Compact AES S-Boxes Initialization in Low Power Non-Volatile FPGAs

New efficient S-box generation in logic area
based on a pair of LFSRs

Miloš Drutarovský, Ľuboš Gašpar

Department of Electronics and Multimedia Communications

Technical University of Košice, Slovak republic

Viktor Fischer, Nathalie Bochard

Laboratoire Traitement du Signal et Instrumentation

UMR 5516 CNRS Université Jean Monnet Saint-Etienne, France

Outline

- Overview of AES cipher implementations in FPGAs
- Most common ways for implementing S-boxes
- S-box generation in logic using pair of LFSRs
- AES cipher implementation
- Comparison of size, speed and power consumption of several S-box implementations
- Comparison of size, speed and power consumption of complete AES implementations
- Conclusions

AES cipher overview

- Symmetric cipher containing the following operations:
  - Non-linear byte substitution function (SubBytes)
  - Cyclic shift of rows (ShiftRows)
  - A mixing operation operating on columns (MixColumns)
  - Addition of a round key (AddRoundKey)
- Final performance of the cipher is mostly determined by
  - SubBytes operation
  - MixColumns operation
- Power consumption issue
  - 75 % consumed by composite S-boxes (A. Satoh et al., IBM)

Outline

- Overview of AES cipher implementation in FPGAs
- Most common ways for implementing S-boxes
- S-box generation in logic using pair of LFSRs
- AES cipher implementation
- Comparison of size, speed and power consumption of several S-box implementations
- Comparison of size, speed and power consumption of complete AES implementations
- Conclusions

SubByte implementations (1/6)

- **SubByte operation consists of the following two successive sub-operations:**
  - **Multiplicative inverse in Galois Field GF(2^8) expressed by AES irreducible polynomial**
  - **Affine transformation applied to the result of the previous operation**

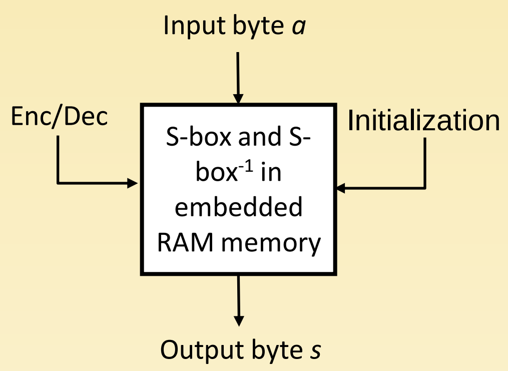
$$\mathbf{AT(c)} = \begin{pmatrix} 1&1&1&1&1&0&0&0\\ 0&1&1&1&1&1&0&0\\ 0&0&1&1&1&1&1&0\\ 0&0&0&1&1&1&1&1\\ 1&0&0&0&1&1&1&1\\ 1&1&0&0&0&1&1&1\\ 1&1&1&0&0&0&1&1\\ 1&1&1&1&0&0&0&1 \end{pmatrix} \cdot \begin{pmatrix} c_7\\ c_6\\ c_5\\ c_4\\ c_3\\ c_2\\ c_1\\ c_0 \end{pmatrix} \oplus \begin{pmatrix} 0\\1\\1\\0\\0\\0\\1\\1 \end{pmatrix}$$

SubByte implementations (2/6)

