Drahtlose Breitbandkommunikation - Kanalcodierung

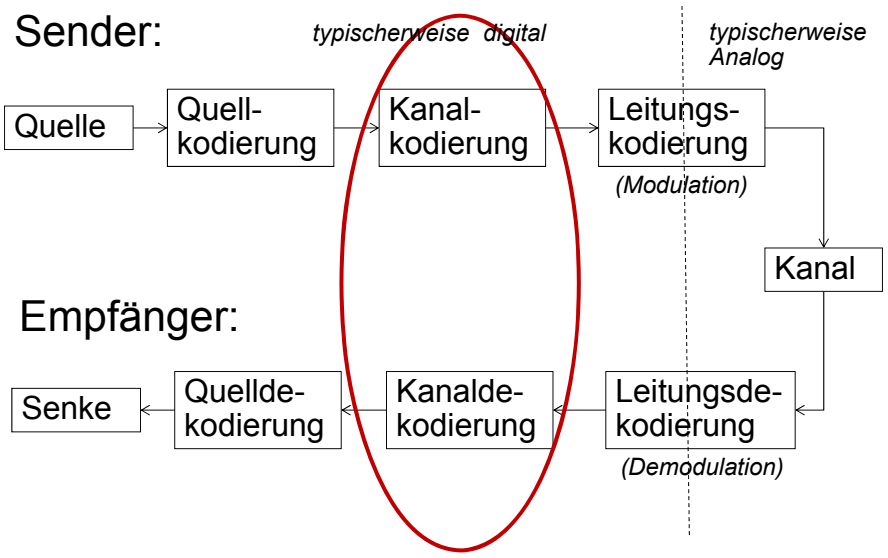
Humboldt-Universität zu Berlin, Institut für Informatik,
IHP, Leibniz Institut für innovative Mikroelektronik, Frankfurt (Oder)

Vorlesung Drahtlose Breitbandkommunikationssysteme

Prof. Dr. Eckhard Grass grass@ihp-microelectronics.com
grass@informatik.hu-berlin.de

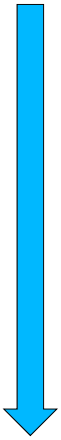
http://www.informatik.hu-berlin.de/~grass/bbk

Breitbandkommunikation (PHY-Modell)



Kanalcodierung

- Einfügen von Redundanz vor der eigentlichen Datenübertragung im Sender
- Übertragung über gestörten Kanal (Bitfehler)
- Fehlerkorrektur und Entfernen der Redundanz im Empfänger



Kanalkodierung

Fehlerkorrekturcodes (Forward Error Correction)

Blockcodes

- **Systematische Blockcodes**
  - Reed Solomon Code (RS)
  - Low Density Parity Check Codes (LDPC)
- **Nicht-Systematische Blockcodes**

Faltungscodes (nicht Syst.)

- ***Viterbi Codes***
- Ungerboeck Codes

Concatenated Codes

Verkettung mehrerer Codes

- z.B. Viterbi + Reed Solomon
- Turbo Codes, e.g.: Serial Concatenated Convolutional Codes

Kanalcodierung

Systematischer Blockcode:

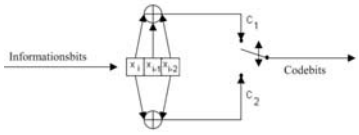
- Die ursprünglichen Informationsbits sind noch in dem Datenstrom enthalten

Nicht-Systematische Codes:

- Die ursprünglichen Informationsbits sind nicht im Datenstrom enthalten sondern wurden z.B. mit Faltungscodierer modifiziert

Faltungskodierer

- Bietet Vorwärtsfehlerkorrektur
- In Praxis fast ausschließlich nicht-systematische Codes
- Rekursiv (mit Rückkopplung) und nicht-rekursiv möglich
- Informationsgehalt wird über mehrere Nutzdatenstellen „verschmiert“
- Entspricht der Faltung in der digitalen Signalverarbeitung
- Leistungsfähiger als Blockcodes
- Kann mit Punktierung kombiniert werden um beliebige Code-Rate zu erreichen

