
ulm university universität ulm

# Physical Unclonable Functions (PUFs) and Error Correction

Martin Bossert
with Sven Müelich and Sven Puchinger

Institute of Communications Engineering, Ulm University, Germany

Banff, October 15, 2015

# Physical Unclonable Functions (PUFs)



**Challenges when implementing a cryptosystem:**

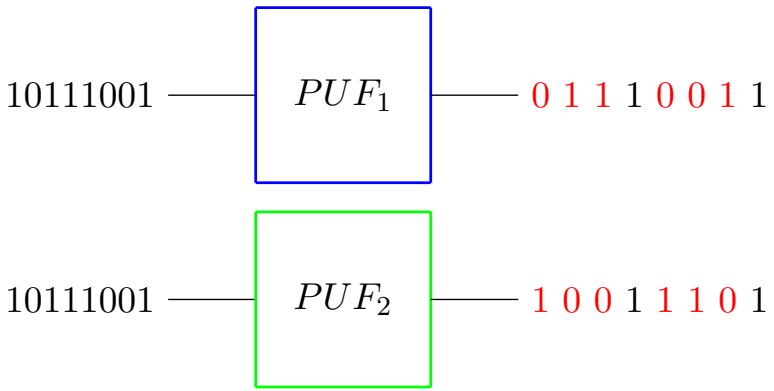
- Secure key generation
  - Random, unique and unpredictable keys
  - Satisfying these properties is hard to achieve
- Secure key storage
  - Key bits in a non-volatile memory
  - Adversaries can gain physical access to (protected) memories

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

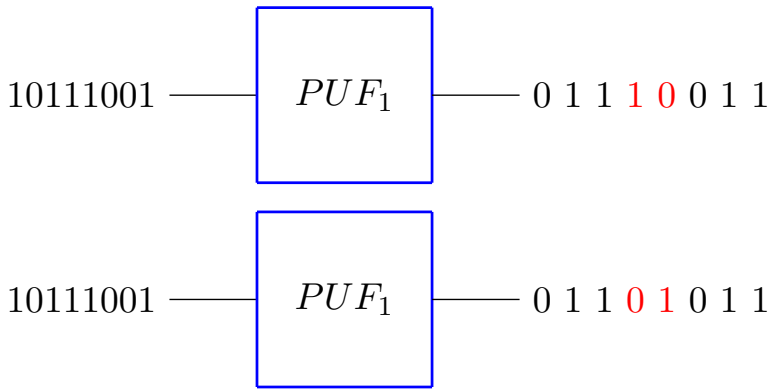
# Physical Unclonable Functions (PUFs)

## What is a PUF?

- Physical entity with challenge-response behavior
- Properties:
  - Uniqueness and Randomness
  - Reproducibility
  - Physical unclonability



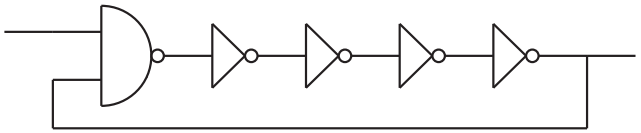

Uniqueness



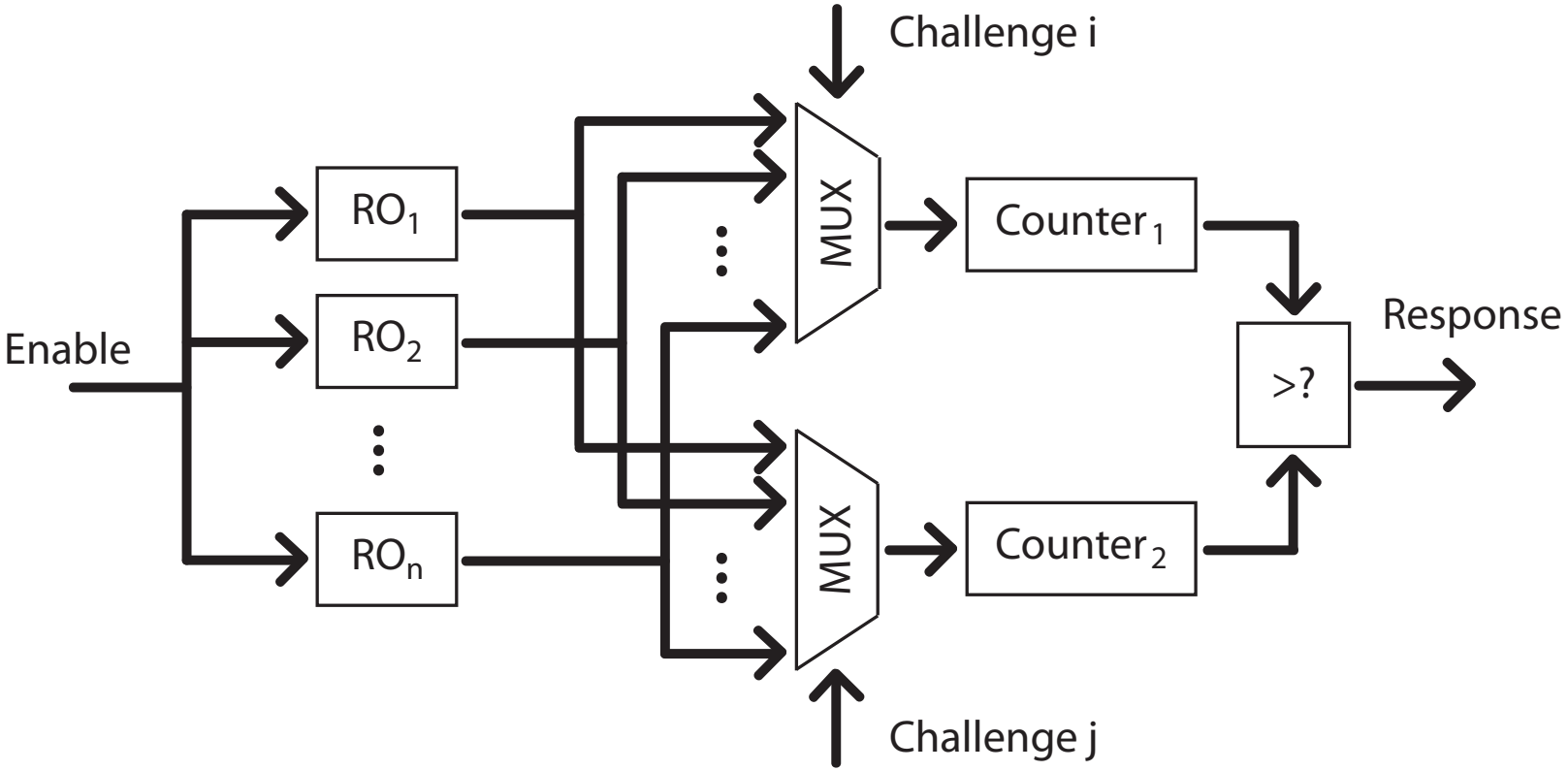

Reproducibility

# Example for a Delay-based PUF

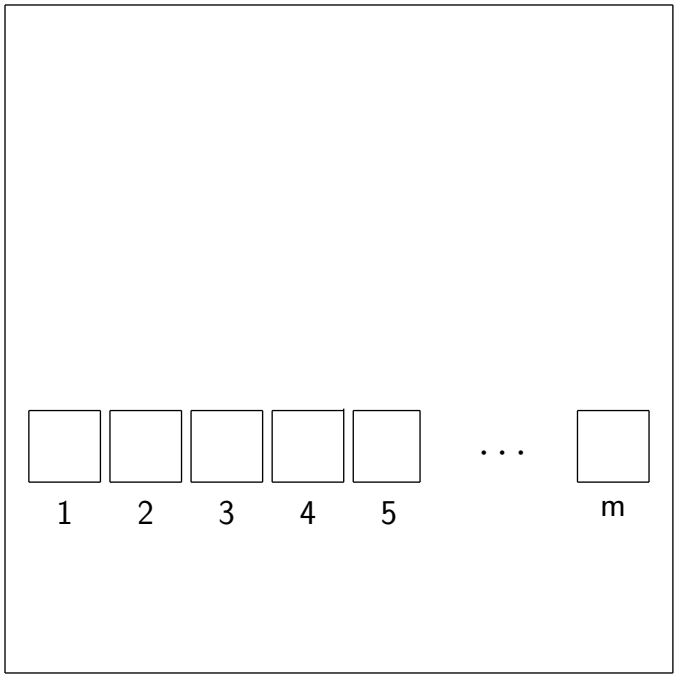




Ring Oscillator (RO) with logic gates

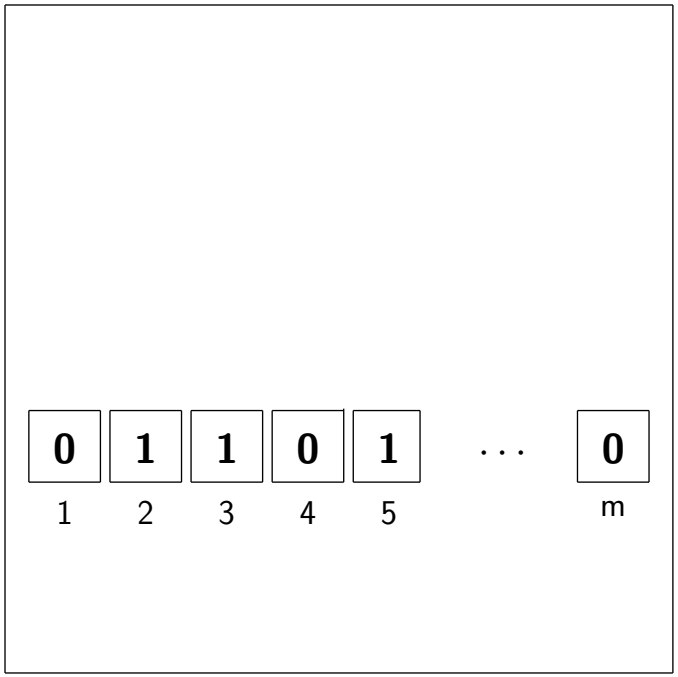


# Example for a Memory-based PUF



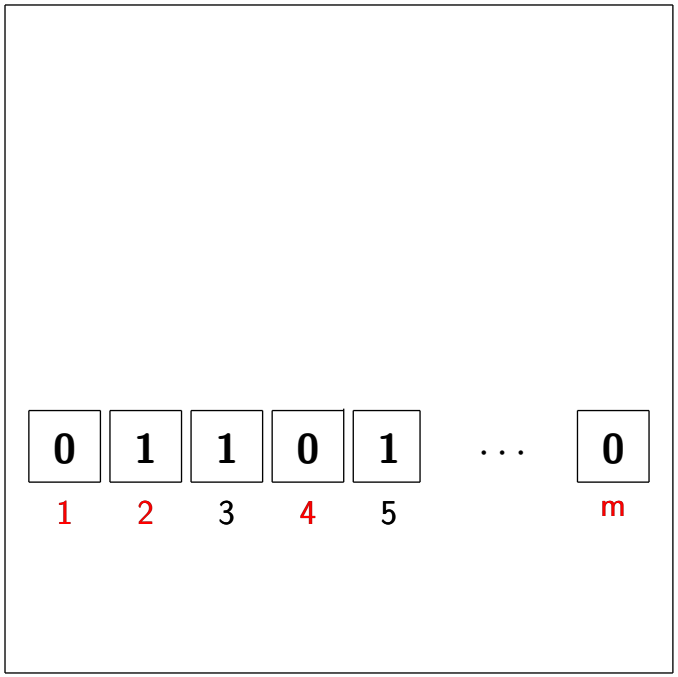

- SRAM PUF, Butterfly PUF, Latch PUF, Flipflop PUF
- device with memory cells

# Example for a Memory-based PUF



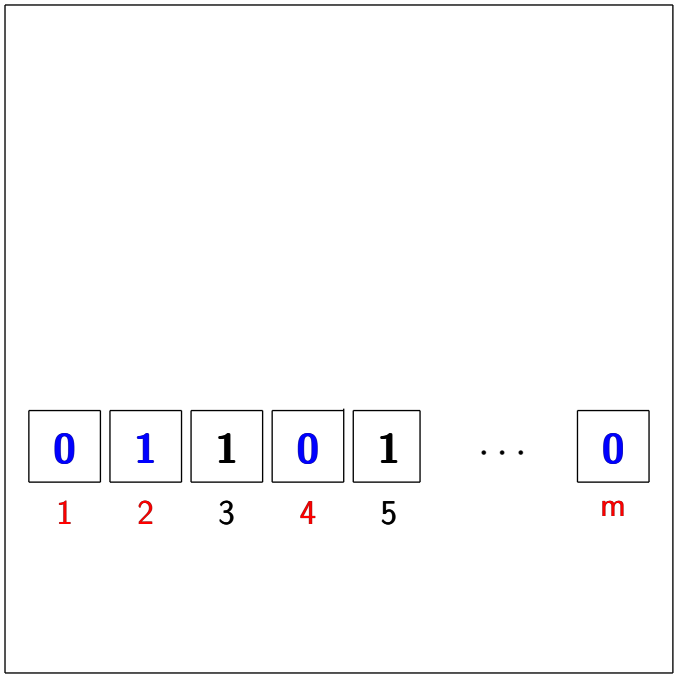

- SRAM PUF, Butterfly PUF, Latch PUF, Flipflop PUF
- device with memory cells
- random initialization when powering on
- randomness static over lifetime