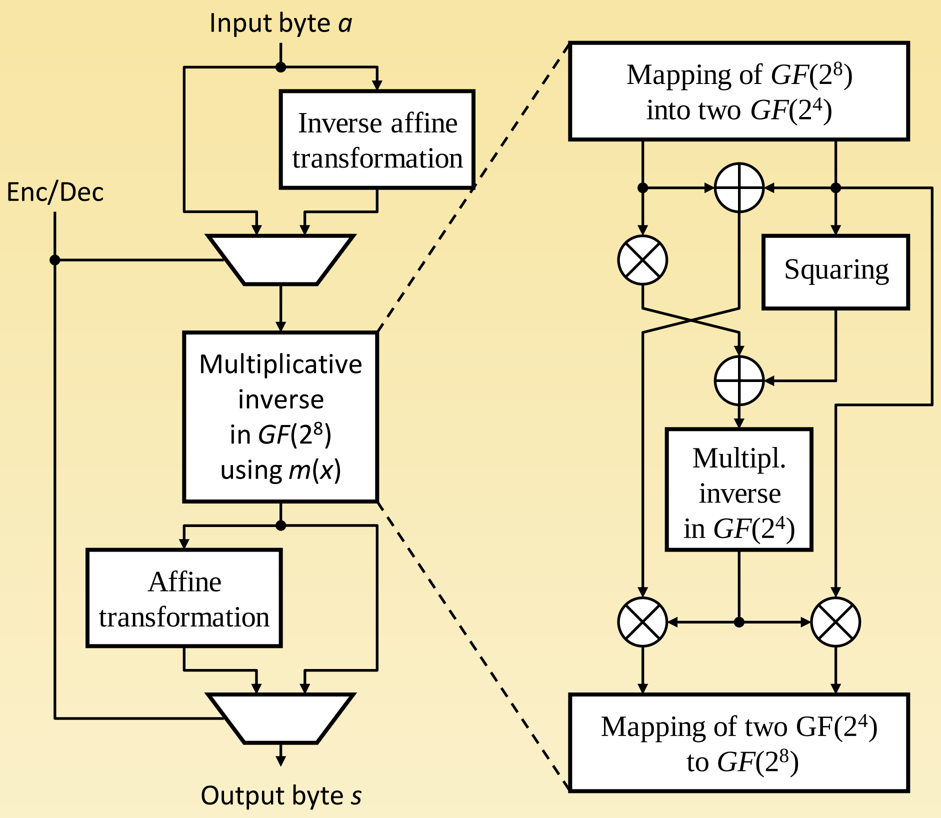
- **Two main approaches in implementation of SubByte function in hardware**
  - Using substitution tables (memory blocks) initialized with S-box values given in standard (NIST, FIPS-197)
  - Implementing the two operations of the substitution in hardware
- **Most preferred implementations of SubByte operation**
  - FPGAs: using LUTs (good logic area/memory balance, smaller delays, not suitable for pipelining)
  - ASICs: using "composite field" approach (all in logic area, bigger delays in combinatorial logic, suitable for pipelining)

SubByte implementations (3/6)

- **S-box implementation approaches**



LUT based approach



Input byte a
Inverse affine transformation
Enc/Dec
Multiplicative inverse in $GF(2^8)$ using $m(x)$
Affine transformation
Output byte s
Mapping of $GF(2^8)$ into two $GF(2^4)$
Squaring
Multipl. inverse in $GF(2^4)$
Mapping of two $GF(2^4)$ to $GF(2^8)$

"composite field" approach (later just COMB_ONLY)

SubByte implementations (4/6)

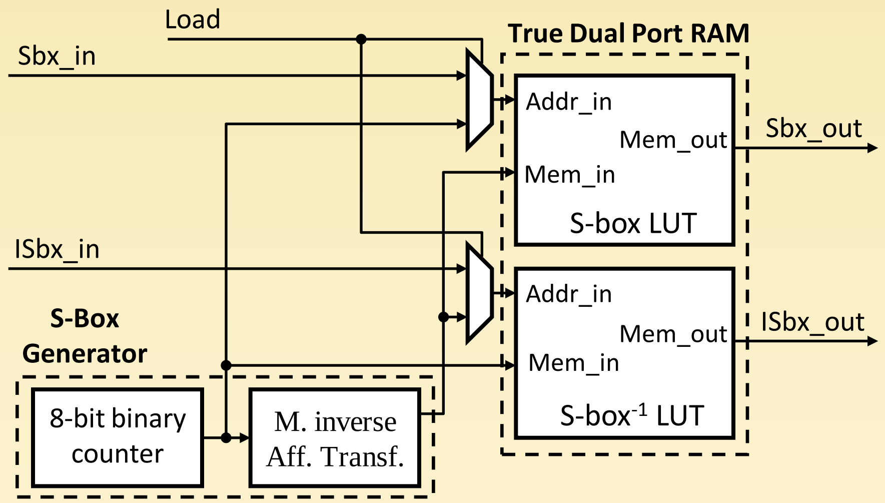
- Aim: to propose power-aware architecture of iterative AES cipher
  - Fact: Non-volatile FPGAs consume much less energy than their volatile counterparts
- When implementing LUT based solution:
  - in volatile FPGAs: initialization of S-boxes is carried out during configuration of the FPGA
  - in non-volatile FPGAs: there is no initial configuration stage!
- Problem: How to generate substitution tables inside the FPGA?

