Lightweight and Efficient Implementation of AES

Raziyeh Salarifard
R_salarifard@sbu.ac.ir

Table of Content

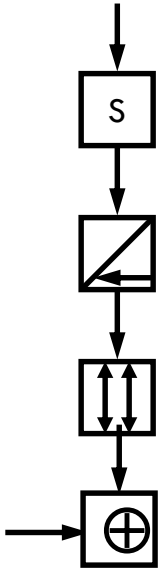
- Introduction
- Software Implementation
  - T-table
- Hardware Implementation
  - S-box logical implementation
  - Basis transformation in finite fields
  - Resource sharing
  - Gating technique
  - Technology mapping optimization
- Evaluation
- Conclusion
- References

Introduction

- Advanced Encryption System (AES)
- NIST standard, 2001
- Two bejian designers
  - Vincent Rijmen, Joan Daemen

AES Structure

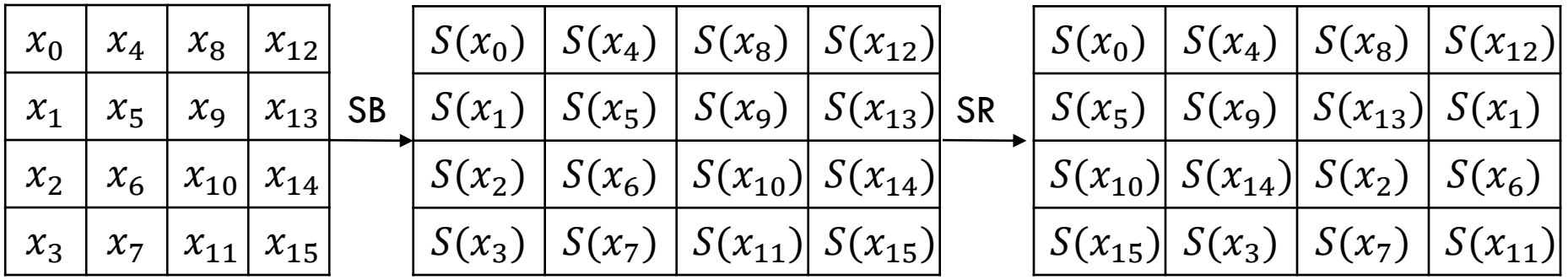
- Block cipher size: 128 bit (16 byte)
- Key size: 128, 192, 256
- Number of rounds: 10, 12, 14
- Each round includes four steps:
  - Substitution byte
  - Shift Row
  - Mix columns (except the last round)
  - Add round key



AES Software Implementation

- No memory limitation
  - In comparison with hardware
- AES Round Precomputation
  - Without Add Round Key

AES Round (without Add Round Key)



- MixColumns for the first column:

$$\xrightarrow{MC} \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} S(x_0) \\ S(x_5) \\ S(x_{10}) \\ S(x_{15}) \end{bmatrix} = \begin{bmatrix} 2.S(x_0) + 3.S(x_5) + S(x_{10}) + S(x_{15}) \\ S(x_0) + 2.S(x_5) + 3.S(x_{10}) + S(x_{15}) \\ S(x_0) + S(x_5) + 2.S(x_{10}) + 3.S(x_{15}) \\ 3.S(x_0) + S(x_5) + S(x_{10}) + 2.S(x_{15}) \end{bmatrix}$$

AES Round Precomputation (without Add Round Key)

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} S(x_0) \\ S(x_5) \\ S(x_{10}) \\ S(x_{15}) \end{bmatrix} = \begin{bmatrix} 2.S(x_0) + 3.S(x_5) + S(x_{10}) + S(x_{15}) \\ S(x_0) + 2.S(x_5) + 3.S(x_{10}) + S(x_{15}) \\ S(x_0) + S(x_5) + 2.S(x_{10}) + 3.S(x_{15}) \\ 3.S(x_0) + S(x_5) + S(x_{10}) + 2.S(x_{15}) \end{bmatrix}$$

$$T_0[z] = \begin{bmatrix} 02.S(z) \\ S(z) \\ S(z) \\ 03.S(z) \end{bmatrix}, T_1[z] = \begin{bmatrix} 03.S(z) \\ 02.S(z) \\ S(z) \\ S(z) \end{bmatrix}, T_2[z] = \begin{bmatrix} S(z) \\ 03.S(z) \\ 02.S(z) \\ S(z) \end{bmatrix}, T_3[z] = \begin{bmatrix} S(z) \\ S(z) \\ 03.S(z) \\ 03.S(z) \end{bmatrix},$$

