ulm university universität ulm

# Error Correction for Physical Unclonable Functions Using Generalized Concatenated Codes

Sven Müelich*, Sven Puchinger*, Martin Bossert*,
Matthias Hiller◇, Georg Sigl◇

*Institute of Communications Engineering, Ulm University, Germany
◇Institute for Security in Information Technology, TU Munich, Germany

ACCT Svetlogorsk, September 7-13, 2014

# Outline

# Outline

# Motivation

**Challenges when implementing a cryptosystem:**

- Secure key generation
  - Random, unique and unpredictable keys
  - Satisfying these properties is hard to achieve
- Secure key storage
  - Key bits in a non-volatile memory
  - Adversaries can gain physical access to (protected) memories

# Motivation



**Challenges when implementing a cryptosystem:**

- Secure key generation
  - Random, unique and unpredictable keys
  - Satisfying these properties is hard to achieve
- Secure key storage
  - Key bits in a non-volatile memory
  - Adversaries can gain physical access to (protected) memories

**Physical Unclonable Functions (PUFs) can be used to realize secure key generation and secure key storage**

# Outline

# Physical Unclonable Functions (PUFs)



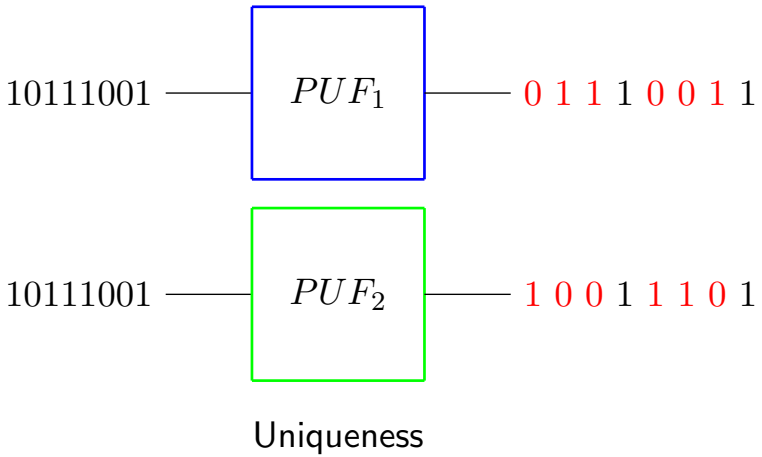
## What is a PUF?

- Physical entity with challenge-response behavior
- Properties:
  - Uniqueness
  - Reproducibility

# Physical Unclonable Functions (PUFs)

## What is a PUF?

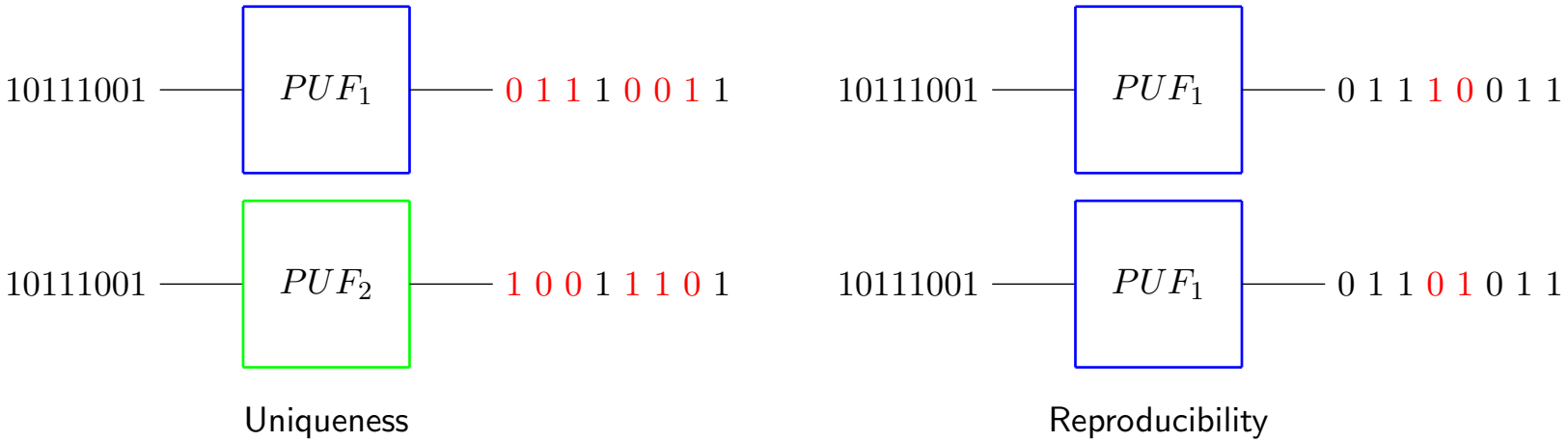
- Physical entity with challenge-response behavior
- Properties:
  - Uniqueness
  - Reproducibility



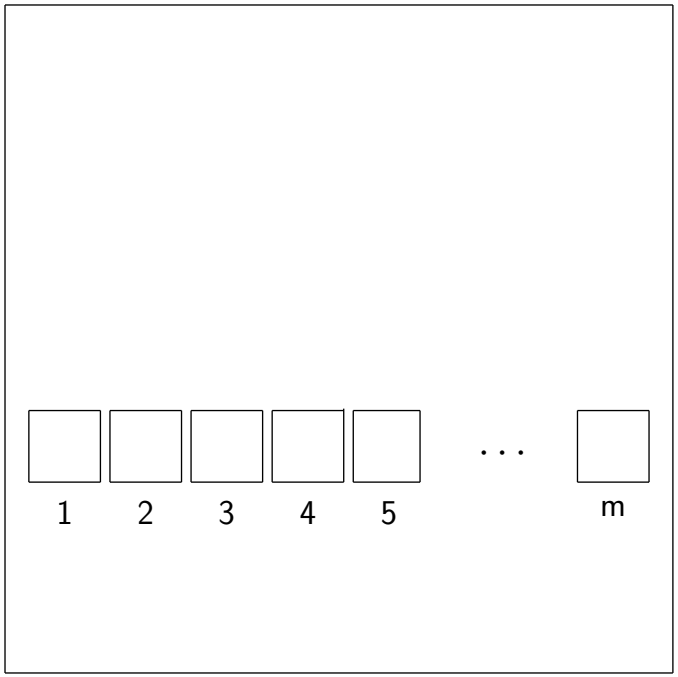
# Physical Unclonable Functions (PUFs)

## What is a PUF?

- Physical entity with challenge-response behavior
- Properties:
  - Uniqueness
  - Reproducibility



