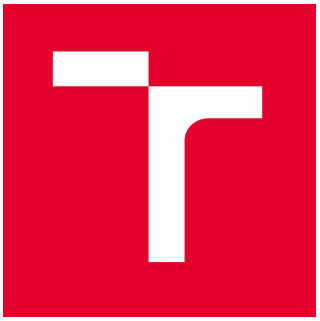


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV MIKROELEKTRONIKY

DEPARTMENT OF MICROELECTRONICS

OCHRANA DAT V PAMĚTI SRAM POMOCÍ SAMOOPRAVNÝCH KÓDŮ

DATA PROTECTION IN SRAM TYPE MEMORY WITH ERROR CORRECTION CODE

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — **Jakub Záhora**

AUTHOR

VEDOUCÍ PRÁCE — **Ing. Vojtěch Dvořák**

SUPERVISOR

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Mikroelektronika a technologie**

Ústav mikroelektroniky

Student: Jakub Záhora ***ID:*** 221288

Ročník: 3 ***Akademický rok:*** 2021/22

NÁZEV TÉMATU:

Ochrana dat v paměti SRAM pomocí samoopravných kódů

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se se známými typy samoopravných kódů (ECC) a jejich vlastnostmi. Vyberte takový kód, který umožňuje detekci dvojnásobné chyby a opravu jednonásobné chyby v každém slovu. V rámci bakalářské práce navrhněte kodér a dekodér v jazyce VHDL, u kterého bude možné nastavit šířku vstupního slova. Kodér i dekodér doplňte další logikou pro ovládání přístupu k paměti SRAM a automatickou opravou chyb v paměti. Výsledný kód ověřte pomocí simulace a proveďte syntézu do zvoleného obvodu FPGA pro určení maximální pracovní frekvence a spotřeby zdrojů.

DOPORUČENÁ LITERATURA:

Podle pokynů vedoucího práce

Termín zadání: 7.2.2022 ***Termín odevzdání:*** 2.6.2022

Vedoucí práce: Ing. Vojtěch Dvořák

doc. Ing. Jiří Háze, Ph.D.
předseda rady studijního programu

Abstrakt

Práca obsahuje základné pojmy potrebné pre pochopenie teórie kódovania. Následne vysvetľuje parametre, ktoré budú u opísaných kódov sledované. Na základe kritérií vyberá BCH kód a Rozšírený Hammingov Kód, za vhodné a ďalej porovnáva ich vlastnosti pre výber efektívnejšieho kódu z tejto dvojice. Výsledkom porovnania vyberá ako vhodnejší Rozšírený Hammingov Kód pre následný návrh kódu. Následne je zobrazený postup návrhu kódu, testy, ktoré overujú funkčnosť kódu a využité zdroje pre implementáciu kódu pre obvod Spartan-3.

Klíčová slova

Samo opravný kód, detekčný kód, korekčný kód, Cyklický kód, Rozšírený Hammingov Kód, BCH kód, zhluková chyba

Abstract

The thesis contains basic concepts necessary for understanding the theory of coding. It then explains the parameters that will be monitored for the described codes. Based on the criteria, it selects the BCH code and the Extended Hamming Code, as appropriate, and further compares their properties to select a more efficient code from this pair. As a result of the comparison, it selects the Extended Hamming Code as more suitable for the subsequent design of the program. The code design procedures, tests that verify the functionality of the code, and the resources used for the Spartan-3 circuit to implement the code for the Spartan-3 circuit are then shown.

Keywords

Self correction code, Error Detection Code, Error Correction code, Cyclic code, Extended Hamming Code, BCH code, Burst error

Bibliografická citace

ZÁHORA, Jakub. *Ochrana dat v paměti SRAM pomocí samoopravných kódů.* Brno, 2022. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/142794. Bakalářska práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav mikroelektroniky. Vedoucí práce Vojtěch Dvořák.

Prohlášení autora o původnosti díla

Jméno a příjmení studenta:	*Jakub Záhora*
VUT ID studenta:	*221288*
Typ práce:	*Bakalářská práce*
Akademický rok:	*2021/22*
Téma závěrečné práce:	Ochrana dat v paměti SRAM pomocí samoopravných kódů

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

.

V Brně dne: 2. červen 2022 ----------

podpis autora

Poděkování

Ďakujem vedúcemu mojej bakalárskej práce Ing. Vojtěchovi Dvořákovi, za poskytnutie literatúry, odborné rady ako aj pedagogický dohľad nad touto prácou, taktiež za poskytnutý čas v rámci konzultácií ako aj kontroly mojej práce. Ďalej ďakujem svojim rodičom za psychickú podporu a čas strávený pri kontrole a čítaní mojej práce.

V Brně dne: 2. červen 2022 ----------

podpis autora

Obsah

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Už od vzniku pamäťových modulov sa stretávame s množstvom problémov, ktoré tieto moduly majú. S postupným zmenšovaním (integráciou) sa začala častejšie objavovať chyba, ktorá sa dá popísať ako zmena hodnoty bitu. Tento jav je najčastejšie spôsobený vplyvom žiarenia (preletom nabitých častíc cez pamäťovú jednotku). Jedná sa o náhodný jav, takže sa nedá presne určiť kedy, ani v ktorej časti sa daná chyba objaví, preto bola potreba vyvinúť spôsob ako by sa daná chyba dala identifikovať a následne aj opraviť. Pre danú problematiku boli vytvorené kódy, ktoré pred zápisom bitového vektoru do pamäte pridajú nadbytočnú informáciu, ktorá je vytvorená v kodére a je následné uložená s užitočnou informáciou do pamäťovej jednotky. Následne pri potrebe použitia danej informácie sa pomocou nadbytočnej informácie v dekodéry prekontrolujú dáta pre overenie, či je daná informácia bez chyby. Pokiaľ je odhalená chyba v dôležitej informácii, v závislosti na type použitého kódu je určité množstvo chýb opravených. Tieto kódy sa nazývajú samo opravné kódy.

V teoretickej časti tejto práce sú vysvetlené základy teórie potrebnej pre ďalšiu prácu so samo opravnými kódmi. Následne sú definované základné parametre, ktoré boli u jednotlivých kódov sledované, ako napríklad počet opravených, či detekovaných chýb, veľkosť dodatočnej a užitočnej informácie. Z jednotlivých vysvetlených kódov boli vybrané kódy, ktoré spĺňajú zadanie opravy jednonásobnej chyby a detekcie dvojnásobnej chyby. Z nich bol následne vyberaný kód s najlepšími parametrami pre realizáciu zadania.

V praktickej časti je pre vybraný najvhodnejší kód z teoretickej časti opísaný postup vývoja kódu resp. postup vytvorenia kódu s vysvetlením funkcie jednotlivých modulov ako aj výsledného kódu. Následne sú rozobrané jednotlivé testy, ktoré boli spravené pre overenie funkčnosti jednotlivých komponentov ako aj celého kódu. Nakoniec bol kód syntetizovaný pre vybraný obvod FPGA, jednalo sa o obvod Spartan-3, pričom bolo sledované využitie jeho zdrojov.

1. ÚVOD DO KÓDOVANIA

V číslicových systémoch je častým problémom zmena hodnoty informácie prenášanej na trase medzi zdrojom a pamäťovým úložiskom, či zmena hodnoty uloženej informácie voči informácii vyslanej zo zdroja. Pre odstránenie týchto nežiadúcich javov sa používajú bezpečnostné kódy.[1]

Kód je ľubovoľne dlhá množina slov, ktorá nie je prázdna.[2]

1.1 Bezpečnostný kód

Jedná sa o kód, ktorý obsahuje, respektíve je do neho zavedená tzv. informačná redundancia. Ku znakom užitočnej informácie, ktorá je prenášaná zo zdroja do pamäte sa pridajú nadbytočné znaky, ktoré nenesú žiadnu informáciu, ale slúžia na kontrolu správnosti užitočnej informácie. Nadbytočné znaky sa volia na základe určitej matematickej kombinácie z užitočnej informácie a sú jedinečné pre rôzne užitočné informácie, potom sa takýmto nadbytočným znakom hovorí tzv. kontrolné slová. V prípade zmeny užitočnej informácie počas prenosu, či počas doby uloženia v pamäti, je možné danú chybu pomocou kontrolných slov odhaliť a za určitých podmienok aj opraviť.[2]

Kódovanie jedná sa proces, pri ktorom sa prenášaná informácia zašifruje pomocou náhodného či vopred daného kľúča, alebo sa jedná o jej doplnenie o nadbytočné znaky, ktoré sa vytvoria na základe matematických kombinácií slov, ktoré prenášaná informácia obsahuje. Kódovanie sa vykonáva čo najbližšie k zdroju prenášanej informácie pre odstránenie nežiadúcich javov, resp. pred uložením informácie do pamäte.[1]

Dekódovanie je proces, pri ktorom sa prenášaná informácia rozšifruje pomocou vopred určeného kľúča, či proces slúžiaci pre kontrolu správnosti prenášanej informácie pomocou nadbytočných znakov. Dekódovanie sa používa pred opätovným použitím informácie uloženej v pamäti, či pri prijatí informácie z externého zdroja.[1]

1.2 Vlastnosti kódu

Jedna z najdôležitejších vlastností kódu je detekovanie a oprava určitého počtu chýb.

Chybový vektor resp. chybu môžeme vnímať ako zmenu kódového vektoru na nekódový vektor, alebo kódový vektor, ktorý má ale rozdielnu hodnotu, čiže môžeme povedať, že chybný vektor je vektor, u ktorého došlo k zmene voči pôvodnému kódovému vektoru. Chyby môžu byť jednonásobné či viacnásobné, resp. nie je určený maximálny počet chýb. Chyba znamená, že došlo k prepisu hodnoty určitého bitu pomocou chybového vektoru, ktorý má na danom bite hodnotu, ktorá sa prepíše na rovnakom bite kódového vektoru, takže napr. v prípade prepisu dvoch bitov kódového vektoru došlo k dvojnásobnej chybe.[1]

Detekcia chyby jedná sa o proces, pri ktorom odhalíme alebo neodhalíme chybu. Pre prípad, keď je chyba odhalená sa jedná o zmenu kódového slova na nekódové slovo, resp. počet chybných bitov, ktoré nekódové slovo obsahuje je menší alebo rovný počtu chýb, ktoré kód dokáže detekovať. Pre prípad, keď chyba nie je odhalená, sa jedná o kódové slovo, ktoré buď nebolo zmenené, alebo obsahuje počet chýb, ktorý je väčší ako počet chýb, ktoré kód dokáže detekovať. Najväčšia násobnosť (počet) chýb, ktorú dokáže kód detekovať je menšia než minimálna kódová vzdialenosť.[1]

Oprava chyby je dej, pri ktorom sa chyba, alebo chyby v kódovom vektore opravia na základe princípu maximálnej vierohodnosti tz. podľa najväčšej pravdepodobnosti správnej opravy. Takže ak zaznamenáme chybu na určitom bite kódového slova oprava prebieha prepisom hodnoty nekódového slova na kódové slovo s najmenšou Hammingovou vzdialenosťou.[1]

2. SLEDOVANÉ PARAMETRE KÓDU

V rámci spracovania kódov bude sledovaných niekoľko parametrov, ktoré budú slúžiť pre porovnanie jednotlivých kódov a určenie výhodnosti ich použitia a výber najvhodnejšieho kódu pre praktickú časť. Medzi sledované parametre budú patriť – dĺžka kódu, počet informačných bitov, počet kontrolných bitov, informačný pomer a minimálnu kódovú vzdialenosť. Nižšie sú uvedené definície jednotlivých sledovaných parametrov, k nim sú pridané pojmy ako Hammingova váha a Hammingova vzdialenosť, ktoré sú dôležité pri detekcii nezávislých chýb.[1]

Dĺžka kódu predstavuje počet súradníc kódového vektoru, resp. celkový počet bitov kódového vektoru zahŕňajúci informačné bity ako aj kontrolné bity. Dĺžku kódu budeme označovať *n*.[1]

Počet informačných bitov je počet bitov kódového vektoru, ktoré obsahujú zdrojovú informáciu t.j. užitočnú informáciu, ktorá je nezávislá na ostatných bitoch, pre výpočty bude ďalej označovaná *k*.[1]

Počet kontrolných bitov jedná sa o bity kódového vektoru, ktoré nenesú zdrojovú informáciu, sú závislé na informačných znakoch daného kódového vektoru a pre rozdielne informačné znaky sú rozdielne aj kontrolné znaky, čo dokazuje ich jedinečnosť. Pre výpočet budú ďalej označované *r*.[1]

Informačný pomer udáva pomer medzi počtom informačných bitov prenášaných kódovým vektorom voči celkovému počtu bitov kódového vektoru. Pre výpočet bude ďalej označovaný ako *R*.[1]

Hammingova váha udáva počet nenulových bitov resp. bitov s hodnotou 1, v súradnici binárneho vektoru a značí sa *w(v)*.[1]

Hammingova vzdialenosť reprezentuje (udáva) počet rozdielnych súradníc medzi dvoma binárnymi vektormi, pričom veľkosť daných vektorov (počet súradníc) musí byť totožná. Následne pre výpočet Hammingovej vzdialenosti platí:

$$d(u,v) = w(u+v), \tag{2.1}$$

kde *d(u,v)* predstavuje Hammingovú vzdialenosť, *w(u+v)* predstavuje hammingovú váhu, *u* a *v* predstavujú skúmané vektory s totožnou dĺžkou .[1]

Príklad výpočtu Hammingovej váhy a vzdialenosti vektorov u = (01111) a v = (11001). Platí, že $w(u) = 4$, $w(v) = 3$, potom $u+v$ = (01111)+(11001) = (01000), a $d(u,v) = w(u+v) = 4$.

