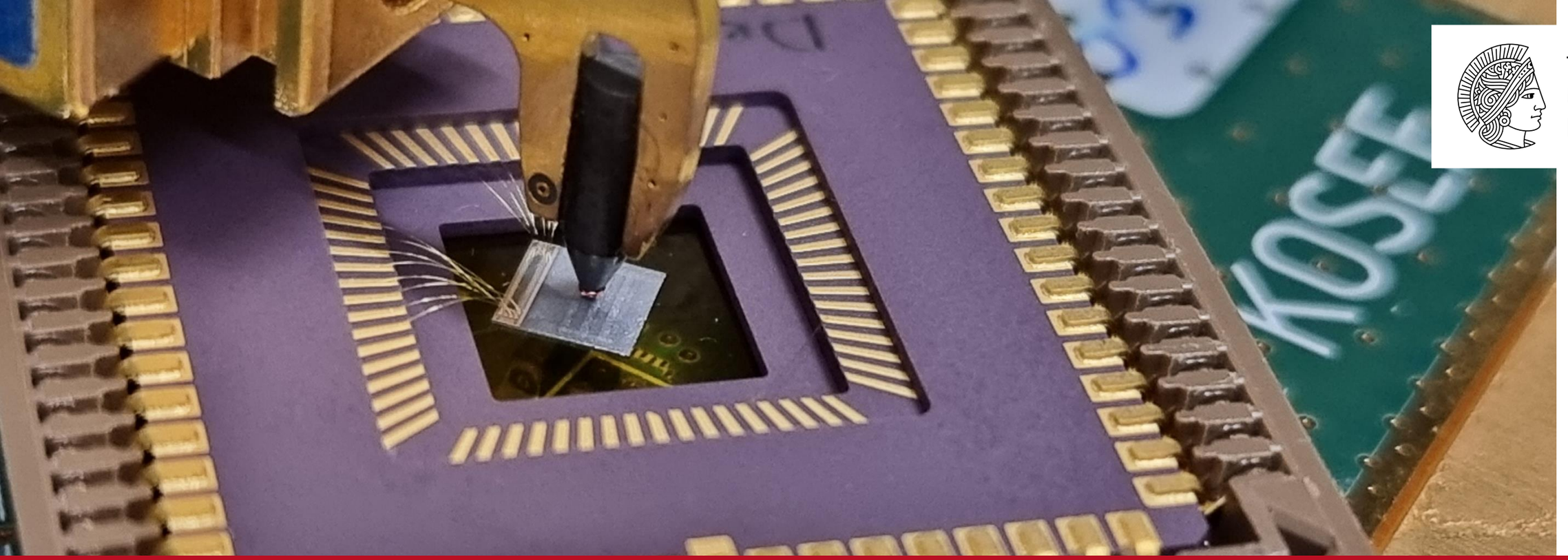


TECHNISCHE UNIVERSITÄT DARMSTADT

Physical Attacks and Countermeasures

2: Introduction

amir.moradi@tu-darmstadt.de

Agenda

- Cryptography and Cryptographic Devices
- Examples of Attacks on Cryptographic Devices
- Physical Attacks
- Basics of Hardware Design

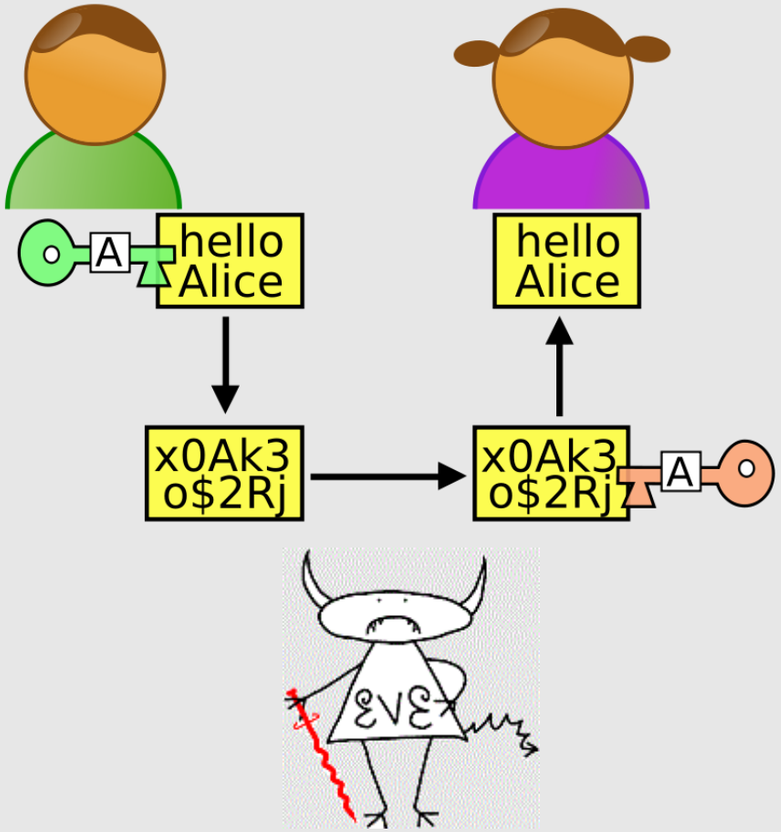
Cryptography

- is the science of writing in secret (too simple?)
  - is the scientific study of techniques for securing digital information, transactions and computations
  - It MUST provide
    - Integrity
    - Confidentiality
    - Authenticity
    - Non-Repudation

Classical Scenario

- Alice and Bob are honest parties and want to communicate

- The communication link between them is not secure
- Eve can listen to the communication



Security

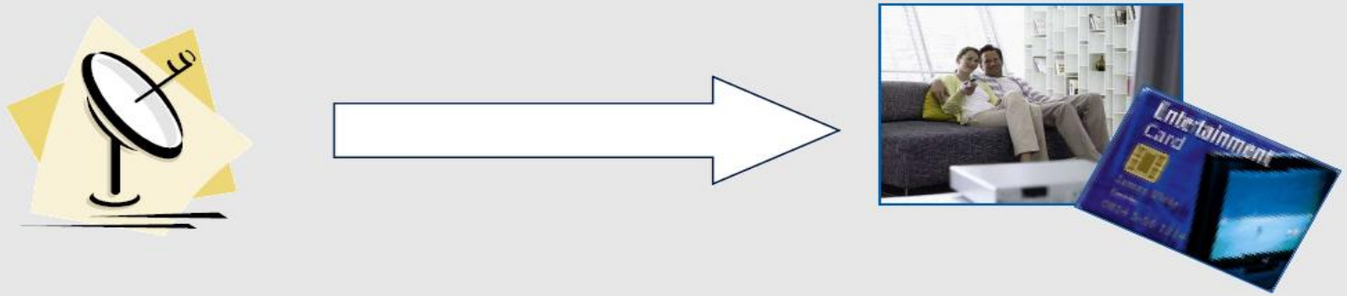
- *"A cryptosystem must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience"* [Kerckhoffs's Principle]
  - The security of cryptosystems is based on the secrecy of a key
    - It is easier to keep a secret key than a secret algorithm
    - Keys can be more easily changed than algorithms
    - Communication with many parties is easier
    - Published designs undergo public review
- *"A cryptographic scheme for a given task is secure, if no adversary of a specified power can achieve a specified break"*

Cryptographic Devices

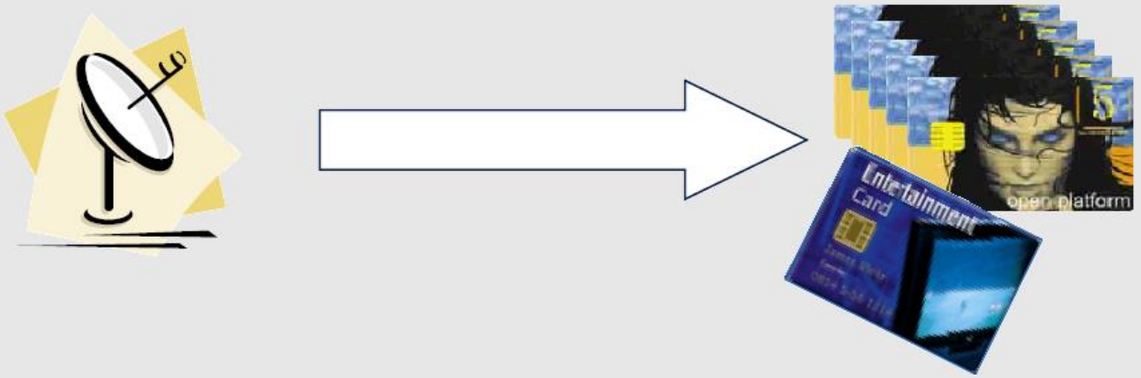
- are electronic devices that implement cryptographic algorithms and that store cryptographic keys [DPAbook]
- their security should not rely on the secrecy of its implementation
  - *no security by obscurity*
- Breaking a cryptographic device means extracting the key of the device not breaking the underlying algorithm.

Cryptographic Devices, Examples

- Pay TV system
  - The broadcasting company gives smart cards to their customers and the customers pay for viewing the content.