SubByte implementations (5/6)

- **Techniques for RAM initialization**
  - using JTAG interface – not practical because of a security issue
  - using embedded FLASH memory – FLASH memory is not available in every low-power device!
  - using S-box generator in the logic area
- **Interfacing the generator with a RAM memory**
  - S-box and S-box^{-1} has to be stored inside the RAM
  - using True Dual Port RAM enables both S-box and S-box^{-1} to be stored simultaneously

SubByte implementations (6/6)

- Initialization of True Dual Port RAM using S-box generator
- This S-box generator consists of 8-bit binary counter and "composite field" S-box (later just COMB_LUT)
  - Disadvantage:
    - slow (big delay)
    - occupies 170 tiles
    - ↑ power consumption
  - Better solution:
    - Using S-box generator based on a pair of LFSRs



Outline

- Overview of AES cipher implementation in FPGAs
- Most common ways for implementing S-boxes
- S-box generation in logic using pair of LFSRs
- AES cipher implementation
- Comparison of size, speed and power consumption of several S-box implementations
- Comparison of size, speed and power consumption of complete AES implementations
- Conclusions

Principle of efficient generation of S-boxes (1/9)

- **S-boxes generation requires understanding some basic properties of Galois Field**
  - Every non-zero element can be obtained as a power of primitive element α: $\alpha^0, \alpha^1, \alpha^2, \ldots, \alpha^{254}$
  - Powering of primitive element β which is the inverse to α : $\beta^0, \beta^1, \beta^2, \ldots, \beta^{254}$

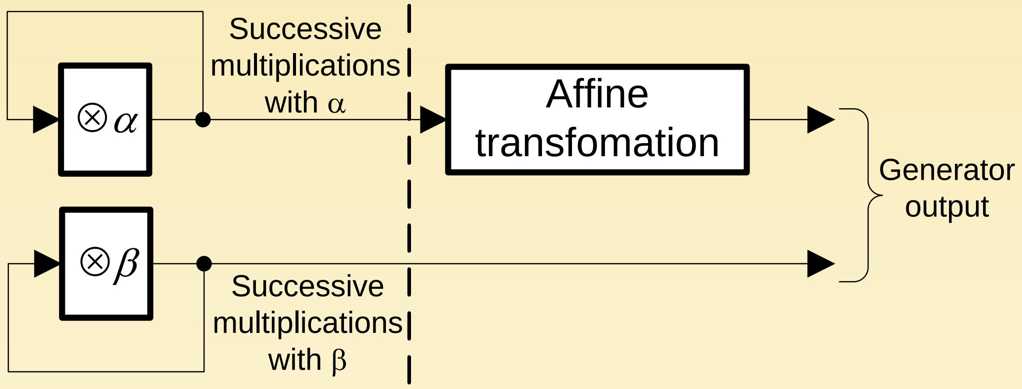
$$\beta = \alpha^{-1}$$

$$\begin{aligned}
\alpha^0 \otimes \beta^0 &= \alpha^0 \otimes \alpha^{255} = \alpha^{255} = 1 \\
\alpha^1 \otimes \beta^1 &= \alpha^1 \otimes \alpha^{254} = \alpha^{255} = 1 \\
\alpha^2 \otimes \beta^2 &= \alpha^2 \otimes \alpha^{253} = \alpha^{255} = 1 \\
&\vdots \qquad\qquad \vdots \qquad\qquad \vdots \\
\alpha^{254} \otimes \beta^{254} &= \alpha^{254} \otimes \alpha^1 = \alpha^{255} = 1 \\
\alpha^{255} \otimes \beta^{255} &= \alpha^{255} \otimes \alpha^0 = \alpha^{255} = 1
\end{aligned}$$

