

Předmět KIV/TI - přednáška 10

# Systematické cyklické kódy

Ing. Václav Vais, Ph.D.

[vais@kiv.zcu.cz](mailto:vais@kiv.zcu.cz)

# Systematické cyklické kódy

- V dalším ukážeme, jak k nesystematickým cyklickým kódům vytvářet ekvivalentní systematické kódy.
- Systematické cyklické kódy vycházejí z jiné konstrukce informačního mnohočle-  
nu (prvky vektorů začínáme indexovat od 0 **zprava**):

(přiřazení mocnin členů polynomu informačním prvkům)

$$\begin{aligned} & \begin{matrix} k-1 & k-2 & & 1 & 0 \end{matrix} \\ \mathbf{u} &= [u_{k-1} \ u_{k-2} \ \dots \ u_1 \ u_0]^T \\ \approx \quad u(x) &= u_{k-1} \cdot x^{k-1} + u_{k-2} \cdot x^{k-2} + \dots + u_1 \cdot x + u_0 \end{aligned}$$

- Přiřazení mocnin  $x$  informačním prvkům je tedy opačné než u nesystematických cyklických kódů.

# Systematické cyklické kódy

- Takto vytvořený mnohočlen vynásobíme členem  $x^{n-k}$  :

$$u(x) \cdot x^{n-k} = u_{k-1} \cdot x^{n-1} + u_{k-2} \cdot x^{n-2} + \dots + u_1 \cdot x^{n-k+1} + u_0 \cdot x^{n-k}$$

- Násobení mnohočlenu členem  $x^{n-k}$  se ve „značkové“ reprezentaci projeví doplněním  $n - k$  nul zprava k informační části:

$$u(x) \cdot x^{n-k} \approx \begin{matrix} \text{(přiřazení mocnin členů informačním prvkům)} \\ \begin{matrix} n-1 & n-2 & & n-k & | & n-k-1 & & 0 \end{matrix} \\ \begin{bmatrix} u_{k-1} & u_{k-2} & \dots & \dots & u_1 & u_0 & | & 0 & 0 & 0 & \dots & \dots & 0 \end{bmatrix}^T \end{matrix}$$

# Systematické cyklické kódy

- Polynom  $u(x) \cdot x^{n-k}$  nyní vydělíme generujícím mnohočlenem  $g(x)$ .  
Výsledkem dělení budou dva mnohočleny - podíl  $q(x)$  a zbytek  $r(x)$

takový, že platí

$$u(x) \cdot x^{n-k} = q(x) \cdot g(x) + r(x)$$

přičemž stupeň zbytku  $r(x)$  je menší než stupeň dělitele  $g(x)$ , tj. nejvýše stupně  $n - k - 1$ . Podíl i zbytek jsou tímto vztahem určeny jednoznačně.

- Odečteme od obou stran rovnice  $r(x)$  (tj. přičteme k oběma stranám opačný prvek  $-r(x)$ ) :

$$u(x) \cdot x^{n-k} - r(x) = q(x) \cdot g(x)$$

# Systematické cyklické kódy

- Levá strana rovnosti  $u(x) \cdot x^{n-k} - r(x)$  je násobkem generujícího mnohočlenu  $g(x)$ , **je tedy mnohočlenem reprezentujícím značku cyklického kódu.**
- Grafické znázornění sečítání na levé straně rovnosti:

$u_{k-1}$	$u_{k-2}$	...	...	$u_1$	$u_0$	0	...	...	0	0
+						$-r_{n-k-1}$	...	...	$-r_1$	$-r_0$
<hr/>										
$u_{k-1}$	$u_{k-2}$	...	...	$u_1$	$u_0$	$-r_{n-k-1}$	...	...	$-r_1$	$-r_0$

# Systematické cyklické kódy

$u_{k-1}$	$u_{k-2}$	...	...	$u_1$	$u_0$	0	...	...	0	0
						+				
						$-r_{n-k-1}$	...	...	$-r_1$	$-r_0$
<hr style="border: 1px solid black;"/>										
$u_{k-1}$	$u_{k-2}$	...	...	$u_1$	$u_0$	$-r_{n-k-1}$	...	...	$-r_1$	$-r_0$

- Efektem tohoto sečítání je „složení“ kódové značky za dvou částí – informační a zabezpečovací, takto vytvořený kód je tedy systematický.
- Dodatečně je tak vysvětleno opačné přiřazení mocnin  $x$  prvkům informační části a násobení informačního mnohočlenu členem  $x^{n-k}$ .

