Hardware Implementation of Block Cipher: Case Study Using AES

Tohoku University

Rei Ueno

Acknowledgments

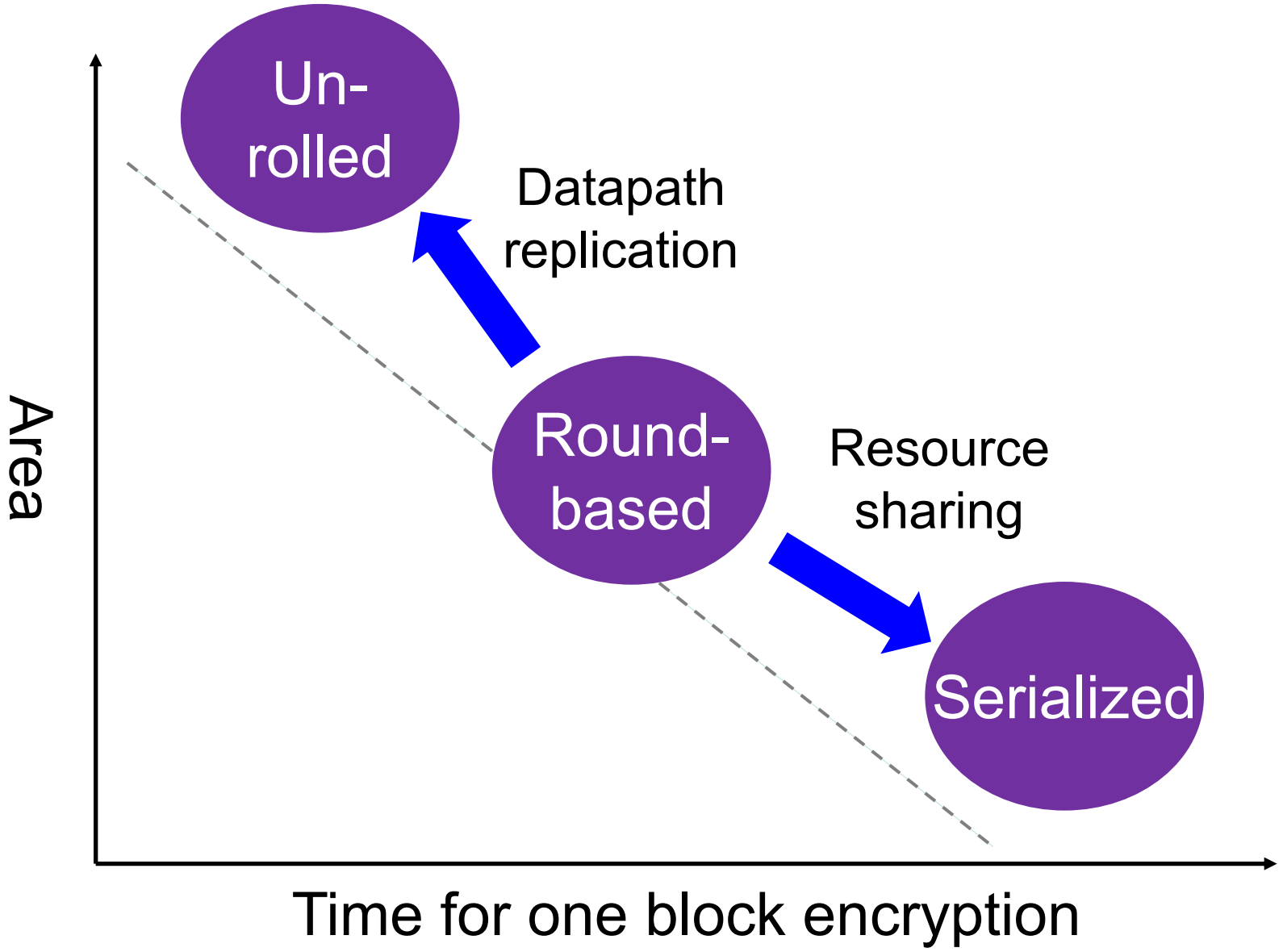
Naofumi Homma, *Tohoku Univ*.
Takafumi Aoki, *Tohoku Univ*.
Sumio Morioka, *Interstellar technologies, Inc*.
Noriyuki Miura, *Kobe Univ*.
Kohei Matsuda, *Kobe Univ*.
Makoto Nagata, *Kobe Univ*.
Shivam Bhasin, *NTU*
Yves Mathieu, *Telecom ParisTech*
Tarik Graba, *Telecom ParisTech*
Jean-Luc Danger, *Telecom ParisTech*

This talk

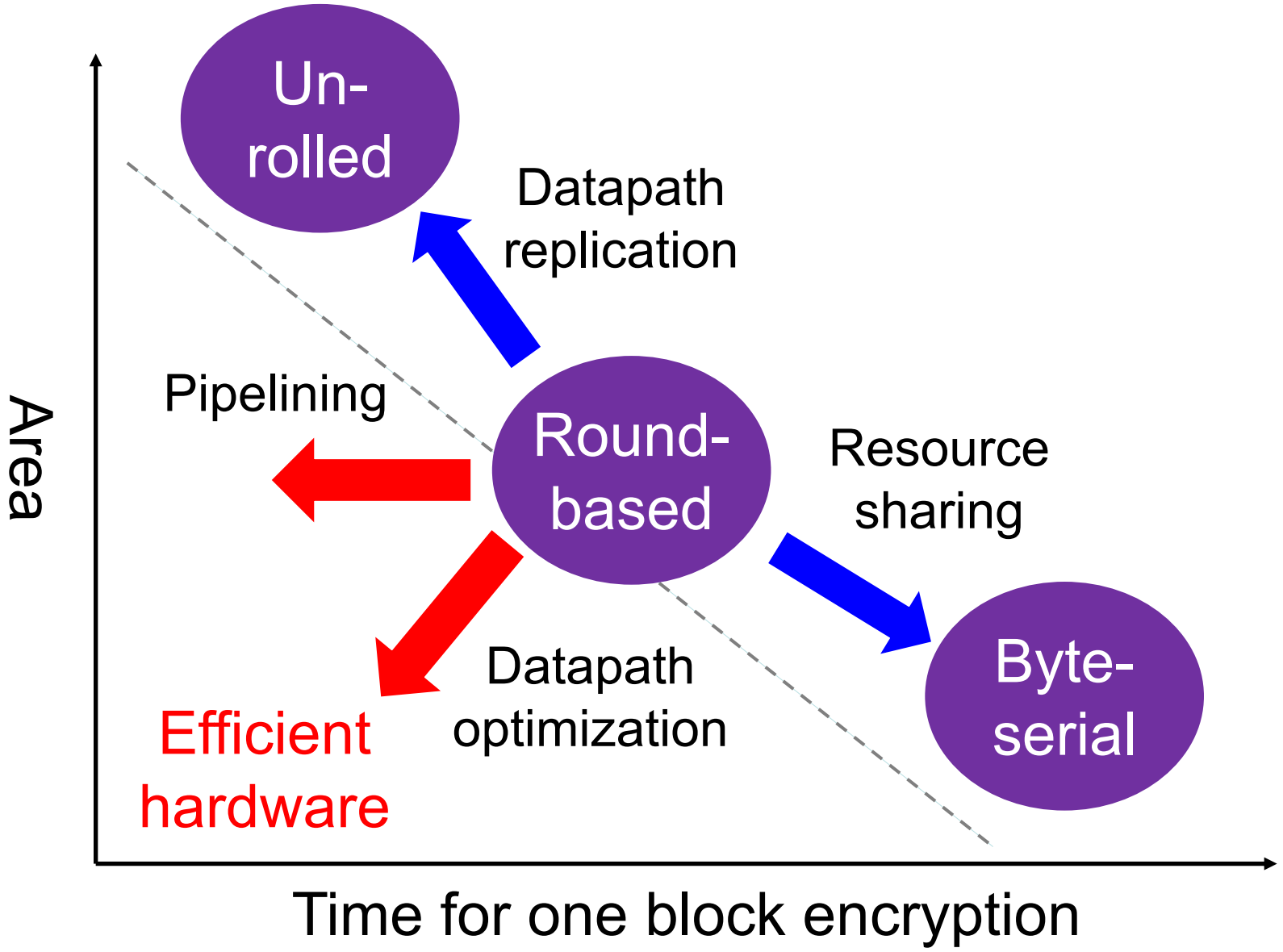
- Given a symmetric key cipher, how hardware designer implement and optimize it
  - For practical application:
    - With higher efficiency, encryption/decryption unified, on-the-fly key scheduling, without block-wise pipelining
  - Case study using AES!

