Efficient Implementation of Rijndael Encryption in Reconfigurable Hardware: Improvements and Design Tradeoffs

FX Standaert (**S**), G Rouvroy,
JJ Quisquater, JD Legat

UCL Crypto Group
Laboratoire de Microélectronique
Université Catholique de Louvain
Belgium

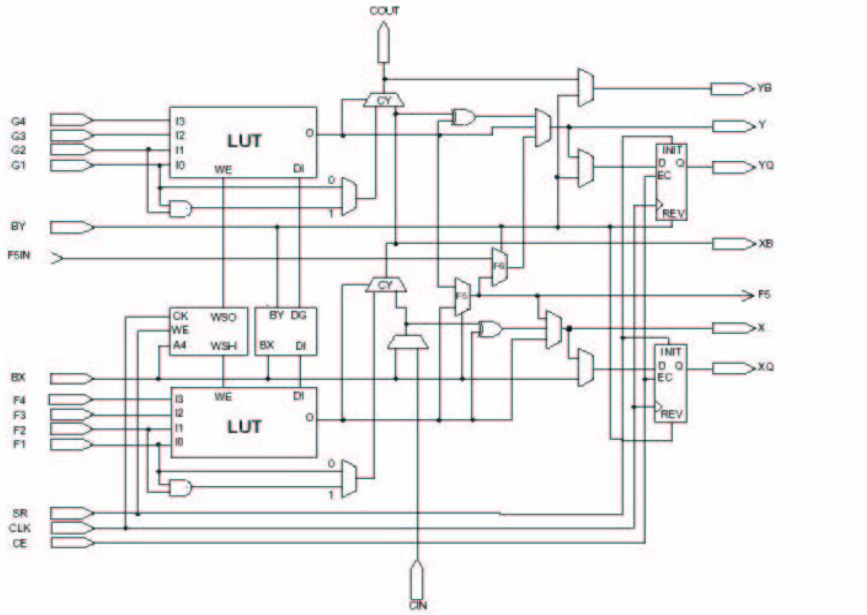
Structure of the talk:

1. Introduction.
2. Hardware description.
3. Block cipher description.
4. Implementation tradeoffs:
   (a) Design methodology.
   (b) Algorithmic optimization.
   (c) Implementation schemes.
   (d) Optimal pipeline.
5. Practical results and comparisons.
6. Further research and conclusion.

1. Introduction.

1. Encryption modes.
2. Implementation of Rijndael:
   (a) Lots of results.
   (b) Sometimes difficult to compare.
3. We present:
   (a) Definitions for hardware efficiency.
   (b) A design methodology.
   (c) Comparison tools.
   (d) Synthesis works, combination of improvements.
4. Practical results: efficiency of best-known designs improved by at least 25%.

2. Hardware description.



FPGA: Xilinx Virtex3200ECG1156-8:

1. 4-input LUTs => RAM, MUX, shift registers, ...
2. Storage elements: D flip-flops, latches.
3. Additional logic: XOR, MUXF5, MUXF6, ...
4. 100 RAM blocks: 1 dual-port 4096-bit RAM $\simeq$ 2 s-boxes 8 x 8.

3. Block cipher description.

a. The round function

$$\rho[K] = \sigma[K] \circ \theta \circ \delta \circ \gamma = \sigma[K]\big(\theta(\delta(\gamma))\big)$$

1. SubBytes, the non-linear layer γ is a non-linear byte substitution, operating on each byte independently. The substitution table (or s-box) is invertible and is constructed by the composition of two operations:
   (a) The multiplicative inverse in $GF(2^8)$.
   (b) An affine transform over $GF(2)$.
2. The ShiftRows transformation δ is a permutation of the bytes.