# Example for a Memory-based PUF




- SRAM PUF, Butterfly PUF, Latch PUF, Flipflop PUF
- device with memory cells
- random initialization when powering on
- randomness static over lifetime
- Challenge: Subset of memory cells

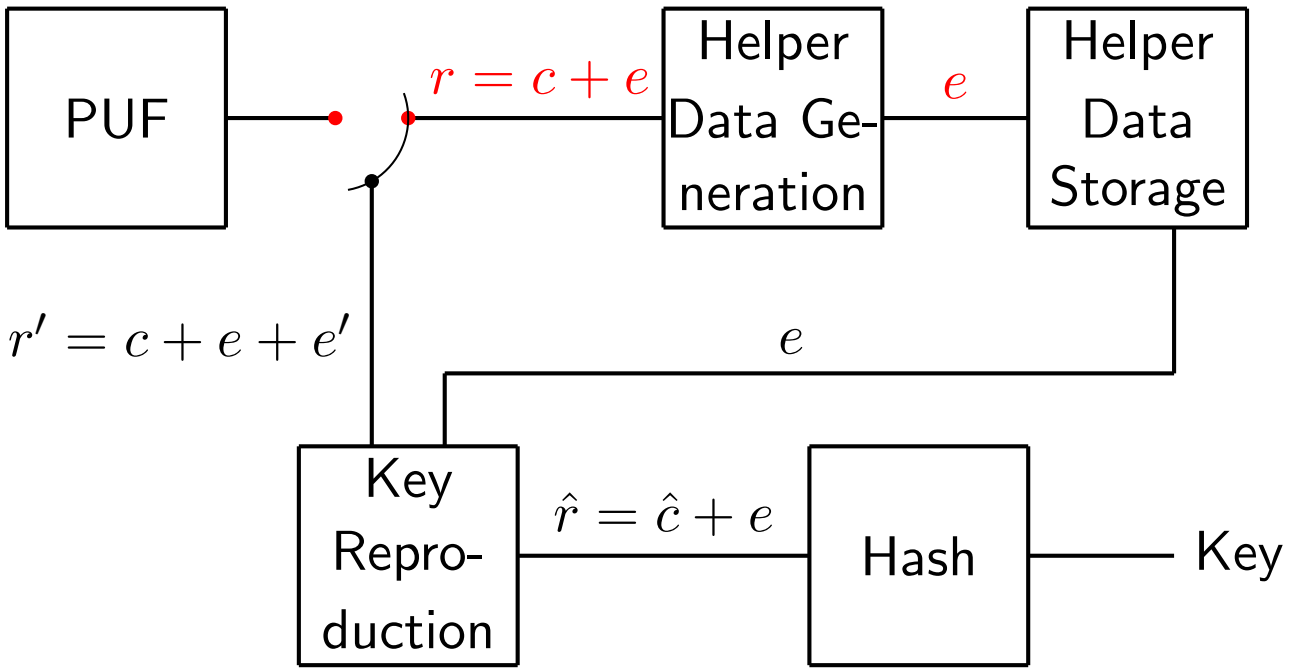
# Example for a Memory-based PUF




- SRAM PUF, Butterfly PUF, Latch PUF, Flipflop PUF
- device with memory cells
- random initialization when powering on
- randomness static over lifetime
- Challenge: Subset of memory cells
- Response: Values in selected memory cells

# Need for Coding Theory

Responses are not perfectly reproducible due to aging, temperature, etc. and hence cannot be used as key directly




Init: coset - code  aging/temperature error: $e'$

# Code Construction



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

**Constraints:**

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

**Typical goal:**

- Design code with block error probability $\mathrm{P}_{\mathrm{err}}$ smaller than a certain threshold

# New Code Construction



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

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

**New Construction should:**

- Generate $128$ bit key with block error probability $\mathrm{P_{err}} < 10^{-9}$
- 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 Codes



**Recursive Construction:**

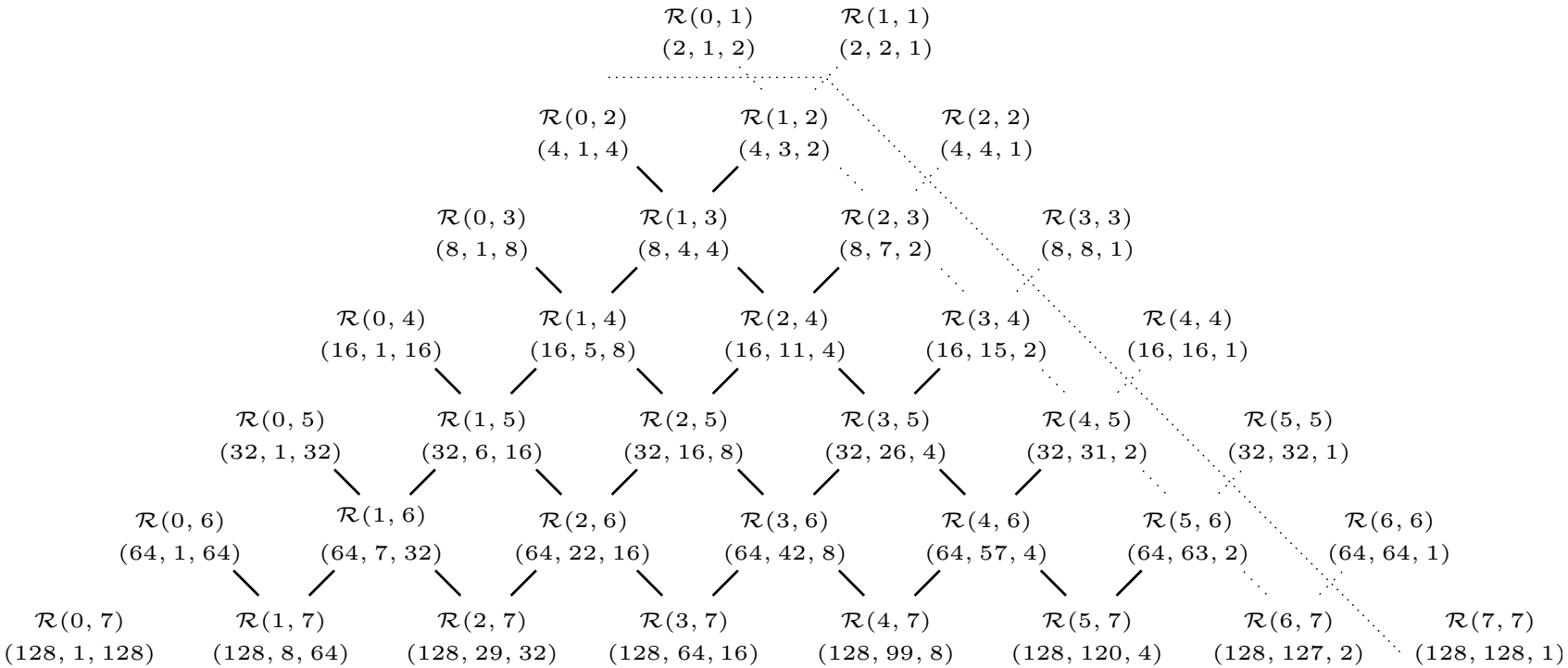
- Reed-Muller code $\mathcal{RM}(r, m)$
  - Binary linear code
  - Order of the code: $r \leq m$
  - Code length: $n = 2^m$
  - Dimension: $k = \sum_{i=0}^{r} \binom{n}{i}$
  - Minimum distance: $d = 2^{m-r}$
- Plotkin Construction

$$\mathcal{RM}(r, m) := \left\{ (\mathbf{u}|\mathbf{u}+\mathbf{v}) : \begin{array}{l} \mathbf{u} \in \mathcal{RM}(r, m-1) \\ \mathbf{v} \in \mathcal{RM}(r-1, m-1) \end{array} \right\}$$

# Reed-Muller Codes



## Recursive Construction:



# Reed-Muller and Generalized Concatenated Codes



**Why Reed-Muller Codes?**

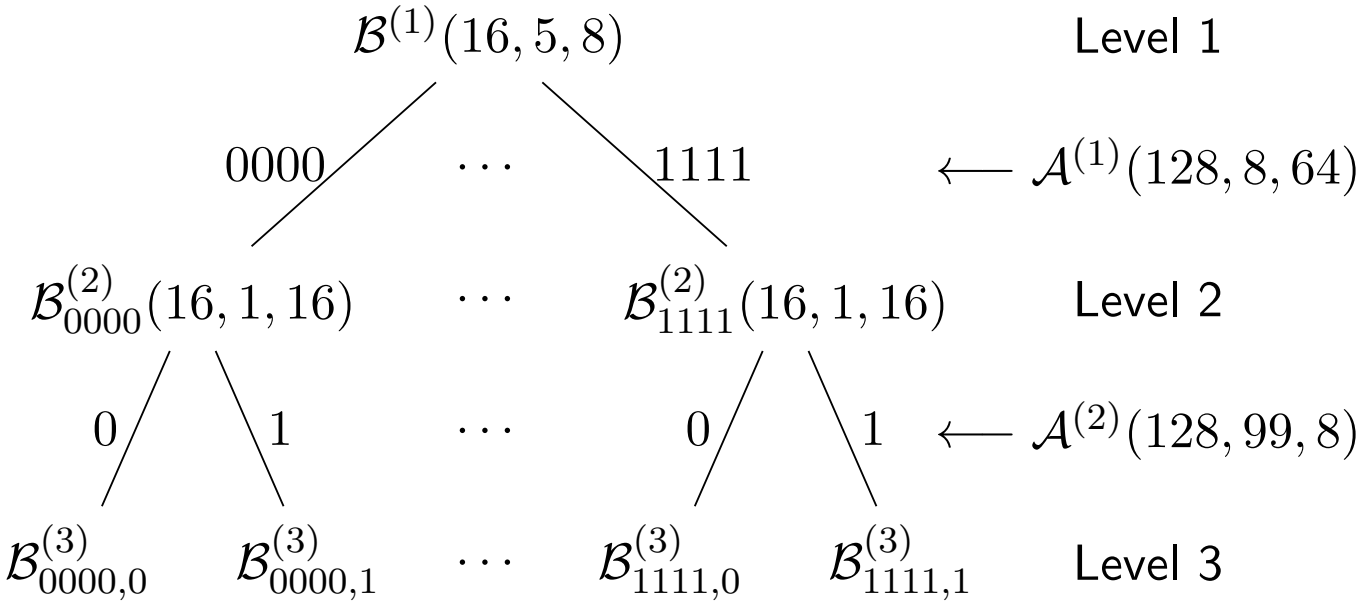
- Decoding easy to implement (recursively); complexity linear in $n$
- Up to $\tau$ errors and $\delta$ erasures correctable if $2\tau + \delta < d$
  $\Rightarrow$ enables GMD decoding
- Decoding of Repetition and Parity Check Codes as usual