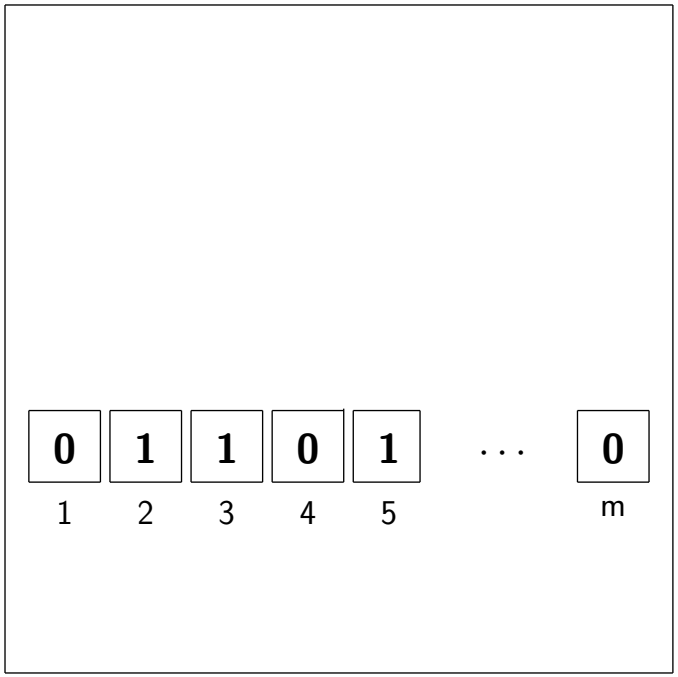
# Physical Unclonable Functions (PUFs)




**Example:** SRAM PUFs

- device with memory cells

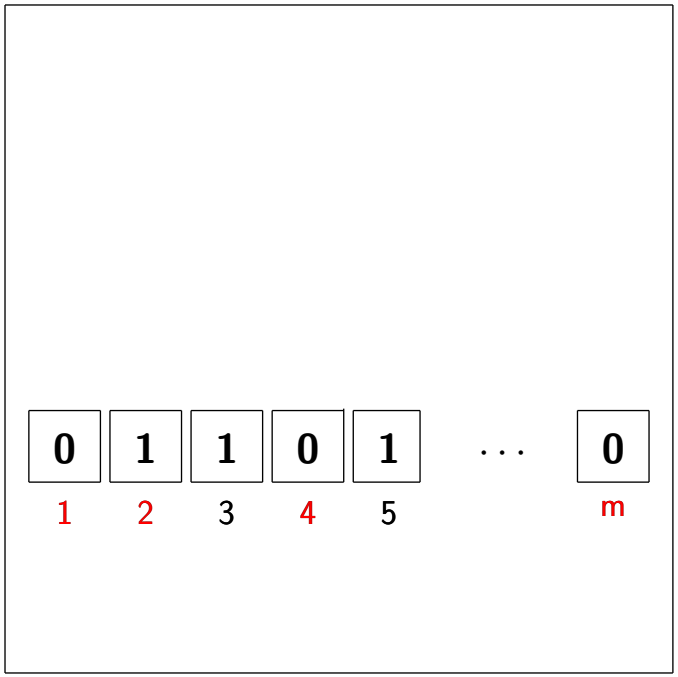
# Physical Unclonable Functions (PUFs)




**Example:** SRAM PUFs

- device with memory cells
- random initialization when powering on
- randomness static over lifetime

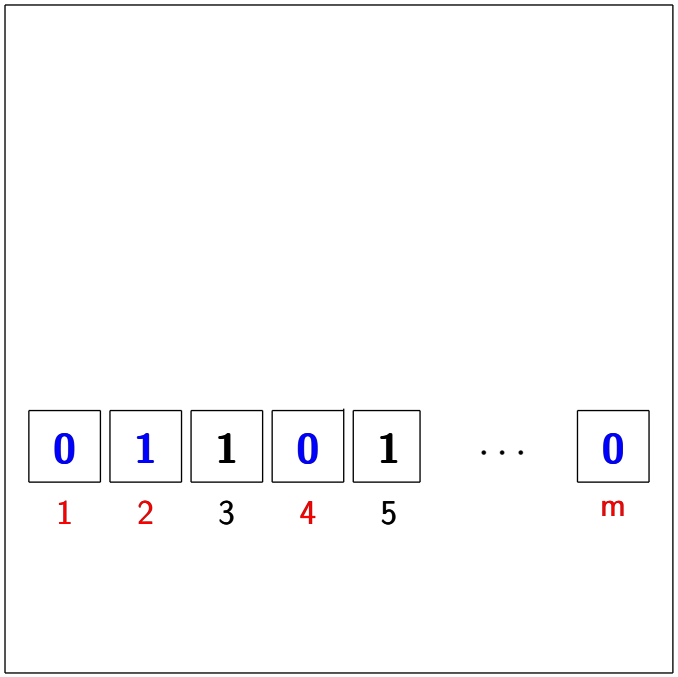
# Physical Unclonable Functions (PUFs)




**Example:** SRAM PUFs

- device with memory cells
- random initialization when powering on
- randomness static over lifetime
- Challenge: Subset of memory cells

# Physical Unclonable Functions (PUFs)




**Example:** SRAM PUFs

- device with memory cells
- random initialization when powering on
- randomness static over lifetime
- Challenge: Subset of memory cells
- Response: Values in selected memory cells

# Physical Unclonable Functions (PUFs)



**Why coding theory?**

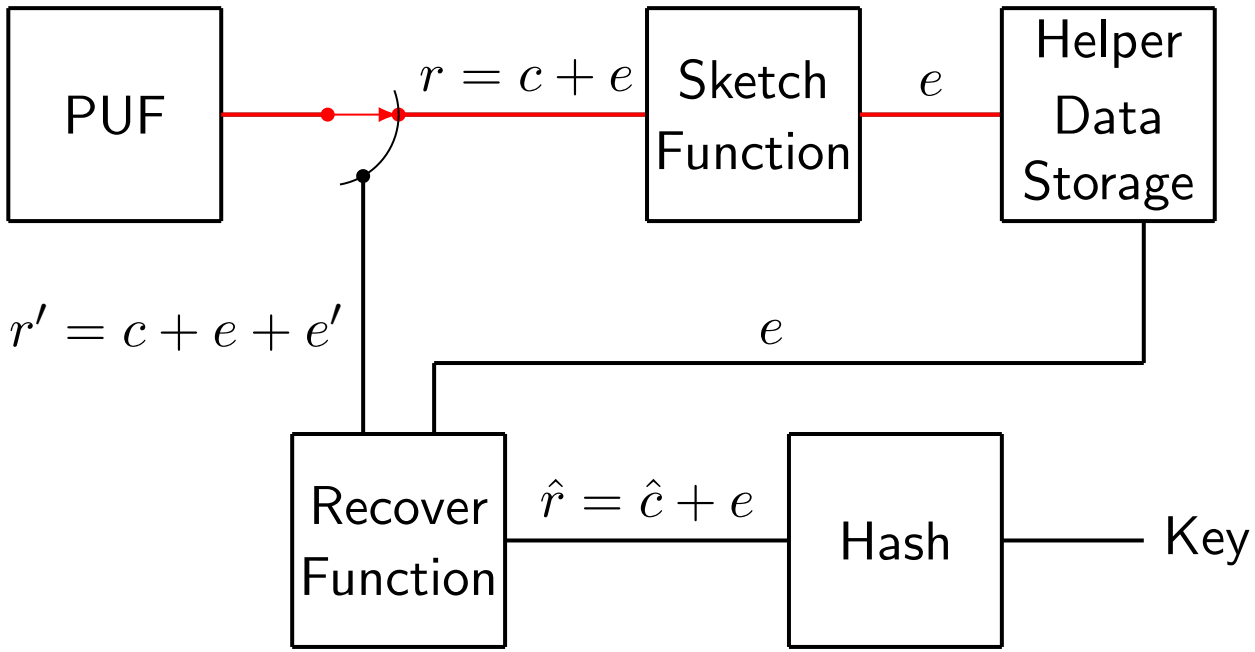
- Responses are not perfectly reproducible and hence cannot be used as key directly