- MixColumns for the first column:

$$T_0[x_0] \oplus T_1[x_5] \oplus T_2[x_{10}] \oplus T_3[x_{15}]$$

- Other columns:

$$T_0[x_4] \oplus T_1[x_9] \oplus T_2[x_{14}] \oplus T_3[x_3]$$
$$T_0[x_8] \oplus T_1[x_{13}] \oplus T_2[x_2] \oplus T_3[x_7]$$
$$T_0[x_{12}] \oplus T_1[x_1] \oplus T_2[x_6] \oplus T_3[x_{11}]$$

T-Table Implementation

- Implementing AES rounds using T_i tables
  - Except the last round
- Each rounds calls T_0 ،T_1 ،T_2 and T_3, 4 times
- The last round
  - No mixColumns
  - Design another tables
  - Using T_i tables

$$T_0[z] = \begin{bmatrix} \\ S(z) \\ \\ \end{bmatrix} \quad T_1[z] = \begin{bmatrix} \\ S(z) \\ \\ \end{bmatrix} \quad T_2[z] = \begin{bmatrix} S(z) \\ \\ \\ \end{bmatrix} \quad T_3[z] = \begin{bmatrix} S(z) \\ \\ \\ \end{bmatrix}$$

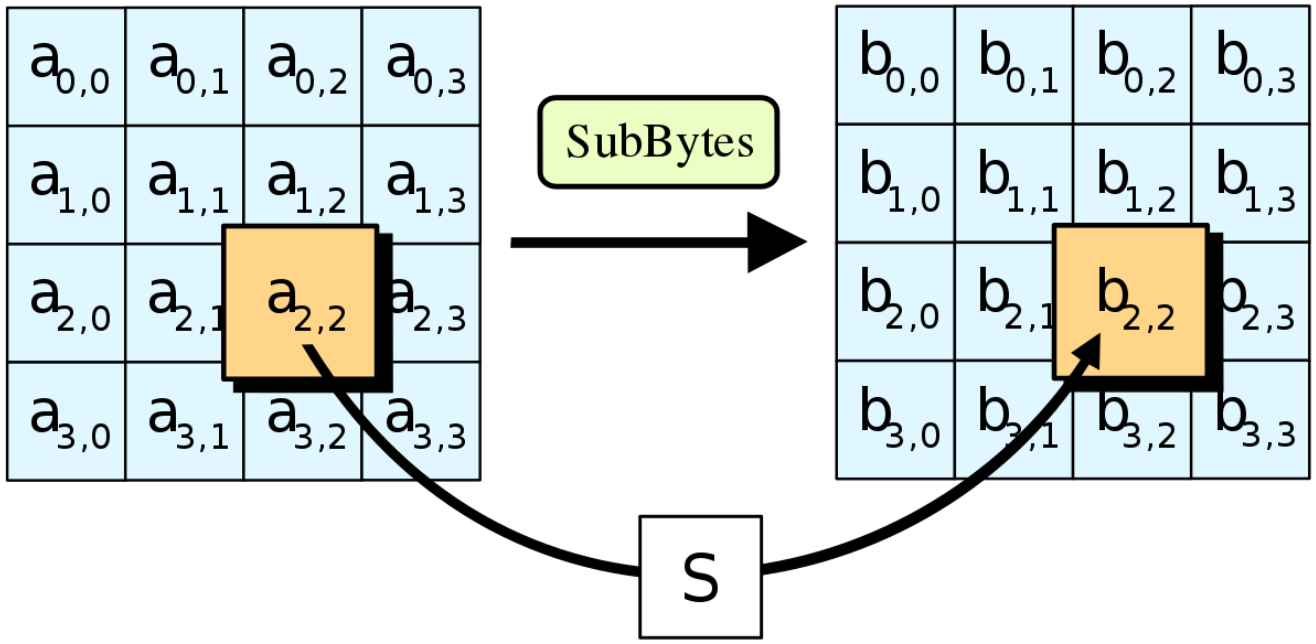
AES Hardware Implementation

- Memory limitation
  - Almost all hardware boards
- Design efficient architecture
  - Efficient frequency and area
- Understanding AES rounds logic
  - Substitution byte
  - Shift Row
  - Mix columns (except the last round)
  - Add round key