- **Consequences:**
  - Sequences generated as powers of α and β are inversed
  - Simultaneous powering of α and β starting with α^0 and β^0 forms multiplicatively inversed pairs

Principle of efficient generation of S-boxes (2/9)

- Powering can be represented as multiplications in the ring
- For complete S-box generator, Affine transformation has to be applied on one of the rings
  - Each ring can be implemented as 8-bit register with combinatorial multiplier in the feedback


$\otimes\alpha$
Successive multiplications with α
Affine transfomation
$\otimes\beta$
Successive multiplications with β
Generator output

Principle of efficient generation of S-boxes (3/9)

- **The smallest primitive element in the $GF(2^8)$ expressed by AES irreducible polynomial**

$$m(x) = x^8 + x^4 + x^3 + x + 1$$

is $\alpha = 03_{16}$ and its inverse is $\beta = \alpha^{-1} = F6_{16}$

- **From other point of view, implementation of such multipliers in combination with registers results in a pair of 8-bit Non-Linear Feedback Shift Registers (NLFSRs)**

Principle of efficient generation of S-boxes (4/9)

- Lets try to improve the principle such that the implementation results in **a pair of LFSRs in series with combinatorial block**
- Let A be one root of m(x), the polynomial basis will be

$$[A^7, A^6, A^5, A^4, A^3, A^2, A^1, A^0],$$

where $A = 02_{16}$ is the root of AES irreducible polynomial

- However, 02_{16} is not a primitive element

Principle of efficient generation of S-boxes (5/9)

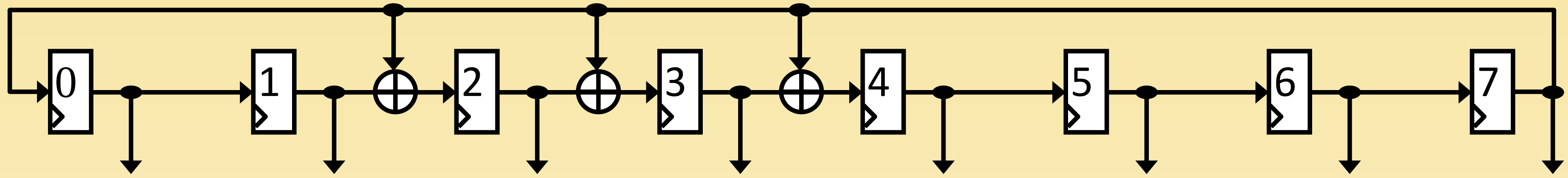
- **Lets examine GF(2^8) expressed by the smallest primitive polynomial of the 8th degree**

$$m'(x) = x^8 + x^4 + x^3 + x^2 + 1$$

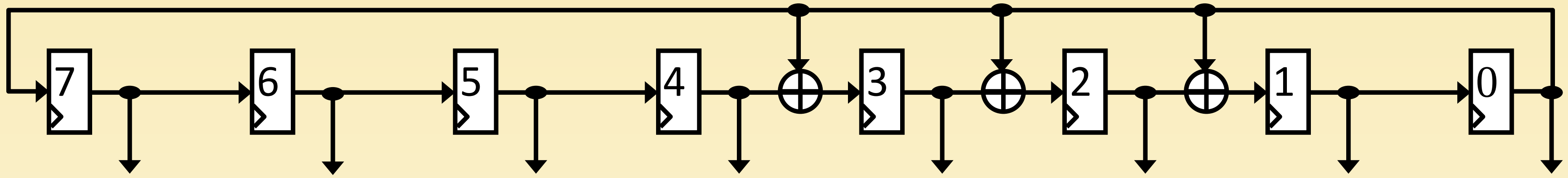
  - **Element 02_{16} is a primitive element**
  - **Element 03_{16} is the root**

- **Thus, in this field $\alpha = 02_{16}$ and its inverse $\beta = \alpha^{-1} = 8E_{16}$**
- **Implementation leads to a pair of LFSRs**

Principle of efficient generation of S-boxes (6/9)



– LFSR with primitive element $\alpha = 02_{16}$ generating $GF(2^8)$



– LFSR with primitive element $\beta = 8E_{16}$ generating $GF^{-1}(2^8)$

Principle of efficient generation of S-boxes (7/9)