- Disclaimer
  - Some modern lightweight ciphers are already optimized and they avoid some concerns in implementing AES
  - But I still believe that optimization of AES implementation can be feedbacked to cipher designs

Hardware architectures of block cipher



Hardware architectures of block cipher



For practical hardware implementation

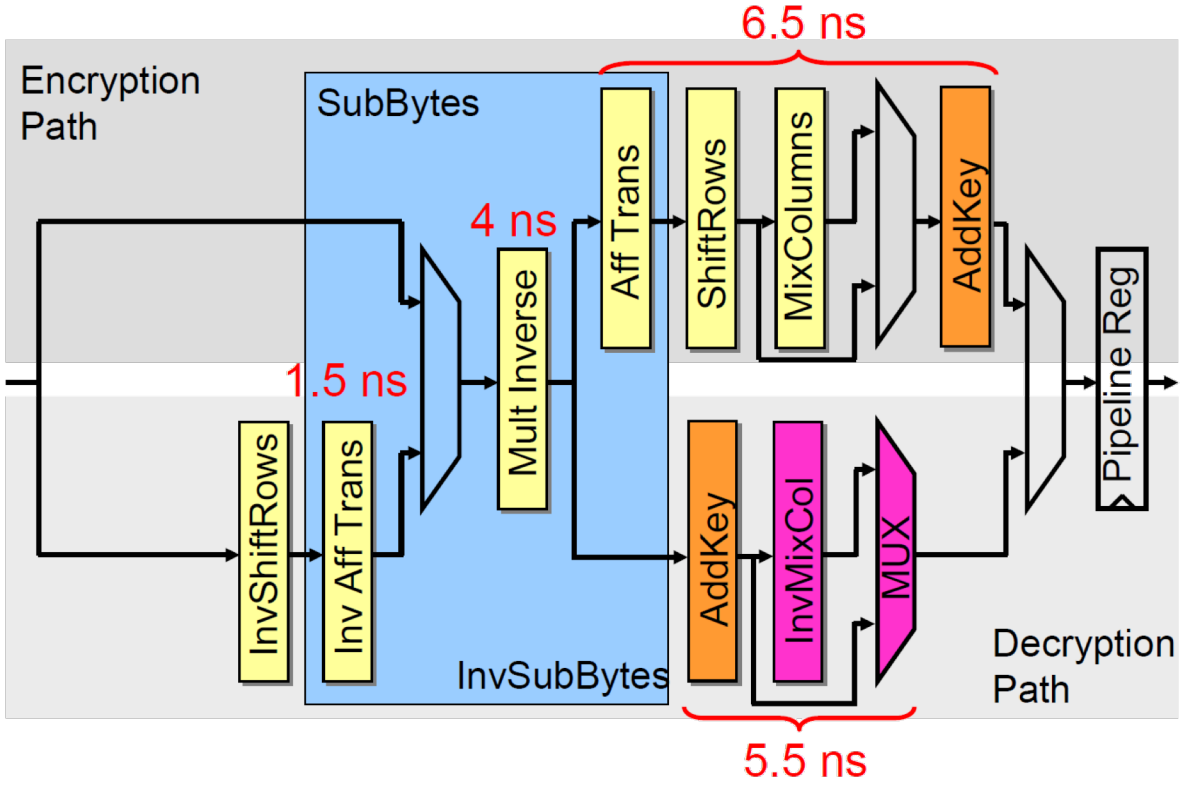
- Block-chaining modes have been widely deployed
  - CBC, CMAC, and CCM…
- (Un)Parallelizability: Issue on block-wise pipelining
  - AES hardware achieves 53Gbps, but works only for parallelizable modes [Mathew+ JSSC2011]
  - Higher throughput ≠ Lower latency
- Both encryption and decryption operations
- Importance of on-the-fly key scheduling
  - Off-the-fly key scheduling requires additional memories to store expanded keys
  - Latency for calculating round keys is nonnegligible if we use AES with key-tweakable modes

Outline

- Introduction
- Related works
- Optimized architecture
- Optimization of linear functions over tower-field
- Performance evaluation
- Concluding remarks

Conventional architecture 1/2 [Lutz+, CHES 2002]

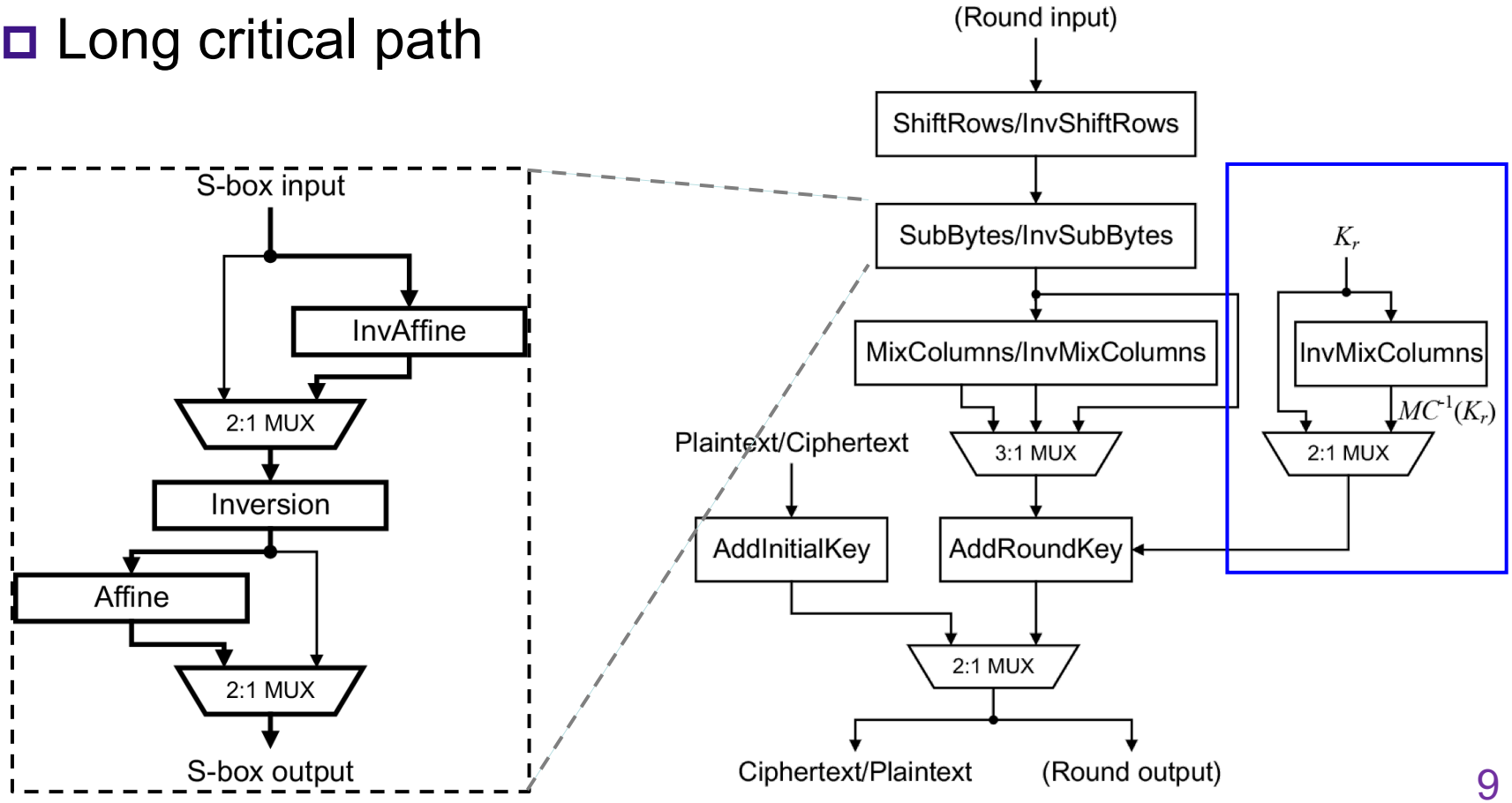
- Enc and Dec datapaths with additional selectors
  - Overhead of selectors for unification is nontrivial
  - False paths appear



www.chesworkshop.org/ches2002/presentations/Lutz.pdf

Conventional architecture 2/2 [Satoh+, AC 2001]

