Fakulta informačních technologií
Vysoké učení technické v Brně

Mikroprocesové a vestavěné systémy
Zabezpečení dat pomocí 16/32-bit. kódu CRC

2018 Patrik Holop

Obsah

1 Úvod

Kód CRC (Cyclic Redundancy Check) sa často využíva v dátovej komunikácii a úložiskových zariadeniach na zabezpečenie a detekciu chýb pri prenose dát[1]. Cieľom projektu je porovnať možnosti výpočtu a realizácie zabezpečenia dát pomocou 16 a 32 bitovej verzie kódu CRC. Výpočet daného kódu je možné realizovať využitím hardwarového modulu čipu Kinetis K60 z platformy FITKit 3, prípadne priamym výpočtom alebo získaním hodnoty z tabuľky. Princíp každej metódy je popísaný v sekcii 2, forma implementácie v sekcii 3. Vyhodnotenie výsledkov a efektivity jednotlivých metód nájdete v sekcii 4.

2 Teoretické podklady

CRC reprezentuje hashovaciu funkciu na získanie konktrolného súčtu. Vstupná postupnosť bitov, ktoré majú byť danou metódou zakódované, reprezentuje polynóm $P(x)$, kde sú hodnoty bitov interpretované ako koeficienty a index daných bitov ako mocnina premennej.

$$\text{vstupné binárne dáta: } 1_2 1_1 0_0 \Leftrightarrow P(x) = x^2 + x^1$$

Rovnakým spôsobom je reprezentovaný polynóm $G(x)$, ktorý sa použije pri výpočte výslednej hodnoty. Varianta CRC-N určuje počet bitov vstupného polynómu rovný hodnote $N + 1$. Výsledný CRC kód $C(x)$ potom zodpovedá:

$$C(x) = P(x) \bmod G(x)$$

Princíp zabezpečenia dát kódom CRC pri komunikácii sa dá rozdeliť na 3 základné kroky:

- Výpočet číselného kódu pre danú postupnosť bitov (vstupné dáta)
- Zaslanie dát spolu s kódom
- Overenie prijatých dát na strane príjemcu

V praxi sa zasielanie kódu a overenie dát vykonáva dvoma spôsobmi.

- Vypočítaný kód sa vloží priamo k zasielaným dátam a v takom prípade príjemca opäť vykoná operáciu $C(x) = P(x) \bmod G(x)$, kde $P(x)$ je hodnota prijatých dát (spoločne s kódom). V takom prípade stačí porovnať výsledok $C(x)$ s očakávanou hodnotou pre daný štandart, najčastejšie 0.
- Druhý spôsob je zaslať kód a dáta samostatne, pričom príjemca zopakuje operáciu rovnako ako odosielateľ nad prijatými dátami a porovná výsledný a prijatý kód.

V tomto projekte sa využíva prvá varianta.

2.1 CRC štandardy

Vyššie popísaný algoritmus popisuje obecný postup pri výpočte kódu CRC. V rôznych oblastiach sa ale používajú rôzne variáce úpravy dát pred a po samotnom výpočte. Varianty CRC-16 s CRC-32 majú vlastné štandardy[2].

Medzi parametre jednotlivých štandardov patria:

- Generujúci polynóm
- Inicializačná hodnota registra pred výpočtom (seed)

- Obrátenie bitov v jednotlivých bytoch vstupných dát
- Obrátenie jednotlivých bytov vstupných dát
- Použitie operácie XOR nad vypočítaným kódom a daná hodnota

V tomto projekte sú použíté nasledovné štandarty:

Varianta CRC	Názov štandardu	Polynóm	Seed	Obr. bitov	Obr. bytov	XOR
CRC-16	CDMA 2000	0xC867	0xFFFF	-	-	0
CRC-32	MPEG-2	0x04C11DB7	0xFFFFFFFF	-	-	0

Varianta CRC-16 CDMA 2000 sa používa v bezdrôtových sieťach na fyzickej vrstve[3] a varianta MPEG-2 v audio dekóderi[4].