3. The MixColumns transformation θ: columns (= 4 bytes) are considered as polynomials over $GF(2^8)$ and multiplied modulo $x^4 + 1$ with a fixed polynomial $c(x)$, given by:

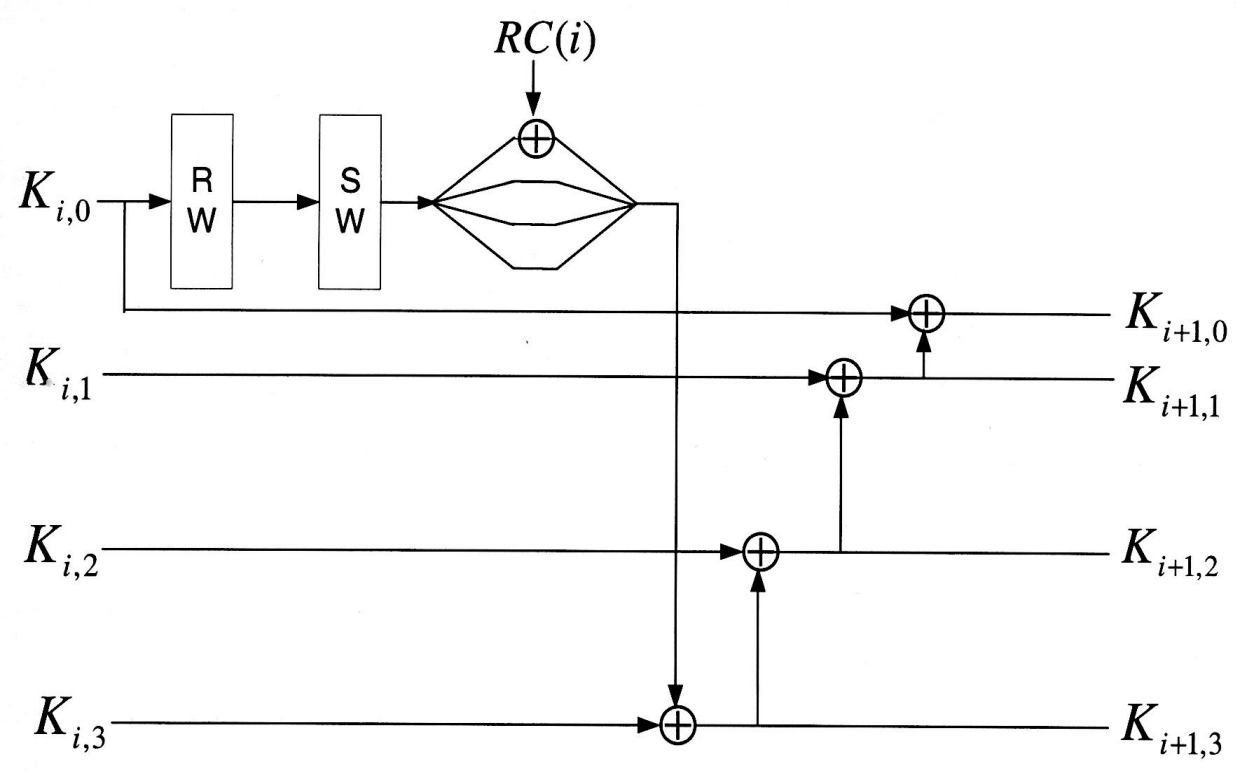
$$c(x) =' 03' x^3 +' 01' x^2 +' 01' x +' 02'$$

=> One output byte can be expressed as:

$$b_0 =' 02' \times a_0 \oplus' 03' \times a_1 \oplus' 01' \times a_2 \oplus' 01' \times a_3$$

4. The round key addition $\sigma[K]$ is a simple bitwise EXOR with a round key.

b. The key round



=> Complete cipher:

$$\alpha[K_0, K_1, ..., K_{10}] = \sigma[K_{10}] \circ \delta \circ \gamma \circ (\bigcirc_{r=1}^{9} \rho[K_r]) \circ \sigma[K_0]$$

4. Implementation tradeoffs

4.1 Design methodology:

FPGAs $\Rightarrow$ implementation constraints:

1. In terms of performances, let the efficiency of a block cipher be the ratio $Throughput$ $(Mbits/s)/Area\ (slices)$.
2. In terms of resources, the efficiency is easily tested by computing the ratio $Nbr\ of\ LUTs/Nbr\ of\ registers$: it should be close to one.

=> Maximum pipeline: $\frac{Nbr.of.LUTs}{Nbr.of.registers}$=1.