- Unify each pair of operation and its inverse
  - RoundKey requires InvMixColumns
  - Some MUXs in unified operations
  - Long critical path



Tower-field implementation

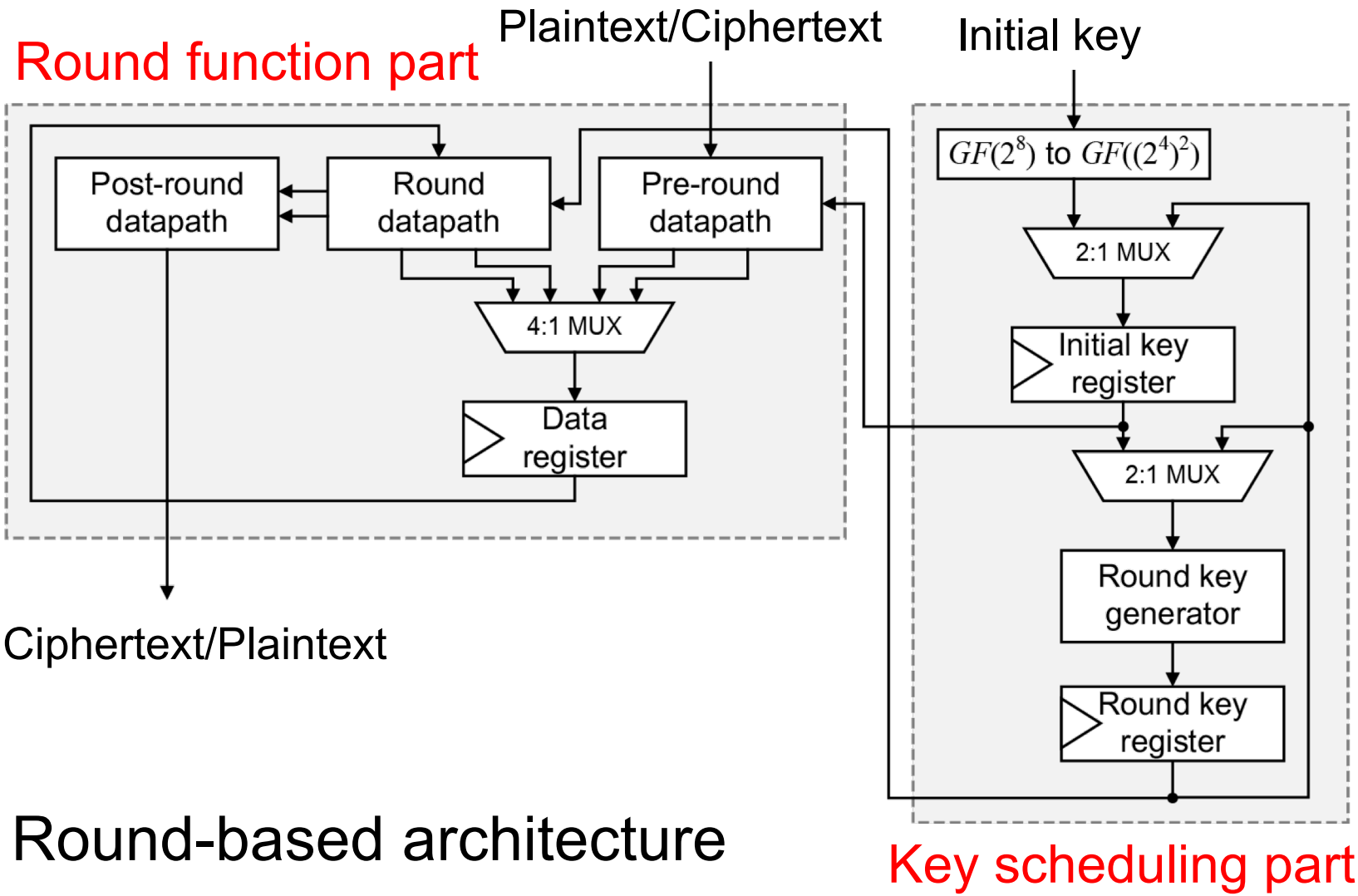
- Inversion should be performed over tower-field
  - Tower-field inversion is more efficient than direct mapping (e.g., table-lookup)
- Two types of tower-field implementation
  - Type-I: only inversion is performed over tower-field
  - Type-II: all operations are performed over tower-field

	Inversion (S-box)	MixColumns InvMixColumns
Type-I	Good	Good
Type-II	Better	Bad

Outline

- Introduction
- Related works
- Optimized architecture
- Optimization of linear functions over tower-field
- Performance evaluation
- Concluding remarks