- However output of these LFSRs have to be **transformed** from GF(2^8) expressed by m'(x) to GF(2^8) expressed by AES irreducible polynomial m(x)
- Our **Basis Transformation** (BT) has the following form

$$\mathbf{BT(d)} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} d_7 \\ d_6 \\ d_5 \\ d_4 \\ d_3 \\ d_2 \\ d_1 \\ d_0 \end{pmatrix} = \begin{pmatrix} d_7 \\ d_7 \oplus d_6 \\ d_7 \oplus d_5 \\ d_7 \oplus d_6 \oplus d_5 \oplus d_4 \\ d_7 \oplus d_3 \\ d_7 \oplus d_6 \oplus d_3 \oplus d_2 \\ d_7 \oplus d_5 \oplus d_3 \oplus d_1 \\ d_7 \oplus d_6 \oplus d_5 \oplus d_4 \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \end{pmatrix}$$

- This Basis Transformation has to be implemented as a combinatorial block in series with the LFSRs

Principle of efficient generation of S-boxes (8/9)

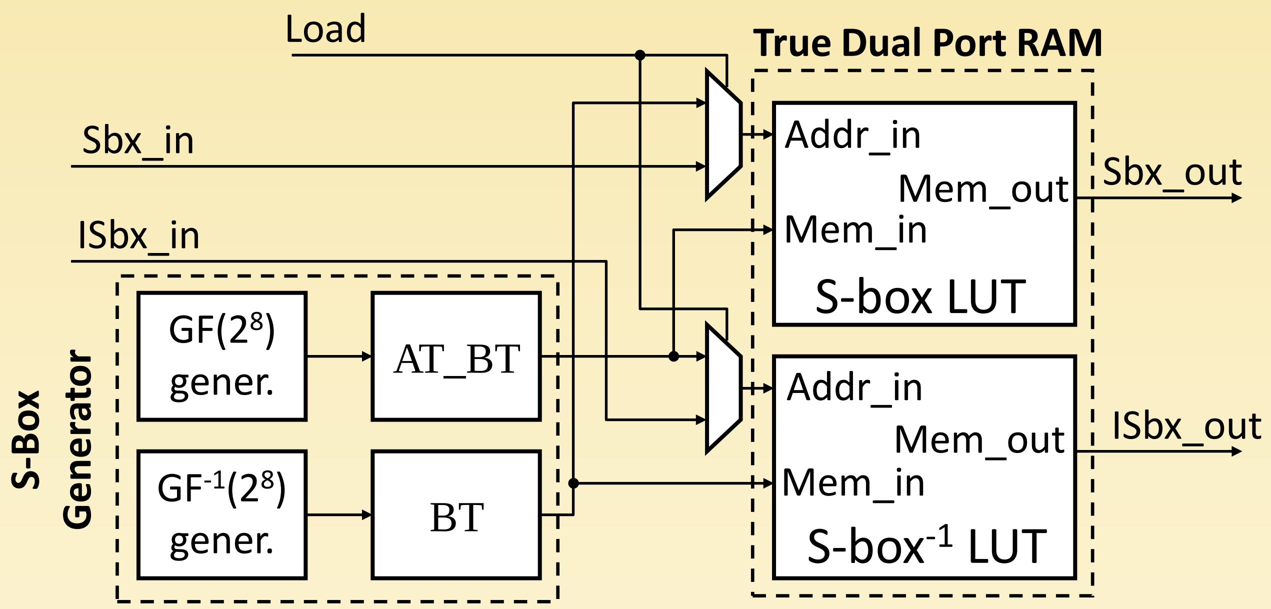
- Moreover, Affine transformation has to be applied at the output of one block of the Basis transformation to obtain complete generator
- Both transformation can be combined together obtaining Combined Affine and Basis Transformation

$$\mathrm{AT\_BT(d)} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} d_7 \\ d_6 \\ d_5 \\ d_4 \\ d_3 \\ d_2 \\ d_1 \\ d_0 \end{pmatrix} \oplus \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} d_7 \oplus d_4 \oplus d_3 \\ (d_7 \oplus d_6 \oplus d_4 \oplus d_2)' \\ (d_7 \oplus d_5 \oplus d_4 \oplus d_3 \oplus d_2 \oplus d_1)' \\ d_7 \oplus d_6 \oplus d_5 \oplus d_0 \\ d_7 \oplus d_4 \oplus d_0 \\ d_7 \oplus d_6 \oplus d_4 \oplus d_3 \oplus d_0 \\ (d_7 \oplus d_5 \oplus d_4 \oplus d_2 \oplus d_0)' \\ (d_7 \oplus d_6 \oplus d_5 \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0)' \end{pmatrix}$$

