Improved Heuristics for Short Linear Programs

Thomas Peyrin   Quan Quan Tan

Nanyang Technological University

CHES 2020

Contributions of this paper:

A new algorithm that finds good implementations of linear systems, to reduce the number of XOR gates/operations.

Our algorithm **performs better than the state-of-the-art** (Paar and Boyar-Peralta algorithms), we tested on existing and also random matrices.

Diffusion Matrices

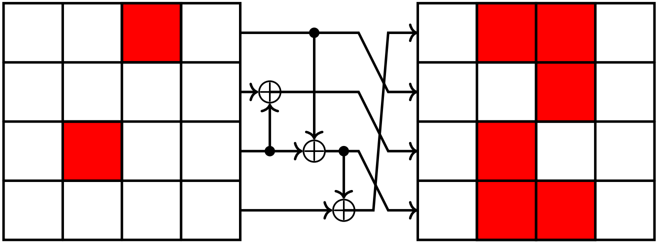


Figure 1: Figure inspired from [Jea16]

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} 2 \cdot w_0 \oplus 3 \cdot w_1 \oplus w_2 \oplus w_3 \\ w_0 \oplus 2 \cdot w_1 \oplus 3 \cdot w_2 \oplus w_3 \\ w_0 \oplus w_1 \oplus 2 \cdot w_2 \oplus 3 \cdot w_3 \\ 3 \cdot w_0 \oplus w_1 \oplus w_2 \oplus 2 \cdot w_3 \end{bmatrix}, w_i \in GF(2^8)$$

Diffusion Matrices

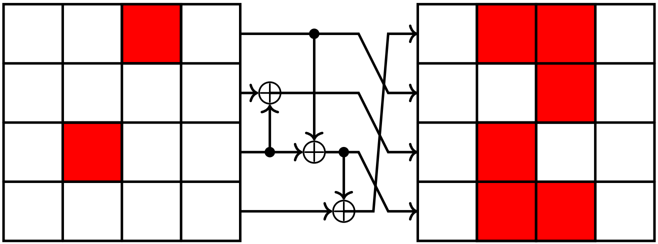


Figure 1: Figure inspired from [Jea16]

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} 2 \cdot w_0 \oplus 3 \cdot w_1 \oplus w_2 \oplus w_3 \\ w_0 \oplus 2 \cdot w_1 \oplus 3 \cdot w_2 \oplus w_3 \\ w_0 \oplus w_1 \oplus 2 \cdot w_2 \oplus 3 \cdot w_3 \\ 3 \cdot w_0 \oplus w_1 \oplus w_2 \oplus 2 \cdot w_3 \end{bmatrix}, w_i \in GF(2^8)$$

From $GF(2^n)$ to $GF(2)$

Multiplication by a fixed element in $GF(2^n)$ can be replaced by a $n \times n$ binary matrix multiplication.

$w_0 = x_7x_6x_5x_4x_3x_2x_1x_0$
irreducible polynomial $= p^8 + p^4 + p^3 + p + 1$

$$3 \times w_0 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_7 \\ x_6 \\ x_5 \\ x_4 \\ x_3 \\ x_2 \\ x_1 \\ x_0 \end{bmatrix}$$

From $GF(2^n)$ to $GF(2)$

Multiplication by a fixed element in $GF(2^n)$ can be replaced by a $n \times n$ binary matrix multiplication.

$w_0 = x_7 x_6 x_5 x_4 x_3 x_2 x_1 x_0$
irreducible polynomial $= p^8 + p^4 + p^3 + p + 1$

$$3 \times w_0 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_7 \\ x_6 \\ x_5 \\ x_4 \\ x_3 \\ x_2 \\ x_1 \\ x_0 \end{bmatrix}$$

Number of Computations

Problem

For any given fixed matrix M, how can we minimize the number of '$\oplus$' operations required to compute it ?

- Naive counting (d-XOR). Compute each row individually.
- Sequential counting (g-XOR). Count the actual number of sequential XORs required for all the rows.