Minimálna kódová vzdialenosť d_{min} je najmenšia hodnota z kódových vzdialeností všetkých dvojíc kódových vektorov kódu *K*. Minimálna kódová vzdialenosť je potom rovná najmenšej Hammingovej váhe zo všetkých nenulových slov.[2]

3. KÓDY

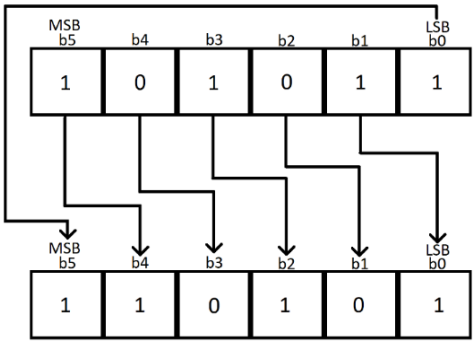
V tejto kapitole sú popísané jednotlivé typy kódov s vysvetlením ich funkčnosti a ich vlastností, ktoré sú vyššie definované. Údaje budú spracované do tabuliek, pre jednoduchší prehľad. Následne z nich bude vybraný kód, ktorý je najvhodnejší pre zadanie tejto práce.[1]

Detekčné kódy z anglického „*Error Detection Codes*“ (EDC) sú kódy, u ktorých je hlavnou funkciou detekcia chyby pri prenose kódového slova od zdroja, k príjemcovi. Chyba je odhalená na základe kontrolného vektoru, ktorý kódové slovo obsahuje. V prípade detekcie chyby sa už jedná o nekódové slovo, avšak detekčné kódy nie sú schopné chybu opraviť, len detekovať. Na základe týchto informácií sa dá povedať, že sa jedná o jednoduché kódy, ktoré sú vhodné pre situácie, kedy v prípade vzniku chyby nie je potreba chybu opraviť. Príkladom sú cyklické kódy.[8]

Korekčné kódy z anglického „*Error Corrections Codes*“ (ECC) sú kódy, u ktorých je hlavnou funkciou detekcia chyby a ich následná korekcia (chybnej hodnoty bitu na definovanej pozícií). Na základe typu kódu, jednotlivé kódy dokážu vykonať korekciu jednonásobnej, ako aj viacnásobnej chyby v závislosti od matematickej funkcie, ktorá je použitá. Na rozdiel od detekčných kódov, dokážu odhalenú chybu aj opraviť, avšak sú zložitejšie, čo zapríčiňuje zložitejší návrh ako aj potrebu väčšej kapacity pamäte pre kód. Patria sem napríklad Hamingov kód, či Bosov-Chaudhuriho-Hocquenghov kód.[9]

3.1 Cyklické kódy

Cyklické kódy sú lineárne (*n,k*) blokové kódy, kde *n* predstavuje celkový počet bitov a *k* predstavuje počet informačných bitov. U týchto platí tzv. cyklickosť, to znamená, že pokiaľ existuje kódové slovo, a to následne bude zarotované resp. uskutoční sa bitový posun, ale so zachovaním rovnakého počtu jednotiek a núl, jedná sa o vznik iného kódového slova. Posuvy sú realizované pomocou posuvných registrov so spätnou väzbou, z anglického „*Shift-Register with Feedback*“. Rotácia funguje napríklad na princípe posuvu vpravo, to sa realizuje pomocou už spomínaného posuvného registra so spätnou väzbou, kde LSB bit sa presunie na pozíciu MSB a následne sú ostatné bity posunuté o jednu pozíciu vpravo, tento dej je zobrazený na Obrázok 3-1.[4]



Obrázok 3-1 Ukážka posuvného registru so spätnou väzbou vytvorené v [skicár]

3.2 Kontrola parity

Kontrola parity, z anglického „*Vertical Redundancy Check*“ (VRC), je najjednoduchší typ cyklických kódov. Pracuje na princípe pridania paritného bytu na základe počtu 1 alebo 0 v kóde (podľa zvoleného typu kodéru párna, alebo nepárna parita) sa priradí paritný bit, ktorý môže mať hodnotu 0 alebo 1. Celková veľkosť kódového slova je potom rovná informačnému slovu a jednému kontrolnému bitu.[3]

Kodér párnej parity v prípade, ak informačné slovo obsahuje párny počet jednotiek, potom kontrolný bit, ktorý bude k informačnému slovu pridaný bude mať hodnotu 0. V prípade, ak informačné slovo bude mať nepárny počet jednotiek, potom kontrolný bit bude mať hodnotu 1.[3]

Kodér nepárnej parity v prípade, ak informačné slovo obsahuje párny počet jednotiek, potom kontrolný bit, ktorý bude k informačnému slovu pridaný bude mať hodnotu 1. V prípade, ak informačné slovo bude mať nepárny počet jednotiek, potom kontrolný bit bude mať hodnotu 0.[3]

Z definície kodéru párnej a nepárnej parity je možné odvodiť, že sa jedná o výslednú paritu kódového slova, takže bit je volený tak, aby hodnota výsledného kódového slova bola párna, alebo nepárna na základe použitého typu kodéru.[3]

Ako je z Tabuľka 3.1 badateľné, tento kód je síce schopný odhaliť chybu, ale nie je schopný ju opraviť, nakoľko v niektorých prípadoch ako napríklad pre hodnoty $w(u_1)$ = 001, $w(u_2)$ = 010 a $w(u_4)$ = 100, sa nedá štatisticky predpokladať najpravdepodobnejšia situácia pri zmene hodnoty bitov s hodnotou 1 alebo niektorého bitu s hodnotou 0, z toho vyplýva, že tento kód dokáže detekovať jednu chybu, ale nedokáže danú chybu efektívne opraviť.

Tabuľka 3.1 Ukážka hodnôt paritného bitu pri 3 bitovom informačnom slove

3 bit dáta			Párna parita		Nepárna parita	
b_2	b_1	b_0	Informácia	Parita	Informácia	Parita
0	0	0	000	0	000	1
0	0	1	001	1	001	0
0	1	0	010	1	010	0
0	1	1	011	0	011	1
1	0	0	100	1	100	0
1	0	1	101	0	101	1
1	1	0	110	0	110	1
1	1	1	111	1	111	0

Ako je vidieť v Tabuľka 3.2, síce sa jedná o veľmi efektívny kód, nakoľko obsahuje len jeden kontrolný bit pre dva a viac informačných bitov, z čoho vyplýva veľká efektivita v rámci ušetrenia priestoru v pamäti, avšak tento kód nedokáže odhaliť dvojnásobnú chybu ani opraviť jednonásobnú chybu, preto tento kód nie je vhodný pre riešenie tejto problematiky.

Tabuľka 3.2 Parametre kódu kontroly parity

Typ kódu:	Kontrola parity	
	funkcia	hodnota
Parametre		
Minimálna vzdialenosť d_{min}	d_{min} (u_x+u_y)	d_{min} =2
Dĺžka slova n	n = k+1	≥3
Počet informačných bitov k	k = n-1	≥2
Informačný pomer R	$R = \frac{k}{n}$	≥0,66
Vlastnosti		
Počet detekovaných chýb		1
Počet opravených chýb		0
Oprava viacnásobných chýb		Nie

3.3 Cyklická Kontrola Redundancie

Cyklická Kontrola Redundancie z anglického „*Cyclic Redundancy Check*“ (CRC), jedná sa o lineárny (n,k) blokový kód s vlastnosťou, že pri každom cyklickom posune kódového slova vzniká nové kódové slovo. V rámci označenia k predstavuje počet informačných bitov odoslaných zo zdroja, n predstavuje celkovú dĺžku správy resp. celkový počet bitov (informačných ako aj kontrolných). CRC kódy sú redukovanou verziou cyklických kódov, sú ľahko implementovateľné pomocou shift-registrov so spätnou väzbou, vďaka čomu sú dosť rozšírené a používané v oblasti detekcie chýb v digitálnej komunikácií.[5]

Generovanie CRC kódu na základe požadovaného počtu bitových kontrol, sa pridá k aktuálnym údajom určitý počet bitov s nulovou hodnotou. Následne sa pomocou matematickej operácie delenia definovaným číslom reprezentovaným v bitovej podobe, vypočíta hodnota zvyšku po delení informačného vektora, táto hodnota nahradí predtým pridaný počet bitov s nulovou hodnotou. Následne sa kód odošle cez vysielač, po prijatí kódu je postup rovnaký vykoná sa operácia delenia definovaným číslom reprezentovaným v bitovej podobe, z ktorej zistíme celočíselný výsledok, ten porovnáme s hodnotou zapísanou v kontrolných bitoch, ak je hodnota zhodná jedná sa o správne prenesené kódové slovo, v prípade že je rozdielna, jedná sa už o nekódové slovo. Delenie je pre kód realizované funkciou XOR.[6]

Príklad: Je známe informačné slovo w(v_0)=1101101 kontrolné slovo w(u)=000

Deliteľ má hodnotu w(d)=1011, určite kódové slovo w(k).

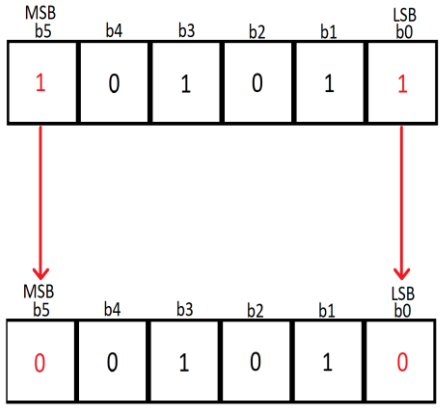
```
1101101 000     <= informačné slovo doplnené o kontrolné slovo
1011            <= deliteľ
0110101 000     <= výsledok logickej operácie XOR prvých štyroch znakov,
                   ostatné sa opíšu

0110101 000
 1011
0011001 000
  1011
0001111 000
   1011
0000100 000
    101 1
0000001 100
      1 011
0000000 111     <= výsledok (zvyšok po delení, ktorý v kódovom slove nahradí kontrolne
                   dovtedy nulové znaky)
```

Výsledok: informačné slovo w(v_0)=1101101 kontrolné slovo w(u)=111 kódové slovo bude mať potom hodnotu w(k)=1101101 111.

