Kódy a kódování

(opakování – zabýváme se jen kódy pro detekci/opravu chyb)

kanál – bez paměti = nezávislé chyby, s pamětí = shluky chyb

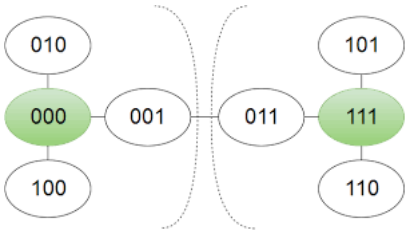
kanál – symetrický (stejná p-podobnost chyb tam a zpět), nesymetrický, jednosměrný

kód – systematický (oddělená informační část a kontrolní část), nesystematický

blokový kód (n,k) – n=celkový počet bitů, k=informašní obsah, r=n-k=redundance (zabezpečení)

kód – detekční (umožnuje detekovat chybu), opravný (umožnuje opravit chybu)

hamingova vzdálenost – min.počet bit.změn mezi dvěmi zprávami



Jednoduché kódy

Opakovací kód (n,1) – informace je obsařena n* za sebou, ham.vzdal=n

Kokatavý kód (nk,k) – informace je po k bitech n* za sebou, ham.vzdal=n

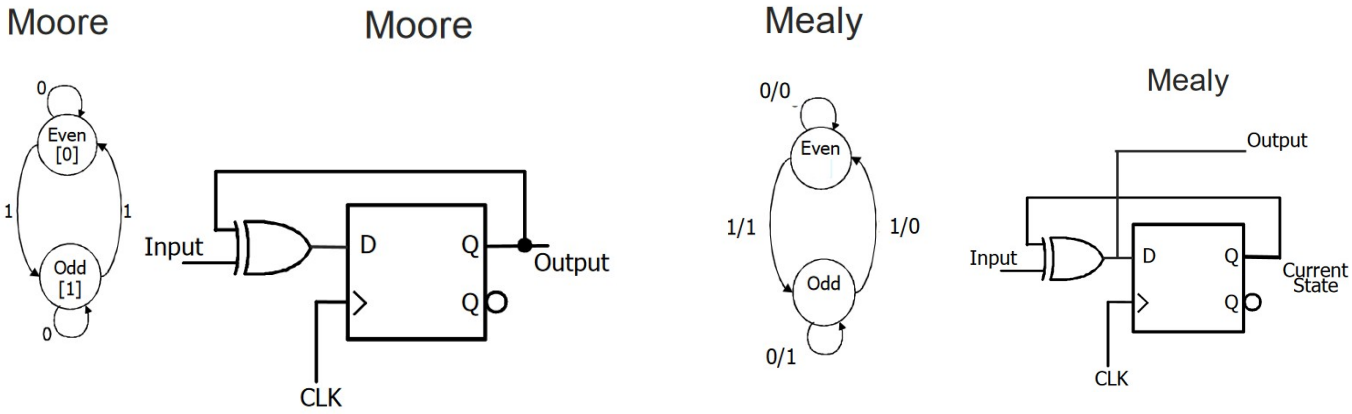
Parita (k+1,k) – bity+parita

- lichá – p=a0 xor a1 xor …. ak

- sudá - – p=a0 xor a1 xor …. ak xor 1

ham.vzdal = 2 , tj. Detekce pouze jedné chyby (bez možnosti opravy)

(detekuje také všchny ostatní liché chyby, naopak všechny sudé nedetekuje)



Příčná a podélná parita

$$\begin{array}{cc|c} a_1 & a_2 & p_1 \\ a_3 & a_4 & p_2 \\ \hline p_3 & p_4 & p_5 \end{array}$$

ham.vzdálenost = 4, tj. Detekce az tří chyb, možná oprava jedné

(též používáno bez podélněpříčné parity (p5) – pak ham.vzdal. = 3)

uvedený příklad kód (9,4), v obecnosti lib.veliká tabulka.