Example

$$y_0 = x_0 \oplus x_1 \oplus x_2$$
$$y_1 = x_1 \oplus x_2 \oplus x_3$$

$$t_0 = x_1 \oplus x_2$$
$$y_0 = x_0 \oplus t_0$$
$$y_1 = t_0 \oplus x_3$$

d-XOR : 4

g-XOR : 3

Past Works: Paar's Algorithm [PR97]

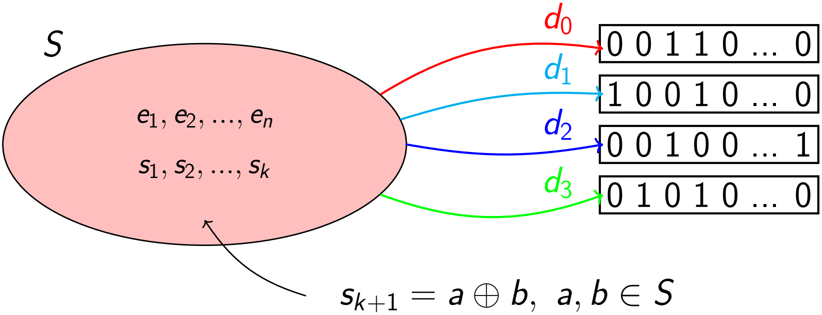
Idea: identify most frequent (x_i, x_j) pairs and use an XOR to compute $x_i \oplus x_j$. Repeat until done.

$$\begin{array}{ccccc} x_0 & x_1 & x_2 & x_3 & x_4 \end{array}$$
$$\begin{pmatrix} \color{red}{1} & 1 & 1 & 1 & \color{red}{1} \\ \color{red}{1} & 1 & 0 & 1 & \color{red}{1} \\ \color{red}{1} & 0 & 1 & 1 & \color{red}{1} \\ 0 & 1 & 0 & 0 & 1 \\ \color{red}{1} & 0 & 1 & 0 & \color{red}{1} \end{pmatrix} \rightarrow \begin{array}{c} \begin{array}{cccccc} x_0 & x_1 & x_2 & x_3 & x_4 & t_0 \end{array} \\ \begin{pmatrix} \color{red}{0} & 1 & 1 & 1 & \color{red}{0} & \color{blue}{1} \\ \color{red}{0} & 1 & 0 & 1 & \color{red}{0} & \color{blue}{1} \\ \color{red}{0} & 0 & 1 & 1 & \color{red}{0} & \color{blue}{1} \\ 0 & 1 & 0 & 0 & 1 & 0 \\ \color{red}{0} & 0 & 1 & 0 & \color{red}{0} & \color{blue}{1} \end{pmatrix} \end{array}$$

In the case of a tie,

- Choose the first one in lexicographical order (Paar1)
- Exhaust all equally frequent options (Paar2)

Past Works: Boyar-Peralta's algorithm [BP10]



1. Choose s_{k+1} such that $d_0 + d_1 + \ldots + d_n$ is minimized
2. L2-norm is used in an event of a tie

An alternative criteria: Shortest-Dist-First
Instead of using the L1-norm as the criteria, the criteria selects the pair that is able to reduce as many "nearest" targets as possible.

Suppose the current distance vector to the targets is $[3, 4, 2, 2, 4, 5]$

Candidate's distance	[2,3,2,2,3,4]	[3,4,1,1,4,5]
BP criteria [BP10]	✓	
SDF criteria [RTA18]		✓

An alternative criteria: Shortest-Dist-First
Instead of using the L1-norm as the criteria, the criteria selects the pair that is able to reduce as many "nearest" targets as possible.

Suppose the current distance vector to the targets is $[3, 4, 2, 2, 4, 5]$

Candidate's distance	[2,3,2,2,3,4]	[3,4,1,1,4,5]
BP criteria [BP10]	✓	
SDF criteria [RTA18]		✓

Randomized Algorithms

Limitations

- BP algorithm's implementation follows a lexicographical order which did not consider all other pairs that are equally good.
- Paar1 suffers from the same issue as BP
- Paar2 exhaustively searches through all the possible pairs, which is costly for matrices that are relatively large

Solution

1. When we have more than one equally good pairs, randomly pick one of them.
2. Repeat the algorithm k times and pick the best circuit.