Detekovanie chyby: v prípade jednonásobnej chyby je nutné, aby mal kontrolný vektor dĺžku minimálne 2. Pre prípad viacnásobnej chyby, ktorá je pre tento typ kódu označovaná ako zhluková chyba (Burst error) je nutné, aby dĺžka kontrolného vektoru bola o jedna väčšia ako dĺžka zhlukovej chyby, aby bola zaručená správnosť opravy chyby.[7]

Zhluková chyba: je definovaná ako rozmedzie medzi dvoma za sebou idúcimi chybami.[7]



Obrázok 3-2 Ukážka zhlukovej chyby vytvorené v [**skicár**]

Ako je z Obrázok 3-2 viditeľné, kódové slovo malo dĺžku sedem, chyba nastala na pozícii MSB a LSB bitu, ktorých vzdialenosť je sedem, čo je zároveň veľkosť zhlukovej chyby, z toho vyplýva, že kontrolný vektor by mal byť o jeden bit dlhší ako zhluková chyba, takže jeho dĺžka by mala byť osem.
Ako je uvedené v

Tabuľka 3.3, tento kód dokáže detekovať jednonásobnú chybu, u viacnásobnej chyby je detekcia chyby závislá na vzdialenosti dvoch po sebe idúcich chýb, ako aj na počte bitov kontrolného slova, pre správnu detekciu dvojnásobnej chyby by ich vzdialenosť musela byť pevne definovaná a na základe toho by sme zvolili dĺžku kontrolného slova. Následne by parameter detekcie dvojnásobnej chyby mohol byť považovaný za splnený. Pre opravu jednonásobnej chyby tento kód nespĺňa požiadavky, nakoľko sa jedná o detekčný a nie korekčný kód. Tento kód je síce jednoduchý, no nie je vhodný pre riešenie zadanej problematiky.

Tabuľka 3.3 Parametre kódu Cyklickej Kontroly Redundancie

Typ kódu:	Cyklická kontrola redundancie	
	funkcia	hodnota
Parametre		
Minimálna vzdialenosť d_{min}	$w_{min}(u_x+u_y)$	d_{min} =2
Dĺžka slova n	n = k+r	≥3
Počet informačných bitov k	k = n-r	≥2
Informačný pomer R	$R = \frac{k}{n}$	≥0,66
Vlastnosti		
Počet detekovaných chýb	n-1	≥1
Počet opravených chýb		0
Oprava viacnásobných chýb		Nie

3.4 Hammingov kód

Hammingov kód patrí medzi dokonalé kódy. Dokonalý kód je typ kódu, ktorý dosahuje maximálny možný pomer medzi informačným slovom a kódovým slovom, pre kódy s určitou dĺžkou bloku a minimálnou kódovou vzdialenosťou d_{min} = 3. Takže Hammingov kód patri medzi korekčné kódy a je ich najjednoduchšou variantou. Kód dokáže detekovať chybu v celom rozsahu informačného slova. Detekciu chyby vykonáva pomocou kontrolných bitov, ktorých hodnota sa priraďuje na základe parity konkrétnej kombinácie bitov obsiahnutých v informačnom slove. Počet kontrolných bitov je závislý na počte informačných bitov. Hammingov kód je univerzálnym kódom, to znamená, že je možné ho použiť pre hocijakú dĺžku informačného slova. Tento typ kódu je však primárne vhodný pre procesy, pri ktorých sa predpokladá malá chybovosť ako sú procesy v pamätiach počítačov.[10]

Funkčnosť Hammingovho kódu, pre určenie počtu kontrolných bitov potrebujeme vedieť dĺžku informačného slova. Na základe dĺžky informačného slova vypočítame dĺžku kontrolného slova. Následne pozícia kontrolných bitov je pevne daná a jedná sa o pozície, ktoré sú vypočítané ako mocniny pri základe 2. Hodnota jednotlivých kontrolných (paritných) bitov je volená na základe matematickej kombinácie informačných bitov. Kombinácia je určená na základe významnosti jednotlivých informačných bitov.[10]

Kontrolný bit (b_1) pokrýva všetky možné kombinácie, ktoré majú nastavený najmenej významný bit (b_1). Kombinuje bity, ktorých binárna hodnota na prvom bite (b_1), majú hodnotu 1. Napríklad $3_{10} = (011)$, $5_{10} = (101)_2$, $7_{10} = (111)_2$.[10]

Kontrolný bit (b_2) pokrýva všetky možné kombinácie, ktoré majú nastavený najmenej významný bit (b_2). Kombinuje bity, ktorých binárna hodnota na druhom bite (b_2), majú hodnotu 1. Napríklad $3_{10} = (011)$, $6_{10} = (110)_2$, $7_{10} = (111)_2$.[10]

Kontrolný bit (b_3) pokrýva všetky možné kombinácie, ktoré majú nastavený najmenej významný bit (b_3). Kombinuje bity, ktorých binárna hodnota na treťom bite (b_3), majú hodnotu 1. Napríklad $5_{10} = (101)$, $6_{10} = (110)_2$, $7_{10} = (111)_2$.[10]

Poznámka: hodnota x_{10} predstavuje pozíciu bitu v kódovom slove. Hodnota $(yzv)_2$ predstavuje premenu hodnoty x_{10} z dekadickej hodnoty na binárnu hodnotu.

Príklad: Informačné slovo má dĺžku 4 a jeho hodnota je w(k)=1101. Určite potrebný počet kontrolných bitov r, kontrolné slovo w(r), dĺžku kódového slova n a kódového slovo w(n).

Výpočet:

Pre určenie počtu kontrolných bitov použijeme vzťah

$$2^r \geq k+r+1 , \qquad (3.1)$$

kde r predstavuje počet kontrolných bitov, k predstavuje počet informačných bitov. Zvolíme si náhodné číslo r=2 a overíme pravdivosť rovnice.

$$2^r \geq k + r + 1 => 2^2 \geq 4 + 2 + 1$$
$$4 \geq 7$$

Hodnota r=2 nespĺňa pravdivosť rovnice, z toho vyplýva, že dva kontrolné bity nebudú stačiť, takže musíme zvoliť inú väčšiu hodnotu, napríklad r=3.

$$2^r \geq k + r + 1 => 2^3 \geq 4 + 3 + 1$$
$$8 \geq 8$$

Hodnota r=3 spĺňa pravdivosť rovnice, z toho vyplýva, že tri kontrolné bity je minimum, ktoré musí byť použité, samozrejme pre väčší počet kontrolných bitov bude rovnica automaticky splnená, ale už znižujeme efektivitu kódu, pretože kontrolné bity budú zaberať väčšie množstvo pamäte.

Pre určenie pozície pre kontrolné a informačné bity v kódovom slove je prvým krokom určenie pozícií kontrolných bitov. Počet kontrolných bitov je r=3, ich pozíciu určíme ako mocninu pri základe 2. Následne voľné bity doplníme informačnými bitmi v stanovenom poradí. Kontrolné bity budú označované b_r a informačné bity b_k.

Tabuľka 3.4 Určenie pozície informačných a kontrolných bitov v kódovom slove

Označenie bitu	b_6	b_5	b_4	b_3	b_2	b_1	b_0
Pozícia bitu	7	6	5	4	3	2	1
Bitová hodnota pozície bitu	111	110	101	100	011	010	001
Označenie informačného bitu	b_{k3}	b_{k2}	b_{k1}		b_{k0}		
Hodnota informačného bitu	1	1	0		1		
Označenie kontrolného bitu				b_{r3}		b_{r2}	b_{r1}
Hodnota kontrolného bitu				-		-	-

Pre určenie hodnoty kontrolných bitov použijeme dopočet pre párnu paritu uvedených bitov.

Pre b_{r1}: Jedná sa o informačné bity b_{k0}, b_{k1}, b_{k2}. Ich hodnoty sú 1,0,1, pre doplnenie párnej parity potom musí byť hodnota $b_{r1} = 0$.

Pre b_{r2}: Jedná sa o informačné bity b_{k0}, b_{k2}, b_{k3}. Ich hodnoty sú 1,1,1, pre doplnenie párnej parity potom musí byť hodnota $b_{r2} = 1$.

Pre b_{r3}: Jedná sa o informačné bity b_{k1}, b_{k2}, b_{k3}. Ich hodnoty sú 0,1,1, pre doplnenie párnej parity potom musí byť hodnota $b_{r1} = 0$.

Z toho vyplýva, že kontrolné slovo bude mať hodnotu w(r) = 010.

Pre určenie veľkosti kódového slova použijeme vzťah

$$n = k + r\,, \tag{3.2}$$

kde k predstavuje počet informačných bitov a r predstavuje počet kontrolných bitov.

$$n = k + r = 3 + 4 = 7$$

Určenie hodnoty kódového slova: Na základe rozloženia bitov znázornených v Tabuľka 3.4, môžeme určiť, že kódové slovo bude w(n) = 1100110.

Na základe Tabuľka 3.5, v ktorej sú uvedené dôležité informácie o Hammingovom kóde, môžeme povedať, že tento typ kódu je vhodný pre prípad návrhu kódu, ktorý dokáže detekovať a opraviť jednonásobnú chybu. Našim cieľom je ale nájsť kód, ktorý je schopný detekovať dvojnásobnú chybu a opraviť jednonásobnú chybu, a na to tento kód nie je vhodný. Z toho vyplýva, že tento kód nie je vhodný pre riešenie zadanej problematiky.

Tabuľka 3.5 Parametre Hammingovho kódu

Typ kódu:	Hammingov kód	
	funkcia	hodnota
Parametre		
Minimálna vzdialenosť d_{min}	$w_{min}(u_x+u_y)$	d_{min} =3
Dĺžka slova n	$n = 2^r-1=k+r$	≥6
Počet informačných bitov k	k = n-r	≥3
Informačný pomer R	$R = \frac{k}{n}$	≥0,429
Vlastnosti		
Počet detekovaných chýb		1
Počet opravených chýb		1
Oprava viacnásobných chýb		Nie

3.4.1 Rozšírený Hammingov kód

Jedná sa o modifikáciu Hammingovho kódu, ktorá je označovaná ako „SECDEC“ (Single Error Corection Double Error Detection). Jedná sa o rozšírenie kontrolného slova v Hammingovom kóde o jeden bit, tento bit slúži pre doplnenie celkovej parity na párnu.[10] Vďaka pridanému bitu sa zvýši minimálna Hammingova vzdialenosť d_{min} = 4.[1] Rozšírený Hammingov kód tým pádom prináša možnosť detekcie dvojnásobnej chyby.[10]

Umiestnenie paritného bitu je na pozícií LSB bitu kódového slova. Ostatné kontrolné znaky sú umiestnené na pozíciách mocniny pri základe 2.[10]

Príklad: Informačné slovo má dĺžku 11 a jeho hodnota je w(k) = 11001101000. Určite potrebný počet kontrolných bitov r, kontrolné slovo w(r), dĺžku kódového slova n a kódového slovo w(n).

Výpočet:

Pre určenie počtu kontrolných bitov použijeme vzťah

$$2^{r-1} \geq k + r, \tag{3.3}$$

kde *r* predstavuje počet kontrolných bitov a *k* predstavuje počet informačných bitov.

Zvolí sa náhodná hodnota *r*=4 pre overenie pravdivosti rovnice.

$$2^{r-1} \geq k + r \Rightarrow 2^{4-1} \geq 11 + 4$$
$$8 \geq 15$$

Hodnota *r*=4 nespĺňa pravdivosť rovnice, z toho vyplýva, že štyri kontrolné bity nebudú stačiť, takže musíme zvoliť inú väčšiu hodnotu, napríklad *r*=5.

$$2^{r-1} \geq k + r \Rightarrow 2^{5-1} \geq 11 + 5$$
$$16 \geq 16$$

Hodnota *r*=5 spĺňa pravdivosť rovnice, z toho vyplýva, že päť kontrolných bitov je minimum, ktoré musí byť použité, samozrejme pre väčší počet kontrolných bitov bude rovnica automaticky splnená, ale už znižujeme efektivitu kódu, pretože kontrolné bity budú zaberať väčšie množstvo pamäte.

Pre určenie pozície pre kontrolné a informačné bity v kódovom slove je prvým krokom určenie pozícií kontrolných bitov. Počet kontrolných bitov je r=5. Pozícia paritného bitu je pevne daná na pozícií LSB bitu, pozíciu ostatných kontrolných bitov určujeme ako mocninu pri základe 2.

