Zabezpečení proti chybám při přenosu digitální televize a rozhlasu, konvoluční a blokové kódy, více úrovňové zabezpečení

Ing. Karel Ulovec, Ph.D.

ČVUT, Fakulta elektrotechnická

xulovec@fel.cvut.cz

Zabezpečení proti chybám při přenosu digitální televize a rozhlasu, konvoluční a blokové kódy, více úrovňové zabezpečení

Obecné principy protichybového zabezpečení, kódový poměr, systematický kód
Lineární blokové binární kódování
Blokové CRC (Cyclic Redundancy Check) kódování, dekódování; zkrácený kód CRC
Blokové LDPC (Low Density Parity Check) kódování, dekódování
Konvoluční kódování, dekódování; punkturování v konvolučním kódování
Více úrovňové zabezpečení

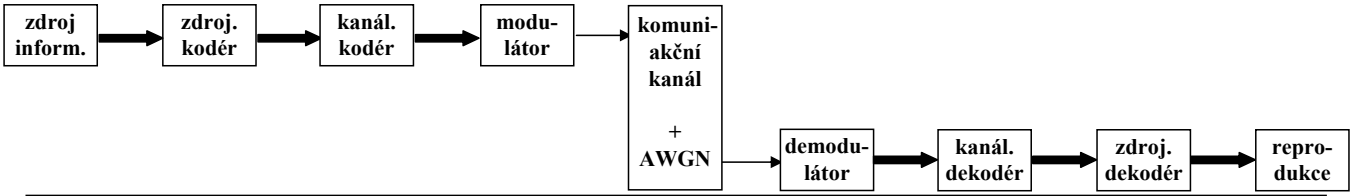
Rozšířená témata pro přípravu studentů ke zkoušce

Příloha (nepovinné) – Viterbiho algoritmus pro dekódování konvolučního kódu

Kanálové zpracování

Kanálové zpracování signálu
- Znáhodnění datového toku
- Ochrana před vznikem chyb při přenosu
  - Prokládání – rozptýlení shluku chyb, rovnoměrné rozprostření informace do časově-kmitočtového prostoru
  - Protichybové kódování (ochranné kanálové kódování) – zavedení úmyslné redundance pro možnost opravy (korekční) či alespoň detekce (detekční) chyb; samoopravné kódy (FEC, Forward Error Correction) – technika umožňující opravy chyb bez potřeby zpětného kanálu
  - Využívání diverzity – paralelní přenosové kanály (např. na různých kmitočtech, nebo systémy s více anténami)
  - Opakování přenosu – vyžaduje zpětný kanál (ARQ, Automatic Repeat Request)
  - Hybridní ARQ (HARQ, Hybrid ARQ) – kombinuje FEC a ARQ (je výhodné využít i částečně porušená data a kombinovat je s daty opakovaně přenášenými, při opakovaném přenosu je možno data více zabezpečit)
  - Adaptivní modulace/kódování – podle stavu kanálu; nutno provádět měření a předávání „naměřených hodnot"

BER (Bit Error Ratio)
- Počet chybně přenesených bitů / počet celkově přenesených bitů za jednotku času
  - Prokládání Např. $2 \cdot 10^{-4}$ odpovídá 2 chybně přeneseným bitům z celkově 10000 přenesených bitů
- Kvalitativní parametr – vyjadřuje kvalitu systému při přenosu informace, snažíme se o minimalizaci



Protichybové zabezpečení

Obecné principy protichybového kódování
- Přenášené informaci **u** (k bitů) je přiřazeno zakódované slovo, kód **c** (n bitů), $n > k$
  - Označujeme kód (n, k) nebo (k, n)
  - Informační prostor = množina informačních slov; 2^k kombinací hodnot bitů (všechny kombinace použity)
  - Prostor kódů = množina všech možných kódových slov **c** použitých pro přenos (platné kódové slovo); rovněž počet 2^k
  - Kódové slovo má n bitů – počet 2^n možných kombinací hodnot bitů, ale pouze 2^k kombinací odpovídá platným kódům **... navýšení redundance v prostoru kódů** ($2^n > 2^k$)
  - Hammingova vzdálenost … počet odlišných bitů mezi dvěma kódovými slovy
  - Minimální Hammingova vzdálenost d_h v prostoru kódů … nejmenší počet odlišných bitů u veškerých dvojic různých (platných) kódů **c** v kódovém prostoru – při návrhu algoritmu kódování se snažíme o maximální d_h => nejlepší vzájemná odlišnost kódových slov
  - Indikace chyb – pokud je přijaté slovo nepřípustnou kombinací bitů, neodpovídá platnému kódu
  - Oprava chyb – navýšená redundance lze využít k odhalení chybného bitu (výpočet tzv. syndromu) nebo je vybráno nejpodobnější (nejvíce pravděpodobné) platné kódové slovo … FEC (forward error correction) – dopředná korekce (nevyužívá zpětný kanál)
  - Indikace/oprava – pouze do jistého množství chyb (existuje hranice schopnosti indikace/opravy)
  - **Kódový poměr** CR (code rate) = k/n

Principy protichybového kódování

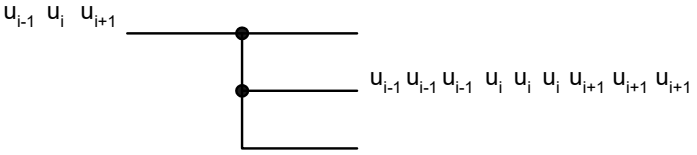
Obecné principy protichybového kódování

- **Systematický kód**
  - Pokud **c** vytvoříme přidáním $m = n - k$ zabezpečovacích (redundantních) bitů
  - Informační obsah je součástí výsledného kódu (přijímač může zpracovat informaci i bez dekodérů; umožňuje nepovinné, volitelné užití ochranného dekódování – např. v případě špatných podmínkách příjmu)
- Prodloužení kódového slova (vs. informace) způsobí pokles užitečné přenosové rychlosti, užitečná přenosová rychlost je přímo úměrná hodnotě CR – se snižujícím CR se snižuje užitečná přenosová rychlost
  - Overhead = 1 – CR, se snižujícím CR narůstá overhead a zvyšuje se schopnost indikace/opravy chyb
- Důležitou charakteristikou je BER = f(SNR) – posuzujeme pokles chybovosti v systému po použití protichybového kódování (viz příklad dále)

Protichybové zabezpečení

Kódování opakování (3, 1)

- Velmi jednoduchý kód



- $n = 3$, $k = 1$, CR = 1/3, užitečná přenosová rychlost 3x menší, overhead 2/3
- Minimální $d_h = 3$ (2 kódová slova: 0 0 0 a 1 1 1)
- Detekuje až dvojnásobnou chybu, obecně lze detekovat chybu až řádu (počet bitů): $d_h - 1$
- Opraví jednoduchou chybu (do max. BER=1/3), obecně lze opravit chybu až řádu (počet bitů):

celé číslo nižší nebo rovno $\left\{\frac{d_h - 1}{2}\right\}$

Min. Hamm. vzdálenost d_h	2	3	4	5	6	7	8	…
Počet detekovatelných chyb	1	2	3	4	5	6	7	…
Počet opravitelných chyb	0	1	1	2	2	3	3	…

Protichybové zabezpečení

Lineární blokové binární kódování

- Princip kódování: Množina kódů (počet 2^k) vzniká algebraickými operacemi, speciálně lineárními funkcemi, nad informačními bity, resp. k bitovými slovy **u** (blokové)
  - Existuje báze generující kód … matice **G** (velikosti k,n) obsahující vektory **g** ve sloupcích, např. 5 vektorů: 10, 01, 10, 11, 01
  - Kódové slovo $\mathbf{c} = [\, c_0 \;\; c_1 \;\; \dots \;\; c_{n-1} \,] = [\, u_0 \;\; u_1 \;\; \dots \;\; u_{k-1} \,]\, \mathbf{G} = \mathbf{uG} = [\, \mathbf{u}\cdot\mathbf{g}_0 \;\; \mathbf{u}\cdot\mathbf{g}_1 \;\; \dots \;\; \mathbf{u}\cdot\mathbf{g}_{n-1} \,]$

    např. $\mathbf{c} = \mathbf{u} \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix}$, $n = 5$, $k = 2$