# Physical Unclonable Functions (PUFs)



## Why coding theory?

- Responses are not perfectly reproducible and hence cannot be used as key directly

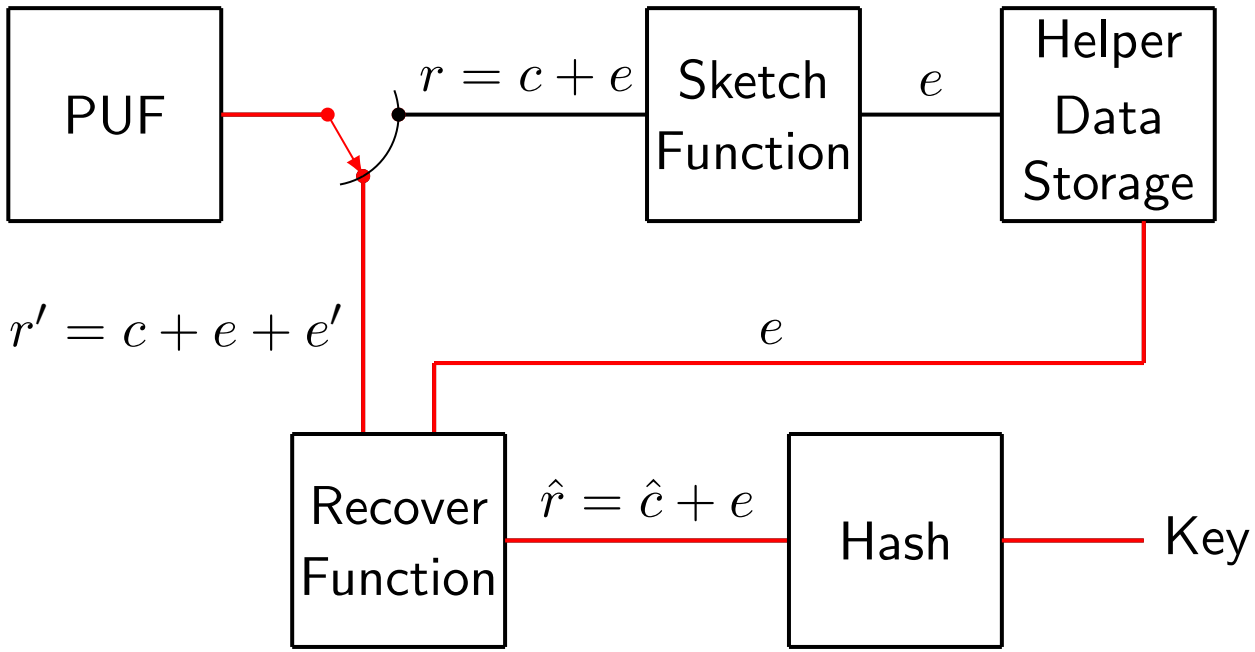


# Physical Unclonable Functions (PUFs)



## Why coding theory?

- Responses are not perfectly reproducible and hence cannot be used as key directly



# Outline

# Example Code construction

**Challenge: Find good codes for Secure Sketches**

# Example Code construction

**Challenge: Find good codes for Secure Sketches**

**Constraints:**

- Time and area consumption
- Binary codes
- Dimension $\geq$ key length
- Codelength as small as possible

# Example Code construction



**Existing scheme given in [Maes2012][1]:**

- Binary Symmetric Channel with $p = 0.14$
- Generate $128$ bit key with block error probability $\mathrm{P}_{\mathrm{err}} = 10^{-9}$
- Concatenation of $(318, 174, 35)$ BCH code and $(7, 1, 7)$ code

[1] R. Maes, A. Herrewege, I. Verbauwhede, "PUFKY: A Fully Functional PUF-Based Cryptographic Key Generator", CHES, 2012

# Example Code construction

**Existing scheme given in [Maes2012][1]:**

- Binary Symmetric Channel with $p = 0.14$
- Generate $128$ bit key with block error probability $\mathrm{P}_{\mathrm{err}} = 10^{-9}$
- Concatenation of $(318, 174, 35)$ BCH code and $(7, 1, 7)$ code

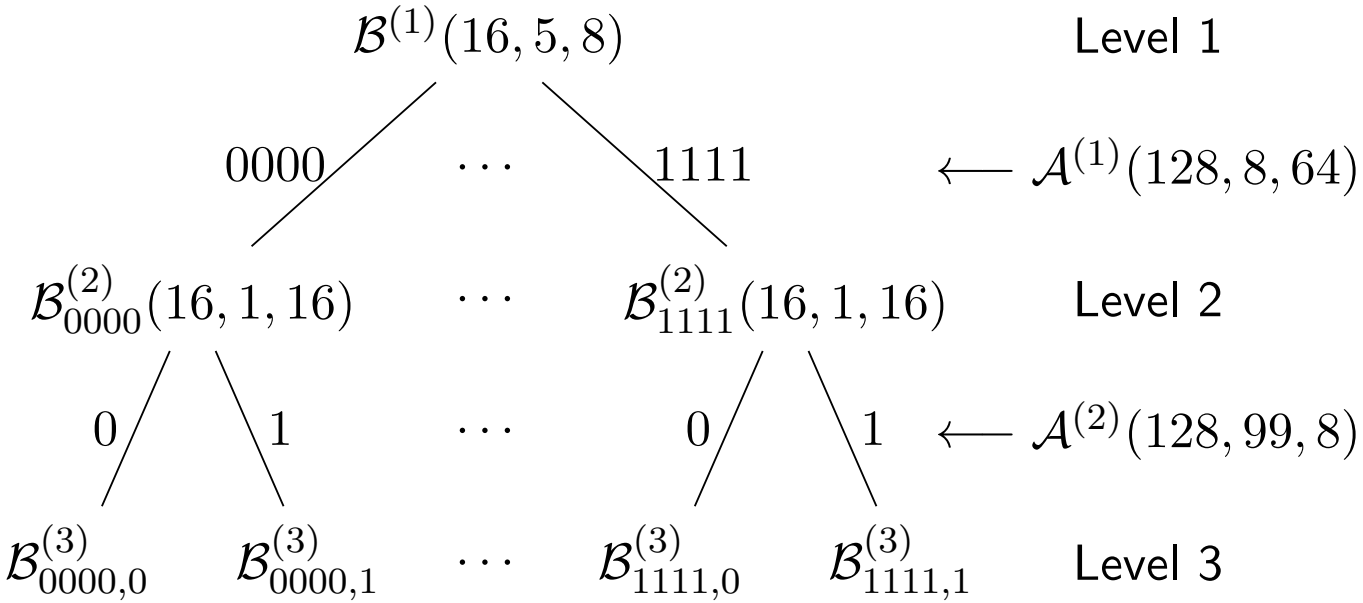
**What is our goal?**

- Generate $128$ bit key with block error probability $\mathrm{P}_{\mathrm{err}} < 10^{-9}$
- Improve existing scheme in
  - Codelength
  - Block error probability
  - Simple implementation

