g18)
y22 = g4 + g26
y15 = g30 + (x31 + g17)
y28 = x5 + (g5 + g23)
y4 = g5 + (x29 + g2)
y13 = g26 + (x13 + g27)
y0 = g6 + (x25 + g20)
y2 = g7 + g29
y29 = g24 + (x29 + g17)
y14 = g31 + (g8 + g15)
y7 = g8 + (x28 + g17)
y24 = x1 + (g6 + g28)
y17 = g21 + g33
y25 = g29 + (x25 + g18)
y6 = g9 + g24
y19 = g3 + (x8 + g22)
y16 = g10 + (x9 + g19)
y1 = g21 + g32
y23 = g8 + (x12 + g27)
y18 = g11 + g14
y11 = g33 + (x27 + g18)
y27 = g32 + (x11 + g22)
y10 = g12 + (x19 + g19)
y8 = g13 + (x17 + g10)
```

[LS16] circulant (32×32), $D = 3$: [34 XOR2 + 37 XOR3] gates, 274.79 GE, from 108 XOR2 gates; baseline 283.45 GE (SFX23). The circuit of Appendix A.1, with 106 XOR2 gates, gives [42 XOR2 + 32 XOR3] gates, i.e., 275.44 GE. Matrix: as in Appendix A.1.

```
g1 = x31 + x17
g2 = x31 + x23
g3 = x5 + g2
g4 = x27 + x9
g5 = x9 + x3
g6 = x21 + g5
g7 = x24 + x16
g8 = x25 + x23
g9 = x24 + g8
g10 = x20 + x13
g11 = x15 + x0
g12 = x17 + g11
g13 = x1 + g11
g14 = x8 + x0
g15 = x15 + x7
g16 = g8 + (x21 + x14)
g17 = x29 + x3
g18 = x9 + x7
g19 = x8 + g18
g20 = x27 + x20
g21 = x2 + g20
g22 = g14 + (x25 + x19)
g23 = x6 + x0
g24 = x5 + (x26 + x11)
g25 = x16 + (x31 + x1)
g26 = g23 + (x30 + x13)
g27 = x7 + (x21 + x13)
y12 = g20 + (x12 + g23)
g28 = x14 + (x28 + x20)
g29 = g15 + (x25 + x11)
g30 = x4 + (x19 + x12)
y8 = g25 + (x2 + g14)
y1 = g25 + (g2 + g4)
g31 = g1 + (x29 + x12)
g32 = x10 + (x18 + x4)
y10 = x25 + g32
g33 = x16 + x14
y31 = g33 + (g1 + g15)
g34 = g17 + (x15 + x4)
y21 = g6 + g34
g35 = x10 + (x17 + x2)
y2 = x28 + g35
g36 = x26 + (x18 + x1)
y18 = x12 + g36
g37 = g33 + (x29 + x22)
y14 = g25 + g37
g38 = x22 + x5
y22 = g38 + (x30 + g18)
g39 = x22 + (x28 + x11)
y28 = g39 + (x16 + x4)
y26 = g21 + (x26 + g4)
y17 = g12 + g29
y11 = x19 + g24
y20 = g28 + (x8 + x3)
y0 = g9 + (x26 + g14)
y4 = g30 + (x30 + x24)
```

```
y30 = g12 + g26
y29 = g3 + g31
y16 = g19 + (x10 + g7)
y5 = g3 + (g8 + g10)
y15 = g13 + (x30 + g2)
y23 = g19 + (x6 + g2)
y24 = g12 + (x18 + g7)
y3 = x11 + (x18 + g17)
y25 = g13 + g22
y6 = x6 + g16
y13 = g27 + (x28 + x1)
y9 = g1 + (x24 + g5)
y7 = g9 + (x22 + g15)
y27 = g6 + (x10 + g4)
y19 = g21 + (x19 + g10)
```

[LW16] (32×32), $D = 3$: [28 XOR2 + 35 XOR3] gates, 247.45 GE, from 98 XOR2 gates; baseline 261.77 GE (SFX23, HILL). From stage 4 started from the circuit of SFX23 (Sec. 5.1). The circuit of Appendix A.1, with 98 XOR2 gates, gives [30 XOR2 + 34 XOR3] gates, i.e., 248.78 GE. Matrix: as in Appendix A.1.