    … Každý bit kódového slova je vypočítán jako vážená suma více bitů informačních slov

    … Operace (pro každý bit) – součet: 0+0=1+1=0, 0+1=1+0=1 (součet modulo 2, exclusive OR) a součin: 0·0=0·1=1·0=0, 1·1=1

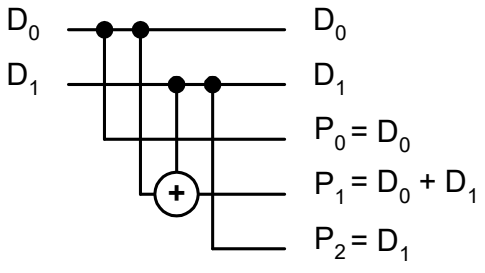
    ((operace pro tzv. Galoisovo pole $\mathbf{GF}^2$))
- Součet různých kódových slov je opět kódové slovo (<=> lineární)
- Jednoduché pro implementaci

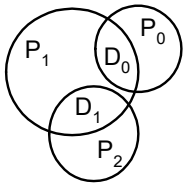
Principy protichybového kódování

Lineární blokové binární kódování

- Např. (5, 2), vektory 1 0, 0 1, 1 0, 1 1, 0 1; (minimální $d_h = 3$):

$\mathbf{c} = \mathbf{u} \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix}$, $n = 5$, $k = 2$; $u_0 = D_0$, $u_1 = D_1$





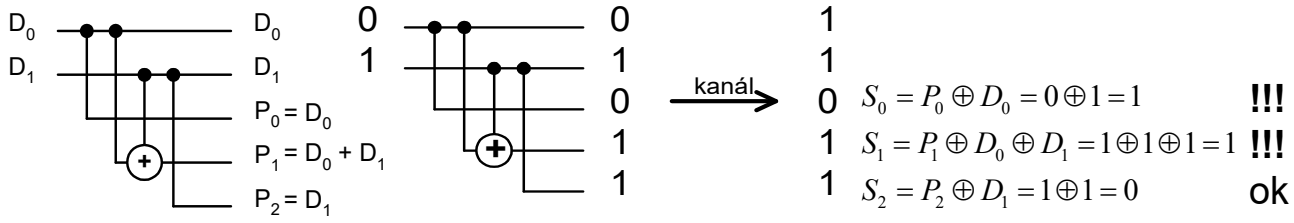
P_0, P_1, P_2 … paritní bity – výsledky paritních součtů

vhodná matice G – v levé části obsahuje diagonální jednotkovou matici … systematický kód

Protichybové zabezpečení

Lineární blokové binární kódování

– Princip dekódování spočívá v kontrole paritních součtů (parity check) – k přijatým hodnotám paritních bitů se přičte výsledek paritních součtů přijatých hodnot podle rovnic v kodéru
  - S_i jsou jednotlivé bity tzv. syndromu
  - Pokud nedošlo k chybě při přenosu, jsou S_i nulové
  - Příklad:



… pokud došlo k jednoduché chybě, dle syndromů jde o bit D_0

D_0: 1 –> 0

$S_0 = P_0 \oplus D_0 = 0 \oplus 0 = 0$ ok

$S_1 = P_1 \oplus D_0 \oplus D_1 = 1 \oplus 0 \oplus 1 = 0$ ok

$S_2 = P_2 \oplus D_1 = 1 \oplus 1 = 0$ ok

Protichybové zabezpečení

Blokové CRC (Cyclic Redundancy Check) kódování

– Do kodéru vstupuje **blok** dat o velikosti *k* bitů, k nim přidáno *m* kontrolních bitů (systematický kód), celkem na výstupu $n = k + m$ bitů

– Vhodný zápis v polynomickém tvaru
  - informační slovo lze zapsat koeficienty: $u = u_{k-1}u_{k-2}...u_1u_0$
    a můžeme jej vyjádřit pomocí polynomu řádu $k-1$:

    $u(x) = \sum_{i=0}^{k-1} u_i x^i$, například 1101 … $x^3 + x^2 + 1$ ($k = 4$)

  - kódové slovo lze zapsat koeficienty: $w = w_{n-1}w_{n-2}...w_1w_0$
    a můžeme jej vyjádřit pomocí polynomu řádu $n-1$:

    $w(x) = \sum_{i=0}^{n-1} w_i x^i$, například 1101001 … $x^6 + x^5 + x^3 + 1$ ($n = 7$)

  - Koeficienty polynomu jsou buď 0 nebo 1

– Operace s koeficienty:
  Součet (rozdíl): 0+0 = 1+1 = 0–0 = 1–1 = 0, 0+1 = 1+0 = 0–1 = 1–0 = 1,
  rozdíl lze nahradit součtem (rozdíl a součet mají totožné výsledky)
  Násobení: 0·0 = 0·1 = 1·0 =0, 1·1 = 1

Protichybové zabezpečení

Blokové CRC (Cyclic Redundancy Check) kódování

– Princip kódování: informační polynom u(x) je dělen polynomem g(x) a zbytek po dělení je připojen za u(x) jako kontrolní bity, výsledné kódové slovo je pak beze zbytku dělitelné polynomem g(x)

$$w(x) = x^{n-k}u(x) + \text{zbytek}\left\{x^{n-k}u(x)/g(x)\right\}$$

- Kódová slova generuje polynom řádu $n-k = m$ (např. $m = 7-4 = 3$)
  $g(x) = \sum_{i=0}^{n-k} g_i x^i$, například 1011 … $x^3 + x + 1$
- Násobení x^{n-k} odpovídá bitovému posunutí o $n-k$ bitů, např.: $n-k = 3$ … $x^3 \cdot 1101 = 1101000$, posunutí vytvoří prostor pro $m = n-k$ kontrolních bitů, následujících po k informačních bitech

– Cyklický kód: pokud w(x) je kódové slovo, pak je také kódovým slovem cyklicky posunuté – nutno respektovat při návrhu polynomu g(x)

$$w(x) = \sum_{i=0}^{n-1} w_i x^i = w_{n-1}x^{n-1} + w_{n-2}x^{n-2} + w_{n-3}x^{n-3} + \ldots + w_1 x^1 + w_0$$

$$w'(x) = w_{n-2}x^{n-1} + w_{n-3}x^{n-2} + \ldots + w_1 x^2 + w_0 x + w_{n-1} = x\left(w_{n-2}x^{n-2} + w_{n-3}x^{n-3} + \ldots + w_1 x^1 + w_0\right) + w_{n-1} +$$

$$+ xw_{n-1}x^{n-1} - xw_{n-1}x^{n-1} = xw(x) + w_{n-1} - w_{n-1}x^n = xw(x) + w_{n-1}\left(1 - x^n\right) = xw(x) + w_{n-1}\left(1 + x^n\right)$$

… polynom (x^n+1) musí být beze zbytku dělitelný polynomem g(x)

Protichybové zabezpečení

Blokové CRC (Cyclic Redundancy Check) kódování

– Dekódování (detekce chyby) využívá stejný algoritmus jako kódování – výhoda CRC

- Po zakódování platí: výsledné kódové slovo je beze zbytku dělitelné polynomem g(x)

  $w(x) = x^{n-k}u(x) + \text{zbytek}\left\{x^{n-k}u(x)/g(x)\right\}$ => $\text{zbytek}\left\{w(x)/g(x)\right\} = 0$

