A High-Performance VLSI Architecture for Advanced Encryption Standard (AES) Algorithm

N. M. Kosaraju, M. Varanasi & Saraju P. Mohanty
VLSI Design and CAD Laboratory
Homepage: http://www.vdcl.cse.unt.edu
University of North Texas, Denton, TX, USA.
Email: smohanty@cs.unt.edu

Outline of the Talk

- Introduction
- The Rijndael Algorithm
- Related Work
- Proposed Architecture
- Prototype Implementation
- Performance Analysis
- Conclusions

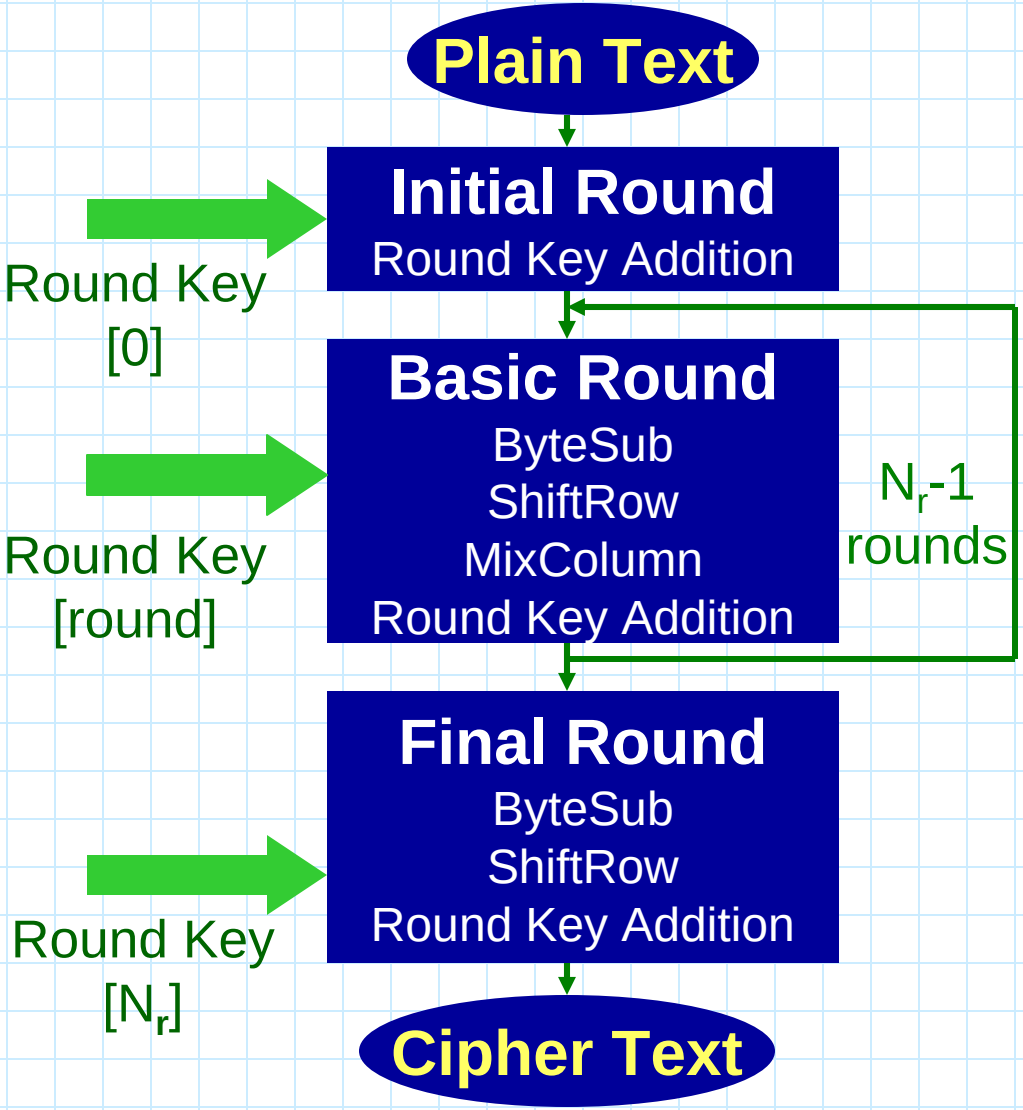
Introduction

- Techniques like cryptography, steganography, watermarking, and scrambling, have been developed to keep data secure, private, and copyright protected.
- The need for secure transactions in ecommerce, private networks, and secure messaging has moved encryption into the commercial realm.
- In October 2000, Rijndael, developed by Joan Daemen and Vincent Rijmen was announced as the new encryption standard replacing data encryption standard (DES).

Rijndael Algorithm

- Rijndael algorithm is an iterative, symmetric block cipher with variable block length and variable key length.
- The number of rounds in the algorithm depends on the block length and the key length.
- The block length is specified to 128 bits (by NIST) and the key length can be either 128, 192 or 256 bits.
- The data block (B) and the key (K) are split into array of bytes (called State) and are represented in matrices arranged in a column major order.