```
g1 = x29 + x6
g2 = x28 + x14
g3 = x19 + g2
g4 = x26 + x16
g5 = x7 + g4
g6 = x30 + x12
g7 = x3 + g6
g8 = x21 + x12
g9 = x28 + x5
g10 = x10 + x0
g11 = x2 + g10
g12 = x27 + x4
g13 = x26 + x18
g14 = x24 + x0
g15 = x29 + x15
g16 = x20 + g15
g17 = x16 + x8
g18 = x25 + g17
g19 = x14 + x1
g20 = x31 + x13
g21 = x23 + x10
g22 = x13 + x4
g23 = x22 + g22
g24 = x16 + x9
g25 = x15 + x2
g26 = x9 + (x25 + x1)
g27 = x17 + x1
g28 = x30 + x17
y25 = x25 + (g21 + g27)
g29 = x3 + (x25 + x20)
y28 = x28 + (x18 + g22)
y17 = g26 + (g25 + g27)
g30 = x7 + (x31 + x10)
y0 = x8 + (g14 + g28)
g31 = x5 + (x26 + x23)
y22 = x22 + (x7 + g6)
y23 = g11 + (g20 + g21)
y10 = x18 + (x27 + g10)
y21 = g1 + (x21 + x11)
y6 = g2 + (x23 + x6)
y9 = g5 + (g24 + g27)
y19 = x9 + (x19 + g12)
y16 = x24 + (g17 + g19)
y30 = g16 + (x30 + g1)
y4 = x4 + (x26 + g8)
y12 = g16 + (x12 + g25)
y26 = x11 + (x2 + g4)
y7 = g5 + (g13 + g15)
y15 = g31 + (x24 + x15)
y5 = g23 + (x5 + g12)
y20 = x10 + (x20 + g9)
y29 = g3 + (x29 + g9)
y2 = g11 + (x19 + g14)
y1 = g26 + (x31 + x18)
y31 = g30 + (x21 + x8)
y14 = g23 + (x14 + g20)
y13 = g7 + (x13 + g8)
y18 = x8 + (x3 + g13)
y11 = g3 + (x11 + g19)
y24 = x22 + (g14 + g24)
y27 = g7 + (x27 + g28)
y8 = g18 + (x6 + x0)
y3 = x11 + g29
```

[BKL16] circulant (32×32), $D = 3$: [32 XOR2 + 36 XOR3] gates, 264.12 GE, from 104 XOR2 gates; baseline 278.44 GE (SFX23, HILL). It is the circuit of Appendix A.1, merged. Matrix: as in Appendix A.1.

