Smashing the Implementation Records of AES S-box

Arash Reyhani-Masoleh, Mostafa Taha, and Doaa Ashmawy

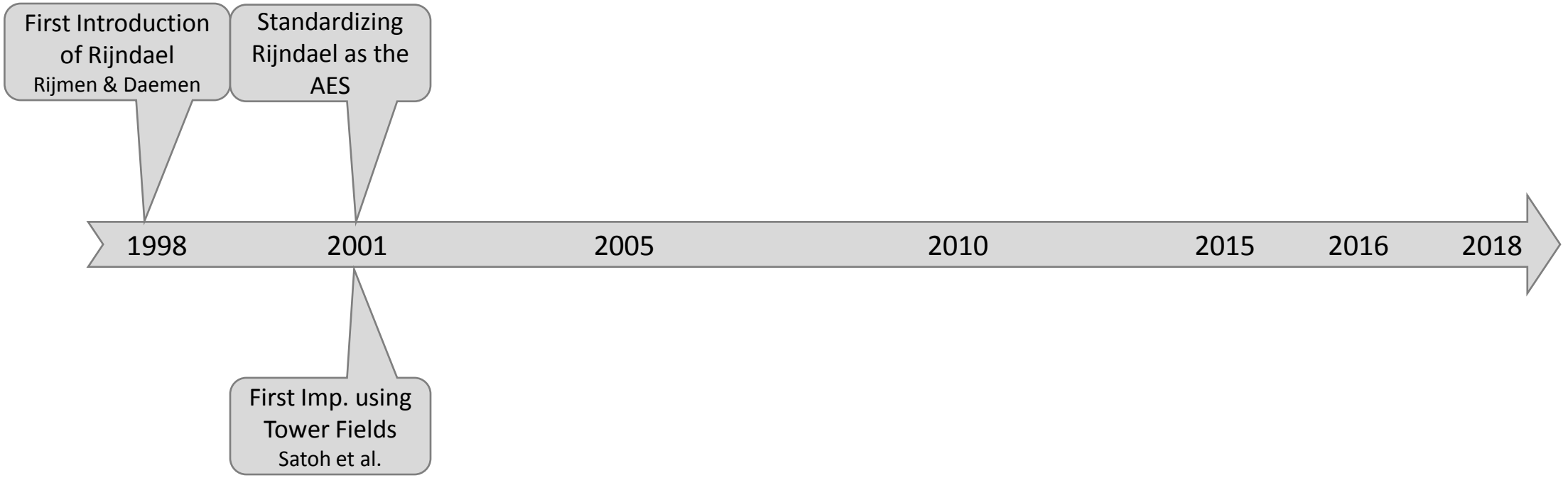
Western University
London, Ontario, Canada

CHES-2018

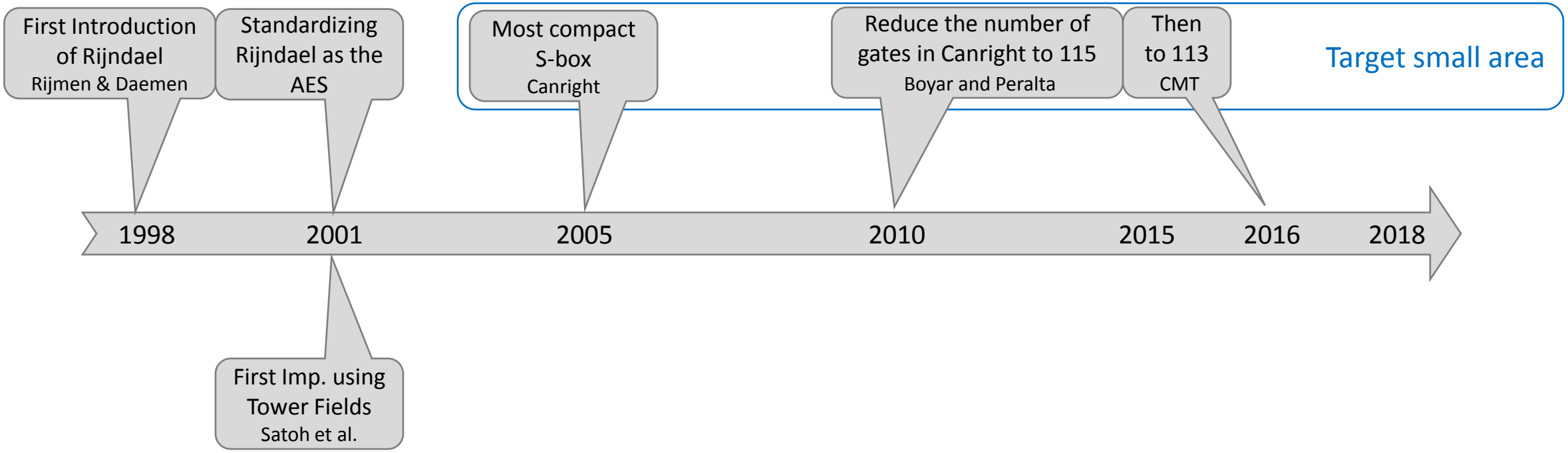
Outline

- Introduction.
- Proposed AES S-box Architecture.
- New Logic-Minimization Algorithms.
- New GF($(2^4)^2$) Inversion.
  - New Exponentiation Stage.
  - New Representation of Subfield Inversion.
  - New Output Multipliers.
- Comparisons and Concluding Remarks.

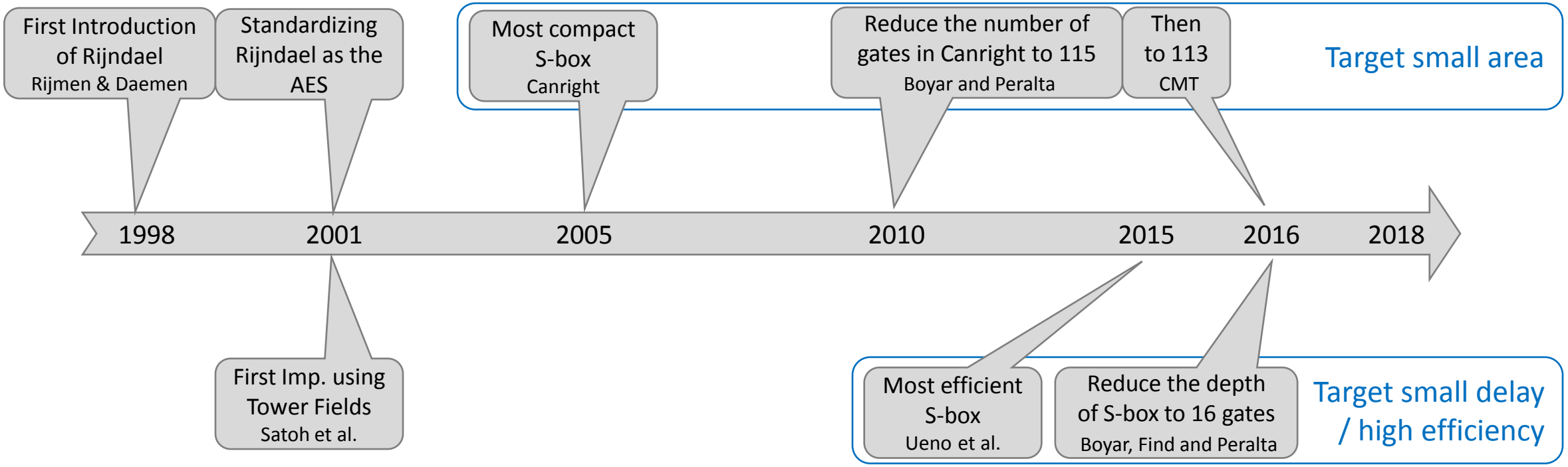
Introduction



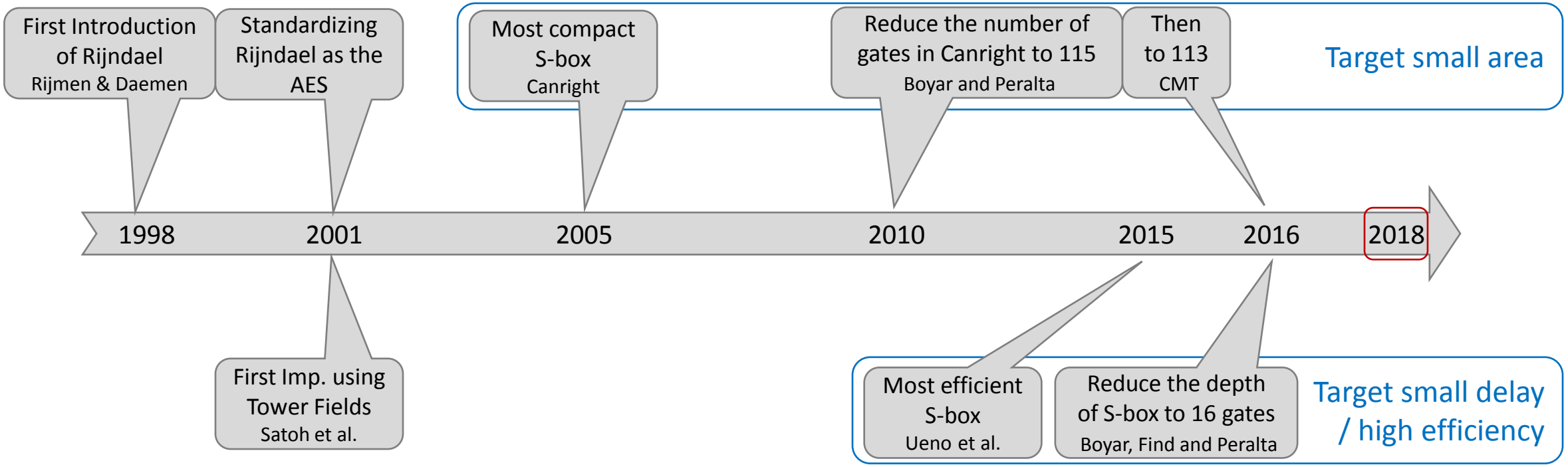
Introduction



Introduction



Introduction



In this paper, we propose:

1. The most compact S-box to date.

2. The most efficient S-box to date.

Implementation Pitfalls

1. Use AND gates,
   when NAND gates have smaller area and delay in all technology libraries.

Implementation Pitfalls

1. Use AND gates,
   when NAND gates have smaller area and delay in all technology libraries.
2. Use only simple gates,
   when compound gates (AND-OR-Invert, OR-AND-Invert) may be more efficient.

Implementation Pitfalls

1. Use AND gates,
   when NAND gates have smaller area and delay in all technology libraries.
2. Use only simple gates,
   when compound gates (AND-OR-Invert, OR-AND-Invert) may be more efficient.