# Systematické cyklické kódy

Rekapitulace postupu pro kódování v systematickém **binárním** cyklickém kódu

1. Informační části přiřadíme polynom  $u(x)$  (tak, že absolutní člen = **pravý** krajní prvek)
2. Vytvoříme součin  $u(x) \cdot x^{n-k}$
3. Vydělíme  $u(x) \cdot x^{n-k} : g(x)$  . Výsledek: podíl  $q(x)$  (k ničemu není) a zbytek  $r(x)$ .
4. Součet  $v(x) = u(x) \cdot x^{n-k} + r(x)$  představuje kódovou značku (tj. zakódovanou informační část).

(V tělese  $Z_2$  je prvek 1 opačným prvkem k sobě samému, proto  $r(x)$  přičítáme ).

# Systematické cyklické kódy

- Kontrola přijaté značky v systematickém binárním cyklickém kódu.
- Přijatá značka se kontroluje dělením generujícím mnohočlenem.
- Protože  $v(x) = u(x) \cdot x^{n-k} + r(x)$  je násobkem generujícího mnohočlenu, je zbytek po dělení  $v(x)$  mnohočlenem  $g(x)$  nulový, tedy

$$w(x) \in K \Leftrightarrow w(x) : g(x) = q(x) \text{ a } r(x) = 0$$

- Praktický důsledek: v komunikačních adaptérech, které implementují kódování a dekódování cyklických kódů hardwarově, se pro kódování i kontrolu přijaté značky používají stejné obvody.



# Dělení mnohočlenů nad tělesem $Z_2$

- Příklad dělení polynomů nad tělesem  $Z_2$  (  $1 + 1 = 0$ ,  $-1 = 1$  )

$$\begin{array}{r} (x^7 + x^6 + x^5 + x^3 + x^2 + 1) : (x^3 + x^2 + 1) = x^4 + x^2 + 1 \quad = \text{podíl } q(x) \\ \underline{-(x^7 + x^6 + x^4)} \\ x^5 + x^4 + x^3 + x^2 + 1 \\ \underline{-(x^5 + x^4 + x^2)} \\ x^3 \quad + 1 \\ \underline{-(x^3 + x^2 + 1)} \\ x^2 \quad = \text{zbytek } r(x) \end{array}$$

# Dělení mnohočlenů nad tělesem $\mathbb{Z}_2$

- Stejný výpočet můžeme provést přímo nad značkami:

$$\begin{array}{r} 11101101 : 1101 = 10101 \quad \text{reprezentuje podíl } q(x) = x^4 + x^2 + 1 \\ +1101 \\ \hline 00111101 \\ \phantom{0}0111101 \\ \phantom{00}+1101 \\ \phantom{000}\hline \phantom{000}001001 \\ \phantom{0000}01001 \\ \phantom{00000}+1101 \\ \phantom{000000}\hline \phantom{000000}0100 \quad \text{reprezentuje zbytek } r(x) = x^2 \end{array}$$

# Cyklické kódy - příklad

Cyklické kódy s generujícím mnohočlenem  $g(x) = x^3 + x + 1$

V nesystematickém cyklickém kódu s generujícím mnohočlenem  $g(x)$  zakódujeme informační část  $\mathbf{u} = [1\ 0\ 1\ 1]^T$  a to jak pomocí generující matice, tak i pomocí generujícího mnohočlenu.

$$\mathbf{G} = \begin{bmatrix} 1101000 \\ 0110100 \\ 0011010 \\ 0001101 \end{bmatrix}$$

$$\mathbf{v} = \mathbf{G}^T \cdot \mathbf{u} = \begin{bmatrix} 1101000 \\ 0110100 \\ 0011010 \\ 0001101 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{1} \\ 0 \\ \mathbf{1} \\ \mathbf{1} \end{bmatrix} = \begin{array}{r} 1101000 \\ 0110100 \\ 0011010 \\ 0001101 \\ \hline \mathbf{[1111111]}^T \end{array}$$

