VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

IMP - Mikroprocesorové a vestavěné systémy

ARM/FITkit3: Zabezpečení dat pomocí 16/32-bit. kódu CRC, S

1. ledna 2018

Adrián Tomašov

Obsah

1 Úvod

Účelom tohto projektu bolo demonštrovať rôzne spôsoby zabezpečenia dát pomocou kódu 16/32-CRC. Tento projekt bol vyvíjaný na doske FitKit3 Minerva s čipom kinetis K60. Výpočty CRC sme mali robiť tromi rôznymi spôsobmi:

1. Použitím hardwarového modulu pre výpočet 16/32-bitového CRC z čipu K60
2. Výpočet na základe pomocnej vyhľadávacej tabuľky, ktorá urýchli výpočty CRC
3. Výpočet CRC pomocou polynómu

2 Vývoj

Tento projekt bol vyvíjaný na virtuálnom stroji s operačným systémom Windows 7. Virtuálny stroj bol spustený na systéme Fedora 27 a vývojová doska bola premostená cez USB do virtuálneho stroja. Program pre virtualizáciu použitý VirtualBox.

Takýto spôsob vývoja bol zvolený, pretože nastali isté komplikácie pri inštalácií a konfigurovacií vývojového prostredia Kinetis Design Studio na operačnom systéme Fedora 27. Pre komunikáciu užívateľa s vývojovou doskou bol použitý program Putty.

3 Kostra programu

Kostru tohto programu je demo program z Kinetis SDK(Software development kit) 2.0, kde sa nachádzali ukážky kódu pre používanie hardwarového modulu, ktorý počíta CRC. V tomto demo programe bol celý projekt prednastavený pre správny preklad a spustenie kódu na vývojovej doske, čo mi značne uľahčilo ďalší postup vo vývoji.

V tejto kostre som pri implementácii ostatných CRC algoritmov upravoval len jeden súbor a mal názov `crc.c`.

4 CRC

Kontrola správnosti cyklickým kódom (Cyclic redundancy check) je spôsob zabezpečenia dát kontrolným súčtom. Tento súčet slúži pre odhalenie chyby v dátach, nad ktorými ju súčet vypočítaný. Najčastejšie sa tento súčet používa v telekomunikačnej technike a počítačových sieťach. Odosielateľ dát vypočíta CRC súčet a pri odosielaní pridá tento súčet k odoslaným dátam. Na strane prijímateľa sa tento proces opakuje a výsledná hodnota CRC súčtu sa porovná so súčtom, ktorý poslal odosielateľ. Ak sú súčty rovnaké, znamená to, že sa dáta preniesli bezchybne.

4.1 CRC v hardwarovom module

V čipe K60 ja zabudovaný hardwarový modul pre akcelerované výpočty CRC. Pre uľahčenie práce s týmto modulom bolo použité **Kinetis SDK 2.0**, kde bol implementovaný driver pre ovládanie CRC modulu. Pred použitím modulu bolo nutné nakonfigurovať tento modul so správnymi parametrami daného CRC algoritmu. Táto konfigurácia sa ukladala do premennej `crc_config_t config`. Hodnoty pre CRC algoritmy sa dali ľahko nájsť kdekoľvek na internete. Následne sa zavolala funkcia `CRC_Init(base, &config)`, ktorá nakonfigurovala daný CRC modul. Táto funckia sa volala niekoľko krát, vždy keď sa menil CRC algoritmus.

Pre zápis dát do modulu sa používala funkcia `CRC_WriteData(base, (uint8_t *)data, size)`, ktorá zapísala do CRC registra dáta, z ktorých sa počítal CRC súčet. Pre výsledok súčtu je v **Kinetis SDK 2.0** implementovaná funkcia `CRC_Get16bitResult()` a `CRC_Get32bitResult()`.

4.1.1 CRC16 CCINT-FALSE - konfigurácia

```
115 static void InitCrc16_CcitFalse(CRC_Type *base, uint32_t seed)
116 {
117     crc_config_t config;
118
119     CRC_GetDefaultConfig(&config);
120     config.seed = seed;
121     CRC_Init(base, &config);
122 }
```

4.1.2 CRC32 - konfigurácia

```
125 static void InitCrc32(CRC_Type *base, uint32_t seed)
126 {
127     crc_config_t config;
128
129     config.polynomial = 0x04C11DB7U;
130     config.seed = seed;
131     config.reflectIn = true;
132     config.reflectOut = true;
133     config.complementChecksum = true;
134     config.crcBits = kCrcBits32;
135     config.crcResult = kCrcFinalChecksum;          136
137     CRC_Init(base, &config);
138 }
```

4.2 CRC pomocou polynómu

Toto je základný algoritmus pre výpočet CRC súčtu. Nemá v sebe žiadne akcelerácie ani optimalizácie pre výpočet. CRC hodnota je uložená v pomocnej premennej a upravuje sa počas celého hlavného cyklu v algoritme. Algoritmus cyklí cez každý bit. V každom kroku cyklu bitovo posunie doľava/doprava premennú s CRC súčtom. Ak aktuálny bit je nastavený na logickú 1, tak ešte s premennou CRC urobí binárnu operáciu XOR s polynómom. Po ukončení cyklu je v premennej CRC uložený správny súčet CRC.

4.3 CRC pomocou vyhľadávacej tabuľky

Toto je optimalizovaný algoritmus pre výpočet CRC. Pre urýchlenie výpočtov využíva vopred vygenerovanú tabuľku s pomocnými výpočtami. V tomto algoritme sa cyklí cez bajty a nie bity. V premennej CRC sa ukladá postupne vypočítaný súčet CRC. V každom kroku cyklu sa urobí operácia XOR aktuálneho bajtu a CRC bitovo posunutým v vľavo/vpravo. Výsledok tejto operácie je hodnota, ktorá slúži pre prístup k číslu z pomocnej tabuľky. Táto hodnota je pomocou operácie XOR pripočítaná k hodnote CRC, ktorá je bitovo posunutá vľavo/vpravo.

5 Porovnanie

Z týchto spôsobov je najrýchlejší hardwarový CRC modul. Nepotrebuje pre svoj chod procesor a svoje výpočty robí priamo na logických obvodoch. Výhody : nízka pamäťová náročnosť, rýchlosť.

Výpočet CRC pomocou matematický výpočtov je najpomalší, pretože cyklí cez každý bit v dátach. Nad týmto bitom sa ďalej robí porovnanie a jednu alebo dve matematické operácie nad premennou. Výhody: nízka pamäťová náročnosť. Nevýhody: tento algoritmus je pomalý.

CRC výpočet pomocou vyhľadávacej tabuľky je, v porovnaní s hardware CRC modulom a matematickým výpočtom, niekde medzi nimi. Cyklí cez všetky bajty a nie bity. Táto vlastnosť ho robí rýchlejším algoritmom oproti výpočtu CRC cez matematický polynóm. Výhody: rýchlosť. Nevýhody: veľké pamäťové nároky kvôli vyhľadávacej tabuľke.