- We improved previous designs using AND gates to the ones using NAND/NOR gates:

S-box	Area (GEs)		Delay (ns)	
	Original	Improved	Original	Improved
Canright [Can05b]	**200**		1.253	
113-gates [Boy16]	202	**194**	1.523	1.346
Depth-16 (2012) [BP12]	230.5	222	0.960	0.906
Depth-16 (2017) [BFP17]	224.5	216	0.957	0.912
Ueno et al. [UHS+15]	256.5	238	**0.831**	**0.772**

Targeting STM 65-nm CMOS standard library

Implementation Pitfalls

1. Use AND gates,
   when NAND gates have smaller area and delay in all technology libraries.
2. Use only simple gates,
   when compound gates (AND-OR-Invert, OR-AND-Invert) may be more efficient.

- We improved previous designs using AND gates to the ones using NAND/NOR gates:

The smallest original

The fastest original

S-box	Area (GEs)		Delay (ns)	
	Original	Improved	Original	Improved
Canright [Can05b]	**200**		1.253	
113-gates [Boy16]	202	**194**	1.523	1.346
Depth-16 (2012) [BP12]	230.5	222	0.960	0.906
Depth-16 (2017) [BFP17]	224.5	216	0.957	0.912
Ueno et al. [UHS+15]	256.5	238	**0.831**	**0.772**

Targeting STM 65-nm CMOS standard library

Implementation Pitfalls

1. Use AND gates,
   when NAND gates have smaller area and delay in all technology libraries.
2. Use only simple gates,
   when compound gates (AND-OR-Invert, OR-AND-Invert) may be more efficient.

- We improved previous designs using AND gates to the ones using NAND/NOR gates:

The smallest original

The smallest improved

The fastest original

The fastest improved

S-box	Area (GEs)		Delay (ns)	
	Original	Improved	Original	Improved
Canright [Can05b]	**200**		1.253	
113-gates [Boy16]	202	**194**	1.523	1.346
Depth-16 (2012) [BP12]	230.5	222	0.960	0.906
Depth-16 (2017) [BFP17]	224.5	216	0.957	0.912
Ueno et al. [UHS+15]	256.5	238	**0.831**	**0.772**

Targeting STM 65-nm CMOS standard library

Implementation Pitfalls

1. Use AND gates,
   when NAND gates have smaller area and delay in all technology libraries.
2. Use only simple gates,
   when compound gates (AND-OR-Invert, OR-AND-Invert) may be more efficient.

- We improved previous designs using AND gates to the ones using NAND/NOR gates:

The smallest original

The smallest improved

The fastest original

The fastest improved

S-box	Area (GEs)		Delay (ns)	
	Original	Improved	Original	Improved
Canright [Can05b]	**200**		1.253	
113-gates [Boy16]	202	**194**	1.523	1.346
Depth-16 (2012) [BP12]	230.5	222	0.960	0.906
Depth-16 (2017) [BFP17]	224.5	216	0.957	0.912
Ueno et al. [UHS+15]	256.5	238	**0.831**	**0.772**

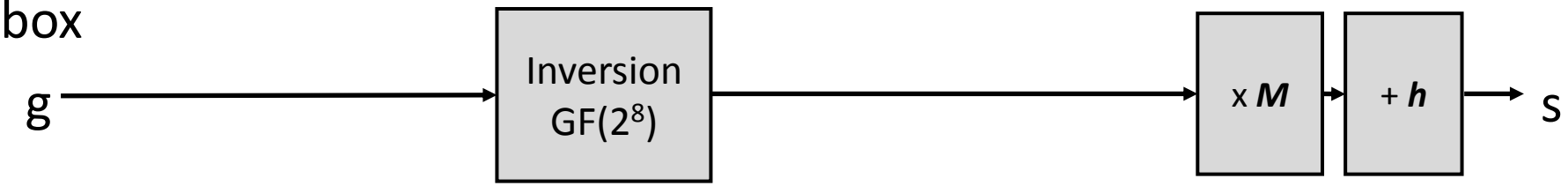
Targeting STM 65-nm CMOS standard library

At the end, we compare only against the Improved Versions.

Formulations of the improved designs are included in the paper.