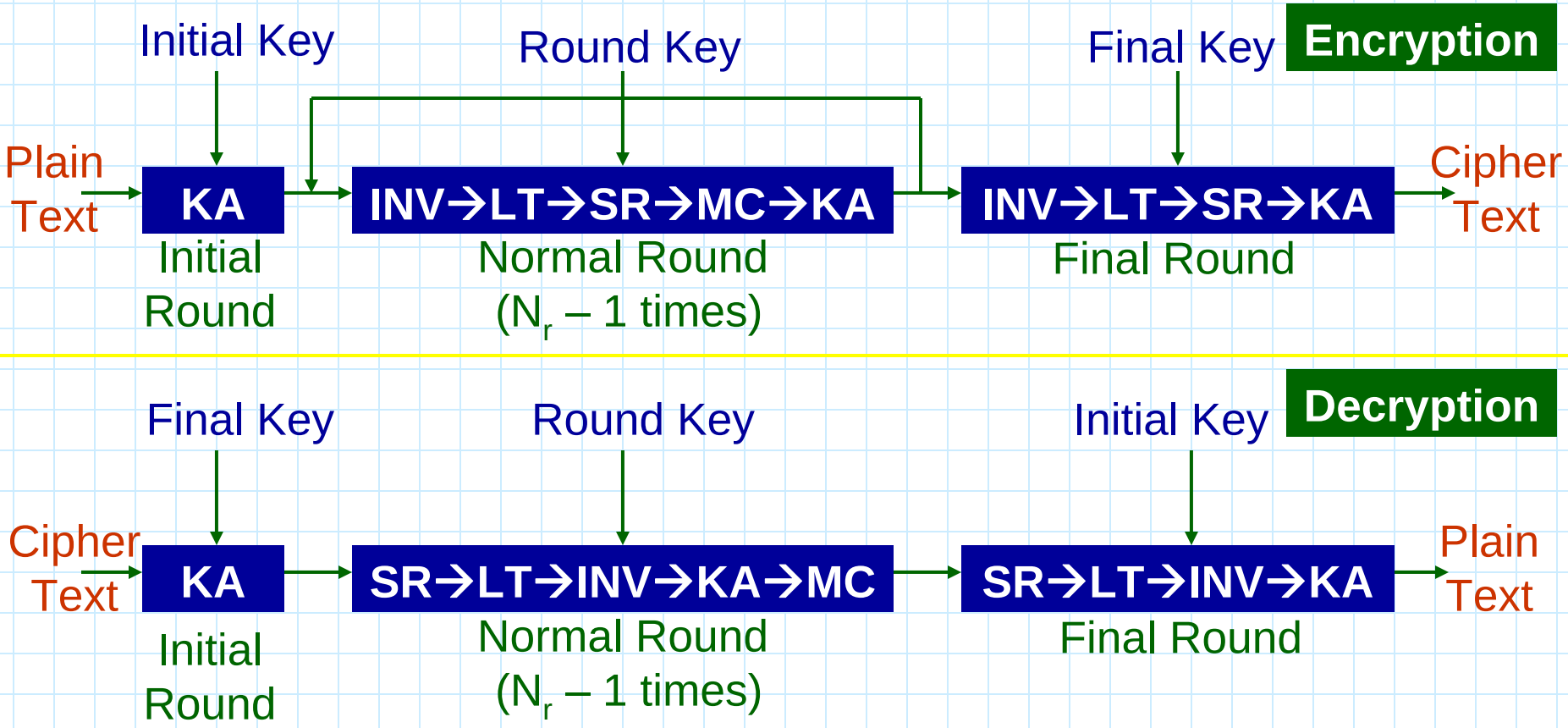
Principle of Rijndael Algorithm



3 distinct phases

1. An initial data/key addition.
2. Nine (128-bits), eleven (192-bits) or thirteen (256-bits) standard rounds. Each round has a new round key with expanded key length $N_b(N_r-1)$.
3. A final round

Order of Operations in Encryption and Decryption



INV → Multiplicative Inverse, LT → Linear Transformation, SR → Shift Row, MC → Mix Column, KA → Key Addition, Nr = 10 for 128-bit input, key length

Related Work (Architectures for Rijndael)