Overall architecture

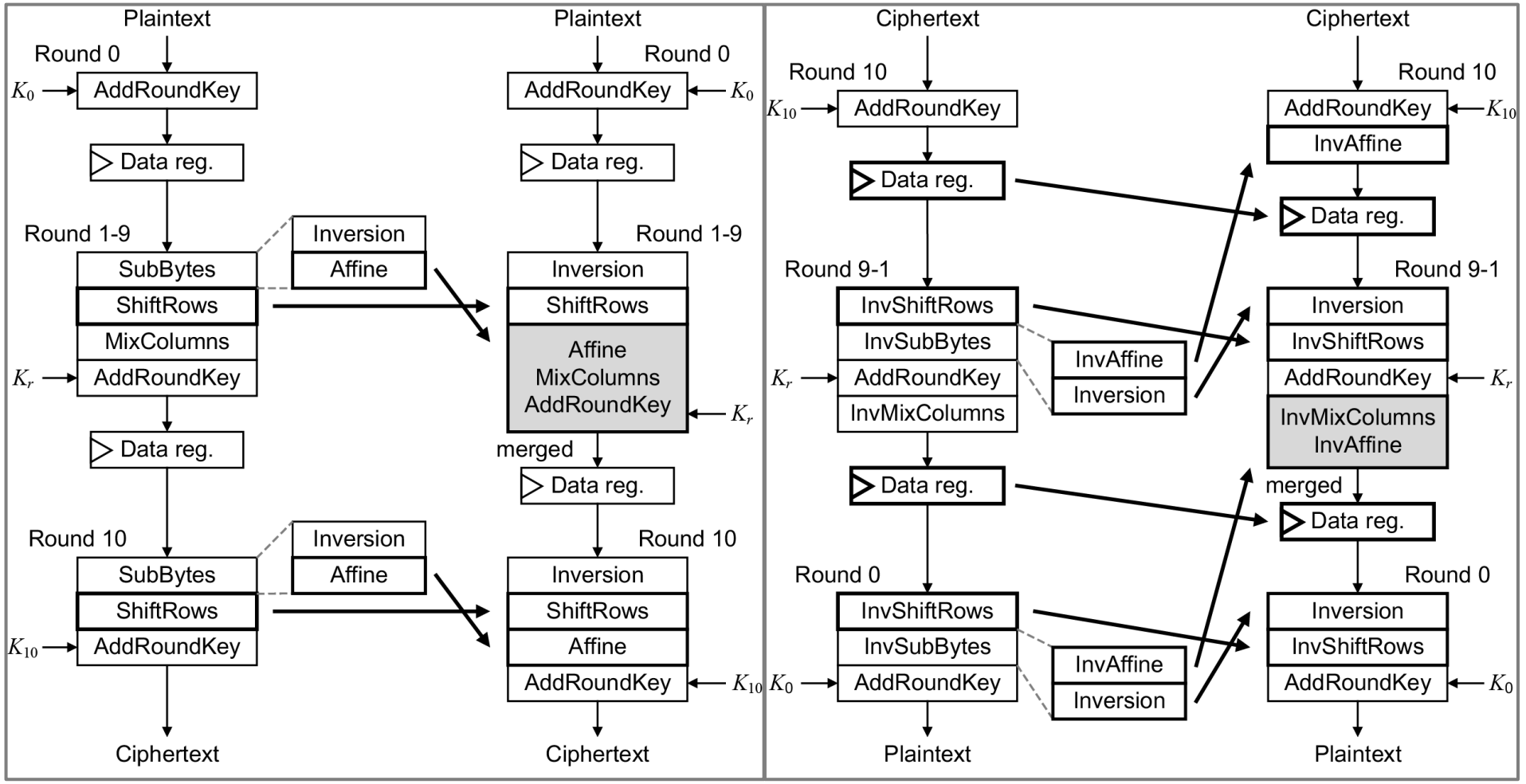


- Round-based architecture
- On-the-fly key scheduler

Round function part

- Compress encryption and decryption datapaths by register-retiming and operation-reordering
  - Unify inversion circuits in encryption and decryption
    - Without any additional selectors (i.e., overheads)
  - Merge linear operations to reduce gates and critical delay
    - Affine/InvAffine and MixColumns/InvMixColumns
    - At most one linear operation for a round
- Type-II tower-field implementation
  - Isomorphic mappings are performed at data I/O
  - Lower-area tower-field (Inv)Affine and (Inv)MixColumns

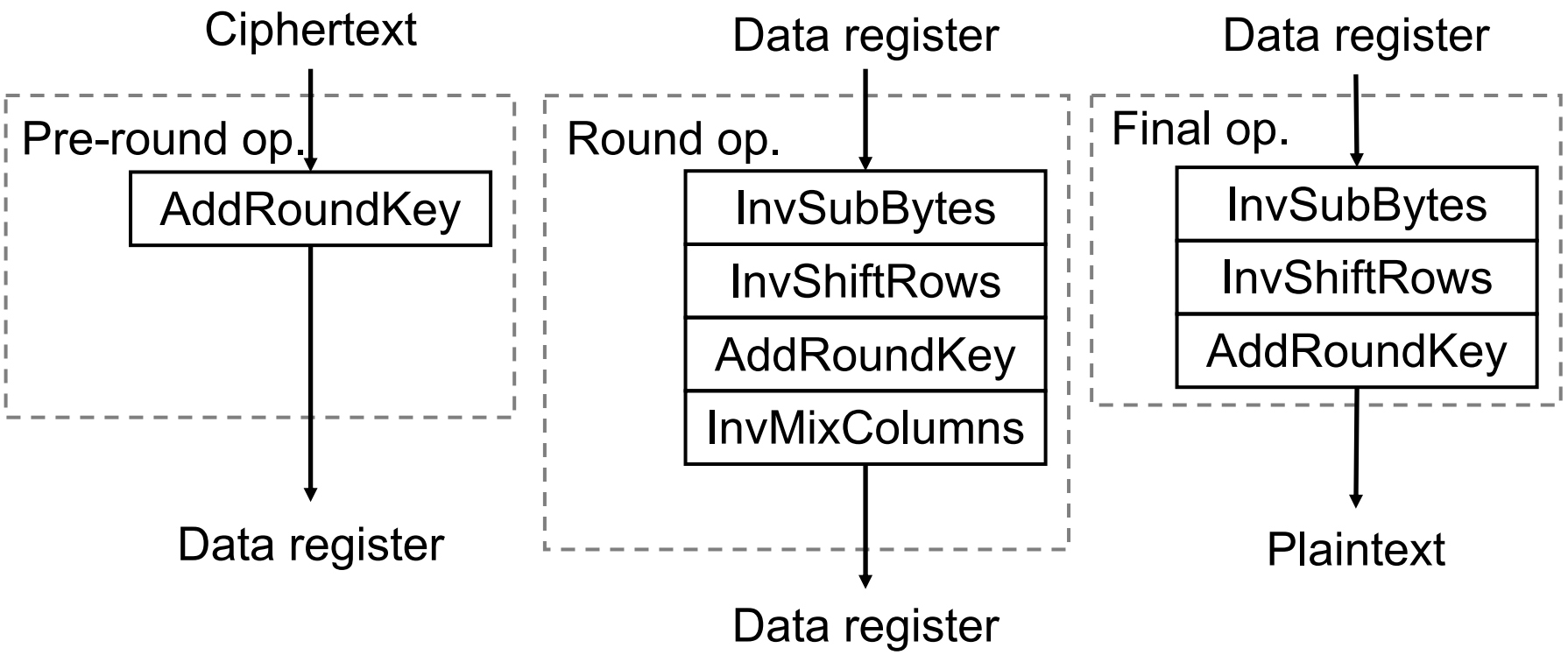
Resister-retiming and operation-reordering



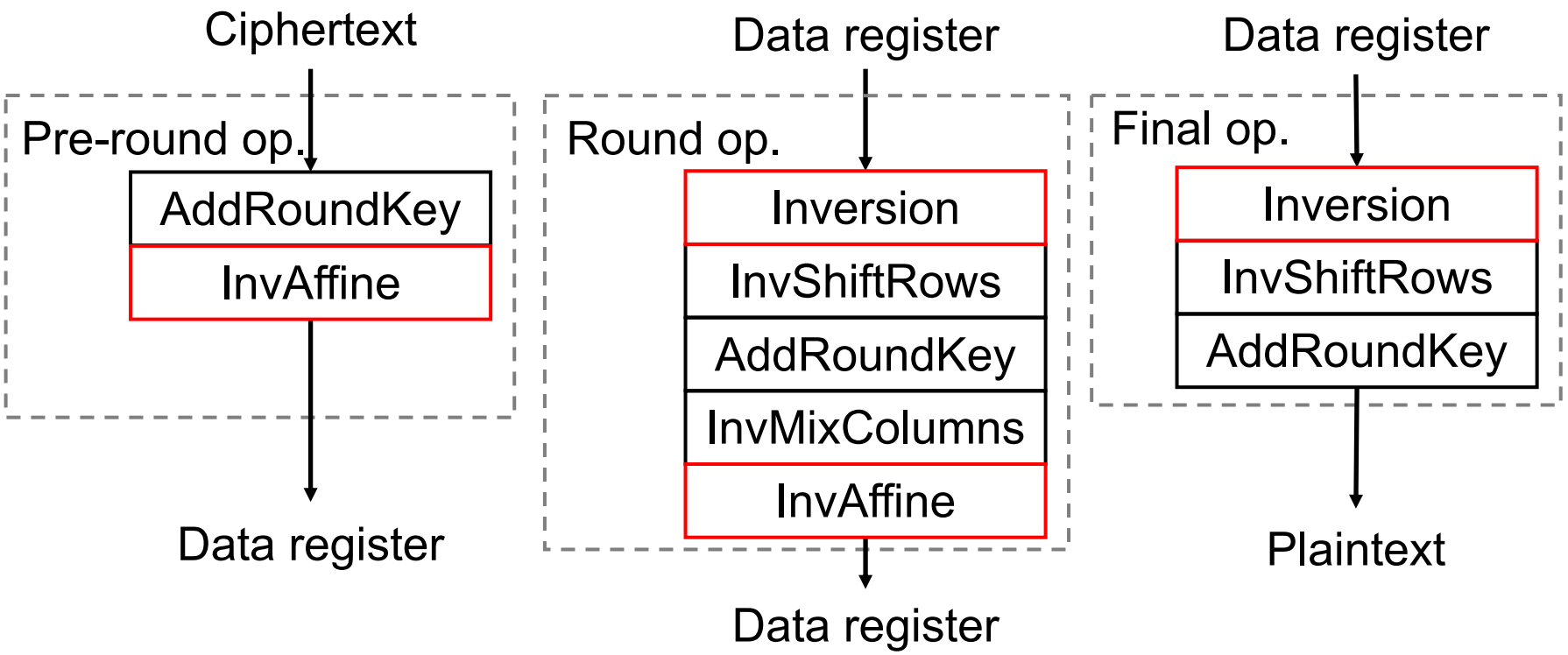
Original Proposed Original Proposed

Encryption Decryption

Key tricks (of decryption)