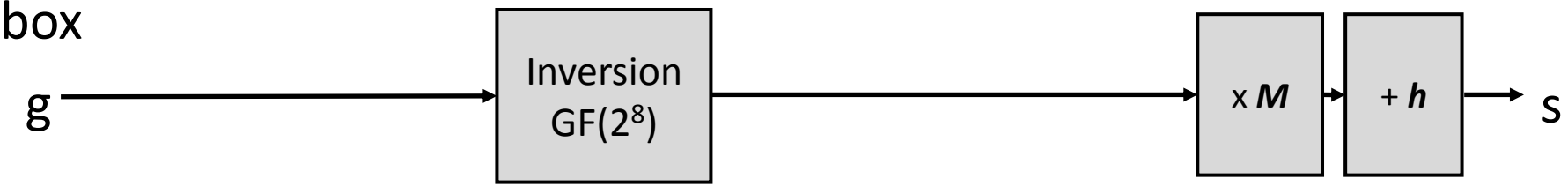
AES S-box

- Original S-box

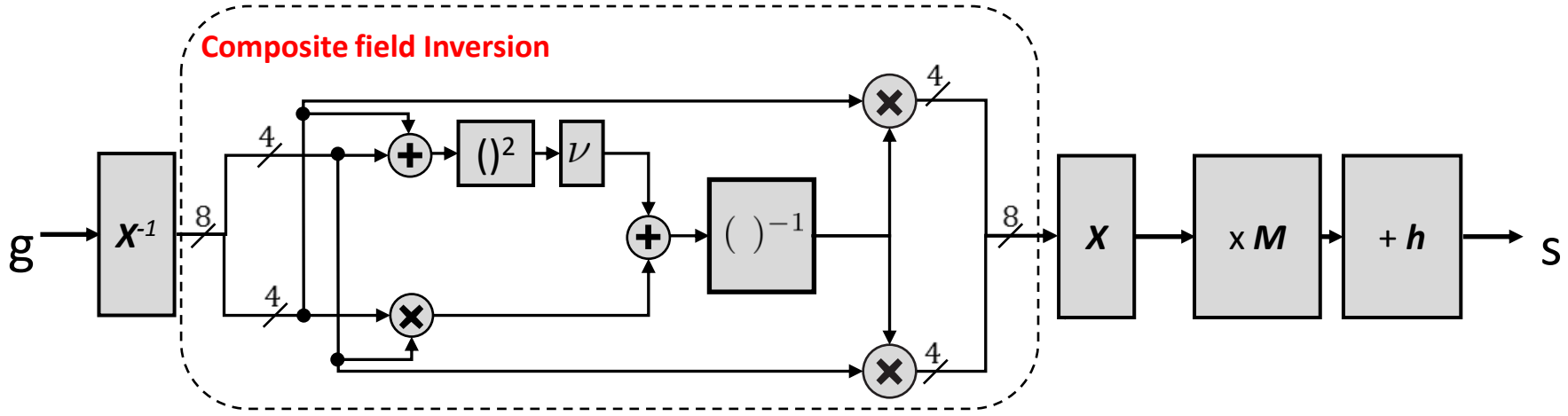


AES S-box

- Original S-box

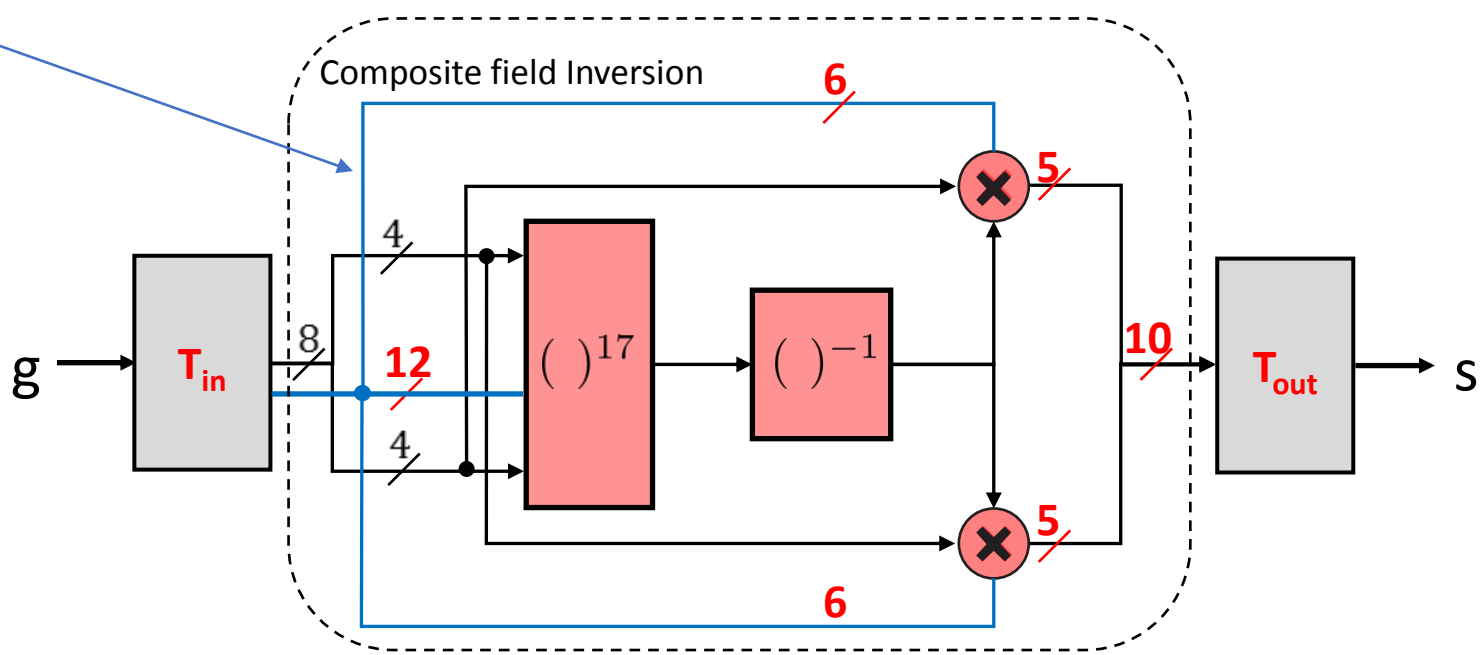


- Typical implementation using Composite Fields in Normal Basis



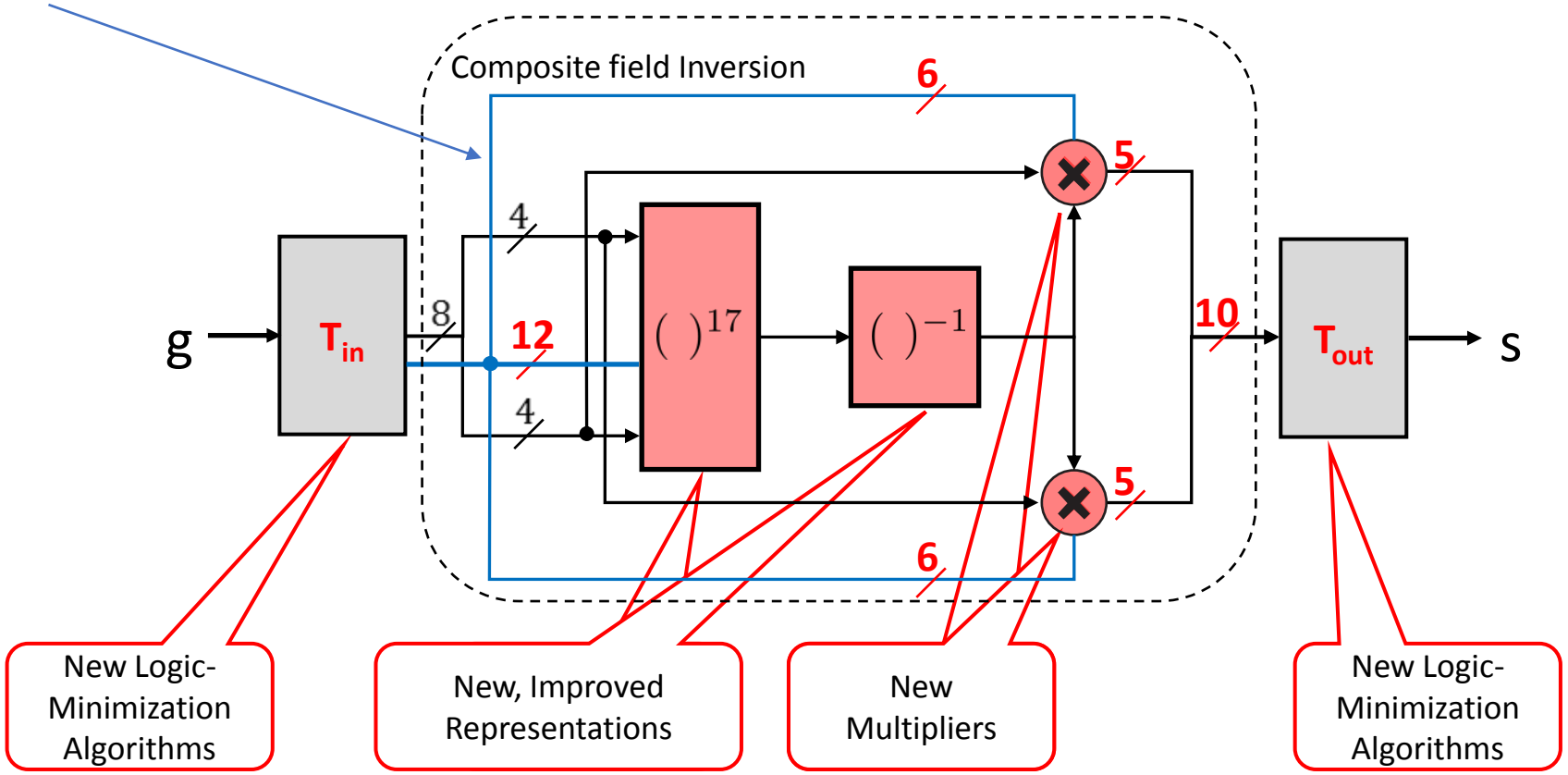
Proposed AES S-box Architecture

- 12 terms are shared between the Exponentiation and Multipliers



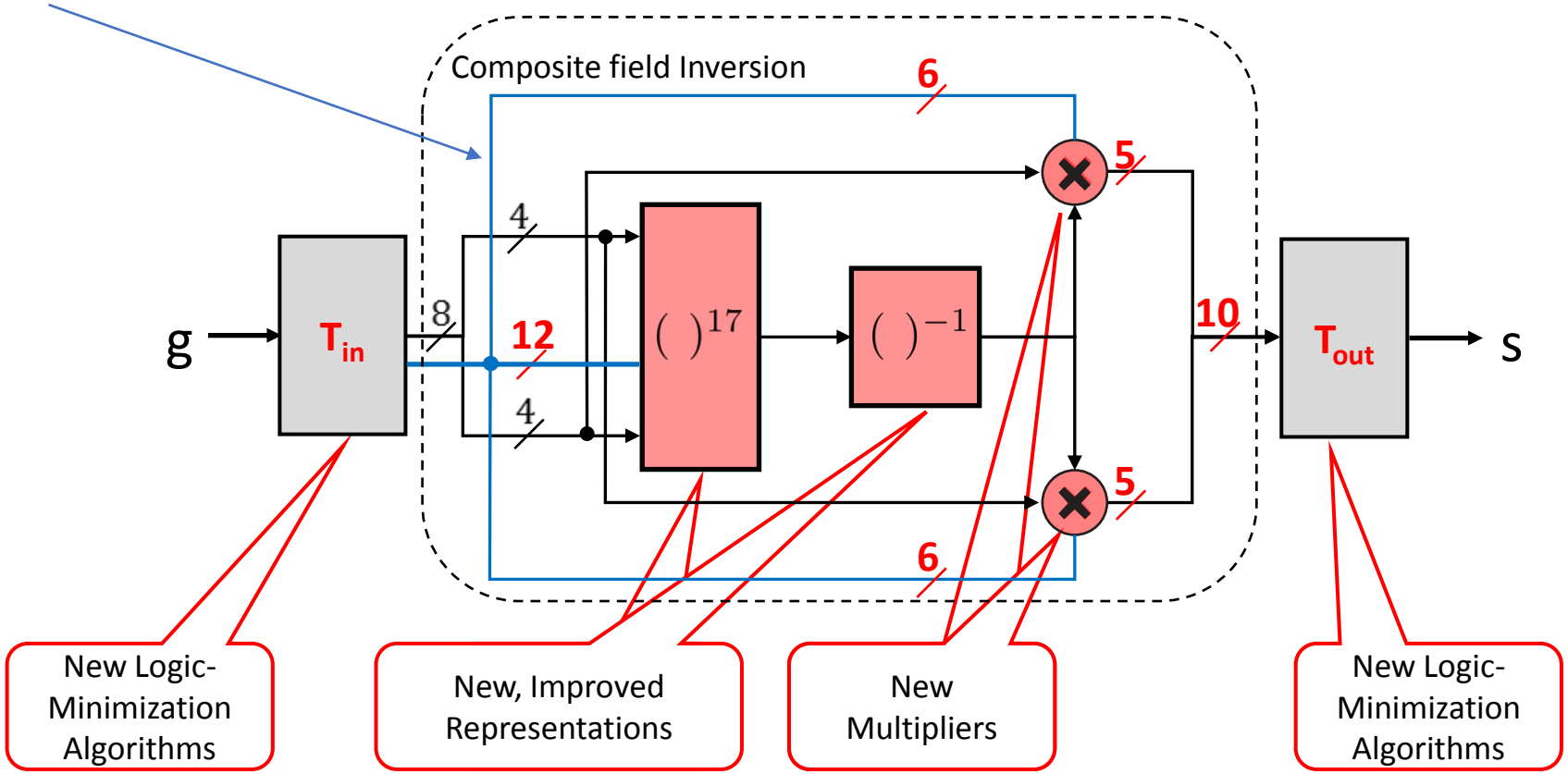
Proposed AES S-box Architecture

- 12 terms are shared between the Exponentiation and Multipliers



Proposed AES S-box Architecture

- 12 terms are shared between the Exponentiation and Multipliers



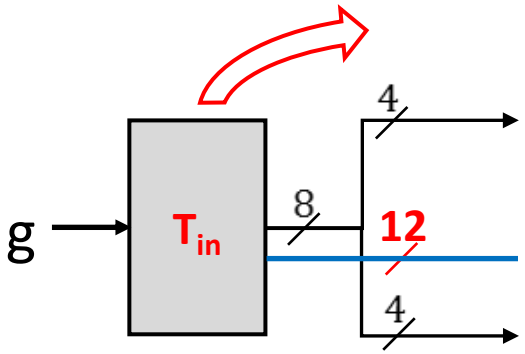
Everything optimized by-hand and by CAD tools at various abstraction levels (promote using NAND/NOR and compound gates)

Outline

- Introduction, Motivation and Previous Work.
- Proposed AES S-box Architecture.
- New Logic-Minimization Algorithms.
- New GF($(2^4)^2$) Inversion.
  - New Exponentiation Stage.
  - New Representation of Subfield Inversion.
  - New Output Multipliers.
- Comparisons and Concluding Remarks.

Logic-Minimization Algorithms

- Implement isomorphic transformation matrices using smallest number of gates.
  - NP-hard problem [BMP08].



$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ b_0 \\ b_1 \\ b_2 \\ b_3 \\ a_{01} \\ a_{02} \\ a_{03} \\ a_{12} \\ a_{13} \\ a_{23} \\ b_{01} \\ b_{02} \\ b_{03} \\ b_{12} \\ b_{13} \\ b_{23} \end{bmatrix} = \begin{bmatrix} 0&0&0&0&1&1&0&1 \\ 0&0&0&0&0&1&0&1 \\ 1&1&1&0&0&0&1&1 \\ 0&0&1&0&0&0&0&1 \\ 0&1&1&1&1&1&0&1 \\ 1&0&1&0&0&1&1&1 \\ 0&1&0&0&1&1&0&1 \\ 1&1&1&1&1&1&0&1 \\ 0&0&0&0&1&0&0&0 \\ 1&1&1&0&1&1&1&0 \\ 0&0&1&0&1&1&0&0 \\ 1&1&1&0&0&1&1&0 \\ 0&0&1&0&0&1&0&0 \\ 1&1&0&0&0&0&1&0 \\ 1&1&0&1&1&0&1&0 \\ 0&0&1&1&0&0&0&0 \\ 1&0&0&0&0&0&0&0 \\ 1&1&1&0&1&0&1&0 \\ 0&1&0&1&1&0&1&0 \\ 1&0&1&1&0&0&0&0 \end{bmatrix} \begin{bmatrix} g_7 \\ g_6 \\ g_5 \\ g_4 \\ g_3 \\ g_2 \\ g_1 \\ g_0 \end{bmatrix}$$

T in

Input Rep. in GF((2⁴)²)

12 shared terms

Logic-Minimization Algorithms (cont.)

- Implement isomorphic transformation matrices using smallest number of gates.
  - NP-hard problem [BMP08].
- Previous work
  - Cancellation-free search:
    Gates are never used to cancel-out common terms, Canright [Can05b] and Paar [Paa94].

First 8 rows of T_{in}

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} g_7 \\ g_6 \\ g_5 \\ g_4 \\ g_3 \\ g_2 \\ g_1 \\ g_0 \end{bmatrix}$$

Logic-Minimization Algorithms (cont.)