Kontrolní součet (checksum)

- přetížený název checksum ve smyslu součtu vs checksum jako libovolný druh zabezpečení

- suma bytů, snadná HW i SW implementace

- nevyhody: nedetekuje chybu v pořadí, nesensitivní na 0, pro 8bit rel.slabe

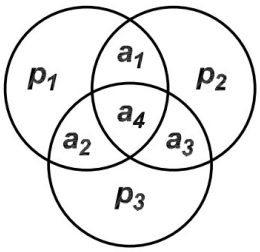
Fletcher-checksum

- zvetšuje délku zab.části detekuje chybu v poradi. Pro detekci používá 2 byte a sčítá v modulo 255

- algoritmus (vystupem je sum1 a sum2),:

```
sum1 = (sum1 + data[index]) % 255;
sum2 = (sum2 + sum1) % 255;
```

Hammingův kód



Hammingův kód patří do blokových lineárních cyklických Bose-Chaudhury-Hochquengen kódů.

Platí že pro každé r>2 existuje kód délky 2^r-1 se zprávou délky 2^r-1-r. (r=redundance)

b=a.G (b=zpráva včetně zabezpečení, a=zpráva/informace, G=generující matice)
G.H‘ = 0 (H=kontrolní matice) b.H‘ = a.G.H’ = 0

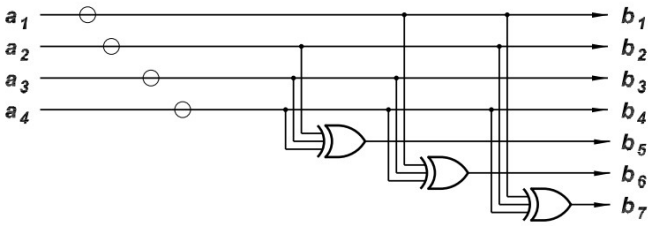
je li v přenosu chyba pak c=b+e (e=chybový vektor)

s = c.H‘ = b.H’+e.H’ = e.H’ (tj. Syndom s je závislý jen na chybovém vektoru)

Příklad (7,4): b=(a1,a3,a3,a4,p1,p2,p3) kde p1=a1^a2^a4, p2=a1^a2^a4, p3=a2^a3^a4

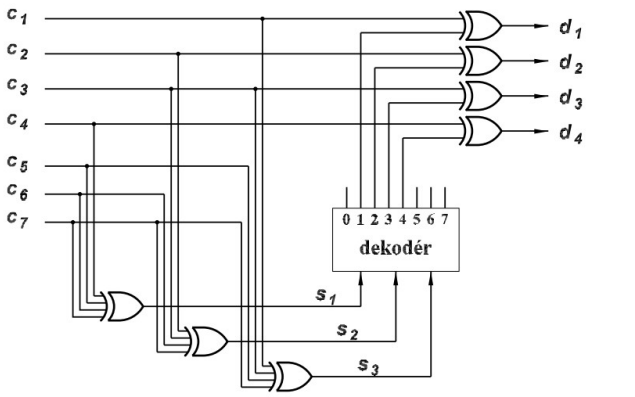
ham.vzdal = 3 (detekce dvou, oprva max.1 chyby)

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



dekodér

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



pozn k dekodéru: stav 0=bez chyby, další stavy „opravují“ paritní bity (tj. není třeba imlementovat)

Hammingův kód je systemtický, „perfektní“ kód SEC (single error correction), tj. Neexistuje žádný syndrom který nepřísluší jedné nebo žádné chybě (tj také. kodová vzdálenost všech kódu je 3)

jsou to kódy např. (3,1) (7,4) (15,11) (31,26) … (2^r-1, 2^r-1-r)

ale i také (12,8) (21,16) (38,32) …

hammingův kód lze rozšířit o paritu čímž získáme ham.vzdál = 4 (=rozšířený ham.kód)

ECC pamětí

ECC = error correction code – tj. (nespecifikovaný) kód pro korekci chyb v paměťových systémech

(pozn. EDC = error detection code)

- typicky se používá pro DRAM a (NAND) FLASH

- pro NAND flash důležitost vzrůstá (technologie SLC,MLC,TLC,QLC,PLC – 1,2,...,5bitů/1tranzistor)

SLC se užívá zabezpečení 1-12b (hamming), MLC 4-40b, TLC >60b (BCH kódy,LDPC) (ale i jiné)

Hammingův kód - škálovatelný přístup:

- Bitové pozice, jejichž číslo je rovné mocnině 2, jsou použity pro paritní bit (1, 2, 4, 8, 16, 32, …).

- Ostatní bitové pozice náleží kódovanému inf. slovu (3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, …).