4.2 Algorithmic optimizations:

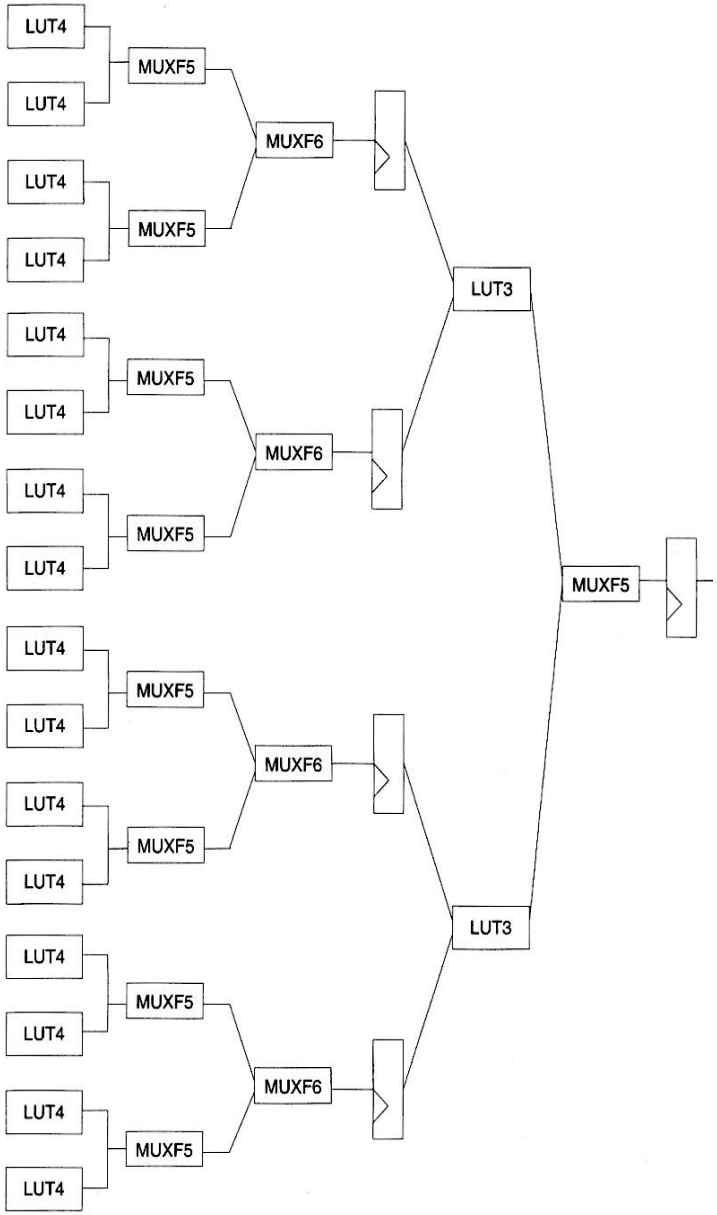
Implementation of the 8 x 8 substitution box:

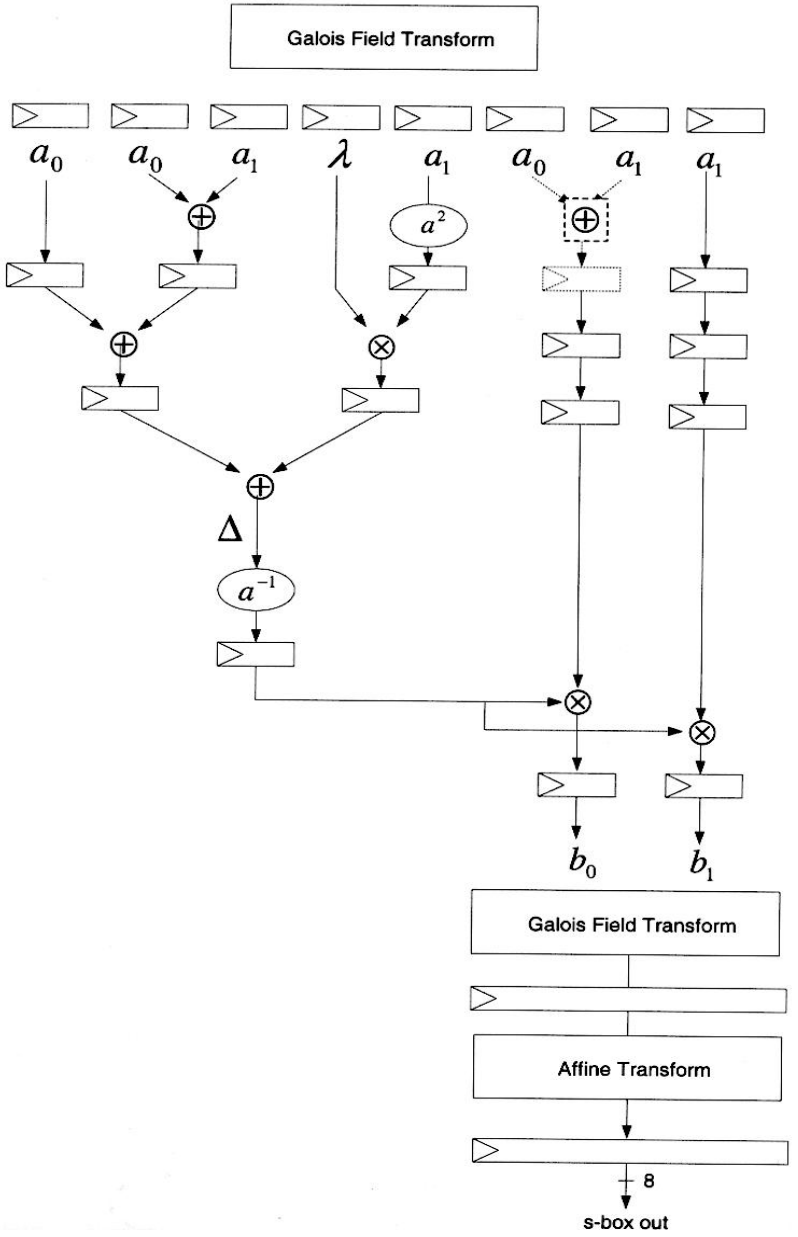
1. RAM-based implementation: 1 RAM block = 2 substitution boxes.
2. The multiplexor model:

Component	Nbr of LUT	Nbr of registers
γ	$144 \times 16 = 2304$	$42 \times 16 = 672$

3. Composite field solution: computations in the field GF(2^8) are replaced by computations in the composite field GF$(2^4)^2$ in order to reduce the size of the tables needed for the inversion.

Component	Nbr of LUT	Nbr of registers
γ	$84 \times 16 = 1344$	$76 \times 16 = 1216$