- Implement isomorphic transformation matrices using smallest number of gates.
  - NP-hard problem [BMP08].
- Previous work
  - Cancellation-free search:
    Gates are never used to cancel-out common terms, Canright [Can05b] and Paar [Paa94].
  - Heuristics (with cancellation):
    Normal-BP (Boyar and Peralta [BP10])

First 8 rows of T_{in}

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} g_7 \\ g_6 \\ g_5 \\ g_4 \\ g_3 \\ g_2 \\ g_1 \\ g_0 \end{bmatrix}$$

$$\begin{aligned} g_7 &= a_0 \\ g_6 &= a_1 \\ g_5 &= a_2 \\ g_4 &= a_3 \\ g_3 &= b_0 \\ g_2 &= b_1 \\ g_1 &= b_2 \\ g_0 &= b_3 \end{aligned}$$

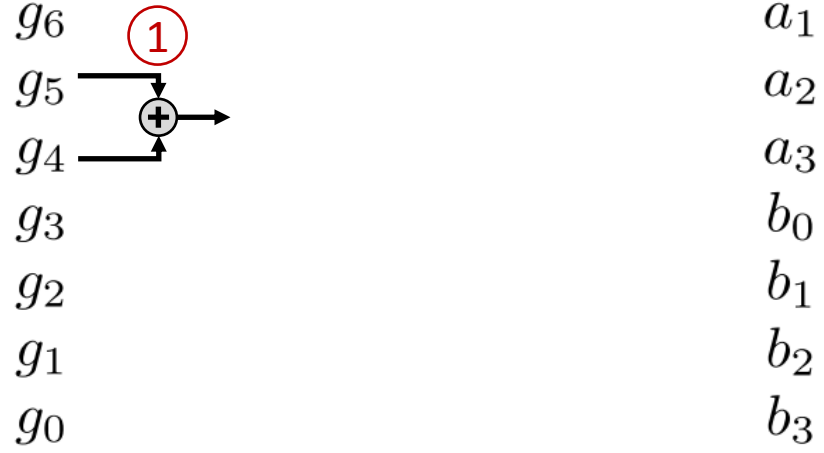
Logic-Minimization Algorithms (cont.)

- Implement isomorphic transformation matrices using smallest number of gates.
  - NP-hard problem [BMP08].
- Previous work
  - Cancellation-free search:
    Gates are never used to cancel-out common terms, Canright [Can05b] and Paar [Paa94].
  - Heuristics (with cancellation):
    Normal-BP (Boyar and Peralta [BP10])

    1. Test adding one gate

First 8 rows of T_{in}

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} g_7 \\ g_6 \\ g_5 \\ g_4 \\ g_3 \\ g_2 \\ g_1 \\ g_0 \end{bmatrix}$$



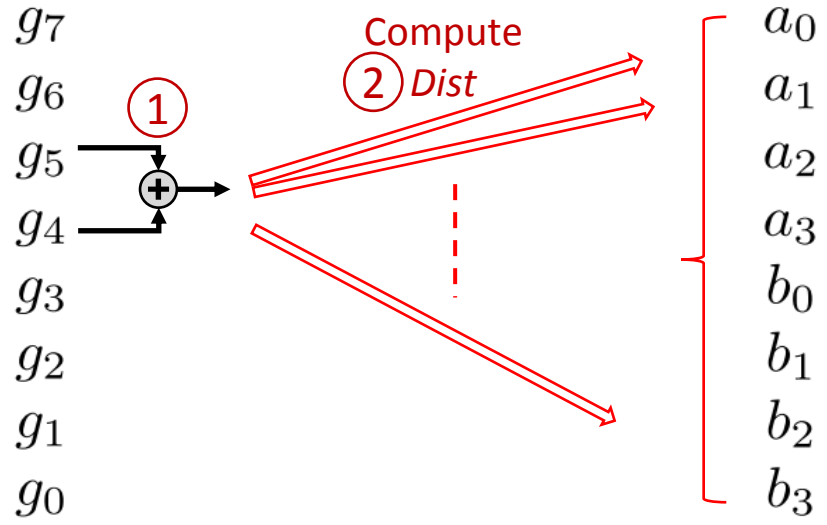
Logic-Minimization Algorithms (cont.)

- Implement isomorphic transformation matrices using smallest number of gates.
  - NP-hard problem [BMP08].
- Previous work
  - Cancellation-free search:
    Gates are never used to cancel-out common terms, Canright [Can05b] and Paar [Paa94].
  - Heuristics (with cancellation):
    Normal-BP (Boyar and Peralta [BP10])

    1. Test adding one gate
    2. Compute *Distance* to each target (assuming no sharing)

First 8 rows of T_{in}

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} g_7 \\ g_6 \\ g_5 \\ g_4 \\ g_3 \\ g_2 \\ g_1 \\ g_0 \end{bmatrix}$$



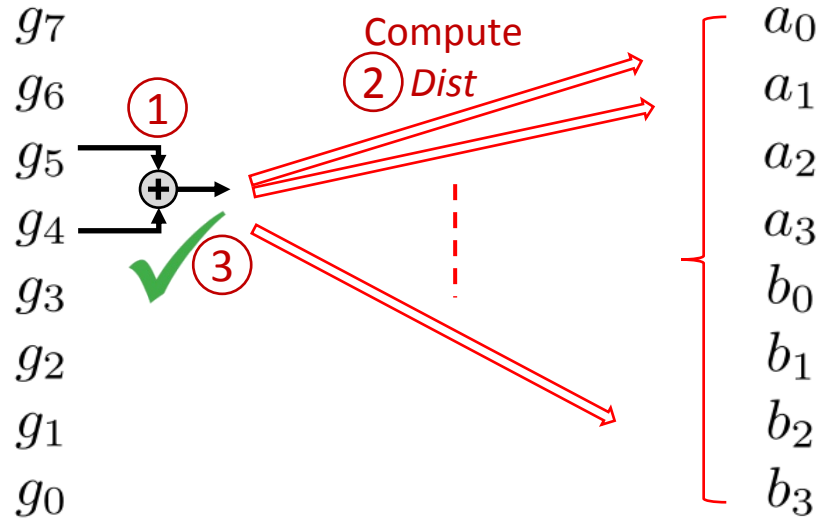
Logic-Minimization Algorithms (cont.)

- Implement isomorphic transformation matrices using smallest number of gates.
  - NP-hard problem [BMP08].
- Previous work
  - Cancellation-free search:
    Gates are never used to cancel-out common terms, Canright [Can05b] and Paar [Paa94].
  - Heuristics (with cancellation):
    Normal-BP (Boyar and Peralta [BP10])

    1. Test adding one gate
    2. Compute *Distance* to each target (assuming no sharing)
    3. Select a gate leading to the (min average *Dist*)
       Resolve ties using different methods.

First 8 rows of T_{in}

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} g_7 \\ g_6 \\ g_5 \\ g_4 \\ g_3 \\ g_2 \\ g_1 \\ g_0 \end{bmatrix}$$



Logic-Minimization Algorithms (cont.)

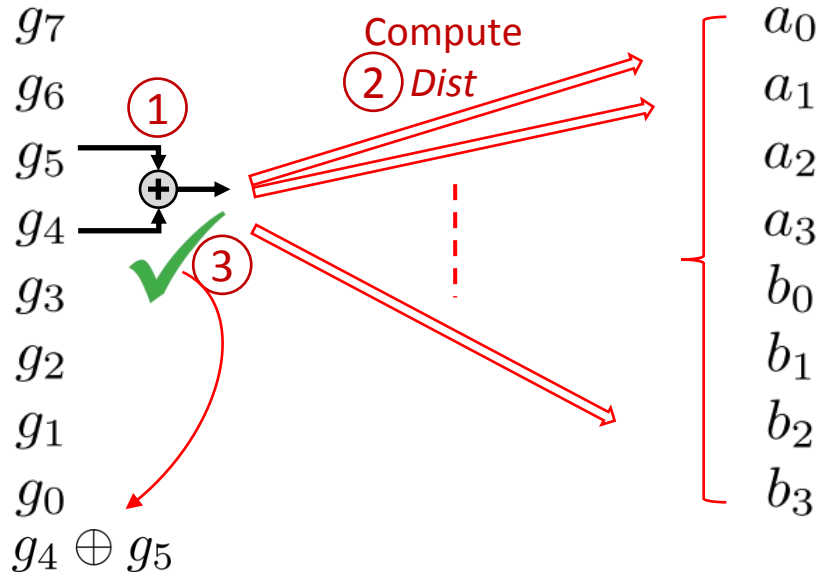
- Implement isomorphic transformation matrices using smallest number of gates.
  - NP-hard problem [BMP08].
- Previous work
  - Cancellation-free search:
    Gates are never used to cancel-out common terms, Canright [Can05b] and Paar [Paa94].
  - Heuristics (with cancellation):
    Normal-BP (Boyar and Peralta [BP10])

Add the selected gate and redo

1. Test adding one gate
2. Compute *Distance* to each target (assuming no sharing)
3. Select a gate leading to the (min average *Dist*) Resolve ties using different methods.

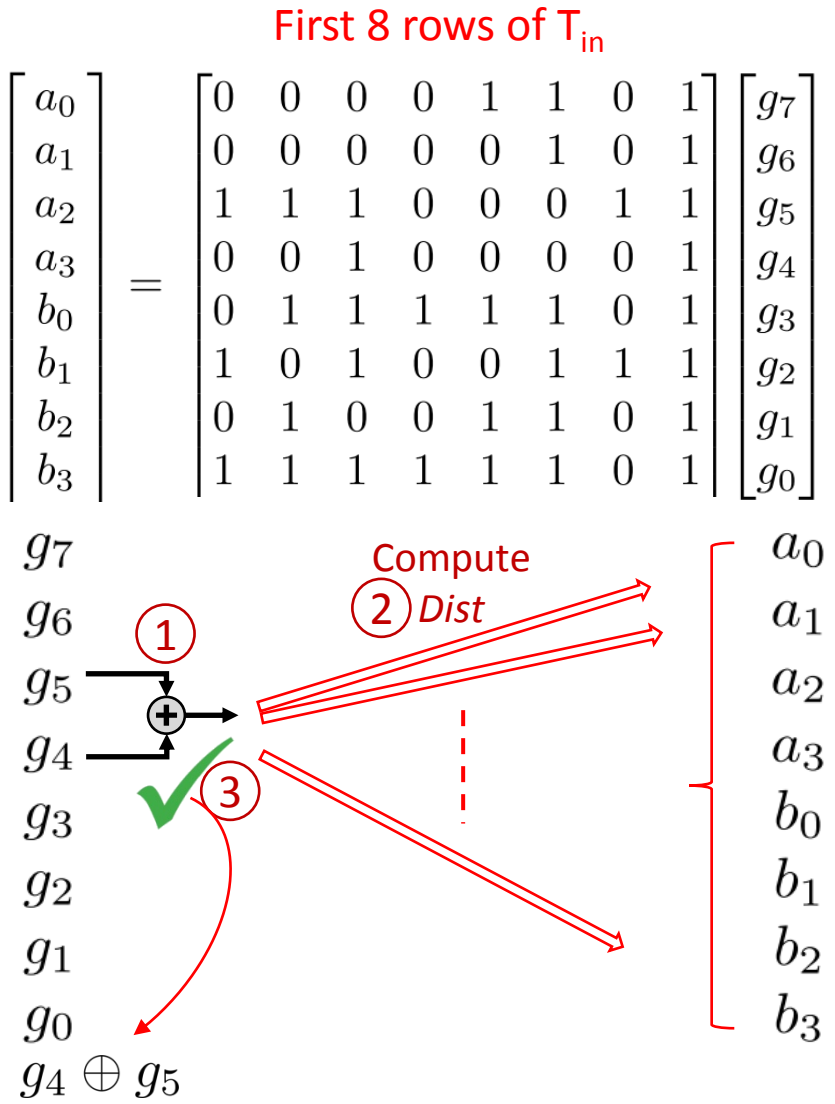
First 8 rows of T_{in}

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} g_7 \\ g_6 \\ g_5 \\ g_4 \\ g_3 \\ g_2 \\ g_1 \\ g_0 \end{bmatrix}$$



Logic-Minimization Algorithms (cont.)