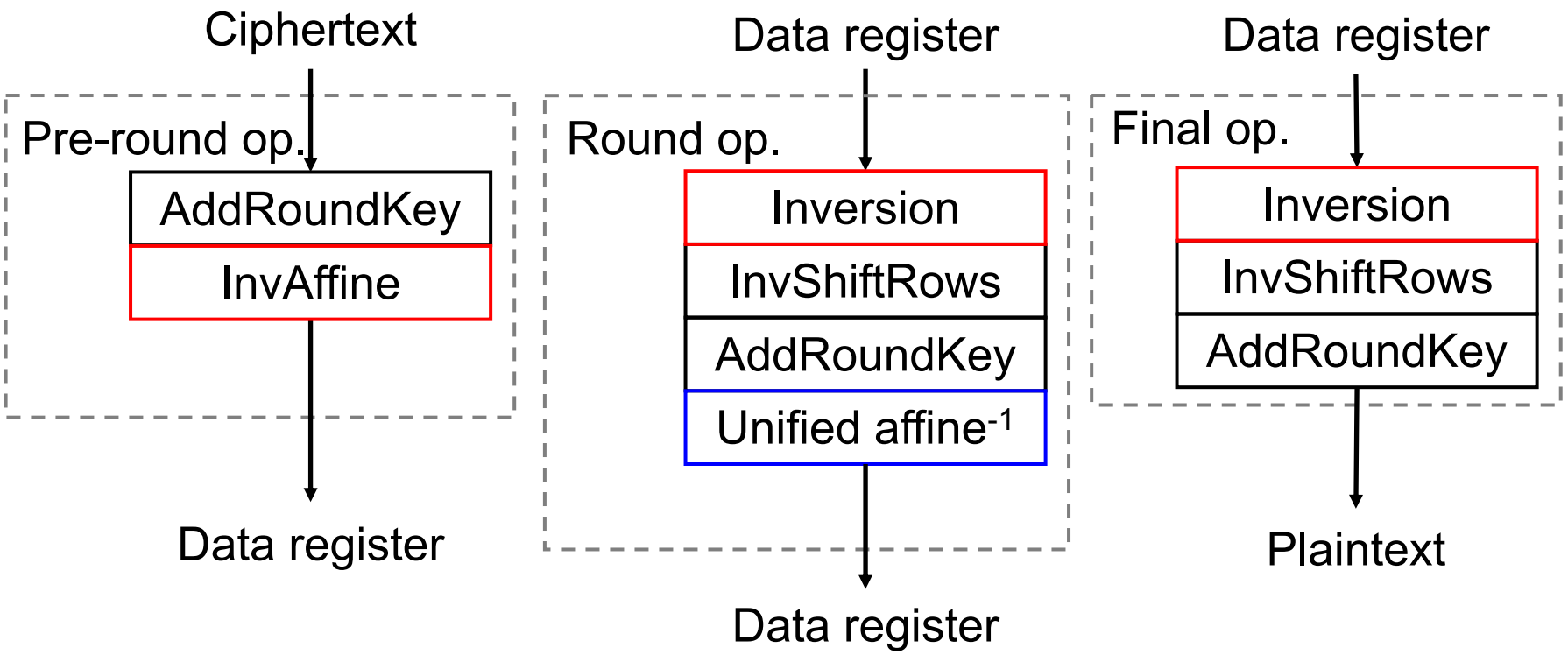


Key tricks (of decryption)



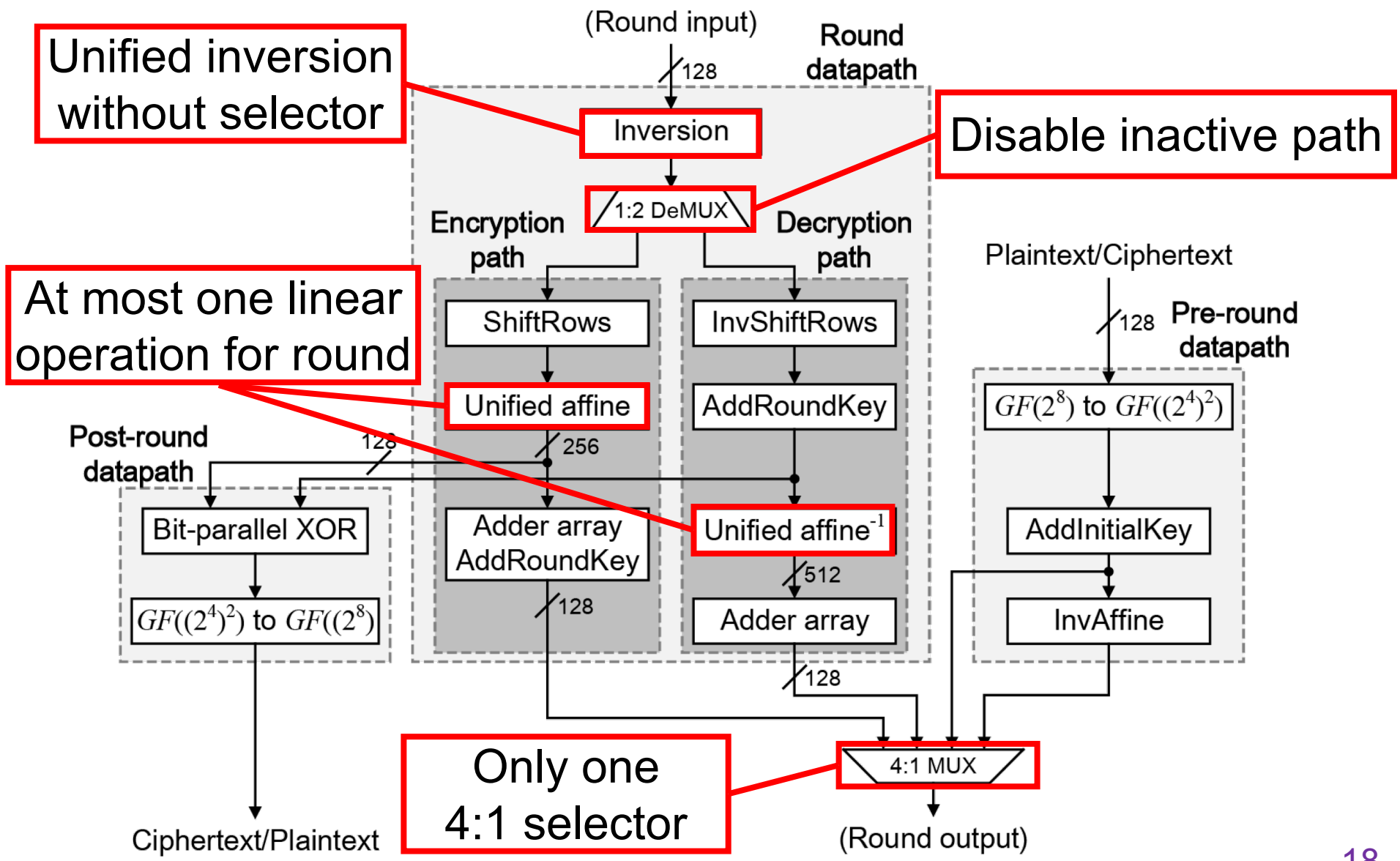
- Decompose InvSubByte to InvAffine and Inversion
- Register-retiming to initially perform inversion in round operations