```
g1 = x23 + x6
g2 = x20 + x9
g3 = x31 + x9
g4 = x30 + x15
g5 = x23 + x1
g6 = x11 + x0
g7 = x28 + x17
g8 = x24 + x3
g9 = x27 + x16
g10 = x21 + x7
g11 = g10 + (x9 + x8)
g12 = x16 + x13
g13 = g12 + (x8 + x2)
g14 = x31 + x13
g15 = x19 + x8
g16 = x17 + x7
g17 = x23 + x5
g18 = x25 + x15
g19 = x24 + x21
g20 = x25 + x4
g21 = x12 + x1
g22 = x20 + g21
g23 = x31 + x14
g24 = x29 + x0
g25 = g14 + (x1 + x0)
y8 = g13 + g25
y18 = x18 + (x26 + g21)
g26 = x8 + x5
g27 = x22 + x7
g28 = g19 + (x16 + x10)
g29 = g17 + (x25 + x24)
y7 = g18 + (x24 + g27)
g30 = g7 + (x22 + x4)
g31 = x29 + x15
y21 = g11 + (x4 + g31)
g32 = g24 + (x24 + x18)
g33 = g26 + (x26 + x0)
y13 = g25 + (x28 + g10)
g34 = x3 + (x28 + x14)
y31 = g16 + (x16 + g23)
g35 = g31 + (x17 + x16)
g36 = x19 + g20
y4 = g36 + (x30 + x12)
y6 = g1 + (x14 + g19)
y14 = x22 + (g23 + g24)
y5 = g29 + (x20 + g14)
y29 = g35 + (x12 + g17)
y20 = g2 + g34
y9 = g3 + (x17 + g8)
y0 = g33 + g29
y22 = x30 + (g26 + g27)
y12 = g22 + (x27 + x6)
y11 = g33 + (x11 + g15)
y16 = g28 + g11
y24 = g32 + g35
y30 = g4 + (x6 + g12)
y15 = g5 + (x0 + g4)
y17 = g6 + (x25 + g16)
y25 = x1 + (g15 + g18)
y27 = g28 + (x27 + g8)
y2 = g7 + (x10 + x2)
y28 = x11 + g30
y26 = g2 + (x26 + x2)
y23 = x8 + (g1 + g3)
y19 = g13 + (x19 + g9)
y1 = g5 + (x9 + g9)
y3 = g32 + (x3 + g6)
y10 = x10 + (x18 + g20)
```

[SS16] Toeplitz (32×32), $D = 3$: [32 XOR2 + 35 XOR3] gates, 259.45 GE, from 102 XOR2 gates; baseline 269.78 GE (SFX23). From stage 4 started from the circuit of SFX23 (Sec. 5.1). The circuit of Appendix A.1, with 101 XOR2 gates, gives [39 XOR2 + 31 XOR3] gates, i.e., 261.77 GE. Matrix: as in Appendix A.1.

```
g1 = x13 + x3
g2 = x23 + x0
g3 = x17 + g2
g4 = x1 + g2
g5 = x16 + x0
g6 = x24 + x8
g7 = g5 + g6
g8 = x10 + x9
g9 = x3 + g8
y16 = g8 + g7
g10 = x11 + x4
g11 = x29 + x6
g12 = x31 + x1
g13 = g5 + g12
g14 = x30 + x14
g15 = x26 + x9
g16 = x18 + (x17 + x1)
g17 = x23 + x7
g18 = x18 + x11
g19 = x25 + g18
g20 = x14 + x6
g21 = x22 + g20
y0 = x25 + (g2 + g6)
y3 = g18 + (x28 + x3)
g22 = x31 + x15
g23 = x31 + x24
g24 = x17 + x10
g25 = x0 + (x20 + x12)
y9 = g9 + (g23 + g24)
g26 = g17 + g23
g27 = x21 + x7
g28 = x8 + (x28 + x14)
y7 = g26 + (x22 + g22)
g29 = x19 + x12
y23 = g26 + (x9 + g6)
y31 = x17 + (g5 + g22)
```

```
y1 = g4 + (g5 + g15)
y2 = x2 + (x27 + g24)
y6 = g26 + (g20 + g27)
g30 = x26 + x19
g31 = x2 + g30
y25 = g31 + (x25 + g15)
y11 = x5 + (x11 + g30)
y18 = g31 + (x18 + g29)
g32 = x28 + x1
y13 = g32 + (x13 + g27)
g33 = x28 + (x16 + x12)
y28 = g33 + (x22 + x5)
g34 = x29 + (x21 + x9)
y21 = g34 + (x15 + x5)
g35 = x30 + (x20 + x5)
y5 = g35 + (x24 + x13)
y30 = g3 + (g14 + g17)
y19 = g1 + (x27 + x19)
y15 = g4 + (x30 + g22)
y10 = g19 + (x10 + g10)
y14 = g21 + (g11 + g12)
y20 = g28 + (x20 + x4)
y4 = x4 + (x29 + g29)
y29 = g3 + (x13 + g11)
y26 = g9 + (x20 + g15)
y17 = g19 + g16
y27 = x27 + (x21 + g10)
y22 = g21 + (x9 + g14)
y24 = g16 + g7
y8 = g13 + (x8 + x2)
y12 = g25 + (x27 + x6)
```

[JPST17] subfield (32×32), $D = 3$: [34 XOR2 + 26 XOR3] gates, 223.42 GE, from 86 XOR2 gates; baseline 240.40 GE (SFX23, HILL). From stage 4 with the UMC area as the objective (Sec. 5.1). The circuit of Appendix A.1, with 86 XOR2 gates, gives [36 XOR2 + 25 XOR3] gates, i.e., 224.75 GE. Matrix: as in Appendix A.1.