# Cyklické kódy - příklad

Cyklické kódy s generujícím mnohočlenem  $g(x) = x^3 + x + 1$

V nesystematickém cyklickém kódu s generujícím mnohočlenem  $g(x)$  zakódujeme informační část  $\mathbf{u} = [1\ 0\ 1\ 1]^T$  a to jak pomocí generující matice, tak i pomocí generujícího mnohočlenu.

$$u(x) = 1 + x^2 + x^3$$

$$\begin{aligned} v(x) &= u(x) \cdot g(x) = (1 + x^2 + x^3) \cdot (1 + x + x^3) = (1 + x + x^3) + (x^2 + x^3 + x^5) + \\ &\quad + (x^3 + x^4 + x^6) = 1 + x + x^2 + x^3 + x^3 + x^3 + x^4 + x^5 + x^6 = \mathbf{1 + x + x^2 + x^3 + x^4 + x^5 + x^6} \end{aligned}$$

# Cyklické kódy - příklad

Cyklické kódy s generujícím mnohočlenem  $g(x) = x^3 + x + 1$

V systematickém cyklickém kódu s generujícím mnohočlenem  $g(x)$  zakódujeme informační část  $\mathbf{u} = [1\ 1\ 0\ 0]^T$ .

$$u(x) = x^3 + x^2 \qquad u(x) \cdot x^{n-k} = (x^3 + x^2) \cdot x^3 = x^6 + x^5$$

$$u(x) \cdot x^{n-k} : g(x) = (x^6 + x^5) : (x^3 + x + 1) = x^3 + x^2 + x$$

$$\begin{array}{r} \underline{-(x^6 + x^4 + x^3)} \\ x^5 + x^4 + x^3 \\ \underline{-(x^5 + x^3 + x^2)} \\ x^4 + x^3 \\ \underline{-(x^4 + x^3 + x)} \end{array}$$

$$x = \text{zbytek } r(x)$$

# Cyklické kódy - příklad

Cyklické kódy s generujícím mnohočlenem  $g(x) = x^3 + x + 1$

V systematickém cyklickém kódu s generujícím mnohočlenem  $g(x)$  zakódujeme informační část  $\mathbf{u} = [1\ 1\ 0\ 0]^T$ .

$$v(x) = u(x) \cdot x^{n-k} + r(x) = x^6 + x^5 + x$$

Informační části  $\mathbf{u} = [1\ 1\ 0\ 0]^T$  tedy přísluší kódová značka  $\mathbf{v} = [1\ 1\ 0\ 0\ 0\ 1\ 0]^T$ .

Cyklické kódy s  $g(x) = x^3 + x + 1$ .

	Informační				Nesystematické kódování				Systematické kódování						
č.	část u				kódová značka v				kódová značka v						
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	0	0	0	1	0	0	0	1	1	0	1	0	0	1	1
2	0	0	1	0	0	0	1	1	0	1	0	0	1	1	0
3	0	0	1	1	0	0	1	0	1	1	1	0	1	0	1
4	0	1	0	0	0	1	1	0	1	0	0	0	1	1	1
5	0	1	0	1	0	1	1	1	0	0	1	0	1	1	0
6	0	1	1	0	0	1	0	1	1	1	1	0	0	0	1
7	0	1	1	1	0	1	0	0	0	1	1	0	1	0	0
8	1	0	0	0	1	1	0	1	0	0	0	1	0	0	1
9	1	0	0	1	1	1	0	0	1	0	1	1	1	1	0
10	1	0	1	0	1	1	1	0	0	1	0	1	0	0	1
11	1	0	1	1	1	1	1	1	1	1	1	1	0	0	0
12	1	1	0	0	1	0	1	1	1	0	0	1	0	0	0
13	1	1	0	1	1	0	1	0	0	0	1	1	0	0	1
14	1	1	1	0	1	0	0	0	1	1	0	1	0	0	0
15	1	1	1	1	1	0	0	1	0	1	1	1	1	1	1

# Cyklické kódy s $g(x) = x^3 + x + 1$ .

- Z množiny značek systematického kódu vybereme značky do generující matice v systematickém tvaru :

$$[1\ 0\ 0\ 0\ 1\ 0\ 1]^T, \quad [0\ 1\ 0\ 0\ 1\ 1\ 1]^T, \quad [0\ 0\ 1\ 0\ 1\ 1\ 0]^T, \quad [0\ 0\ 0\ 1\ 0\ 1\ 1]^T$$

$$\mathbf{G} = \begin{bmatrix} 1000 & 101 \\ 0100 & 111 \\ 0010 & 110 \\ 0001 & 011 \end{bmatrix}, \quad \text{tedy} \quad \mathbf{H} = [-\mathbf{B}^T \mid \mathbf{I}_{n-k}] = \begin{bmatrix} 1110 & 100 \\ 0111 & 010 \\ 1101 & 001 \end{bmatrix}$$

- Je to cyklický Hammingův kód (7, 4).