**Why GC Codes?**

- Simple decoding
- More codewords than an ordinary concatenated code with same parameters $n$ and $d$
  $\Rightarrow$ higher coderate
- Decoding of many short codes instead of one long codes

# Reed-Muller Example Code Construction

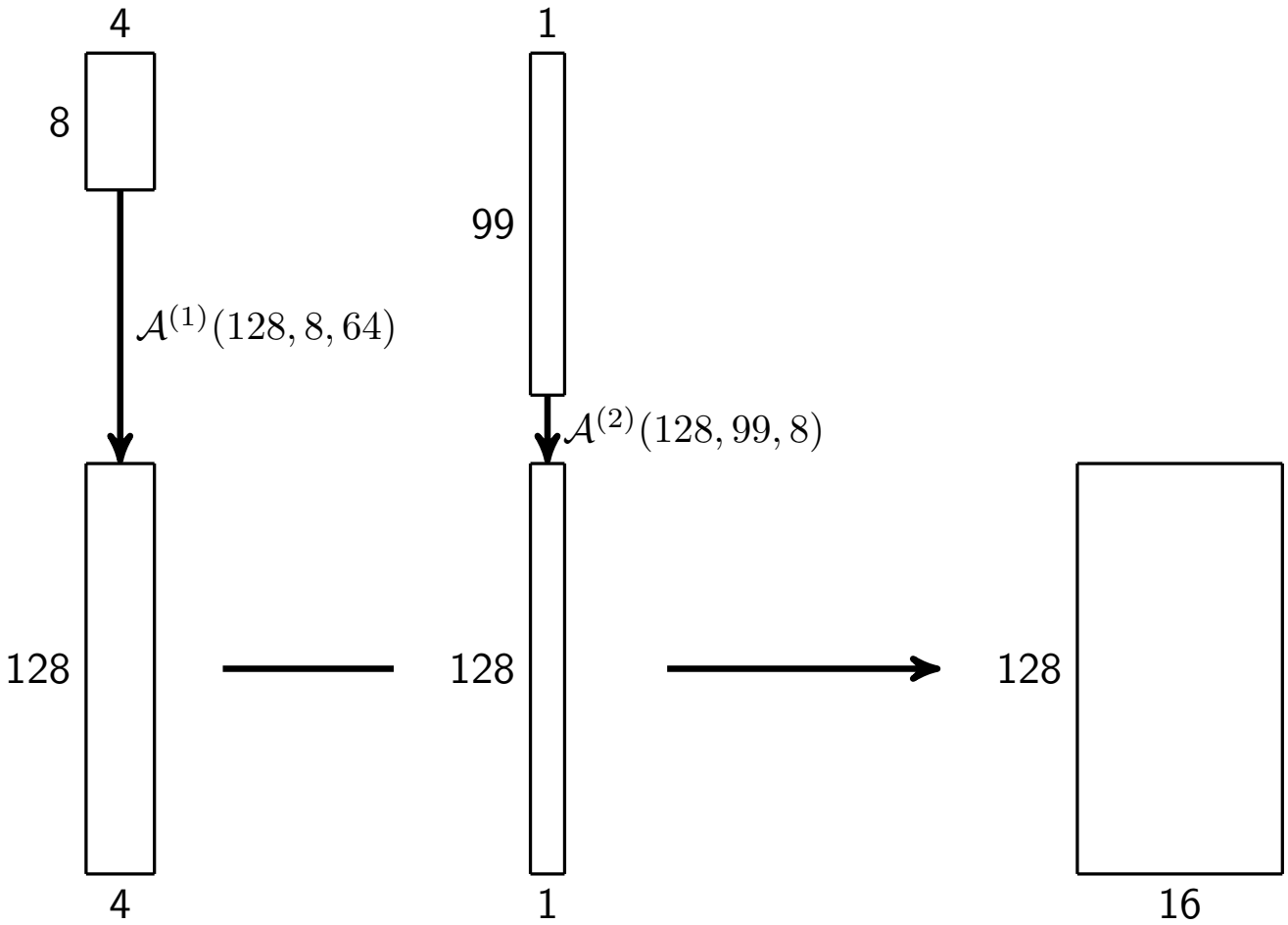
**Partitioning:**



# Reed-Muller Example Code Construction

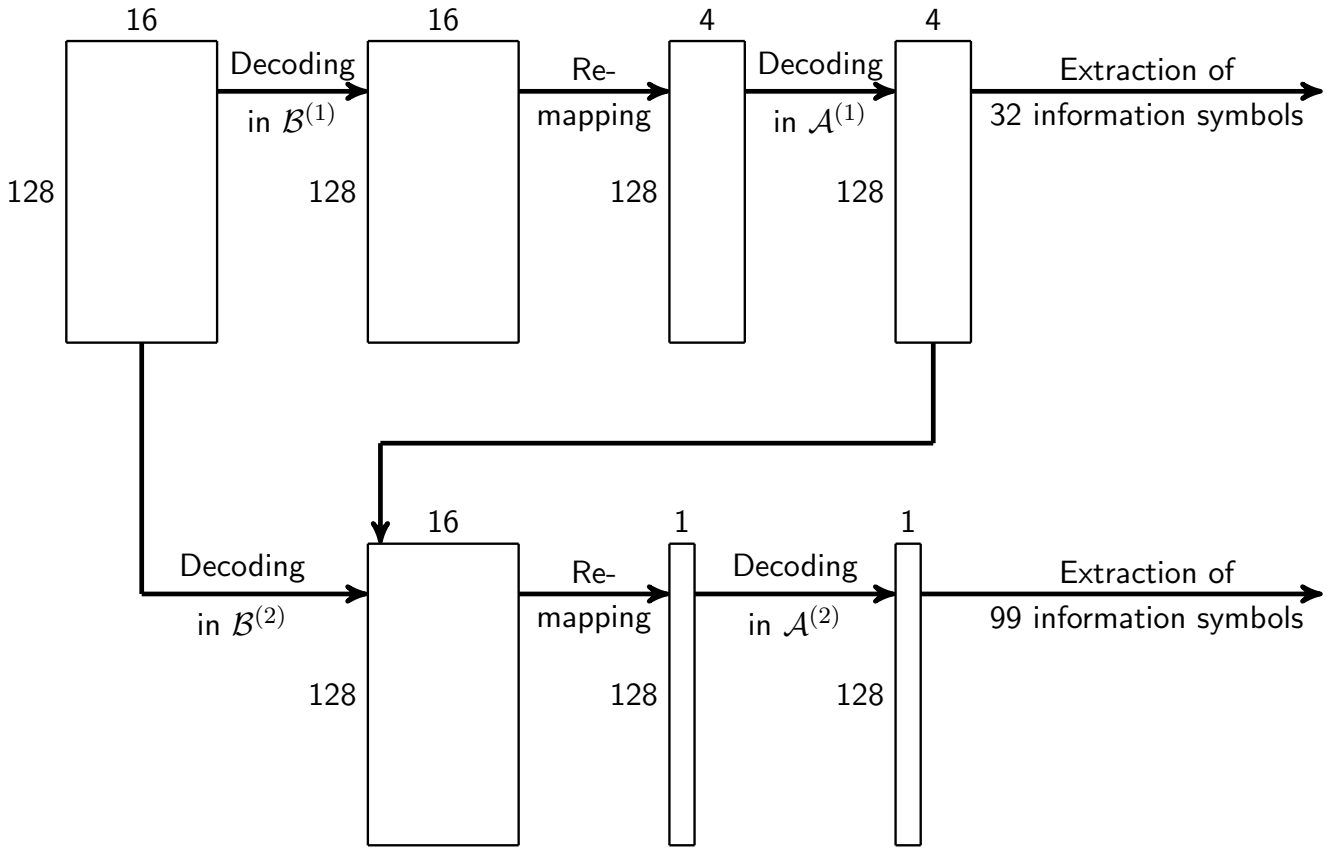