- Operation ()’ represents logic negation

Principle of efficient generation of S-boxes (9/9)

- The **final implementation** of the S-box based on a pair of LFSRs + transformation blocks (later just **LFSR_LUT**)



Outline

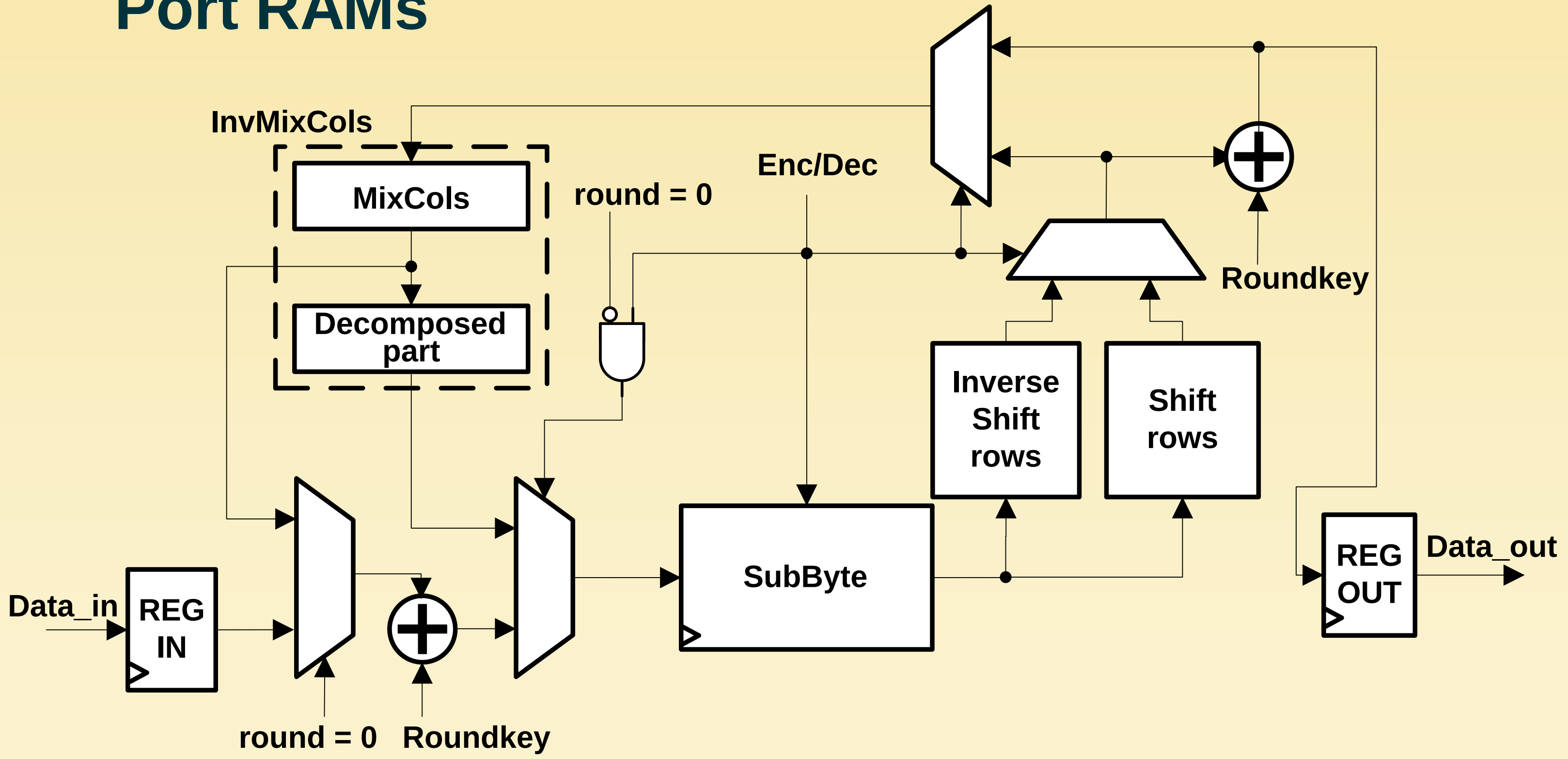
- Overview of AES cipher implementation in FPGAs
- Most common ways for implementing S-boxes
- S-box generation in logic using pair of LFSRs
- AES cipher implementation
- Comparison of size, speed and power consumption of several S-box implementations
- Comparison of size, speed and power consumption of complete AES implementations
- Conclusions