  - The broadcasting company cannot assume that all customers are honest. In fact, a customer might want to duplicate her card to enable also her friends to watch the content



Cryptographic Devices, Examples (cont'd)

- Electronic Money (inside the chip card of the bank cards)
- ID cards, passports
- Car Remote Controls/Immobilizers
- USB Tokens
- RFID tags

  (smart cards were the starting point,

  but still are sensitive cases)
- Several cryptographic devices are involved in our daily life
  - There are also a huge interest to break them
    - For financial gains, fun, illegal copying

Attack Scenarios and Goals

- The attacker is a legitimate owner of the device
  - She can use the device showing that she is an honest owner
  - She can manipulate, disturb, and observe the device
- The goal of the attack is to find the secret key
- Once the secret key is revealed, the security is gone….
- Later she may use the revealed key for other purposes
  - Deciphering the communication of the others
  - Counterfeiting the contents, …
  - Illegal access to the secured objects

Examples of Attacks on Crypto Devices

- Physical attack means to use the physical properties of the cryptographic device in order to break it.
  - then, PHYSICAL ACCESS to the device is needed
- We start with an example:
  - Eve finds a USB stick with secret data on it
  - In order to access this secret information an 8-digit PIN needs to be entered correctly
  - The device has a delayed response - so one trail takes 1s.
  - For a Brute-Force Attack, there are 10^8 possible PIN codes
  - Trying all possible PINs takes 10^8 s ≈ 1157 days ≈3 years
- Can she find a better attack?

Examples of Attacks on Crypto Devices

- Suppose that she knows the implementation platform
  - a processor
  - A part of the software is responsible for PIN checking
  - The checking is done digit by digit (like strcmp)
    - Once the wrong digit is detected, the function terminates
  - Execution time of this process is data-dependent

How to attack?

Attacking strcmp

- She should be able to measure the time
  - from sending the PIN (pressing the ENTER) to get the response
- She selects a PIN and change only the first digit while measuring the time
  - The longest time shows the correct first digit
- Finding the first one, she can do the same for the rest
- This procedure takes 10×8=80 timing measurements!
  - In the case that she cannot measure the timing accurately, she needs to get more timing measurements, and get the average

Attacking strcmp (cont'd)

- Running time of strcmp MUST be fixed independent of the given PIN and the stored one
  - is not easy and straightforward, but feasible

  Does Eve have any further chance?

  YES!
- Most of the digital circuits are implemented by CMOS technology
  - Power consumption of the processor depends on the executed instruction and processed data
    - We will see them in more detail

Attacking strcmp (cont'd)

- She should be able to measure the power consumption during PIN checking process

  [The Attack scenario is similar to the last one]

- She selects a PIN and change only the first digit while measuring the power consumption
  - The most different power consumption shows the correct first digit
- Finding the first one, she can do the same for the rest
- Complexity and noise problem, the same as timing attack

Attacking strcmp (cont'd)

- Overcoming this problem is not easy
  - The power consumption of the device should be somehow independent of the processed data/the executed path
  - Or the interesting part for Eve should be hidden somewhere
    - We will see the countermeasure scheme in detail

  Does Eve have any further chance?

  YES!

Attacking strcmp (cont'd)

- Digital circuits need special conditions to operate correctly
  - Frequency of operation
  - Temperature
  - Voltage
- If Eve can change this condition while entering the PIN
  - by changing the frequency (clock glitch)
  - by changing the supply voltage (power glitch)
  - She may be able to skip the instruction checking the PIN
    - Yes, is hard and needs deep knowledge about the design/when the PIN is checked/…
    - If so, she does not get the PIN, she can use it once and pass the authentication

Physical Attacks

are classified into groups:

- From the attacker's behavior point of view
  - Passive Attacks
    - The device works normally; the attacker just listens
  - Active Attacks
    - The attacker makes trouble for the device
- From invasiveness point of view
  - Invasive Attacks
    - Without any limit; the strongest attacker model
  - Semi-Invasive Attacks
    - The device is depacked, but no electrical contact
      - to read/induce faults
  - Non-Invasive Attacks
    - No modification at all

Physical Attacks (examples)

	Passive	Active
Invasive	Probing	Forcing Permanent Faults
Semi-Invasive	Optical Inspection	Light Attack Radiation Attack
Non-Invasive	Side-Channel Attacks	Clock Glitch Power Glitch Temperature

Side-Channel Attacks: Power Analysis/EM Analysis/Timing Analysis

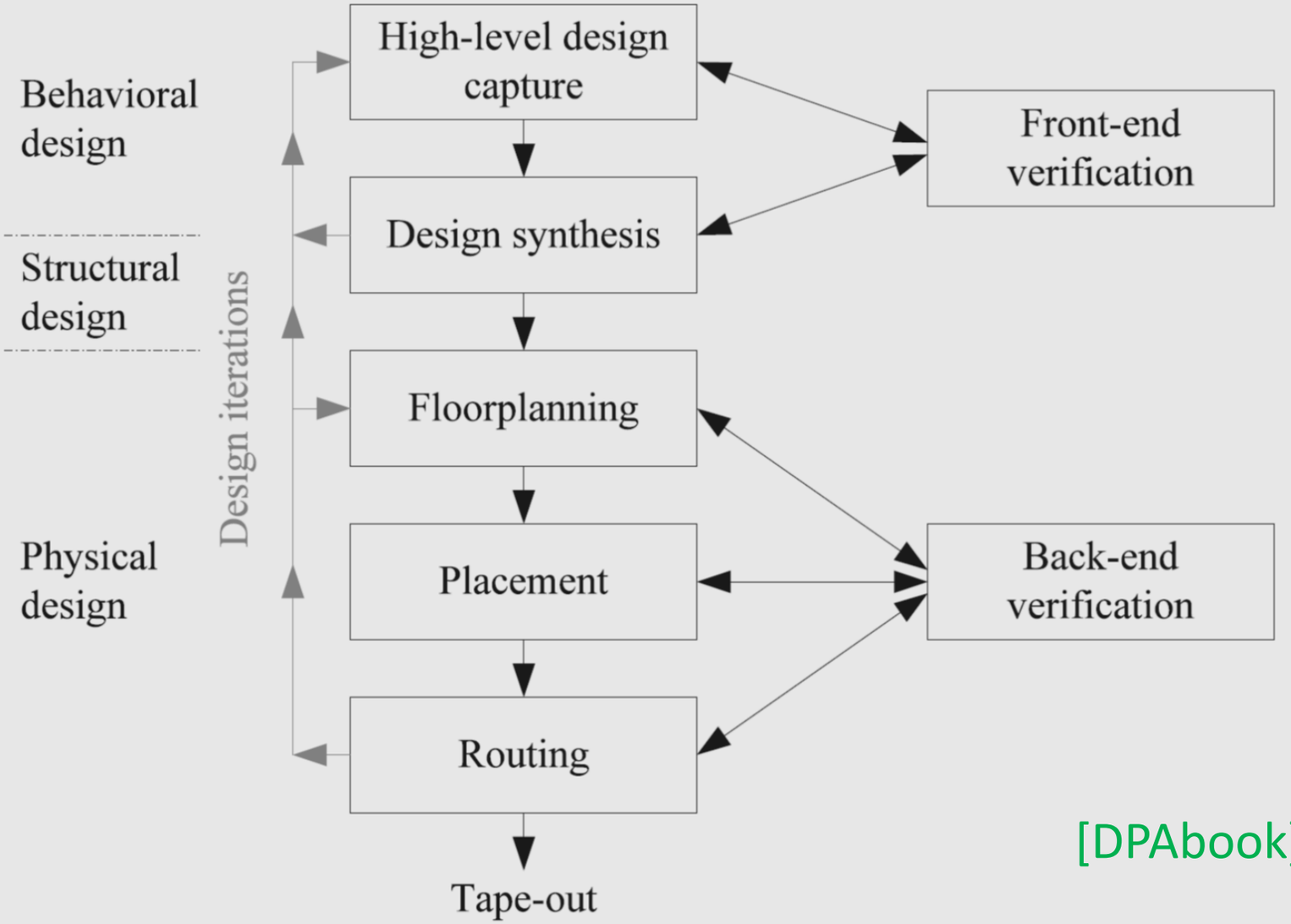
Basics of Hardware Design

- Traditionally divided into (iteratively)
  - Specification (High-level prog. lang.)
  - Behavioral Design (RTL)
    - sequences, bit widths, co-design, synch/asynch
    - HDL code
  - Structural Design
    - netlist transistors/cells
  - Physical Design
    - placement, routing

Semi-Custom Design

- A standard-cell library is used
- Many things can be done automatically
  - from “behavioral design” to the end
  - called SYNTHESIS
- The output of the synthesis is a netlist. In fact, it is a graph, where each node corresponds to a gate
- When making the LAYOUT, the gates are placed and connected
- The layout is a geometrical description of the design. It describes which material should be placed where