Key tricks (of decryption)

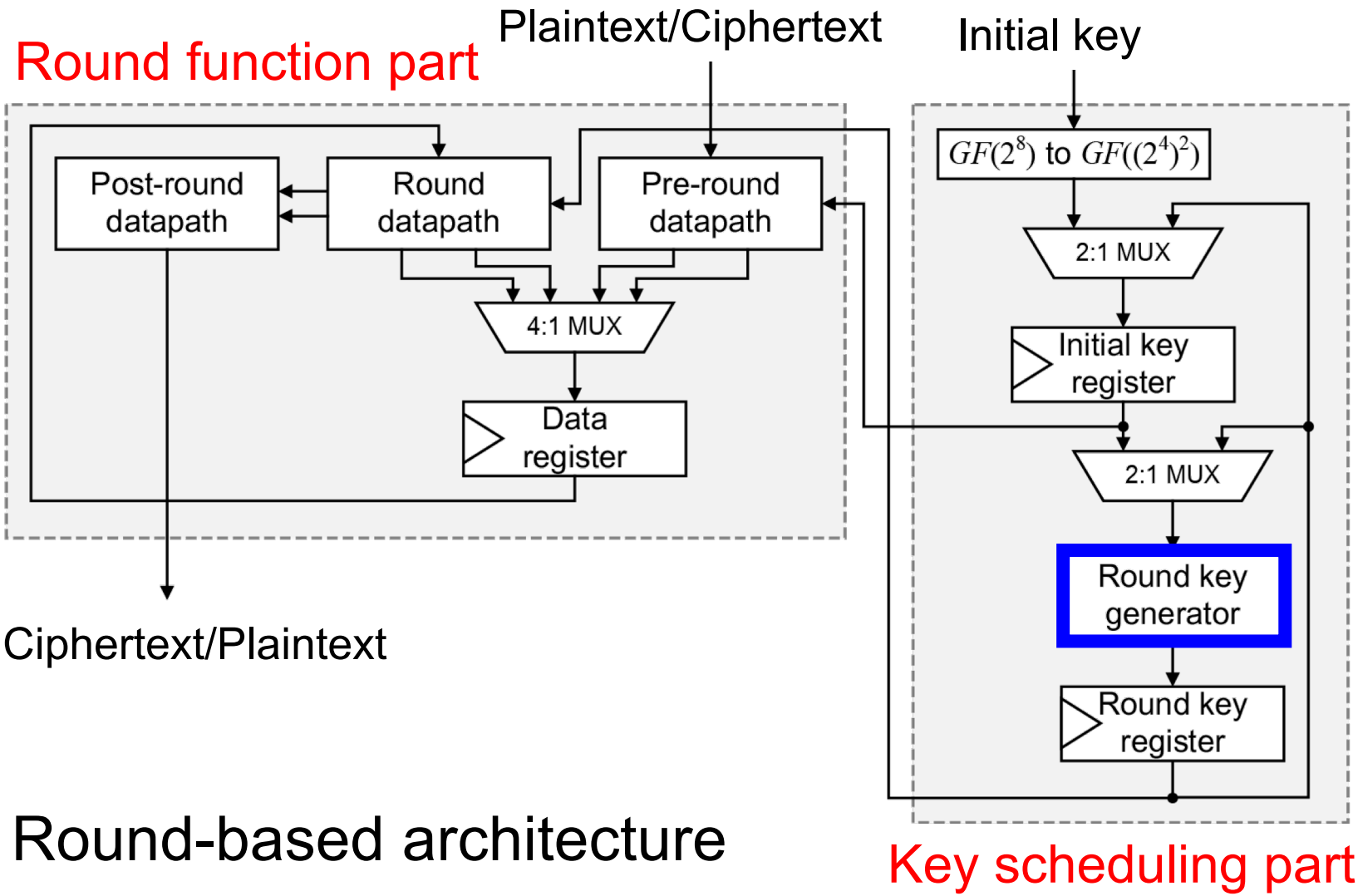


- Merge linear operations as Unified affine^{-1}
  - InvAffine and InvMixColumns
- Distinct AddRoundKey to avoid additional selectors or InvMixColumns for RoundKey

Resulting datapath



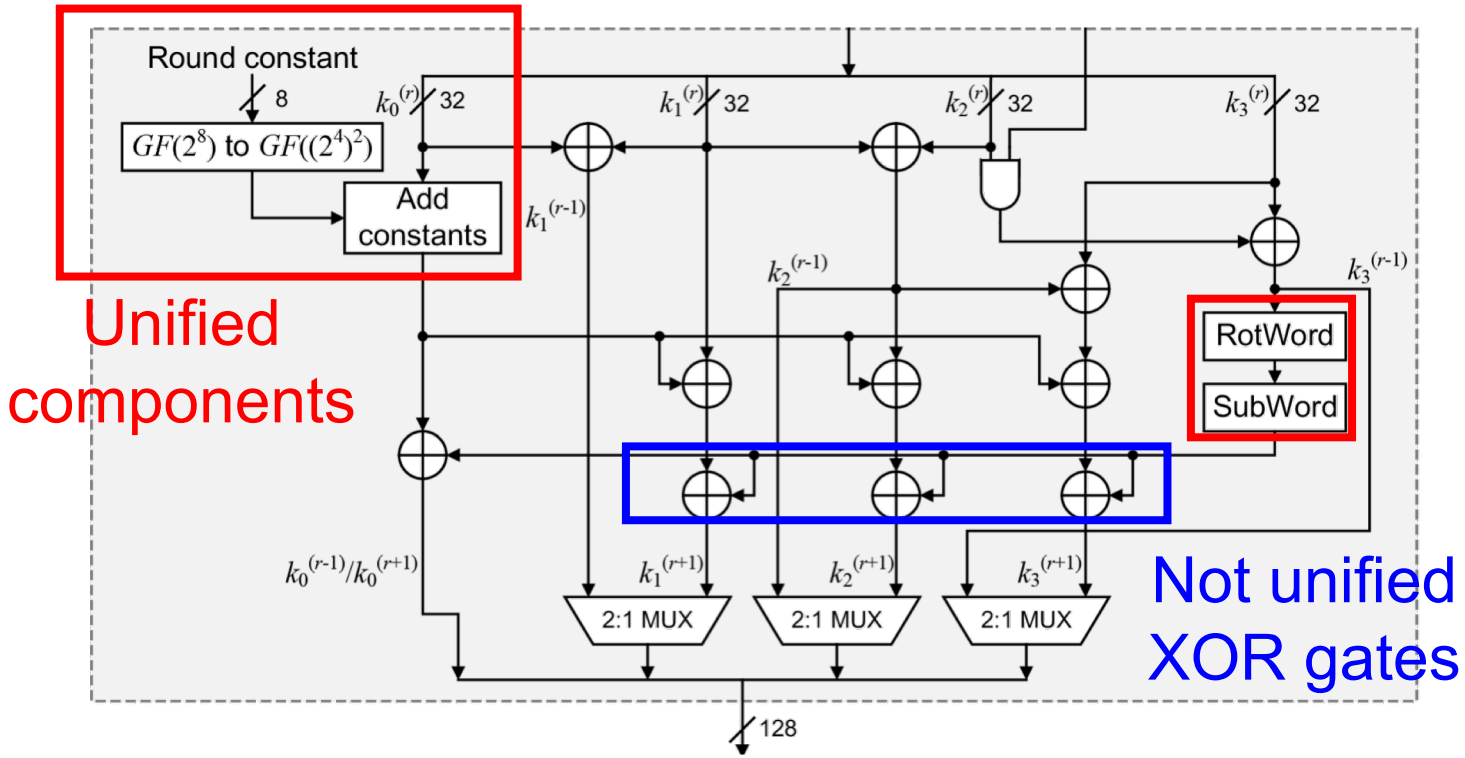
Overall architecture



- Round-based architecture
- On-the-fly key scheduler

Key scheduling part

- Round key generator is dominant
  - Unify encryption and decryption datapaths
  - Shorten critical delay than round function part by NOT unifying some XOR gates

