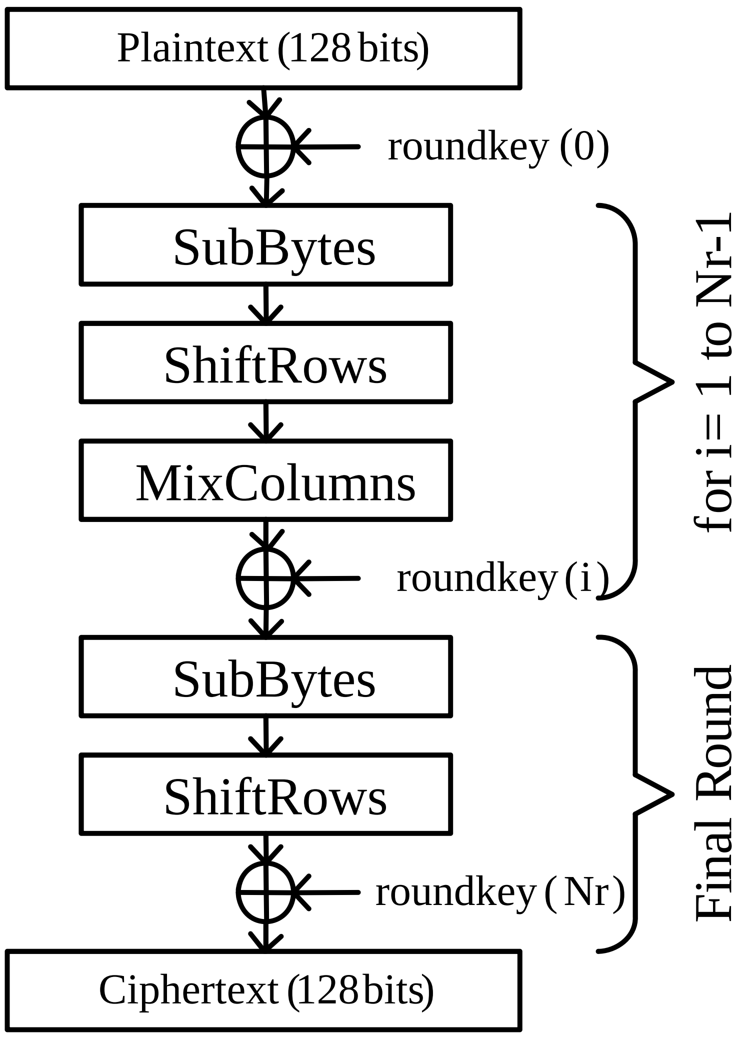
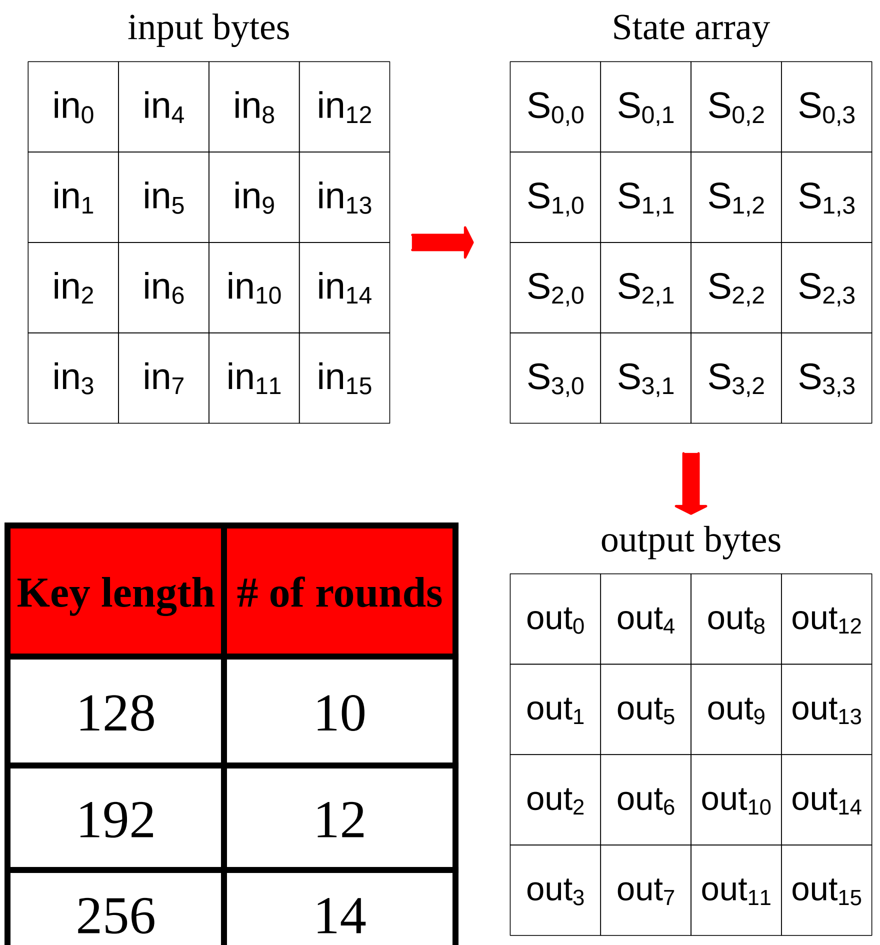
VLSI Architectures of XTS-AES for Data Storage