[1] R. Maes, A. Herrewege, I. Verbauwhede, "PUFKY: A Fully Functional PUF-Based Cryptographic Key Generator", CHES, 2012

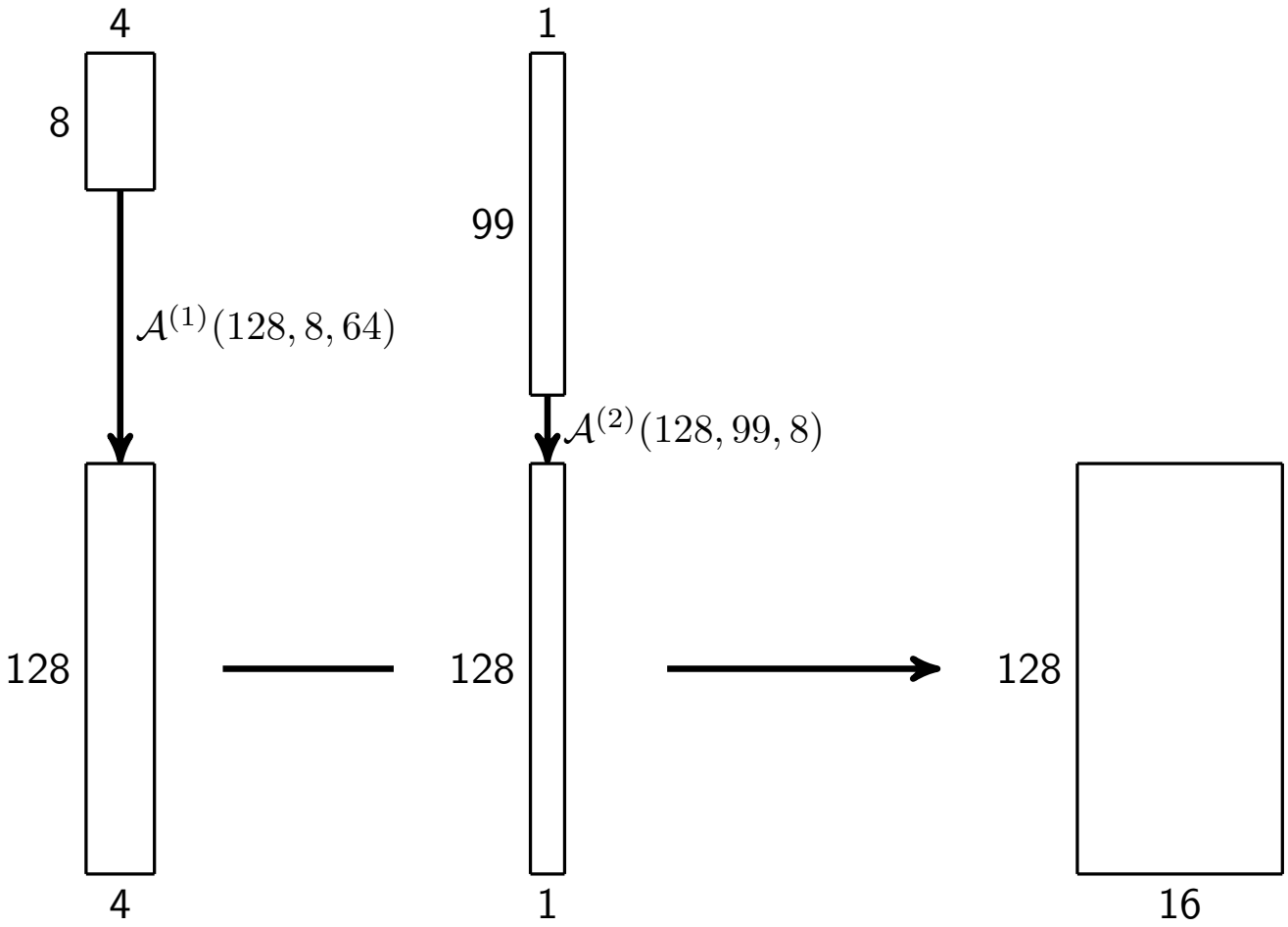
# Reed-Muller Example Code Construction

**Partitioning:**



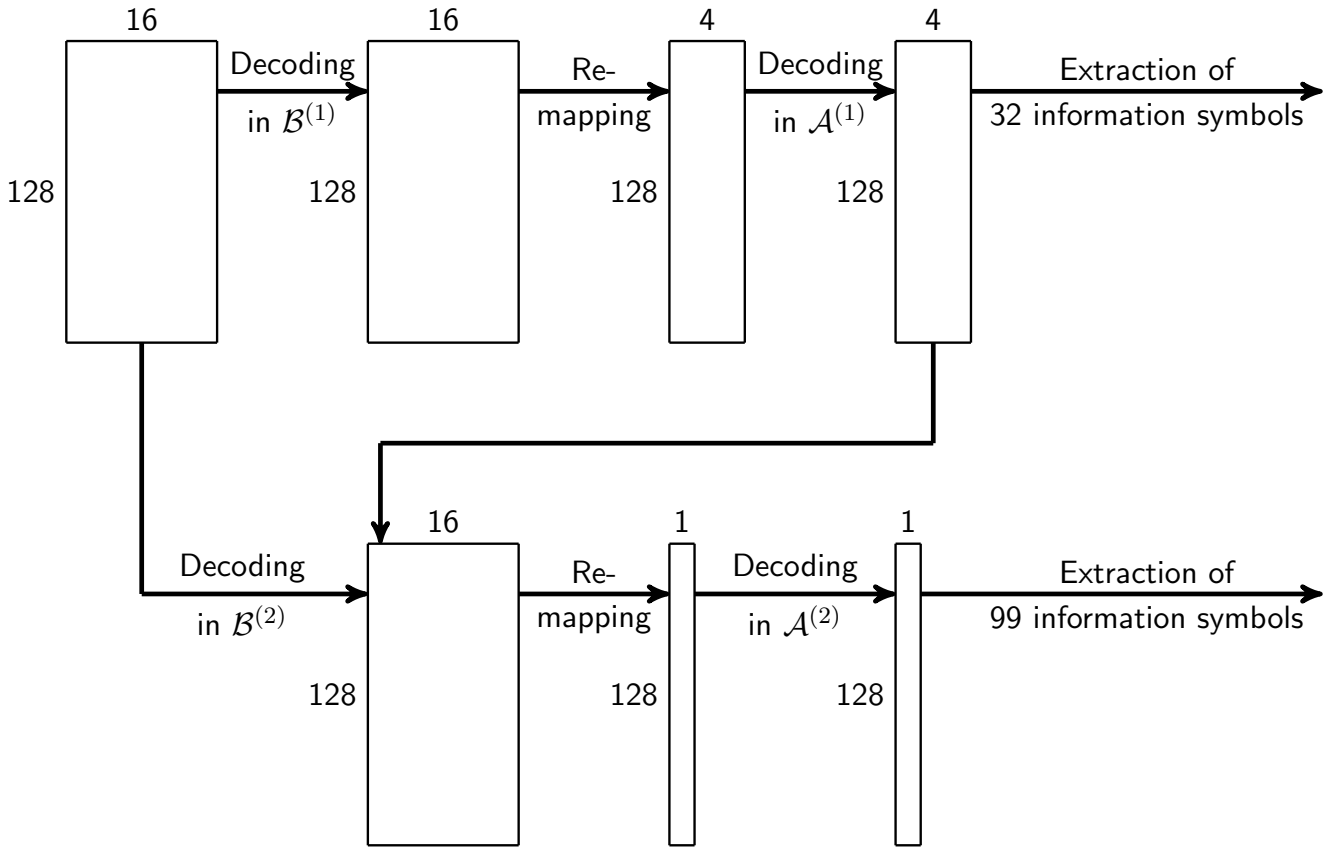
# Reed-Muller Example Code Construction