Hardware Implementation Techniques

- S-box logical implementation
  - S-box
- Basis transformation in finite fields
  - S-box
- Resource sharing
  - MixColumns
- Gating technique
  - AES rounds
  - ShiftRow
- Technology mapping optimization
  - The state and key path

S-box logical implementation



Forward S-box

- $s = S(c)$
  - c 8-bit input
  - s 8-bit output
  - Polynomials over GF(2)
- The input "s" is mapped to its multiplicative inverse in:
  - $GF(2^8) = GF(2)[x]/(x^8 + x^4 + x^3 + x + 1)$.

Forward S-box (cont.)

- The multiplicative inverse is then transformed using the following affine transformation:

$$\begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

$$s = b \oplus (b \lll 1) \oplus (b \lll 2) \oplus (b \lll 3) \oplus (b \lll 4) \oplus 63_{16}$$

- where [s_7, …, s_0] is the S-box output and [b_7, …, b_0] is the multiplicative inverse as a vector.

Inverse S-box

- Simply the S-box run in reverse.
- Inverse affine transformation:

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

- The multiplicative inverse

S-box logical implementation

- Multiplicative Inverse
- Affine Transformation
  - Forward S-box
    - Xor, rotate to left

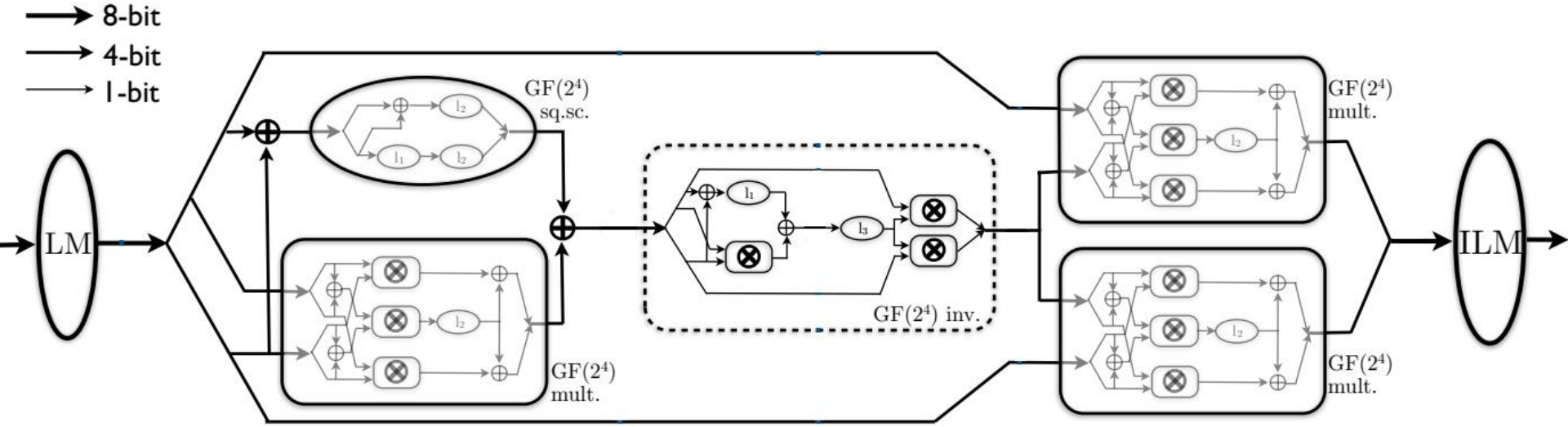
$$s = b \oplus (b \lll 1) \oplus (b \lll 2) \oplus (b \lll 3) \oplus (b \lll 4) \oplus 63_{16}$$

Multiplicative Inverse

- Composite fields
- GF(2^8)
  - GF(2^4)
    - GF(2^2)
      - GF(2)
- GF(2^8): inverse
  - GF(2^4): addition, multiplication, inverse
    - GF(2^2) : addition, multiplication, inverse
      - GF(2): XOR, AND