Xinmiao Zhang

SanDisk

The AES Algorithm (Encryption)





input bytes

in_0	in_4	in_8	in_{12}
in_1	in_5	in_9	in_{13}
in_2	in_6	in_{10}	in_{14}
in_3	in_7	in_{11}	in_{15}

State array

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

output bytes

out_0	out_4	out_8	out_{12}
out_1	out_5	out_9	out_{13}
out_2	out_6	out_{10}	out_{14}
out_3	out_7	out_{11}	out_{15}

Key length	# of rounds
128	10
192	12
256	14

Each byte is considered as a GF(2^8) symbol

Transformations in the AES Algorithm

- SubBytes Transformation $S'_{i,j} = M \times S^{-1}{}_{i,j} + b$ M, b: constant
- ShiftRows Transformation

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$	$\xrightarrow{\text{no shift}}$	$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$	$\xrightarrow{\text{shift one byte}}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$	$S_{1,0}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$	$\xrightarrow{\text{shift two bytes}}$	$S_{2,2}$	$S_{2,3}$	$S_{2,0}$	$S_{2,1}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$	$\xrightarrow{\text{shift three bytes}}$	$S_{3,3}$	$S_{3,0}$	$S_{3,1}$	$S_{3,2}$

- MixColumns Transformation: multiply $A(x) \bmod x^4 + 1$

$$A(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$$

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix} \qquad 0 \le c < 4$$

- InvShiftRows Transformation

Implementation of Finite Field Arithmetic

- Basis representation
  - Addition: XOR
  - Multiplication: modular polynomial multiplication
  - Inversion: difficult
- Power representation
  - Addition: difficult
  - Multiplication: exponent addition
  - Inversion: inverse exponent modulo reduction
- Composite field representation
  - Addition: XOR
  - Multiplication: shorter modular polynomial multiplication
  - Inversion: easy

Inversion over Composite Field GF((2^n)^2)

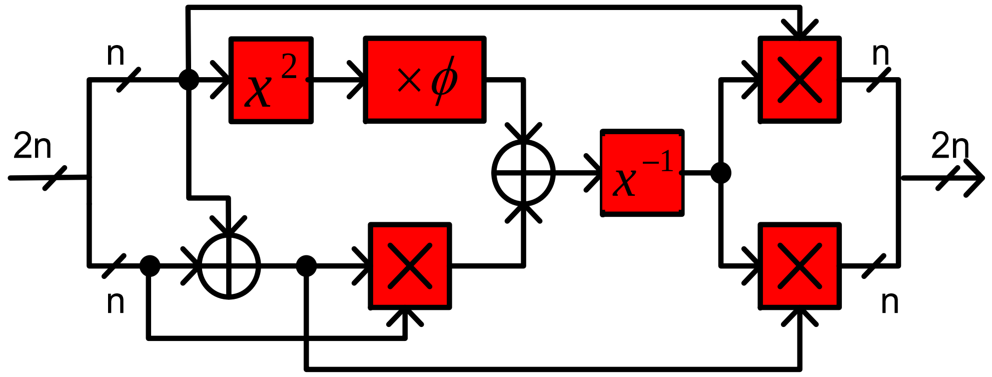
- An element $a \in GF((2^n)^2)$ can be represented as

$$a(x) = a_1 x + a_0, \quad a_1, a_0 \in GF(2^n)$$

- Assume that $GF((2^n)^2)$ is constructed from $GF(2^n)$ using irreducible polynomial $x^2 + x + \phi$ over $GF(2^n)$