```
g1 = x25 + x10
g2 = x27 + x1
g3 = x30 + x23
g4 = x24 + x17
g5 = x8 + x1
g6 = x28 + x13
g7 = x21 + x4
g8 = x9 + g2
g9 = x12 + x5
g10 = x23 + x6
g11 = g9 + g10
g12 = x15 + x7
y7 = g3 + g12
g13 = x14 + (x20 + x7)
g14 = x29 + x6
g15 = x31 + x21
g16 = x5 + g15
y15 = g15 + g13
g17 = g5 + (x19 + x2)
g18 = g10 + (x20 + x13)
g19 = x20 + (x31 + x12)
y23 = g18 + g19
g20 = g7 + (x22 + x15)
g21 = x16 + x9
g22 = x10 + x3
g23 = g21 + g22
g24 = x0 + (x27 + x16)
g25 = x19 + x11
g26 = g22 + (x17 + x0)
y11 = g24 + g26
g27 = x26 + x3
y3 = g25 + g27
y26 = g27 + g26
g28 = x11 + (x18 + x0)
y17 = g26 + (x11 + g1)
y29 = g18 + (x7 + g14)
y12 = g20 + (x28 + x20)
y28 = g11 + (x28 + x4)
y24 = g17 + (x24 + x0)
y6 = x14 + (x22 + g14)
y16 = g23 + (x24 + x8)
y2 = g1 + (x18 + x2)
y19 = g8 + g17
y27 = g2 + g28
y0 = x8 + g24
y22 = x30 + (x22 + g9)
y4 = x4 + g19
y14 = g3 + g18
y21 = x29 + (g7 + g12)
y31 = g16 + g20
y9 = x25 + (g5 + g25)
y1 = g4 + g8
y18 = g5 + (x26 + x18)
y20 = g13 + (x12 + g6)
y8 = g28 + (x16 + g4)
y25 = g23 + (x19 + g1)
y13 = g11 + (x15 + g14)
y5 = g6 + g16
y30 = g7 + (x30 + x14)
y10 = x2 + (x26 + g21)
```

[SKOP15] involutory (32×32), $D = 3$: [43 XOR2 + 26 XOR3] gates, 250.42 GE, from 95 XOR2 gates; baseline 256.76 GE (SFX23, HILL). The circuit of Appendix A.1, with 94 XOR2 gates, gives [50 XOR2 + 22 XOR3] gates, i.e., 252.74 GE. Matrix: as in Appendix A.1.