- Přijaté kódové slovo w´(x) se může od vyslaného kódového slova w(x) lišit (pokud došlo k chybám při přenosu)
- V případě přenosu bez chyb

  $\text{zbytek}\left\{w'(x)/g(x)\right\} = 0$

  např.: 1101001 / 1011 = 1111 a zbytek 000 … **OK**

- V případě chyby/chyb získáváme výpočtem

  $\text{zbytek}\left\{w'(x)/g(x)\right\} = s(x) \neq 0$

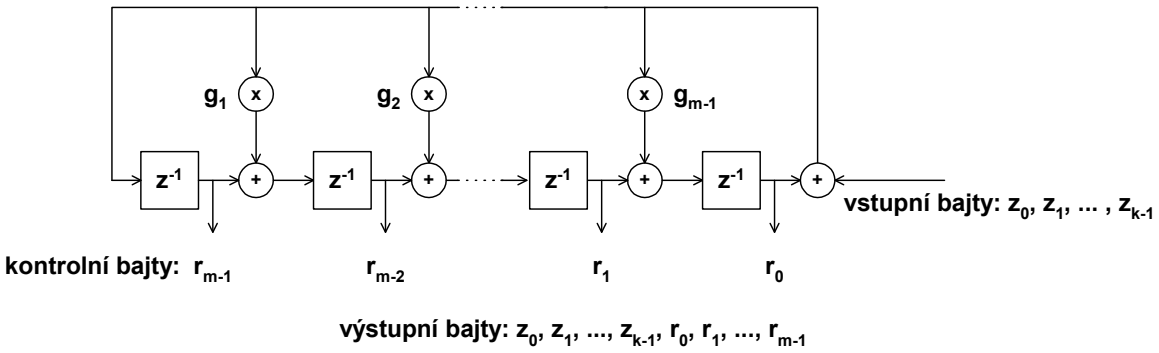
  nenulový syndrom ($m = n-k$ bitů), ze kterého lze odvodit pozice chybných bitů, pokud není překročena schopnost opravy (oprava chyb)

  např.: 1**0**01001 / 1011 = 1010 a zbytek 111 … **!!!**
  (pozn.: algoritmy určení pozice chybných bitů jsou nad rámec přednášky)

Protichybové zabezpečení

Blokové CRC (Cyclic Redundancy Check) kódování

– Blokové schéma s registry pro výpočet zbytku po dělení (obecně kodér s posuvnými registry může zpracovávat jak bajty tak i bity; pro zjednodušení lze místo bajtů uvažovat bity)

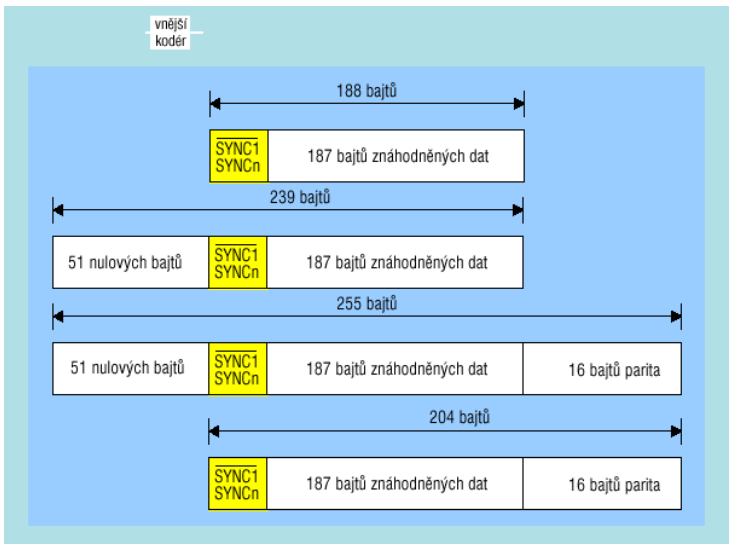


– Do kodéru vstupuje blok dat o velikosti *k* bajtů, k nim přidáno *m* kontrolních bajtů, celkem na výstupu $n = k + m$ bajtů, opravit lze až $m/2$ bajtů
– RS kódování (Reed Solomonovo) – např. RS(255 B, 239 B), tj. $k = 239$ B, $n = 255$ B (=2040 bitů) a opravit lze až 8 bajtů;
– BCH kódování (Bose, Chaudhuriho, Hocquenghema) – např. BCH(32400, 32208), tj. $k = 32400$, $n = 32208$ a opravit lze 12 bitů

Protichybové zabezpečení

Zkrácený kód CRC kódování

– Pro kratší zprávy, než pro jakou délku je navržen kód
– Zpráva je doplněna nulovými bity
– Např. obr. … RS(255, 239) pro pakety délky 188 bajtů
  - Kódování lze využít po doplnění 51 nulových bajtů před 188 bajtů dat
  - Po výpočtu kontrolních bajtů a sestavení kódového slova se opět oddělí prvních 51 bytů
  - Výsledkem je kódové slovo dlouhé 204 bajtů
  - Zkrácený kód se značí RS(204, 188)



Protichybové zabezpečení

Blokové LDPC (Low Density Parity Check) kódování

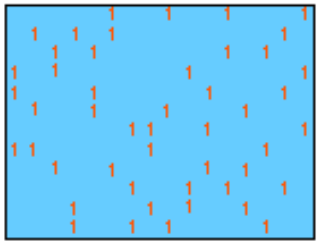
- Lineární blokový kód (bloky velmi rozměrné) & maticové vyjádření
- Skupině informačních bitů **u** délky *k* se přiřadí kódové slovo **c** délky *n* maticovým násobením s maticí **G** (generující matice) o velikosti *k*,*n*

  c = **u** x **G**, např. *k* = 48600 a *n* = 64800 bitů, CR = 3/4
- Specifické pro LDPC je způsob sestavení **G**, tak aby bylo kódování dobře implementované a aby bylo možno nalézt matici **H** pro dekódování, která bude obsahovat velký počet nulových prvků
  - **G** x $\mathbf{H}^T$ = **0** (nulové hodnoty představují přenos bez chyb)
  - Matice **H**

    … **low density**: řídká – malý počet nenulových prvků

    … **parity check**: na přijímací straně slouží ke kontrole **c**x$\mathbf{H}^T$=**0**
  - Návrh kódu spočívá ve vytvoření matice **G**

H =



Protichybové zabezpečení

Blokové LDPC (Low Density Parity Check) kódování

- Způsob sestavení **G**, tak aby bylo kódování dobře implementované a aby bylo možno nalézt matici **H** (velikosti *m*,*n*; *m*=*n*-*k*) pro dekódování, která bude obsahovat velký počet nulových prvků

  … např.: eIRA LDPC, extended irregular repeat accumulate
  - **H** = [$\mathbf{H_1}$ $\mathbf{H_2}$] … $\mathbf{H_1}$ o velikosti *m*,*k* je řídká a $\mathbf{H_2}$ má rozměr *m*,*m* a je tvaru:

$$\mathbf{H}_2 = \begin{bmatrix} 1 & & & & & \\ 1 & 1 & & & & \\ & 1 & 1 & & & \\ & & & \ddots & & \\ & & & & 1 & 1 \\ & & & & & 1 \end{bmatrix}$$

$$\mathbf{H}_2^{-T} = \begin{bmatrix} 1 & 1 & \cdots & 1 & 1 \\ & 1 & \cdots & 1 & 1 \\ & & \ddots & \vdots & \vdots \\ & & & & \\ & & & 1 & 1 \\ & & & & 1 \end{bmatrix}$$

  - **G** pak má tvar **G** = [**I P**], **P** = $\mathbf{H_1}^T$ x $\mathbf{H_2}^{-T}$ … **I** je diagonální jednotková matice => systematický kód,

    $\mathbf{H_2}^{-T}$ představuje diferenciální kodér (relativně jednoduchá implementace – akumulátor),

