Struktura a architektura počítačů

Katedra číslicového návrhu
Fakulta informačních technologií
České vysoké učení technické

Lineární kódy: příklady implementace

BI-SAP, duben 2021

Obsah

Doplnění BI-LIN o praktické využití teorie

- Bezpečnostní kódy
  - Detekční: objevují (zjišťují) chyby, tzn. zjistí zda slovo je kódové
  - Samoopravné: opravují chyby, tzn. slovo nahradí „nejpodobnějším“ kódovým slovem
- Lineární kód, lineární – vektorový prostor
- Implementace

Využití zdrojů:

Komentované podklady z 5. přednášky BI-LIN

Souvislosti a terminologie z BI-JPO

BI-JPO BI-LIN

Problém k řešení

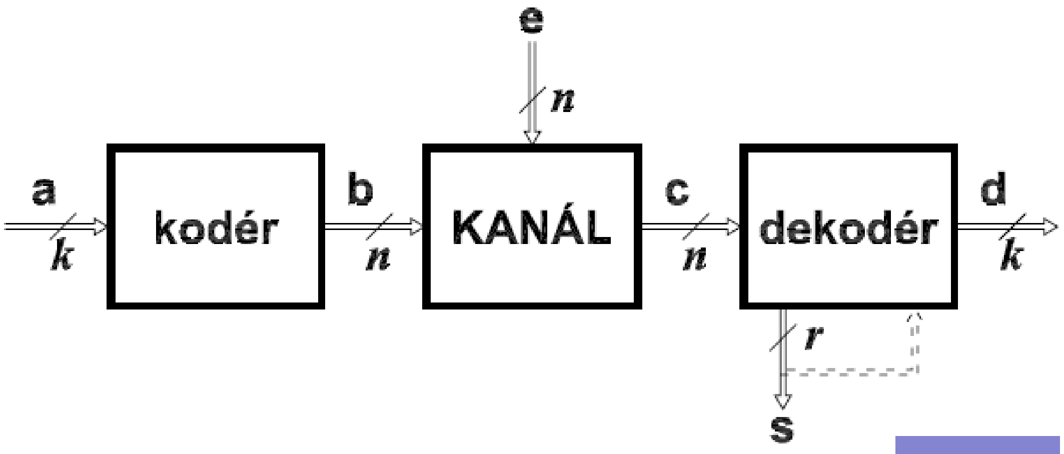
Víme, že při přenosu dat mezi pamětí a procesorem dojde občas k chybě (záleží na aplikaci, a na tom, zda to vadí a proč, jak často, ...).

Chceme buď:

- Vědět, že k chybě došlo a zařídit se podle toho (např. opakovat přenos)
- Chybu opravit, a to tak, aby se nepoznalo, že nastala