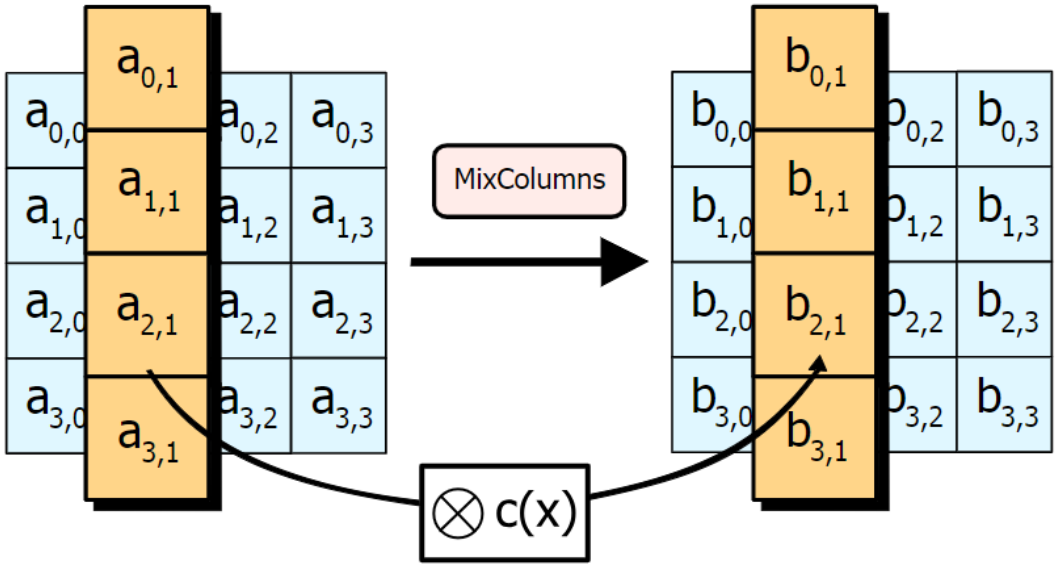
Canright GF(2^8) Inversion

- CHES-2005
- Hardware Implementation Techniques
  - S-box logical implementation
  - Basis transformation in finite fields



Resource sharing

- MixColumns



Matrix representation of MixColumns

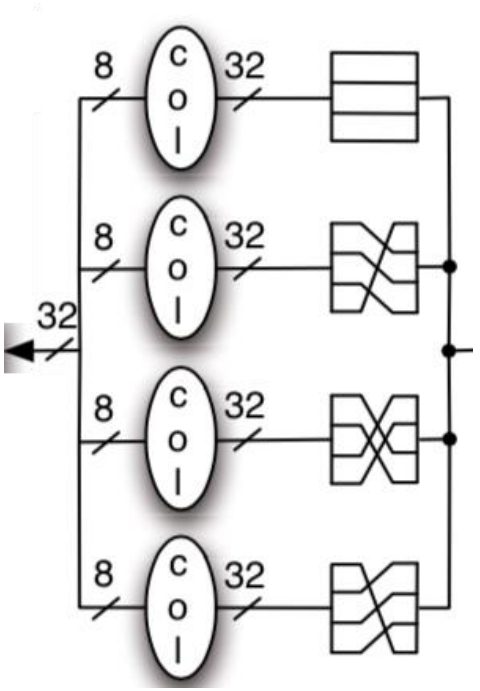
- A circulant matrix

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Resource sharing in MixColumns Implementation

- [Moradi et al, 2011]
- A circulant matrix
- One multiplier circuit
  - Shift inputs