# Důsledek cykličnosti

- Cyklické kódy umožňují detekovat shluky chyb.
- *Shlukovou chybou délky  $b$*  rozumíme takové chybové slovo  $e$  , jehož všechny chybové prvky (u binárních kódů jedničky) leží v úseku ohraničeném indexy  $i$  a  $i + b - 1$  , přičemž krajní prvky shluku jsou nenulové (u binárních kódů jedničky).
- **Cyklické kódy detekují všechny shluky chyb délky  $b \leq n - k$  .**
- Ilustrace – detekce shlukových chyb délky  $\leq 3$  cyklickým Hammingovým kódem (7,4).

Návrat k příkladu s  $g(x) = x^3 + x + 1$ .

$$\mathbf{G} = \begin{bmatrix} 1000 & 101 \\ 0100 & 111 \\ 0010 & 110 \\ 0001 & 011 \end{bmatrix}, \quad \mathbf{H} = [-\mathbf{B}^T \mid \mathbf{I}_{n-k}] = \begin{bmatrix} 1110 & 100 \\ 0111 & 010 \\ 1101 & 001 \end{bmatrix}$$

- Spočítáme syndromy pro tři vybraná chybová slova (shluky délky 2 a 3):

$$\begin{aligned} \mathbf{e}_1 &= [1100000]^T & \mathbf{s}_1 &= \mathbf{H} \cdot \mathbf{e}_1 = [010]^T \\ \mathbf{e}_2 &= [1010000]^T & \mathbf{s}_2 &= \mathbf{H} \cdot \mathbf{e}_2 = [011]^T \\ \mathbf{e}_3 &= [1110000]^T & \mathbf{s}_3 &= \mathbf{H} \cdot \mathbf{e}_3 = [100]^T \end{aligned}$$

Návrat k příkladu s  $g(x) = x^3 + x + 1$ .

$$\begin{array}{ll} e_1 = [1100000]^T & s_1 = H \cdot e_1 = [010]^T \\ e_2 = [1010000]^T & s_2 = H \cdot e_2 = [011]^T \\ e_3 = [1110000]^T & s_3 = H \cdot e_3 = [100]^T \end{array}$$

- Tyto chybové vektory jsou „vzorce“ shluku délky 2 a všech shluků délky 3.
- Všechny tyto chybové vektory generují nenulový syndrom  $\Rightarrow$  kód je detekuje.
- Jestliže je  $w$  nekódovým slovem, je nekódovým slovem i jeho libovolný cyklický posuv (snadno lze dokázat sporem).
- Kód tedy detekuje nejen výše uvedené chybové vektory, ale i jejich libovolné cyklické posuvy  $\Rightarrow$  kód detekuje všechny shluky chyb délky  $\leq 3$ .

# Praktické použití systematických binárních cyklických kódů

- Systematické cyklické kódy (CRC – Cyclic Redundance Check) se nepoužívají k opravám chyb; ale k detekci (100% shlukových chyb délky  $b \leq n - k$  ).
- Standardní reprezentace generujících mnohočlenů hexadecimálním číslem:

$x^{16}$		+		$x^{12}$				+		$x^5$				+		1
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1
1				0				2				1				

- Existuje i alternativní způsob (Koopman) - nezobrazuje jedničku u absolutního členu, rámeček je pak posunut o jednu pozici doleva a výsledek je pak 0x**8810** .