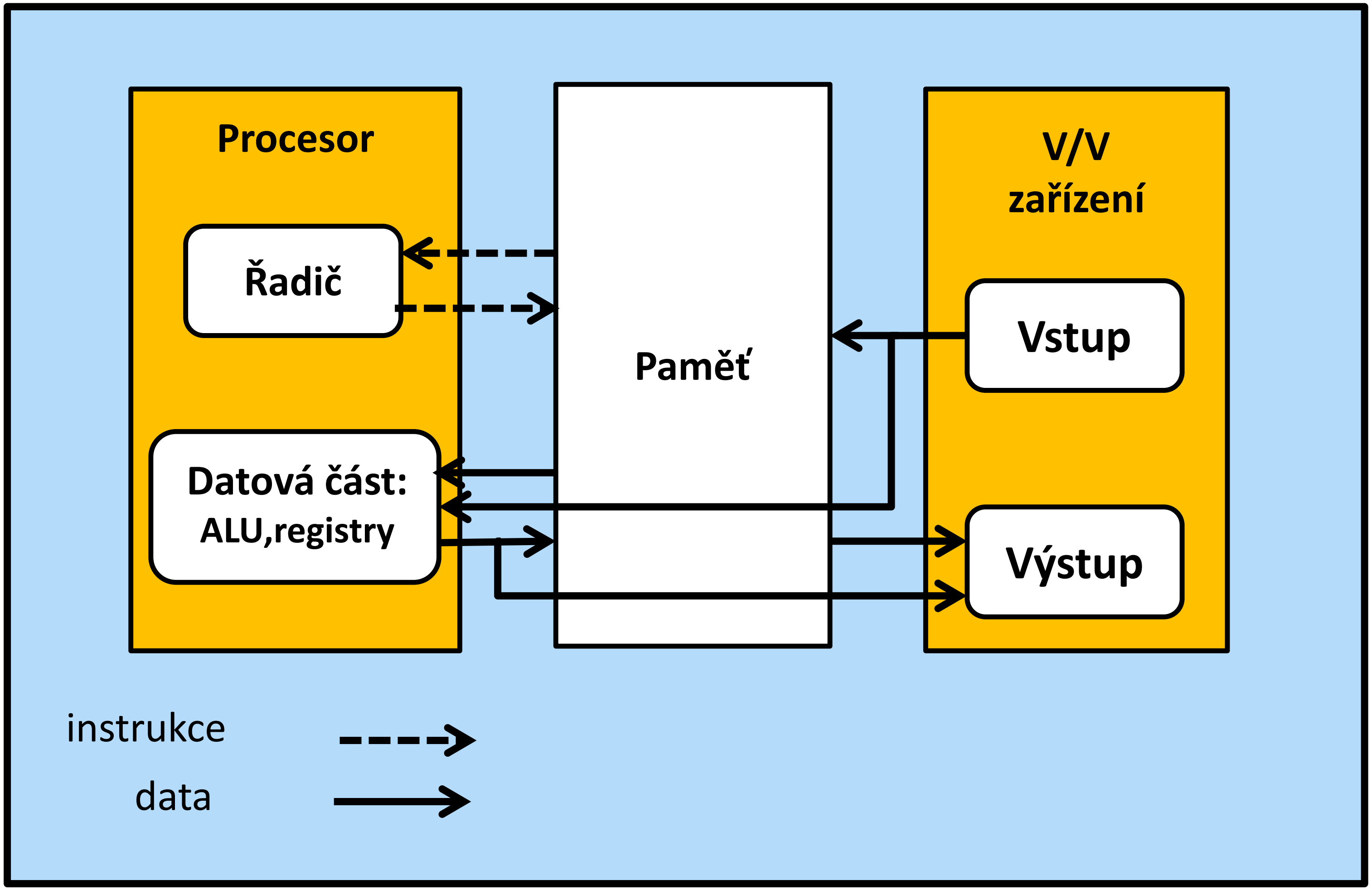
Ale vždycky chceme aby:

- se snadno a efektivně kódovalo a dekódovalo, a to pro požadovanou délku slova (tzn. informační bity)
- nutná redundance byla co nejmenší



BI-JPO

Počítač von Neumannova typu



Lineární kód

BI-LIN

Definice

*Buď $\mathcal{A}$ konečné těleso, K nazveme **lineární** (n, k)-**kód**, jestliže je K podprostor $\mathcal{A}^n$ dimenze k.*
*Nechť $k \in \{1, 2, \ldots, n-1\}$. Matici $G_K \in \mathcal{A}^{k,n}$, jejíž řádky tvoří bázi K, nazýváme **generující maticí** K, matici $H_K \in \mathcal{A}^{n-k,n}$ takovou, že K je množina řešení soustavy $H_K\mathbb{x} = \theta$, nazýváme **kontrolní maticí** K.*

.... *H* je generující maticí ortogonálního podprostoru k *G*

Věta

*Lineární (n, k)-kód má k informačních a $n-k$ kontrolních bitů. Navíc platí, že je systematický právě tehdy, když generující matici **lze volit** ve tvaru*

$$G_K = \begin{pmatrix} \mathbb{E}_k & \mathbb{A} \end{pmatrix},$$

kde $\mathbb{E}_k$ je jednotková matice typu $k \times k$ a $\mathbb{A}$ je nějaká matice typu $k \times n-k$.

.... jsme schopni oddělit informační bity od kontrolních

GF(2) – dvouprvkové těleso

2prvkové těleso GF(2): [Galois field]

$$\begin{array}{ll} 0+0=0 & 0\cdot 0=0 \\ 0+1=1 & 0\cdot 1=0 \\ 1+0=1 & 1\cdot 0=0 \\ 1+1=0 & 1\cdot 1=1 \end{array}$$

tzn. XOR **tzn. AND**

$x+x=0 \Rightarrow \boxed{-x=+x}$

$\boxed{x+0=x}$ a $\boxed{x+1=\overline{x}}$

$\underline{v}=(v_1,\ldots,v_j)$, $\underline{u}=(u_1,\ldots,u_j)$, a $\underline{0}=(0,\ldots,0)$