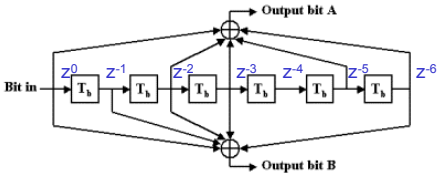


Constraint length (K) = 3

$c_1 = x_i + x_{i-1} + x_{i-2}$

$c_2 = x_i + x_{i-2}$

Convolutional Code (7_8, 5_8)



Constraint length (K) = 7, R=1/2

Out_A = g1(z) = $1+z^{-2}+z^{-3}+z^{-5}+z^{-6}$

Out_B = g2(z) = $1+z^{-1}+z^{-2}+z^{-3}+z^{-6}$

Convolutional Code (133_8,171_8)

Viterbi Decoding Algorithm

1. Beginne Trellis im Nullzustand zum Zeitpunkt t = 0
2. Berechne Hammingdistanz (für Soft Decision Viterbi: Euklidische Distanz) zwischen empfangenen Codewort y(t) und allen möglichen Codeworten a(t)
3. Addiere unter 2) berechnete Pfadmetriken zu alten Zustandsmetriken *Mj(t−1), j = 0. . .2m−1* **Add** **Compare**
4. An jedem Zustand Auswahl desjenigen Pfades mit kleinster Hammingdistanz (euklidischer Distanz) (größter Korrelationsmetrik) und Verwerfung (keine weitere Berücksichtigung) der anderen Pfade **Select**

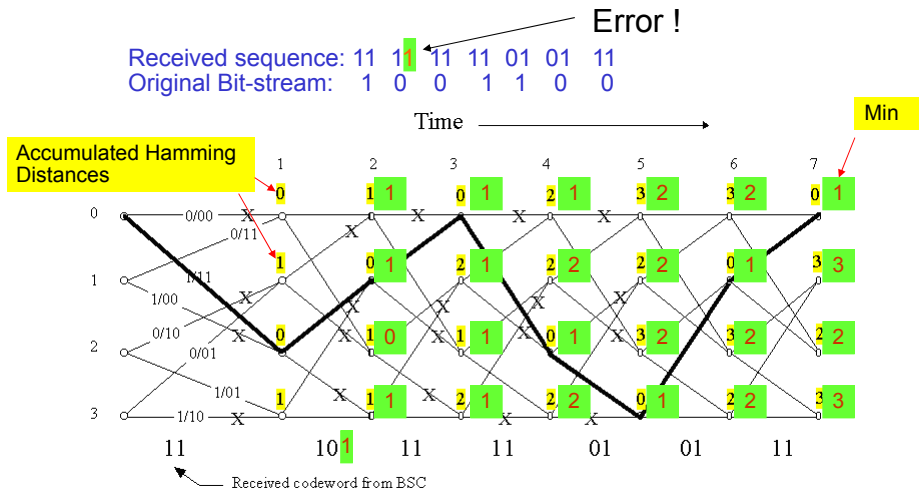
   ⇒Aufwand wächst **nur linear mit Pfadlänge (nicht exponentiell)**
5. Wiederholung ab 2), bis alle *N empfangenen Worte abgearbeitet wurden*
6. Ende des Trellisdiagramms:
   - Terminierte Codes (Trellis endet im Nullzustand):

     ⇒Bestimmung des Pfades mit der besten Metrik *M0(N) im Nullzustand*
   - Nicht-terminierte Codes:

     ⇒Bestimmung des Pfades mit der global besten Zustandsmetrik *Mj(N), j = 0⋯2m−1*
7. Zurückverfolgen des in 6) bestimmten Pfades (*Survivor) und Ausgabe der zugehörigen Informationsbit*

Nach Volker Kühn & Dirk Wübben; Vorlesungsskript Kanalcodierung I, Uni Bremen, 2011

Viterbi Decoder - Principle Operation



VHDL 2: Viterbi - Add Compare Select

```
ACS_comb: process (DISTANCE_1, HAM_1, DISTANCE_2,  HAM_2)
variable new_h1, new_h2 : integer;
begin
new_h1 :=  HAM_1 + DISTANCE_1;     -- DISTANCE_1 & 2 are the old ones
new_h2 :=  HAM_2 + DISTANCE_2;

   if new_h1 < new_h2 then
     NEW_DISTANCE <= new_h1;       -- Updated distance for 'this_state'
     back_state <= PRED_STATE_1;
     TRANS <= '1';                 -- indication for path into -"-
   else
     NEW_DISTANCE <= new_h2;       -- Updated distance for 'this_state'
     back_state <= PRED_STATE_2;
     TRANS <= '0';                 -- indication for path into -"-
   end if;
end process ACS_comb;
```