# Nejpoužívanější standardizované cyklické kódy

Název	generující mnohočlen	standardní reprezentace	použití (standarty)
CRC-8	$x^8 + x^2 + x + 1$	0x07	vysokorychlostní protokol ATM (záhlaví)
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x + 1$		vysokorychlostní protokol ATM (vrstva AAL)
CRC-12	$x^{12} + x^{11} + x^3 + x^2 + x + 1$		Telekomunikační systémy
CRC-16	$x^{16} + x^{15} + x^2 + 1$	0x8005	„firemní“ protokol IBM
CCITT-16	$x^{16} + x^{12} + x^5 + 1$	0x1021	HDLC, X.25, V41, IEEE 802, V.42, AAL 5
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$	0x04C11DB7	IEE 802.3, Ethernet

# Praktické použití systematických binárních cyklických kódů

- V komunikačních adaptérech jsou kódovací a dekódovací postupy realizovány hardwarově, ve vyšších vrstvách architektury podle modelu ISO/OSI softwarově.
- Bloky dat zabezpečované CRC nemusí mít stejnou délku.
- Dělením je datový blok doplněný na sudou délku následovaný  $n - k$  nulami.
- Každý CRC mnohočlen má standardem definovanou maximální délku bloku dat, kterou může zabezpečit.
- Prakticky používané generující mnohočleny stupně  $n - k$  :
  - primitivní mnohočlen stupně  $n - k$  nebo
  - součin primitivní mnohočlen stupně  $n - k - 1$   $\times (x + 1)$

# Praktické použití systematických binárních cyklických kódů

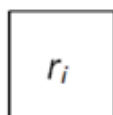
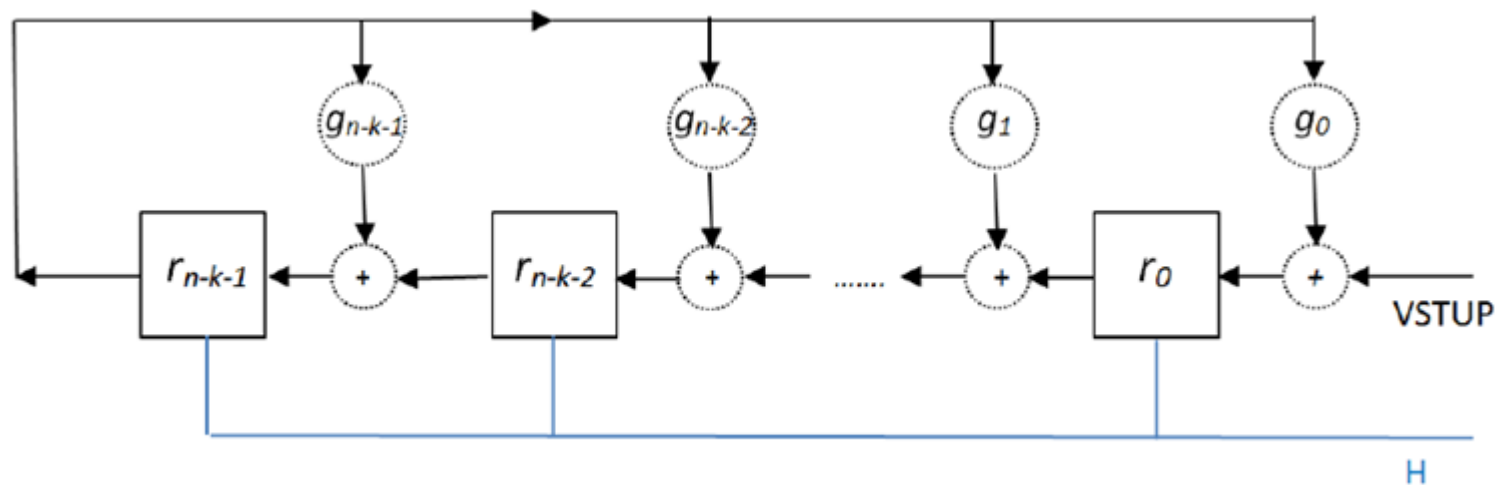
- primitivní mnohočlen stupně  $n - k$ 
  - maximální délka bloku dat  $2^{n-k} - 1$  bitů
  - detekuje všechny shluky délky  $\leq n - k$
  - detekuje všechny jednoduché a dvojité chyby
- součin primitivní mnohočlen stupně  $n - k - 1 \times (x + 1)$ 
  - maximální délka bloku dat (jen)  $2^{n-k-1} - 1$  bitů (ale navíc ještě)
  - detekuje všechny trojitě chyby
  - detekuje všechna chybová slova se sudým počtem jedniček
  - s p-stí  $1 - 2^{n-k-1}$  detekuje i náhodné shluky s délkou větší než  $n - k$