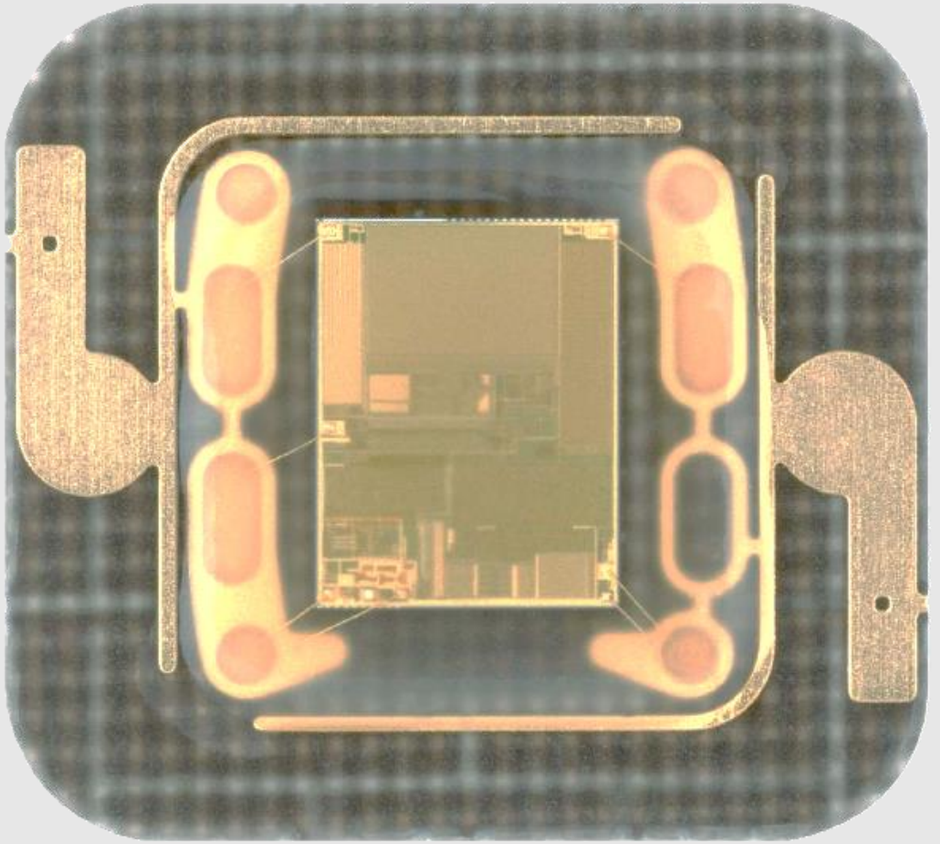
Semi-Custom Design (cont'd)



[DPAbook]

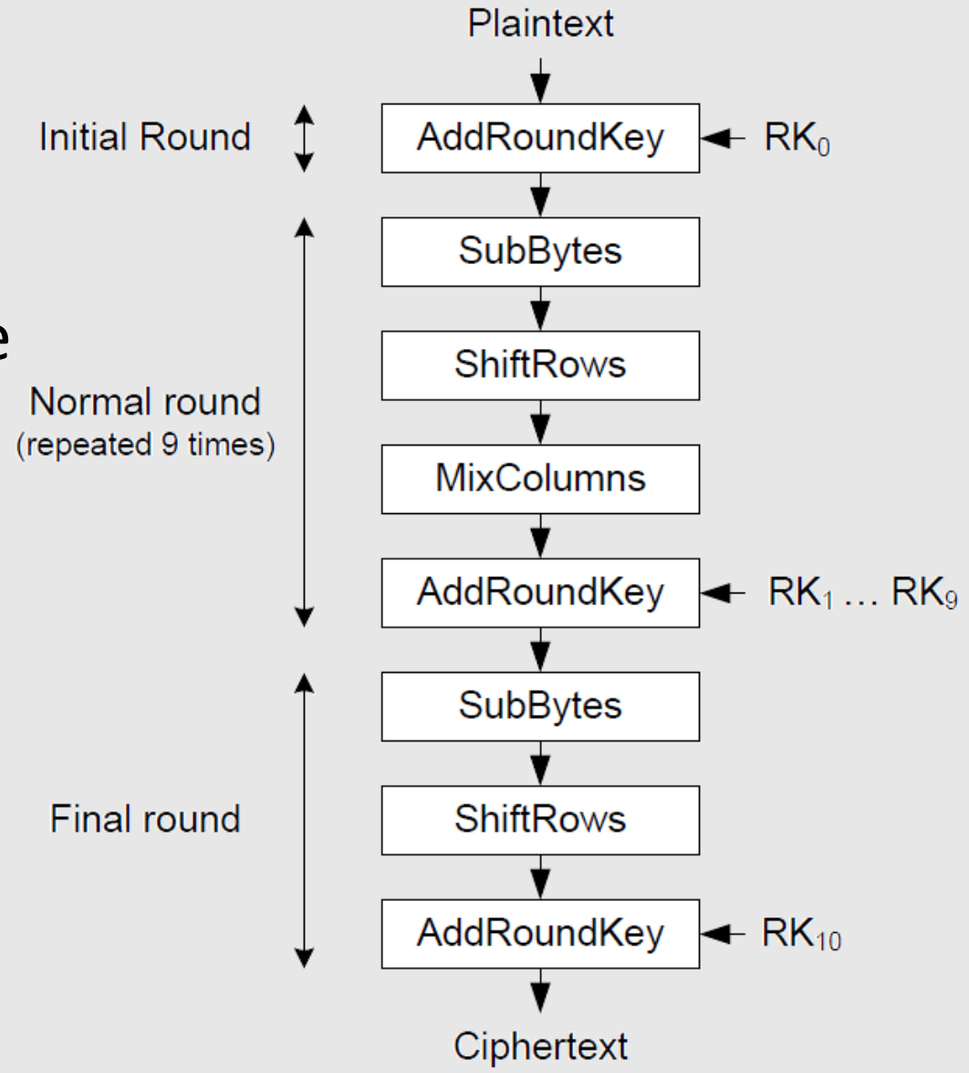
An Example


NAME DES GELDINSTITUTES
99999
9
PHILIP
SCHUMAN
1234567890
987654
SPECIMEN
edc



Short Review of AES

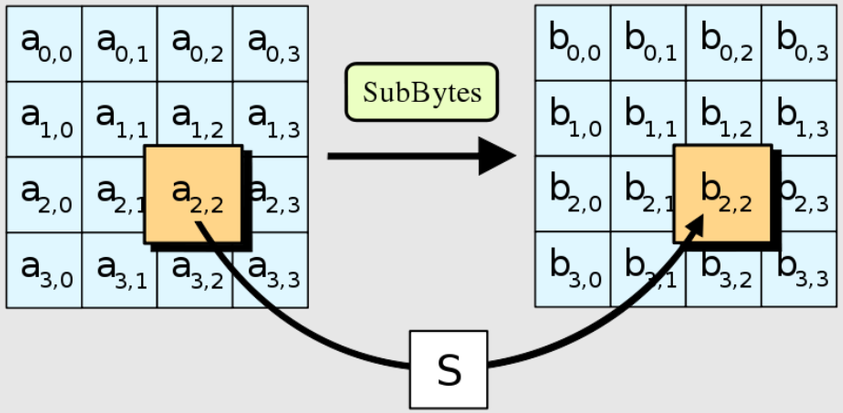
- There are 3 different versions: 128-, 192-, and 256-bit key size
- We consider only the 128-bit version.
- Plaintext and ciphertext are always 128-bit wide
- The output is ready after 10 rounds of computations
  - the last round is different



AES (AddRoundKey and SubBytes)

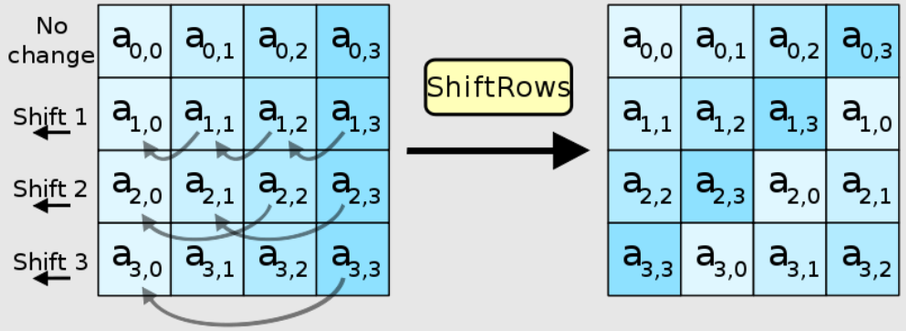
- All the computations are done on a 4×4 byte-wise matrix
- All the fundamental operations are defined considering matrix representation
  - AddRoundKey is a bitwise XOR operation
    - runs over two 128-bit matrices
  - SubBytes runs the Sbox over each byte independently
    - all the Sboxes are the same
    - is an inversion in GF(2^8)

      in addition to an affine transformation

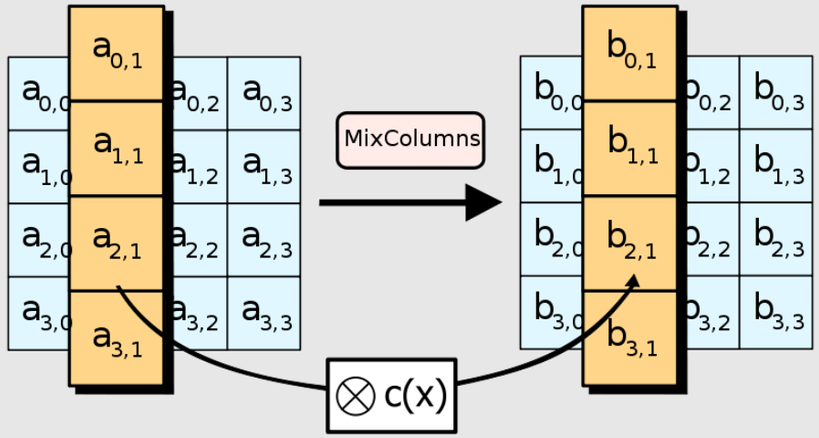


AES (ShiftRows and MixColumns)

- ShiftRows rotates 1st row, 2nd row, 3rd row, and 4th row 0, 1, 2, 3 bytes to the left respectively



- MixColumns runs on each column separately
- Each column is multiplied to a fixed matrix
  - the multiplication is by means of the same polynomial as in Sbox