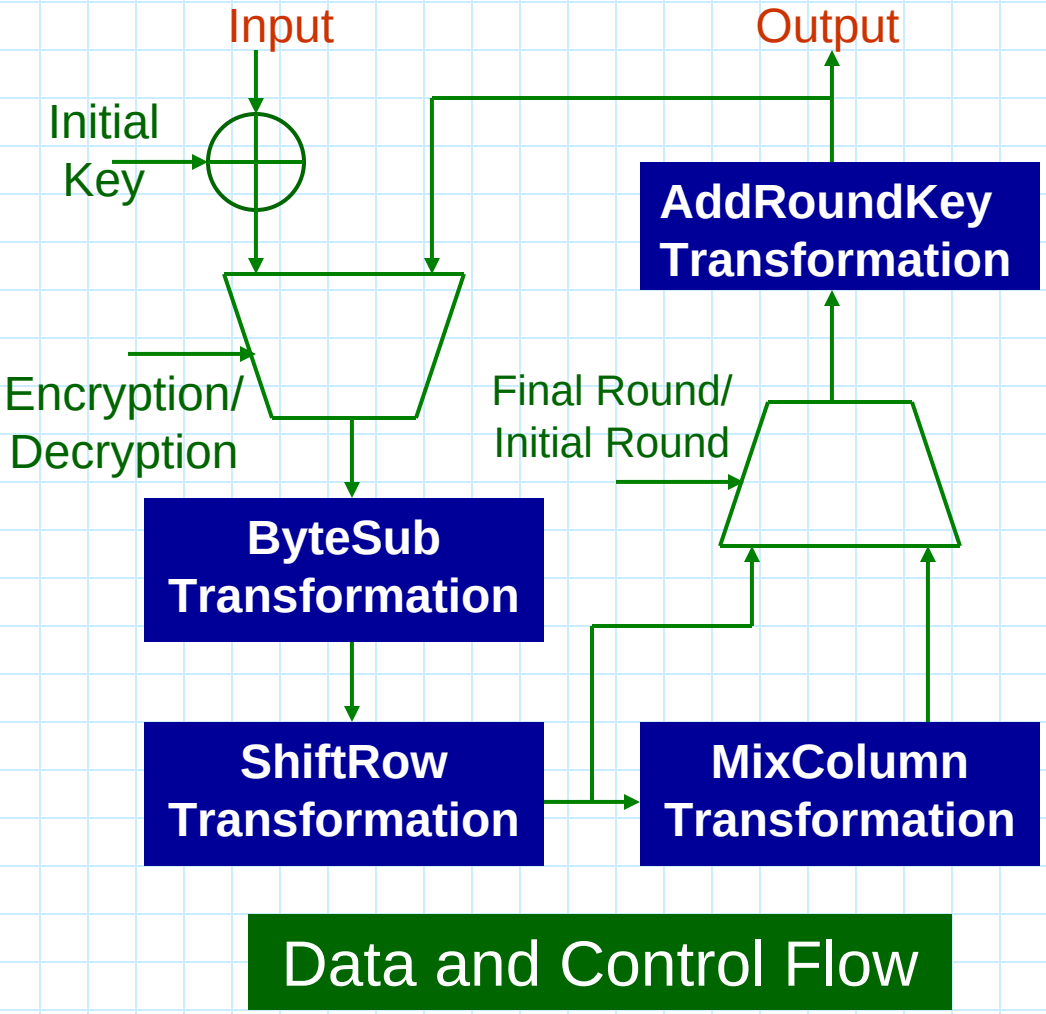
- **Kuo and Verbauwhede [2001]:**
  - Encryption module to generate intermediate data.
  - A key scheduling module to generate the round keys.
  - Data encryption done at a rate of 1.82*Gbps*.
- **McLoone and McCanny [2001]:**
  - High performance single - chip FPGA implementation.
  - Supports different key sizes.
  - 192-bit key design run at 5.8*Gbps &* 256-bit key design run at 5.2*Gbps*.
- **Mangard, et. al. [2003]:**
  - Combinational paths are relatively short and balanced.
  - S-boxes have pipelined implementation using combinational logic.
  - The high performance versions achieved 241*Mbps*.
- **Sodon, et. al. [2005]:**
  - A low cost architecture using bit-serial approach.
  - FPGA based prototype has a maximum clock frequency of 510*MHz* with a throughput of 0.37*Gbs.*

Salient Features of our Architecture

- A high performance, high throughput and area efficient VLSI architecture.
- Architecture is optimized for high throughput in terms of the encryption and decryption data rates using pipelining.
- Polynomial multiplication is implemented using XOR operation instead of using multipliers to decrease the hardware complexity.
- Both encryption and decryption modes use common hardware resources, thus making the architecture and corresponding implementation area efficient.
- Selective use of look-up tables and combinational logic further enhances the architecture's memory optimization, area, and performance.
- An effective solution of online (real-time) round key generation needing significantly less storage for buffering.

Architecture : Data and Control Flow

- Architecture consists of data unit and key unit.
- Data Unit: (i) byte substitution, (ii) shift row, (iii) mix column, and (iv) round key addition
- Key Unit: Key scheduling and expansion.



Architecture : Modes of Operation

- Different modes of operations in which the block cipher algorithm can be implemented are :
  - Electronic Code Book Mode (ECB)
  - Cipher Back Chaining Mode (CBC)
  - Cipher Feed-Back Mode (CFB)
  - Output Feed-Back Mode (OFB)
- For modes with feedback operations, pipelined design has no additional advantage since the encryption depends on the previous results.
  - ECB mode of operation is chosen for our implementation.

Architecture: Pipelining and Looping

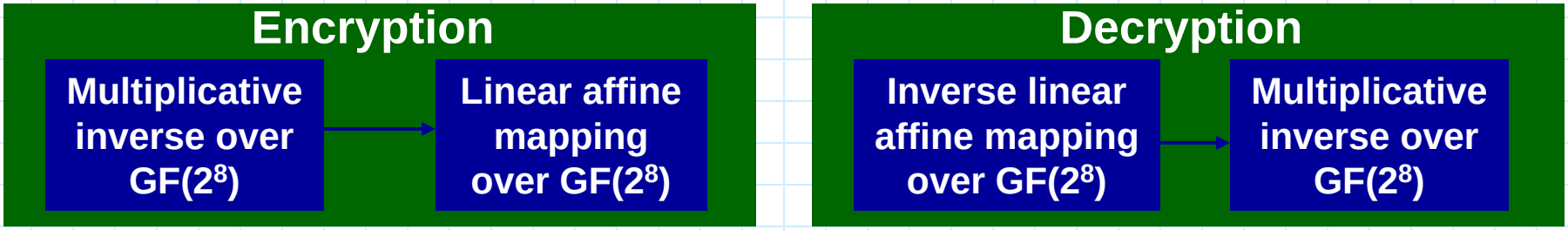
- The Rijndael algorithm is implemented in hardware considering the basic concepts:
  - Pipelining: Replicating same rounds and placing registers in between.
    - Advantage: Increases the throughput.
  - Iterative Looping: One round of hardware design, which forces the algorithm to reuse the same hardware.
    - Advantage: Reduces the amount of area.