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$



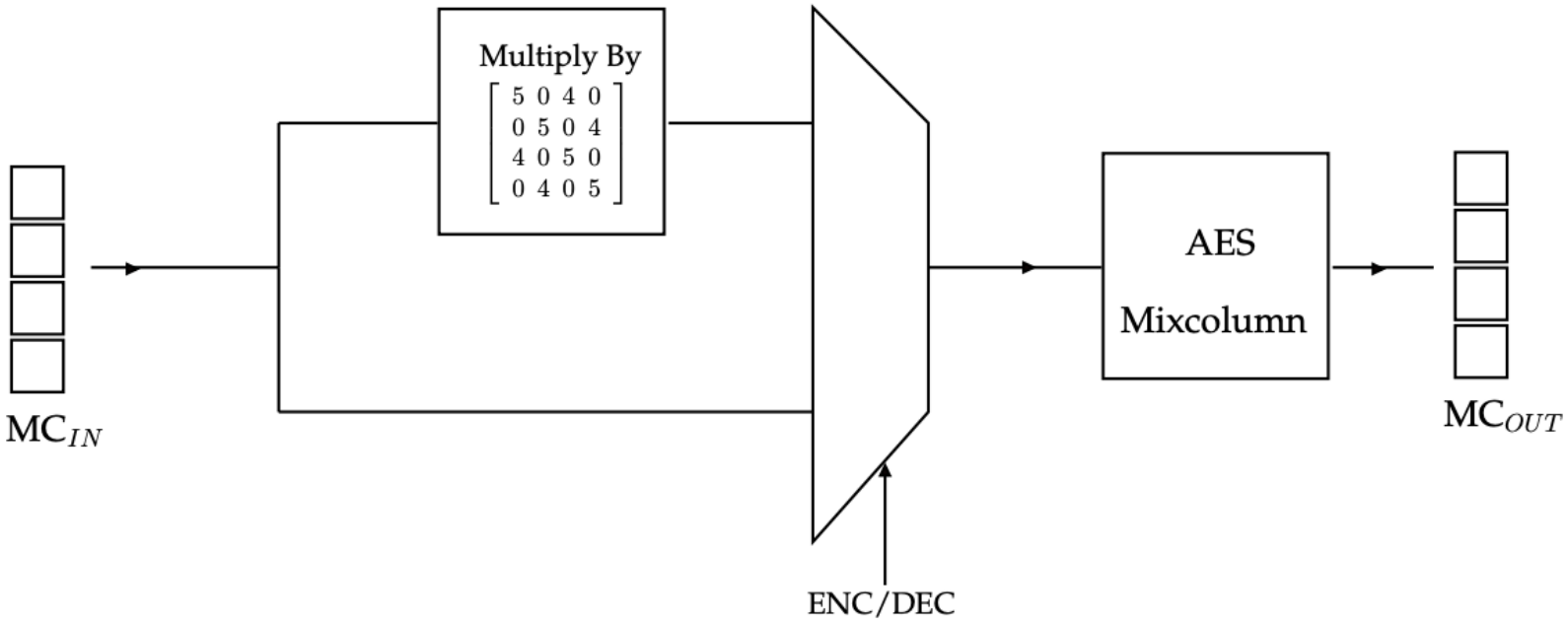
Matrix representation of InverseMixColumns

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 14 & 11 & 13 & 9 \\ 9 & 14 & 11 & 13 \\ 13 & 9 & 14 & 11 \\ 11 & 13 & 9 & 14 \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix}$$

$$\begin{bmatrix} 14 & 11 & 13 & 9 \\ 9 & 14 & 11 & 13 \\ 13 & 9 & 14 & 11 \\ 11 & 13 & 9 & 14 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} 5 & 0 & 0 & 4 \\ 0 & 5 & 0 & 4 \\ 4 & 0 & 5 & 0 \\ 0 & 4 & 5 & 0 \end{bmatrix}$$