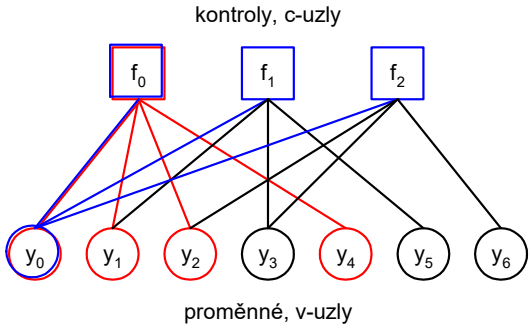
    $\mathbf{H_1}^T$ je opět řídká (malý počet násobení)

Protichybové zabezpečení

Blokové LDPC (Low Density Parity Check) kódování

– Dekódování: oprava hodnot přijatého slova **y** tak, aby platilo $\mathbf{y}\times\mathbf{H}^T=\mathbf{0}$
  - Může probíhat iterativně (Gallager, 1960) … lze se přiblížit Shannonnovu limitu (cca o 1 dB)
  - Výhodná je grafická reprezentace matice **H** … Tannerův graf (Tanner, 1981),
    n uzlů proměnných … v-uzly (variable), hodnoty y_i
    m uzlů kontrol … c-uzly (check), hodnoty f_i
    Např. kód (7, 4),
    $n = 7$, $k = 4$,
    $m = 3$

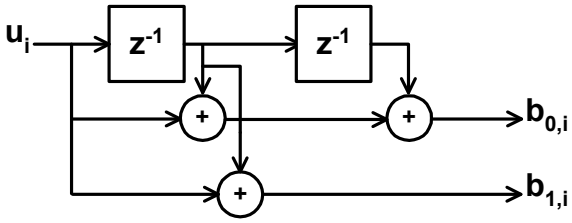
$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$



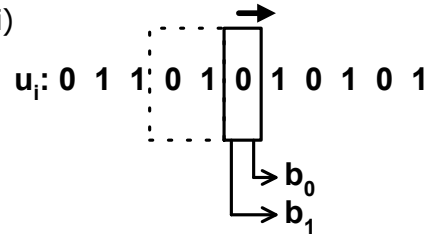
  - Algoritmus je založen na předávání hodnot (message passing algorithm) mezi v-uzly a c-uzly:
    a) Z v-uzlů proměnných (z vnějšků) vstupují hodnoty na základě přijatého kódového slova
    b) V opačném směru z c-uzlů jsou předávány výsledky paritních součtů – v případě přenosu bez chyb je součtem hodnot všech bitů směřujících do c-uzlu 0
    - Principem je iterativní oprava hodnot přijatého slova (proměnných) – opakování kroků a) a b), tak aby paritní součty (kontroly) byly 0, resp. $\mathbf{c}\times\mathbf{H}^T=\mathbf{0}$, v každé iteraci se opraví odhalené chyby v přijatém slově
    - Algoritmus končí, pokud jsou všechny součtu nulové, nebo pro zvolený max. počet iterací

Protichybové zabezpečení

Konvoluční kódování

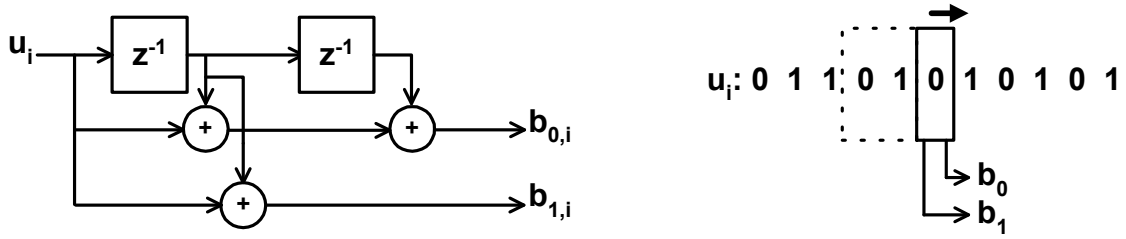


– Rozdíly proti dříve popsaným blokovým kódům
  - Nejsou (obecně) systematické
  - Vstupní informace je v kodéru zpracována bit po bitu
– Kódové slovo tvoří *N* bitů (*N* výsledků součtů) pro každý bit na vstupu
– Princip kódování: výpočet součtu aktuální a předchozích hodnot bitu v různých výstupních větvích v rámci klouzavého okna
  - Vstupní informační slovo představuje časová posloupnost hodnot bitu
  - Paměť předchozích hodnot bitu zajišťuje posuvný registr (zpožďovací členy z^{-1}, inicializace nulovými hodnotami)
  - Součet: modulo 2 (členy ⊕)
  - *N* výstupních větví (v obr. *N*=2)
  - Klouzavé okno se posouvá s krokem jeden bit přes informační slovo



Protichybové zabezpečení

Konvoluční kódování



– Délka okna (souvisí s počtem členů z^{-1}) … hloubka kódu (constraint length) K (v obr. K=3)
  - Větší K =>
    každý informační bit ovlivní více součtů ≈ větší odolnost proti chybám (robustnost), ale
    vyšší výpočetní náročnost kódování (a dekódování)

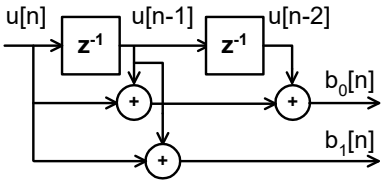
– Počet výstupních větví, N
  - Kódový poměr: CR=1/N (v obr. CR=1/2)
  - Větší N =>
    nižší CR … větší robustnost, ale
    větší overhead (1 – CR) … nižší užitečná přenosová rychlost bitového toku

Protichybové zabezpečení

Konvoluční kódování

– Polynomický zápis kódování
  - Rovnice pro výsledné součty

$$b_0[n] = u[n] \oplus u[n-1] \oplus u[n-2]$$
$$b_1[n] = u[n] \oplus u[n-1]$$



lze zapsat obecně

$$b_i[n] = \left(\sum_{j=0}^{K-1} g_i[j]\, u[n-j] \right) \bmod 2 \qquad i = 0,\ 1, \ldots, N-1$$

… g je generujícím polynomem, např. pro kodér na obr.
jsou koeficienty g_0 a g_1: (1 1 1) a (1 1 0)

  - Obecný zápis rovnic odpovídá **konvoluci** informačních bitů s generujícím polynomem

– Snadná implementace kódování s posuvnými registry (připomíná filtraci, při které výstupní signál vzniká konvolucí vzorků vstupního signálu s impulsovou odezvou filtru)

Protichybové zabezpečení

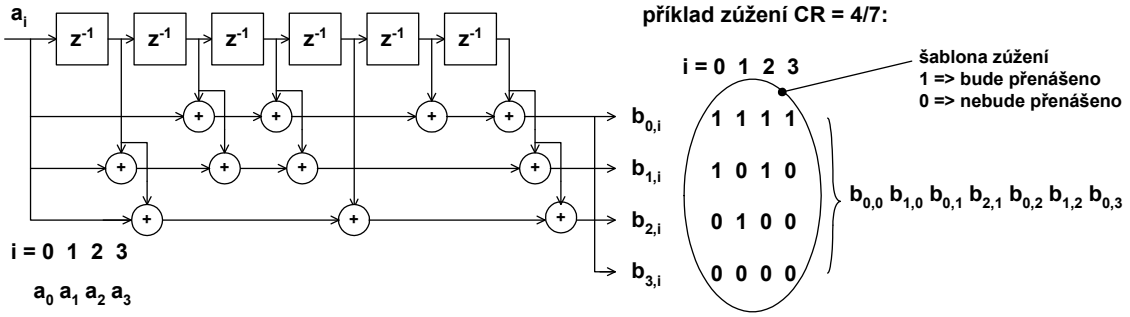
Konvoluční kódování

- Princip dekódování spočívá v nalezení nejpravděpodobnějšího (ML, maximal likelihood) vyslaného kódového slova pro známe přijaté slovo s možným výskytem chyb – na základě nejmenší Hamm. vzdálenosti d_h
- Např.:
  - Informaci 1 1 0 0 odpovídá kódové slovo z kodéru 11 00 01 10
  - Po průchodu kanálem obdržíme 1**0** 0**1** 01 10 (vyskytly se 2 chyby)
  - Pro 4 informační bity je možno obdržet 16 kódových slov
  - Nejblíže (min. d_h) přijatému slovu je 11 00 01 10, které odpovídá informaci 1 1 0 0
  - (Zároveň lze vypočítat BER před opravou: přijaté slovo se od kódového liší ve 2 bitech přeneseno bylo 8 bitů; BER = 2/8)
- Takový postup je nemyslitelný pro obvyklé velikosti zpráv
  - Např. pro 256 bitů je $2^{256} \approx 1{,}158e+77$
  - Existují výpočetně jednodušší algoritmy (např. Viterbiho algoritmus z roku 1957)