- Každý paritní bit je vypočítán z některých bitů informačního slova. Pozice paritního bitu udává sekvenci bitů, které jsou v kódovém slově zjišťovány a které přeskočeny.

tj. Pro paritní bit p1 (pozice 1) se ve zbylém kódovém slově 1 bit přeskočí, 1 zkontroluje, 1 bit přeskočí, 1 zkontroluje, atd.

Pro paritní bit p2 (pozice 2) se přeskočí první bit, 2 zkontrolují, 2 přeskočí, 2 zkontrolují, atd.

Pro p3 (pozice 4) se přeskočí první 3 bity, 4 zkontrolují, 4 přeskočí, 4 zkontrolují, atd.

Pro kód (7 , 4) platí b = (p1 , p2 , a1 , p3 , a2 , a3 , a4)

např.

Pro DRAM 64bit se pužívá 8b zabezpečení

Pro Flash blok 2048bitů (512B) se popužívá EEC 24b (2b detection, 1b correction) (viz tab a obr)

Assignment of Data Bits With ECC Code

ECC	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Ecc0[1]	LP07	LP06	LP05	LP04	LP03	LP02	LP01	LP00
Ecc1[2]	LP15	LP14	LP13	LP12	LP11	LP10	LP09	LP08
Ecc2[3]	CP5	CP4	CP3	CP2	CP1	CP0	LP17	LP16

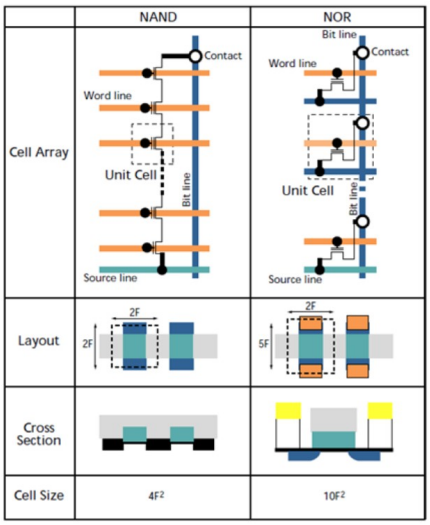
Notes:
1. The first byte (Ecc0) contains line parity bits LP0–LP07.
2. The second byte (Ecc1) contains line parity bits LP08–LP15.
3. The third byte (Ecc2) contains column parity bits CP0–CP5, plus LP16 and LP17 generated only for a 512-byte input.

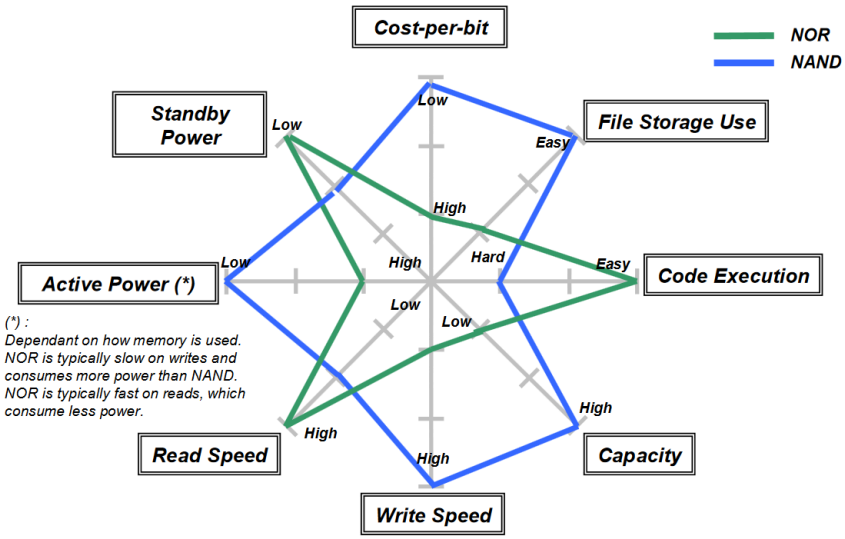
Parity Generation

Byte 0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	LPO	LPO2	… LP14[2]
Byte 1	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	LP1		
Byte 2	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	LPO	LPO3	
Byte 3	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	LP1		
…	…	…	…	…	…	…	…	…			
Byte 252	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	LPO	LPO2	… LP15[2]
Byte 253	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	LP1		
Byte 254	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	LPO	LPO3	
Byte 255[1]	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	LP1		
	CP1	CP0	CP1	CP0	CP1	CP0	CP1	CP0			
	CP3		CP2		CP3		CP2				
	CP5				CP4						