Mixcolumns and InverseMixcolumns Implementation



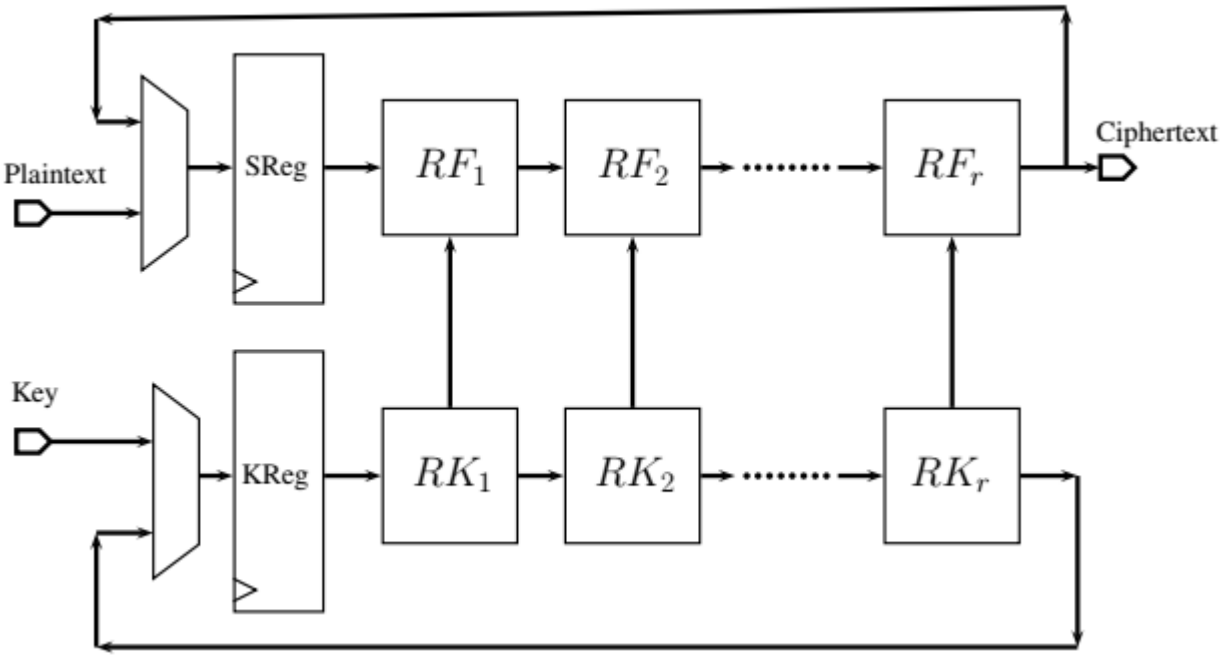
S. Banik et al

Gating technique

- Reduce dynamic power consumption
- Round gating
- Clock gating

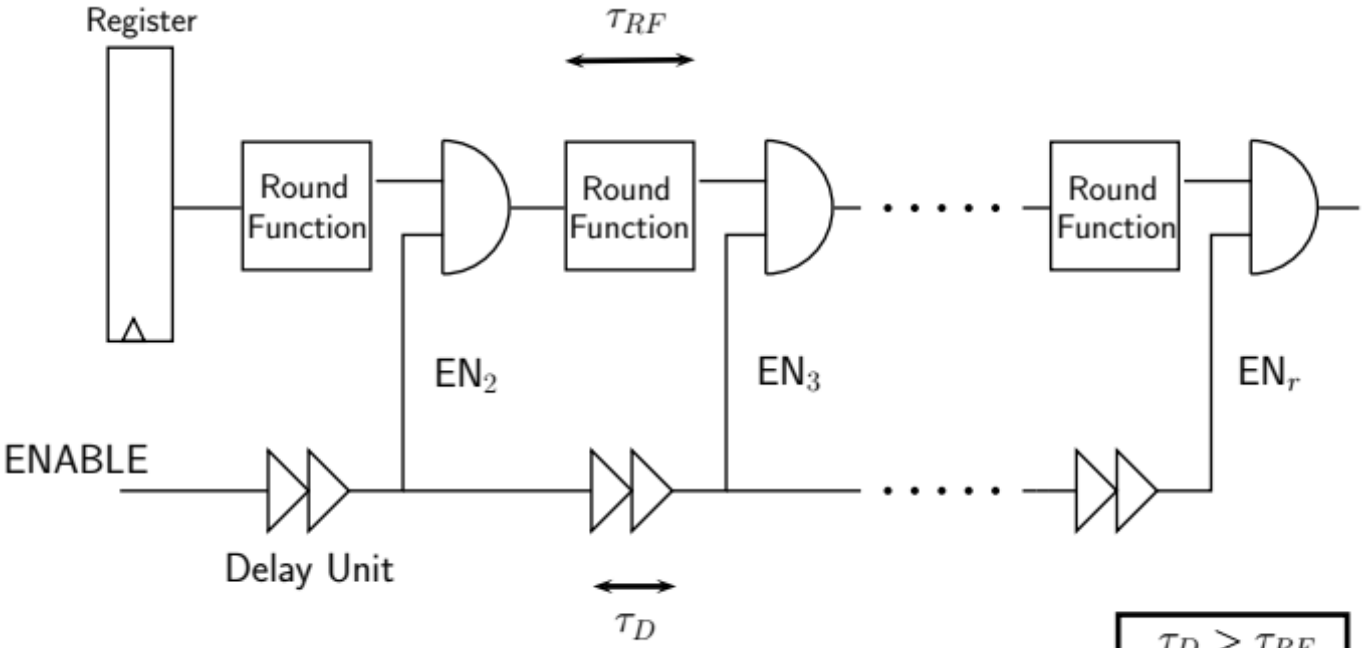
Round gating

- Unrolled implementation
- r rounds: r < R (number of rounds)



Round gating (cont.)