Randomized Algorithms

Limitations

- BP algorithm's implementation follows a lexicographical order which did not consider all other pairs that are equally good.
- Paar1 suffers from the same issue as BP
- Paar2 exhaustively searches through all the possible pairs, which is costly for matrices that are relatively large

Solution

1. When we have more than one equally good pairs, randomly pick one of them.
2. Repeat the algorithm k times and pick the best circuit.

Our Criteria

Relaxing the criteria of having to reduce as many nearest targets as possible + maintaining the "main path" using L1-norm.

1. Shortlist all pairs such that at least one of the "nearest" targets is reduced
2. Apply L1-norm criteria to the remaining pairs. (A1)
3. If there is a tie, apply L2-norm criteria. (A2)

Suppose the current distance vector to the targets is $[3, 4, 2, 2, 4, 5]$

Candidate's distance	[2,3,2,2,3,5]	[3,4,1,1,4,5]	[3,3,1,2,4,4]
BP criteria [BP10]	✓		
SDF criteria [RTA18]		✓	
Our criteria			✓

Our Criteria

Relaxing the criteria of having to reduce as many nearest targets as possible + maintaining the "main path" using L1-norm.

1. Shortlist all pairs such that at least one of the "nearest" targets is reduced
2. Apply L1-norm criteria to the remaining pairs. (A1)
3. If there is a tie, apply L2-norm criteria. (A2)

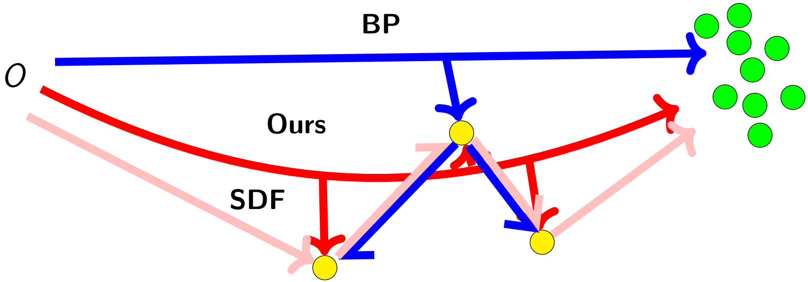
Suppose the current distance vector to the targets is $[3, 4, 2, 2, 4, 5]$

Candidate's distance	[2,3,2,2,3,5]	[3,4,1,1,4,5]	[3,3,1,2,4,4]
BP criteria [BP10]	✓		
SDF criteria [RTA18]		✓	
Our criteria			✓

Rationale of our Criteria

Our guess: targets with high distance often cluster together

- High distance targets dominate the path from the start
- Targets with a lower distance can play a part in the path towards targets with a higher distance value.



Local Optimization

Given a circuit, find some ways to reduce the number of XORs.

Yosys [Wol]

Verilog RTL synthesis tool that does some optimization

Our local optimization techniques

$$\vdots$$

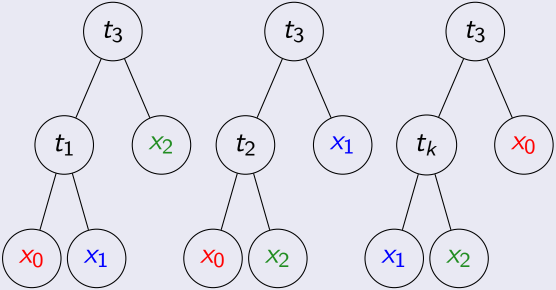
$$t_1 = x_0 \oplus x_1$$

$$t_2 = x_0 \oplus x_2$$

$$t_3 = x_2 \oplus t_1$$

$$t_4 = x_3 \oplus t_2$$

$$\vdots$$



Results (Random Matrices [VSP18])

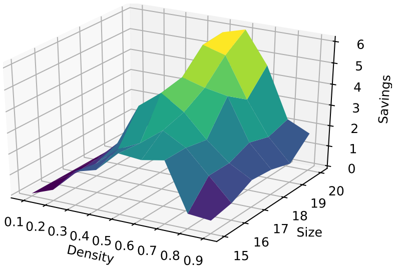


Figure 2: Average XOR count difference (A1 vs BP)

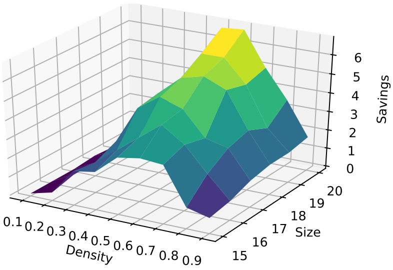


Figure 3: Average XOR count difference (A2 vs BP)

Our algorithms outperform BP for random matrices. The improvement is more obvious with the increase in size.