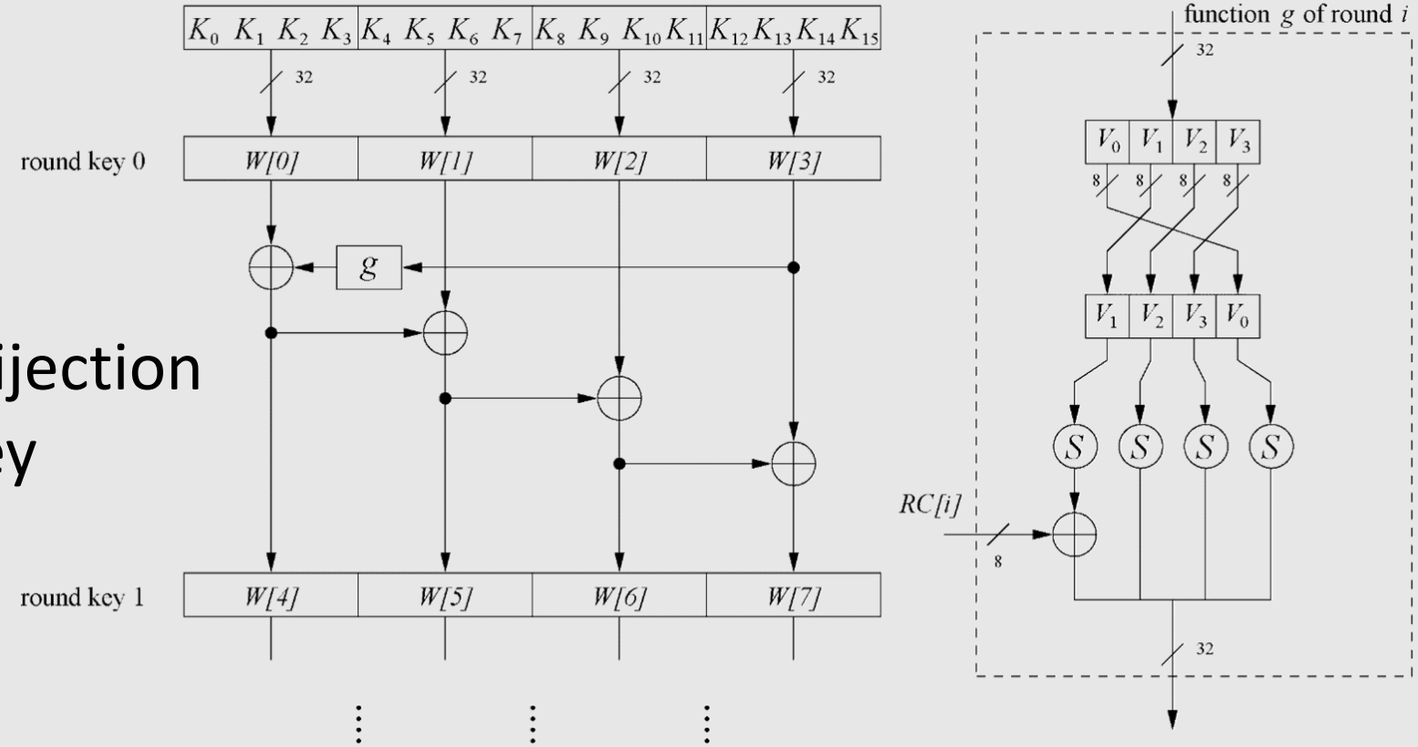


AES (key schedule)

- gets 128-bit main key and provides ten 128-bit roundkeys
  - 11 roundkeys are needed, the first one is the main key
- A byte-wise rotation, 4 Sboxes, and XOR operations are used to make one 128-bit roundkey from the last roundkey
- RC[] is a round constant and is

  1, 2, 4, 8, 10, 20, 40, 80, 1B, 36

  (all in HEX)
- Each round of key schedule is a bijection knowing a round key, the main key can be trivially computed



AES (how to implement)

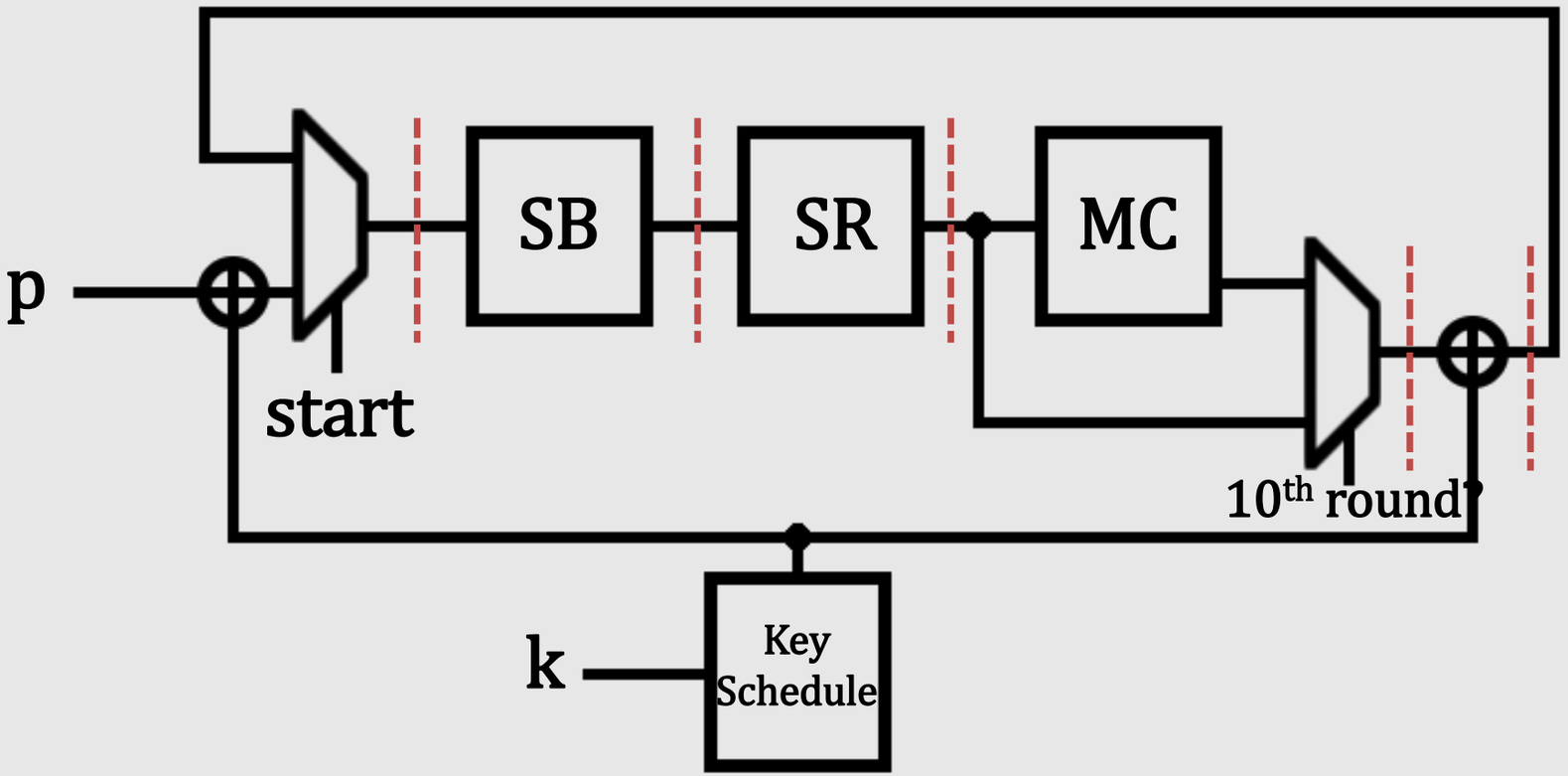
- Both hardware and software implementations are straightforward
- Sboxes are too big for the hardware case unless implemented by lookup tables (RAM/ROM cells)
- MixColumns is the most expensive operation in processors if the memory is restricted
- 4 Sboxes and a MixColumns can be merged to make an 8-bit to 32-bit lookup table known as T-table
- When enough memory (4kb) is available, both software and hardware implementations contain only the T-tables and XORs, which is the fastest way to realize AES

Pseudo-code AES

INPUT: Plaintext p

OUTPUT: Ciphertext c

```
x = p
x = x ⊕ k_0
for i = 1 to 10
    x = SB(x)
    x = SR(x)
    if i ≠ 10
        x = MC(x)
    x = x ⊕ k_i
c = x
```