Tabuľka 3.6 Určenie pozície informačných a kontrolných bitov v kódovom slove

Označenie bitu	b_{15}	b_{14}	b_{13}	b_{12}	b_{11}	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
Pozícia bitu	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bitová hodnota pozície bitu	1 1 1 1	1 1 1 0	1 1 0 1	1 1 0 0	1 0 1 1	1 0 1 0	1 0 0 1	1 0 0 0	0 1 1 1	0 1 1 0	0 1 0 1	0 1 0 0	0 0 1 1	0 0 1 0	0 0 0 1	0 0 0 0
Označenie informačného bitu	b_{k10}	b_{k9}	b_{k8}	b_{k7}	b_{k6}	b_{k5}	b_{k4}		b_{k3}	b_{k2}	b_{k1}		b_{k0}			
Hodnota informačného bitu	1	1	0	0	1	1	0		1	0	0		0			
Označenie kontrolného bitu								b_{r4}				b_{r3}		b_{r2}	b_{r1}	b_{r0}
Hodnota kontrolného bitu								-				-		-	-	-

Pre určenie hodnoty kontrolných bitov sa použije doplnenia parity pre definovanú kombináciu informačných bitov.

Pre b_{r1}: Jedná sa o informačné bity b_{k0}, b_{k1}, b_{k3}, b_{k4}, b_{k6}, b_{k8}, b_{k10}. Ich hodnoty sú 1,0,1,0,1,0,0 pre doplnenie párnej parity potom musí byť hodnota $b_{r1} = 1$.

Pre b_{r2}: Jedná sa o informačné bity b_{k0}, b_{k2}, b_{k3}, b_{k5}, b_{k6}, b_{k9}, b_{k10}. Ich hodnoty sú 1,1,1,1,1,0,0 pre doplnenie párnej parity potom musí byť hodnota $b_{r2} = 1$.

Pre b_{r3}: Jedná sa o informačné bity b_{k1}, b_{k2}, b_{k3}, b_{k7}, b_{k8}, b_{k9}, b_{k10}. Ich hodnoty sú 1,1,0,0,1,0,0 pre doplnenie párnej parity potom musí byť hodnota $b_{r1} = 1$.

Pre b_{r4}: Jedná sa o informačné bity b_{k4}, b_{k5}, b_{k6}, b_{k7}, b_{k8}, b_{k9}, b_{k10}. Ich hodnoty sú 1,1,0,0,1,1,0 pre doplnenie párnej parity potom musí byť hodnota $b_{r1} = 0$.

Pre b_{r0}: Hodnota sa určuje na základe celkovej parity ostatných bitov, takže parity bitov b_{15} až b_1. Celkový počet log. 1

Z toho vyplýva, že kontrolné slovo bude mať hodnotu w(r) = 01111.

Pre určenie veľkosti kódového slova je použitý vzťah

$$n = k + r, \tag{3.4}$$

kde *n* predstavuje veľkosť kódového slova, *r* predstavuje veľkosť kontrolného slova a *k* predstavuje veľkosť informačného slova.

$$n = k + r = 5 + 11 = 16$$

Výsledok:

Určenie hodnoty kódového slova je na základe rozloženia bitov znázornených v Tabuľka 3.7, kódové slovo bude w(n) = 1100110110010101.

Poznámka: Odhalenie jednonásobnej chyby je pomocou paritného bitu, ak celková parita kódového slova nie je párna, znamená to, že sa v kódovom slove nachádza bit s chybnou hodnotou, resp. sa už nejedná o kódové slovo. Pre určenie, ktorý polohy chybného bitu v kódovom slove slúžia kontrolné bity b_{r4} až b_{r1}. Detekcia dvojnásobnej chyby nastáva v prípade, ak je dosiahnutá párna parita kódového slova, avšak pri vnútornej kontrole pomocou kontrolných bitov b_{r4} až b_{r1} je odhalená chyba. V takomto prípade sa nedá určiť presná poloha chýb, nakoľko by bolo nutné lokalizovať dve chybné hodnoty a SECDEC dokáže lokalizovať len jednu chybu. Kódové slovo sa vyhodnotí ako nekódové. Pri zvyšovaní počtu informačných bitov bude síce dochádzať k zvyšovaniu kontrolných bitov, ale nie úmerne (počet informačných bitov bude narastať oveľa pomalšie). Takže pre dosiahnutie lepšieho informačného pomeru je výhodné používanie dlhších informačných slov.

Na základe Tabuľka 3.7 môžeme povedať, že Rozšírený Hammingov kód spĺňa požiadavky, ktoré sú zadaním tejto práce a to, že by hľadaný kód by mal mať schopnosť detekovať dvojnásobnú chybu a opraviť jednonásobnú chybu. Z toho vyplýva, že Rozšírený Hammingov kód je vhodným kandidátom pre následný praktický návrh.

Tabuľka 3.7 Parametre Rozšíreného Hammingovho kódu

Typ kódu:	Rozšírený Hammingov kód	
	funkcia	hodnota
Parametre		
Minimálna vzdialenosť d_{min}	$w_{min}(u_x+u_y)$	d_{min} =4
Dĺžka slova n	$n = 2^r-1=k+r$	≥8
Počet informačných bitov k	k = n-r	≥4
Informačný pomer R	$R = \frac{k}{n}$	≥0,5
Vlastnosti		
Počet detekovaných chýb		2
Počet opravených chýb		1
Oprava viacnásobných chýb		Nie

3.5 BCH kód

BCH je skratkou názvu „Bose-Chaudhuri-Hocquenghem“. Jedná sa o kód, ktorý pozostáva z polynomických funkcií, zložených nad konečným polom. Jedná sa o kódy, u ktorých existuje možnosť zadefinovania presného počtu chýb, ktoré budú opravené už pri návrhu, to znamená, že na základe návrhu kódu sa definuje maximum chýb, ktoré sa v kódovom slove môžu vyskytnúť. Pre detekciu chýb sa využívajú algebraické funkcie známe ako syndróm dekódovania. BCH kódy predstavujú zovšeobecnenú verziu Hammingových kódov.[11]

Dôležité parametre: Pre určenie minimálnej Hammingovej vzdialenosti platí vzťah $d_{min} \geq 2t+1$, kde t predstavuje maximálny počet chýb, ktoré v kódovom slove budú opravené, d_{min} predstavuje minimálnu Hammingovú vzdialenosť. Následne je potrebné definovať dĺžku kódového slova $n=2^m-1$, kde m predstavuje reálne celé číslo, a zároveň $m \geq 3$. Maximálna dĺžka informačného slova sa určí ako $k_{max} \geq n-m*t$, jednotlivé parametre sú uvedené vyššie. Dĺžka kontrolného slova sa určuje podľa vzťahu r=n-k, kde k predstavuje dĺžku informačného slova. Informačný pomer R=k/n.[2]

Príklad: Dĺžka informačného slova je k=6, informačný vektor má hodnotu w(k)=101101, Počet chýb vyskytujúcich sa v kódovom slove t=2, hodnota m=4. Vypočítajte dĺžku kódového vektoru, ako aj jeho hodnotu w(n), maximálnu dĺžku informačného slova k_{max}, minimálnu hammingovú vzdialenosť d_{min}, dĺžku kontrolného vektoru r, tvar generovaného polynómu g(x), tvar informačného polynómu u(x), polynóm kódového slova c(x).

Výpočet:

Pre výpočet dĺžky kódového slova je použitý vzťah

$$n = 2^m - 1 , \tag{3.5}$$

kde *n* predstavuje veľkosť kódového slova a *m* predstavuje reálne celé číslo s hodnotou $m \geq 3$.

$$n = 2^m - 1 = 2^4 - 1$$
$$n = 15$$

Pre výpočet maximálnej dĺžky informačného slova platí vzťah

$$k_{max} = n - m * t , \tag{3.6}$$

kde k_{max} predstavuje maximálnu dĺžku informačného slova, *n* predstavuje dĺžku kódového slova, *m* predstavuje reálne celé číslo s hodnotou $m \geq 3$ a *t* predstavuje vopred definovaný počet chýb, ktoré majú byť opravené.

$$k_{max} = n - m * t = 15 - 4 * 2$$
$$k_{max} = 7$$

Pre výpočet minimálnej Hammingovej vzdialenosti sa použije vzťah

$$d_{min} \geq 2 * t + 1 \,, \tag{3.7}$$

kde d_{min} predstavuje minimálnu Hammingovú vzdialenosť, t predstavuje vopred definovaný počet chýb, ktoré majú byť opravené.

$$d_{min} \geq 2 * t + 1$$
$$d_{min} \geq 5$$

Pre výpočet dĺžky kontrolného slova platí vzťah

$$r = n - k \,, \tag{3.8}$$

kde r predstavuje dĺžku kontrolného slova, n predstavuje dĺžku kódového slova a k predstavuje dĺžku informačného slova.

$$r = n - k = 15 - 7$$
$$r = 8$$

Určenie hodnoty generovaného polynómu kontrolného slova:

Je nutné na začiatok zadefinovať tvar polynómov, ktoré sa pre výpočet generovaného polynómu budú používať.[11]

$$ø_1(x) = ø_2(x) = ø_3(x) = x^4 + x + 1 \,, \tag{3.9}$$

$$ø_3(x) = x^4 + x^3 + x^2 + x + 1, \tag{3.10}$$

kde $ø(x)$ predstavuje elementárnu polynomickú funkciu závislú na hodnote x a x predstavuje korene polynómu.

Následne môžeme určiť hodnotu generovaného polynómu:

$$g(x) = lmc\big(ø_1(x), ø_2(x), \dots, ø_m(x)\big), \tag{3.11}$$

kde $g(x)$ predstavuje generovaný polynóm, závislý na hodnote x, $ø_1(x)$ predstavuje elementárne polynómy, závislé na hodnote x a x predstavuje korene polynómov.

$$g(x) = lmc\big(ø_1(x), ø_2(x), \dots, ø_m(x)\big)$$
$$g(x) = lmc\big(ø_1(x), ø_2(x), ø_3(x), ø_4(x)\big)$$

Ak je párny počet polynómov s rovnakými koreňmi v generovanom polynóme, zanedbávame ich hodnotu z výpočtu. V prípade ak je nepárny počet s rovnakými koreňmi v generovanom polynóme, pre následné výpočty použijeme len jeden, ostatné sa z výpočtu zanedbávajú.[2]

$$g(x) = lmc\big(ø_1(x), ø_3(x)\big)$$
$$g(x) = \big(ø_1(x) * ø_3(x)\big)$$
$$g(x) = (x^4 + x + 1) * (x^4 + x^3 + x^2 + x + 1)$$
$$g(x) = x^8 + x^7 + x^6 + 2 * x^5 + 3 * x^4 + 2 * x^3 + 2 * x^2 + 2 * x + 1$$

V prípade ak hodnota a z rovnice $a*x^i$, je párne daný prvok sa v generovanom polynóme vyskytovať nebude v prípade, ak je nepárna daný prvok sa v rovnici bude vyskytovať ako x^i. Hodnota a je reálne celé číslo, x predstavuje koreň polynómu, i je mocnina.[2]

Výsledný tvar generovaného polynómu kontrolného slova:

$$g(x) = x^8 + x^7 + x^6 + x^4 + 1$$

Premena informačného slova na polynóm sa spraví pomocou vzťahu

$$u(x) = \sum_{i=0}^{n-1} b_{ki} * x^i, \tag{3.12}$$

kde *u(x)* predstavuje polynóm vytvorený z informačného slova, b_{ki} predstavuje hodnotu bitu v informačnom slove, x^i predstavuje daný koreň polynómu.[11]

$$u(x) = 1 * x^5 + 0 * x^4 + 1 * x^3 + 1 * x^2 + 0 * x^1 + 1 * x^0$$

$$u(x) = x^5 + x^3 + x^2 + 1$$

Pre výpočet polynómu kontrolného slova je použitý vzťah

$$c(x) = u(x) * g(x), \tag{3.13}$$

kde *c(x)* predstavuje polynóm kódového slova, *u(x)* predstavuje polynóm informačného slova a *g(x)* predstavuje polynóm kontrolného slova.[2]

$$c(x) = u(x) * g(x)$$

$$c(x) = (x^5 + x^3 + x^2 + 1) * (x^8 + x^7 + x^6 + x^4 + 1)$$

$$c(x) = x^{13} + x^{12} + x^9 + x^5 + x^4 + x^3 + x^2 + 1$$

Výsledok:

Z toho vyplýva, že kódový vektor bude mať hodnotu w(n)=011001000111101.