**Encoding:**



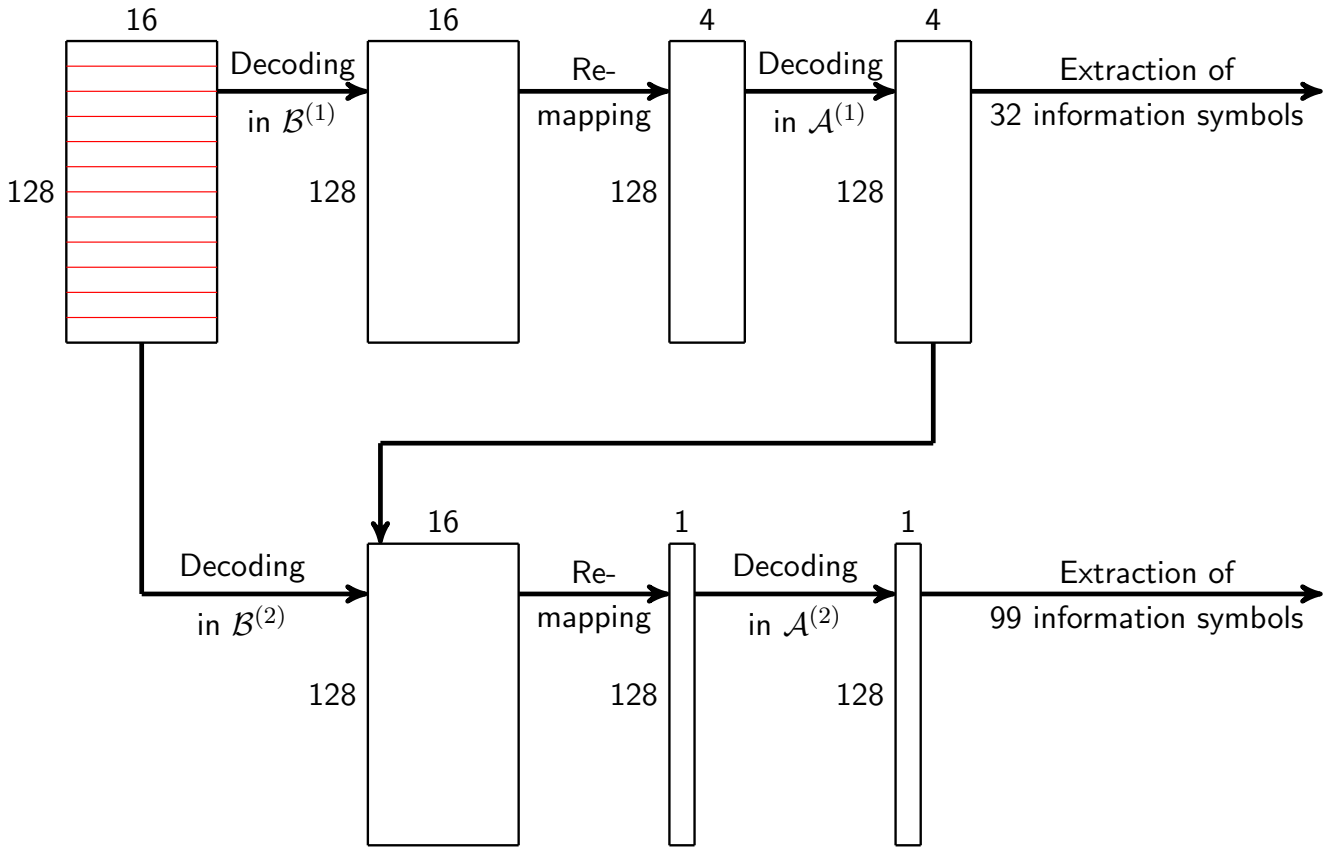
# Reed-Muller Example Code Construction

## Decoding:



# Reed-Muller Example Code Construction

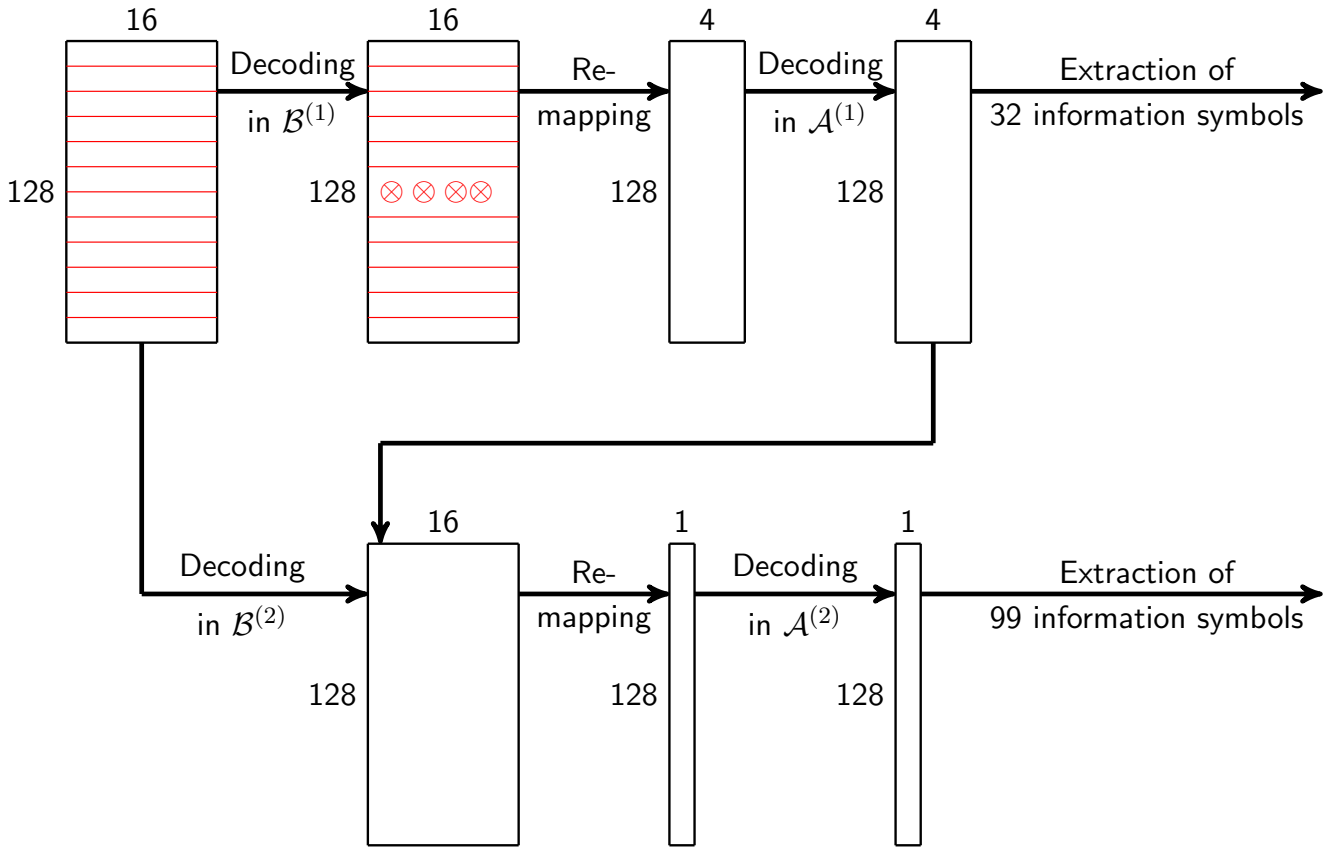
## Decoding:



# Reed-Muller Example Code Construction

NACHRICHTENTECHNIK Algebraic Coding

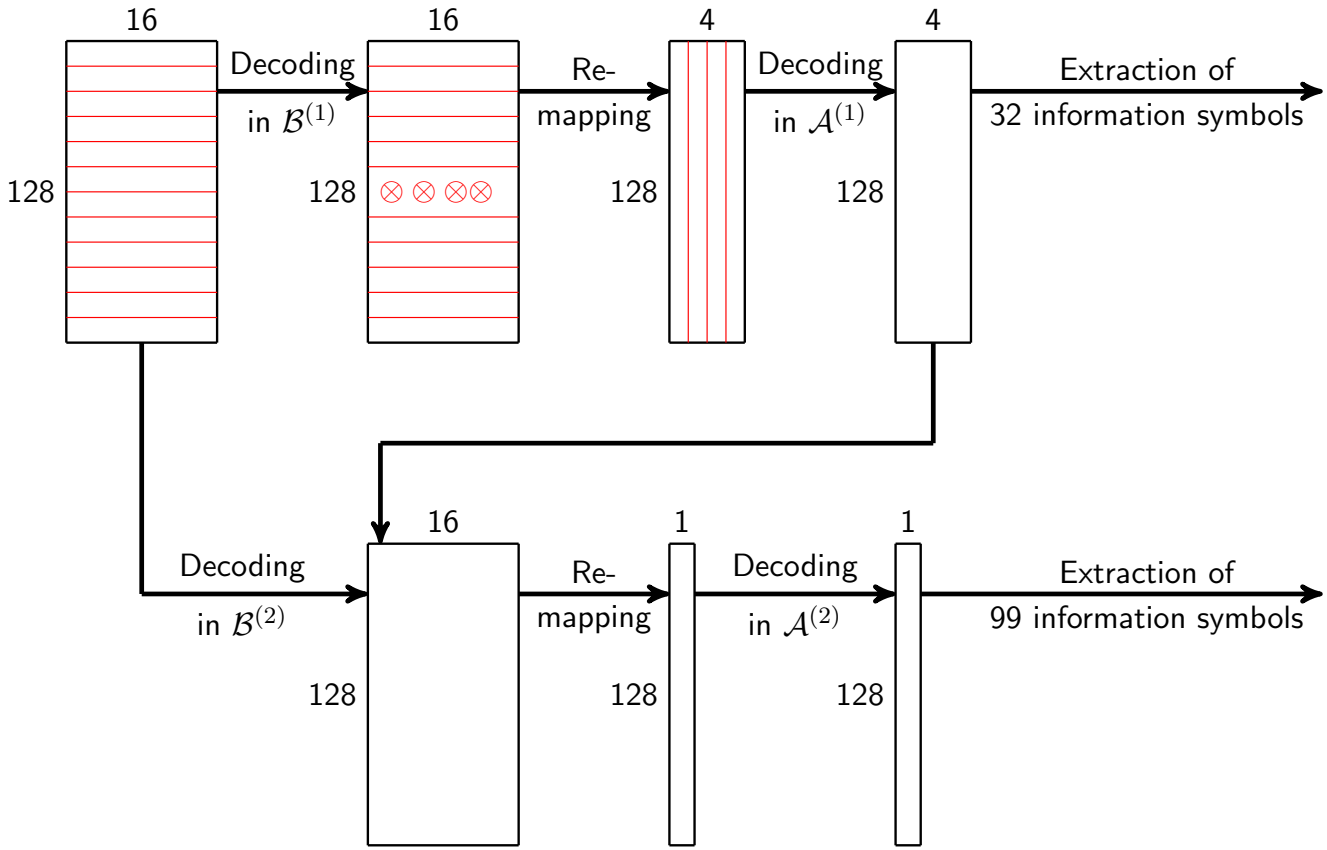
## Decoding:



# Reed-Muller Example Code Construction

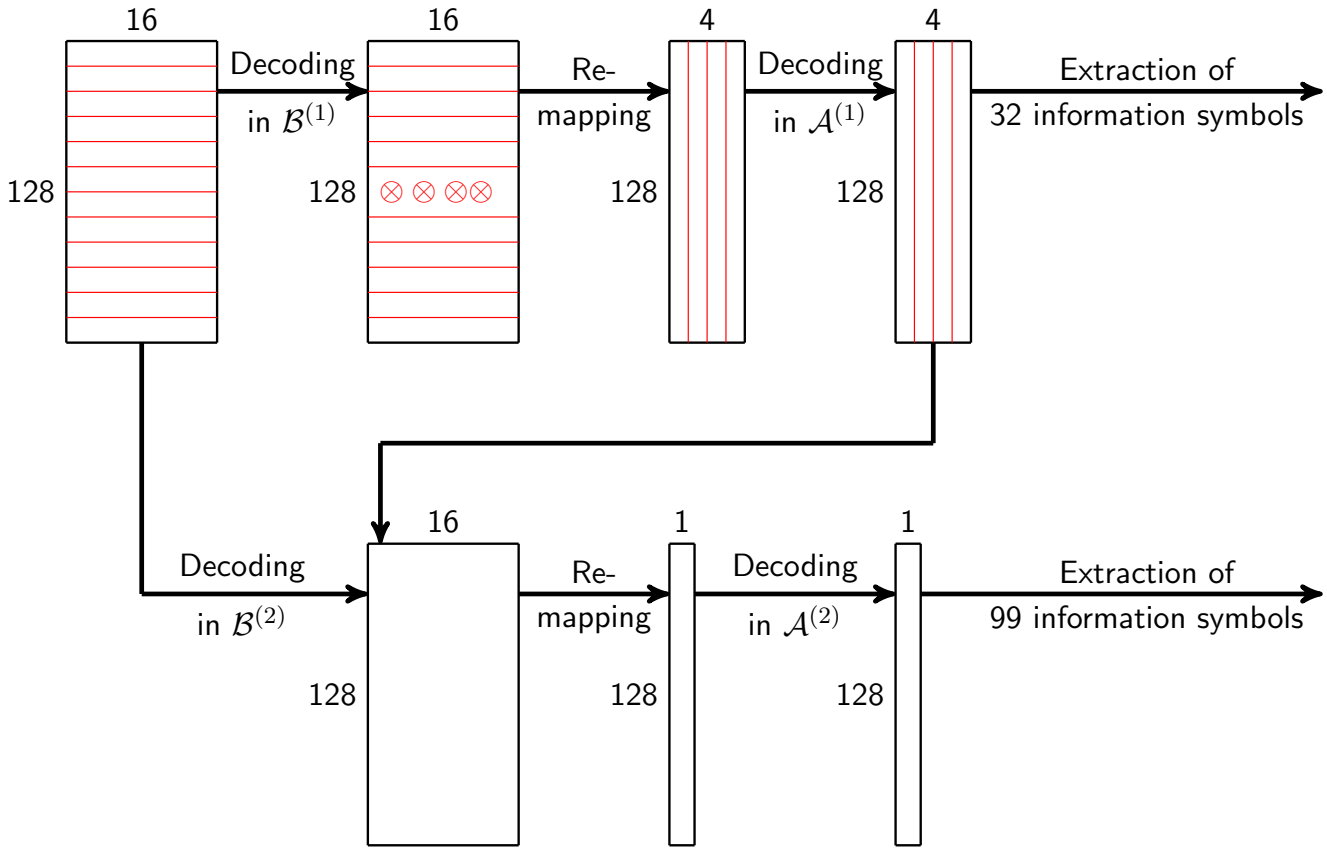
NACHRICHTENTECHNIK Algebraic Coding

## Decoding:



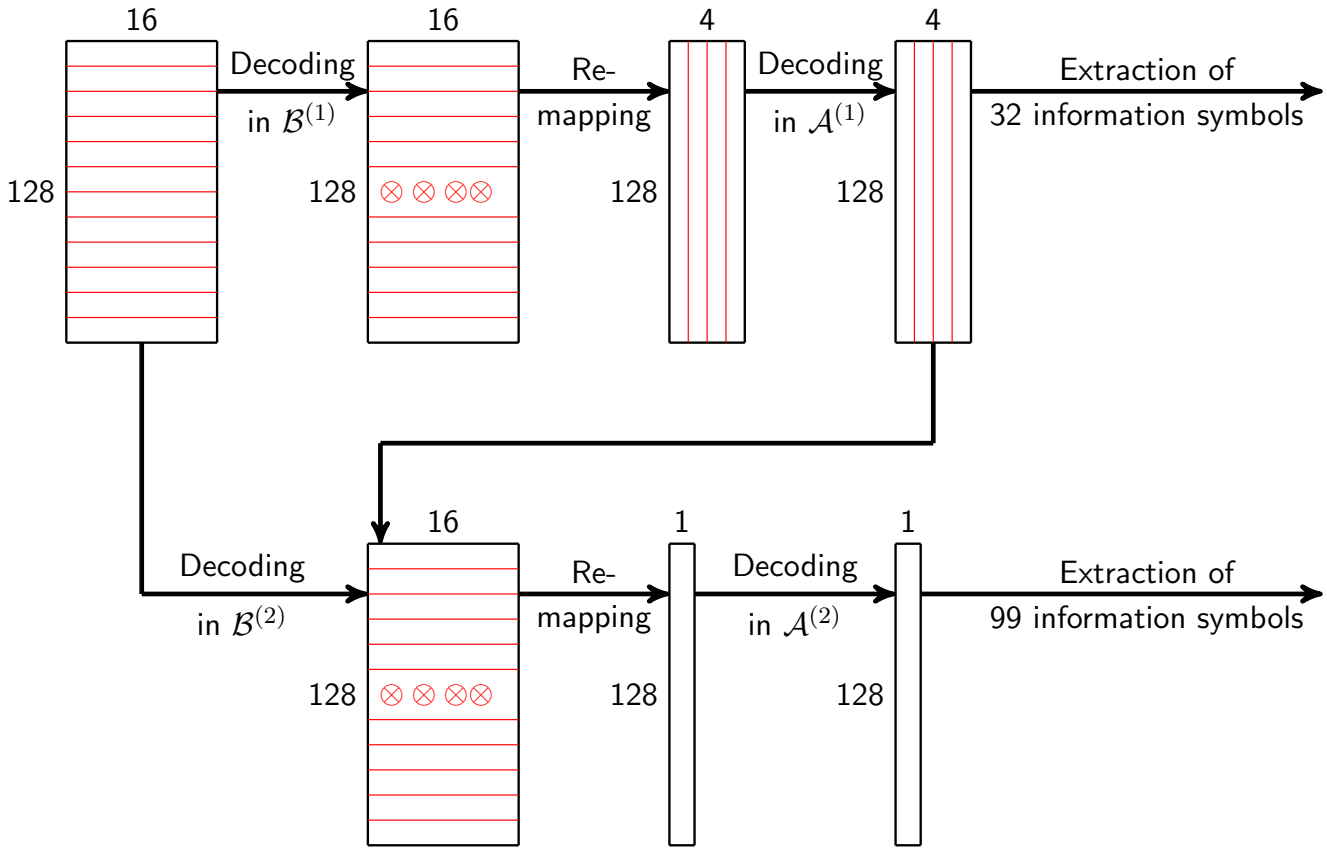
# Reed-Muller Example Code Construction

## Decoding:



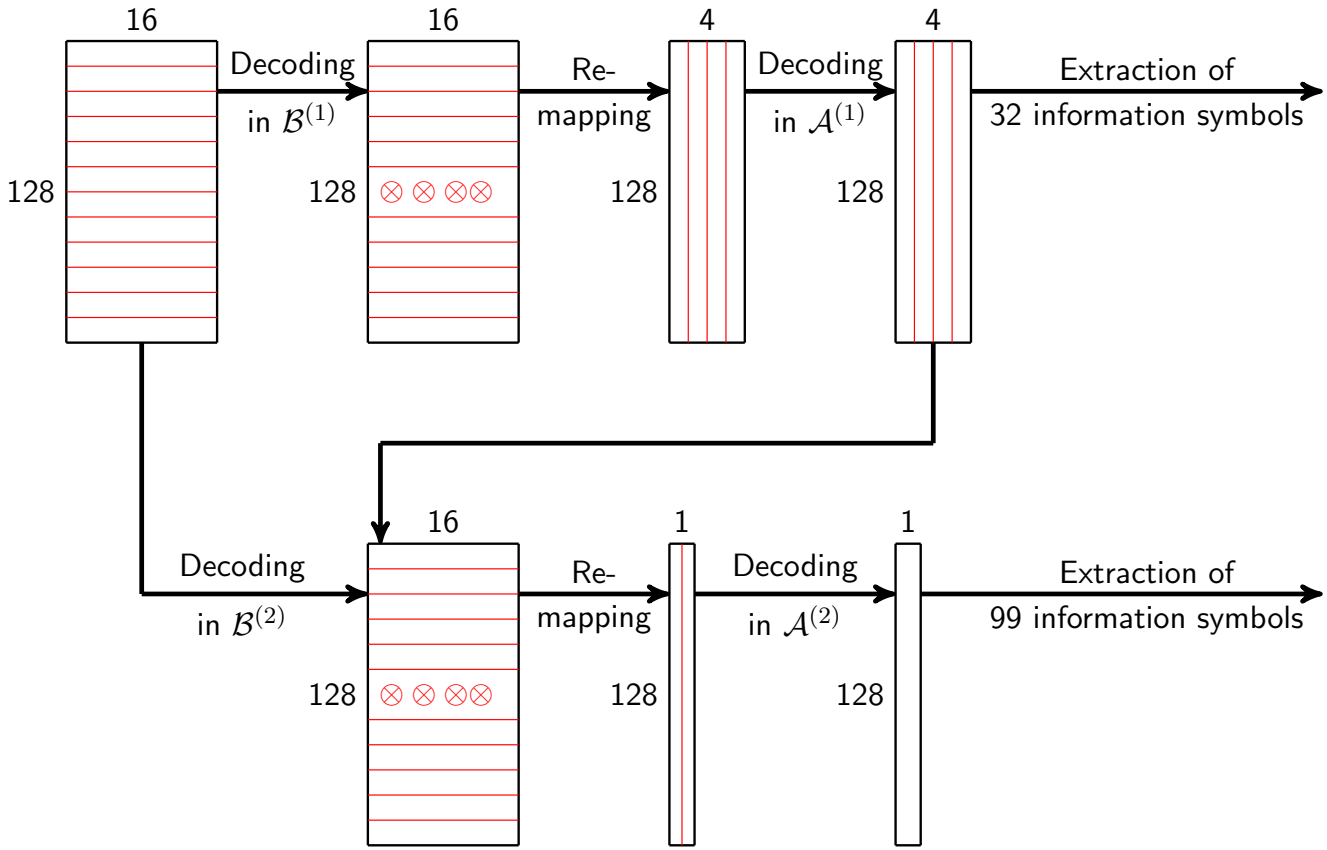
# Reed-Muller Example Code Construction

## Decoding:



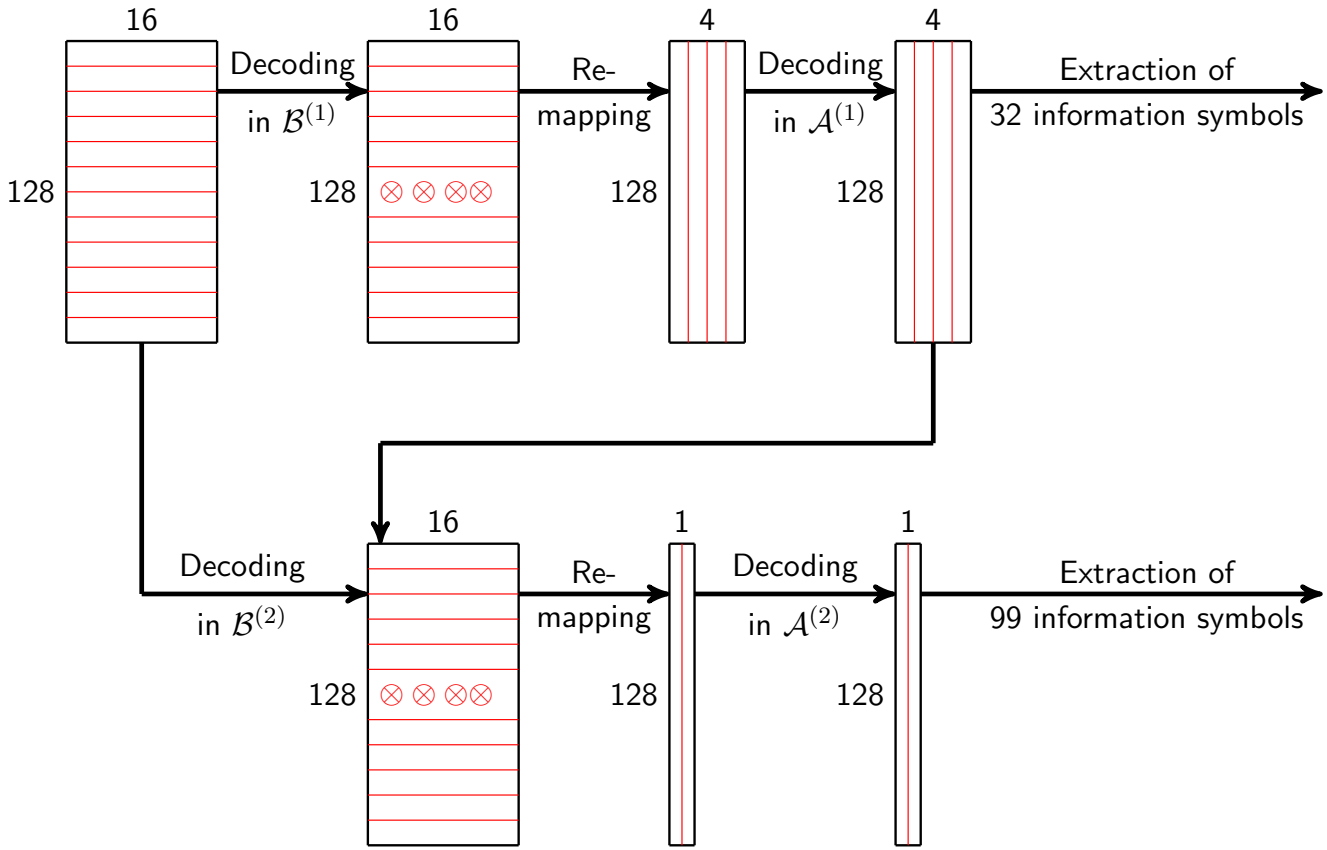
# Reed-Muller Example Code Construction

## 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

# Reed-Muller Example: Analysis



**Upper bound on $\mathbf{P_{err}}$:**

- Event $S_1$: Decoding fails in step 1.
- Event $S_2$: Decoding fails in step 2.
- $\mathrm{P}_{\mathrm{err}} = \mathrm{P}(S_1 \cup S_2) \leq \mathrm{P}(S_1) + \mathrm{P}(S_2)$

**Transform BSC to binary error and erasure channel**

- $\mathrm{P}(\mathrm{error}) = 0.020698$
- $\mathrm{P}(\mathrm{erasure}) = 0.155532$

# Reed-Muller Example: Analysis



**The error-erasure decoder of the outer $(128, 8, 64)$ code can decode correctly if $2\tau + \delta < 64$**

- $\mathrm{P}(S_1) = \mathrm{P}(2\tau + \delta \geq 64)$
  $= \sum_{i=0}^{128} \mathrm{P}(\delta = i)\mathrm{P}(2\tau \geq 64 - i | \delta = i)$
  $\approx 9.51 \cdot 10^{-12}$