- Proposed Logic-Minimization Algorithms
  - Improved-BP:
    - Test all the ties.
    - Monitor progress of the delay.
  - Shortest-Dist-First:
    - Select a gate leading to many small (short) *Distances* (prioritize small *Distances*, not the average).
    - Test all the ties and monitor the delay.
  - Focused-Search:
    - Select a gate leading to any small (short) *Distance* (ignore the count and search through more cases) (close to exhaustive search).
    - Test all the ties and monitor the delay.



Logic-Minimization Algorithms (cont.)

- Studied T_{in} and T_{out} for all possible isomorphic transformations (a total of 96 matrices).

Logic-Minimization Algorithms (cont.)

- Studied T_{in} and T_{out} for all possible isomorphic transformations (a total of 96 matrices).
- The proposed algorithms consistently lead to equal or better implementations.

Logic-Minimization Algorithms (cont.)

- Studied T_{in} and T_{out} for all possible isomorphic transformations (a total of 96 matrices).
- The proposed algorithms consistently lead to equal or better implementations.
- Lightweight Implementation

	Optimized by CAD tools	Normal-BP	Improved-BP	Shortest-Dist-First	Focused-Search
T_{in} (#gates)	29	19	19	19	19
T_{out} (#gates)	23	19	17	17	**16**

Logic-Minimization Algorithms (cont.)

- Studied T_{in} and T_{out} for all possible isomorphic transformations (a total of 96 matrices).
- The proposed algorithms consistently lead to equal or better implementations.
- Lightweight Implementation

	Optimized by CAD tools	Normal-BP	Improved-BP	Shortest-Dist-First	Focused-Search
T_{in} (#gates)	29	19	19	19	19
T_{out} (#gates)	23	19	17	17	**16**

- Fast Implementation

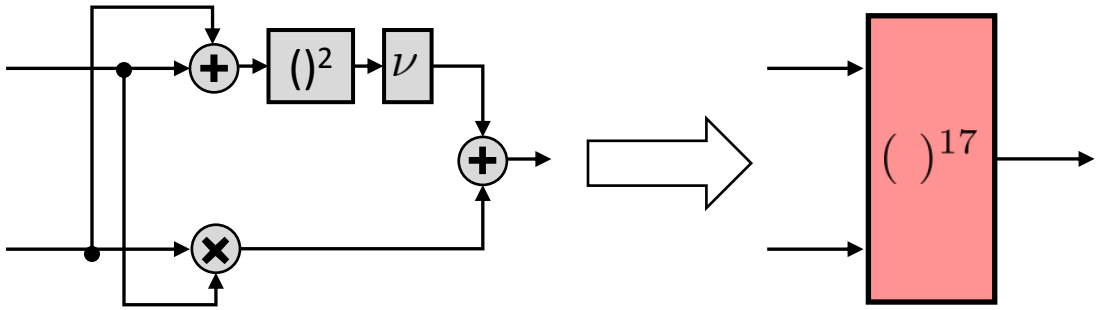
	Area (# XOR gates)	Delay (levels of XOR gates)
T_{in} (#gates)	24	3
T_{out} (#gates)	21	3

Outline

- Introduction, Motivation and Previous Work.
- Proposed AES S-box Architecture.
- New Logic-Minimization Algorithms.
- New GF($(2^4)^2$) Inversion.
  - New Exponentiation Stage.
  - New Representation of Subfield Inversion.
  - New Output Multipliers.
- Comparisons and Concluding Remarks.

New Exponentiation Stage

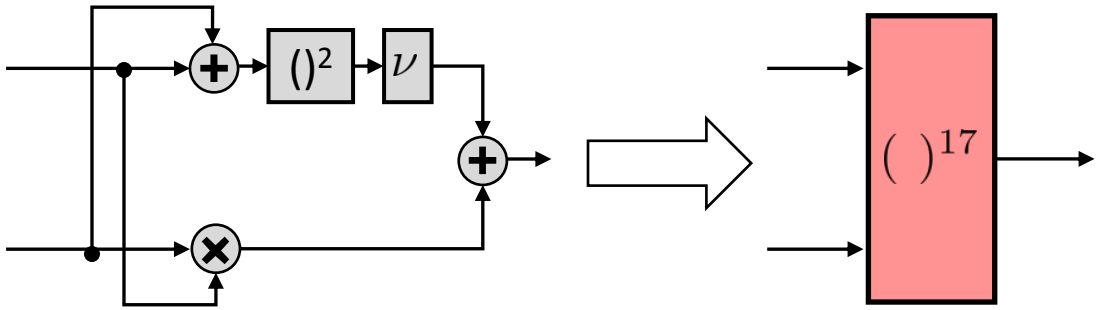
- Express as one operation with closed-form equations (allows for maximum sharing).



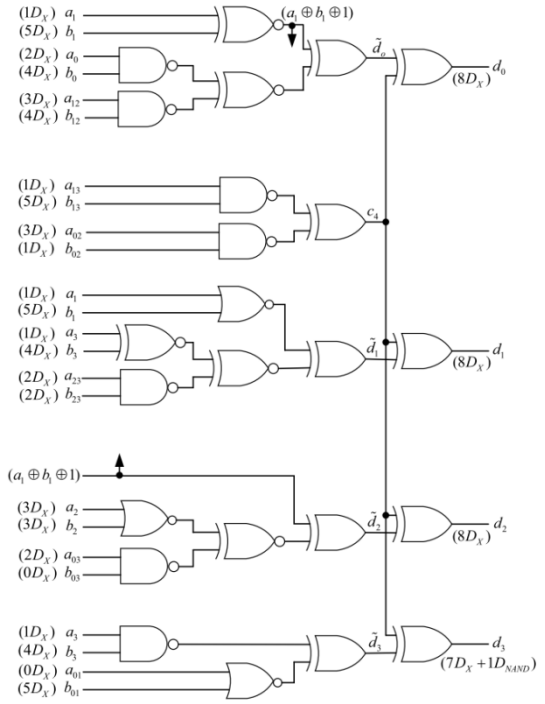
New Exponentiation Stage

- Express as one operation with closed-form equations (allows for maximum sharing).
- Two designs: Lightweight and Fast.

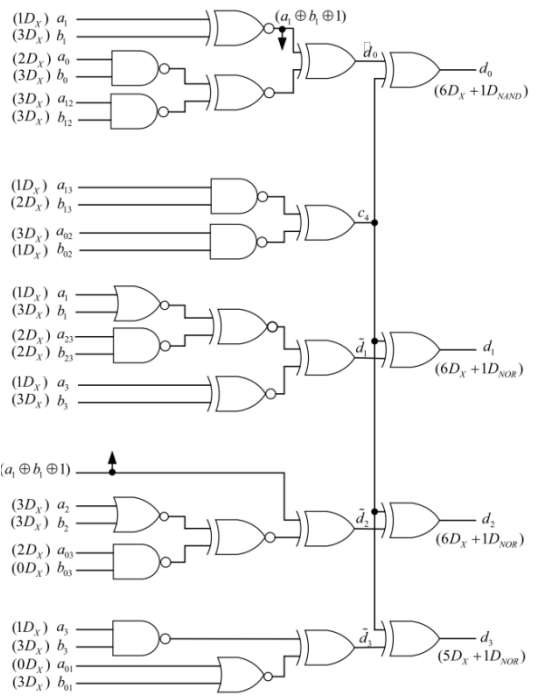
  (Optimized by hand)
- One design optimized by CAD tools.



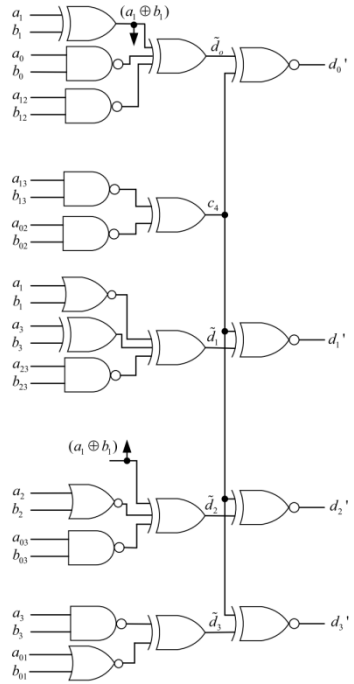
New Exponentiation Stage (cont.)



1. Lightweight
(optimized by-hand)



2. Fast
(optimized by-hand)

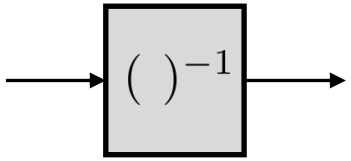


3. Optimized by CAD tool
(Used XOR3 gates)

	Area (GEs)	Delay (ns)
1. Lightweight (optimized by-hand)	30	0.103
2. Fast (optimized by-hand)	30	**0.091**
3. Optimized by CAD tool	**29.25**	0.100

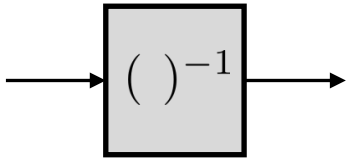
New Subfield Inversion

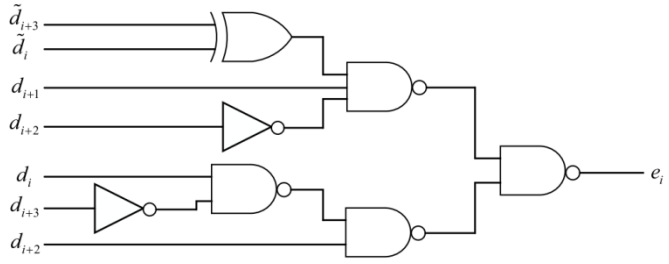
- Express in closed-form equations
- Derive 12 equivalent functions using Karnough maps, and optimize by-hand.
- Optimized using CAD tools.