```
g1 = x10 + x3
g2 = x25 + x19
g3 = x15 + x6
g4 = x20 + x4
g5 = x15 + g4
g6 = x30 + x14
g7 = x5 + (x13 + x12)
g8 = x24 + x8
g9 = x11 + g8
g10 = x16 + x0
g11 = x27 + g10
g12 = x17 + x16
g13 = x8 + x1
g14 = g12 + g13
y11 = g9 + g14
g15 = g10 + (x26 + x10)
g16 = x28 + x12
g17 = x23 + g16
g18 = x22 + x6
g19 = x28 + x4
g20 = g19 + (x21 + x5)
g21 = x31 + g16
y31 = g20 + g21
g22 = x9 + x0
g23 = x30 + x23
y30 = g23 + g5
g24 = x22 + x7
g25 = g23 + g24
y22 = g24 + g21
g26 = x19 + x2
y2 = g26 + g9
g27 = x24 + (x25 + x17)
g28 = g19 + (x29 + x13)
y23 = g17 + g28
g29 = x7 + g4
y7 = g28 + g29
g30 = x31 + x14
y14 = g30 + g29
y20 = g7 + (g4 + g6)
g31 = x27 + x11
g32 = g8 + (x18 + x2)
g33 = x9 + x3
g34 = g31 + (x18 + x3)
y19 = x8 + (g2 + g22)
y3 = g27 + (g12 + g33)
g35 = g3 + g30
y9 = x26 + (g26 + g33)
y5 = g35 + (x5 + g18)
y21 = g35 + (x21 + g6)
g36 = g31 + (x26 + x19)
y26 = g11 + g36
y16 = g15 + (x9 + g13)
g37 = x29 + (x21 + x20)
y4 = g37 + (g6 + g19)
y17 = g36 + (x17 + g26)
y12 = g37 + (g16 + g18)
y18 = g9 + g34
y10 = g1 + g11
y27 = g14 + g11
y29 = g25 + (x29 + g6)
y0 = g15 + g27
y25 = g2 + (x18 + g1)
y15 = g20 + g5
y24 = g32 + (x1 + g22)
y8 = g32 + (x25 + g12)
y13 = g25 + (x13 + g18)
y28 = g7 + (g18 + g19)
y6 = g3 + g17
y1 = g34 + (x1 + g1)
```

[LW16] involutory (32×32), $D = 3$: [53 XOR2 + 18 XOR3] gates, 243.06 GE, from 89 XOR2 gates; baseline 247.42 GE (HILL). From stage 4 started from the circuit of SFX23 (Sec. 5.1). The circuit of Appendix A.1, with 87 XOR2 gates, gives [63 XOR2 + 12 XOR3] gates, i.e., 245.04 GE. Matrix: as in Appendix A.1.

```
g1 = x13 + (x29 + x4)
g2 = x16 + x1
g3 = x24 + g2
g4 = x26 + x3
g5 = x18 + g4
g6 = x26 + x10
y1 = g6 + g3
g7 = x18 + x2
g8 = g7 + (x16 + x0)
g9 = g6 + (x24 + x8)
g10 = x14 + x6
g11 = x31 + g10
g12 = x9 + x1
g13 = g7 + g12
g14 = x28 + x12
y3 = g14 + g5
g15 = x27 + x2
g16 = x11 + g15
y18 = g13 + g16
g17 = x25 + x17
y2 = g17 + g16
g18 = x27 + x19
y4 = g18 + g1
g19 = x19 + g4
y26 = g12 + g19
g20 = x21 + x14
g21 = x25 + x8
g22 = g6 + g17
y10 = g19 + g22
g23 = x0 + g21
y17 = g22 + g23
y25 = g7 + g23
g24 = x20 + x4
g25 = x10 + g15
y27 = g24 + g25
g26 = g14 + g18
y19 = g25 + g26
g27 = x21 + (x28 + x5)
y12 = g26 + g27
g28 = x11 + x3
g29 = g24 + g28
y28 = g28 + g27
g30 = x30 + x22
g31 = x31 + x23
g32 = x22 + x13
g33 = x7 + g30
y7 = g9 + g33
g34 = x15 + x7
g35 = g32 + (x12 + x6)
y29 = g1 + g35
g36 = x17 + x9
y24 = g34 + (x24 + g36)
y16 = g2 + (x25 + g34)
y15 = g8 + (x15 + g30)
g37 = g20 + (x29 + x15)
y6 = g11 + g37
g38 = g20 + (x30 + x20)
y5 = g27 + g38
y13 = g35 + (x20 + g14)
y0 = x0 + (g31 + g36)
g39 = g32 + (x23 + x5)
y30 = g33 + g39
y21 = g38 + (x12 + g24)
y8 = g21 + (x1 + g31)
y22 = g39 + (x15 + g31)
y14 = g37 + (x23 + g34)
y20 = g1 + g29
y9 = g13 + g3
y11 = g29 + g5
y23 = g9 + (x23 + g10)
y31 = g8 + g11
```

[SS16] involutory (32×32), $D = 4$: [30 XOR2 + 33 XOR3] gates, 244.11 GE, from 96 XOR2 gates; baseline 263.41 GE (SFX23). From stage 4 with the UMC area as the objective (Sec. 5.1). The circuit of Appendix A.1, with 94 XOR2 gates, gives [50 XOR2 + 22 XOR3] gates, i.e., 252.74 GE. Matrix: as in Appendix A.1.