$\boxed{\underline{v}+\underline{u}=(v_1+u_1,\ldots,v_j+u_j)}$

$\boxed{\underline{v}+\underline{0}=\underline{v}}$ a $\boxed{\underline{v}+\underline{v}=\underline{0}} \Rightarrow \boxed{-\underline{v}=+\underline{v}}$

$\boxed{0\cdot\underline{v}=\underline{0}}$ a $\boxed{1\cdot\underline{v}=\underline{v}}$

lineární / vektorový prostor nad tělesem GF(2):

BI-JPO

Kódová vzdálenost

- Hammingova vzdálenost

BI-LIN

Definice

Pro dvě slova $u = u_1 u_2 \dots u_n$ a $v = v_1 v_2 \dots v_n$ stejné délky n a nad stejnou abecedou definujeme ***Hammingovu vzdálenost*** *jako*

$$d(u, v) = \text{ počet indexů } i \in \hat{n} \text{ takových, že } u_i \neq v_i.$$

Hammingova vzdálenost $d(u, v)$ vlastně vyjadřuje, kolik nejméně *chyb* musím udělat, abych ze slova u vyrobil slovo v.

.... počet odlišných bitů, např. vzdálenost 11100 a 01011 je 4

- Kódová vzdálenost (kvzd): minimální vzdálenost dvojic kódových slov

Kódová vzdálenost

- Hammingova váha: počet nenulových bitů
- Kódová vzdálenost = minimální váze nenulového kódového slova

$kvzd = 2 \Rightarrow$ lze zjistit 1 chybu (ale nelze ji opravit)

$kvzd = 3 \Rightarrow$ lze zjistit 2 chyby (ale nelze je opravit)
anebo zjistit a opravit 1 chybu
— je třeba se rozhodnout pro jednu alternativu!

$kvzd = 4 \Rightarrow$ lze zjistit 3 chyby (ale nelze je opravit)
anebo zjistit 2 chyby a opravit jenom 1 chybu

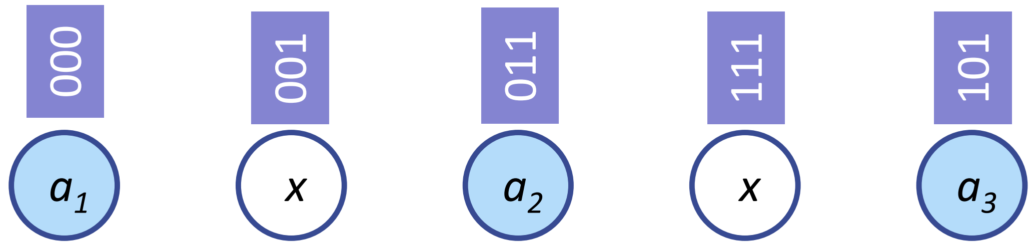
BI-JPO

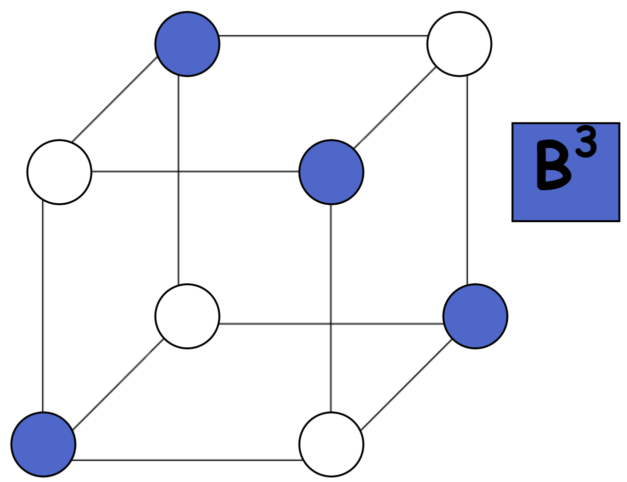
Interpretace na příkladech:

kvzd = 2

(3,2)-kód

2^2 slov

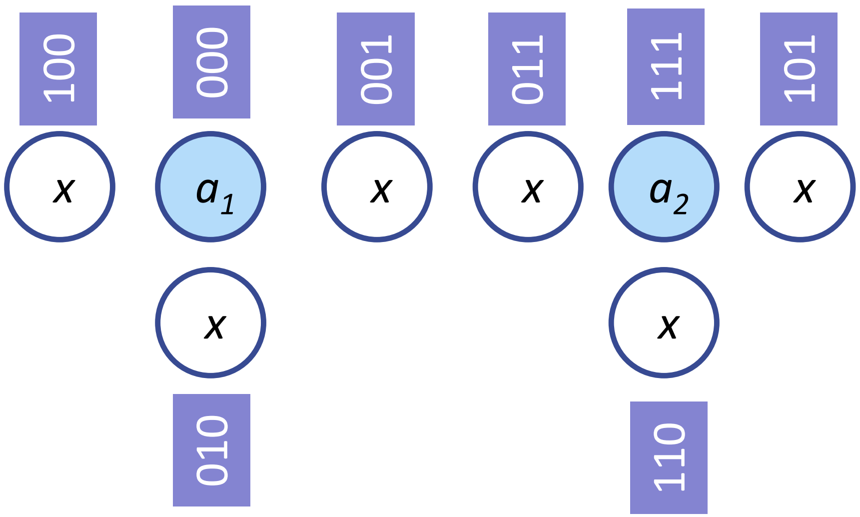




Interpretace na příkladech:

kvzd = 3, tzn. oprava chyby

Jaký bude (*n,k*) kód?



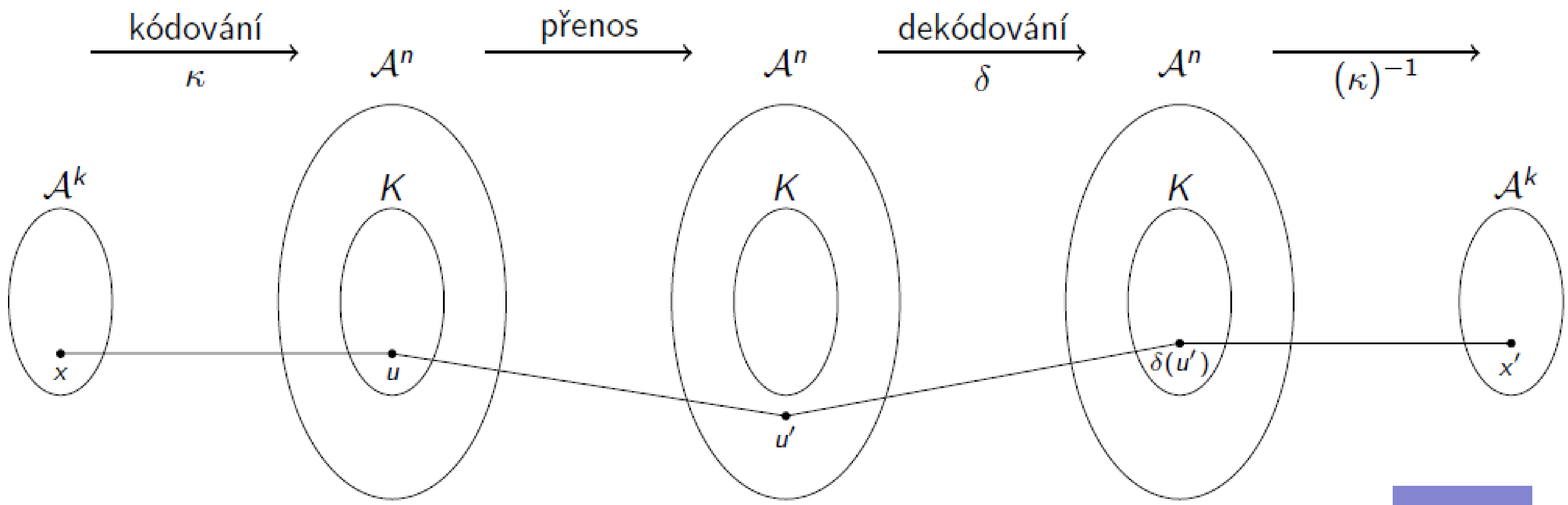
ALE! Záleží na počtu bitů, které musíme přidat, abychom toho dosáhli, a to pro všechna slova!

Redundance .... Kolik je zde (*n,k*)?

Kódování a dekódování

Věta

Lineární kód K objevuje, resp. opravuje t-chyby právě tehdy, když je soubor ***libovolných*** *t, resp. 2t sloupců v jeho kontrolní matici LN.*



BI-LIN

Lineární prostor, generující a kontrolní matice ... jejich implemetace

Příklad

Mějme 4 bitová slova. Jaký kód použít pro **detekci** jedné (jednonásobné) chyby a jaký pro **opravu** jedné chyby?

Kolik bude všech kódových slov?

Jaký bude rozměr generující a kontrolní matice?