Results (Random Matrices [VSP18])

Table 1: Percentage of best circuits obtained

Matrix Size	**BP** [BP10]	**Paar1** [PR97]	**RPaar1** [New]	**SDF** [RTA18]	**RNBP** [New]	**A1** [New]	**A2** [New]
15×15	25.56	14.44	14.44	**70.00**	38.89	58.89	66.67
16×16	21.11	8.89	10.00	61.11	28.89	53.33	**73.33**
17×17	17.78	11.11	11.11	62.22	26.67	53.33	**72.22**
18×18	15.56	8.89	11.11	41.11	31.11	52.22	**85.56**
19×19	14.44	11.11	11.11	32.22	26.67	54.44	**74.44**
20×20	12.22	11.11	11.11	25.56	23.33	58.89	**87.78**

Results (Matrices from [DL18])

Table 2: XOR count of 16 × 16 matrices

Matrix	Instantiation (α, β, γ)	Const. [BP10]	BP [PR97]	Paar2 [RTA18]	RSDF [DL18]	RNBP [New]	A1 [New]	A2 [New]
$M_{4,5}^{9,3}$	$(A_4, -, -)$	35	38	45	**36**	37	39	37
$M_{4,5}^{9,3}$	$(A_4^{-1}$	36	40	46	38	39	38	**35**
$M_{4,6}^{8,3}$	$(A_4, -, -)$	35	38	45	**37**	38	39	38
$M_{4,6}^{8,3}$	$(A_4^{-1}$	35	40	46	36	38	38	**35**
$M_{4,5}^{8,3}$	$(A_4^{-1}, A_4, A_4^{-2})$	36	40	47	40	39	**38**	**38**
$M_{4,4}^{9,4}$	$(A_4, -, -)$	39	41	47	41	40	**39**	**39**
$M_{4,4}^{9,3}$	$(A_4^{-1}, A_4, A_4^{-2})$	40	40	43	40	**39**	41	41
$M_{4,4}^{8,4}$	$(A_4, -, -)$	38	40	43	41	**39**	40	**39**
$M_{4,4}^{8,4'}$	$(A_4, -, -)$	38	43	41	**38**	41	39	**38**
$M_{4,4}^{8,4''}$	$(A_4, -, -)$	37	40	43	40	40	40	**39**
$M_{4,3}^{9,5}$	$(A_4, -, -)$	41	**40**	43	41	**40**	41	**40**
$M_{4,3}^{9,5}$	$(A_4^{-1}, -, -)$	41	43	44	44	41	41	**40**

Results (Matrices from [DL18])

Table 3: XOR count of 32×32 matrices

Matrix	Instantiation (α, β, γ)	Const. [DL18]	BP [BP10]	Paar2 [PR97]	RSDF [RTA18]	RNBP [New]	A1 [New]	A2 [New]
$M_{4,5}^{9,3}$	$(A_8, -, -)$	67	74	88	74	**67**	77	69
$M_{4,5}^{9,3}$	$(A_8^{-1}, -, -)$	67	71	89	79	69	78	**68**
$M_{4,6}^{8,3}$	$(A_8, -, -)$	67	74	88	71	**67**	76	69
$M_{4,6}^{8,3}$	$(A_8^{-1}, -, -)$	67	71	89	78	69	78	**68**
$M_{4,5}^{8,3}$	$(A_8^{-1}, A_8, A_8^{-2})$	68	75	77	81	**68**	**68**	**68**
$M_{4,4}^{9,4}$	$(A_8, -, -)$	76	77	92	84	**76**	**76**	**76**
$M_{4,4}^{9,3}$	(A_8^{-1}, A_8, A_8^{2})	76	76	83	79	**75**	76	76
$M_{4,4}^{8,4}$	$(A_8, -, -)$	70	72	74	77	**70**	**70**	**70**
$M_{4,4}^{8,4'}$	$(A_8, -, -)$	70	81	79	76	76	72	**71**
$M_{4,4}^{8,4''}$	$(A_8, -, -)$	69	72	85	77	**69**	76	70
$M_{4,3}^{9,5}$	$(A_8, -, -)$	77	**76**	86	82	**76**	**76**	**76**
$M_{4,3}^{9,5}$	$(A_8^{-1}, -, -)$	77	79	86	85	**77**	**77**	**77**