Operations Needed in Architecture

1. Byte Substitution Transformation
2. Shift Row Transformation
3. Mix Column Transformation
4. Key Addition Transformation

Architecture : Byte substitution

- The Byte Substitution transformation is applied to each byte individually and is a nonlinear byte-wise substitution. It consists of two phases:
  - Multiplicative inverse of a state byte in GF(2^8)
  - An affine/inverse affine mapping transformation over GF(2) for encryption/decryption



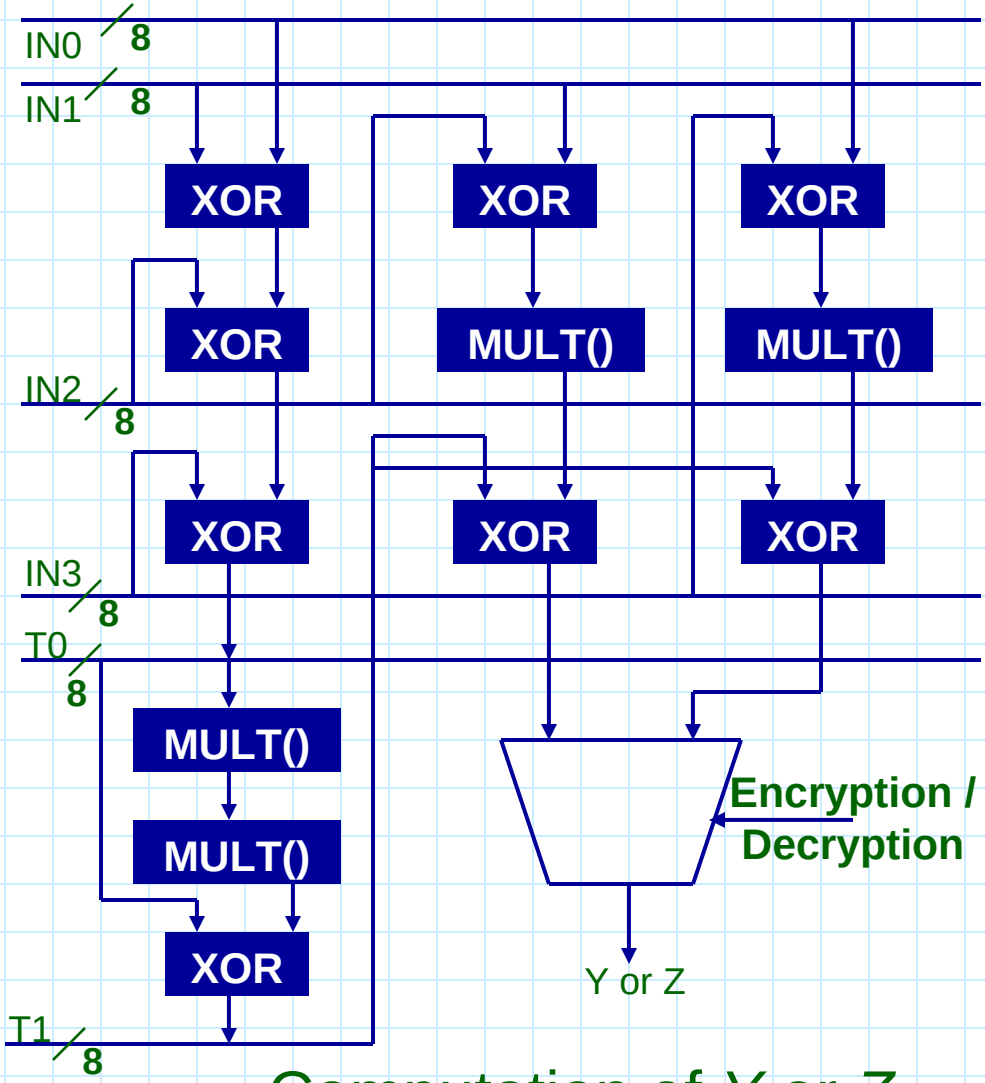
Architecture : Shift Row

- The rows of the state matrix are cyclically shifted to the left during encryption and to the right during decryption by certain offset for each row.
  - For a data block of length 128-bits, the offsets for each row are as follows:
    - Row 0 is shifted by 0 bytes
    - Row 1 is shifted by 1 byte
    - Row 2 is shifted by 2 bytes
    - Row 3 is shifted by 3 bytes
  - Shift Row transformation is implemented using combinational logic instead of look-up tables which allows for area minimization