# Hardwarová realizace kodéru systematických binárních cyklických kódů

- „Dělení“ bloku dat generujícím mnohočlenem se provádí v *lineárním zpětnovazebním registru* (LFSR).
- Nastavení zpětných vazeb je určeno použitým generujícím mnohočlenem.
- Do LSFR vstupuje dělenec; při kódování informační blok délky  $k$  následovaný  $n - k$  nulami, při kontrole přijaté značky celý přijatý blok dat délky  $n$ .
- S každým hodinovým pulzem dojde ke vstupu jednoho znaku a k zápisu do paměťových prvků.
- Po  $n$  pulzech je v paměťových prvcích  $r_{n-k-1}, r_{n-k-2}, \dots, r_1, r_0$  zapsán zbytek po dělení  $u(x) \cdot x^{n-k} : g(x)$ , respektive  $w(x) : g(x)$ .



# Obecné uspořádání LSFR pro dělení generujícím mnohočlenem $g(x)$



synchronní binární paměťový prvek typu D („buzení = příští stav“)

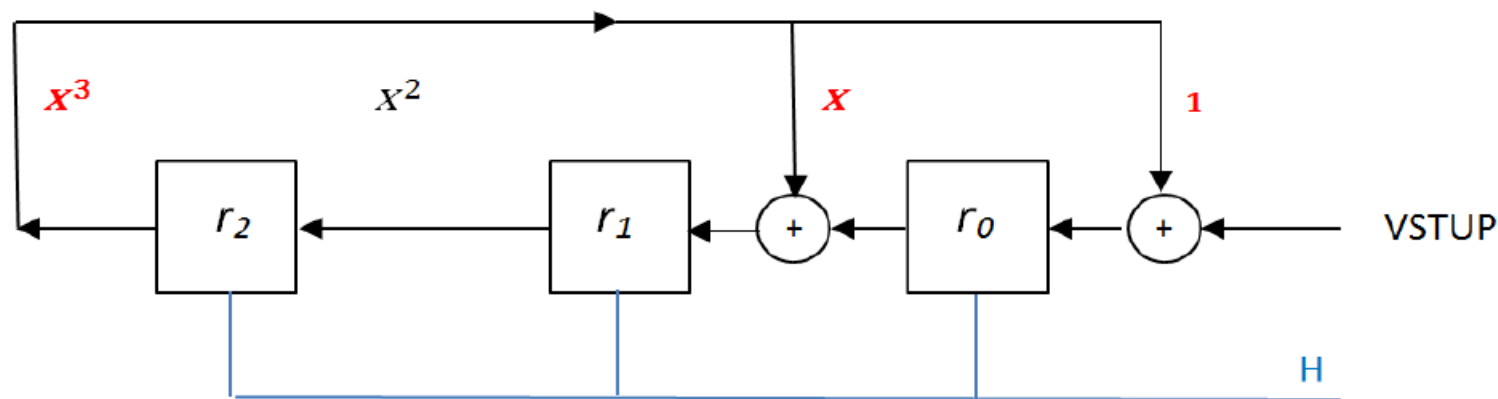


„Kvazipropojka“. Je-li koeficient  $g_i$  v generujícím mnohočlenu 1, je zpětná vazba přivedena na součtové hradlo před vstup příslušného paměťového členu. Je-li koeficient  $g_i$  roven 0, součtové hradlo před vstupem příslušného paměťového členu není.