Ako je badateľné z vyššie uvedeného príkladu, tento kód je univerzálny, to znamená, že v prípade zmeny počtu chýb vyskytujúcich sa v kódovom slove stačí zmeniť hodnotu kontrolného slova poprípade dĺžky kódového slova. Kód je vhodný pre opravu jednonásobnej ako aj viacnásobnej chyby. Počet opravených chýb, je závislý na dizajne generovaného polynómu g(x). BCH kód nedosahuje tak dobrého informačného pomeru, respektíve, ako je vidieť v príklade, obsahuje menší počet informačných bitov voči Hammingovmu kódu.

Ako je vidieť v Tabuľka 3.8 BCH kód je vhodným kódom pre riešenie problematiky tejto práce nakoľko nielen, že spĺňa zadanie, aby detekoval dvojnásobnú chybu a opravil jednonásobnú chybu, ale zadanie prevyšuje.

Tabuľka 3.8 Parametre BCH kódu

Typ kódu:	BCH kód	
	funkcia	hodnota
Parametre		
Minimálna vzdialenosť d_{min}	$w_{min}(u_x+u_y)$	$d_{min} \geq 5$
Dĺžka slova n	$n \leq 2^m - 1$	≥7
Počet informačných bitov k	$k \geq n - 2*m$	≥1
Informačný pomer R	$R = \frac{k}{n}$	≥0,143
Vlastnosti		
Počet detekovaných chýb		≥3
Počet opravených chýb		≥2
Oprava viacnásobných chýb		Áno

4. POROVNANIE VHODNÝCH KÓDOV

V predošlej kapitole sme si uviedli viacero typov kódov s praktickými príkladmi ich riešenia, ako aj parametrami a tým, či spĺňajú zadanie tejto práce. Zo všetkých kódov, ktoré boli popísanie zadaniu vyhovujú Rozšírený Hammingov kód a BCH kód. Z týchto kódov teraz vyberieme kód, ktorý bude následne použitý pre praktické riešenie tejto práce.

Vybraný kód by mal spĺňať tieto požiadavky:

1. Schopnosť detekovať dvojnásobnú chybu a opraviť jednonásobnú chybu
2. Čo najlepší informačný pomer R (pomer medzi dĺžkou informačného slova a kódového slova), tento parameter je dôležitý nakoľko je potrebou dosiahnutie čo najefektívnejšieho zaplnenia pamäte

Ako je badateľné z Tabuľka 4.1, oba kódy splňujú zadanie, avšak každý z nich má určité výhody ako aj nevýhody. Tieto vlastnosti budú rozobrané a následne bude vybraný vhodnejší z nich.

Tabuľka 4.1 Porovnanie parametrov Rozšíreného Hammingovho kódu a BCH kódu

Typ kódu:	Rozšírený Hammingov kód		BCH kód	
	funkcia	hodnota	funkcia	hodnota
Parametre				
Minimálna vzdialenosť dmin	wmin(ux+uy)	dmin =4	wmin(ux+uy)	dmin ≥5
Dĺžka slova n	n = 2r-1=k+r	≥8	$n \leq$ 2m - 1	≥7
Počet informačných bitov k	k = n-r	≥4	k ≥ n – 2*m	≥1
Informačný pomer R	$R = \frac{k}{n}$	≥0,5	$R = \frac{k}{n}$	≥0,143
Vlastnosti				
Počet detekovaných chýb		2		≥3
Počet opravených chýb		1		≥2
Oprava viacnásobných chýb		Nie		Áno

Medzi **výhody Rozšíreného Hammingovho kódu** patrí:

- Kód je pomerne jednoduchý z pohľadu pochopenia pre človeka, avšak čo sa týka návrhu kódu nedokážem vopred povedať, či je oproti BCH kódu efektívnejší z pohľadu rýchlosti, využitej plochy čipu.

- Má lepší informačný pomer oproti BCH kódu, takže v prípade totožných dát (informačných slov), ktoré budú zakódované, bude kódové slovo vytvorené pomocou Rozšíreného Hammingovho kódu zaberať oveľa menšiu časť pamäte.

Medzi **nevýhody Rozšíreného Hammingovho kódu** patrí:

-Tento kód nedokážeme modifikovať v prípade potreby zmeny počtu chýb, ktoré sa v kódovom slove môžu vyskytovať, takže v prípade potreby zmeny by bolo nutné zmeniť typ kódu použitého pre kodér a dekodér.

Medzi **výhody BCH kódu** patrí:

-Jednoduchá modifikovateľnosť v prípade zmeny počtu chýb, ktoré by mal kód detekovať, či opraviť.

-Možnosť opravy viacnásobných chýb.

Medzi **nevýhody BCH kódu** patrí:

-Horší informačný pomer oproti Rozšírenému Hammingovmu kódu, čo by v prípade použitia tohto kódu spôsobovalo využitia väčšej časti pamäte.

-Zložitejšie pochopenie návrhu kódu z pohľadu programátora, ako aj zložitejší návrh kódu, avšak čo sa týka následnej implementácie nedokážem povedať, či by bol kód zložitejší alebo nie.

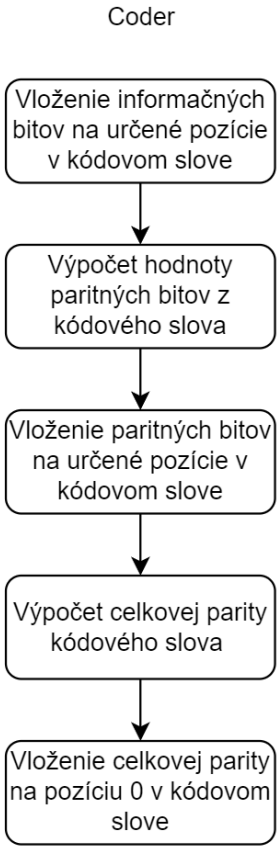
Na základe vyššie uvedených informácií je výhodnejšie použitie Rozšíreného Hammingovho kódu, pretože pre zadané požiadavky má výhodnejšie parametre, ako napríklad informačný pomer, ktorý je pre Rozšírený Hammingov kód $R \geq 0,5$, oproti tomu BCH kód má informačný pomer $R \geq 0,143$. Samozrejme BCH kód má mnoho predností, v ktorých prevyšuje Rozšírený Hammingov kód, no tieto prednosti sú pre tento prípad nepotrebné.

5. REALIZÁCIA HAMMINGOVHO KÓDU

Na základe teórie uvedenej v predchádzajúcich kapitolách bol pre vytvorenie kódu zvolený Hammingov kód. Bol zvolený postup „od zdola na hor“, čo znamená, že prvým krokom bolo vytvorenie konceptu kodéru a dekodéru, ktoré boli následne spojené a bola testovaná ich funkčnosť. Následne bol vytvorený radič pre komunikáciu medzi užívateľom a pamäti SRAM a vnútornej kontrolu pamäte. Nakoniec bol spravený celkový test funkčnosti kódu. Jednotlivé moduly ako aj ich testy budú rozobrané nižšie. V rámci návrhu boli definované konštanty pre jednoduchšie ovládanie užívateľa, tieto konštanty sú v samostatné slohe.

5.1 Kodér

Jedná sa o modul, ktorého funkcia je vysvetlená v kapitole 0, je vytvorený čisto pomocou kombinačnej logiky. Postup vytvorenia kódového slova (vektor na výstupe kodéru) z informačného slova (vektor na vstupe kodéru) je zobrazený na Obrázok 5-1, tento obrázok slúži len na popis postupu vytvorenia kódového slova a lepšie pochopenie funkcie modulu, v realite sa jedná čisto o kombinačnú logiku.

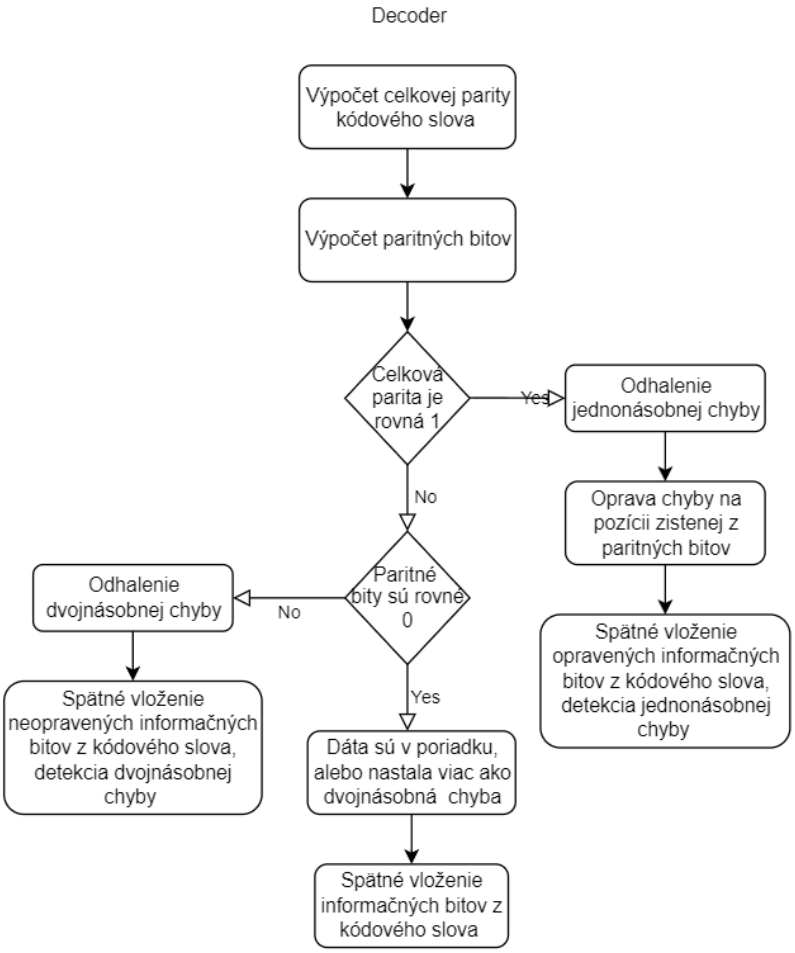


Obrázok 5-1 Ukážka postupu vytvorenia kódového slova [app.diagrams.net]

Kde prvým krokom je priradenie pozície informačným bitom, pričom pozícia je nemenná a je definovaná na základe ukážky v kapitola 0, následne sa z informačných bitov obsadených z kódového slova vypočíta parita, tento proces sa vykonáva v určitých skupinách bitov, ktoré sú pevne, následne po určení hodnoty paritných bitov sa tieto bity obsadia do kódového slova na definované pozície. Vykoná sa výpočet celkovej parity kódového slova (informačných bitov a paritných bitov), táto hodnota je dosadená na pozíciu 0 v kódovom slove.

5.2 Dekodér

Jedná sa o modul vykonávajúci opätovné vytvorenie informačného slova (slovo na výstupe) z kódového slova (slovo na vstupe). Tento modul obsahuje detektor chyby, ktorý dokáže detekovať dvojnásobnú chybu a zároveň opraviť jednonásobnú chybu. Jedná sa o reverzný modul voči kodéru. Popis vytvorenia informačného slova z kódového slova je zobrazený na Obrázok 5-2, tento popis slúži len pre lepšie pochopenie procesu dekódovania v realite sa jedná čisto o kombinačnú logiku.



Obrázok 5-2 Ukážka postupu dekódovania informačného slova [app.diagrams.net]

Kde prvým krokom je výpočet celkovej parity kódového, nasleduje výpočet paritných bitov. Následne sa vyhodnotí celková parita:

- Ak je hodnota rovná 1 je detekovaná jednonásobná chyba, v takomto prípade sa chyba nachádza na pozícii definovanej paritnými bitmi, detekovaná chyba je opravená a z opraveného kódového slova je opätovne vytvorené informačné slovo, zároveň je oznámené užívateľovi, že v kódovom slove bola detekovaná chyba, ktorá bola opravená.