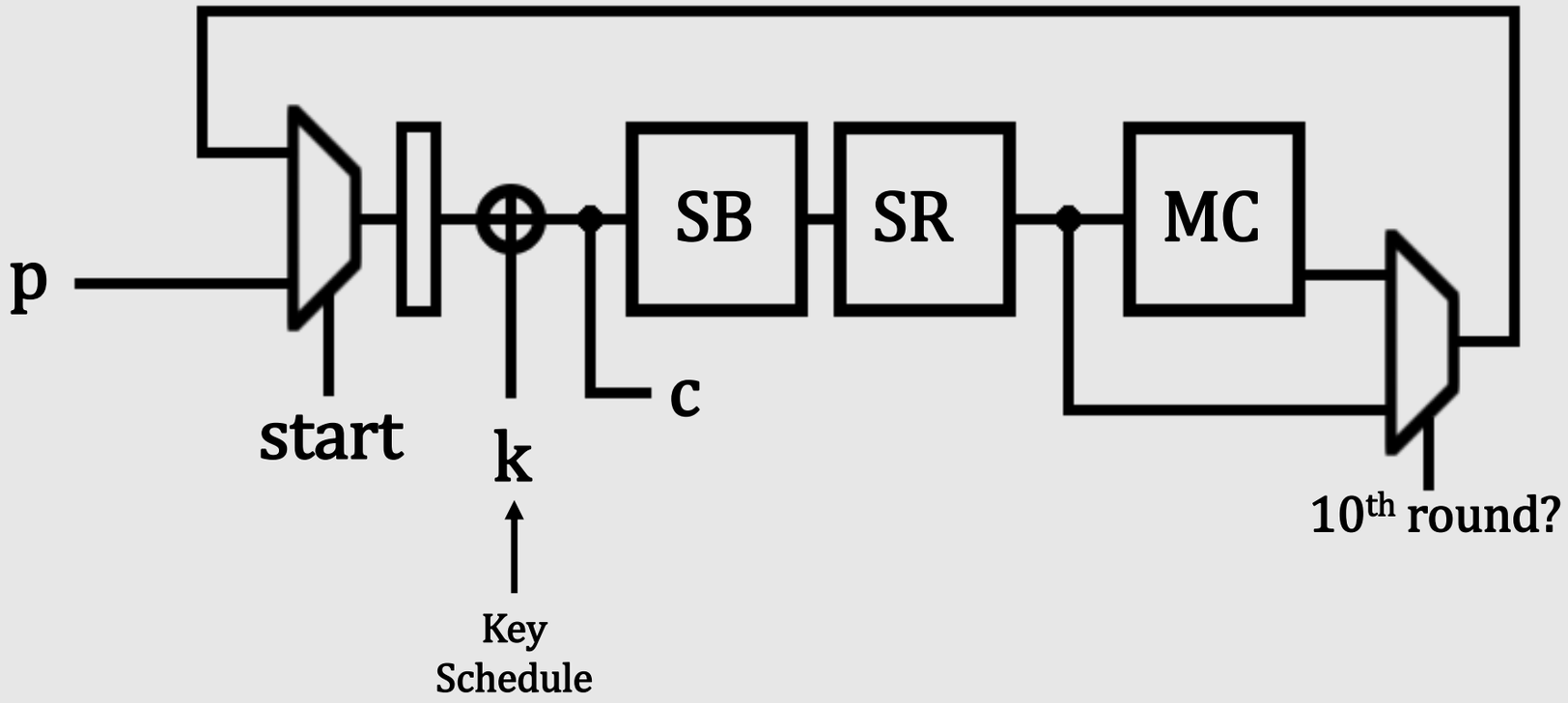
- It does not work without registers
- Registers can be placed between any two stages

Pseudo-code AES (optimizied)

INPUT: Plaintext p

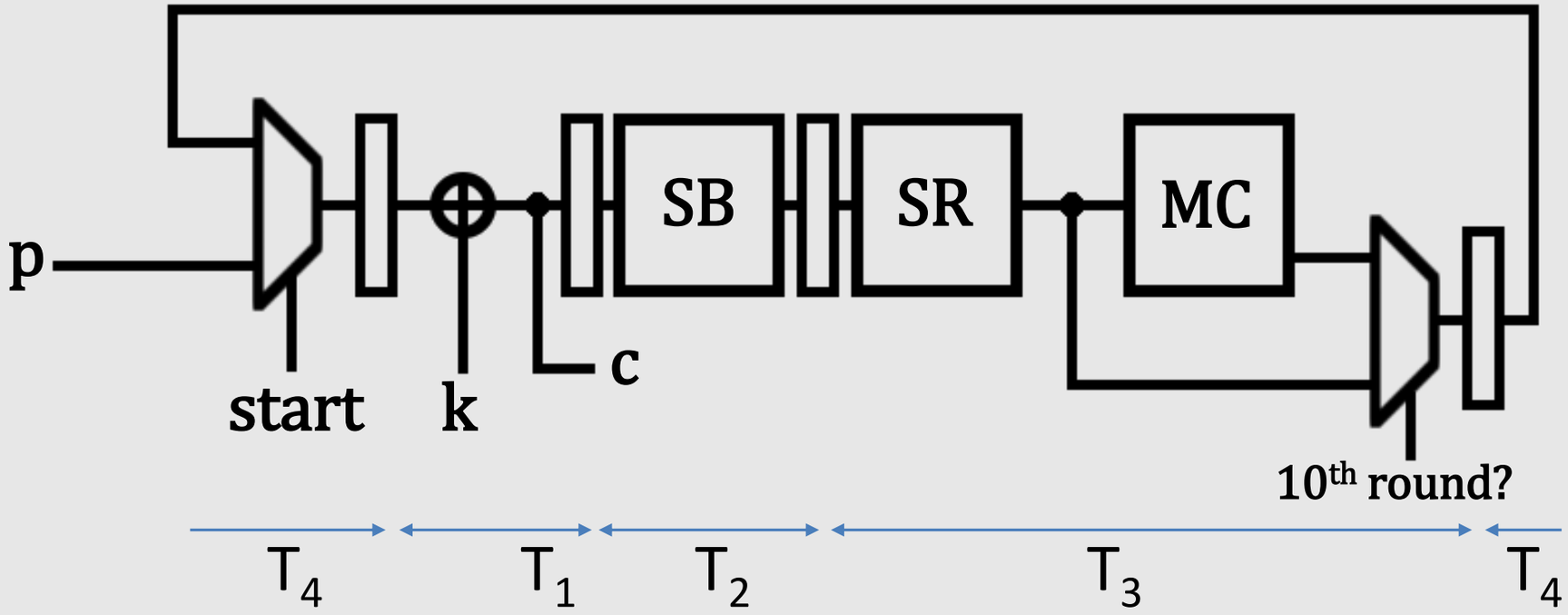
OUTPUT: Ciphertext c

```
x = p
for i = 0 to 10
    x = x ⊕ k_i
    if i ≠ 10
        x = SB(x)
        x = SR(x)
        if i ≠ 9
            x = MC(x)
c = x
```



Critical Path Delay

The maximum clock frequency depends on the critical path delay: the longest path between two consecutive registers.



$$\text{Max. } f = \frac{1}{T} = \frac{1}{\max(T_1, T_2, T_3, T_4)}$$

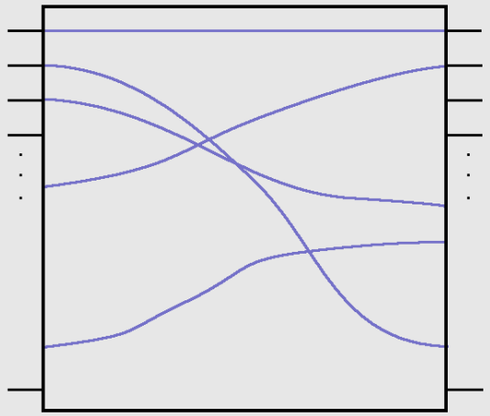
Maximum Clock Frequency

- Why ShiftRows and MixColumns are packed together?
  - ShiftRows in hardware can be realized without any combinatorial component, and has (in an ideal model) no delay.