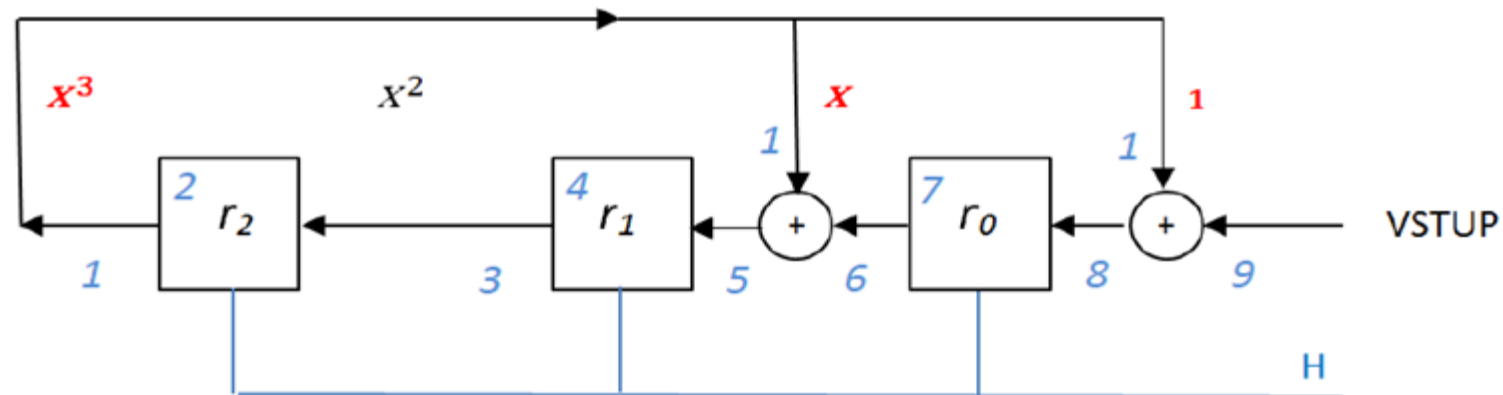


Součtové hradlo, realizuje sčítání definované v tělese  $Z_2$  (názvy v elektrotechnické terminologii - hradlo XOR, Exclusive OR, Nonekvivalence)

# Uspořádání LSFR pro dělení generujícím mnohočlenem $g(x) = x^3 + x + 1$



- Následující slajd = průběh zpracování konkrétního vstupního řetězce 1110000



	1	2	3	4	5	6	7	8	9		
	0	0	0	0	0	0	0	1	1	$u_3$	stav před prvním hodinovým pulzem
	0	0	0	0	1	1	1	1	1	$u_2$	stav po 1. pulzu
	0	0	1	1	1	1	1	1	1	$u_1$	stav po 2. pulzu
$q_3$	1	1	1	1	0	1	1	1	0	$u_0$	stav po 3. pulzu
$q_2$	1	1	0	0	0	1	1	1	0		stav po 4. pulzu
$q_1$	0	0	0	0	1	1	1	0	0		stav po 5. pulzu
$q_0$	0	0	1	1	0	0	0	0	0		stav po 6. pulzu
	1	1	0	0	1	0	0	1	0		stav po 7. pulzu
		$r_2$		$r_1$			$r_0$				

### Co říkají sloupce tabulky?

Sloupec 1 - výstup do zpětné vazby

Sloupec 2 – obsah prvku  $r_2$

Sloupec 3 – výstup z prvku  $r_1$

Sloupec 4 - obsah prvku  $r_1$

Sloupec 5 – výstup z hradla XOR (= vstup  $r_1$ )

Sloupec 6 – výstup z prvku  $r_0$

Sloupec 7 - obsah prvku  $r_0$

Sloupec 8 – výstup z hradla XOR (= vstup  $r_0$ )

Sloupec 9 – vstup

V tabulce jsou žlutě podbareveny cifry vstupujícího dělence, zeleně pak cifry reprezentující cifry podílu. Stavy paměťových prvků v jednotlivých taktech jsou podbarveny růžově. Je zřejmé, že je zapotřebí  $n - k$  hodinových pulsů, aby se jednička představující řád dělence dostala do paměťového prvku s nejvyšším indexem a následujících  $n - k - 1$  cifer dělence do dalších paměťových prvků. Teprve tehdy se začne projevovat vliv zpětné vazby (objeví se v ní jednička reprezentující řád podílu). Při každém dalším hodinovém pulzu je do zpětné vazby vyslána další cifra podílu. Po  $n$  pulzech jsou v paměťových prvcích zapsány koeficienty zbytku po dělení.

# Dělení mnohočlenem $g(x) = x^3 + x + 1$

- Je zřejmé, že zpracování vstupního bloku dat zpětnovazebním registrem odpovídá „ručnímu výpočtu“:

$$\begin{array}{r} 1110000 : 1011 = 1100 \quad \text{reprezentuje podíl} \quad q(x) = x^3 + x^2 \\ + 1011 \\ \hline 0101000 \\ + 1011 \\ \hline 000100 \\ 00100 \\ 0100 \quad \text{reprezentuje zbytek} \quad r(x) = x^2 \end{array}$$

- Zpětná vazba realizuje přičítání hodnoty (0 nebo 1) · 1011 k modifikovanému dělenci.