**Calculate $\mathrm{S_2}$ similarly**

- $\mathrm{P}(S_2) \approx 1.48 \cdot 10^{-9}$

**Together:**

- $\mathrm{P_{err}} \leq 9.51 \cdot 10^{-12} + 1.48 \cdot 10^{-9} \approx 1.49 \cdot 10^{-9}$

**Using GMD decoding:**

- $\mathrm{P_{err}} \approx 5.37 \cdot 10^{-10}$ (remember goal: $\mathrm{P_{err}} < 10^{-9}$)

# Reed-Muller Example: Summary



**How good is this code construction?**

| Code | $P_{\text{err}}$ | Length | Largest Field |
|---|---|---|---|
| BCH Rep. | $10^{-9}$ | 2226 | $\mathbb{F}_{2^8}$ (BCH) |
| GC RM | $5.37 \cdot 10^{-10}$ | 2048 | $\mathbb{F}_2$ |

**Can we do better?**

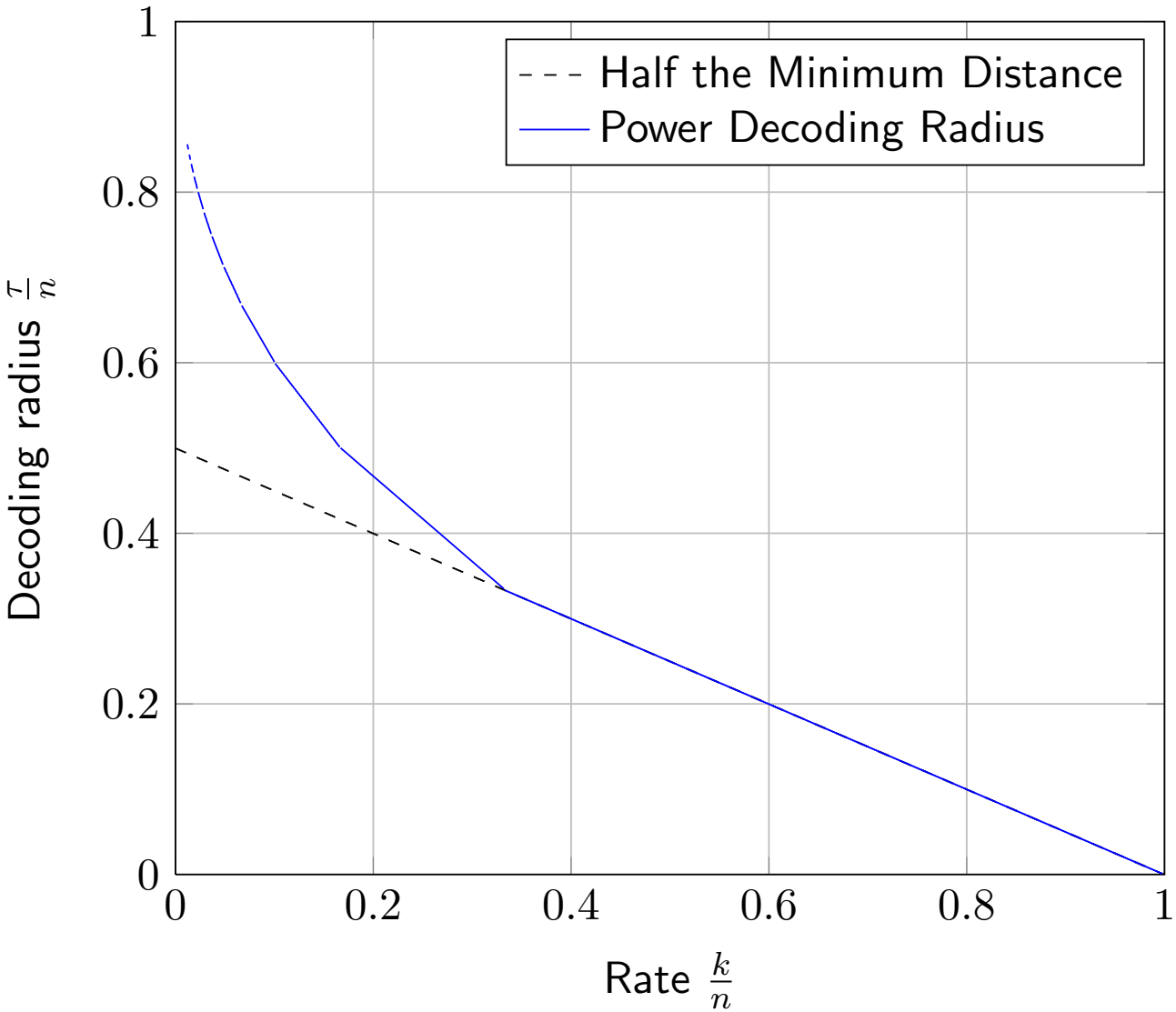
# Reed-Solomon Codes



## Evaluation Codes

- $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_q$, all nonzero and $\alpha_i \neq \alpha_j$
- $\mathbf{c} = \{(C(\alpha_1), \ldots, C(\alpha_n)) \in \mathbb{F}_q^n | deg(C) < k\}$
- Maximum Distance Separable
- More flexible than Reed-Muller codes

# Reed-Solomon Codes: Power Decoding



# Reed-Solomon Example Code Constructions



**Ordinarily Concatenated Scheme 1**

- Inner Code: $\mathcal{RM}(1,5) = \mathcal{C}(32,6,16)$
- Outer Code: $\mathcal{RS}(2^6; 64, 22)$
- $n = 64$ for outer code $\Rightarrow n = 64 \cdot 32 = 2048$
- $\mathrm{P}_{\mathrm{err}} \approx 6.79 \cdot 10^{-37}$

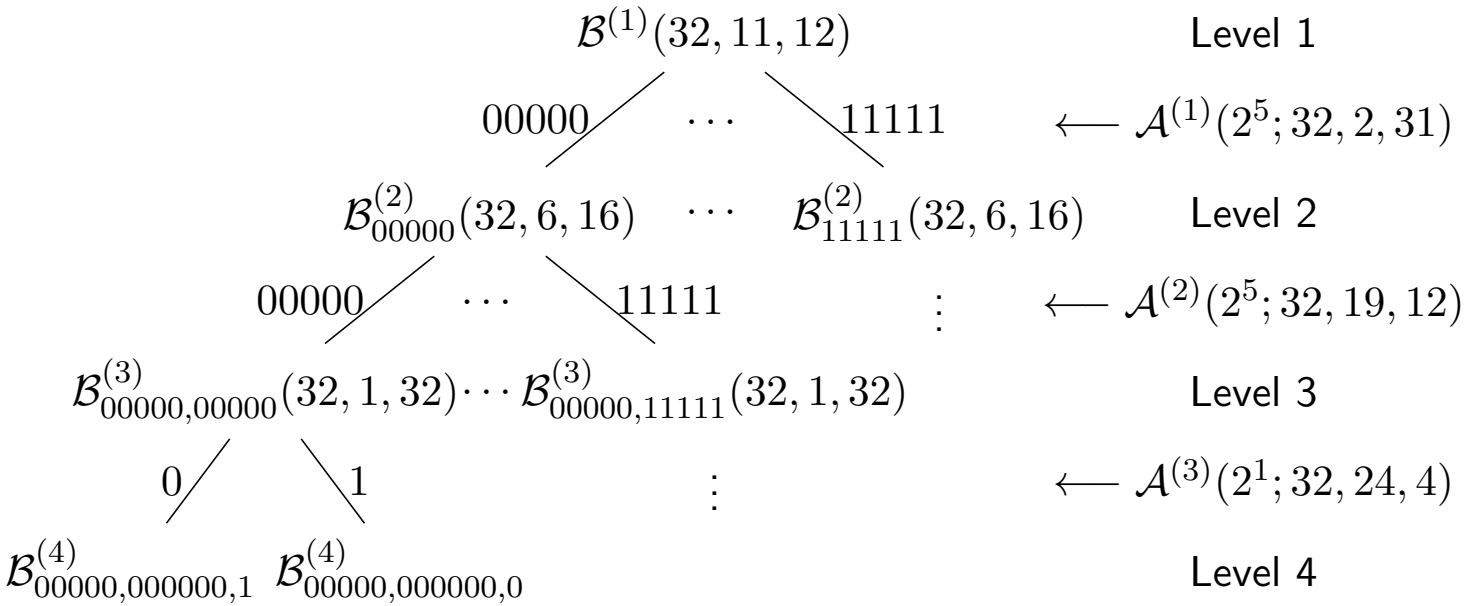
**Ordinarily Concatenated Scheme 2**

- Inner Code: $\mathcal{RM}(1,5) = \mathcal{C}(32,6,16)$
- Outer Code: $\mathcal{RS}(2^6; 36, 22)$
- Code length can be reduced to $n = 1152$
- $\mathrm{P}_{\mathrm{err}} \approx 1.19 \cdot 10^{-10}$

# Reed-Solomon Example Code Constructions



**Partitioning:**




**Encoding and Decoding:** Similar to Reed-Muller Example

# Reed-Solomon Example: Analysis



**Step 1:**

- ML decoding inner code $\Rightarrow$ Binary error and erasure channel with $\mathrm{P}(\mathrm{error}) = 0.037808$ and $\mathrm{P}(\mathrm{erasure}) = 0.174488$
- Decoding outer code $\Rightarrow \mathrm{P}(S_1) \approx 1.03 \cdot 10^{-8}$
- Power Decoding $\Rightarrow \mathrm{P}(S_1) \approx 1.48 \cdot 10^{-11}$

# Reed-Solomon Example: Analysis



**Step 1:**

- ML decoding inner code $\Rightarrow$ Binary error and erasure channel with $\mathrm{P}(\mathrm{error}) = 0.037808$ and $\mathrm{P}(\mathrm{erasure}) = 0.174488$
- Decoding outer code $\Rightarrow \mathrm{P}(S_1) \approx 1.03 \cdot 10^{-8}$
- Power Decoding $\Rightarrow \mathrm{P}(S_1) \approx 1.48 \cdot 10^{-11}$

**Step 2:**

- Binary error and erasure channel with $\mathrm{P}(\mathrm{error}) = 0.0032167$ and $\mathrm{P}(\mathrm{erasure}) = 0.0175397$
- $\mathrm{P}(S_2) \approx 3.11 \cdot 10^{-10}$

# Reed-Solomon Example: Analysis



**Step 3:**

- $\mathrm{P}(S_3) \approx 2.13 \cdot 10^{-11}$

**Overall block error probability:**

- $\mathrm{P}_{\mathrm{err}} \leq \mathrm{P}(S_1) + \mathrm{P}(S_2) + \mathrm{P}(S_3) \approx 3.47 \cdot 10^{-10}$

**Advantage of this construction:**

- Code length reduced to $n = 32 \cdot 32 = 1024$

# Conclusion



**How good are our code constructions?**

| Code | $P_{err}$ | Length | Largest Field |
|---|---|---|---|
| BCH Rep. | $10^{-9}$ | 2226 | $\mathbb{F}_{2^8}$ (BCH) |
| GC RM | $5.37 \cdot 10^{-10}$ | 2048 | $\mathbb{F}_2$ |
| RS I | $6.79 \cdot 10^{-37}$ | 2048 | $\mathbb{F}_{2^6}$ |
| RS II | $1.19 \cdot 10^{-10}$ | 1152 | $\mathbb{F}_{2^6}$ |
| GC RS | $3.47 \cdot 10^{-10}$ | 1024 | $\mathbb{F}_{2^5}$ |

# Code Concatenation



**Distinguish:**

- (Ordinary) Code Concatenation (CC)
- Generalized Code Concatenation (GCC)