2.2 Metódy výpočtu CRC

2.2.1 Metóda priameho výpočtu

Výpočet CRC sa dá realizovať iteratívne pomocou posuvného registru (left shit register). Použitá varianta CRC-N určuje pridanie N bitov nulovej hodnoty pred výpočtom ku vstupnej postupnosti, aby sa pri postupnom posuve pri výpočte dopĺňali neutrálne hodnoty. Použitie bitového operátu « v jazyku C nám umožňuje túto skutočnosť zanedbať, pretože bity s hodnotou 0 sú doplnené automaticky. Postupným posuvom registru sa dá zistiť, či hodnota bitu, ktorý bude v danom kroku z registra odstránený, má hodnotu 1. Ak áno, je potrebné pomocou operácie XOR medzi aktuálnou hodnotou registra a generujúcim polynómom aktualizovať hodnotu CRC. Úvodná hodnota CRC (medzivýpočtu) sa často označuje ako *seed*.

Predpokladajme, že vstupné dáta sú rozdelené na jednotlivé byty a hodnota výsledného CRC kódu je inicializovaná hodnotou seed. V takom prípade budú pri výpočte problematické práve oblasti, kde by sa hodnoty jednotlivých bytov vstupných dát mohli v posuvnom registri prekrývať. Tento fenomén sa dá odstrániť tak, že pri spracovaní každého ďaľšieho bytu sa táto hodnota pomocou operácie XOR vloží ako MSB aktuálnej hodnoty medzivýpočtu.

```
Data: Hodnota CRC inicializovaná hodnotou seed
Data: Vstupné dáta rozdelené na byty
Result: Hodnota CRC
for byte in bytes do
    crc = crc_msb XOR byte;
    for bit in byte do
        crc = crc « 1;
        if shifted bit in crc == 1 then
            crc = crc XOR polynom;
        end
    end
end
```

2.2.2 Metóda použitia tabuľky

Metóda použitia tabuľky navrhuje jednoduché zrýchlenie predchádzajúceho postupu. Vychádza z myšlienky, že deliteľ (generujúci polynóm) bude vždy rovnaký a byte, s ktorý sa má postupne vykonávať operácia XOR, môže nadobúdať iba obmedzený počet hodnôt (256). V takom prípade je možné mať

uložené v pamäti (vyhľadávacia tabuľka) všetky hodnoty, ktoré môžu pri operácii XOR týchto dvoch hodnôt nastať.

Vždy, keď sa hodnota MSB CRC aktualizuje operáciou XOR s daným bytom, môžeme si podľa tejto hodnoty okamžite vyhľadať správny výsledok namiesto postupného opakovania operácie XOR s polynómom po každom posunutí registra a už len aktualizovať zvyšok hodnoty CRC pomocou operácie XOR s hodnotou z tabuľky.

2.2.3 CRC modul čipu K60

Dokumentácia modulu CRC podrobne rozpisuje popis konfigurácie a postupu práce s daným modulom[5]. Modul poskytuje register CRC_CTRL, v ktorom sa dajú nastaviť hodnoty TOT a TOTR pre obracanie bitov a bytov vstupných dát, hodnota WAS na určenie, či sa do dátového registru modulu zapisujú seed dáta alebo vstupné dáta a FXOR, ktorý pri hodnote 1 po výpočte pomocou operácie XOR upraví výsledok.

Keďže modul vracia výsledok okamžite po zápise do dátového registru a umožňuje pracovať naraz s maximálne 32b dátami, pri väčšom počte dát je nutné do registru zapisovať postupne.