Informace	Kódové slovo	Přijaté slovo	h_d
0 0 0 0	00 00 00 00	10 01 01 10	4
0 0 0 1	00 00 00 11		4
0 0 1 0	00 00 11 11		4
0 0 1 1	00 00 11 01		5
0 1 0 0	00 11 11 10		3
0 1 0 1	00 11 11 01		5
0 1 1 0	00 11 01 00		3
0 1 1 1	00 11 00 10		3
1 0 0 0	11 11 10 00		5
1 0 0 1	11 11 10 11		5
1 0 1 0	11 11 01 11		3
1 0 1 1	11 11 01 00		3
1 1 0 0	**11 00 01 10**		**2**
1 1 0 1	11 00 01 01		4
1 1 1 0	11 00 10 01		6
1 1 1 1	11 00 10 10		4

Protichybové zabezpečení

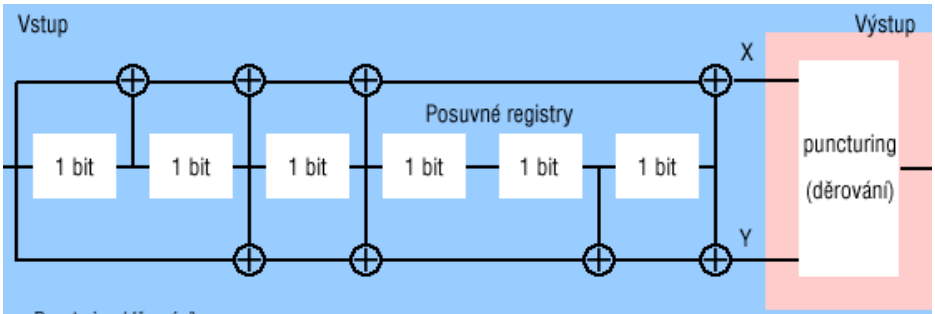
Punkturování v konvolučním kódování



- Nebo také zúžení nebo také děrování … snížení počtu výstupních bitů z více větví konvolučního kodéru (viz obr.)
- Kód se všemi výstupními bity (nepunkturovaný) označujeme za mateřský (např. v obr. CR=1/4)
- Po punkturování … zvýší se CR (např. v obr. na 4/7)
  - Zvýší se užitečná přenosová rychlost
  - Sníží se robustnost
- Jediná struktura kódu pro daný systém nabídne množinu kódů s různými CR (pro různé šablony zúžení) – volitelný parametr systému
- (Uvedený kodér použit v DAB a v DRM)

Protichybové zabezpečení

Punkturování v konvolučním kódování
- Další příklad
- 5 hodnot CR dle různého punkturování
- (Uvedený kodér použit v DVB, $K = 7$, g0 = 1011011, g1 = 1111001)

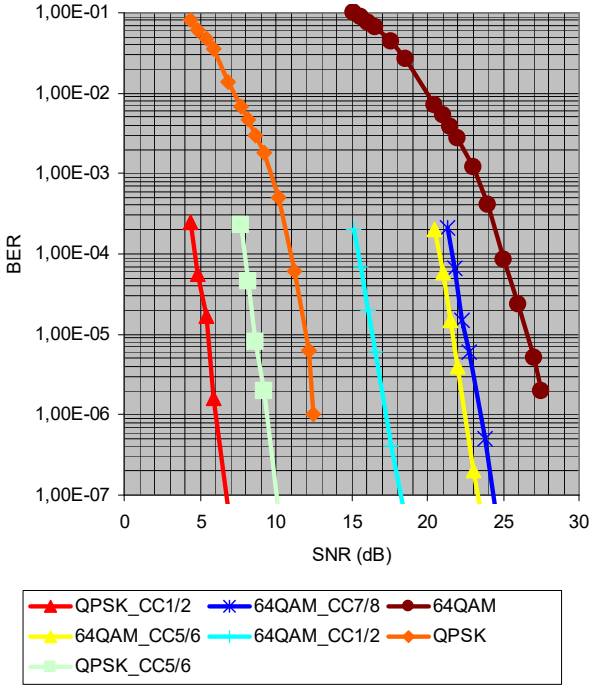


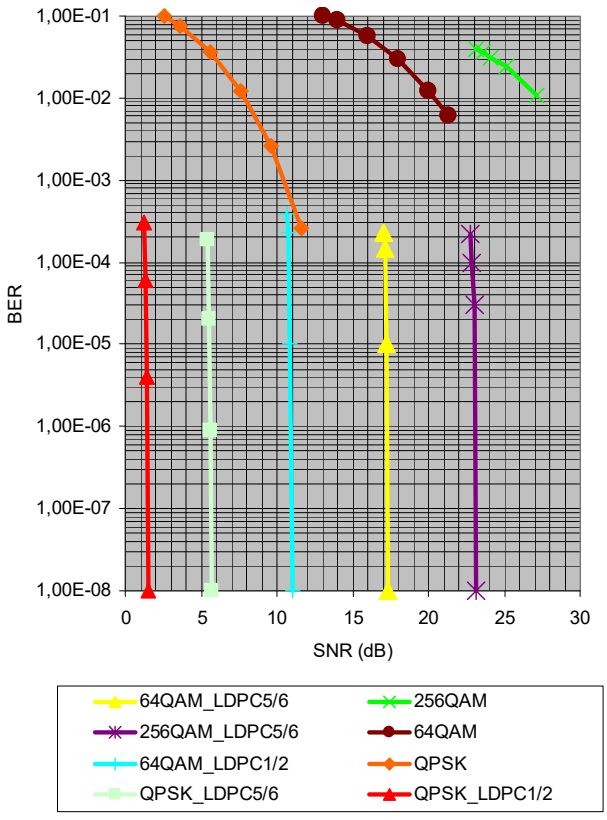
Puncturing (děrování)

Kodový poměr	Šablona puncturingu (děrování)		Výsledná sekvence
1/2	X: 1	Y: 1	X1 Y1
2/3	X: 1 0	Y: 1 1	X1 Y1 Y2
3/4	X: 1 0 1	Y: 1 1 0	X1 Y1 Y2 X3
5/6	X: 1 0 1 0 1	Y: 1 1 0 1 0	X1 Y1 Y2 X3 Y4 X5
7/8	X: 1 0 0 0 1 0 1	Y: 1 1 1 1 0 1 0	X1 Y1 Y2 Y3 Y4 X5 Y6 X7

Protichybové zabezpečení

Srovnání konvolučního kódování (CC) a LDPC
- Reálné měření chybovosti (DVB-T/T2)