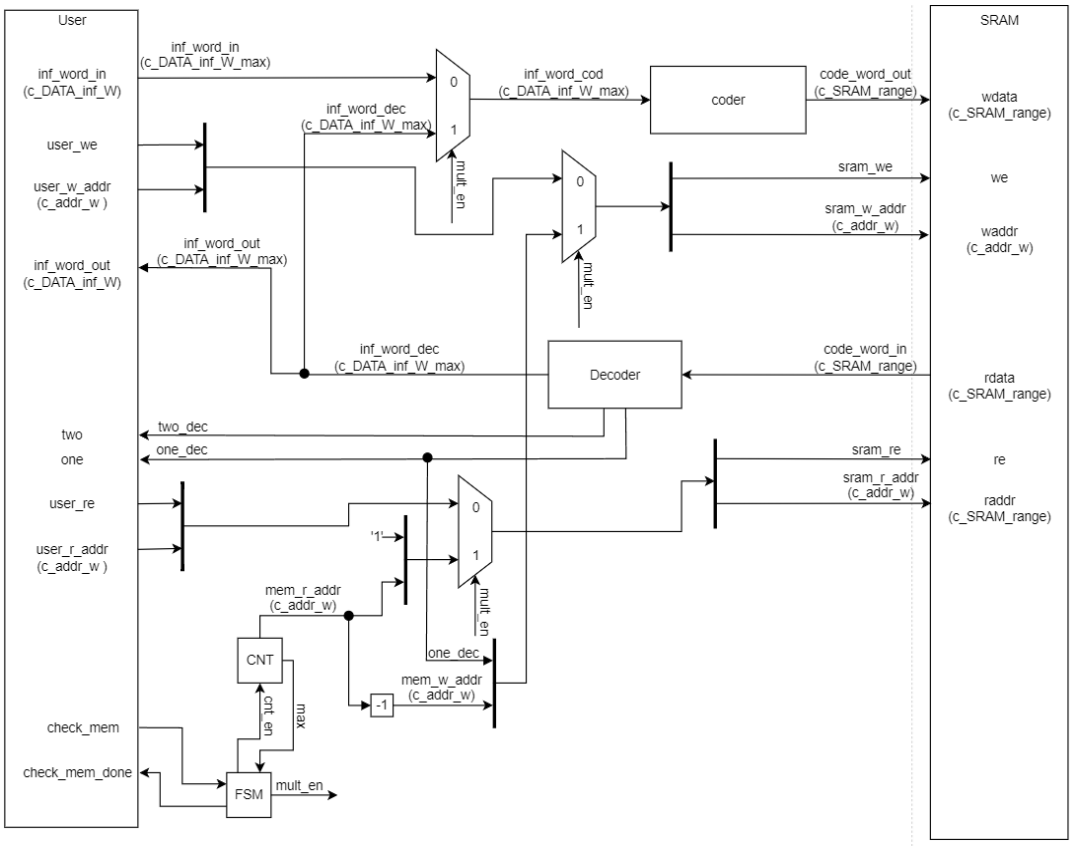
-Ak je hodnota rovná 0 jedná sa o prípad výskytu dvojnásobnej chyby alebo sa v kódovom slove nevyskytuje žiadna chyba, čo je vyhodnotené na základe hodnoty paritných bitov:

- Ak je hodnota paritných bitov rovná 0 jedná sa o prípad, kedy sa v kódovom slove nenachádza žiadna chyba alebo je viac ako dvojnásobná, čo tento kód nedokáže detekovať, následne je z kódového slova opätovne vytvorené informačné slovo.

-Ak je hodnota paritných bitov rozdielna od 0 jedná sa o prípad detekcie dvojnásobnej chyby, v takomto prípade kód chybu nedokáže opraviť, a tak je opätovne vytvorené informačné slovo z kódového slova, a zároveň je užívateľovi ohlásený výskyt dvojnásobnej chyby.

5.3 Radič pro automatickou opravu dát v pamäti

Jedná sa o najvyšší vrstvu v rámci návrhu, ktorej funkciou je vytvorenie prepojení medzi užívateľom a SRAM a zároveň vnútorného prepojenia pre prípad vnútornej kontroly dát uložených v SRAM. Jej schéma je zobrazené na Obrázok 5-3.



Obrázok 5-3 Prepojenie prostredia užívateľa s prostredím SRAM [app.diagrams.net]

Na základe hodnoty signálu mult_en je rozhodnutý príjem dát a to:

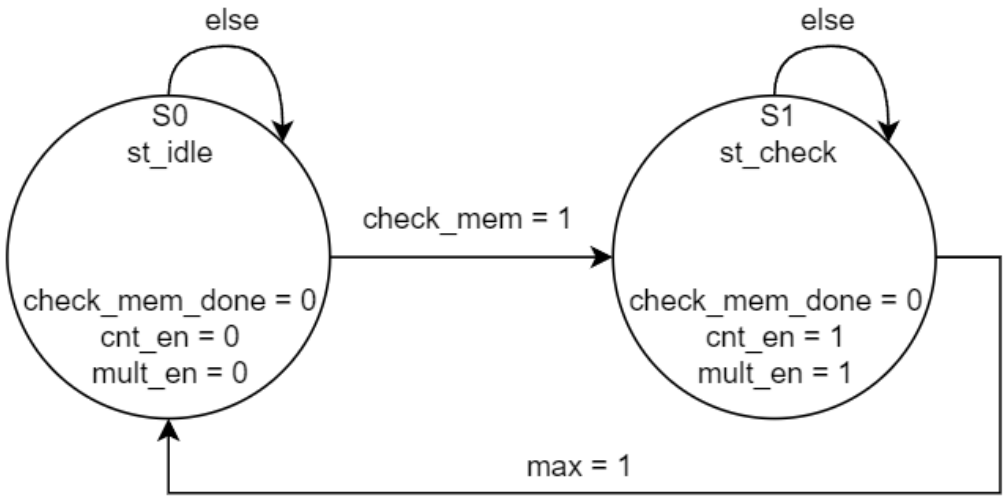
-Ak je hodnota mult_en rovná 0 jedná sa o stav, kedy má prístup do pamäte používateľ. Užívateľ posiela informačné slovo cez do kodéru, z ktorého je následne uložený do SRAM na pozíciu, ktorú zadá užívateľ pomocou vstupu user_w_addr a povolením zápisu do SRAM user_we. Následne pri potrebe prečítania dát užívateľ zadá adresu pomocou user_r_addr a povolením čítania z SRAM user_re. Následne sa vyčtu dáta, ktoré sa cez kodér prekontrolujú a pošlu na výstup užívateľovi.

-Ak je hodnota mult_en rovná 1 tak je spustená vnútorná kontrola dát v pamäti SRAM. Prvým krokom je prečítanie dát na adrese, ktorá je získaná z počítadla následne sa vykoná kontrola dát. Ak je odhalená jednonásobná chyba vykoná sa jej oprava následne získané informačné slovo sa pošle do kodéru, kde sa vytvorí nové kódové slovo (opravené), ktoré nahradí vyčítané poškodené kódové slovo. Ak je odhalená dvojnásobná chyba alebo nie je odhalená žiadna chyba dekodér len zaregistruje tento stav (nastala dvojnásobná chyba, ktorú len nahlási avšak ju nie je schopný opraviť alebo

nenastala žiadna chyba a nehlási nič) , nedochádza k prepisu dát. Medzi adresu čítania a zápisu je vložené oneskorenie o jeden hodinový cyklus, čo je spôsobené implementovanou SRAM, ktorej zápis aj čítanie je riadené hodinovým signálom.

V rámci návrhu bola spracovaná aj možnosť, kedy rozsah vstupného informačného slova neodpovedá maximálnej hodnote zvoleného rozsahu informačného slova, nakoľko sa jedná o definované hodnoty, ktorých príklad výpočtu je uvedený v kapitola 0 príkladom je napríklad pre vstupné slovo v podobe ASCII znaku (8-bitový vektor) je hodnota vstupného informačného slova prevedena na 11-bitový vektor. V takomto prípade je kód ošetrený a na vrchné neobsadené bity (bit 11-9) dosadí hodnotu 0, zároveň pri čítaní dát je informačné slovo orezané na pôvodnú dĺžku. Rovnako je spracovaná aj možnosť rozdielnej dĺžky kódového slova voči šírke slova pre zápis do SRAM, kde v prípade nerovnosti dĺžok vektorov je na vrchné neobsadené bity dosadená hodnota 0.

Stavový automat, jedná sa o časť radiče vo vrstve komunikácie medzi užívateľom a SRAM, ktorá slúži pre ovládanie priority slova medzi užívateľom a kontrolou SRAM, resp. sa jedná o časť kódu, ktorá prideľuje slovo. Priradenie slova je ovládané užívateľom. Jeho stavový diagram je znázornenie na Obrázok 5-4.



Obrázok 5-4 Diagram stavov stavového automatu [app.diagrams.net]

Stavový automat má dva stavy. Prvým stavom S0 je prípad, keď má prístup užívateľ. Druhý stav S1 nastáva v moment, kedy je zaznamenaný požiadavok na kontrolu správnosti dát v pamäti dochádza k povoleniu čítača signálom cnt_en, pričom čítač slúži pre posun v adresnom rozsahu SRAM. V priebehu kontroly dát nie je možné prepnúť stav do stavu S0, to nastáva až v moment, keď je zaznamenané prekontrolovanie dát, ktoré je signalizované pomocou výstupného bitu check_mem_done, ktorého hodnota sa na jeden hodinový takt zmení do 1 a tým užívateľovi signalizuje koniec kontroly dát uložených v pamäti a automaticky prechádza do stavu S0.

6. VERIFIÁCIE NÁVRHU

V tejto kapitolu budú popísané jednotlivé testy, ktoré boli vykonané pre overenie funkčnosti návrhu. Jedná sa o testy overenia funkčnosti modulov kodéru a dekodéru a následne overenia funkčnosti radiče pamäti SRAM.

6.1 Verifikácia Kodéru a Dekodéru

V rámci verifikácie boli vykonané tri testy, pričom test prebiehal postupným naplnením informačného slova od 0 do maximálnej hodnoty určenej rozsahom informačného slova. Rozdiel jednotlivých testov spočíval vo vytváraní chýb v kódovom slove pred jeho dekódovaním. Výstup z kodéru bol prepojený na vstup dekodéru. Jednotlivé typy testov budú rozobrané nižšie.

Prvým testom bolo overenie toho, či informačné slovo na vstupe odpovedá informačnému slovu na výstupe, týmto testom sa overila správnosť vytvorenia kódového slova v kodéry a následne spätné zloženie informačného slova v dekodéry. V rámci testu bola vytvorená chybová hláška. V prípade ak sa informačné slovo zapísané užívateľom nerovná informačnému slovu získanému z dekodéru je vyhlásená chyba. Výsledky testu sú zobrazené v Tabuľka 6.1.

Tabuľka 6.1 Kontrola správnosti funkcie zápisu a výpisu informačného slova

Kontrola prenosu informačného slova pomocou zakódovania (Coder) a následneho dekódovania (Decoder)					
Dĺžka informačného slova	Rozsah informačného slova	Dĺžka kódového slova	Dĺžka kontrolného slova	Overenie funkčnosti	Čas v simulácii
[-]	[-]	[-]	[-]	[-]	[s]
1	1	3	4	splnené	4,00E-08
2	2 až 4	8	5	splnené	8,00E-08
3	2 až 4	8	5	splnené	1,60E-07
4	2 až 4	8	5	splnené	3,20E-07
5	5 až 11	16	6	splnené	6,50E-07
6	5 až 11	16	6	splnené	1,30E-06
7	5 až 11	16	6	splnené	2,60E-06
8	5 až 11	16	6	splnené	5,11E-06
9	5 až 11	16	6	splnené	1,02E-05
10	5 až 11	16	6	splnené	2,05E-05
11	5 až 11	16	6	splnené	4,10E-05
12	12 až 26	32	7	splnené	8,19E-05
13	12 až 26	32	7	splnené	1,64E-04
14	12 až 26	32	7	splnené	3,28E-04
15	12 až 26	32	7	splnené	6,55E-04
16	12 až 26	32	7	splnené	1,31E-03
17	12 až 26	32	7	splnené	2,62E-03
18	12 až 26	32	7	splnené	5,24E-03
19	12 až 26	32	7	splnené	1,05E-02
20	12 až 26	32	7	splnené	2,10E-02
21	12 až 26	32	7	splnené	4,19E-02
22	12 až 26	32	7	splnené	8,39E-02
23	12 až 26	32	7	splnené	1,68E-01
24	12 až 26	32	7	splnené	3,36E-01
25	12 až 26	32	7	splnené	6,71E-01
26	12 až 26	32	7	splnené	1,34

Následne bol vykonaný test, pri ktorom bola pozmenená hodnota kódového slova na výstupe z kodéru jednalo sa o maskovanie jedného bitu v kódovom slove, pričom chybný bit sa menil pomocou rotácie v chybovom vektore, takto bola overená funkčnosť dekodéru v celom rozsahu kódového slova pre odhalenie jednonásobnej chyby. Pre overenie funkčnosti boli vytvorené chybové hlášky v prípade ak nie je detekovaná chyba alebo hodnota informačného slova na výstupe nie je zhodná s hodnotou informačného slova na vstupe je ohlásená chyba, ktorá nenastala v žiadnom z overených rozsahov informačného slova. Jednotlivé výsledky testu sú uvedenie v Tabuľka 6.2.