**Encoding:**



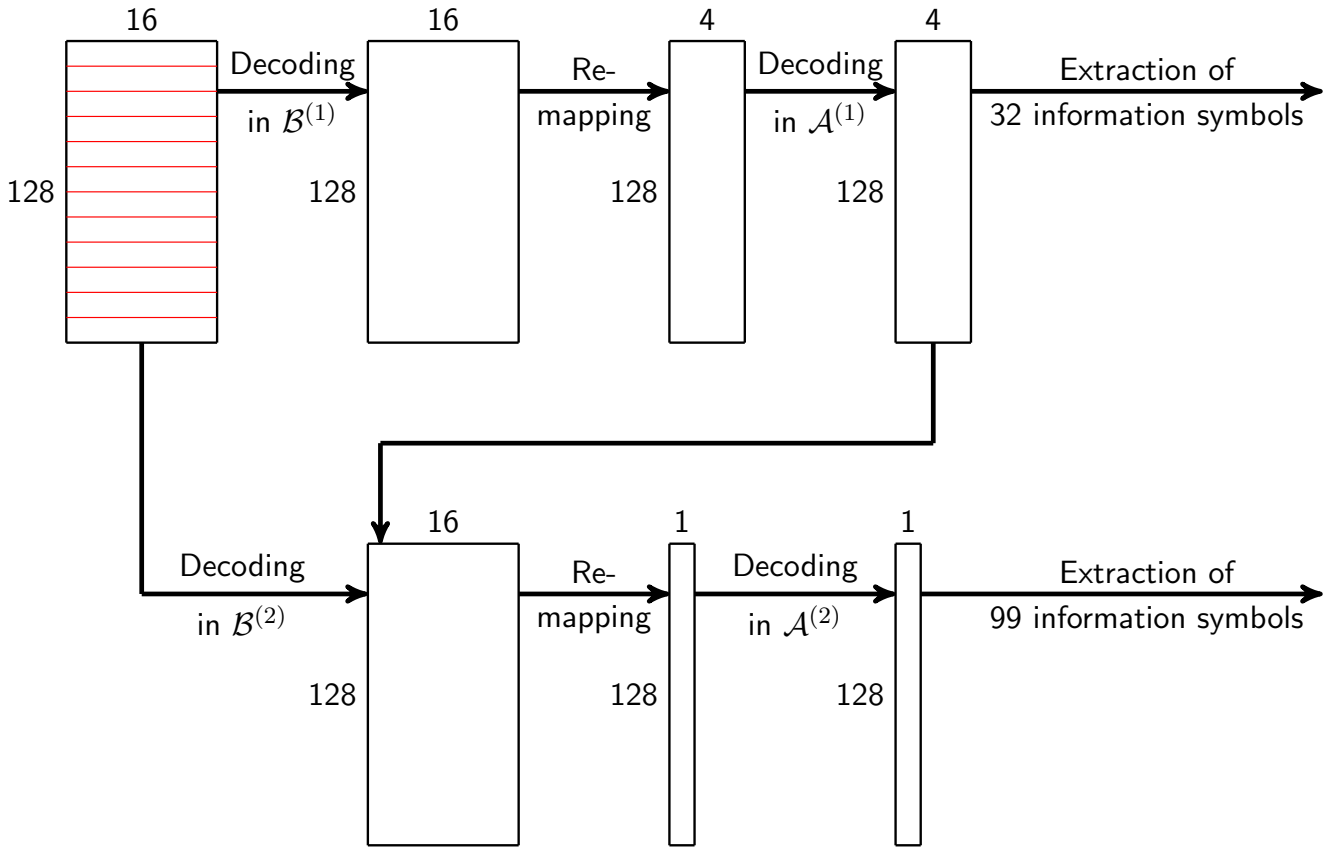
# Reed-Muller Example Code Construction

## Decoding:



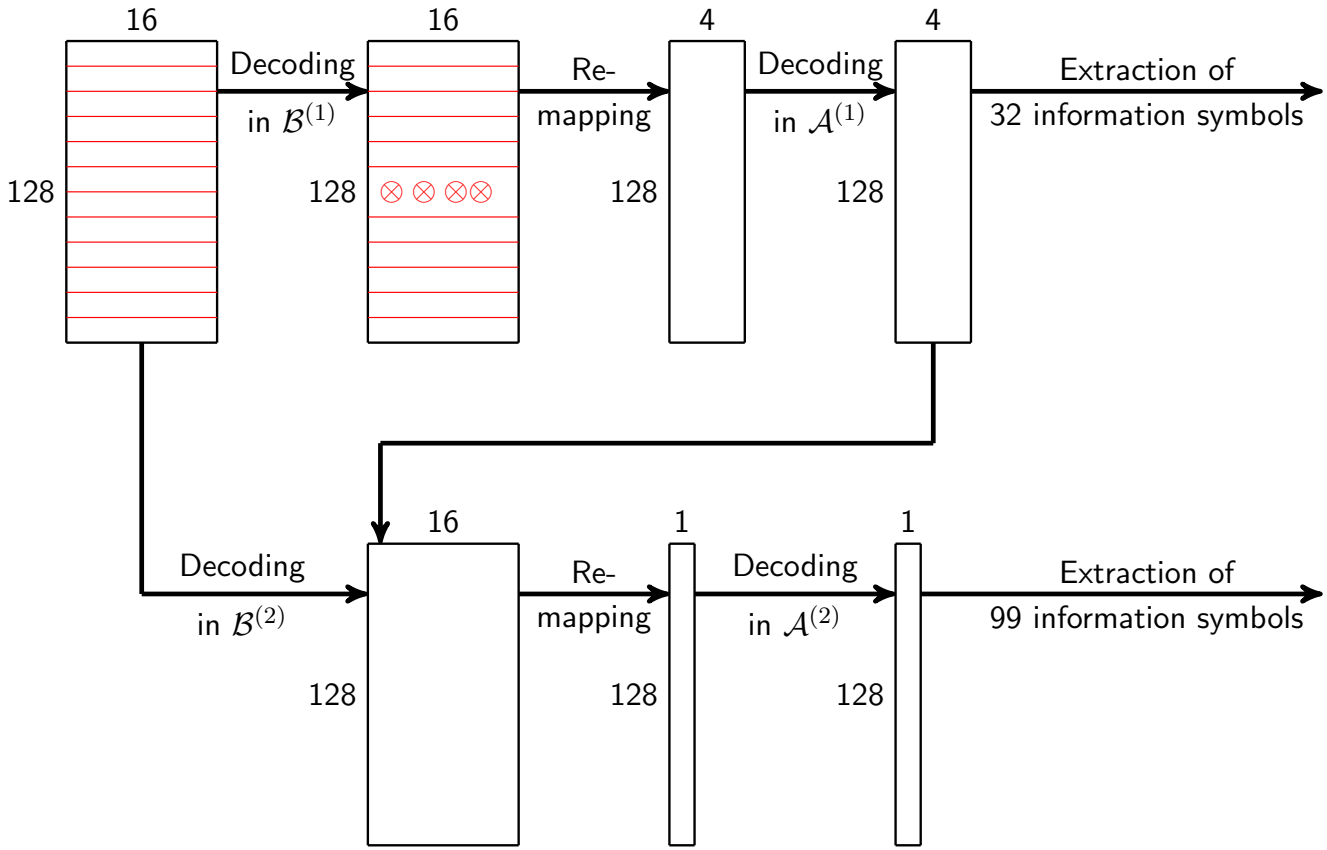
# Reed-Muller Example Code Construction

## Decoding:



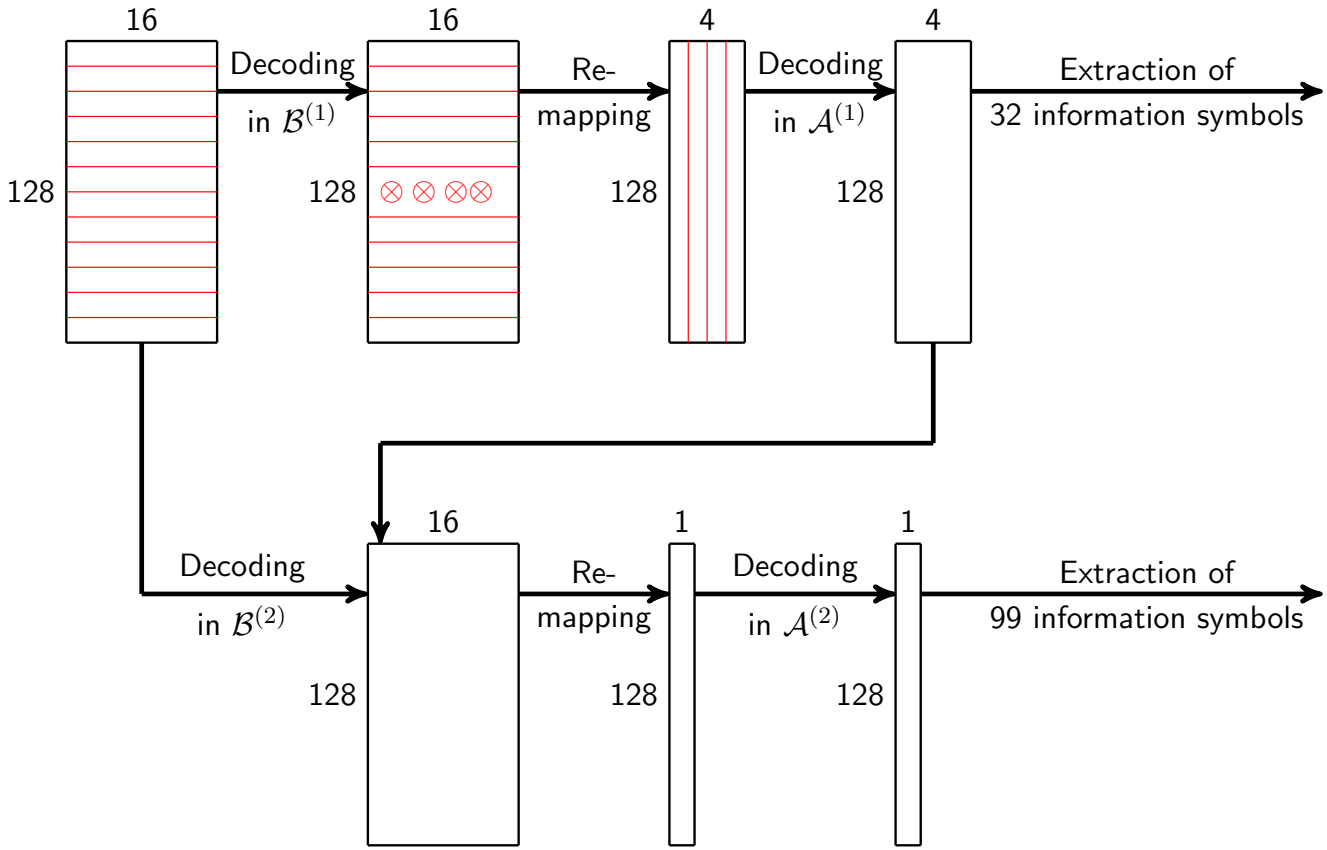
# Reed-Muller Example Code Construction

## Decoding:



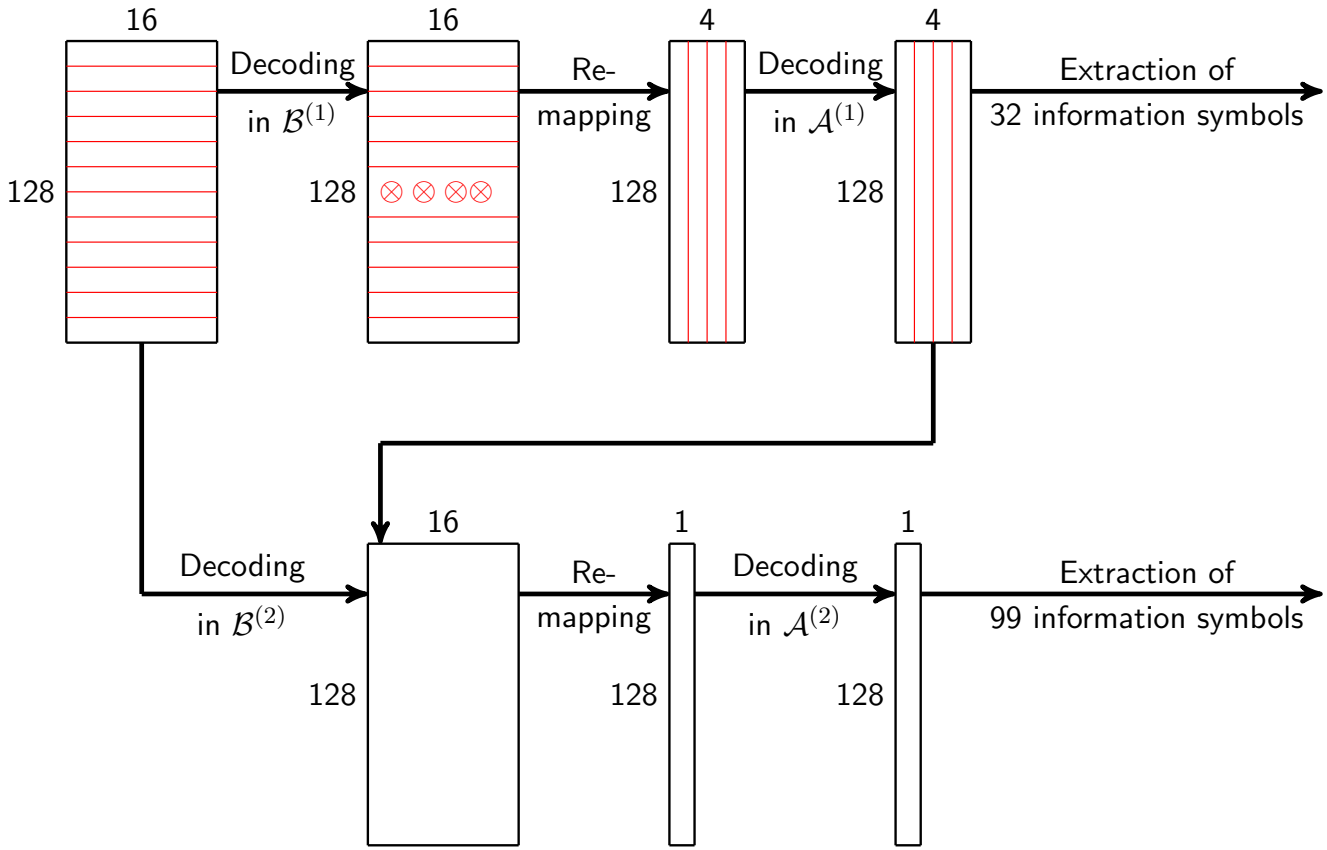