Flash paměti, SSD

Flash – NOR vs NAND





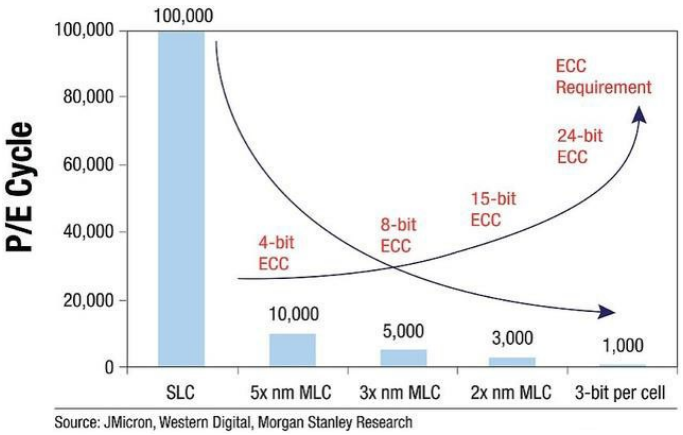
NAND (SSD) – multilevel cell, porovnání.

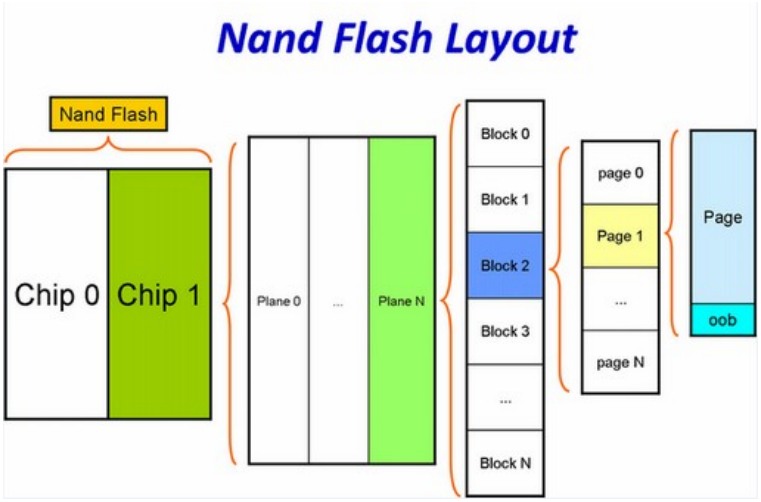
Erase (block). Read/Write(!)/Error-correction (page,”sector”(!)). Wear-leveling.

Např. 256MB SLC, 128kB block. 2048B page (+64B ECC), 8bit error repair), 512B „sector“.

	SLC	MLC	TLC
Bits per Cell	1	2	3
P/E Cycles (2Xnm)	100,000	3,000	1,000
Read Time	25us	50us	~75us
Program Time	200-300us	600-900us	~900-1350us
Erase Time	1.5-2ms	3ms	~4.5ms
Typical retention	5 years	1 year	3-12 months
Typical Endurance	10,000 cycles	3000 cycles	500 cycles

Table 1: A comparison of the different classes of flash memory.