Tabuľka 6.2 Kontrola správnosti funkcie detekcie a opravenia jednonásobnej chyby

Vloženie jednej chyby do informačného slova pre kontrolu detekcie jednonásobnej chyby a jej opravu					
Dĺžka informačného slova	Rozsah informačného slova	Dĺžka kódového slova	Dĺžka kontrolného slova	Overenie funkčnosti	Čas v simulácii
[-]	[-]	[-]	[-]	[-]	[s]
1	1	3	4	splnené	1,60E-07
2	2 až 4	8	5	splnené	6,40E-07
3	2 až 4	8	5	splnené	1,28E-06
4	2 až 4	8	5	splnené	2,56E-06
5	5 až 11	16	6	splnené	1,02E-05
6	5 až 11	16	6	splnené	2,05E-05
7	5 až 11	16	6	splnené	4,10E-05
8	5 až 11	16	6	splnené	8,19E-05
9	5 až 11	16	6	splnené	1,20E-04
10	5 až 11	16	6	splnené	3,28E-04
11	5 až 11	16	6	splnené	6,55E-04
12	12 až 26	32	7	splnené	2,62E-03
13	12 až 26	32	7	splnené	4,90E-03
14	12 až 26	32	7	splnené	1,05E-02
15	12 až 26	32	7	splnené	2,10E-02
16	12 až 26	32	7	splnené	4,19E-02
17	12 až 26	32	7	splnené	8,39E-02
18	12 až 26	32	7	splnené	1,68E-01
19	12 až 26	32	7	splnené	0,34
20	12 až 26	32	7	splnené	0,67
21	12 až 26	32	7	splnené	1,34
22	12 až 26	32	7	splnené	2,68
23	12 až 26	32	7	splnené	5,37
24	12 až 26	32	7	splnené	10,74

Posledným testom bolo overenie detekcie dvojnásobnej chyby, pričom bola menená hodnota kódového slova na výstupe z kodéru, jednalo sa o maskovanie pomocou dvoj bitovej chyby v celom rozsahu kódového slova, pričom chyba nebola statická, takže bola overená každá možná kombinácia dvojica chybných bitov. V rámci testu bola vytvorená chybová hláška v prípade detekcie dvojnásobnej chyby v dekodéry musí byť tento stav ohlásený v prípade ak by nebol ohlásený je detekovaná chyba kódu, ktorá v rámci testu nebola detekovaná. Jednotlivé výsledky sú zobrazené v

Tabuľka 6.3 Kontrola správnosti funkcie detekcie dvojnásobnej chyby

Vloženie dvoch chýb do informačného slova pre kontrolu detekcie dvojnásobnej chyby					
Dĺžka informačného slova	Rozsah informačného slova	Dĺžka kódového slova	Dĺžka kontrolného slova	Overenie funkčnosti	Čas v simulácii
[-]	[-]	[-]	[-]	[-]	[s]
1	1	3	4	splnené	2,40E-07
2	2 až 4	8	5	splnené	2,22E-06
3	2 až 4	8	5	splnené	4,48E-06
4	2 až 4	8	5	splnené	8,94E-06
5	5 až 11	16	6	splnené	7,68E-05
6	5 až 11	16	6	splnené	1,54E-04
7	5 až 11	16	6	splnené	3,07E-04
8	5 až 11	16	6	splnené	6,14E-04
9	5 až 11	16	6	splnené	1,23E-03
10	5 až 11	16	6	splnené	2,46E-03
11	5 až 11	16	6	splnené	5,00E-03
12	12 až 26	32	7	splnené	3,50E-02
13	12 až 26	32	7	splnené	8,13E-02
14	12 až 26	32	7	splnené	1,56E-01
15	12 až 26	32	7	splnené	3,15E-01
16	12 až 26	32	7	splnené	0,63
17	12 až 26	32	7	splnené	1,26
18	12 až 26	32	7	splnené	2,52
19	12 až 26	32	7	splnené	5,04
20	12 až 26	32	7	splnené	10,08

Rozdielna dĺžka slova, ktorá bola overená v daných rozsahoch je zapríčinená časom, ktorý daný test zabral v reálnom čase nakoľko uvedený čas v simulácii odpovedá času po ktorý dochádzalo k zmene hodnoty informačného slova pre overenie každej možnej kombinácie, a nie reálnemu času, ktorý daný test zabral v reálnom čase.

6.2 Verifikácia radiče prístupu do pamäti

Pre overenie funkčnosti bol vytvorený test, ktorého hlavnou úlohou bolo overiť zhodu vstupných a výstupných informačných vektorov, overenie funkcie kontroly pamäte a overenie opravy ako aj detekcie chyby.

Prvým krokom v testu bolo naplnenie pamäte SRAM v zvolenom rozsahu vygenerovanými dátami resp. na určenú adresu bola zapísaná hodnota adresy v binárnej podobe. Týmto krokom bola overená správnosť komunikácie medzi užívateľom a SRAM.

Druhým krokom bola vykonaná kontrola zapísaných dát, pri ktorej bolo kontrolované, či uložené v SRAM odpovedajú dátam, ktoré sa na danej adrese mali nachádzať, zároveň bola overená komunikácia na ceste SRAM – dekoder ako aj vyhodnotenie, že dáta sú správne resp. nenachádza sa v nich žiadna chyba.

Tretím krokom bolo čítanie dát uložených v SRAM užívateľom, pri ktorej bolo vytvorené počítadlo, ktoré slúžilo na porovnanie získanej hodnoty informačného slova z SRAM voči generovanému slovu, ktoré odpovedalo pôvodnej správnej hodnote, v prípade ak by sa tieto hodnoty nerovnali bola by registrovaná chybová hláška, ktorá však v tomto úseku testu nebola detekovaná.

Štvrtým krokom bolo naplnenie pamäte SRAM v zvolenom rozsahu vygenerovanými dátami resp. na určenú adresu bola zapísaná hodnota adresy v binárnej podobe. Následne pred zápisom dát do SRAM bola hodnota kódového slova vymaskovaná pomocou jednobitovej chyby, ktorá ale nebola statická a menila sa v celom rozsahu kódového slova.

Piatym krokom bola vykonaná kontrola zapísaných dát, pri ktorej bola sledovaná detekcia jednonásobnej chyby pri dekódovaní, detekciou jednonásobnej chyby ako bolo vyššie uvedené bol sprístupnený prepis hodnoty poškodeného kódového slova na opravené kódové slovo.

Šiestym krokom bolo čítanie dát uložených v SRAM užívateľom, pri ktorej bolo vytvorené počítadlo, ktoré slúžilo na porovnanie získanej hodnoty informačného slova z SRAM voči generovanému slovu, ktoré odpovedalo pôvodnej správnej hodnote, v prípade ak by sa tieto hodnoty nerovnali bola by registrovaná chybová hláška, ktorá však v tomto úseku testu nebola detekovaná. Taktiež sa vykonávala kontrola či sú dáta zapísané na správnej adrese (overenie potreby oneskorenia medzi adresou čítania a zápisu)

Siedmim krokom bolo naplnenie pamäte SRAM v zvolenom rozsahu vygenerovanými dátami resp. na určenú adresu bola zapísaná hodnota adresy v binárnej podobe. Následne pred zápisom dát do SRAM bola hodnota kódového slova vymaskovaná pomocou dvojbitovej chyby, ktorá ale nebola statická a menila sa v celom rozsahu kódového slova.

Ôsmim krokom bola vykonaná kontrola zapísaných dát, pri ktorej bola sledovaná detekcia dvojnásobnej chyby pri dekódovaní, detekciou dvojnásobnej chyby overené, že v prípade výskytu chyby je chyba detekovaná aj keď nie je opravená.

Deviatym krokom bolo čítanie dát uložených v SRAM užívateľom, pri ktorej bolo vytvorené počítadlo, ktoré slúžilo na porovnanie získanej hodnoty informačného slova z SRAM voči generovanému slovu, ktoré odpovedalo pôvodnej správnej hodnote, v prípade ak by sa tieto hodnoty rovnali bola by registrovaná chybová hláška, ktorá však v tomto úseku testu nebola detekovaná. Bolo overené, že kód nedokáže opraviť dvojnásobnú chybu avšak počas čítania bola detekovaná dvojnásobná chyba v dekodéry, ktorá slúži pre upozornenie užívateľa na chybnosť uložených dát. Taktiež sa vykonávala kontrola či sú dáta zapísané na správnej adrese (overenie potreby oneskorenia medzi adresou čítania a zápisu).

7. VÝSLEDKY IMPLEMENTÁCIE

V tejto kapitole sú zobrazené potrebné zdroje, ktoré sú potrebné pre implementáciu programu. Na implementáciu programu bolo použité FPGA Spartan-3. Nakoľko je možné použiť čisto moduly kodéra a dekodéra bez radiče pamäti je zahrnutá aj implementácia jednotlivých modulov. Tabuľky reflektujú minimálne požiadavky na výsledný obvod ako aj obsadenie pre Spartan-3 v percentách, taktiež maximálnu frekvenciu jedná sa o parametre vypočítané programom, takže sa od reálnych môžu líšiť.

Tabuľka 7.1 Využité zdroje radiče pamäti pri syntéze pre Spartan-3

Dĺžka informačného slova	Rozsah informačného slova	Dĺžka kódového slova	Dĺžka kontrolného slova	Počet zdrojov registrov	Počet 4 vstupových náhľadových tabuliek	Maximálne oneskorenie na kombinačnej trase	Maximálna frekvenica
[-]	[-]	[-]	[-]	[-]	[-]	[ns]	[MHz]
1	1	3	4	20(0%)	104(2%)	10,675	93,677
2	2 až 4	8	5	20(0%)	131(3%)	17,562	56,941
3	2 až 4	8	5	20(0%)	132(3%)	16,512	60,562
4	2 až 4	8	5	20(0%)	128(3%)	17,572	56,909
5	5 až 11	16	6	20(0%)	174(4%)	21,028	47,556
6	5 až 11	16	6	20(0%)	172(4%)	21,861	45,744
7	5 až 11	16	6	20(0%)	172(4%)	22,086	45,278
8	5 až 11	16	6	20(0%)	172(4%)	22,459	44,526
9	5 až 11	16	6	20(0%)	172(4%)	22,459	44,526
10	5 až 11	16	6	20(0%)	172(4%)	22,459	44,526
11	5 až 11	16	6	20(0%)	172(4%)	23,967	41,724
12	12 až 26	32	7	20(0%)	287(7%)	27,852	35,904
13	12 až 26	32	7	20(0%)	289(7%)	27,852	35,904
14	12 až 26	32	7	20(0%)	291(7%)	27,852	35,904
15	12 až 26	32	7	20(0%)	296(7%)	27,666	36,145
16	12 až 26	32	7	20(0%)	296(7%)	27,437	36,447
17	12 až 26	32	7	20(0%)	295(7%)	26,835	37,265
18	12 až 26	32	7	20(0%)	293(7%)	26,855	37,237
19	12 až 26	32	7	20(0%)	295(7%)	27,53	36,324
20	12 až 26	32	7	20(0%)	297(7%)	27,53	36,324
21	12 až 26	32	7	20(0%)	274(7%)	31,274	31,975
22	12 až 26	32	7	20(0%)	272(7%)	31,178	32,074
23	12 až 26	32	7	20(0%)	272(7%)	31,178	32,074
24	12 až 26	32	7	20(0%)	272(7%)	31,324	31,924
25	12 až 26	32	7	20(0%)	272(7%)	31,223	32,028
26	12 až 26	32	7	20(0%)	273(7%)	30,952	32,308

Ako je viditeľné z Tabuľka 7.1 výsledný kód nezaťažuje obvod nakoľko využíva minimálne zdroje, avšak je potrebné počítať s časovým oneskorením pre jednotlivé smery (Užívateľ – SRAM alebo SRAM – Užívateľ) reálne neodpovedá jednému hodinovému impulzu pre dĺžku kódového slova rovnú alebo väčšiu ako 8 bit.