AES cipher implementation (1/2)

- **Aim:**
  - implementation of shared AES cipher/decipher
  - advanced resource sharing between cipher and decipher:
    - AddRoundKey operation
    - SubByte operation
    - InvMixColumns shared with MixColumns (InvMixColumns serial decomposition)
    - Forward key expansion share resources with backward key expansion
    - Input and Output registers

AES cipher implementation (2/2)

- Shared AES cipher/decipher implementation
  - 16 SubByte blocks implemented using 8 True Dual Port RAMs



Outline

- Overview of AES cipher implementation in FPGAs
- Most common ways for implementing S-boxes
- S-box generation in logic using pair of LFSRs
- AES cipher implementation
- Comparison of size, speed and power consumption of several S-box implementations
- Comparison of size, speed and power consumption of complete AES implementations
- Conclusions

S-box implementations comparison (1/4)

- **Compared implementations**
  - COMB_ONLY: realized entirely in logic area
  - COMB_LUT: employed 4-kbit True Dual Port RAM
  - LFSR_LUT: employed 4-kbit True Dual Port RAM
- **Each port** of 4-kbit True Dual Port RAM block can implement two independent dual S-boxes used in parallel
  - For fare comparison it is necessary to implement two parallel S-boxes in COMB_ONLY solution too
- **Used hardware platform**
  - Actel Igloo AGL600 device

S-box implementations comparison (2/4)

- area and speed values and the power consumption estimation given for the maximum module frequencies
- COMB_LUT and LFSR_LUT concern S-box substitution table generation

	Area	f_{max}	Power@f_{max} (mW)			
	(Tiles)	(MHz)	Tot	Mem	Gate	Net
COMB_ONLY	274	29.4	1.1	-	0.3	0.8
COMB_LUT	170	35.7	1.2	0.2	0.2	0.8
RAM	-	-	0.3	0.2	0.0	0.1
S-box	-	-	0.7	-	0.2	0.5
LFSR_LUT	75	73.5	1.2	0.2	0.2	0.6
RAM	-	-	0.3	0.2	0.0	0.1
S-box	-	-	0.5	-	0.1	0.4

S-box implementations comparison (3/4)

- estimation given for the 24 MHz module frequency

	Power@24 MHz (mW)			
	Tot	Mem	Gate	Net
COMB_ONLY	0.9	-	0.2	0.7
COMB_LUT	0.8	0.1	0.2	0.5
RAM	0.2	0.1	0.0	0.1
S-box	0.4	-	0.1	0.3
LFSR_LUT	0.4	0.1	0.1	0.2
RAM	0.2	0.1	0.0	0.1
S-box	0.2	-	0.1	0.1

S-box implementations comparison (4/4)

- **Speed concerns**
  - In **LUT based** solutions: after initialization, the speed of the operation is limited only by the maximum frequency of a RAM
  - In **COMB_ONLY**: the speed of the operation is limited by the delays of logic elements and routes
  - Generation of S-box table with LFSR_LUT solution can be performed almost two times faster than with COMB_LUT
- **Area concerns**
  - **LFSR_LUT** solution utilizes only **75 tiles** when compared to 170 tiles utilized by COMB_LUT solution and 274 tiles utilized by COMB_ONLY solution
- **Power concerns**
  - **LFSR_LUT** solution consumes **half** the power when compared with two other solutions