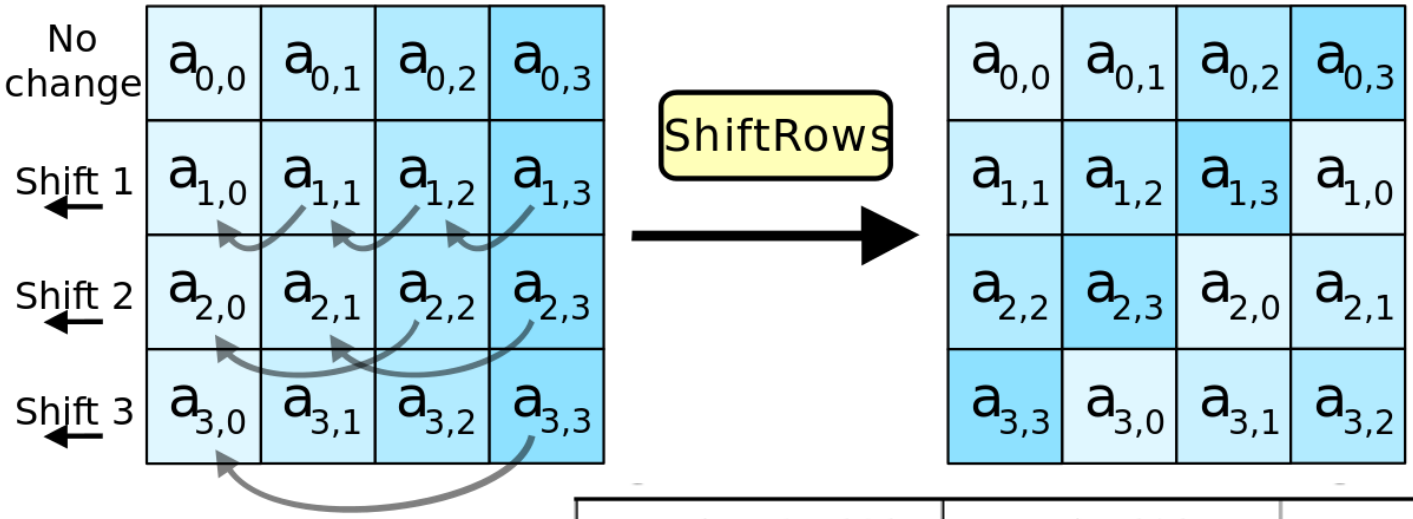
- Glitches
  - More power consumption
- Disable next rounds