Viterbi Decoding Algorithm – General Remarks

- Traceback length L_T should be L >= 5 x K
- Use of K tailbits to bring the decoder into the zero state
- This improves the coding performance at the end of the frame since it is known that the decoder needs to end in state zero
- The next frame (or symbol) can be decoded right away since the decoder has returned to the initial state
- The code rate for small packets is slightly deteriorated since the tail bits have to be added to the user data (as overhead)
- Convolutional Codes and the Viterbi Decoder perform poorly if burst errors are in the data stream
  - Use of an interleaver distributes burst errors in data stream
  - Combination with (outer) block code (RS) can significantly reduce the effect of burst errors

Soft Decision Demapper and Viterbi Decoder

… at White Board:

Blockcodes

Establish checksums over the data.
If checksums are correct:
- no errors in block
  -> use of x_{ij}

If checksums are incorrect:
- Identify location(s) of erroneous bits
  -> Flip those bits and re-calculate checksums

… the naive approach:

x_{11}	x_{12}	…	x_{1n}	CS-R1
x_{21}	**x_{22}**	…	x_{2n}	**<- CS-R2**
…	…	…	…	
x_{m1}	x_{m2}	…	x_{mn}	CS-Rm
CS-C1	**CS-C2**		CS-Cn	

CS-Rm = check-sum row “m”
CS-Cn = check-sum column “n”

Reed Solomon Codes

- In 1960, Irving Reed and Gus Solomon published a paper in the Journal of the Society for Industrial and Applied Mathematics describing a new class of error-correcting codes that are now called Reed-Solomon (R-S) codes.
- These codes have great power and utility, and are today found in many applications from CD-players to safety critical applications.
- Reed Solomon Codes are systematic codes i.e. the original bits are part of the coded bits
- The Reed-Solomon (R-S) codes are particularly powerful in a burst-error scenario

Source: Bernard Sklar

Properties of Reed Solomon Codes

- A Reed-Solomon code is a block code which can be specified as RS(n, k), with m-bit symbols

  n – the total number of symbols ;
  k – the number of information (data) symbols ;

  The difference n-k (usually called 2t) is the number of parity symbols appended to data symbols to make an encoded block (codeword) ;

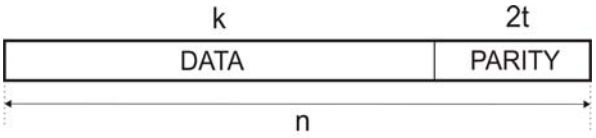


Fig. 2 The structure of a RS codeword

- The maximum number of symbol errors in a block which can be corrected by RS decoding algorithm is t = (n-k)/2
  Each symbol error may contain any number of bit errors

Source: M. Marinkovic -> ???

Error Correction with Reed Solomon Codes

- RS codes are well suited for correcting burst errors
  An example : RS(255,239) code, very often used in communications
  Error correction capability : (255-239)/2 = 8 symbols

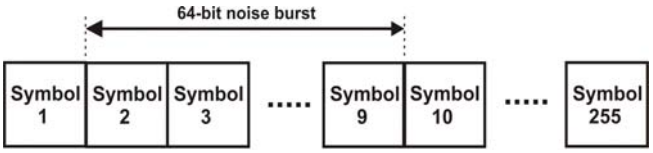


Fig.3 A codeword RS(255,239) distrubed by 64-bit noise burst

- Burst of noise that lasts for a duration of 64 consecutive bits corrupts exactly 8 symbols
  The RS decoder will correct these errors since it replaces the incorrect symbols with the correct ones

Source: M. Marinkovic -> ???