```
g1 = x30 + x15
g2 = x23 + x17
g3 = x12 + x4
g4 = x9 + x1
y23 = g1 + (x23 + g4)
g5 = x21 + x15
g6 = x6 + g5
y15 = x31 + g6
g7 = x29 + x7
g8 = x14 + g7
y7 = x23 + g8
g9 = x8 + x0
g10 = x19 + (x28 + x4)
g11 = x27 + (x20 + x12)
g12 = g4 + (x22 + x7)
y31 = x31 + g12
g13 = g4 + g9
g14 = x10 + (x24 + x2)
g15 = g2 + (x26 + x2)
g16 = g13 + (x14 + x6)
y20 = g11 + g16
g17 = x25 + x9
g18 = x13 + (x5 + g9)
g19 = x18 + (x27 + x3)
y27 = g19 + g18
g20 = x0 + (x24 + x16)
g21 = x31 + x25
g22 = x18 + x10
g23 = g21 + g22
y18 = g3 + g23
g24 = x11 + x3
g25 = x22 + (x28 + x13)
g26 = x21 + g25
g27 = g22 + (x26 + x9)
g28 = x17 + x1
g29 = x30 + (x20 + x5)
g30 = x29 + g29
y1 = g21 + (x8 + g28)
y10 = g12 + (x16 + g27)
g31 = x19 + (x26 + x11)
y8 = g12 + (g13 + g20)
y29 = y23 + (g7 + g29)
y0 = g1 + g20
y25 = x16 + (g17 + g24)
y14 = y7 + g30
y2 = g14 + (g1 + g27)
y3 = x19 + (g19 + g23)
y22 = y31 + (g8 + g13)
y6 = g26 + y15
y16 = g20 + (x31 + g14)
y12 = g11 + (x28 + g19)
y9 = g2 + (x0 + g17)
y17 = x24 + (g24 + g28)
y21 = y31 + (g5 + g25)
y19 = g31 + g18
y26 = g3 + g15
y11 = g31 + (x27 + g15)
y30 = y23 + (g6 + g13)
y28 = g10 + g16
y5 = g30 + (g5 + g11)
y4 = g31 + (x20 + g10)
y13 = g26 + (g7 + g10)
y24 = g14 + (x23 + x8)
```

[KLSW17] (32×32), $D = 3$: [42 XOR2 + 20 XOR3] gates, 219.40 GE, from 82 XOR2 gates; baseline 230.72 GE (SFX23, HILL, for HW160 #93). From stage 4 started from the circuit of SFX23 (Sec. 5.1). The circuit of Appendix A.1, with 80 XOR2 gates, gives [52 XOR2 + 14 XOR3] gates, i.e., 221.38 GE. Matrix: as in Appendix A.1.