Results (AES)

Matrix	BP [BP10]	RSDF [RTA18]	RNBP [New]	A1 [New]	A2 [New]
AES MixCol	97 [KLSW17]	102	95	95	**94**
AES InvMixCol	155	162	153	153	**152**

Very recently, [Max19, XZL^{+}20] further improved our result for AES matrix to 92 XORs

Conclusion and Future Works

- A1 and A2 criteria perform the best when the densities of the matrices are about 0.4-0.5.
- However, our algorithm is BP-like (like [RTA18]) which makes it too costly if the matrix grows very large
- More techniques in local optimization may lead to even lower XOR count.
- The average (XOR) cost of implementing a matrix with density 0.9 is actually less than one with a density of 0.2.

Conclusion and Future Works

- A1 and A2 criteria perform the best when the densities of the matrices are about 0.4-0.5.
- However, our algorithm is BP-like (like [RTA18]) which makes it too costly if the matrix grows very large
- More techniques in local optimization may lead to even lower XOR count.
- The average (XOR) cost of implementing a matrix with density 0.9 is actually less than one with a density of 0.2.

Conclusion and Future Works

- A1 and A2 criteria perform the best when the densities of the matrices are about 0.4-0.5.
- However, our algorithm is BP-like (like [RTA18]) which makes it too costly if the matrix grows very large
- More techniques in local optimization may lead to even lower XOR count.
- The average (XOR) cost of implementing a matrix with density 0.9 is actually less than one with a density of 0.2.

Conclusion and Future Works

- A1 and A2 criteria perform the best when the densities of the matrices are about 0.4-0.5.
- However, our algorithm is BP-like (like [RTA18]) which makes it too costly if the matrix grows very large
- More techniques in local optimization may lead to even lower XOR count.
- The average (XOR) cost of implementing a matrix with density 0.9 is actually less than one with a density of 0.2.

References I

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, Ischia Island, Naples, Italy, May 20-22, 2010. Proceedings*, volume 6049 of *Lecture Notes in Computer Science*, pages 178–189. Springer, 2010.

Sébastien Duval and Gaëtan Leurent.
MDS Matrices with Lightweight Circuits.
IACR Trans. Symmetric Cryptol., 2018(2):48–78, 2018.

Jérémy Jean.
TikZ for Cryptographers.
`https://www.iacr.org/authors/tikz/`, 2016.

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.

Alexander Maximov.
AES MixColumn with 92 XOR gates.
IACR Cryptology ePrint Archive, 2019:833, 2019.

References II

Christof Paar and Martin Rosner.
Comparison of arithmetic architectures for Reed-Solomon decoders in reconfigurable hardware.
In *5th IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM '97), 16-18 April 1997, Napa Valley, CA, USA*, pages 219–225. IEEE Computer Society, 1997.

Arash Reyhani-Masoleh, Mostafa M. I. Taha, and Doaa Ashmawy.
Smashing the Implementation Records of AES S-box.
IACR Trans. Cryptogr. Hardw. Embed. Syst., 2018(2):298–336, 2018.

Andrea Visconti, Chiara Valentina Schiavo, and René Peralta.
Improved upper bounds for the expected circuit complexity of dense systems of linear equations over GF(2).
Inf. Process. Lett., 137:1–5, 2018.

Clifford Wolf.
Yosys open synthesis suite.
http://www.clifford.at/yosys/.

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.