Outline

- Overview of AES cipher implementation in FPGAs
- Most common ways for implementing S-boxes
- S-box generation in logic using pair of LFSRs
- AES cipher implementation
- Comparison of size, speed and power consumption of several S-box implementations
- Comparison of size, speed and power consumption of complete AES implementations
- Conclusions

AES implementations comparison (1/6)

- **Implementation details**
  - USB interface included to source AES 128-bit data and 128-bit key inputs
- **Hardware platform**
  - Actel Igloo AGL600 device - low-power device
  - Actel Fusion AFS600 device - high-performance device
  - Altera Cyclone III C5E144 - low-cost SRAM based device

AES implementations comparison (2/6)

- Implementation results of the AES cipher including USB interface in the Altera Cyclone III C5E144 device
  - **Results estimated by Quartus PowerPlay Power Analyser**

	USB only	AES & USB LFSR_LUT
f_{max} (MHz)	-	111.4
Total logic elements	463	1705
ALMs	455	1678
Regs	102	505
Memory blocks	0	10
Power@24MHz (mW)	-	82.2
Power@0MHz (mW)	-	53.1

AES implementations comparison (3/6)

- Implementation results of the three AES cipher versions including USB interface in the Actel Fusion AFS600 device
  - **Results measured on the Actel Fusion evaluation board**

	USB only	AES & USB		
		COMB	COMB_LUT	LFSR_LUT
f_{max} (MHz)	-	28.7	45.8	58.5
Tiles: Total	618	5499	3096	2987
Combin.	288	4569	2297	2171
Sequen.	330	930	799	816
Memory blocks	0	0	10	10
Power@24MHz (mW)	-	125.3	57.2	59.3
Power@10MHz (mW)	-	66.4	30.8	31.7
Power@0MHz (mW)	-	8.5	8.2	8.2

AES implementations comparison (4/6)

- Implementation results of the three AES cipher versions including USB interface in the Actel Igloo AGL600 device
  - **Results measured on the Actel Igloo evaluation board**

	USB only	AES & USB		
		COMB	COMB_LUT	LFSR_LUT
f_{max} (MHz)	-	11.1	18.9	24.3
Tiles: Total	623	5721	3198	3087
Combin.	293	4791	2397	2271
Sequen.	330	930	801	816
Memory blocks	0	0	10	10
Power@24MHz (mW)	-	-	-	36.8
Power@10MHz (mW)	-	43.7	15.7	16.9
Power@0MHz (mW)	-	0.3	0.3	0.3

AES implementations comparison (5/6)

- Speed concerns
  - The proposed AES with LFSR_LUT solution more than doubles the maximum frequency for both Actel families in contrast to other two solutions
  - This implementation in Altera Cyclone III is 4x faster and in Actel Fusion is 2x faster than in Actel Igloo
- Area concerns
  - AES with COMB_ONLY solution in Actel Fusion and Actel Igloo utilizes 2x more tiles than AES with LFSR_LUT solution
- Power concerns
  - In both Actel devices, the two LUT-based AES solutions give comparable results, while the AES with COMB_ONLY solution is about two times more power expensive
  - Reason: S-boxes implemented in RAM are more power-efficient than those implemented in the logic area

AES implementations comparison (6/6)

- Actel Igloo family consumes
  - 2x less dynamic power
  - 150x less static power

  than the most economic Altera family with a comparable size.

- It is therefore clear, that stopping the clock (e.g. in a stand-by mode) will reduce the power consumption to a negligible value, while maintaining the cipher internal state.

Outline

- Overview of AES cipher implementation in FPGAs
- Most common ways for implementing S-boxes
- S-box generation in logic using pair of LFSRs
- AES cipher implementation
- Comparison of size, speed and power consumption of several S-box implementations
- Comparison of size, speed and power consumption of complete AES implementations
- Conclusions

Conclusion

- Unique S-box generation method has been proposed while maintaining very low power consumption, small size and high speed
- The implementation results in the Actel Igloo family show that the proposed architecture could also be used in battery-powered mobile applications, which was not the case in FPGA applications up to now

Thank you for your time and attention