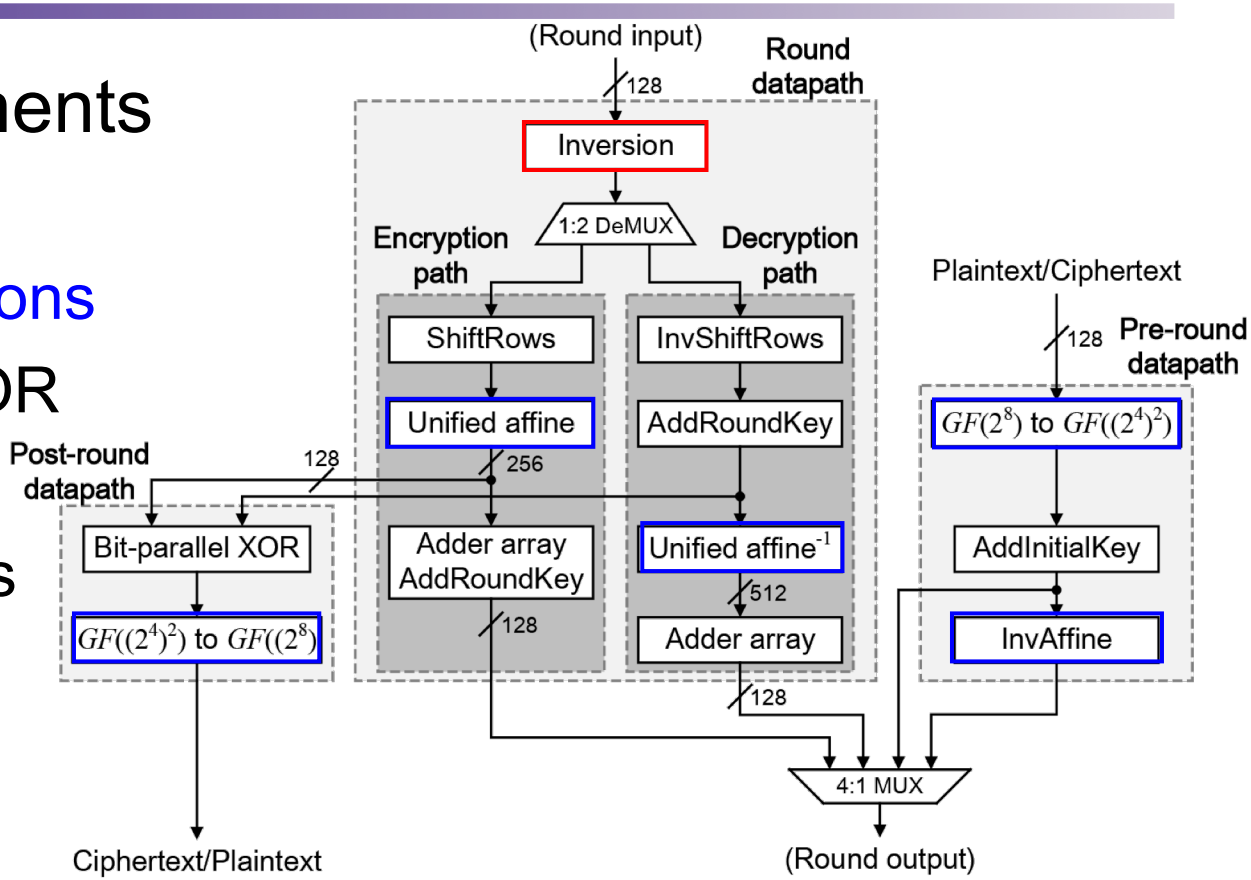


Outline

- Introduction
- Related works
- Optimized architecture
- Optimization of linear functions over tower-field
- Performance evaluation
- Concluding remarks

Coming back to round function part

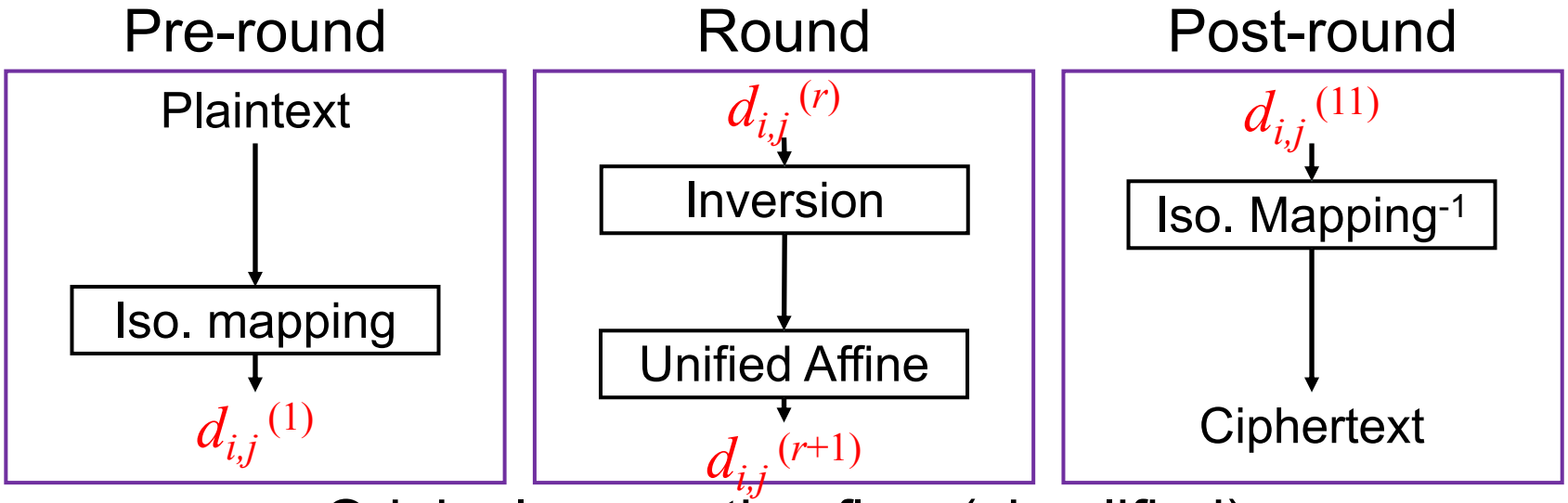
- Major components
  - Inversion
  - Linear operations
  - Bit-parallel XOR
  - Selectors
  - (Inv)ShiftRows



- Performance depends on constructions of inversion and linear operations
  - Inversion: Use state-of-the-art adoptable one
  - Linear operations: Depends on XOR matrices