# Reed-Muller Example Code Construction

## Decoding:



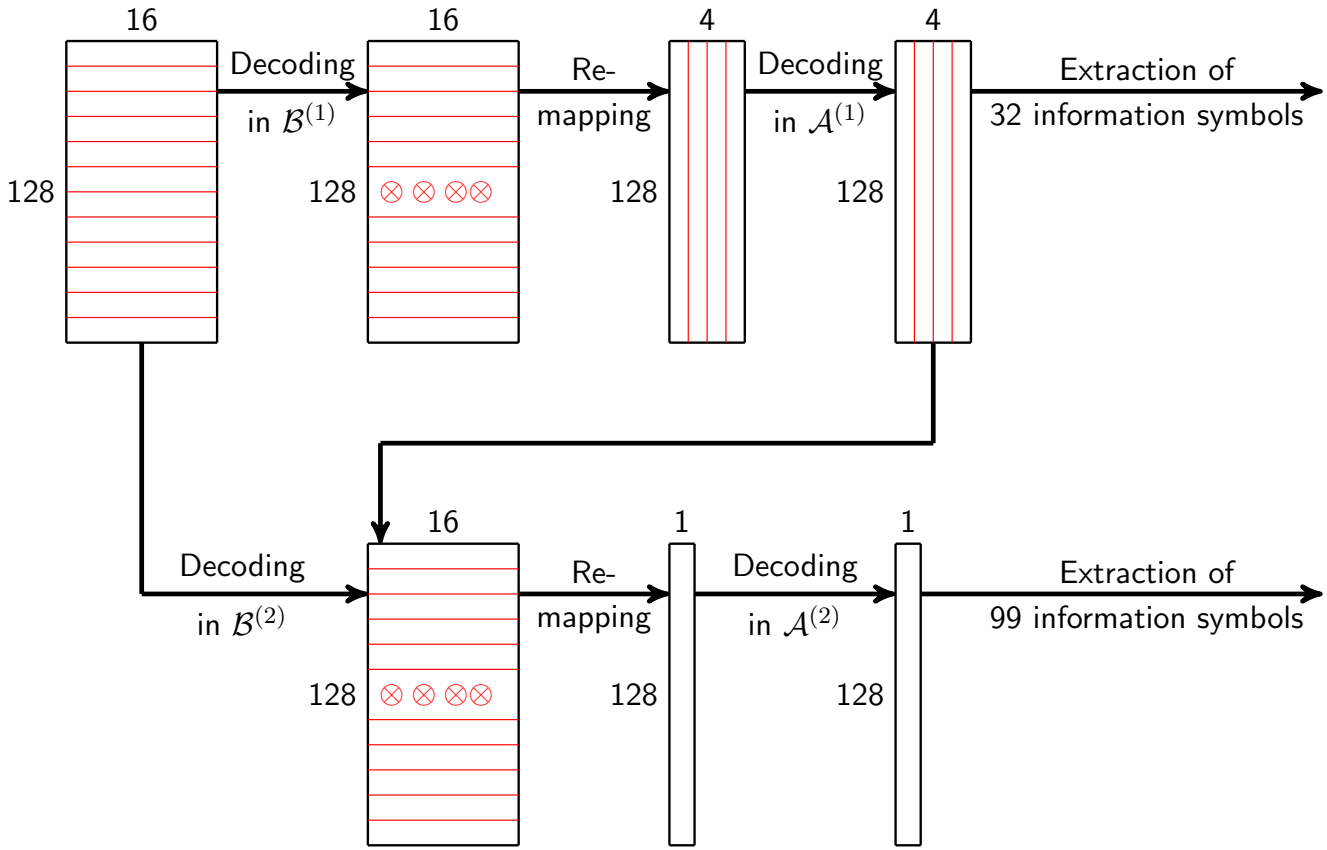
# Reed-Muller Example Code Construction

## Decoding:



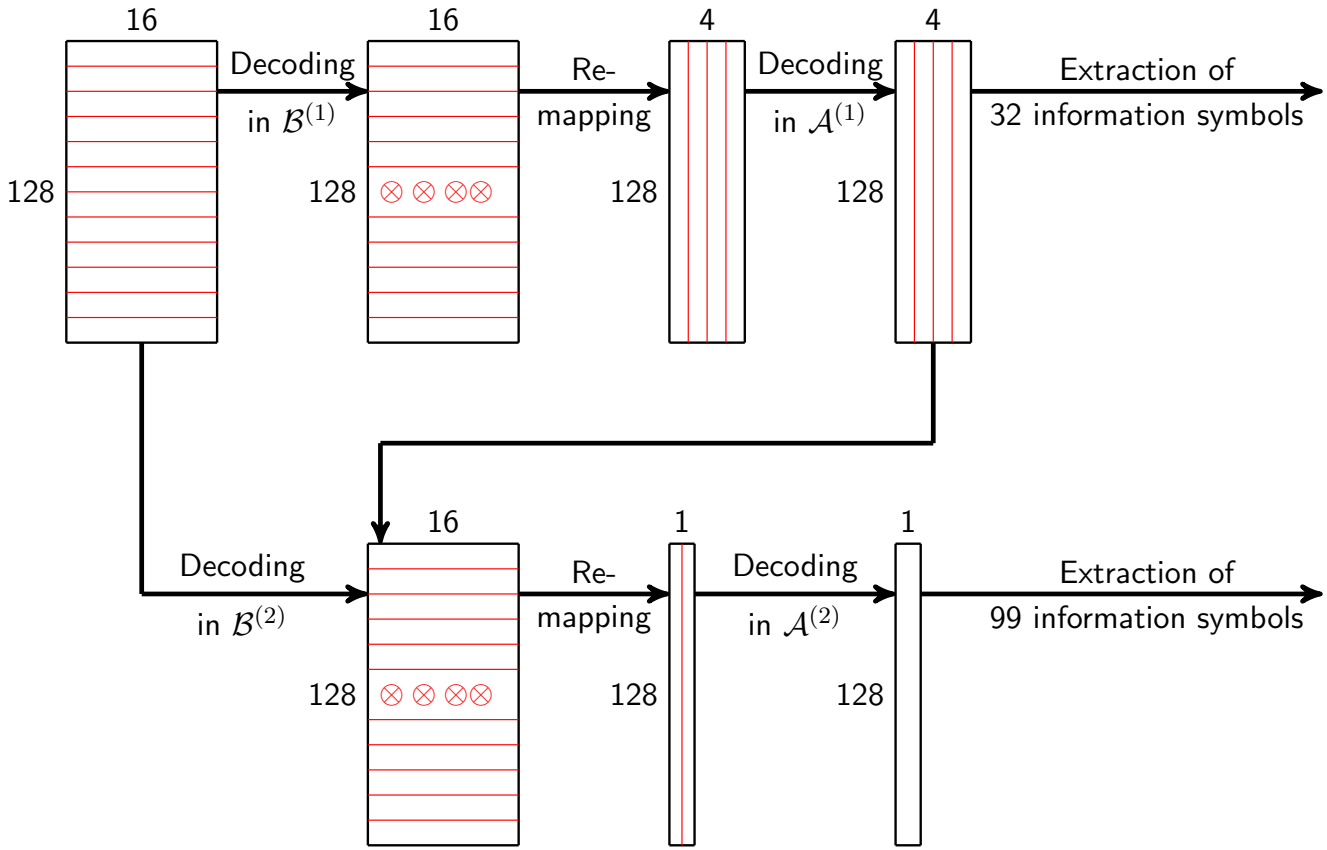
# Reed-Muller Example Code Construction

## Decoding:



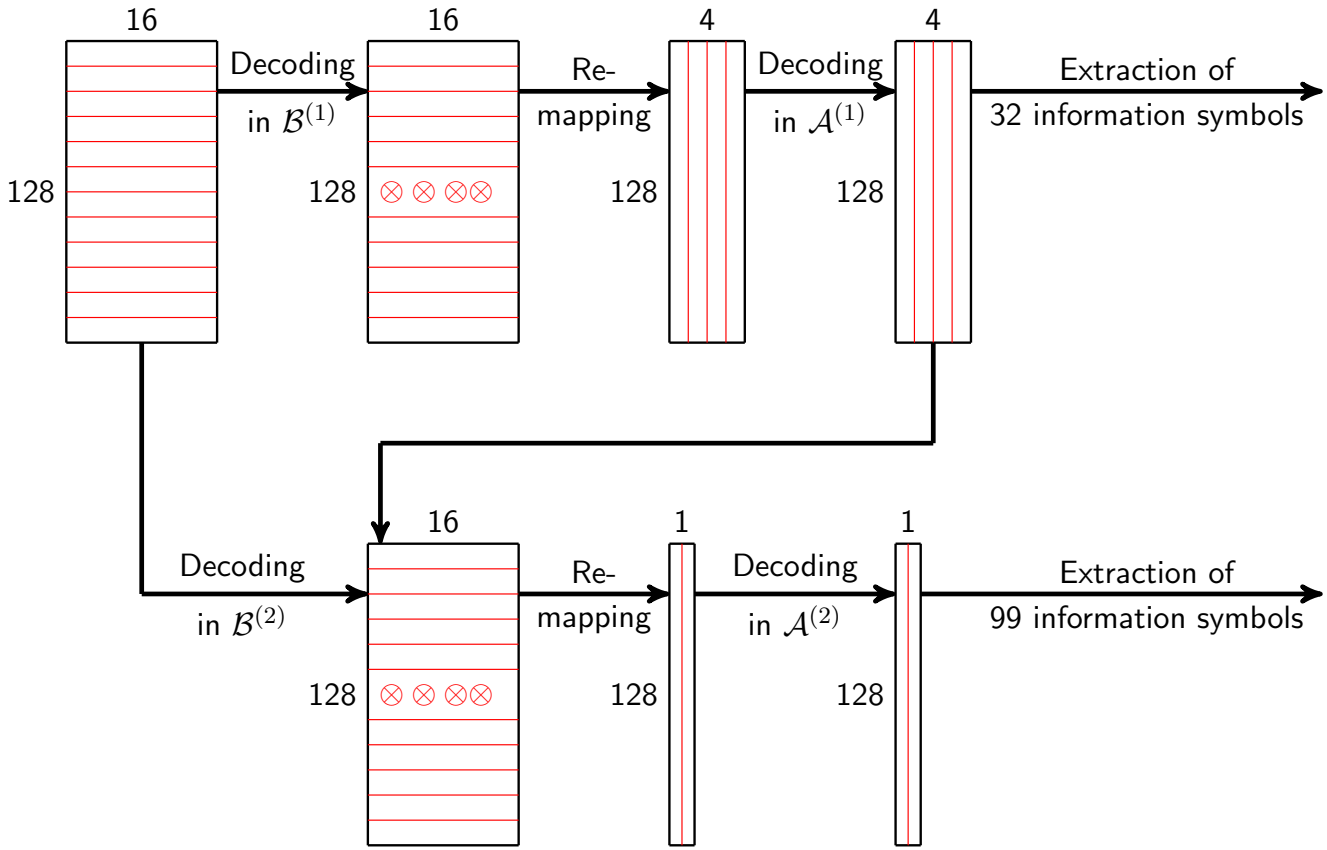
# Reed-Muller Example Code Construction

## Decoding:



# Reed-Muller Example Code Construction

## Decoding:



# Reed-Muller Example Code Construction



**Used decoding methods:**

- Generalized Concatenated Codes (GC Codes)
- RM Error Erasure Decoding
- Generalized Minimum Distance (GMD) Decoding

# Outline

# Conclusion

**How good is our code construction?**

| Code | $\mathrm{P}_{\mathrm{err}}$ | Length | Largest Field |
|---|---|---|---|
| [Maes2012] | $\approx 10^{-9}$ | 2226 | $\mathbb{F}_{2^8}$ (BCH) |
| New | $\approx 5.37 \cdot 10^{-10}$ | 2048 | $\mathbb{F}_2$ |

Thank you for your attention.