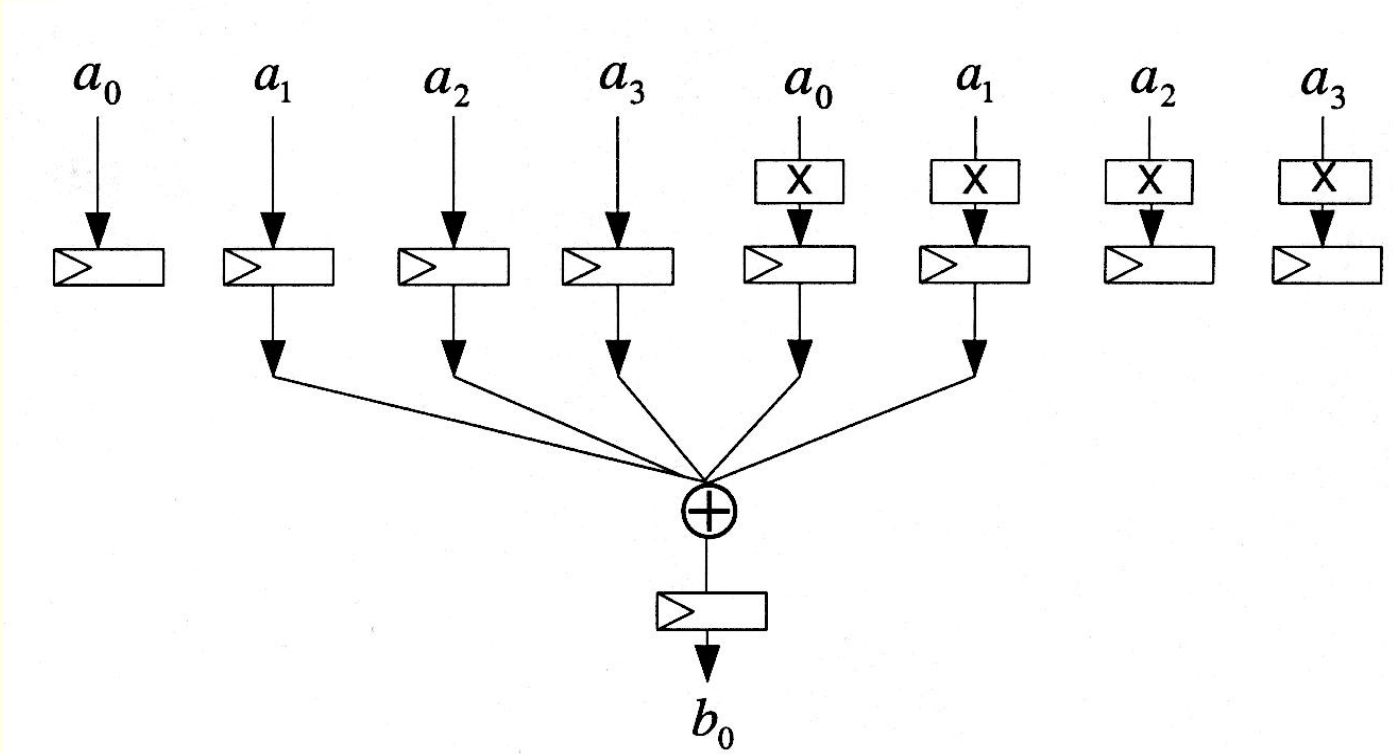

LUT4
MUXF5
MUXF6
LUT3


Galois Field Transform
a_0
a_1
λ
a^2
Δ
a^{-1}
b_0
b_1
Galois Field Transform
Affine Transform
8
s-box out

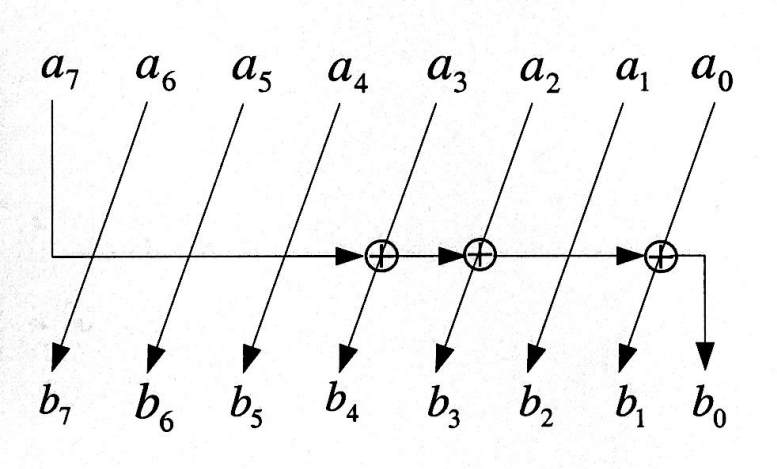
Mixadd combination:

Mixcolum operates on a 4-byte column and corresponds to multiplications and additions in $GF(2^8)$. For example, for the output byte b_0, we have:

$$b_0 =' 02' a_0 +' 03' a_1 +' 01' a_2 +' 01' a_3$$



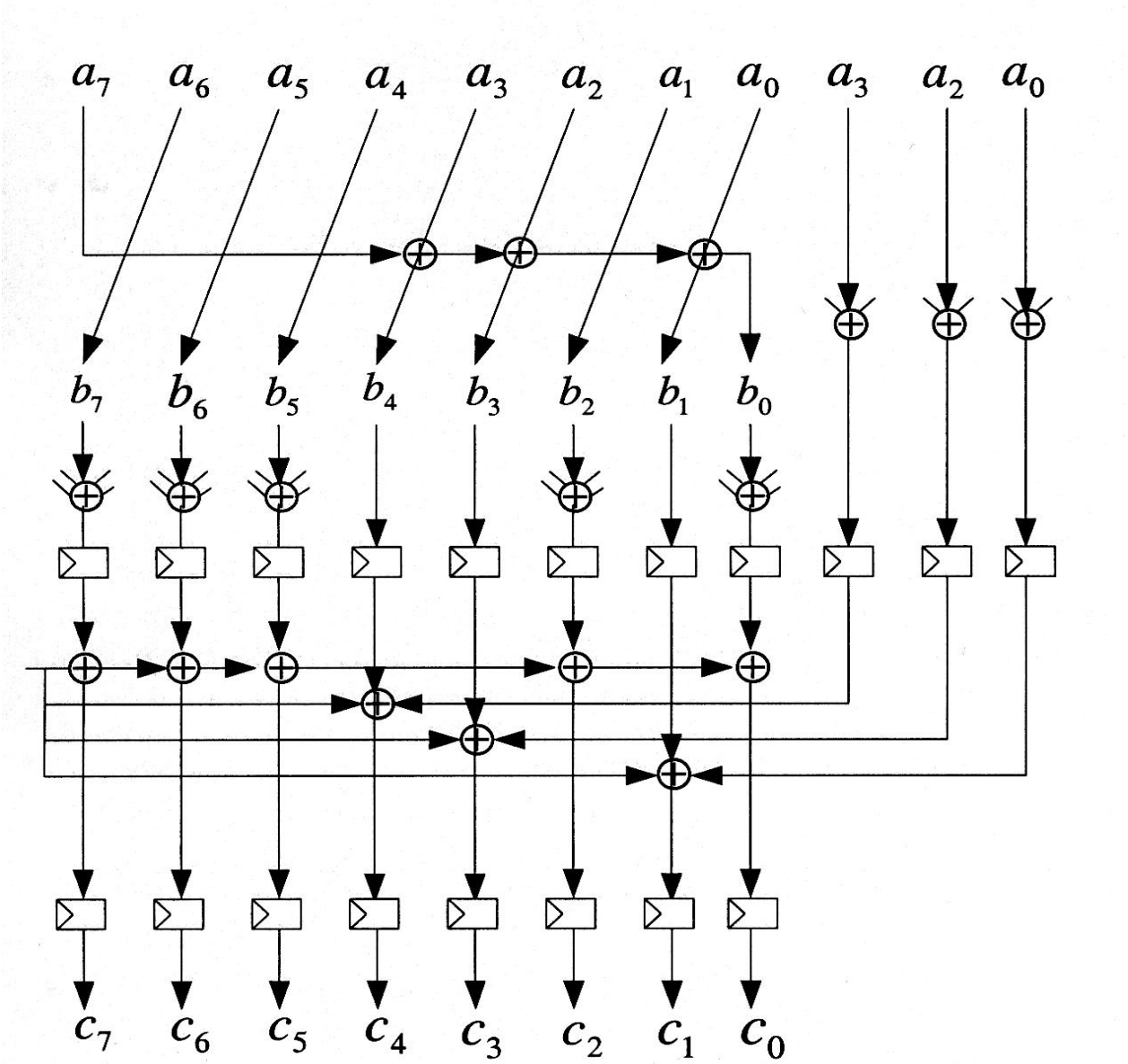
5 bits of function X are just shifted:

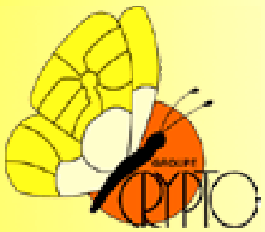


=> For these ones, only one register is needed to pipeline the diffusion layer. The remaining ones are combined with the key addition.
=> 2 pipeline stages in place of 3 !