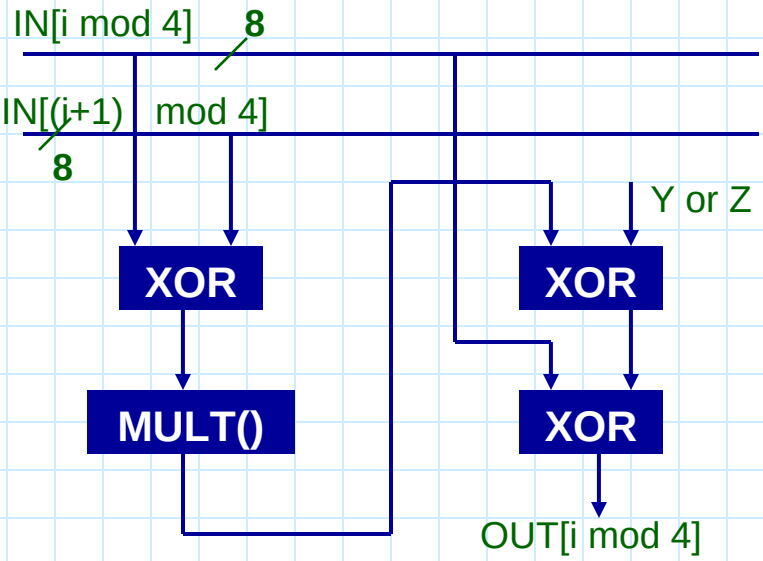
Architecture : Mix Column

- Mix Column transformation is applied to columns of the state matrix, each column being considered as a polynomial over GF(2^8).
  - During encryption, each column is multiplied by a fixed polynomial.
  - During decryption, each column is multiplied by a fixed polynomial.
- The multiplication by fixed polynomials over GF(2^8) is implemented using XOR operation instead of the multipliers.
- The inverse mix column transformation is more complex than the mix column transformation, as the coefficients involved in the decryption polynomial are of higher order.