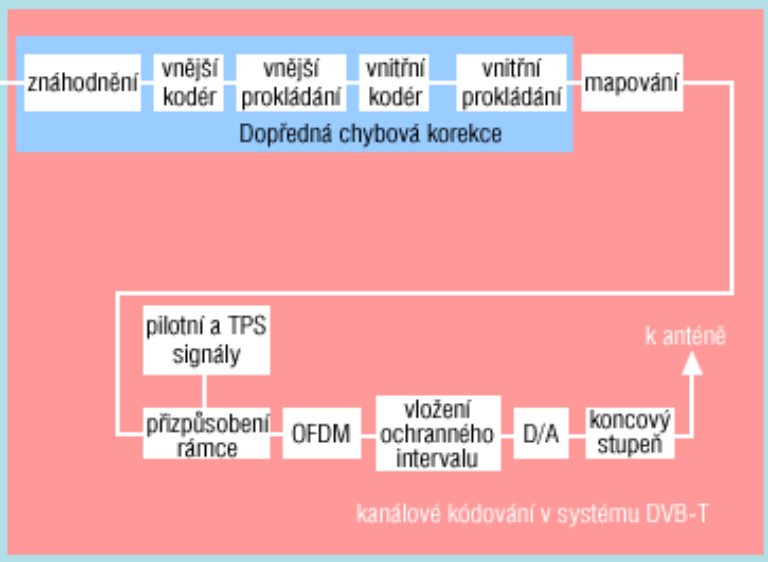




Protichybové zabezpečení

Více úrovňové zabezpečení (též také řetězový kód)

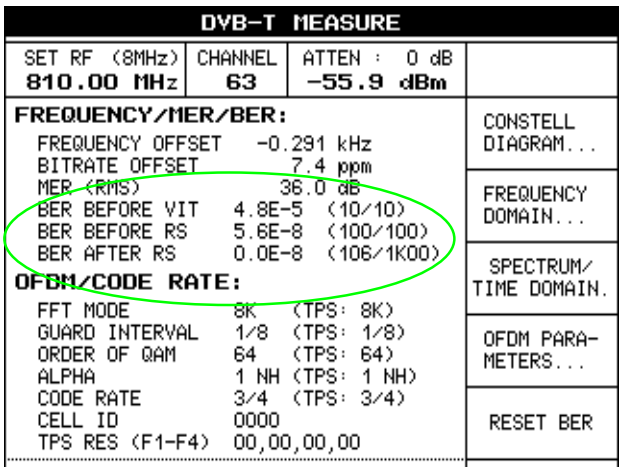
- Další snížení chybovosti při přenosu zajistí řetězení více kódů – možno sériově, paralelně, kombinace
- Na obr. příklad dvou úrovňového sériově řazeného zabezpečení
  - Vysílací strana (na obr.)
    - Vnější (např. RS nebo BCH) kodér
    - Vnější prokládání
    - Vnitřní (např. konvoluční nebo LDPC) kodér
    - Vnitřní prokládání
  - Přijímací strana
    - Vnitřní zpětné poskládání
    - Vnitřní dekodér (např. konvoluční nebo LDPC)
    - Vnější zpětné poskládání
    - Vnější dekodér (např. RS nebo BCH)

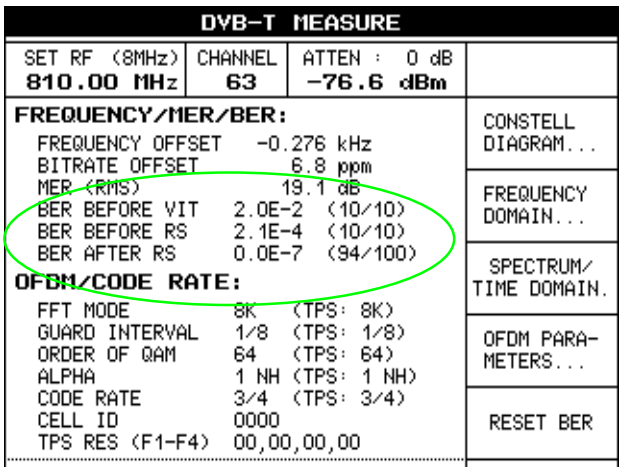


Protichybové zabezpečení

Více úrovňové zabezpečení

- Reálné měření chybovosti (DVB-T) – sériové řazení Reed-Solomon a konvoluční kodér, konvoluční (VIT) dekodér a Reed-Solomonův (RS) dekodér

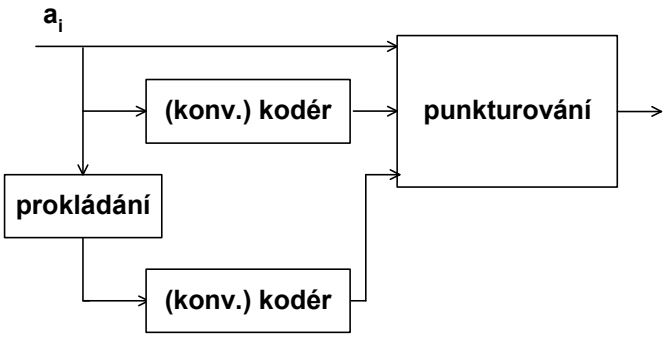




Před opravou chyb = BEFORE VIT
Za vnitřním dekodérem = BEFORE RS
Za vnějším dekodérem = AFTER RS

Protichybové zabezpečení

Více úrovňové zabezpečení
- Turbo kodér
  - Princip spočívá v paralelním užití více kodérů (v obr. například 2 konvoluční kodéry)
  - Podstatným prvkem je užití prokládání



Témata pro přípravu studentů ke zkoušce

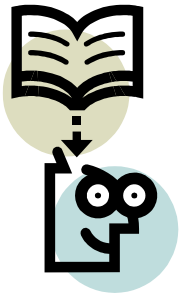
Protichybové zabezpečení
- Obecné principy, kódový poměr, systematický kód

Blokové CRC (Cyclic Redundancy Check) kódování
- Princip kódování a dekódování
- Zkrácený kód

Blokové LDPC (Low Density Parity Check) kódování
- Princip kódování a dekódování

Konvoluční kódování
- Princip kódování a dekódování
- Punkturování v konvolučním kódování

Děkuji za pozornost,



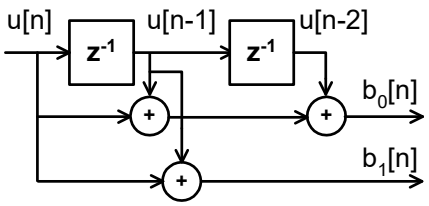
prosím vaše dotazy …

Protichybové zabezpečení

Konvoluční kódování

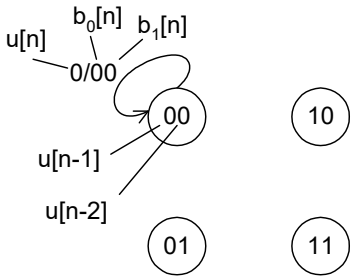
– Existují výpočetně méně náročnější algoritmy pro výběr nejpravděpodobnějšího, populárním je **Viterbiho algoritmus** (1957) – využívající popis kódování jako stavový diagram (obr. dole)

- Např. pro kód podle obr. (schéma s registry)

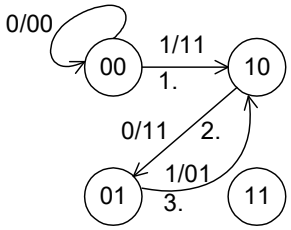


- *K* – *1* vnitřních stavů (*K* hloubka kódu) … např. u[n-1] a u[n-2] pro *K*=3
- 2^{K-1} možných kombinací vnitřních stavů … např. 4 pro u[n-1] a u[n-2]