Component	Nbr of LUT	Nbr of registers
ϵ	304	304

Mixadd (at the bit level) :

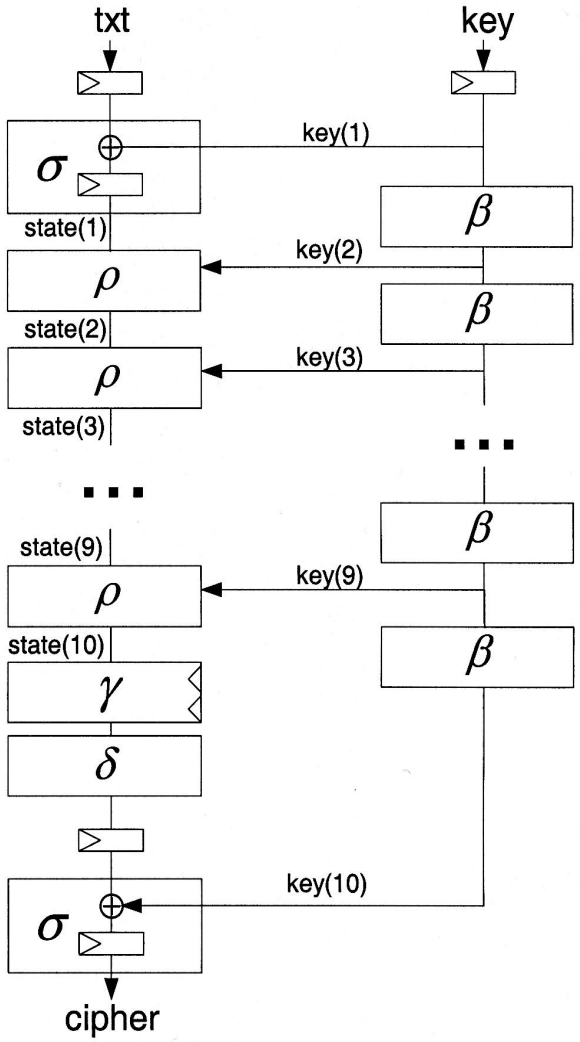


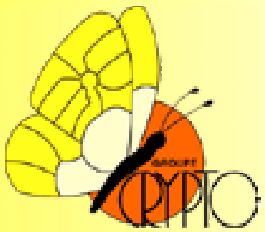


4.3 Implementation schemes:

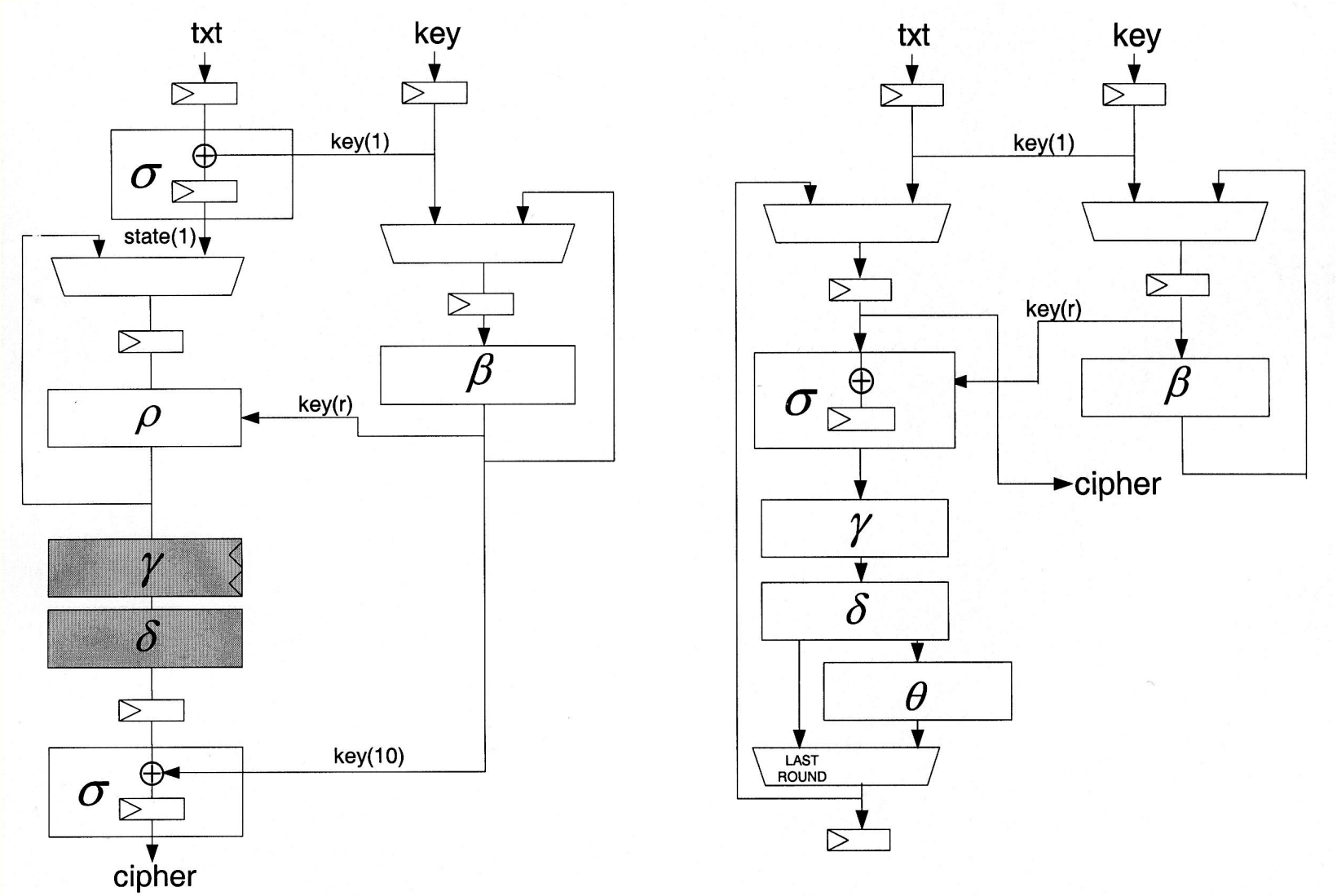
High throughput constraints:

Fully unrolled
pipeline scheme





Low area constraints: loop architecture.



4.4 Optimal pipeline search:

Optimal pipeline search

1. Start from the maximal pipeline, i.e. implement Rijndael with the best ratio $\frac{Nbr.of.LUTs}{Nbr.of.registers}$;
2. After implementation, compute the efficiency $E_{cur} = Throughput\ (Mbits/s)/Area\ (slices)$;
3. $OK = 0$;

While $OK = 0$ **do** {

1. Remove the pipeline stage that involves the lowest frequency reduction and re-implement Rijndael;
2. After implementation, compute the efficiency $E_{nxt} = Throughput\ (Mbits/s)/Area\ (slices)$;
3. **If** $E_{cur} \geq E_{nxt}$ **then** $OK = 1$; **else** $E_{cur} = E_{nxt}$;

}

4. The final efficiency E_{cur} specifies the optimal pipeline;

5. Practical results and comparisons.

Type	Nbr of LUTs	Nbr of slices	RAM blocks	Throughput (Mbits/s)	Throughput/Area ($\frac{Mbits/s}{slices, LUTs}$)
McLoone et al.	/	2222	100	6956	3.1
Our design	**3516**	**2784**	**100**	**11776**	**4.2**
Helion tech.	899	/	10	1187	1.32
Our design	**877**	**542**	**10**	**1450**	**1.65**
Satoh et al. composite	/	1880	0	589	0.31
Our design	**2524**	**1767**	**0**	**2085**	**1.17**
Satoh et al. mux	/	2529	0	833	0.33
Our design	**3846**	**2257**	**0**	**2008**	**0.88**

6. Further research and conclusions

1. Synthesis work.
2. A good methodology may improve the efficiency of existing designs.
3. Definitions of HW efficiency, maximum pipeline, optimal pipeline.
4. Circuits size is the bottleneck for Rijndael.
5. Forthcoming research:
   (a) Very compact designs.
   (b) Combinations of encryption and decryption.
   (c) Encryption modes and algorithms.

Questions?