CRC kódy

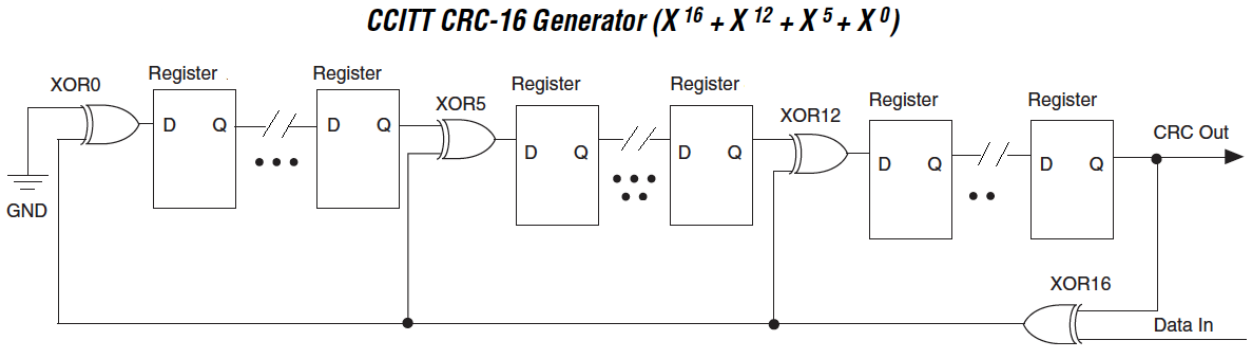
CRC = cyclic redyndancy check

patří do třídy cyklických polynomiálních kódů

typické parametry - detekce všech jednobitových a dvoubitových chyb (má li polynom více než tři členy), detekce všech lichých chyb (má li polynom členy X+1), detekce všech burst chyb délky <r (redundance), 1-1/(2^r) ostatních burst chyb.

Obvykle jen detekce chyb, ne oprava (je složitá)

velmi rychlé HW, ke kódování dekódování používáme posuvné registry. Stejný obvod pro vysílání i příjem.



SW implementace pomalejší, možno použít loop-up tabulky (postup po více bitech)

Implementace ještš závisí na startovní hodnotě CRC (nemusí být 0), na přadí vstupních bitů, na počadí výstupních bitů č případné negaci/xorování výsledku.

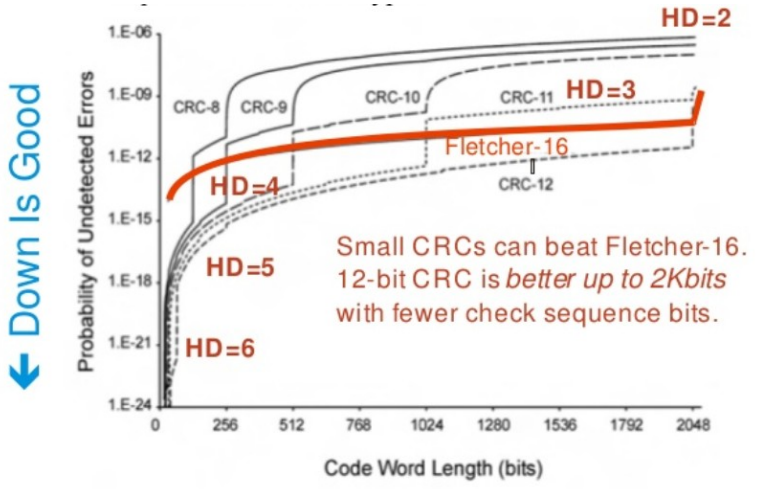
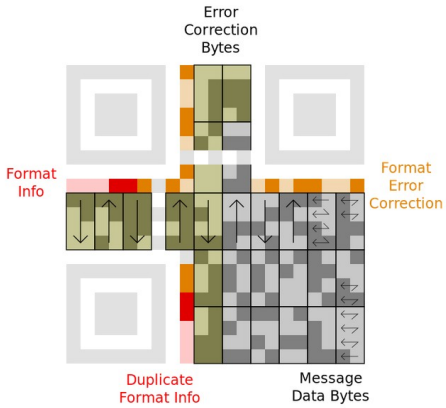


Fig. 12. Probability of undetected errors for Fletcher-16 and CRC bounds for different CRC widths at a BER of 10^{-5}. Data values for Fletcher-16 are the mean of 10 trials using random data.

Reed-Solomonovy kódy, BCH kódy



Široce používané – CD, QR kódy, bezdrát.komunikace

patří do skopiny blokových (nebinárních) lineárních cyklických Bose-Chaudhury-Hochquengen (BCH) kódů.

Je optimálním kódem ve smyslu že minimální distance je maximálně možně dosažitelná na velikosti (n,k). (n=celk.počet bitů, k=počet informačních bitů) – tj. Ham.vzdálenost je n-k+1

Typicky se velikost bloku volí 2^M-1 (např.255). Počet datových symbolů (obvykle se staví na nad vícebitovými bloky, typ.8bit=symbol) k je volitelný. Pro blok 255 se často používá např. Volba k=223, tj. 32 zabezpečovacích bitů/bytů – a je schopná opravit chyby do 16 bitů/bytů.

Jsou vhodné pro zebezpečení pro aplikace kde chyby přicházejí v „burstech“. Je li charakter chyb náhodný jsou vhodnější kódování.