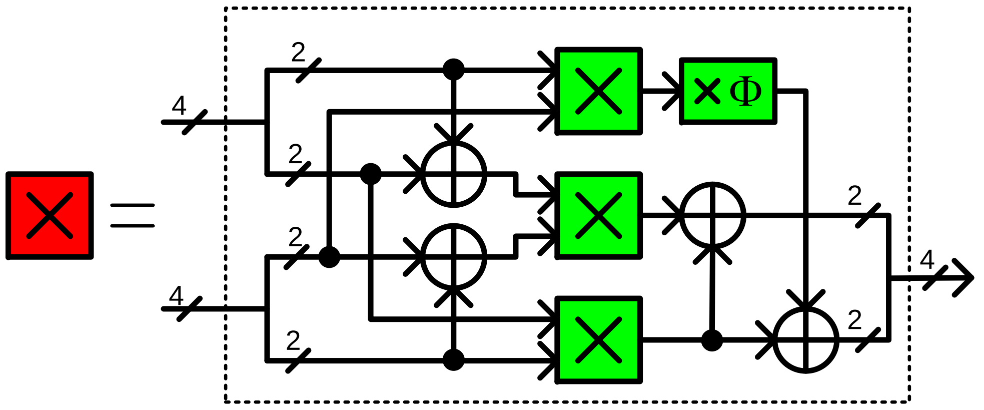
- Apply the Extended Euclidean algorithm

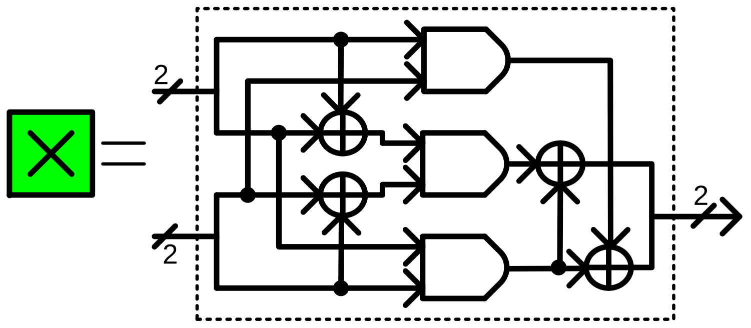
$$a^{-1}(x) = a_1 \Delta x + (a_1 + a_0)\Delta \qquad \Delta = (a_0(a_1 + a_0) + \phi a_1^2)^{-1}$$

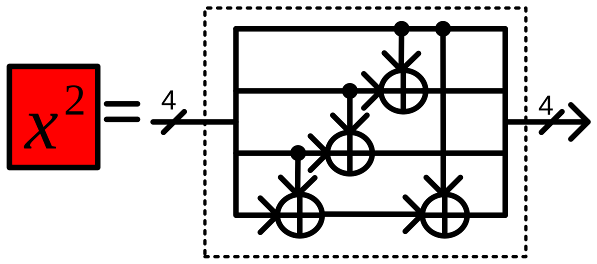


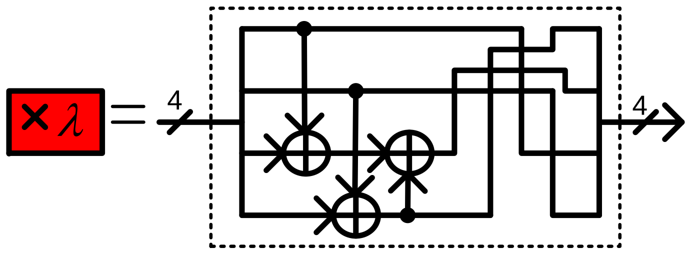
- The computations are over subfield $GF(2^n)$, are much simpler