```
g1 = x30 + x7
g2 = x22 + x15
g3 = x18 + x11
g4 = x14 + x4
g5 = x12 + x6
g6 = x27 + x17
g7 = x24 + g3
y3 = g6 + (x10 + x3)
g8 = x15 + x5
g9 = x31 + g8
g10 = g5 + g8
y20 = x20 + g9
g11 = x13 + x7
g12 = x23 + g11
g13 = g4 + g11
g14 = x31 + x21
g15 = g14 + (x30 + x14)
y13 = g13 + g15
g16 = x11 + x1
g17 = x8 + x2
g18 = g16 + g17
g19 = x3 + (x19 + x9)
g20 = g1 + (x20 + x6)
g21 = x29 + x23
g22 = g21 + (x22 + x6)
g23 = x26 + x16
g24 = x10 + x0
y27 = g3 + (x27 + g17)
g25 = g2 + (x28 + x14)
y31 = g2 + (x31 + g5)
g26 = g16 + (x27 + x10)
g27 = g24 + (x26 + x9)
y8 = g23 + (x8 + x3)
g28 = x19 + x2
y2 = x9 + (g23 + g28)
g29 = g24 + (x25 + x16)
y25 = g19 + g29
g30 = g28 + (x25 + x18)
y26 = g26 + (x26 + g6)
y24 = x24 + g19
y6 = g12 + g20
y10 = g7 + g26
y7 = g1 + g15
y1 = g30 + g18
y0 = x0 + g7
y23 = g1 + (x23 + g4)
y14 = g9 + g25
y15 = g2 + g22
y11 = g3 + g30
y21 = g10 + (x28 + g14)
y16 = x16 + (x27 + g16)
y28 = x28 + g12
y5 = g22 + g10
y22 = g12 + g22
y17 = g18 + (x24 + g6)
y19 = g19 + g27
y9 = g27 + y3
y29 = g13 + (x20 + g21)
y4 = g4 + g25
y30 = g15 + g9
y12 = g5 + g20
y18 = g19 + g30
```

[LSL+19] HW168 #130 (32×32), $D = 3$: [41 XOR2 + 21 XOR3] gates, 221.07 GE, from 83 XOR2 gates; baseline 229.39 GE (SFX23, HILL). From stage 4 started from the circuit of SFX23 (Sec. 5.1). The circuit of Appendix A.1, with 82 XOR2 gates, gives [46 XOR2 + 18 XOR3] gates, i.e., 222.06 GE. Matrix: as in Appendix A.1.

```
g1 = x15 + x7
g2 = x29 + x11
g3 = x19 + g2
g4 = x29 + x21
g5 = x3 + g4
g6 = x28 + x20
g7 = x2 + g6
g8 = x31 + x13
g9 = g4 + g8
y1 = x1 + g9
g10 = x30 + x12
g11 = g6 + g10
y0 = x0 + g11
g12 = x28 + x10
g13 = x18 + g12
g14 = x23 + x15
g15 = g14 + (x25 + x13)
y3 = g5 + g15
g16 = x27 + x9
g17 = g14 + g16
g18 = x24 + x8
g19 = x9 + x1
g20 = x19 + g19
y9 = g19 + g5
g21 = g18 + (x18 + x0)
y24 = x16 + g21
g22 = g18 + (x26 + x14)
g23 = x22 + x4
g24 = g23 + (x30 + x20)
y10 = g12 + g24
g25 = x31 + x5
g26 = g25 + (x23 + x21)
y11 = g2 + g26
g27 = x25 + x17
y25 = g27 + g20
g28 = x24 + x22
y22 = g22 + (x6 + g28)
g29 = g28 + (x14 + x12)
g30 = x30 + x16
y30 = g30 + (x14 + x6)
y12 = g29 + y30
y4 = g22 + (g23 + g30)
y5 = g17 + (g25 + g27)
y26 = g7 + (x26 + g12)
y31 = g1 + (x31 + x17)
y13 = g15 + y31
y18 = g13 + (x2 + g10)
y16 = x16 + (x0 + g12)
y8 = g7 + (x8 + x0)
y20 = g24 + g29
y15 = g20 + (x15 + g16)
y21 = g15 + g26
y27 = g5 + (x27 + g2)
y23 = x7 + g17
y29 = g9 + g26
y2 = g29 + g7
y6 = g13 + (x26 + x6)
y7 = g3 + (x27 + x7)
y14 = g22 + g21
y19 = g3 + (x3 + g8)
y17 = g2 + (x17 + x1)
y28 = g24 + g11
```