Detekce:

- parita (sudá parita je lineární kód) … kvzd = 2

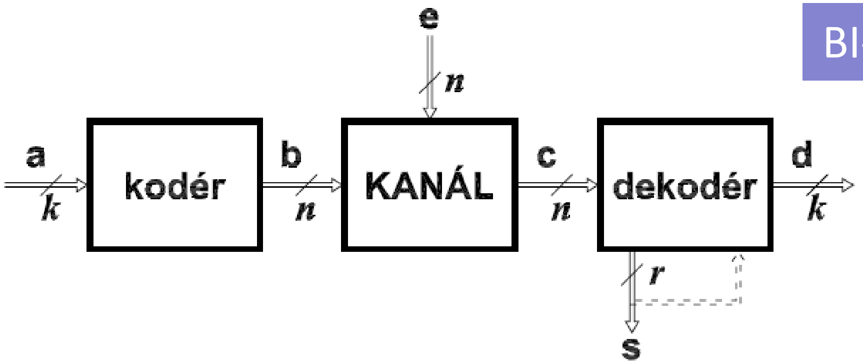
****

- Lichá parita? … kvzd = 2 … není lineární kód. Proč?

Parita: BI-LIN ++

BI-JPO

$$G_K = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$



všech 4 bitových slov .... 2^4
všech kódových slov .... 2^4

$H_K = [1\ \ 1\ \ 1\ \ 1\ \ 1\ ]$

Kód K je (5, 4) - kód

Kódování: slovo a = (1 0 1 1) zakódujeme na (1 0 1 1 1)

$$b_K = a \times G = (1\,0\,1\,1\,1)$$

Dekódování: $b_K \times H^T = (1\,0\,1\,1\,1) \times [1\ \ 1\ \ 1\ \ 1\ \ 1\ ]^T = 0$

s chybou (lichým počtem chyb):

$$c_K \times H^T = (1\,0\,1\,0\,1) \times [1\ \ 1\ \ 1\ \ 1\ \ 1\ ]^T = 1$$

Realizace parity

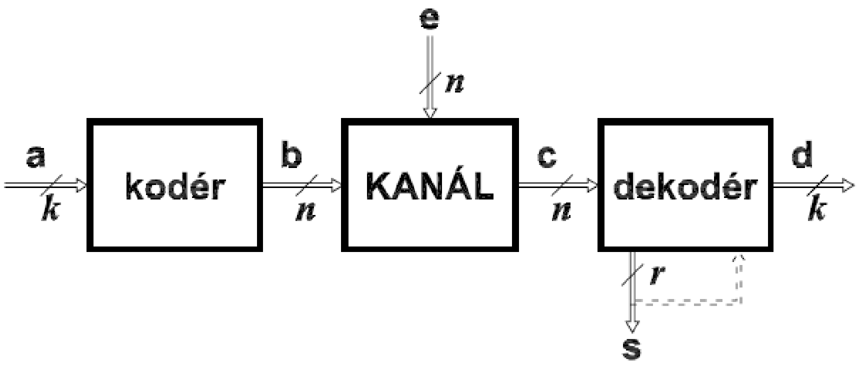
$b_1 = a_1$

$b_2 = a_2$

$b_3 = a_3$

$b_4 = a_4$

$b_5 = a_1 + a_2 + a_3 + a_4$



c je správně když : $c_1 + c_2 + c_3 + c_4 + c_5 = 0$ a

d jsou jen informační bity bez parity

..... implementace XORem .... ($\oplus \equiv +$)

Syndrom ***s*** závisí pouze na chybě:

$$\boldsymbol{s} = \boldsymbol{c} \cdot H^T = (\boldsymbol{b} + \boldsymbol{e}) \cdot H^T = \boldsymbol{0} + \boldsymbol{e} \cdot H^T = \boldsymbol{e} \cdot H^T$$

Pokračování příkladu: oprava

Oprava:

- Hammingův kód (lineární, samoopravný a perfektní kód)

Proč právě tyhle vlastnosti a co znamenají?

- lineární → existuje generující a kontrolní matice (++)
- samoopravný → kódová vzdálenost 3
- perfektní → má minimální redundanci

Oprava chyb (jedné chyby), Hamming

- Kódová vzdálenost (*kvzd* = 3), (*n*,*k*)-kód
- Lineární kód pro opravu jedné chyby s minimální redundancí → Hammingův kód

$k = 4$ (k řádků matice G), kolik bude n?