**Advantages of (Generalized) Code Concatenation:**

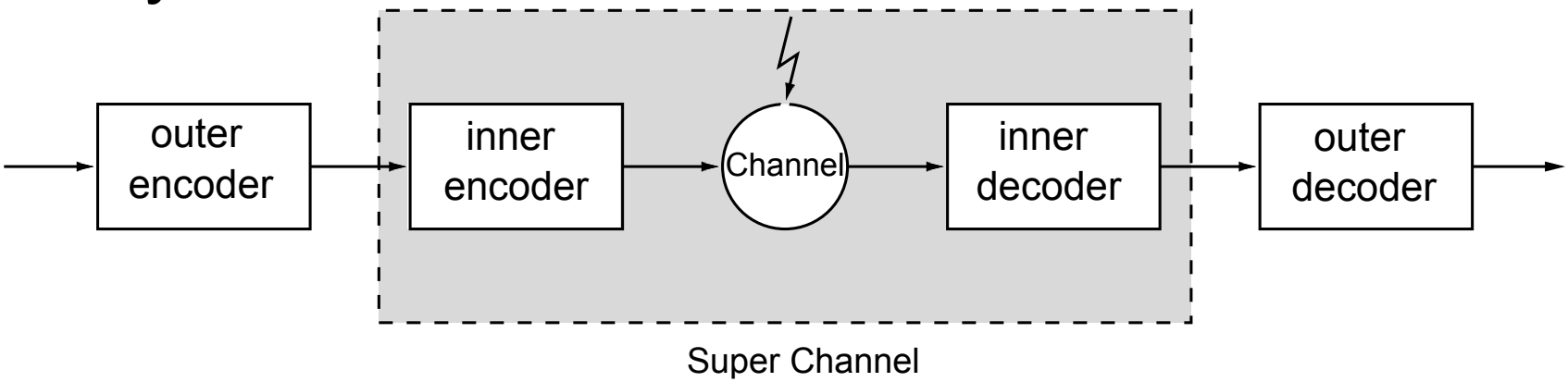
- Very long codes can be constructed by concatenation based on short codes
- Smaller decoding complexity compared to arbitrary code of same length
- Concatenated codes can correct burst errors and independent single errors at the same time

**Disadvantage:**

- Soft decision decoding has not been satisfactorily solved

# (Ordinary) Code Concatenation

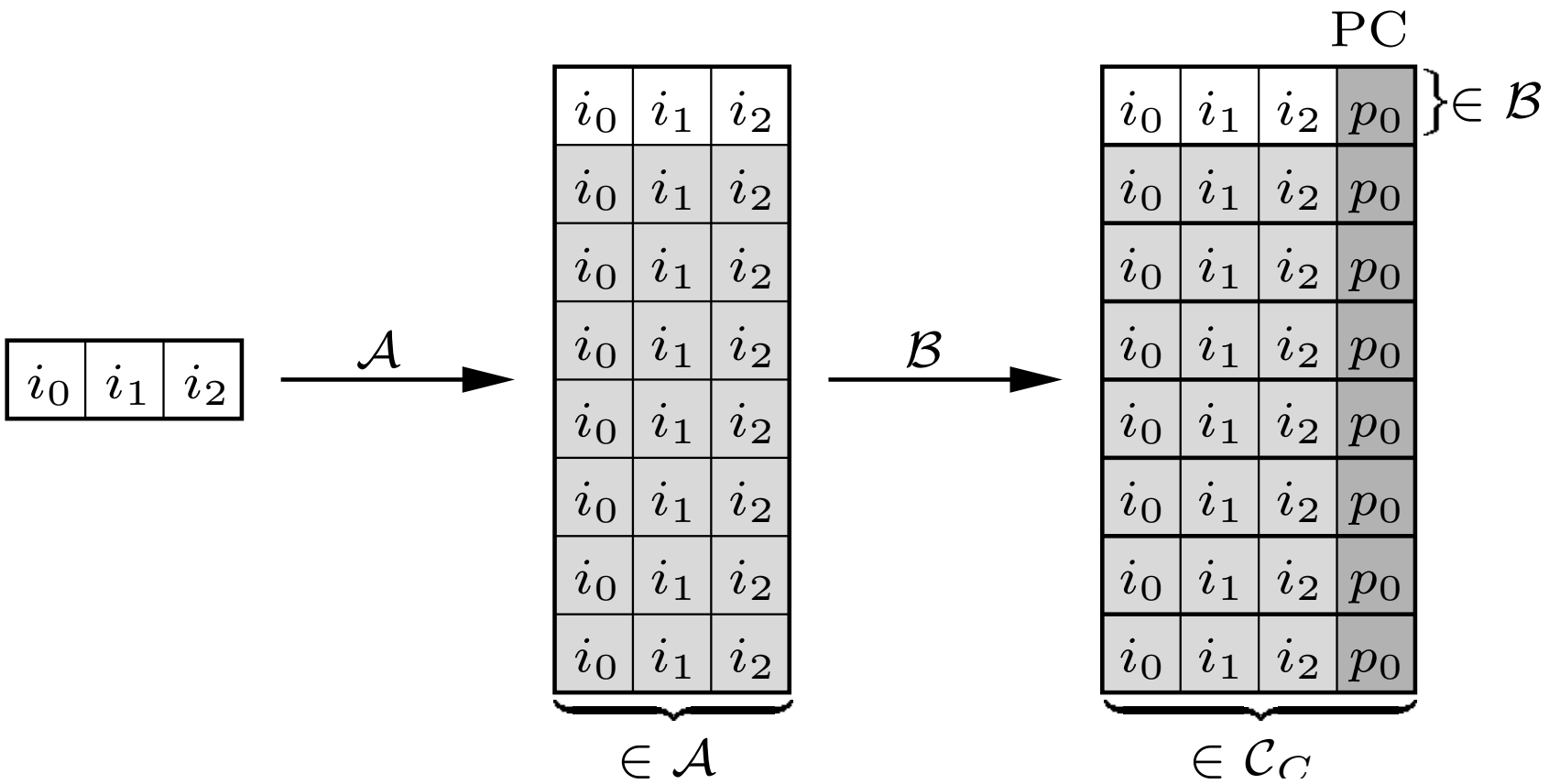
**Forney 1966**



# (Ordinary) Code Concatenation

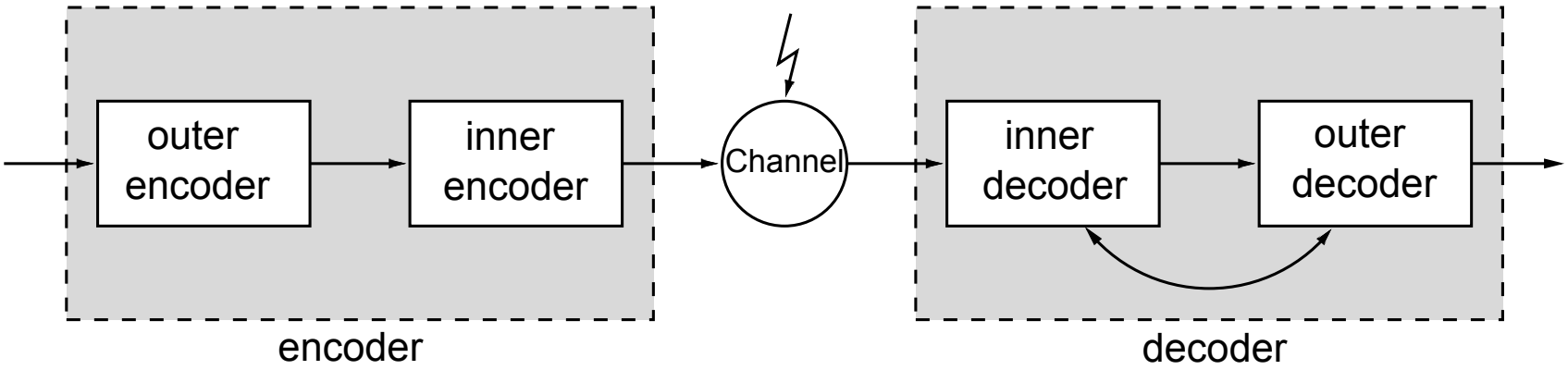
**Example:**

- Outer Code: $\mathcal{A}(2^3; 8, 1, 8)$
- Inner Code: $\mathcal{B}(2; 4, 3, 2)$
- Concatenated Code: $\mathcal{C}_C(32, 3, 2 \cdot 8 = 16)$



# Generalized Code Concatenation

**Blokh, Zyablov 1974:**



# Generalized Code Concatenation



**Further advantages:**

- Codes with unequal error protection can be constructed (UEP)
- More codewords (and hence a higher coderate) than an ordinary concatenated code with same parameters $n$ and $d$
- Larger minimum distance than an ordinary concatenated code with same number of codewords (or with same coderate)

# Generalized Code Concatenation



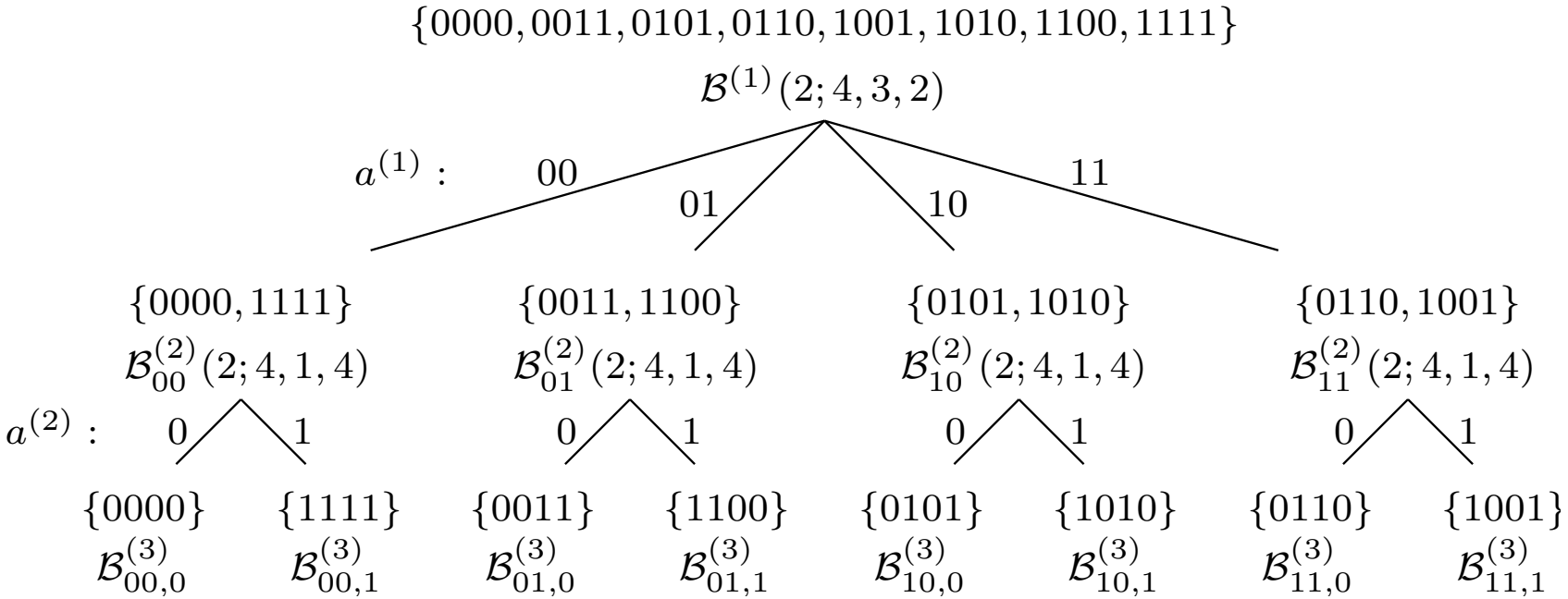
**Example:**

- Inner Code: $\mathcal{B}(2;4,3,2)$
- Outer Codes: $\mathcal{A}(2^2;8,1,8)$, $\mathcal{A}_H(2;8,4,4)$
- Generalized Concatenated Code: $\mathcal{C}_{GC}(2;32,6,16)$
  Note: same minimum distance than before, but $2^6 = 64$ codewords (before: $2^3 = 8$ codewords)