0	4	8	12
1	5	9	13
2	6	10	14
3	7	11	15

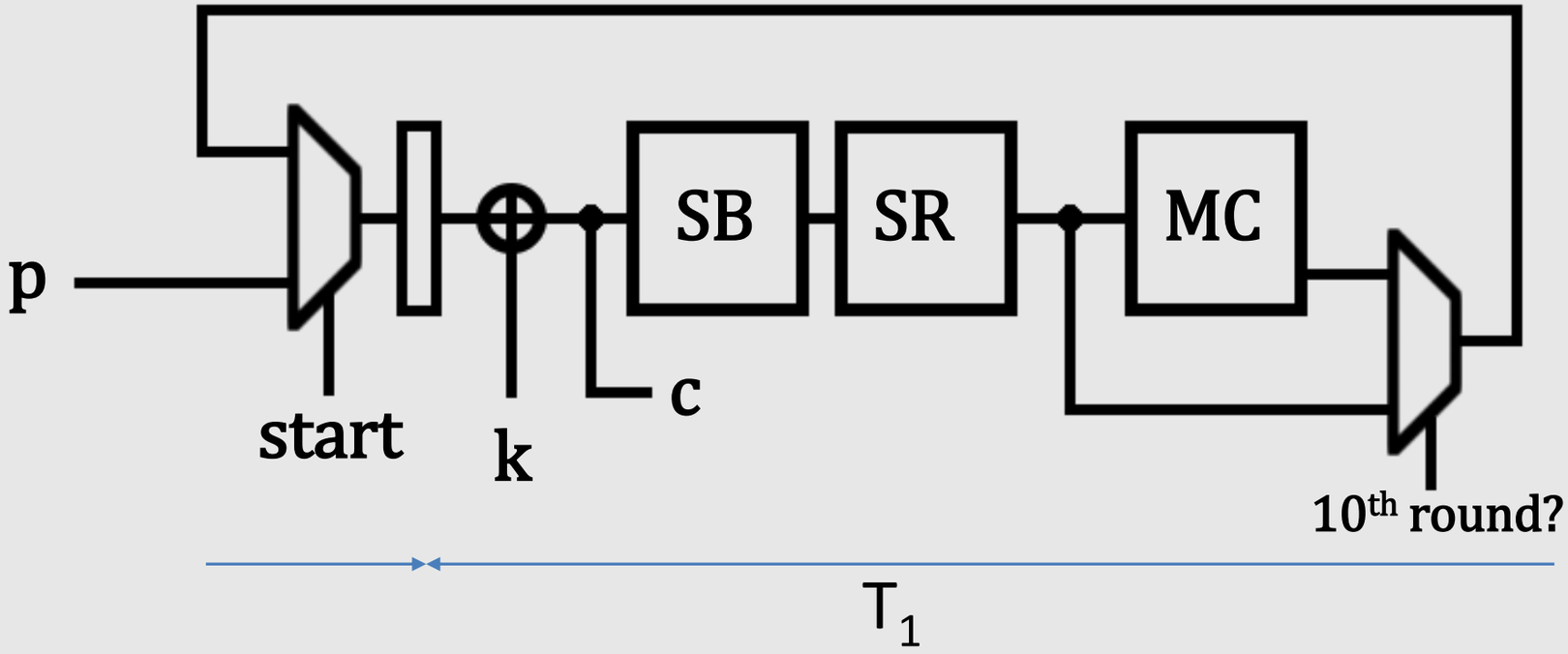
SR

Shift Rows



0	4	8	12
5	9	13	1
10	14	2	6
15	3	7	11

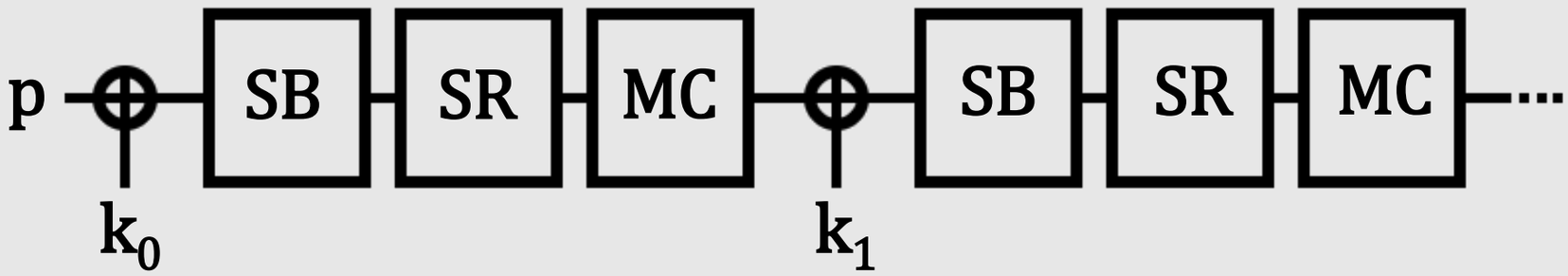
Critical Path Delay



$$\text{Max. } f = \frac{1}{T} = \frac{1}{T_1}$$

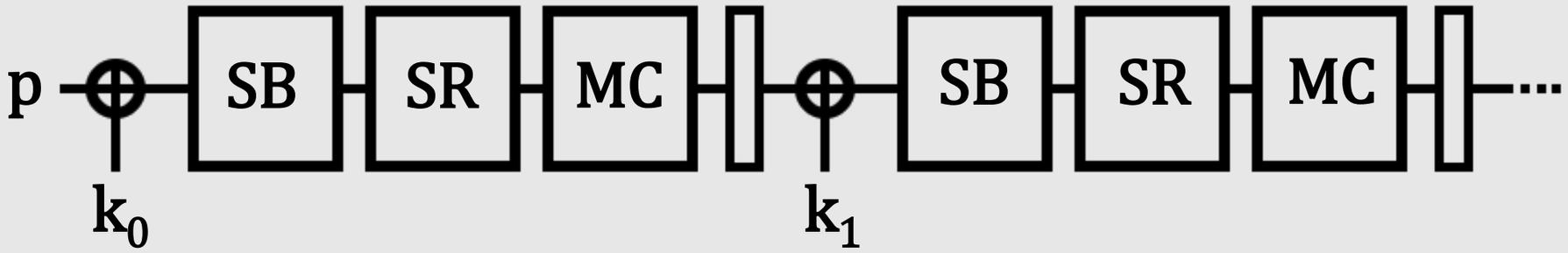
Unrolling

- Fully combinatorial
- Fastest implementation (not WRT clock, but WRT latency)
- Very large area and long delay due to the combinatorial circuits



Pipeline Unrolling

- Insert registers in the unrolled structure
- Critical path delay becomes smaller
- Frequency can be increased



Pipeline Unrolling

Consider an architecture with 10 registers:

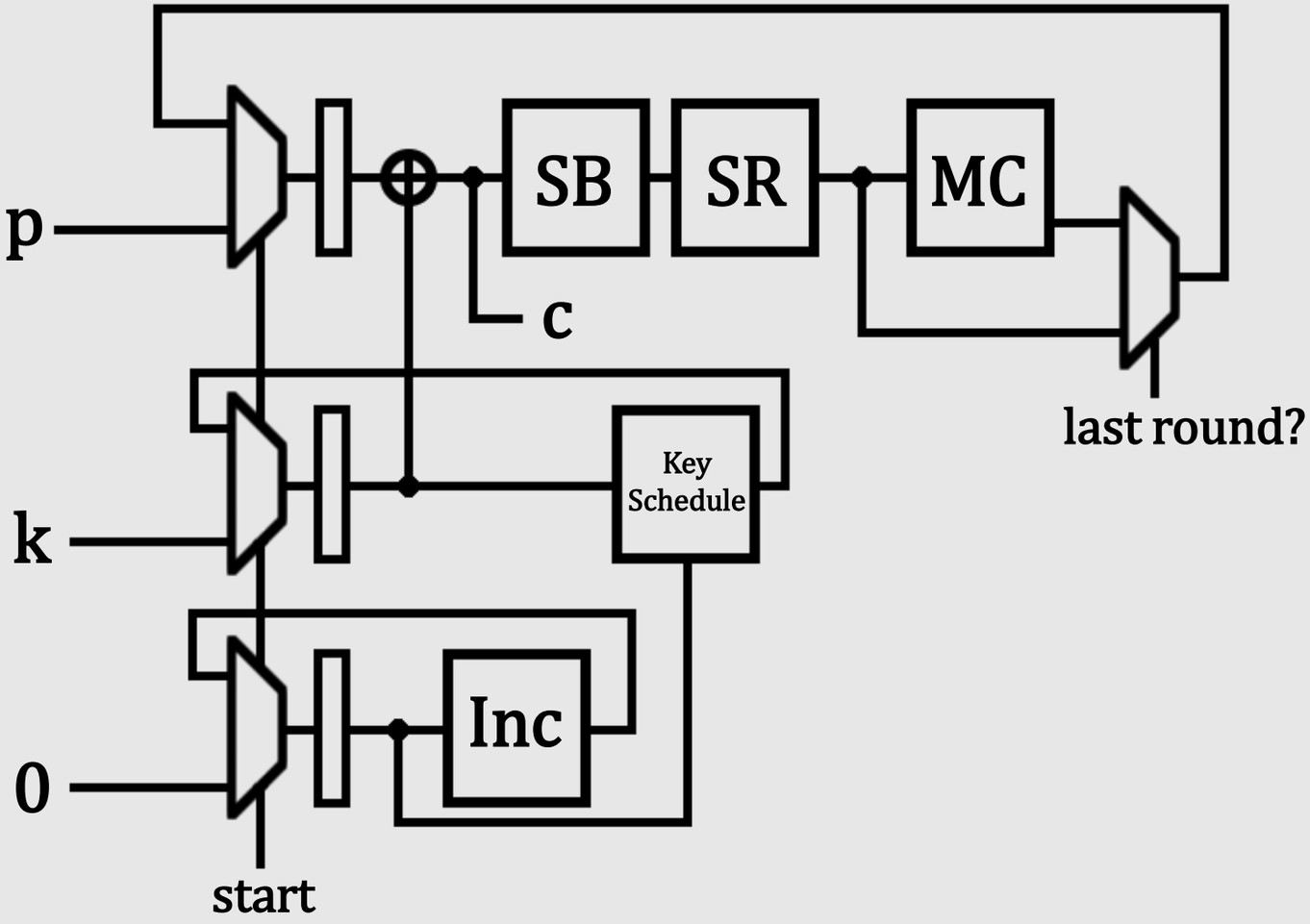
- Latency of 10 clock cycles
  After 10 clock cycles is the first ciphertext on output