Tedy podle definice (slide 5) např. :

$$G_K = \begin{array}{c} \begin{array}{ccccccc} a_1 & a_2 & a_3 & a_4 & p_1 & p_2 & p_3 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \end{array}$$

Kódová slova jsou lineární kombinace řádků *G*. Množina kódových slov tvoří lineární (vektorový) prostor.

(7, 4)-kód. Proč?

Kódování a dekódování

Kódování: slovo a = (1 0 1 1) zakódujeme

$b_K = a \times G = (1\ 0\ 1\ 1\ 0\ 1\ 0\ )$

Dekódování: $b_K \times H^T = (1\ 0\ 1\ 1\ 0\ 1\ 0\ ) \times [H]^T = (0, 0, 0)^T$

$$H = \begin{array}{ccccccc} a_1 & a_2 & a_3 & a_4 & p_1 & p_2 & p_3 \end{array}$$
$$ \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

s chybou (nejen lichým počtem chyb):

$$c_K \times H^T \neq \underline{0}$$

$(1\ 0\ 1\ \mathbf{0}\ 0\ 1\ 0\ ) \times [H]^T = (1, 1, 1)^T$

Opravíme 4. sloupec
… negace chybového bitu

Realizace a úpravy

Sloupce matice H musí být nenulové a vzájemně různé, takže např:

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

Lze provádět lineární úpravy (systematický kód, viz slide 5)

$$\underline{\mathbf{H}}' = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

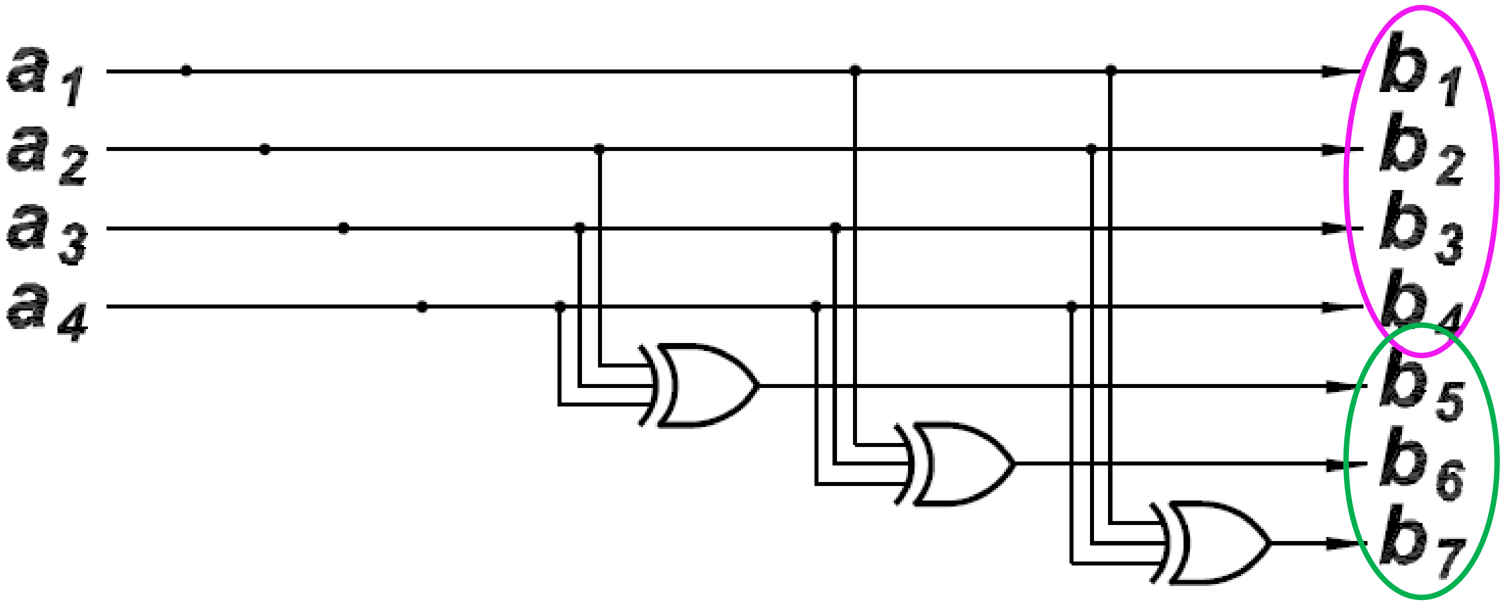
$$\underline{\mathbf{G}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Jednotkové matice, vzájemný převod → není ale nutný, stačí matice H