Clock gating

- Clock gating
  - Switching activity
  - Disable FFs clock
  - ShiftRow

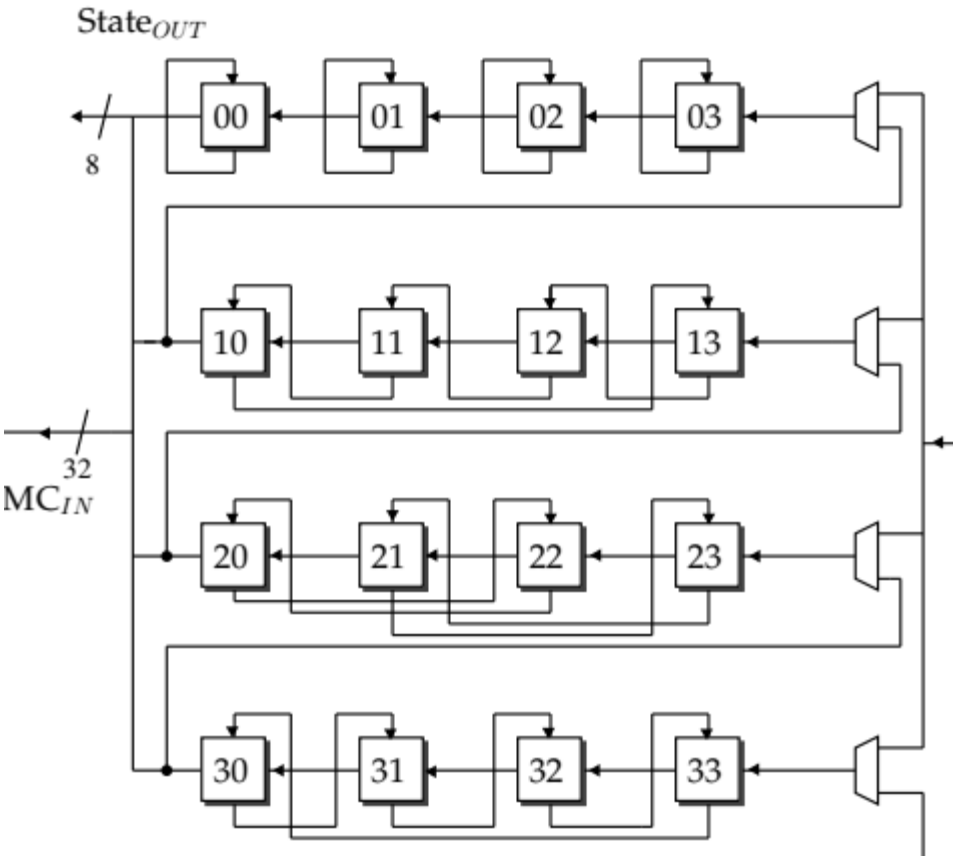
Clock gating (cont.)



کلاک‌های معکوس جابه‌جایی سطری			کلاک‌های جابه‌جایی سطری			شماره سطر
2	1	0	2	1	0	
F	F	F	F	F	F	0
O	O	O	O	F	F	1
O	O	F	O	O	F	2
O	F	F	O	O	O	3

Technology mapping optimization

- Choose the optimized cells
- Example:
  - Nangate 45
    - Xnor
  - Scan FFS
    - States and roundKeys



Evaluation

- SW
  - Software code (C, C++, Java, etc)
  - Time
- HW
  - HDL
    - Verilog
    - VHDL
  - FPGA
  - ASIC

Hardware Evaluation (FPGA)

- Xilinx ISE
- Device (Zynq, Vertix, etc.)
  - Related work
- Slice, DSP, CPD, #CCs
- AT = Area X Time
  - Area in FPGA?
    - DSPs
    - Slices
    - One DSP = 100 Slices [Salarifard et al, TCAS-I, 2019].

Hardware Evaluation (ASIC)

- Technology
  - 32 nm, 45 nm, 65 nm , etc.
- Standard Cell Libraries
  - Nangate, TSMC, UMC, etc.
  - Not available in web!
- Estimation
  - Gate (NAND, NOR)
  - CPD
    - Gate delay
  - Delay
    - Number of maximum FFs from input to output
- Implementation
  - Design Compiler
    - Area, CPD, Power, Energy, etc.
  - No free, Vmware (CentOS).

Conclusion

- Introduction
- Software Implementation
  - T-table
- Hardware Implementation
  - S-box logical implementation
  - Resource sharing
  - Basis transformation in finite fields
  - Gating technique
  - Technology mapping optimization
- Evaluation
  - SW
    - Time
  - HW
    - FPGA and ASIC

References

- جهانبانی, محسن, et al. "مروری بر پیاده‌سازی‌های سخت‌افزاری سبک‌وزن رمز AES." *دوفصل نامه علمی ترویجی منادی امنیت فضای تولید و تبادل اطلاعات (افتا)* ۶,۲ (۲۰۱۸): ۳-۲۰.

- Moradi, Amir, et al. "Pushing the limits: A very compact and a threshold implementation of AES." *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, Berlin, Heidelberg, 2011.

- Banik, Subhadeep, Andrey Bogdanov, and Francesco Regazzoni. "Atomic-AES: A compact implementation of the AES encryption/decryption core." *International Conference on Cryptology in India*. Springer, Cham, 2016.

- Canright, David. "A very compact S-box for AES." *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, Berlin, Heidelberg, 2005.

- Banik, S., Bogdanov, A., Regazzoni, F., Isobe, T., Hiwatari, H., & Akishita, T. (2016). *Round gating for low energy block ciphers. 2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. doi:10.1109/hst.2016.7495556.

Any question?

R_salarifard@sbu.ac.ir