3 Implementácia

Projekt je implementovaný v jazyku C (bez použitia Kinetis SDK) a navrhnutý ako sada funkcií, ktoré implementujú algoritmy v kapitole 2. Na začiatku programu je nutné inicializovať hardwarový modul tak, že sa spustia hodiny MCU a povolia pre daný modul, pretože jeho práca je závislá od MCU hodín. Rovnako sa definujú vyhľadávacie tabuľky realizované poliami v hlavičkovom súbore. Dané tabuľky boli vygenerované pri tvorbe projektu pomocou úpravy funkcie na priamy výpočet. Následne sa nadefinujú demonštračné vstupné dáta do poľa input_data o veľkosti 64 bitov a zavolá sa funkcia test_and_verify. Táto funkcia pracuje nasledovne:

Postupne sa pre metódu priameho výpočtu, výpočtu cez tabuľku a HW modulom pre obe varianty CRC-16 a CRC-32:

- Vypočíta CRC kód vstupných dát pomocou príslušnej funkcie
- Vytvorí sa pole testovacích dát, ktoré bude okrem vstupných dát obsahovať aj vypočítaný CRC kód
- Overí sa integrita testovaných dát pomocou prepočtu CRC kódu
- Krok 1 a 2 sa zopakujú pre testované dáta s invertovaným prvým bytom, ktorý simuluje poškodenie dát pri prenose.

Na konci funkcie je možné porovnať všetky získané hodnoty, ktoré boli postupne vypisované do prehľadnej tabuľky. Všetky vypočítáné CRC hodnoty pre dáta obsahujúce CRC kód by mali mať hodnotu 0 značiacu správny prenos, pričom hodnoty nad poškodenými dátami nenulovú hodnotu.
Na konci programu sa nachádza nekonečný cyklus pre zachytenie činnosti MCU.

4 Vyhodnotenie metód

4.1 Detekcia chýb

CRC pracuje podobne ako hashovacia funkcia. Je nutné si uvedomiť, že čím nižšiu variantu CRC-N zvolíme, tým je väčšia pravdepodobnosť kolízie, čo znamená, že rôzne postupnosti vstupných dát budú

mať rovnakú výslednú hodnotu CRC kódu.

Pri náhodnej chybe a rovnomernom rozložení hashovaných hodnôt vzniká pravdepodobnosť akceptovania danej správy (kolízia) pri variante CRC-16 $\frac{1}{2^{16}}$ a pri CRC-32 $\frac{1}{2^{32}}$. Táto pravdepodobnosť je ale ovplynená aj ostatnými faktormi ako napríklad dĺžka správy a počet možných správ.
Druhá vlastnosť, ktorá určuje schopnosť detekovať chyby je polynóm daného CRC algoritmu. Štúdia dokazuje, že pri vhodne zvolenom polynóme sme vždy schopný detekovať až trojnásobné chyby a vždy nepárne chyby[6].

4.2 Rýchlosť výpočtu

Najpomalšou metódou na výpočet CRC kódu je jednoznačne priamy výpočet. Pri tomto algoritme je nutné iterovať cez každý bit vstupných dát s vysokým možným počtom operácií XOR. Za predpokladu, že operácia XOR je vykonávaná pre bity v jednom kroku (paralelne), je predpokladaná zložitosť v najhoršom prípade ak neuvažujeme úvodnú inicializáciu a záverečný XOR:

Počet bitov vstupnej správy * 4

V prípade vyhľadávacej tabuľky sa tak po odstránení réžie postupných operácií pri každom bite rýchlosť zvyšuje na

Počet bytov vstupnej správy * 4

Hardwarový modul je ale najrýchlejšia varianta. Hlavná výhoda je paralelizmus. Umožňuje naraz spracovať až s 32b hodnotami bez dodatočnej réžie. Tým sa rýchlosť zvyšuje na

Počet bytov vstupnej správy / 4.

4.3 Realizačná réžia a prenositeľnosť

Metóda priameho výpočtu ako aj použitie tabuľky sú založené čisto na implementácii pomocou jazyka C a sú tak prenositeľné na akúkoľvek platformu s príslušným prekladačom. Na rozdiel od použitia HW modulu nepotrebujú platformu s presne špecifikovaným rozhraním reprezentujúci daný modul.