Reed-Solomon Encoder Circuit

- **Task of RS Encoder**
  - Calculation of parity symbols which shall be appended to data symbols

- **Main components of RS Encoder**
  - Multipliers and Adders over Galois Field (GF)
  - Registers (flip-flops)

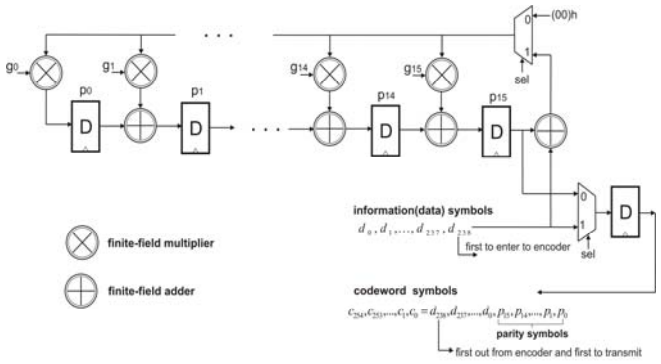


Fig. 5 RS(255,239) Encoder Architecture

Source: M. Marinkovic -> ???

Reed-Solomon Decoder

- Task of RS Decoder
  Detection and correction of errors in a RS block (codeword)

- Decoding process is far harder task than encoding process
  Decoder architecture is more complex then encoder architecture

- Decoding process includes several steps :

  1. Calculating the *syndrome values* from the received codeword;
  2. Computing the *error locator* and *error evaluator* polynomial (Solving *Key Equation*);
  3. Finding the *error locations* (*Chien search* algorithm);
  4. Computing *error magnitudes* (*Forney's formula*);
  5. Adding *error magnitudes* to corrupted symbols in order to obtain corrected symbols.

Source: M. Marinkovic -> ???

Reed-Solomon Decoder Circuit

A RS decoder circuit consists of several major components according to the steps of the decoding algorithm

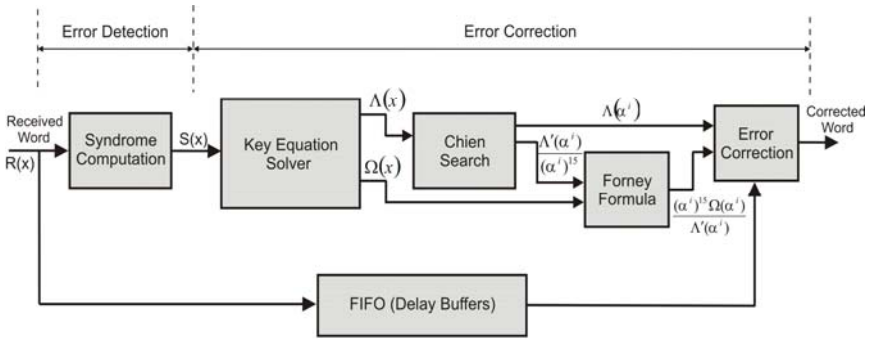


Fig. 6 Block diagram of RS Decoder

Source: M. Marinkovic -> ???

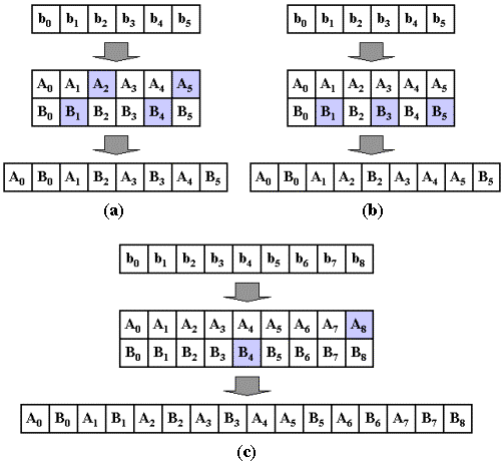
Kanalcodierung

Punktierung :

- Bei Faltungscodes lässt sich durch eine sogenannte Punktierung des Codewortes gezielt eine bestimmte Coderate R_c wählen.
- Bei der Punktierung werden bestimmte Bitpositionen des Codewortes weggelassen("punktiert") und dadurch die Coderate erhöht.
- Der Decoder muss dieses sogenannte Punktierungsschema kennen und bei der Decodierung berücksichtigen. An den Positionen punktierter Bits werden zufällige Werte (bei Softdecision Decodern „unentschieden“) eingetragen
- Punktierung kann auch genutzt werden, die Codewortlängen genau auf eine bestimmte Rahmenlänge für die nachfolgende Datenübertragung bzw. Datenspeicherungen anzupassen.
- Durch Punktierung kommt es allerdings auch zu einer Reduktion der Korrekturleistung und damit ggf. zur Erhöhung der Bitfehlerrate (BER).