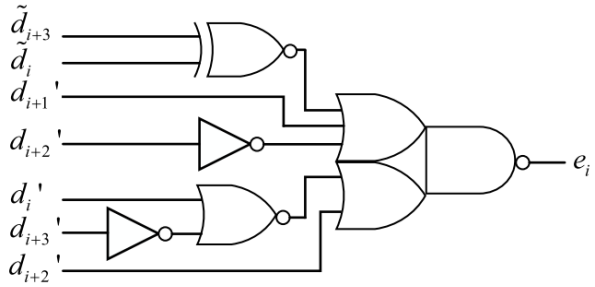
New Subfield Inversion

- Express in closed-form equations
- Derive 12 equivalent functions using Karnough maps, and optimize by-hand.
- Optimized using CAD tools.





Lightweight and fast, optimized by-hand
Used NAND3 gates

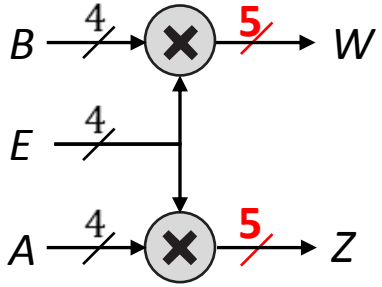


Optimized by CAD tools
Used OR-AND-Invert gates

	Area (GEs)	Delay (ns)
Lightweight and fast (optimized by-hand)	36	0.121
Optimized by CAD tools	**31**	**0.102**

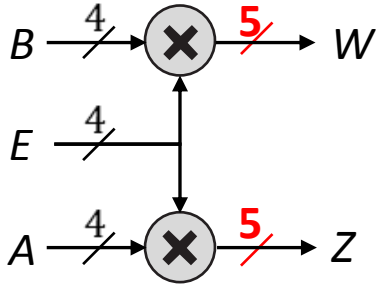
New Output Multipliers

- Two multipliers with a common input:
  $W = B \times E$ & $Z = A \times E$



New Output Multipliers

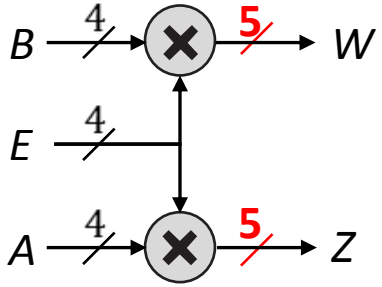
- Two multipliers with a common input:
  $W = B \times E$ & $Z = A \times E$



- Input and output terms represented as
  4 bits x 4 bits → 5 bits
  Reduction from 5 bits back to 4 bits is part of T_{out} .

New Output Multipliers

- Two multipliers with a common input:
  $W = B \times E$ & $Z = A \times E$

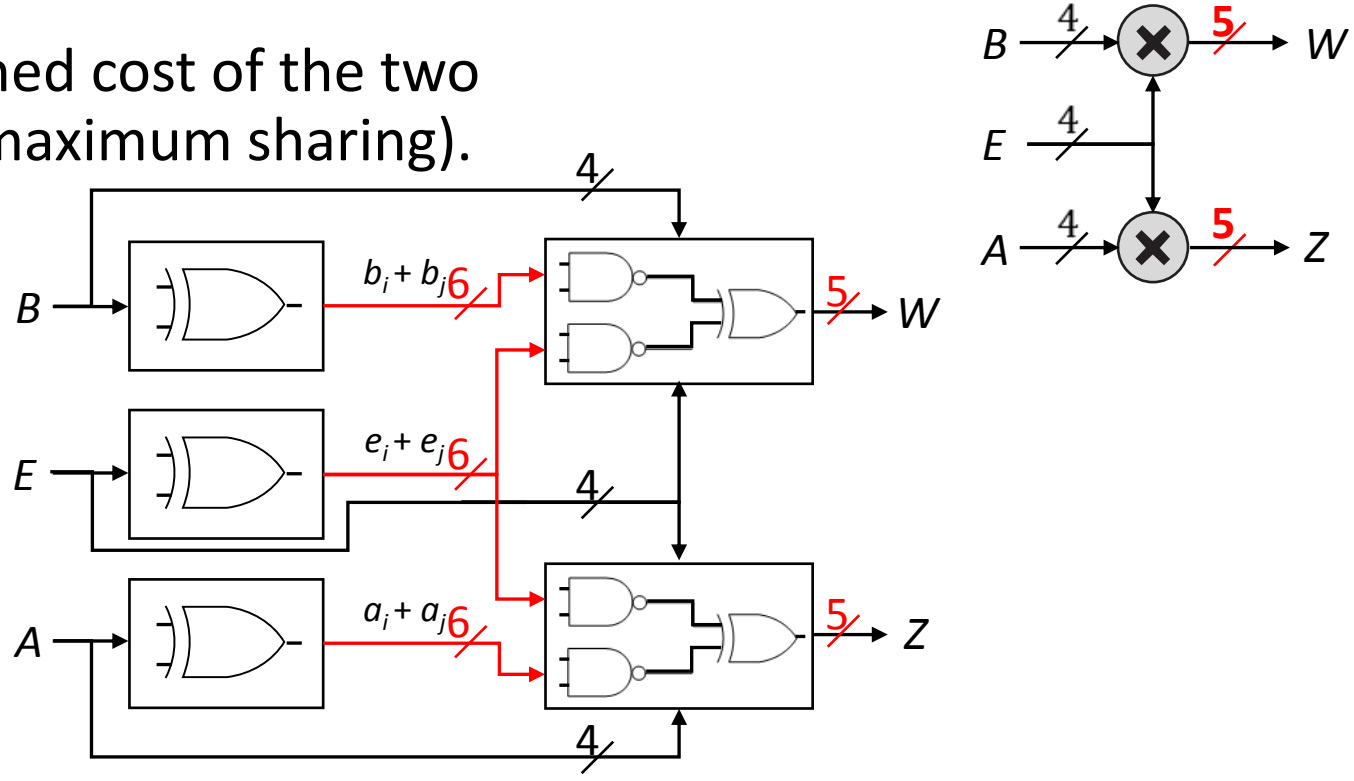


- Input and output terms represented as
  4 bits x 4 bits → 5 bits
  Reduction from 5 bits back to 4 bits is part of T_{out} .

- Previous work:
  4x4 → 4 [Can05b], 5x5 → 5 [NNI12], 4x5 → 5 [UHS+15]

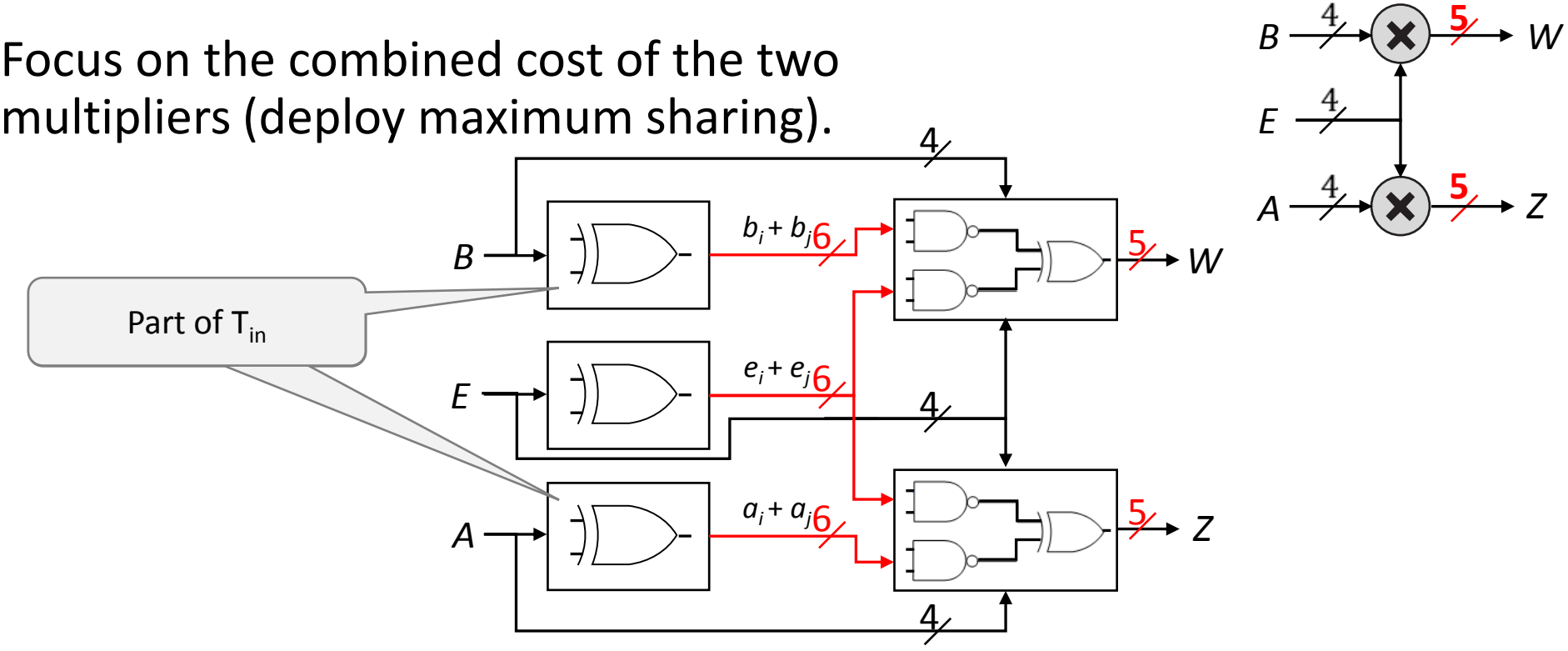
New Output Multipliers (cont.)

- Focus on the combined cost of the two multipliers (deploy maximum sharing).