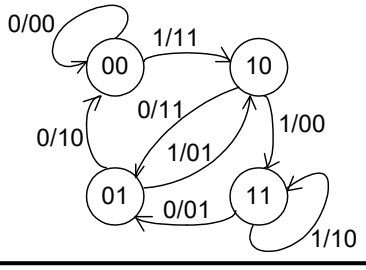
Počáteční stav



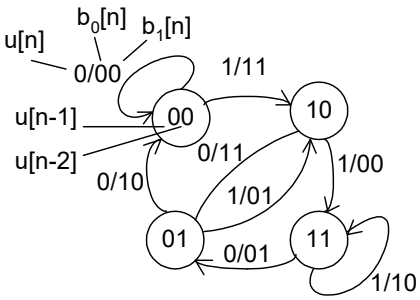
Přechody postupně pro 3 bity na vstupu s hodnotami 1, 0, 1



Všechny možné přechody

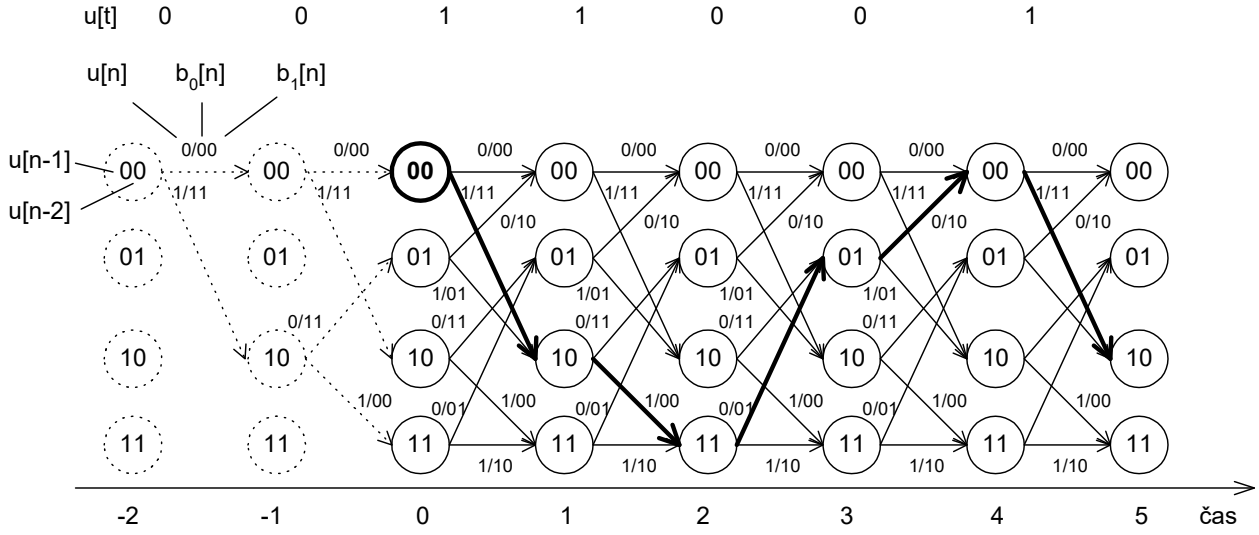


Protichybové zabezpečení

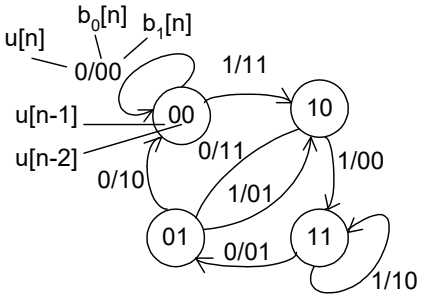


Konvoluční kódování

- Stavový popis velmi jednoduše popisuje kódování
  - Začíná se od 0/00 00
  - O volbě přechodu rozhoduje aktuální bit informačního slova **u** na vstupu
  - Ze zvoleného přechodu vyplývá výstupní kódové slovo a nový stav kodéru atd. … na obr. dole v závislosti na čase – mřížka (trellis), pro vstupní bity 1 1 0 0 1 je výstupem 11 00 01 10 11

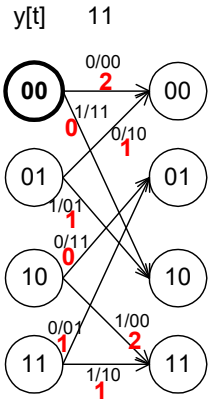


Protichybové zabezpečení

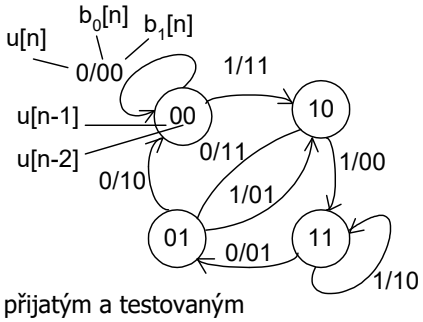


Konvoluční kódování

- Dekódování Viterbiho algoritmem
  - Sledují se možné vnitřní stavy a přechody mezi nimi v závislosti na čase
  - Např. pro vstupní bity 1 1 0 0 1 je vyslané kódové slovo 11 00 01 10 11 a přijaté slovo y 11 0**1** 01 10 11 (1 chyba)
  - Sledovány jsou dvě metriky: BM (branch metric) – Hamm. vzdálenost mezi přijatým a testovaným (potenciálně vyslaným) slovem, na obr. červeně a …



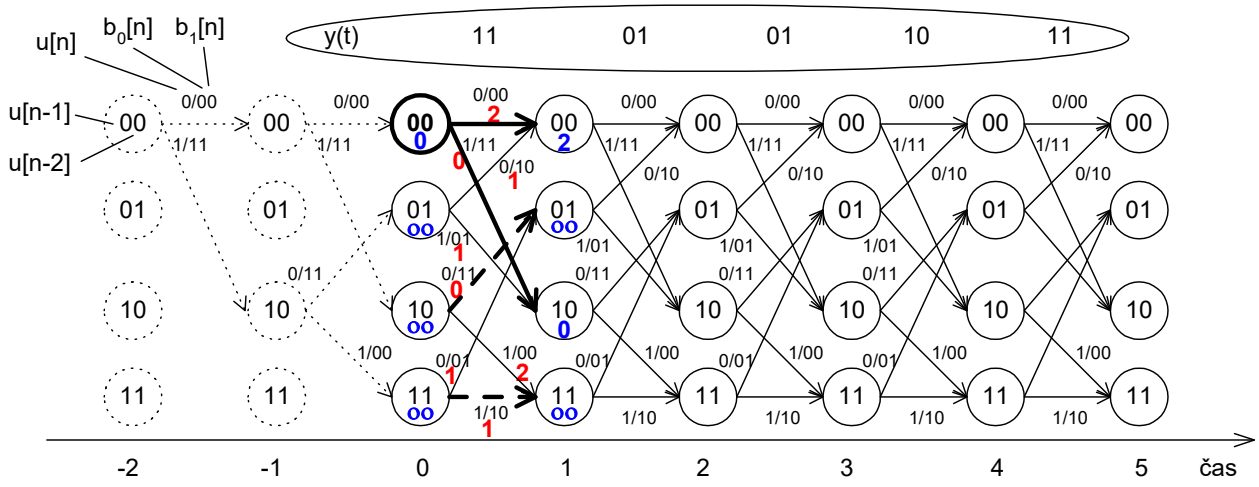
Protichybové zabezpečení



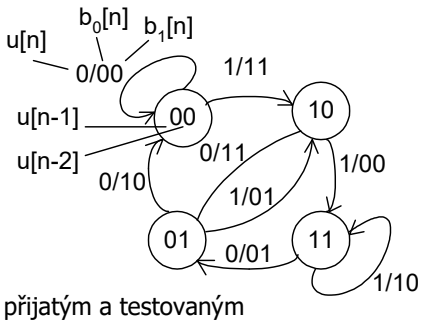
Konvoluční kódování

– Dekódování Viterbiho algoritmem

- Např. pro vstupní bity 1 1 0 0 1 je vyslané kódové slovo
  11 00 01 10 11 a přijaté slovo y 11 0**1** 01 10 11 (1 chyba)
- Sledovány jsou dvě metriky: BM (branch metric) – Hamm. vzdálenost mezi přijatým a testovaným (potenciálně vyslaným) slovem, na obr. červeně a
  PM (path metric) – Hamm. vzdálenost celé cesty od počátečního stavu do aktuálního, na obr. modře
- Cesta s minimálním PM je nejpravděpodobnější – PM je postupně počítána na základě předchozích stavů a aktuálních BM, do každého stavu zvolíme cestu s menší PM