Multiplicative-offset

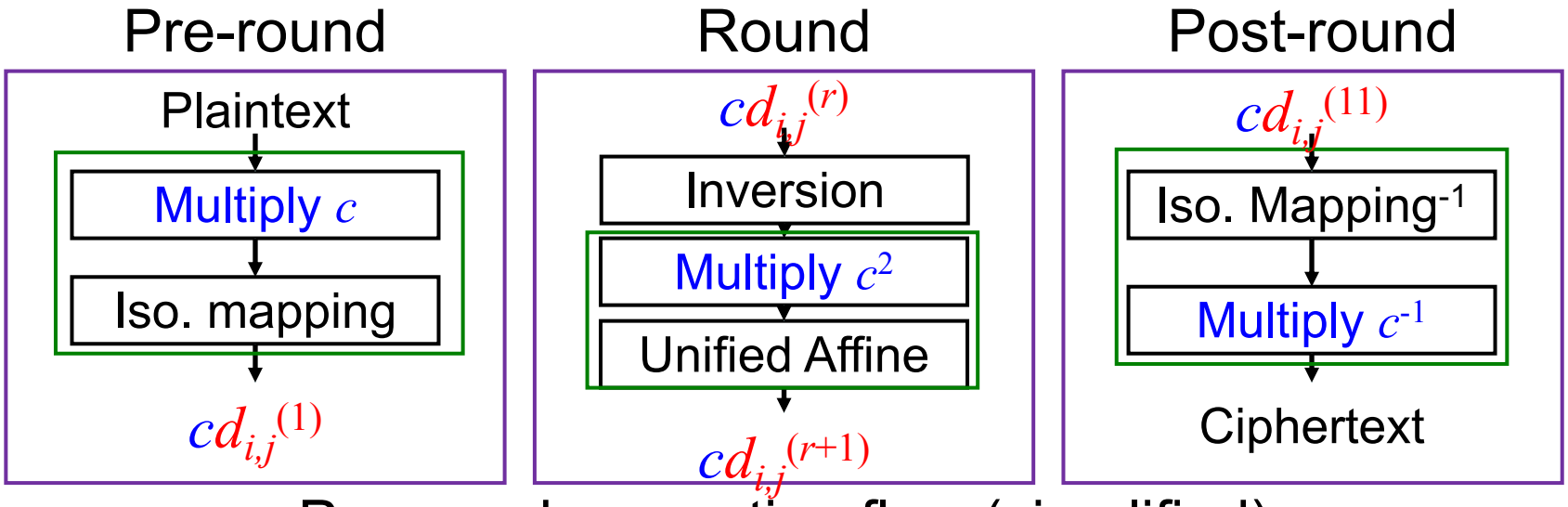
- Increase variation of construction of XOR matrices
  - To find optimal XOR matrices with lower HWs
- Multiply offset value c to intermediate value $d_{i,j}^{(r)}$ and store $cd_{i,j}^{(r)}$ into register
  - Multiplication with fixed value is XOR matrix operation
  - c is taken from $GF(2^8)$ excluding 0



Original encryption flow (simplified)

Multiplicative-offset

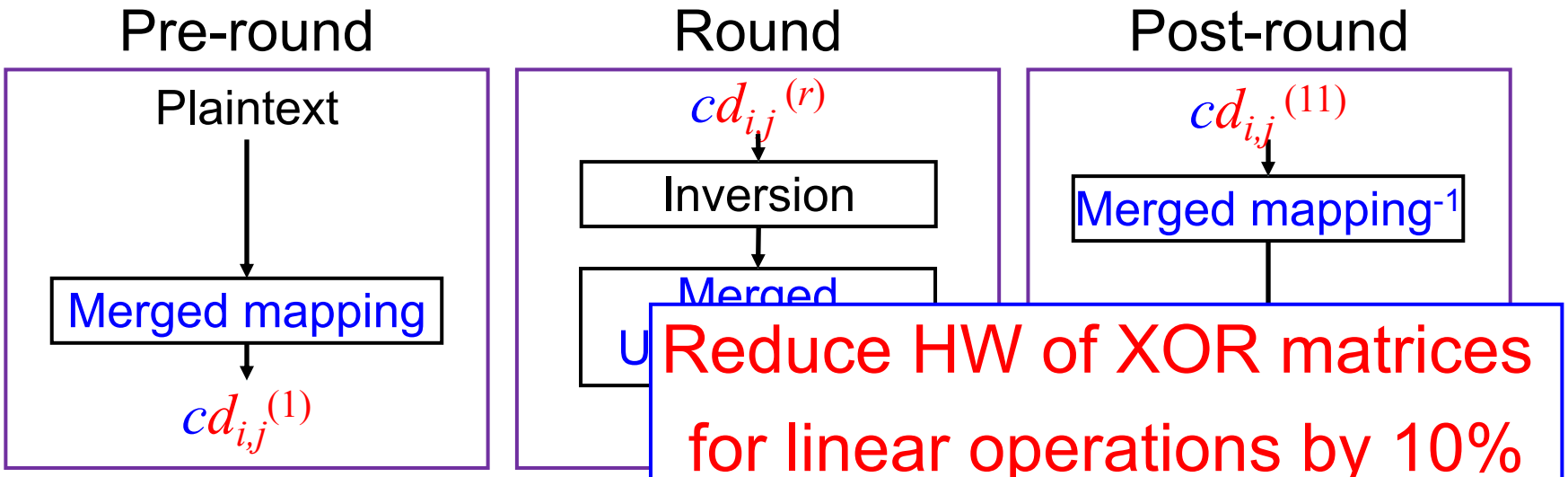
- Increase variation of construction of XOR matrices
  - To find optimal XOR matrices with lower HWs
- Multiply offset value c to intermediate value $d_{i,j}^{(r)}$ and store $cd_{i,j}^{(r)}$ into register
  - Multiplication with fixed value is XOR matrix operation
  - c is taken from $GF(2^8)$ excluding 0



Proposed encryption flow (simplified)

Multiplicative-offset

- Increase variation of construction of XOR matrices
  - To find optimal XOR matrices with lower HWs
- Multiply offset value c to intermediate value $d_{i,j}^{(r)}$ and store $cd_{i,j}^{(r)}$ into register
  - Multiplication with fixed value is XOR matrix operation
  - c is taken from $GF(2^8)$ excluding 0



Original encryption flow (simplified)

Performance comparison

- Synthesized proposed and conventional archs.
  - Logic synthesis: Design Compiler
  - Technology: Nangate 45-nm Open Cell Library

	Area (GE)	Latency (ns)	Throughput (Gbps)	Efficiency (Kbps/GE)
Satoh et al.	16,628.67	24.97	5.64	339.10
Lutz et al.	28,301.33	16.20	7.90	279.18
Liu et al.	15,335.67	29.70	4.74	309.13
Mathew et al.	21,429.33	30.80	4.57	213.33
This work w/o MO	18,013.00	16.28	8.65	480.49
This work w/ MO	17,368,67	15.84	8.89	511.78

- 51—57% higher efficient than conventional ones
  - Multiplicative-offset (MO) improves efficiency by 7—9%

Evaluation of power/energy consumption

- Gate-level timing simulation with back-annotation for estimating power consumption
  - With regarding glitch-effects

Power consumption and power-latency product at encryption

	Power [uW] @ 100 MHz	PL product
Satoh et al.	902	22,523
Lutz et al.	735	11,907
Liu et al.	1,010	29,997
Mathew et al.	1,390	42,812
This work w/o MO	569	9,263
This work w/ MO	465	7,366

- Our architecture achieved lowest power/energy
  - MO achieves further reduction by 7—24%

Encryption only architecture

- Designed encryption-only hardware based on our philosophy
  - Compared with representative open-source IP (SASEBO IP) and state-of-the-art one [ARITH 2016]

		Area (GE)	Latency (ns)	Thru (Gbps)	Thru/GE	Power (uW)	PL product
SASEBO IP	Table	23,085.00	11.64	12.00	519.66	352	4,097
	Comp	11,431.67	23.04	6.06	530.16	513	11,820
ARITH 2016	Type-I	12,108.33	23.87	5.90	487.16	655	14,266
	Type-II	13,249.33	21.78	6.46	487.92	755	18,022
This work		12,127,00	13.97	10.08	831.10	279	3,898

- Our architecture is 58—64% higher efficient
  - Also advantageous in power/energy consumption

Massages take away

- Round-based implementation of block ciphers may be essential for evaluating their performance
  - Should be conscious of mode-of-operations, applications, etc.
  - Optimizing round datapath is valuable and essential

- Feedback to block cipher design?
  - Optimized MDS matrices for cryptanalyses ≠ optimized for implementation (area and latency)
    - But it can be optimized at implementation for implementation
  - Inversion-based 8-bit Sbox makes many spaces for architectural/design optimization

References

- R. Ueno et al., “A High Throughput/Gate AES Hardware Architecture by Compressing Encryption and Decryption Datapaths—Toward efficient CBC-Mode Implementation,” CHES 2016.

- R. Ueno et al., “High Throughput/Gate AES Hardware Architectures Based on Datapath Compression,” IEEE Trans. Comput., 2019. (Early Access)