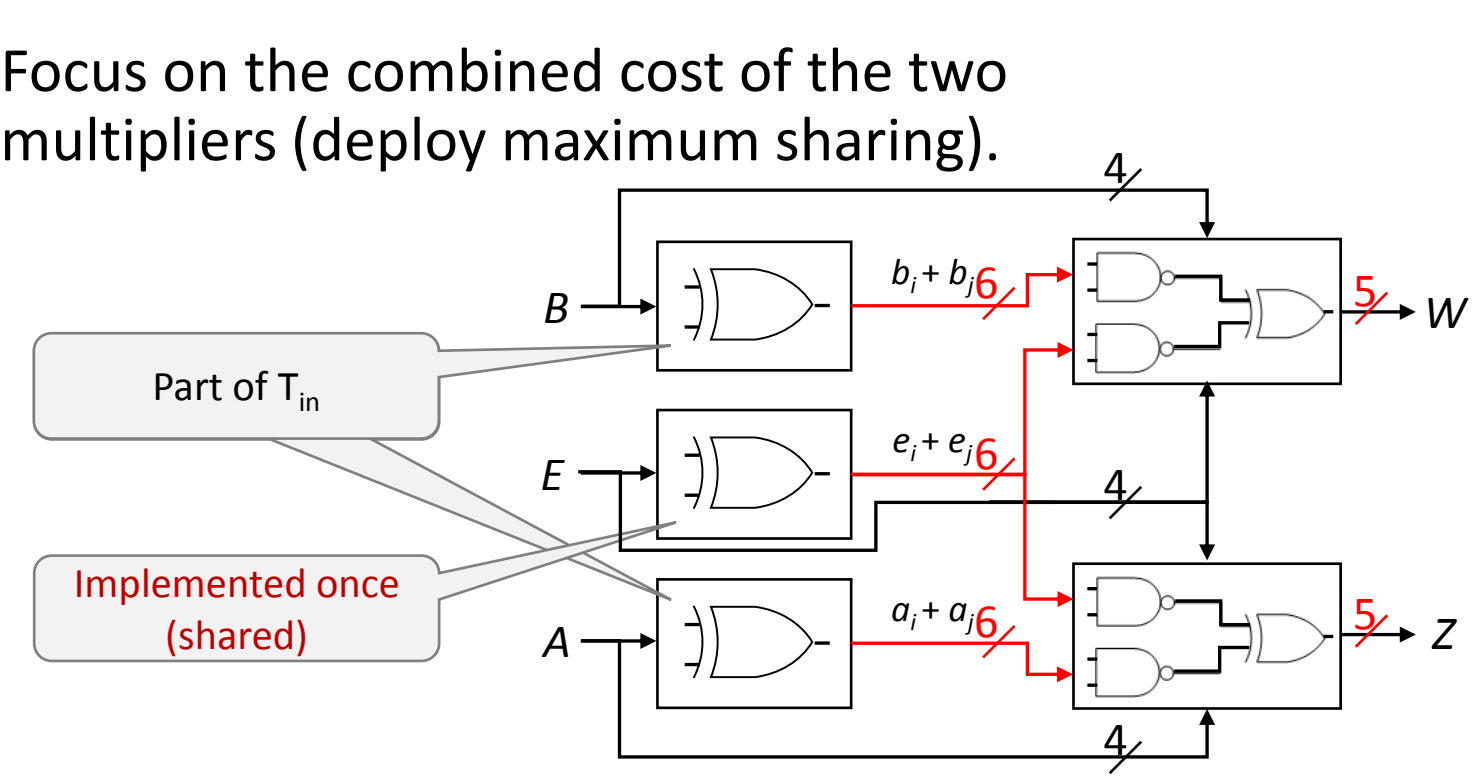
New Output Multipliers (cont.)

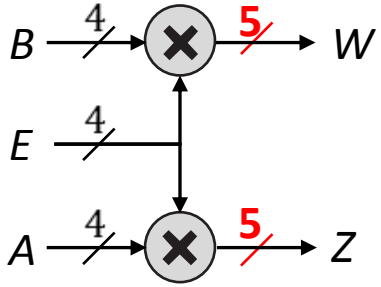
- Focus on the combined cost of the two multipliers (deploy maximum sharing).



New Output Multipliers (cont.)

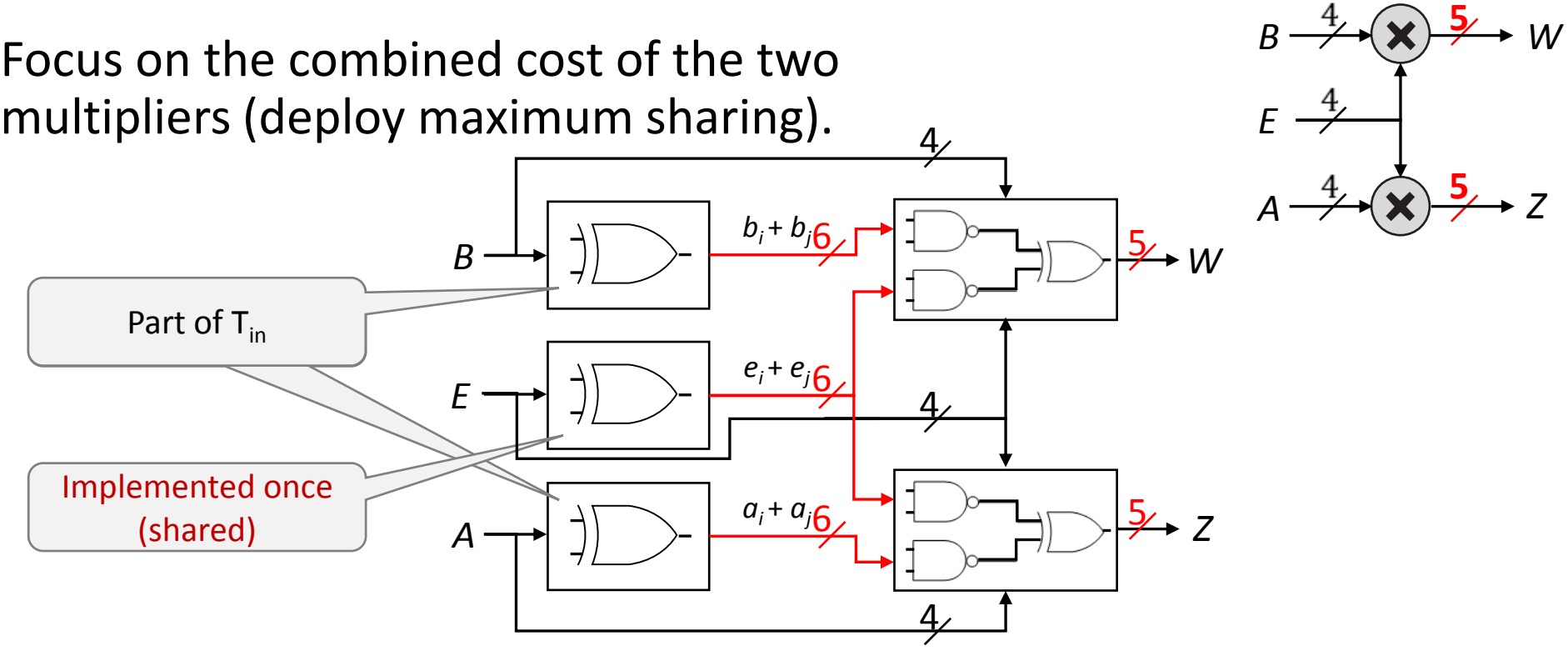
- Focus on the combined cost of the two multipliers (deploy maximum sharing).





New Output Multipliers (cont.)

- Focus on the combined cost of the two multipliers (deploy maximum sharing).



- Some multipliers do not allow sharing ([Mas91], [RDJ+01] and [GM16]).

New Output Multipliers (cont.)

Space and time complexities of a single multiplier

	Multiplier used in	Space Complexity	Time Complexity
$GF(((2^2)^2)^2)$	Satoh et al. [SMTM01]	21 XOR + 9 AND	$4\ D_X + D_{AD}$
	Canright [Can05b]	20 XOR + 9 NAND	$4\ D_X + D_{ND}$
	Nogami et al. [NNT+10]	21 XOR + 9 AND	$4\ D_X + D_{AD}$
$GF((2^4)^2)$	Rudra et al. [RDJ+01]	15 XOR + 16 AND	$3\ D_X + D_{AD}$
	Gueron et al. [GM16]	15 XOR + 16 AND	$3\ D_X + D_{ND}$
	Nekado et al. [NNI12]	25 XOR + 10 AND	$2\ D_X + D_{AD}$
	Ueno et al. [UHS+15]	21 XOR + 10 AND	$2\ D_X + D_{AD}$
	This work	17 XOR + 10 NAND	$2\ D_X + D_{ND}$

New Output Multipliers (cont.)

Space and time complexities of a single multiplier

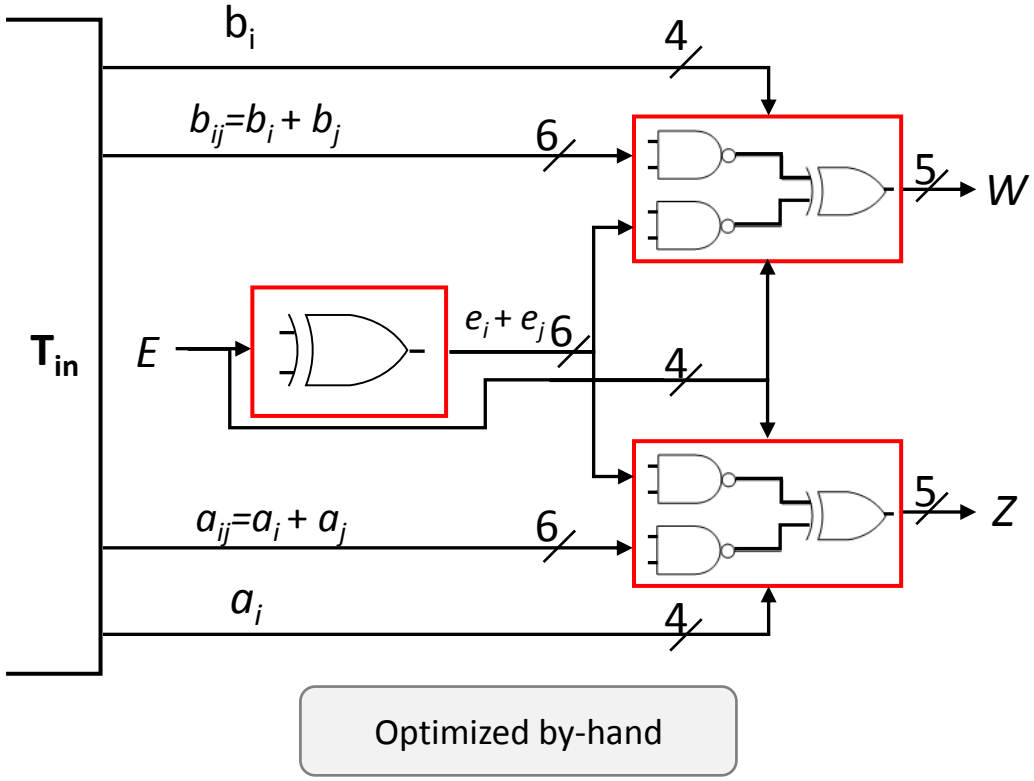
	Multiplier used in	Space Complexity	Time Complexity
$GF(((2^2)^2)^2)$	Satoh et al. [SMTM01]	21 XOR + 9 AND	$4\ D_X + D_{AD}$
	Canright [Can05b]	20 XOR + 9 NAND	$4\ D_X + D_{ND}$
	Nogami et al. [NNT+10]	21 XOR + 9 AND	$4\ D_X + D_{AD}$
$GF((2^4)^2)$	Rudra et al. [RDJ+01]	15 XOR + 16 AND	$3\ D_X + D_{AD}$
	Gueron et al. [GM16]	15 XOR + 16 AND	$3\ D_X + D_{ND}$
	Nekado et al. [NNI12]	25 XOR + 10 AND	$2\ D_X + D_{AD}$
	Ueno et al. [UHS+15]	21 XOR + 10 AND	$2\ D_X + D_{AD}$
	This work	17 XOR + 10 NAND	$2\ D_X + D_{ND}$

The smallest and fastest 4-bit multiplier to date among all the $GF((2^4)^2)$ and $GF(((2^2)^2)^2)$ multipliers

New Output Multipliers (cont.)