- THEN: at each clock cycle, a further ciphertext is given as output

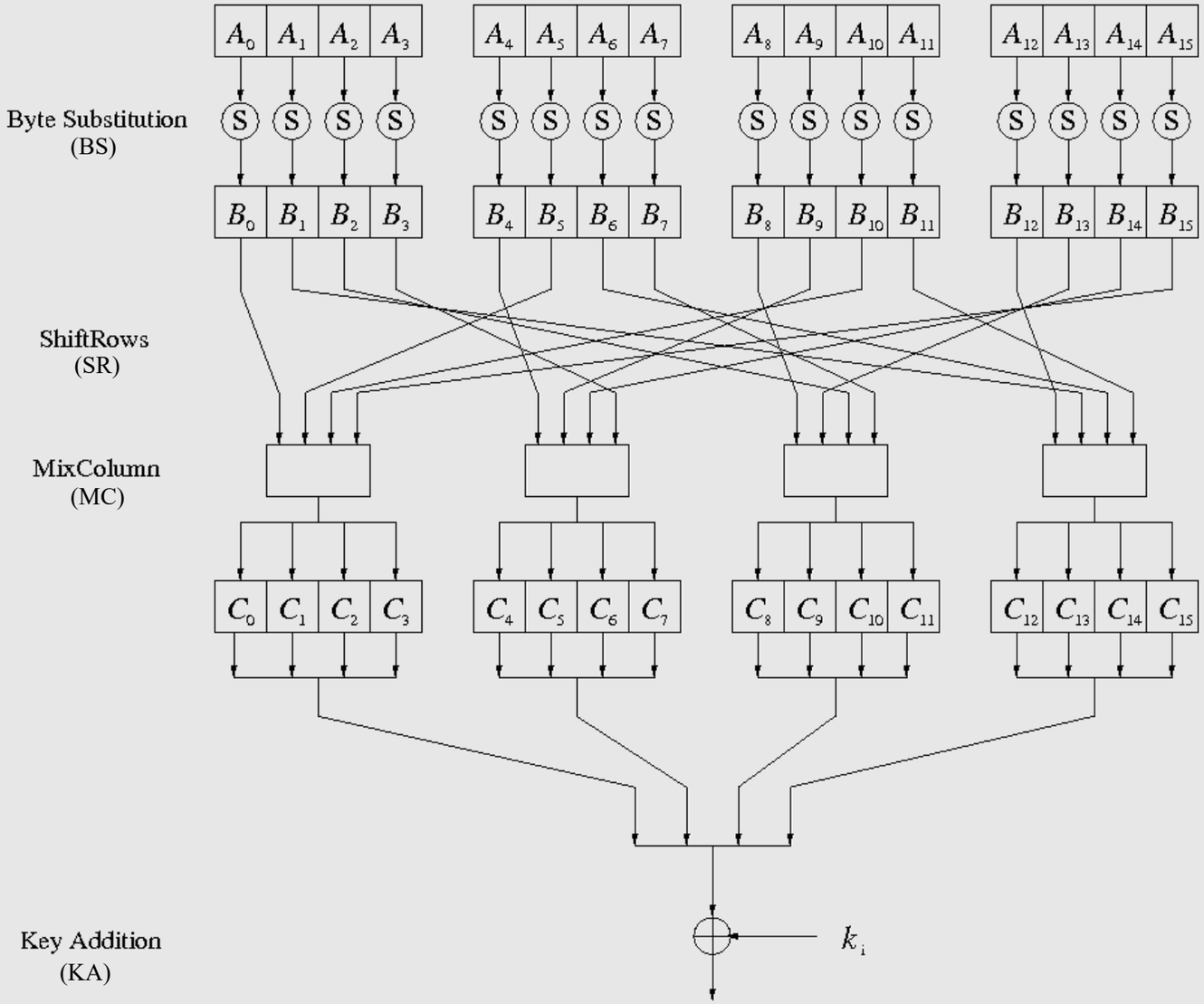
In practice, it is not usually used, but good for high throughput purposes

→COPACOBANA (Rivyera): DES with unrolled pipelining
120 FPGAs, 8 DES/FPGA,…
finding the key in a week

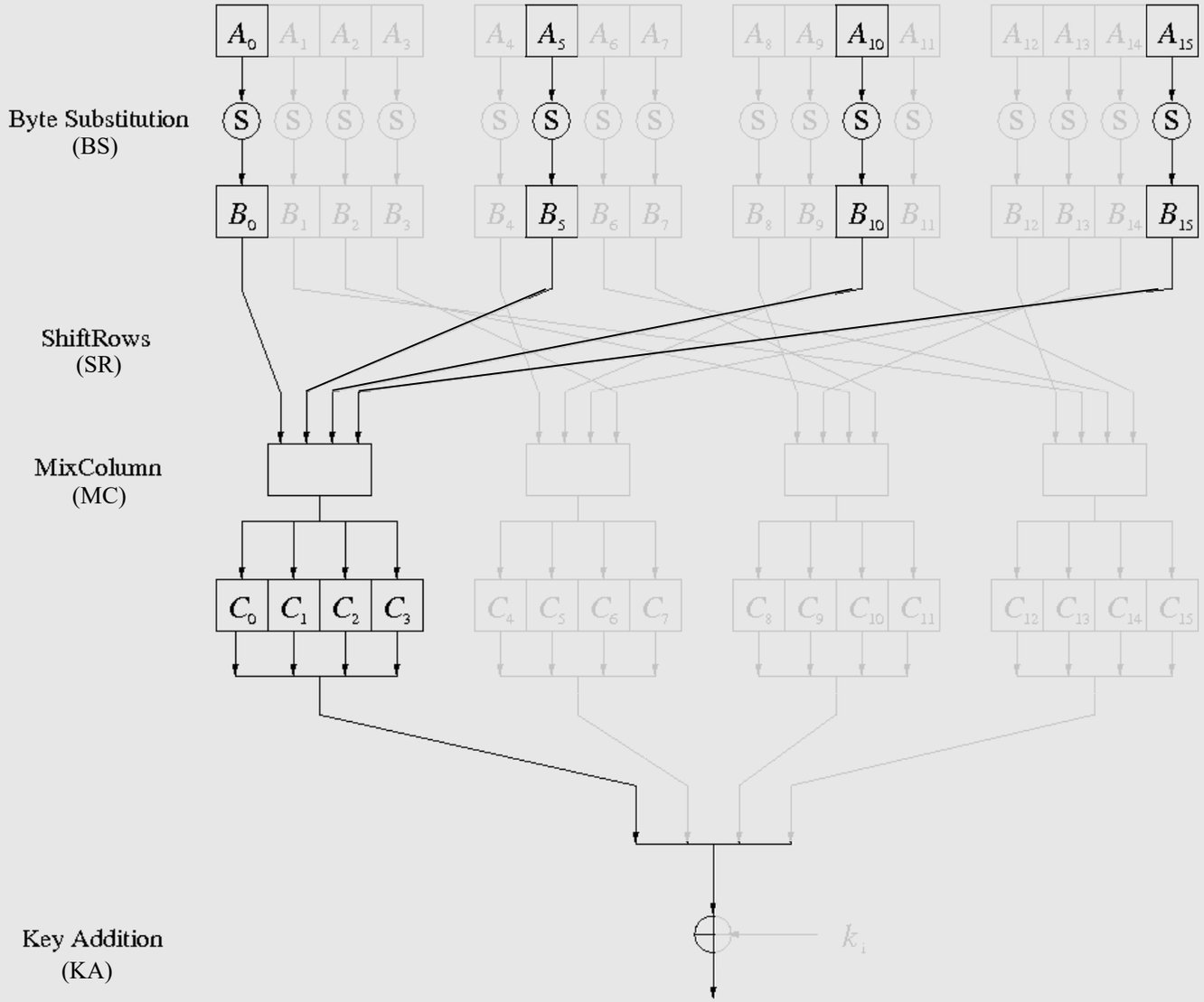
Round-Based AES



Round Function of AES-128



Round Function of AES-128



T-Table Implementation

- The most hardware designs use the 8-bit structures of the AES
- There is an alternative approach for 32-bit processors to integrate the primitive operations in the S-Box
- This results in four $T_i\,[x]$ tables with 256 x 32-bit entries (that is 8192 bits per table)
- These tables show the AES encryption as

$$T_0[x] = \begin{bmatrix} S[x] \times 02 \\ S[x] \\ S[x] \\ S[x] \times 03 \end{bmatrix} \quad T_1[x] = \begin{bmatrix} S[x] \times 03 \\ S[x] \times 02 \\ S[x] \\ S[x] \end{bmatrix} \quad T_2[x] = \begin{bmatrix} S[x] \\ S[x] \times 03 \\ S[x] \times 02 \\ S[x] \end{bmatrix} \quad T_3[x] = \begin{bmatrix} S[x] \\ S[x] \\ S[x] \times 03 \\ S[x] \times 02 \end{bmatrix}$$

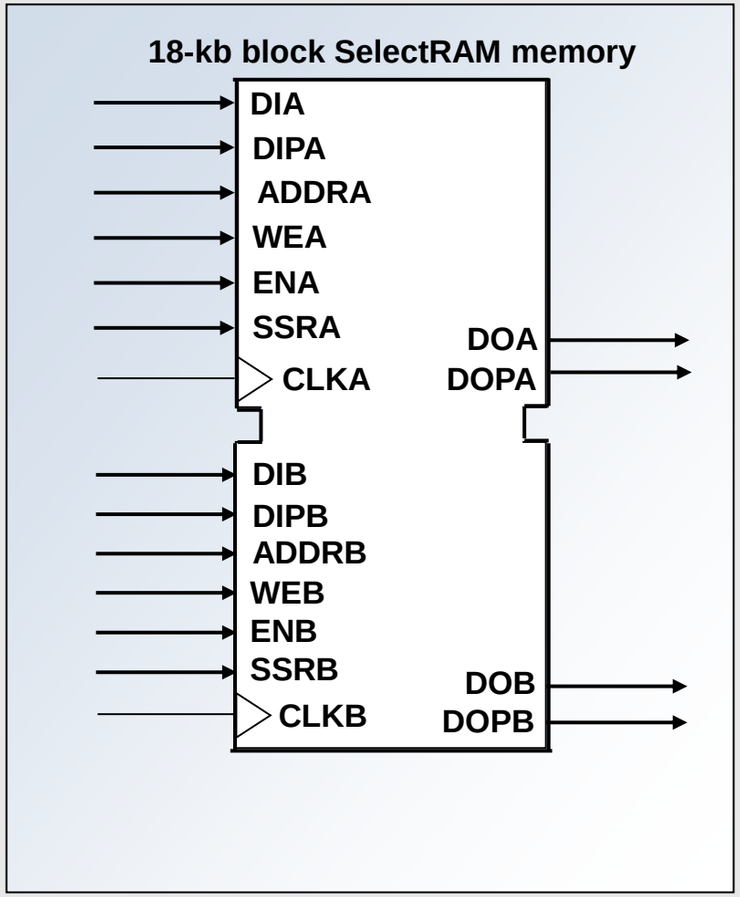
$$C_j \quad = \quad K_{r,j} \oplus T_0[a_{4j}] \oplus T_1[a_{(4j+5) \bmod 16}] \oplus T_2[a_{(4j+10) \bmod 16}] \oplus T_3[a_{(4j+15) \bmod 16}]$$

where C_j represents 32 bits of the AES round and $K_{r,j}$ a 32-bit round key

FPGA Block Memory (BRAM)