Metóda priameho výpočtu má popri svojej činnosti minimálnu pamäťovú náročnosť. Pri použití tabuľky je nutné ju staticky umiestniť do pamäti alebo pred použitím predvypočítať, čo môže byť problematické na vstavaných zariadeniach s minimálnou veľkosťou pamäte.

5 Záver

Cieľom projektu bolo naimplementovať a porovnať rôzne metódy výpočtu a zabezpečenia dát pomocou kódu CRC. Jednoznačne najrýchlejšia bola implementácia pomocou HW modulu. CRC je schopné detekovať aj viacnásobné chyby s pomerne vysokou presnosťou.

6 Prílohy

Ukážkový výstup aplikácie pre vstupné dáta 0x1234567887654321.

```
P&E Semihosting Console

* Project: CRC-16 & CRC-32 *

************* CRC 16 RESULTS *************
*                                        *
*   CRC-16 HW module:          0x5279    *
*   CRC-16 direct calculation: 0x5279    *
*   CRC-16 lookup table:       0x5279    *
*                                        *
******** CRC 16 VERIFICATION OK **********
*                                        *
*   CRC-16 HW module:          0000      *
*   CRC-16 direct calculation: 0000      *
*   CRC-16 lookup table:       0000      *
*                                        *
******* CRC 16 VERIFICATION ERROR ********
*                                        *
*   CRC-16 HW module:           0x40d8   *
*   CRC-16 direct calculation:  0x40d8   *
*   CRC-16 lookup table:        0x40d8   *
*                                        *
************* CRC 32 RESULTS *************
*                                        *
*  CRC-32 HW module:          0xc15a147d *
*  CRC-32 direct calculation: 0xc15a147d *
*  CRC-32 lookup table:       0xc15a147d *
*                                        *
******** CRC 32 VERIFICATION OK **********
*                                        *
*  CRC-32 HW module:          00000000   *
*  CRC-32 direct calculation: 00000000   *
*  CRC-32 lookup table:       00000000   *
*                                        *
******** CRC 32 VERIFICATION ERROR *******
*                                        *
*  CRC-32 HW module:          0x99c5421  *
*  CRC-32 direct calculation: 0x99c5421  *
*  CRC-32 lookup table:       0x99c5421  *
*                                        *
*************** Finished *****************
```

7 Literatúra

[1] MESANDER B. *Understanding the Cyclic Redundancy Check*. [online, cit. 20.11.2018]. Cardinal Peak. Dostupné z https://cardinalpeak.com/blog/understanding-the-cyclic-redundancy-check/

[2] K. Witzke and C. Leung, *A Comparison of Some Error Detecting CRC Code Standards,"in IEEE Transactions on Communications*, vol. 33, no. 9, pp. 996-998, September 1985. doi: 10.1109/TCOM.1985.1096411

[3] CAROLUS K. *An Overview of cdma2000 Technology Concepts*. [online, cit. 25.11.2018]. Agilent Technologies. Dostupné z: https://www.keysight.com/upload/cmc_upload/All/07-16-02-and-06-24-02-Overview-cdma2000_670.pdf

[4] HOEG W., LAUTERBACH T. *Digital Audio Broadcasting - Principles and applications of Digital Radio*. Wiley. ISBN: 978-0-470-51037-7

[5] *K60 Sub-Family Reference Manual*. [online]. Freescale Semiconductor. Dostupné z: http://www.fit.vutbr.cz/ strnadel/public/fitkit3/k60p144m100sf2v2rm.pdf

[6] HÖST S. *Error detection by CRC*. [online]. Faculty of Engineering LTH. LUND University. Dostupné z: https://www.eit.lth.se/fileadmin/eit/courses/ets130/CRC.pdf