Architecture : Mix Column



Computation of *Y* or *Z*

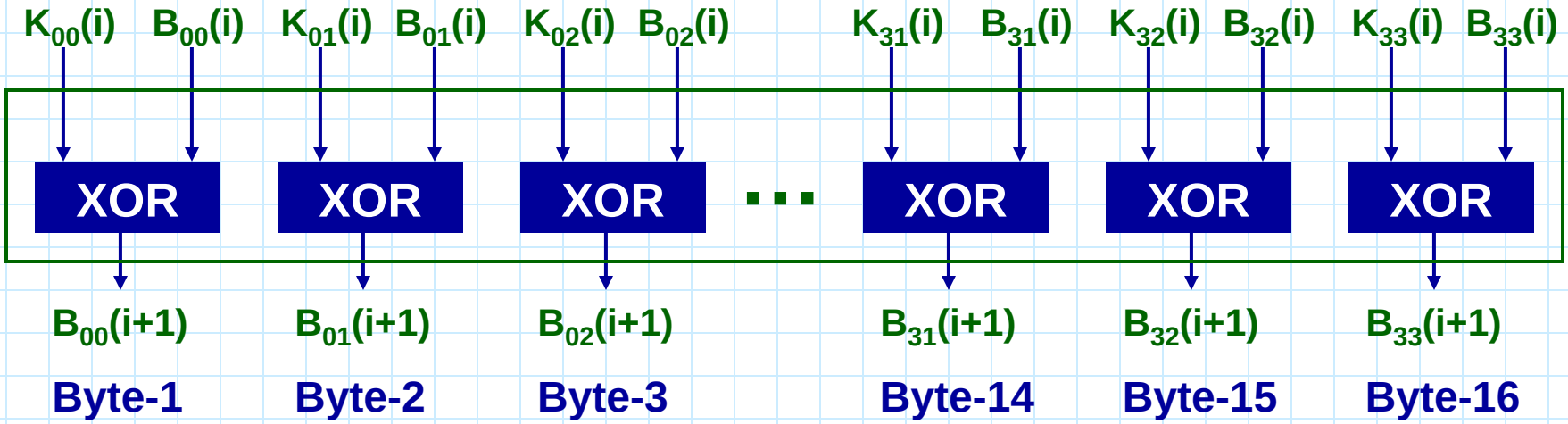


Computation of Output

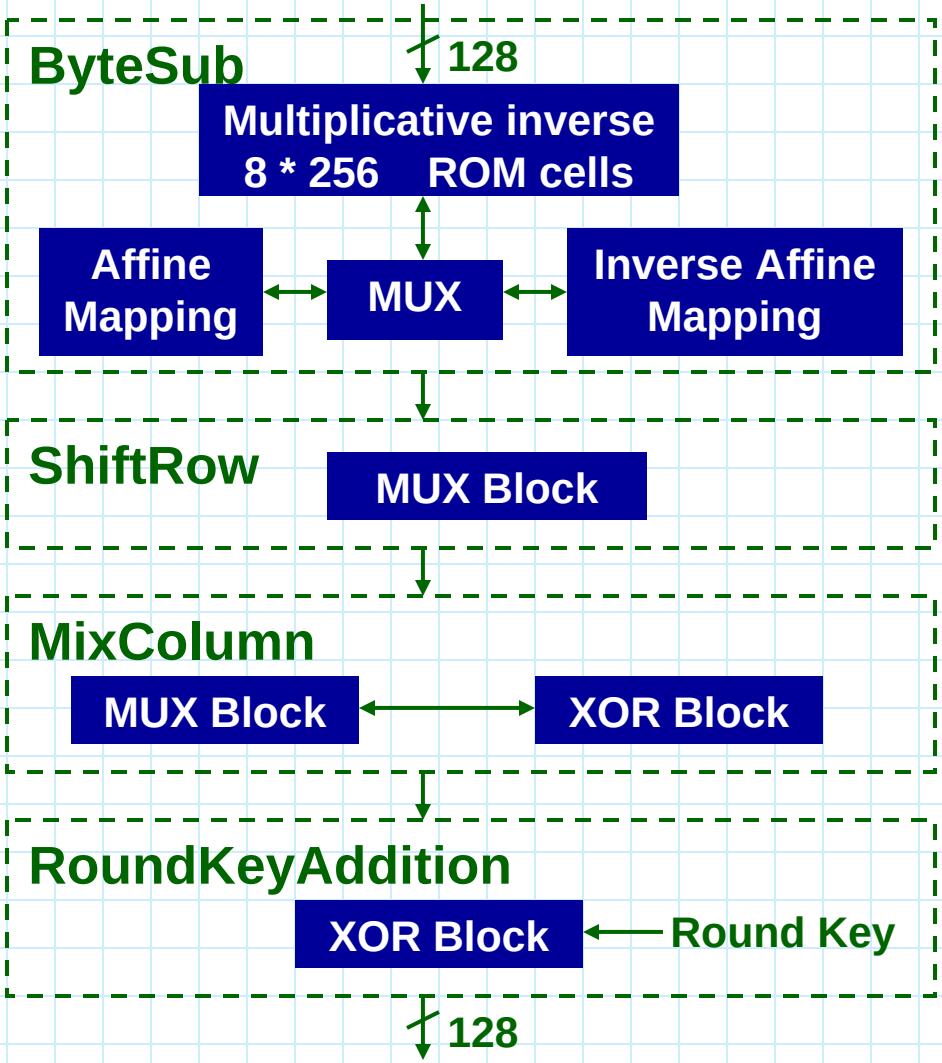
NOTE: MULT() is the multiplication of the byte by X over GF(2^8).

Architecture : Round Key Addition

- The state bytes and the appropriate round key generated by the key scheduling module are XORed.