Additional area and delay required for the multipliers

	Area (GEs)	Delay (ns)
Optimized by-hand	52	0.099
Optimized by CAD tools	**53.5**	**0.121**



Outline

- Introduction, Motivation and Previous Work.
- Architecture of the Proposed AES S-box.
- New Logic-Minimization Algorithms.
- New GF($(2^4)^2$) Inversion.
  - New Exponentiation Stage.
  - New Representation of Subfield Inversion.
  - New Output Multipliers.
- Comparisons and Concluding Remarks.

Comparisons

- Targeting Lightweight Implementation

S-box	Area (GEs)	Delay (ns)	Area-Time Product
Canright [Can05b]	200	1.25	250
Improved 113-gates	194	1.35	261.9
This work (Lightweight)	**182.25**	**1.20**	**218.7**

The smallest, fastest and most efficient Lightweight S-box

Comparisons

- Targeting Lightweight Implementation

S-box	Area (GEs)	Delay (ns)	Area-Time Product
Canright [Can05b]	200	1.25	250
Improved 113-gates	194	1.35	261.9
This work (Lightweight)	**182.25**	**1.20**	**218.7**

The smallest, fastest and most efficient Lightweight S-box

- Targeting Fast Implementation

S-box	Area (GEs)	Delay (ns)	Area-Time Product
Improved Depth-16 (2012)	222	0.91	202.02
Improved Depth-16 (2017)	216	0.91	196.56
Improved Ueno et al.	238	**0.77**	183.26
This work (Fast)	**208**	**0.78**	**162.24**

At STM 65-nm CMOS standard technology library

The smallest, fastest and most efficient Fast S-box

Comparisons

- Targeting Lightweight Implementation

S-box	Area (GEs)	Delay (ns)	Area-Time Product
Canright [Can05b]	200	1.25	250
Improved 113-gates	194	1.35	261.9
This work (Lightweight)	**182.25**	**1.20**	**218.7**

The smallest, fastest and most efficient Lightweight S-box

- Targeting Fast Implementation

S-box	Area (GEs)	Delay (ns)	Area-Time Product
Improved Depth-16 (2012)	222	0.91	202.02
Improved Depth-16 (2017)	216	0.91	196.56
Improved Ueno et al.	238	**0.77**	183.26
This work (Fast)	**208**	**0.78**	**162.24**

At STM 65-nm CMOS standard technology library

The smallest, fastest and most efficient Fast S-box

As compared against the improved versions proposed in this paper

As a result of testing more than 46 pieces of VHDL code, at various abstraction levels of the designs

Effect of Target Library

- Industrial technology libraries (e.g., STM and TSMC):
  - Lightweight: Used XOR3 and OAI32 → 182.25 GEs.
  - Fast: Used NAND3 → 208 GEs.

Effect of Target Library

- Industrial technology libraries (e.g., STM and TSMC):
  - Lightweight: Used XOR3 and OAI32 → 182.25 GEs.
  - Fast: Used NAND3 → 208 GEs.
- NanGate45nm:
  - Lightweight: Used AOI12 and OAI12 gates → 186 GEs.
  - Fast: Used NAND3 → 208 GEs (no change).

Effect of Target Library

- Industrial technology libraries (e.g., STM and TSMC):
  - Lightweight: Used XOR3 and OAI32 → 182.25 GEs.
  - Fast: Used NAND3 → 208 GEs.
- NanGate45nm:
  - Lightweight: Used AOI12 and OAI12 gates → 186 GEs.
  - Fast: Used NAND3 → 208 GEs (no change).
- Without using any compound gate:
  - Lightweight: 191 GEs (best previous work: 194 GEs)
  - Fast: 211 GEs (best previous work: 216 GEs)

Effect of Target Library

- Industrial technology libraries (e.g., STM and TSMC):
  - Lightweight: Used XOR3 and OAI32 → 182.25 GEs.
  - Fast: Used NAND3 → 208 GEs.
- NanGate45nm:
  - Lightweight: Used AOI12 and OAI12 gates → 186 GEs.
  - Fast: Used NAND3 → 208 GEs (no change).
- Without using any compound gate:
  - Lightweight: 191 GEs (best previous work: 194 GEs)
  - Fast: 211 GEs (best previous work: 216 GEs)

The proposed designs are superior under any restriction by the target library.

Concluding Remarks

- In this paper, we proposed:
  - Two new designs for the AES S-box: Lightweight and fast.
  - New logic-minimization heuristics.
  - New formulations for each stage of the S-box.
  - New output multipliers.
  - Design methodology for an optimum synergy between theoretical analysis and technology-assisted CAD tools.

References

- [Can05b] David Canright. A very compact S-box for AES. CHES-2005.
- [Boy16] CMT: Circuit minimization team, 2016. http://www.cs.yale.edu/homes/peralta/CircuitStuff/CMT.html,
- [BP12] Joan Boyar and René Peralta. A small depth-16 circuit for the AES S-box. Information Security and Privacy Conference, SEC 2012.
- [BFP17] Joan Boyar, Magnus Find, and René Peralta. Low-depth, low-size circuits for cryptographic applications. In Boolean Functions and their Applications BFA-2017.
- [UHS+15] Rei Ueno, Naofumi Homma, Yukihiro Sugawara, Yasuyuki Nogami, and Takafumi Aoki. Highly efficient GF(2^8) inversion circuit based on redundant GF arithmetic and its application to AES design. CHES-2015.
- [BMP08] Joan Boyar, Philip Matthews, and René Peralta. On the shortest linear straight-line program for computing linear forms. Mathematical Foundations of Computer Science, MFCS 2008.
- [Paa94] Christof Paar. Efficient VLSI architectures for bit parallel computation in Galios fields. PhD thesis, University of Duisburg-Essen, Germany, 1994.
- [BP10] Joan Boyar and René Peralta. A new combinational logic minimization technique with applications to cryptology. Symposium on Experimental Algorithms, SEA 2010.
- [NNI12] Kenta Nekado, Yasuyuki Nogami, and Kengo Iokibe. Very short critical path implementation of AES with direct logic gates. International Workshop on Security, IWSEC 2012.
- [Mas91] E. D. Mastrovito. VLSI Architectures for Computation in Galois Fields. PhD thesis, Linkoping Univ., Linkoping Sweden, 1991.
- [RDJ+01] Atri Rudra, Pradeep K. Dubey, Charanjit S. Jutla, Vijay Kumar, Josyula R.Rao, and Pankaj Rohatgi. Efficient Rijndael encryption implementation with composite field arithmetic. CHES 2001.
- [GM16] Shay Gueron and Sanu Mathew. Hardware implementation of AES using area-optimal polynomials for composite-field representation GF($(2^4)^2$) of GF(2^8). ARITH 2016.
- [SMTM01] Akashi Satoh, Sumio Morioka, Kohji Takano, and Seiji Munetoh. A compact Rijndael hardware architecture with S-box optimization. ASIACRYPT 2001.
- [NNT+10] Yasuyuki Nogami, Kenta Nekado, Tetsumi Toyota, Naoto Hongo, and Yoshitaka Morikawa. Mixed bases for efficient inversion in F($(2^2)^2)^2$ and conversion matrices of subbytes of AES. CHES-2010.

Thank You,
Questions?

Logic-Minimization Algorithms

Input and *Dist*, using original the inputs

w_0	3	s_7
w_1	7	s_6
w_2	7	s_5
w_3	5	s_4
w_4	3	s_3
z_0	3	s_2
z_1	1	s_1
z_2	5	s_0
z_3		
z_4		

First, add all gates with *Dist*=1

w_0	3	s_7
w_1	6	s_6
w_2	7	s_5
w_3	5	s_4
z_0	3	s_3
z_1	2	s_2
z_2	1	s_1
z_3	5	s_0
z_4		

$w_2 \oplus w_4$

Dist, assume using w_0+w_1

w_0	3	s_7
w_1	5	s_6
w_2	6	s_5
w_3	4	s_4
z_0	3	s_3
z_1	2	s_2
z_2	1	s_1
z_3	5	s_0
z_4		

$w_2 \oplus w_4$
$w_0 \oplus w_1$

Sum(*Dist*) = 29

Dist, assume using w_0+w_2

w_0	3	s_7
w_1	6	s_6
w_2	7	s_5
w_3	5	s_4
z_0	3	s_3
z_1	2	s_2
z_2	1	s_1
z_3	5	s_0
z_4		

$w_2 \oplus w_4$
$w_0 \oplus w_2$

Sum(*Dist*) = 32

Dist, assume using w_0+w_3

w_0	3	s_7
w_1	6	s_6
w_2	6	s_5
w_3	5	s_4
z_0	3	s_3
z_1	2	s_2
z_2	1	s_1
z_3	5	s_0
z_4		

$w_2 \oplus w_4$
$w_0 \oplus w_3$

Sum(*Dist*) = 31

Dist, assume using w_0+w_4

w_0	3	s_7
w_1	6	s_6
w_2	6	s_5
w_3	5	s_4
z_0	3	s_3
z_1	2	s_2
z_2	1	s_1
z_3	5	s_0
z_4		

$w_2 \oplus w_4$
$w_0 \oplus w_4$

Sum(*Dist*) = 31

T_{out}

$$\begin{bmatrix} s_7 \\ s_6 \\ s_5 \\ s_4 \\ s_3 \\ s_2 \\ s_1 \\ s_0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \\ z_0 \\ z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix}$$

- **Normal-BP:**
  1.Test all the possible XOR gates that can use the previous level gates (the inputs and (w_2+w_4)). That is: from (w_0+w_1) all the way to (z_4 + (w_2+w_4)).
  2.Select one gate that leads to [min (sum (*Dist*))]. In case of ties, select one gate based on different tie breaking criteria.
  For example, within the best gates, select one gate that maximizes the Euclidean norm of *Dist*
- **Improved-BP**:
  Similar to Normal-BP, but try all the tie, and monitor progress of the Delay.
- **Shortest-Dist-First**
  Similar to Norma-BP, but select all the gates that as many small numbers in the *Dist* as possible. If we consider the four cases above, we will select all of them because the smallest number is 2 (excluding ones), and this number (2) appears one time in each case. If it were to appear twice in any case, I would have selected that case. If the smallest number is 3, so that is the smallest *Dist*, and select the case that leads to as many (Dist=3) as possible.
- **Focused-Search**
  Similar to 'Shortest-Dist-First', but we ignore the count of (Dist=2) or (Dist=3). Here, we select all the gates that include (Dist=2) within the vector of Distances. We do not differentiate based on the count. If there is no gate that lead to Dist=2, select all the gates that include Dist=3, and so on.