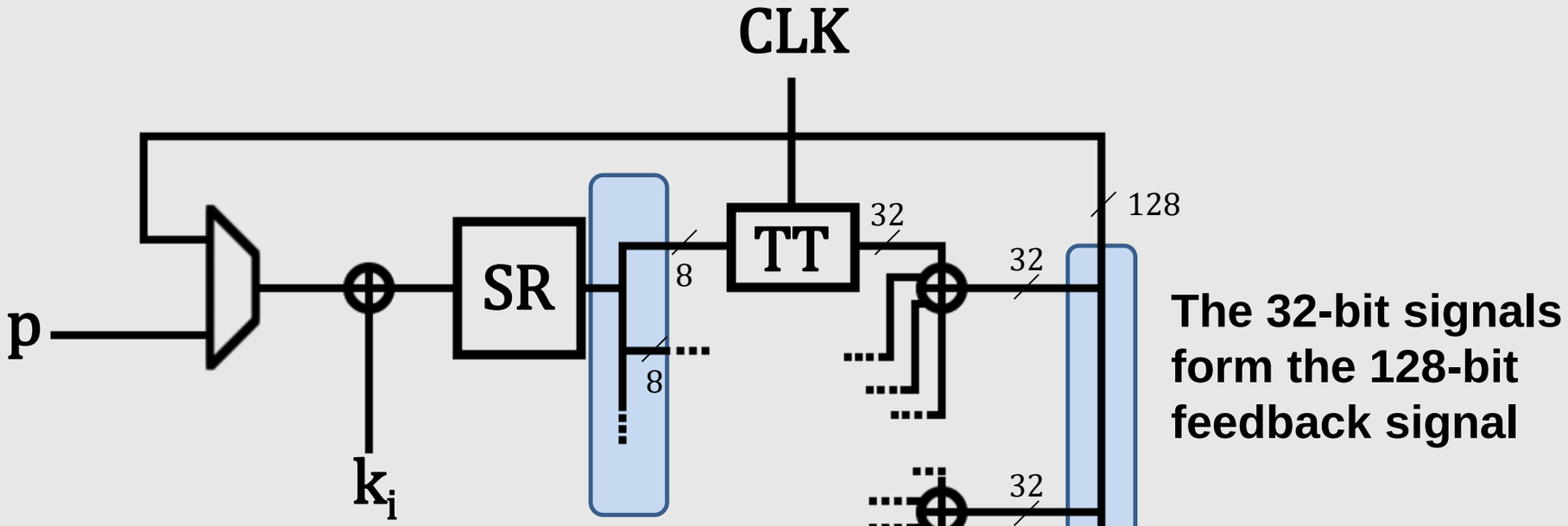
- Dedicated memory in 18/36-Kbit blocks

- True-Dual-Port
  Two Ports: each one can at the same time and independently be accessed for read and write

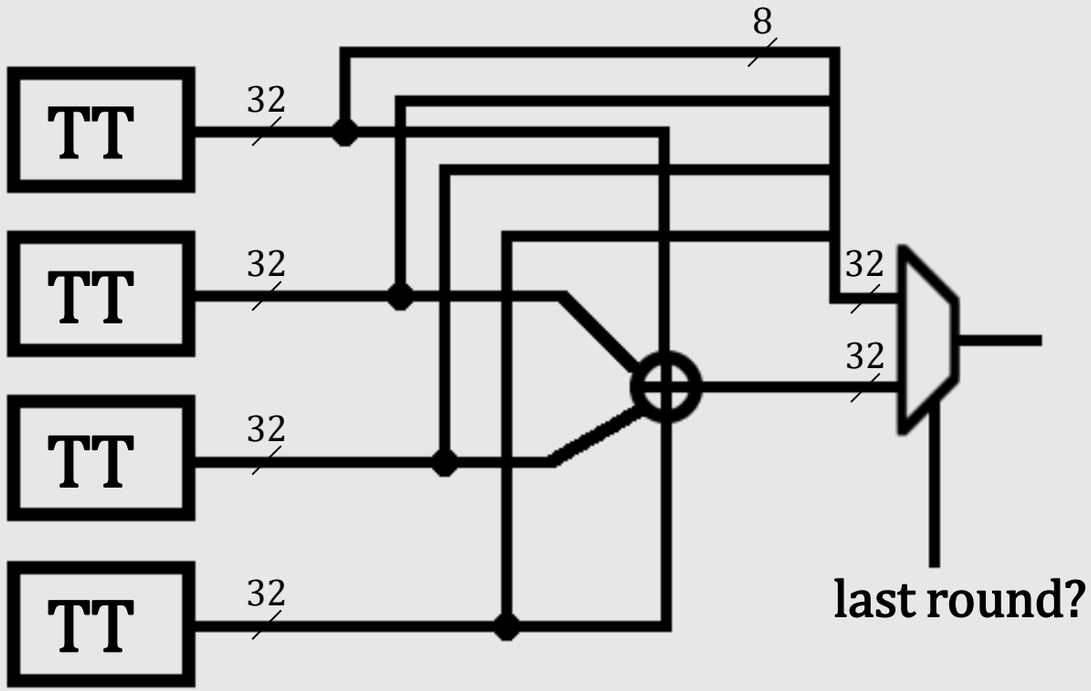


T-Table AES (making use of FPGA BRAMs)

- BRAM is „clocked“
  - It means that no extra register is necessary.
- True Dual-Port also allows to employ only one BRAM for two T-Tables.



T-Table Last Round



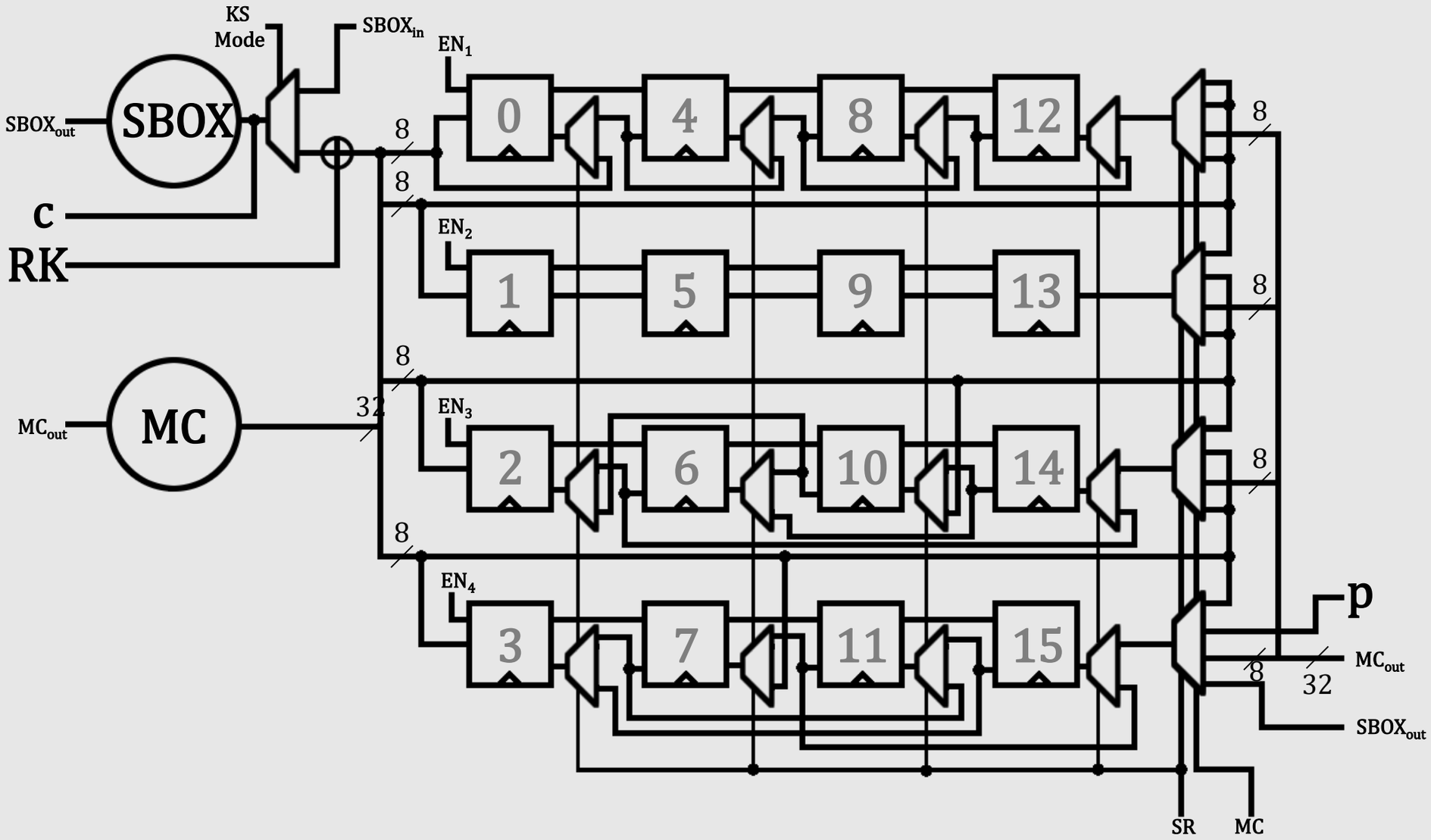
T-Table AES

- 16 T-Tables are required
- 8 BRAMs are enough
  - Because of the True Dual-Ports
- Very fast, but a large implementation wrt BRAMs

AES with Small Area

- The Sboxes need very much area
- Idea: instanciate only one Sbox module and share it
- This results in „Serialized AES“

Serielized AES with MC in one clock cycle



Serielized AES: Key Schedule