# Generalized Code Concatenation

NACHRICHTENTECHNIK Algebraic Coding

**Partitioning of inner code:**



# Generalized Code Concatenation



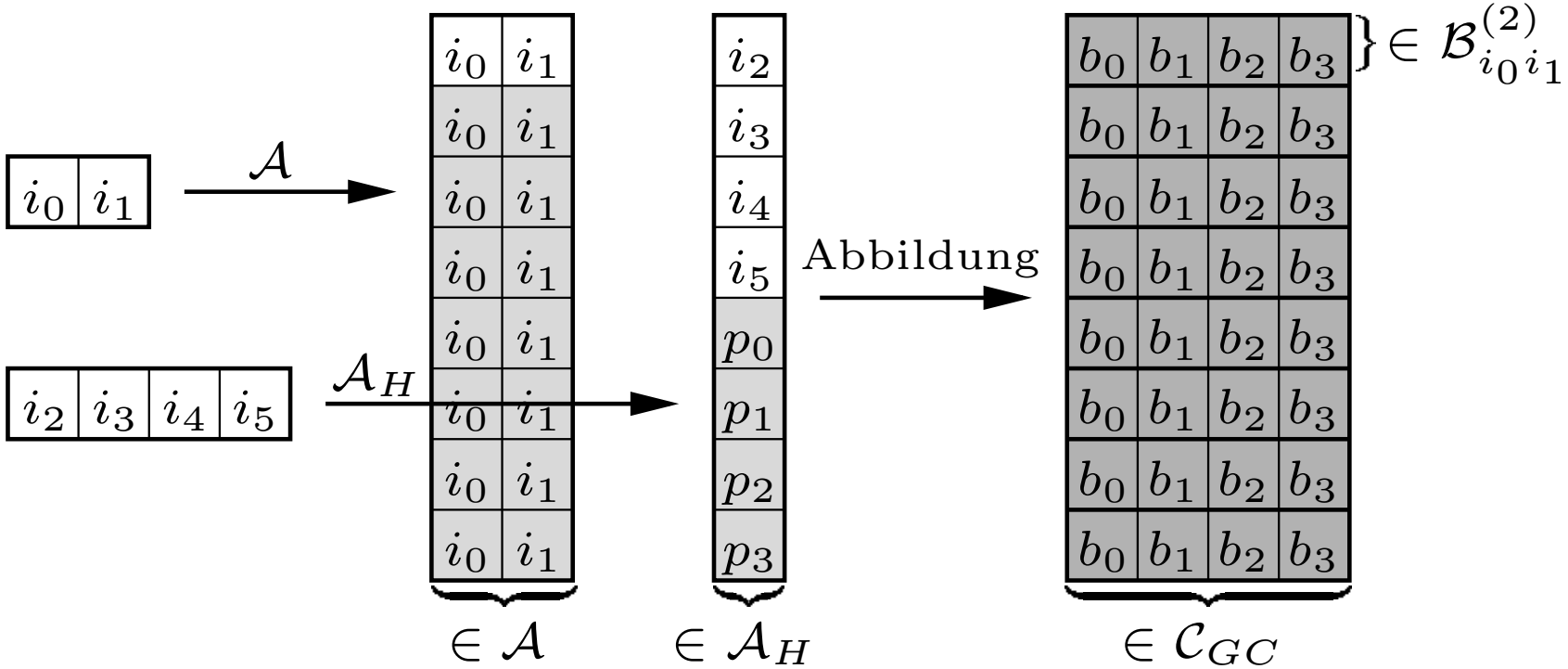
**Partitioning of inner code:**

- Partition code such that minimum distance in subcodes is as large as possible
- Each codeword can be identified uniquely by the numeration of the path in the partition tree
- Protect numeration with an outer code at each level

# Generalized Code Concatenation



**Encoding:**



# Generalized Code Concatenation



**Given:**

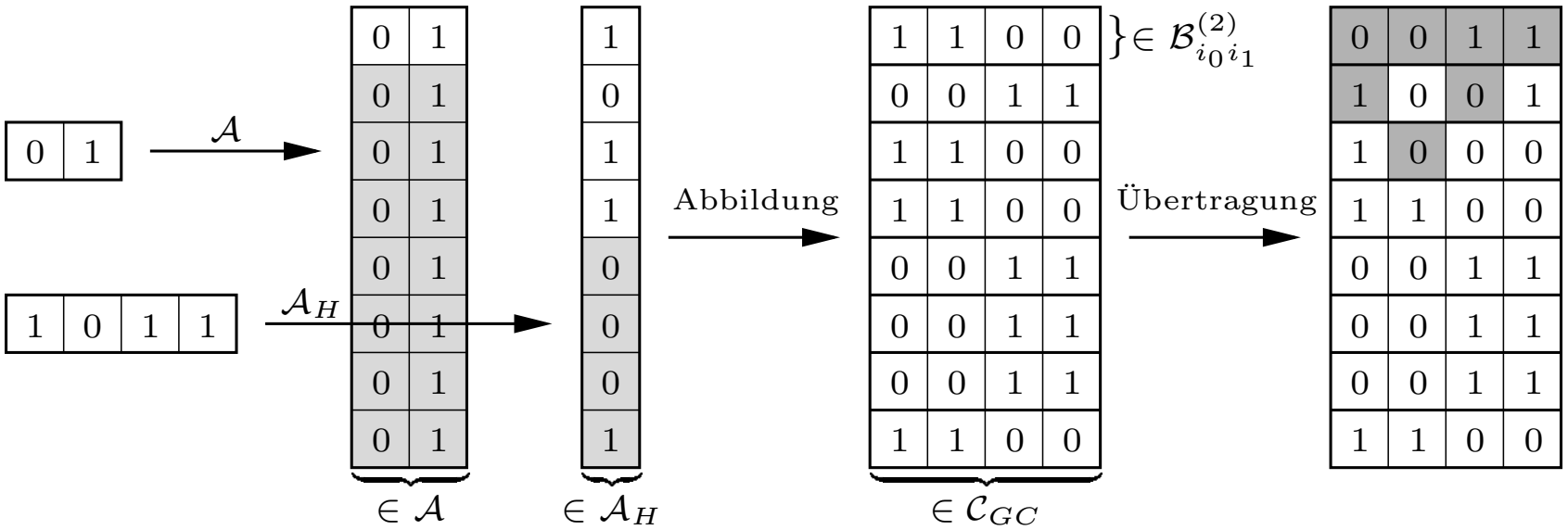
- Inner codes: $\mathcal{B}^{(\ell)}(p; n_i, k_i^{(\ell)}, d_i^{(\ell)})$
- Outer codes: $\mathcal{A}^{(\ell)}(p^m; n_o, k_o^{(\ell)}, d_o^{(\ell)})$, $\ell = 1, \ldots, l$

**Parameters of a GCC:**

- Length $n = n_i n_o$
- Dimension $k = \sum_{\ell=1}^{l} k_o^{(\ell)}$
- Minimum Distance $\mathbf{d} \geq \min_{\ell=\mathbf{1},\ldots,\mathbf{l}} \left\{ \mathbf{d_i^{(\ell)}} \cdot \mathbf{d_o^{(\ell)}} \right\}$

# Generalized Code Concatenation

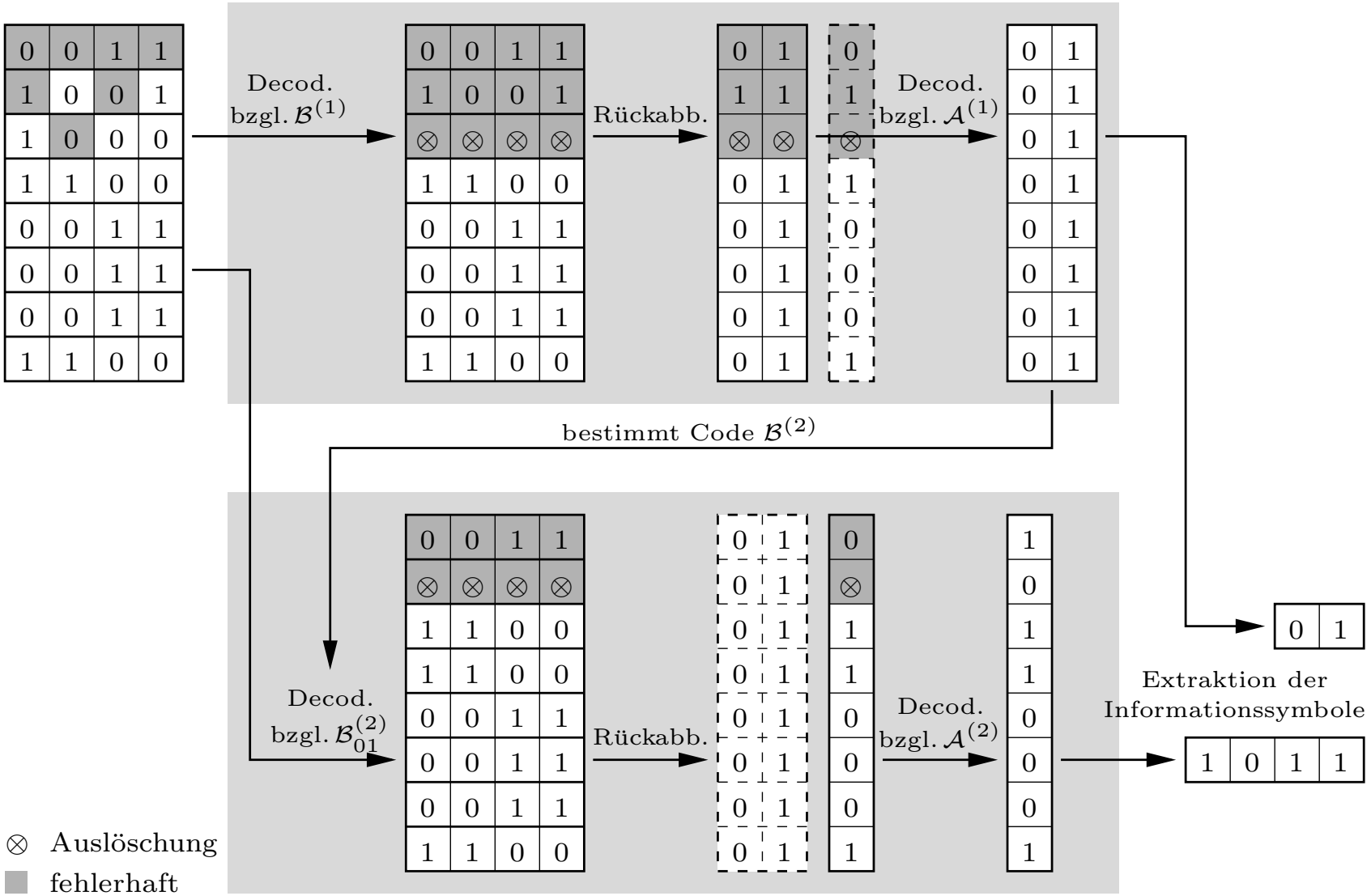
**Transmission:**



# Generalized Code Concatenation



## Decoding:



# Literature

**Generalized Code Concatenation**

- Martin Bossert, Channel Coding for Telecommunications, John Wiley & Sons, 1999
- Martin Bossert, Kanalkodierung, 3. Auflage, Oldenbourg Wissenschaftsverlag, 2013

**Physical Unclonable Functions**

- Roel Maes, Physically Unclonable Functions: Constructions, Properties and Applications (Dissertation), 2012

**Example Code Constructions presented in this lecture**

- S. Müelich, S. Puchinger, M. Bossert, M. Hiller, G. Sigl Error Correction for Physical Unclonable Functions Using Generalized Concatenated Codes, Svetlogorsk 2014
- S. Müelich, S. Puchinger, M. Bossert, M. Hiller, G. Sigl On Error Correction for Physical Unclonable Functions, Hamburg 2015