Tabuľka 7.2 Využité zdroje pre modul dekodéru pri syntéze pre Spartan-3

Dĺžka informačného slova	Rozsah informačného slova	Dĺžka kódového slova	Dĺžka kontrolného slova	Počet 4 vstupových náhľadových tabuliek	Maximálne oneskorenie na kombinačnej trase	Maximálna frekvenica
[-]	[-]	[-]	[-]	[-]	[ns]	[MHz]
1	1	3	4	3(0%)	9,063	110,339
2	2 až 4	8	5	20(0%)	14,314	69,862
3	2 až 4	8	5	20(0%)	14,314	69,862
4	2 až 4	8	5	20(0%)	14,314	69,862
5	5 až 11	16	6	44(1%)	16,912	59,130
6	5 až 11	16	6	44(1%)	16,912	59,130
7	5 až 11	16	6	44(1%)	16,912	59,130
8	5 až 11	16	6	44(1%)	16,912	59,130
9	5 až 11	16	6	44(1%)	16,912	59,130
10	5 až 11	16	6	44(1%)	16,912	59,130
11	5 až 11	16	6	44(1%)	16,912	59,130
12	12 až 26	32	7	112(2%)	21,35	46,838
13	12 až 26	32	7	112(2%)	21,35	46,838
14	12 až 26	32	7	112(2%)	21,35	46,838
15	12 až 26	32	7	112(2%)	21,35	46,838
16	12 až 26	32	7	112(2%)	21,35	46,838
17	12 až 26	32	7	112(2%)	21,35	46,838
18	12 až 26	32	7	112(2%)	21,35	46,838
19	12 až 26	32	7	112(2%)	21,35	46,838
20	12 až 26	32	7	112(2%)	21,35	46,838
21	12 až 26	32	7	112(2%)	21,35	46,838
22	12 až 26	32	7	112(2%)	21,35	46,838
23	12 až 26	32	7	112(2%)	21,35	46,838
24	12 až 26	32	7	112(2%)	21,35	46,838
25	12 až 26	32	7	112(2%)	21,35	46,838
26	12 až 26	32	7	112(2%)	21,35	46,838

Ako je viditeľné z Tabuľka 7.2 samotný modul dekodéru mimo vrchnú vrstvu zaberá omnoho menej zdrojov, čo vyplýva z toho, že sa jedná len o časť kódu, taktiež pre maximálne oneskorenie, dosahuje väčšiu frekvenciu, takže je možné počítať s menším oneskorením, resp. je možné počítať s jedným hodinovým impulzom na ceste medzi SRAM a užívateľom pre dĺžku kódového slova do 16 bit.

Tabuľka 7.3 Využité zdroje pre modul kodéru pri syntéze pre Spartan-3

Dĺžka informačného slova	Rozsah informačného slova	Dĺžka kódového slova	Dĺžka kontrolného slova	Počet 4 vstupových náhľadových tabuliek	Maximálne oneskorenie na kombinačnej trase	Maximálna frekvenica
[-]	[-]	[-]	[-]	[-]	[ns]	[MHz]
1	1	3	4	-	7,382	135,465
2	2 až 4	8	5	4(0%)	9,073	110,217
3	2 až 4	8	5	4(0%)	9,073	110,217
4	2 až 4	8	5	4(0%)	9,073	110,217
5	5 až 11	16	6	10(0%)	10,537	94,904
6	5 až 11	16	6	10(0%)	10,537	94,904
7	5 až 11	16	6	10(0%)	10,537	94,904
8	5 až 11	16	6	10(0%)	10,537	94,904
9	5 až 11	16	6	10(0%)	10,537	94,904
10	5 až 11	16	6	10(0%)	10,537	94,904
11	5 až 11	16	6	10(0%)	10,537	94,904
12	12 až 26	32	7	10(0%)	10,537	94,904
13	12 až 26	32	7	36(0%)	15,05	66,445
14	12 až 26	32	7	36(0%)	15,05	66,445
15	12 až 26	32	7	36(0%)	15,05	66,445
16	12 až 26	32	7	36(0%)	15,05	66,445
17	12 až 26	32	7	36(0%)	15,05	66,445
18	12 až 26	32	7	36(0%)	15,05	66,445
19	12 až 26	32	7	36(0%)	15,05	66,445
20	12 až 26	32	7	36(0%)	15,05	66,445
21	12 až 26	32	7	36(0%)	15,05	66,445
22	12 až 26	32	7	36(0%)	15,05	66,445
23	12 až 26	32	7	36(0%)	15,05	66,445
24	12 až 26	32	7	36(0%)	15,05	66,445
25	12 až 26	32	7	36(0%)	15,05	66,445
26	12 až 26	32	7	36(0%)	15,05	66,445

Ako je vidieť z Tabuľka 7.3 samotný modul kodéru zaberá omnoho menej zdrojov a dosahuje aj lepšiu frekvenciu nakoľko sa jedná len o modul, ktorý je súčasťou celého kódu, taktiež dosahuje lepšie parametre oproti dekodéru, čo je zapríčinené menšou zložitosťou oproti nemu. V celom kontrolovanom rozsahu je možné počítať s využitím jedného hodinového impulzu ako oneskorenia na trase medzi užívateľom a SRAM.

ZÁVER

Na základe požiadaviek, ktoré sú uvedenie v zadaní, je v prvých troch kapitolách rozobraná základná teória k samo opravným kódom, s ukážkami niekoľkých detekčných kódov pre lepšie pochopenie nasledujúcich samo opravných kódov. Z uvedených kódov v týchto kapitolách splňujú požiadavky len Rozšírený Hammingov kód a BCH.

V štvrtej kapitole je porovnanie vhodných kódov, ich výhody a nevýhody. Následne je tu rozobraný dôvod výberu Rozšírený Hammingov kód, v podstate je vybraný pre jeho výhodnejšie parametre ako napríklad informačný pomer.

V piatej kapitole je rozobraný postup návrhu kódu. Poradie podkapitol odpovedá postupu návrhu, bola zvolená metóda „od zdola na hor“, čo znamená, že najskôr boli vytvorené moduly kodéru a dekodéru až následne vrchná logika pre komunikáciu medzi užívateľom a SRAM.

V šiestej kapitole sú rozobrané testy, ktoré boli vytvorené pre overenie funkčnosti jednotlivých modulov ako aj celého kódu. V rámci overenia funkčnosti kodéru a dekodéru boli spravené tri testy. Prvý overuje správnosť vytvorenia kódového slova z informačného slova a opätovné vytvorenie informačného slova z kódového slova. Druhý overuje prípad výskytu jednonásobnej chyby, pričom je skúmané, či dekodér jednonásobnú chybu detekuje a je schopný ju opraviť. Tretí overuje výskyt dvojnásobnej chyby, pričom tento kód nie je schopný dvojnásobnú chybu opraviť avšak je schopný ju detekovať. Následne je overená funkčnosť komunikačného rozhrania medzi užívateľom a SRAM, pričom sa jedná o rozsiahly test, ktorý obsahuje spomenuté overenia opravy a detekcie chyby, ako aj priradenie slova pre zápis dát užívateľom a vnútornej kontroly obsahu pamäti SRAM.

V poslednej kapitole sú rozobrané dosiahnuté výsledky syntézy, pričom pre syntézu bol zvolený obvod FPGA Spartan-3. Údaje uvedené v tejto kapitole sú vypočítané programom, takže od reálnych hodnôt sa môžu líšiť, taktiež boli uvedené potrebné zdroje pre jednotlivé moduly nakoľko v prípade potreby využitia len modulov je možné vrstvu top vynechať poprípade nahradiť, či modifikovať.

LITERATÚRA

[1] *HLAVIČKA, J. Číslicové systémy odolné proti poruchám. 1. vyd. Praha: ČVUT, 1992, 330 s. ISBN 80-01-00852-5. [cit. 2021-11-14]*

[2] ADÁMEK, J. Kódování. 1. Vydání. Praha: SNTL, 1989

[3] ZIEMER, RodgerE a Wiliam H. TRANTER. Principles of communication: systems, modulation, and noise. *WorldCat* [online]. Hoboken, New Jersey, 2015 [cit. 2021-11-14]. Dostupné z: https://www.worldcat.org/title/principles-of-communication-systems-modulation-and-noise/oclc/856647730

[4] VAN LINT, J. H. Introduction to Coding Theory. *INTERNET ARCHIVE* [kniha]. 1998 [cit. 2021-11-21]. Dostupné z: https://archive.org/details/introductiontoco0000lint_a3b9

[5] Bill. An Algorithm for Error Correcting Cyclic Redundance Checks. *Dr.Dobb´s* [online]. University of Maryland, 2003 [cit. 2021-11-21]. Dostupné z: http://www.drdobbs.com/an-algorithm-for-error-correcting-cyclic/184401662

[6] MUSTAFA, Ergen. 2.3.3 Error Detection Coding. *Springer Link* [online]. 1998 [cit. 2021-11-21]. Dostupné z: https://link.springer.com/chapter/10.1007%2F978-0-387-68192-4_2

[7] GILBERT, E. N. Capacity of a burst-noise channel. *IEEE Xplore* [online]. 1960 [cit. 2021-11-21]. Dostupné z: https://link.springer.com/chapter/10.1007%2F978-0-387-68192-4_2

[8] BEMENT, Arden L., BOND, Philip J., ed. *SECURITY REQUIREMENTS FOR CRYPTOGRAPHIC MODULES* [online]. Washington, 1960 [cit. 2021-11-21]. Dostupné z: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-2.pdf

[9] GLOVER, Neal a Trent DUDLEY. *Practical Error Correction Design For Engineers* [kniha]. Colorado USA, 1990 [cit. 2021-11-21].

[10] HAMMING, Richard Wesley. Error detecting and error correcting codes. *IEEEXplore* [online]. New Jersey, 1950 [cit. 2021-12-09]. Dostupné z: https://ieeexplore.ieee.org/document/6772729/authors#authors

[11] BOSE, R. C. a D. K. RAY-CHAUDHURI. ON A CLASS OF ERROR CORRECTING BINARY GROUP CODES. *Ncsu.edu* [online]. Washington D. C., 1959 [cit. 2021-12-09]. Dostupné z: https://repository.lib.ncsu.edu/bitstream/handle/1840.4/2137/ISMS_1959_240.pdf;jsessionid=8A6974D04B863B54755C1BC888CF97AC?sequence=1

SEZNAM SYMBOLŮ A ZKRATEK

Zkratky:

EDC	Error Detection Codes
ECC	Error Corrections Codes
LSB	Least Significant Bit
MSB	Most Significant Bit
VRC	Vertical Redundancy Check
CRC	Cyclic Redundancy Check
XOR	eXclusive OR
SECDEC	Single Error Correction Double Error Detection
BHC	Bose-Chaudhuri-Hocquenghem
SRAM	Static Random Access Memory
FPGA	Field-programmable gate array

Symboly:

n	Dĺžka kódu	(-)
k	Počet informačných bitov	(-)
r	Počet kontrolných bitov	(-)
R	Informačný pomer	(-)
$w(v)$	Hammingova váha	(-)
$d(u,v)$	Hammingova vzdialenosť	(-)
d_{min}	Minimálna kódová vzdialenosť	(-)
b	Bit	(-)
$g(x)$	Polynom kontrolného slova	(-)
$u(x)$	Polynom informačného slova	(-)
$c(x)$	Polynom kódového slova	(-)
$ø(x)$	Elementárny polynom	(-)