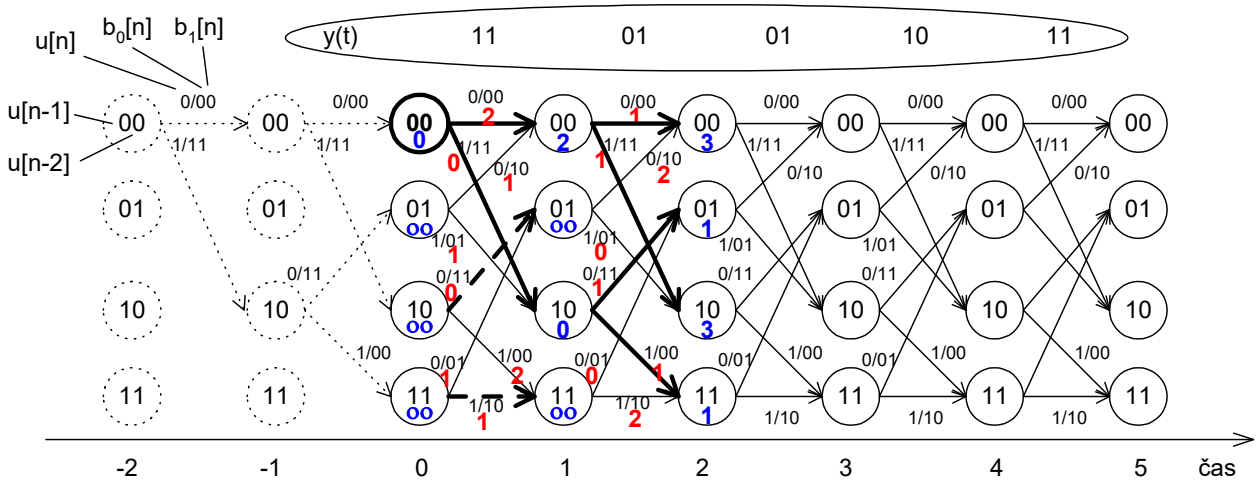
Protichybové zabezpečení



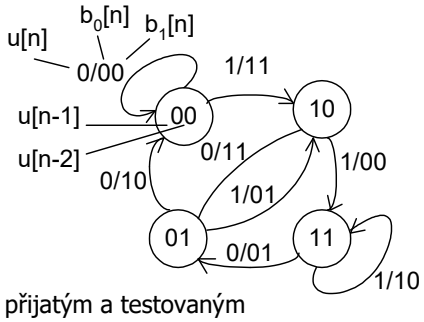
Konvoluční kódování

– Dekódování Viterbiho algoritmem

- Např. pro vstupní bity 1 1 0 0 1 je vyslané kódové slovo
  11 00 01 10 11 a přijaté slovo y 11 0**1** 01 10 11 (1 chyba)
- Sledovány jsou dvě metriky: BM (branch metric) – Hamm. vzdálenost mezi přijatým a testovaným (potenciálně vyslaným) slovem, na obr. červeně a
  PM (path metric) – Hamm. vzdálenost celé cesty od počátečního stavu do aktuálního, na obr. modře
- Cesta s minimálním PM je nejpravděpodobnější – PM je postupně počítána na základě předchozích stavů a aktuálních BM, do každého stavu zvolíme cestu s menší PM



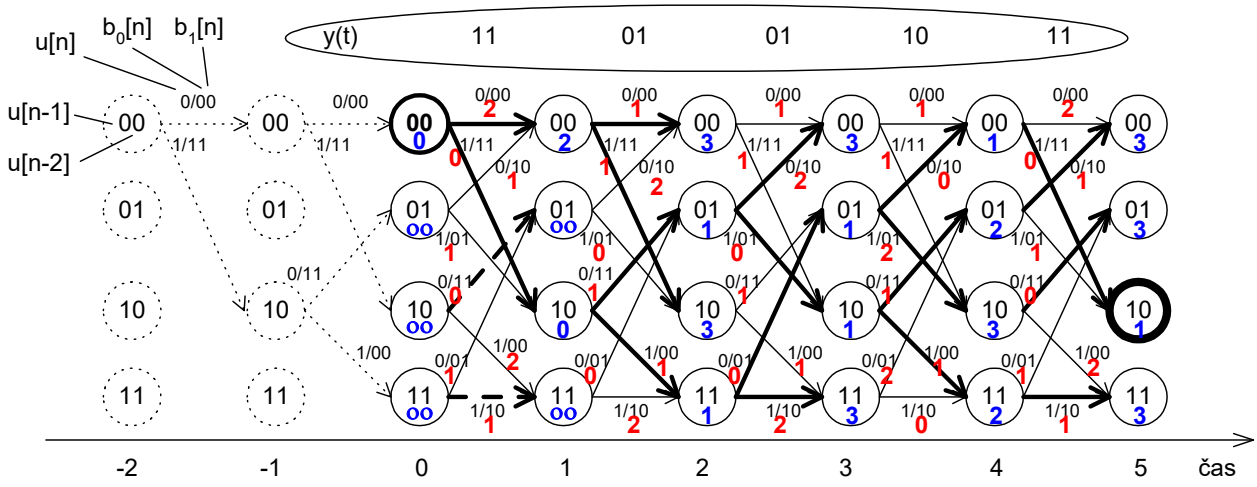
Protichybové zabezpečení



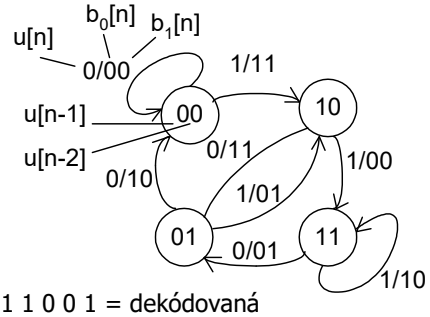
Konvoluční kódování

– Dekódování Viterbiho algoritmem

- Např. pro vstupní bity 1 1 0 0 1 je vyslané kódové slovo
  11 00 01 10 11 a přijaté slovo y 11 0**1** 01 10 11 (1 chyba)
- Sledovány jsou dvě metriky: BM (branch metric) – Hamm. vzdálenost mezi přijatým a testovaným (potenciálně vyslaným) slovem, na obr. červeně a
  PM (path metric) – Hamm. vzdálenost celé cesty od počátečního stavu do aktuálního, na obr. modře
- Cesta s minimálním PM je nejpravděpodobnější – PM je postupně počítána na základě předchozích stavů a aktuálních BM, do každého stavu zvolíme cestu s menší PM



Protichybové zabezpečení



Konvoluční kódování

– Dekódování Viterbiho algoritmem

- Např. pro vstupní bity 1 1 0 0 1 je vyslané kódové slovo
  11 00 01 10 11 a přijaté slovo y 11 0**1** 01 10 11 (1 chyba)
- Nejpravděpodobnější cesta má nakonec PM=1, vede přes informační bity: 1 1 0 0 1 = dekódovaná informace
- Optimální nejvíce pravděpodobnou cestu lze odhalit až po zpracování celé přijaté zprávy – nevýhoda (existují algoritmy pracující s kratšími úseky zprávy – délka souvisí s násobkem hloubky *K*)