Architecture : Different Rounds

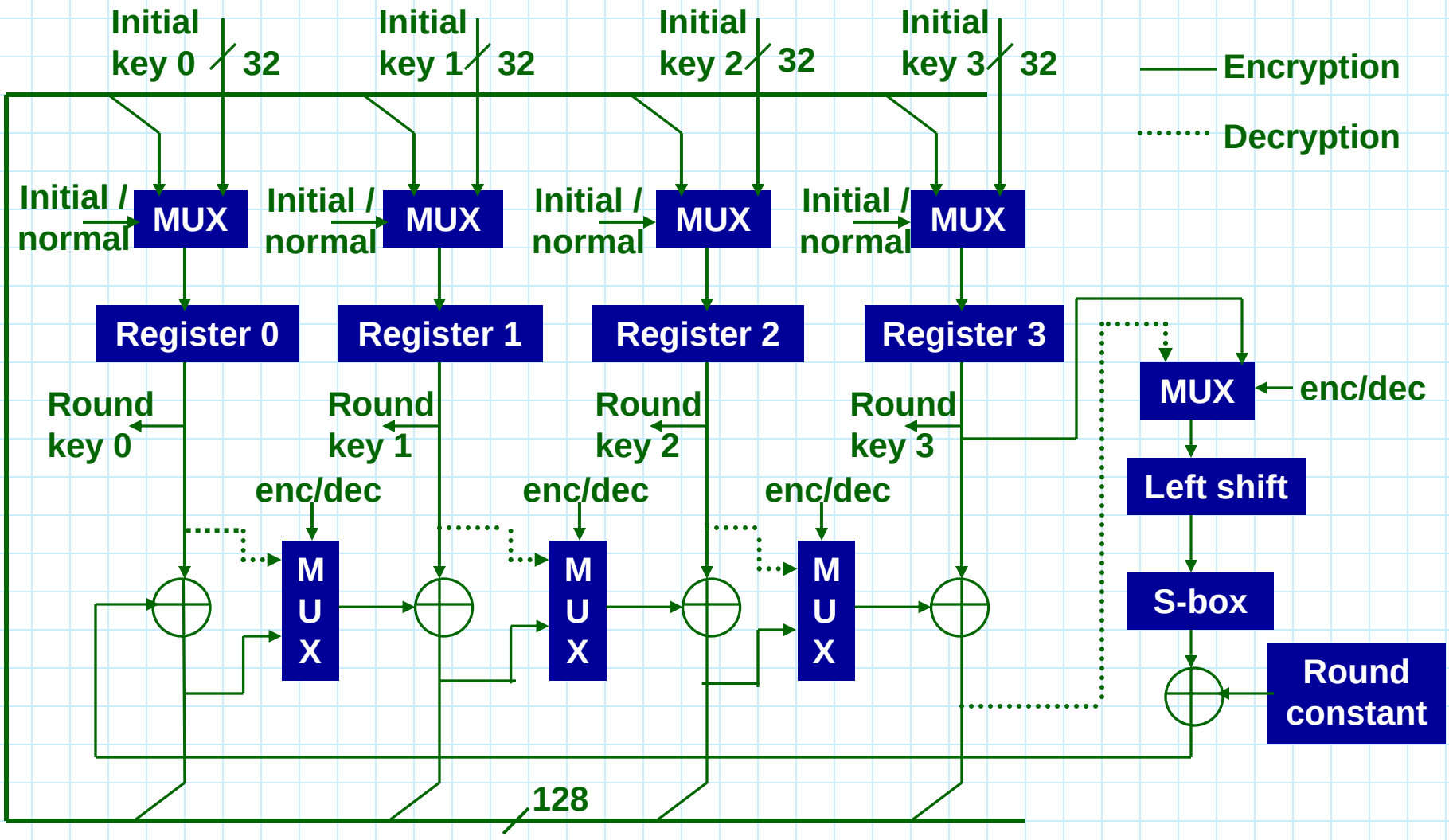


- Standard round architecture has all four transformations:
  1. Byte Substitution
  2. Shift Row
  3. Mix Column
  4. Round Key Addition
- Initial Round has (4)
- Final Round has (1), (2), (4)

Architecture : Key Generation

- Key Generation has two parts
  - Key Expansion
    - The initial key is represented as a linear array W, where $K_0=(W_0,W_1,W_2,W_3)$
    - The initial key is expanded into a linear array of 32-bit words of length $N_b * (N_r -1)$.
  - Key Scheduling
    - A round key of length 128 bits generated in every clock cycle is given as input to the data unit of the encryption/ decryption module.