Kodér (7, 4)-Hammingova kódu

BI-JPO

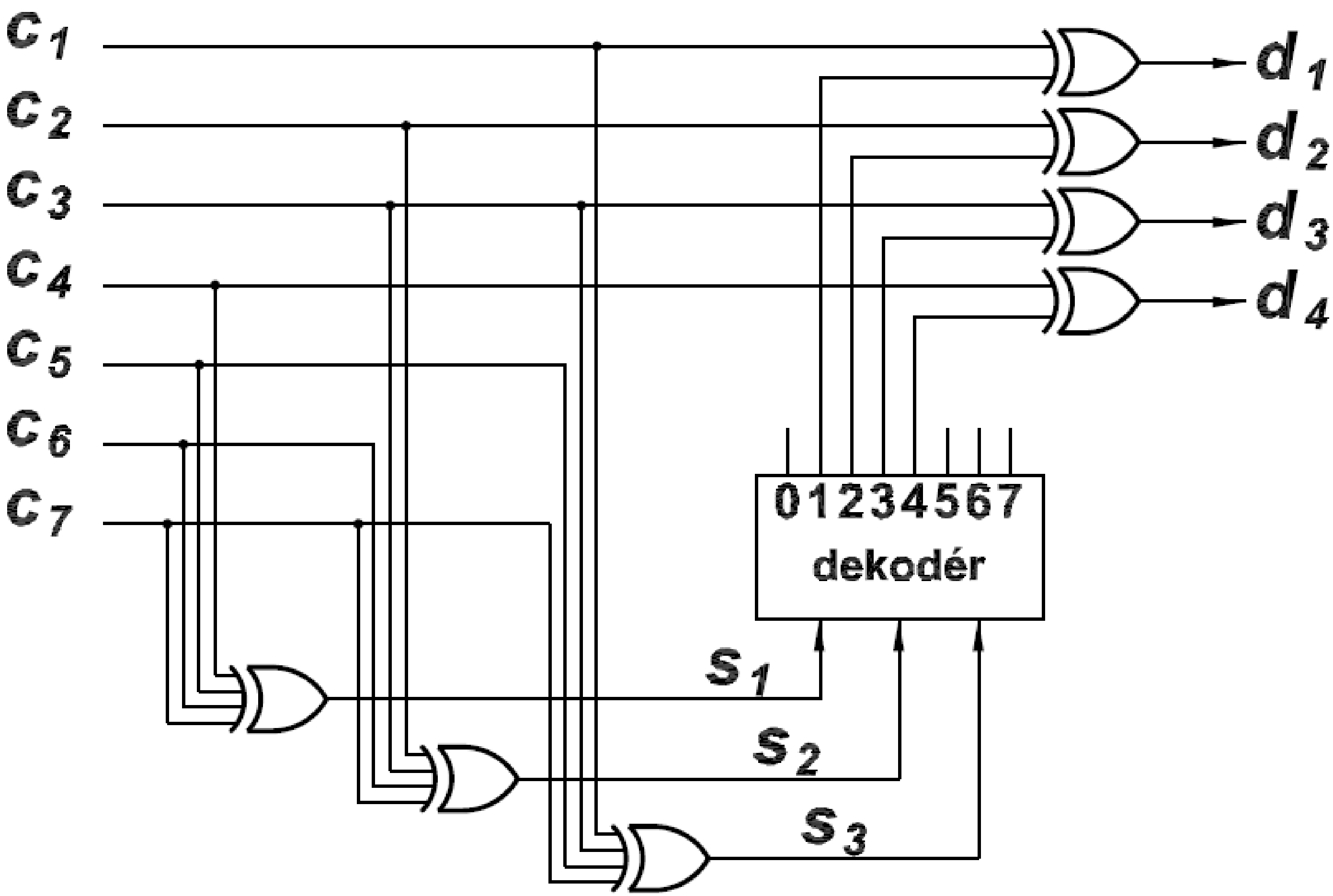
$$\begin{array}{c} \begin{array}{ccccccc} a_1 & a_2 & a_3 & a_4 & p_1 & p_2 & p_3 \end{array} \\ \underline{\mathbf{G}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \end{array}$$