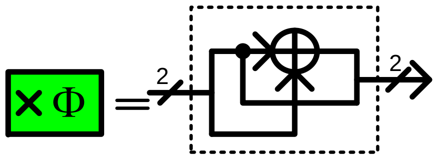
Computation Units over Composite Field



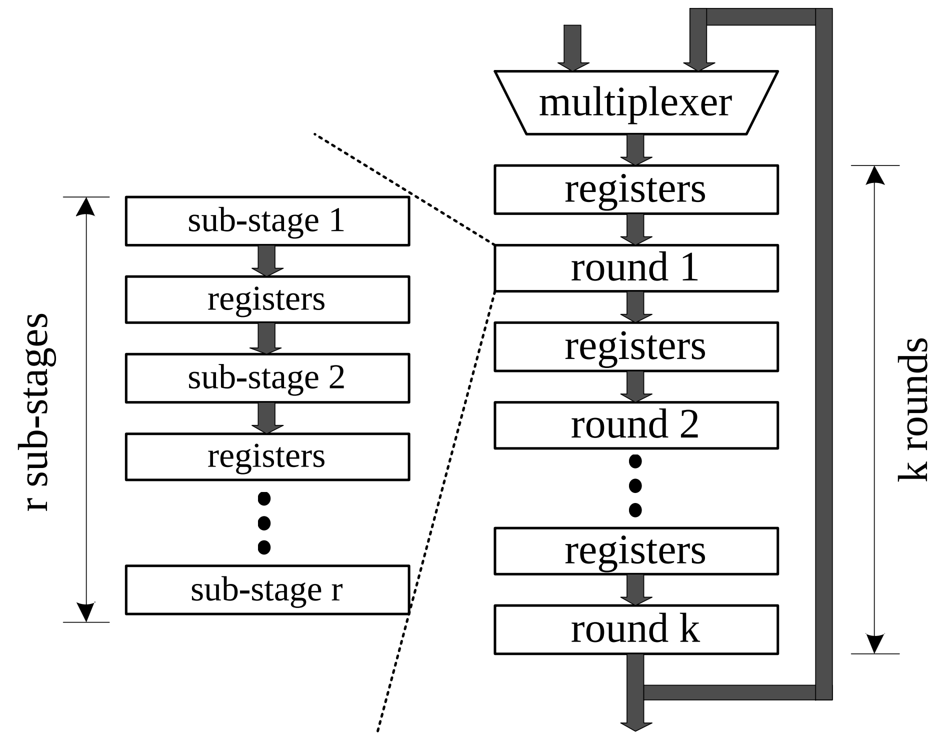








Sub-pipelined Architecture



$$\frac{\text{throughput}_{\text{sub-pipelining}}}{\text{throughput}_{\text{pipelining}}}$$

$$= \frac{r(1+\tau)}{1+r\tau}$$

$$\tau = \frac{t_{\text{setup}} + t_{\text{prop}} + t_{\text{mux}}}{t_{\text{round}}}$$

FPGA Implementation Results

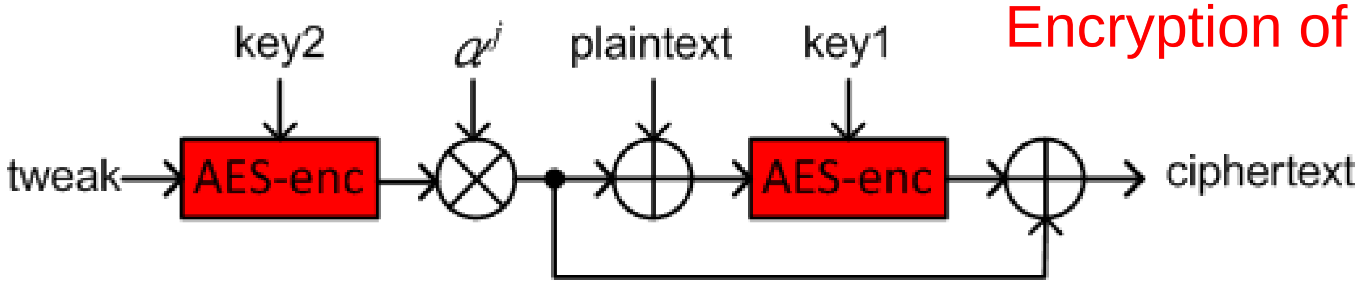
	Device	Freq.(Mhz)	Throughput (Mbps)	Slices	BRAMs	Mbps/ Slice
Elbirt *et al*	XCV1000-4	31.8	1938	10992	0	0.176
McLoone *et al*	XCV812e-8	93.9	12020	2000	244	0.362
Jarvinen *et al*	XCV1000e-8	129.2	16500	11719	0	1.408
Saggese *et al*	XCV2000e-8	158	20300	5810	100	1.091
Standaert *et al*	XCV3200e-8	145	18560	15112	0	1.228
This design*(r=3)	XCV812e-8	93.5	11965	9406	0	1.272
This design*(r=7)	XCV1000e-8	168.4	21556	11022	0	1.956

* X. Zhang and K. K. Parhi, "High-speed VLSI architectures for the AES algorithm," IEEE Trans. on VLSI Systems, vol. 12(9), pp. 957-967, Sep. 2004.

XTS-AES Mode for Data Storage

- XTS: XEX (xor-encrypt-xor) encryption mode with tweak and ciphertext stealing
- Data block: 128 bits
- Key (key1, key2): 256 or 512 bits
- tweak: 128 bits, usually the address of the first block
- j: index of the 128-bit block within data unit
- α: a primitive element of $GF(2^{128})$ constructed by $x^{128}+x^7+x^2+x+1$

Encryption of a block



Conclusions

- The AES algorithm contributes to the majority of the XTS-AES standard complexity
- Composite field arithmetic substantially simplifies the involved computations over finite fields
- Substructure sharing further reduces gate count
- Sub-pipelined architecture leads to higher throughput and hardware efficiency
- Roundkeys can be generated on the fly using a high-speed retimed architecture

Questions?

xinmiao.zhang@sandisk.com