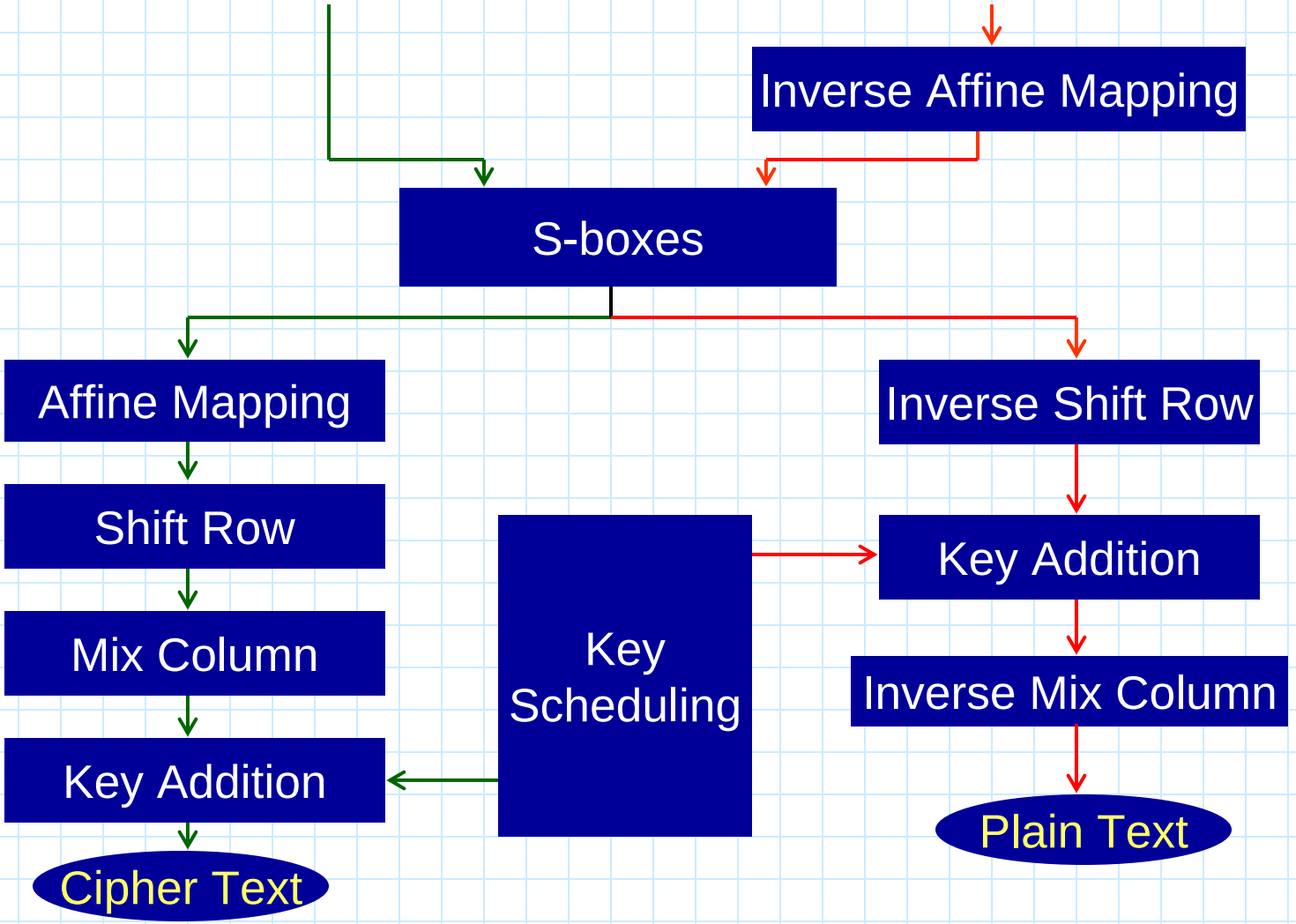
Architecture : Key Generation



Architectural Analysis

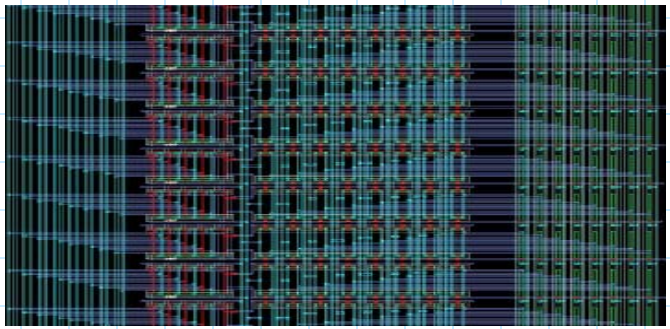
- The forward and the reverse key scheduling is implemented on the same device, thus allowing for area minimization.
- The generation of round key for each round takes 1 clock cycle.
- Decryption requires more cycles than encryption because it needs pre-scheduling to generate the last key value and the Inverse Mix Column transformation has a longer critical path compared to the Mix Column transformation.
- Round Keys are generated during the process when required, thus reducing the amount of storage for the buffer.
- Some of the modules need to be duplicated to get all the required operations done in one clock cycle for one round.

Resource Sharing between Encryption and Decryption

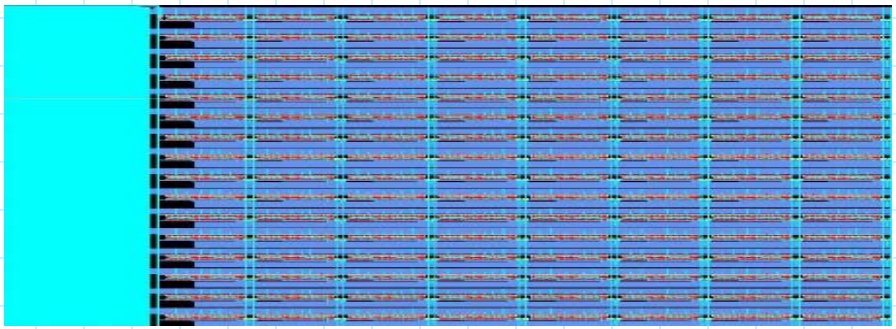


Prototype Implementation : Layouts

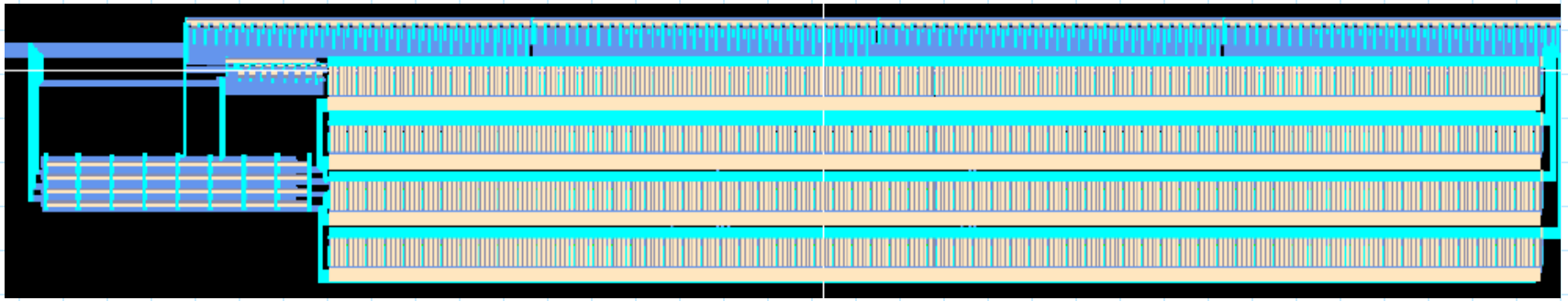
- The proposed architecture is custom designed using Cadence Virtuoso design layout with 0.35µ CMOS technology.
- The simulation tools used are Hspice.



Multiplicative Inverse



Affine and Inverse Affine Mapping



Key Alignment

Prototype Implementation : Summary

Module / Component	Our Architecture	Mangard et al. [7]
Data Unit		
S-Boxes	16	16
32-bit Registers	8	16
Multiplexers	240	384
32-bit Multiplexers	180	NA
128-bit Multiplexers	60	NA
Multipliers	0	16
Key Unit		
S-Boxes	4	NA
32-bit Registers	4	NA
32-bit Multiplexers	4	NA

Prototype Implementation : Performance

Architecture	Clock Cycles	Throughput (Mbps)
Proposed Architecture	11	232
Mangard et al [7] - Standard	64	128
Mangard et al [7] – High Performance	34	241

- Throughput = (Block length * Clock Frequency) / (Cycles per Block).
- Pipelined version of our architecture has throughput of 1.83Gbps

Conclusions

- A VLSI architecture for the Rijndael, AES algorithm is presented.
- The key length and the data block length are specified to 128 bits.
- Feedback and pipelining architectures were used for the implementation.
- The algorithm was implemented in the ECB mode of operation.
- Pipelined architecture could process data at 1.83 Gbits/sec

Thank You!