# Hardwarová realizace kodéru systematických binárních cyklických kódů

- Některé standardizované cyklické kódy uvedený základní postup ještě modifikují některým z těchto způsobů:
  - počátečním stavem paměťových prvků při HW realizaci nejsou nuly, ale jedničky
  - kontrolní znaky (zbytek po dělení) se před připojením za datový blok invertují (tj. negují „bit po bitu“)
  - před připojením za datový blok se provede reverze řetězce kontrolních znaků (tj. řetězec kontrolních znaků se připojí „pozpátku“)

# Princip softwarové implementace dělení generujícím mnohočlenem nad tělesem $\mathbb{Z}_2$

- Pro urychlení výpočtu se pro konkrétní generující mnohočlen předem vypočítá tabulka CRC\_table, která obsahuje zbytky po dělení všech dělenců od 0 do 255 v nejvyšším bytu slova (popisuje to následující pseudokód v jazyku C):

```
for (dividend = 0; dividend < 256; ++dividend) {
    remainder = dividend << (WIDTH-8); /*posunutí dělence*/
    for (bit = 8; bit > 0; --bit) {
        if (remainder & TOPBIT) { /* je-li v nejvyšším bitu 1 */
            remainder = (remainder << 1) ^ POLYNOMIAL; /*posuň, přičti g(x)*/
        }
        else {
            remainder = (remainder << 1); /*posuň rámeček*/
        }
    }
    crcTable[dividend] = remainder; /*zapiš zbytek do CRC_table*/
}
```

# Princip softwarové implementace dělení generujícím mnohočlenem nad tělesem $Z_2$

- Jednotlivé byty z datového bloku se pak používají jako indexy do CRC\_table, a zbytky z CRC\_table se „průběžně XORují“ datovými byty takto:

```
for (byte = 0; byte < nBytes; ++byte) {  
    data = message[byte]^(remainder >> (WIDTH - 8));  
    remainder = crcTable[data] ^ (remainder << 8);  
}
```

- Užitečný zdroj:

Barr, M.: CRC Implementation Code in C/C++

<https://barrgroup.com/Embedded-Systems/How-To/CRC-Calculatation-C-Code>



# Dělení mnohočlenů nad tělesy $\mathbb{Z}_p$ příklad $\mathbb{Z}_5$

Sečítání

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

Násobení

.	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

Opačné prvky

0	1	2	3	4
0	4	3	2	1

Inverzní prvky

1	2	3	4
1	3	2	4

# Dělení mnohočlenů nad tělesy $Z_p$ příklad $Z_5$

$$\begin{array}{r} (4x^5 + 3x^4 + 2x^3 + 3x + 2) : (3x^4 + 2x^3 + 4x^2 + 3) = 3x + 4 = \text{podíl } q(x) \\ \underline{-(4x^5 + x^4 + 2x^3 + 4x)} \\ 2x^4 + 4x + 2 \\ \underline{-(2x^4 + 3x^3 + x^2 + 2)} \\ 2x^3 + 4x^2 + 4x \end{array} = \text{zbytek } r(x)$$

Koeficienty **3** a **4** u členů podílu vzniknou dělením členů  $4x^5 : 3x^4$ , respektive  $2x^4 : 3x^4$ , tedy

$$\begin{aligned} 4x^5 : 3x^4 &= (4 : 3) x = (4 \cdot 3^{-1}) x = (4 \cdot 2) x = 3x, \text{ respektive} \\ 2x^4 : 3x^4 &= (2 : 3) = (2 \cdot 3^{-1}) = (2 \cdot 2) = 4 \end{aligned}$$

# Dělení mnohočlenů nad tělesy $Z_p$ příklad $Z_5$

- Stejný výpočet můžeme provést přímo nad značkami:

$$\begin{array}{r} 432032 : 32403 = 34 \quad \text{reprezentuje podíl } q(x) = 3x + 4 \\ \underline{-41204} \\ 020042 \\ \underline{-23102} \\ 02440 \end{array} \quad \text{reprezentuje zbytek } r(x) = 2x^3 + 4x^2 + 4x$$