Realizace dekodéru a opravy

BI-JPO

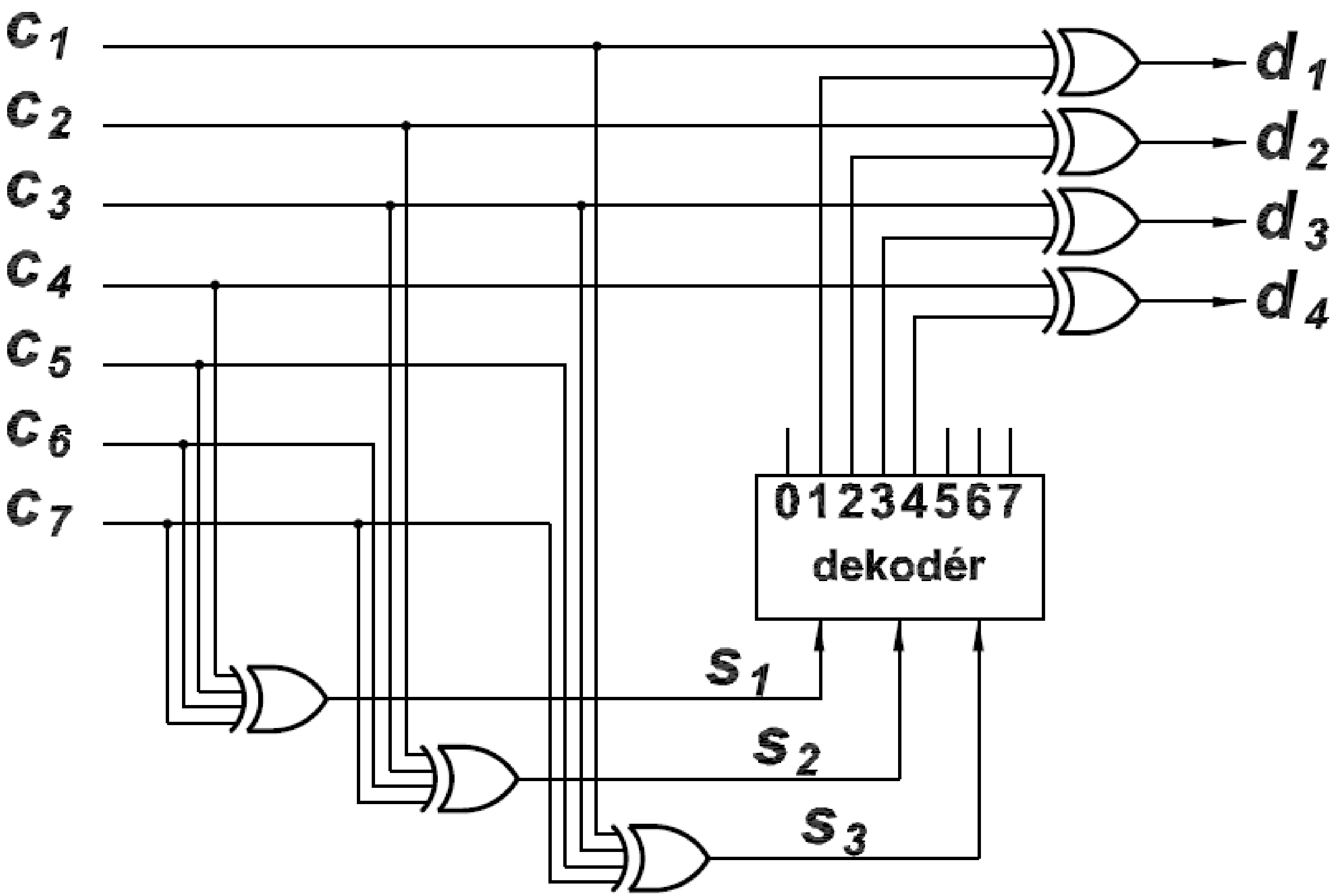
$$\underline{H} = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$



BI-JPO

$$\begin{array}{ccccccc} p3 & p2 & a1 & p1 & a2 & a3 & a4 \end{array}$$

$$\underline{\mathrm{H}} = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$



Realita, použití

- Hammingův kód má minimální redundanci jen když:

$$n = 2^{(n-k)} - 1$$

(*n* sloupců matice *H,* (*n-k*) řádků, *tzn.* kontrolních bitů)

	n	k
2	3	1
3	7	4
4	15	11
5	31	26
6	63	57
7	127	120
8	255	247
9	511	502
10	1023	1013
⋮	⋮	⋮

Pro zabezpečení standardního slova je třeba použít zkrácených kódů, ale s dodržením kódové vzdálenosti, např.:
(12, 8), (71, 64)