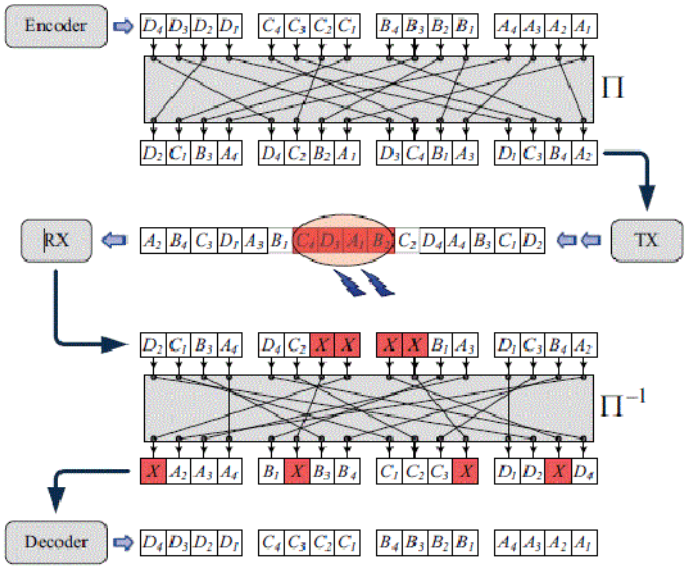
Kanalcodierung - Punktierung

Punktierung (Bit-Steeling-Algorithm) nach IEEE802.11a



Bit-stealing” algorithm for code rate conversions:
a) 1/2 to 3/4; b) 1/2 to 2/3; c) 1/2 to 9/16.

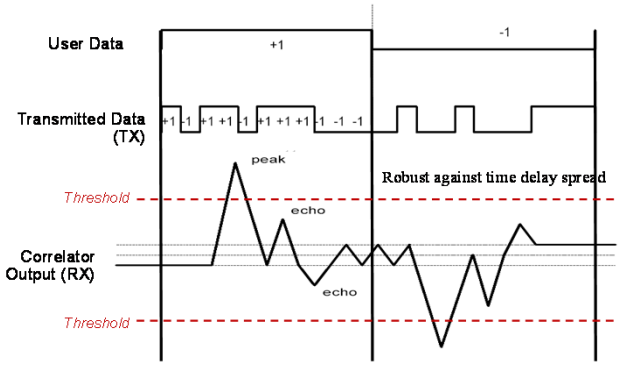
Interleaving



Aus: Rizwan Asghar, Ph.D., Uni Linköping, 2010

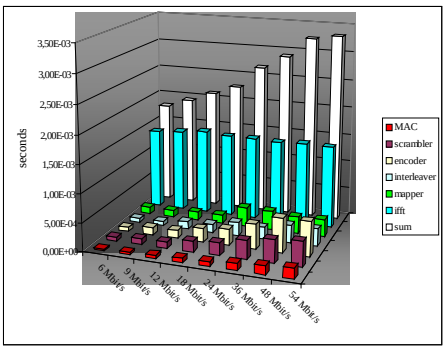
Inherent Error Correction (FEC) in DSSS Systems

- Direct Sequence Spread Spectrum (DSSS) systems have an inherent error correction capability
- Spreading introduces redundancy in transmitted data
- In some cases, no additional FEC is needed
- In some applications, a combination with Reed-Solomon Code can be beneficial to reduce burst errors

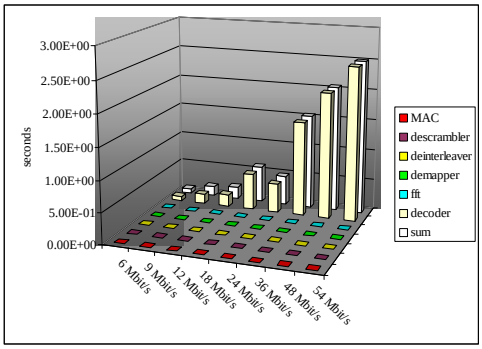


Processing Requirements IEEE802.11a TX & RX

Processing demands per OFDM Symbol for C-model in transmit mode (a) and receive mode (b)



a)



b)