Hard- und Softwareaspekte zur Optimierung von embedded Systemdesigns

Erhöhung der Systemsicherheit durch die Wahl ECC geeigneter Prozessoren und Speicher

Kei Thomsen, MicroSys Electronics GmbH

Mit hochintegrierten und leistungsfähigen System on Chip Lösungen wandert zunehmend Intelligenz bis auf die Sensorebene von komplexen embedded Anwendungen. Wie Zuverlässigkeit im Design auch bei kleinen Systemstrukturen zu erreichen ist, gepaart mit hoher Performance und geringer Leistungsaufnahme, ist nach wie vor eine wichtige Kompetenz modernen Systemengineerings.
Im Vortrag/Artikel werden gleichschnell getaktete ARM-, PowerPC- und X86-Plattformen bezüglich Gesamtperformance und Systemsicherheit (Safety) verglichen. Anhand zweier C-Code Anwendungsbeispiele, die in erster Betrachtung fast identisch aufgebaut sind, wird erklärt, wie mit durchdachter Programmierung Leistungssteigerungen bis zu Faktor 30 möglich sind. Durch die immer kleiner werdenden Chip-Strukturen wird die Wichtigkeit von ECC-Memory (Error Correcting Code) und Prozessorunterstützung erläutert. Darüber hinaus gibt es neue Erkenntnisse zum Thema NAND-Flash-Speicher. Hier werden Methoden wie z.B. das Scrubbing bei NAND Flashes erklärt.

Teil 1: CPU-Plattformvergleich hinsichtlich Systemleistung
- C-Code Beispiel: "gute" und "schlechte" Programmierung bei Speicherzugriffen
- Ergebnisse des C-Beispiels auf ARM, PowerPC und X86 Plattformen mit etwa gleich schnell getakteten CPUs
- Ursache der Unterschiede und was bedeutet das für die Gesamtperformance

Teil 2: Speicherarchitekturen und Systemzuverlässigkeit
- Hintergrundinformationen zu immer kleineren Chipstrukturen bei Speichern und verbundene Fehlerquellen
- Erhöhung der Zuverlässigkeit für CPU Plattformen und Speicher durch ECC
- NAND (SLC, MLC, TLC) Flashes und deren Sicherung gegen Datenverlust

Zusammenfassung & Diskussion

CPU-Plattformenvergleich hinsichtlich Systemleistung
Es gibt immer wieder Streit darüber, welches der performantere und bessere Prozessor für eine gegebene Anwendung ist. Meist stehen hier X86, PowerPC und ARM in ihren unterschiedlichen Ausprägungen gegeneinander. Vorweg: Gleich schnell getaktete Prozessoren sind auch in etwa gleich schnell, +/- ein paar Prozente. Da ein Großteil der lokalen Daten im Cache landen, sind hier kaum Unterschiede sichtbar. Sobald es jedoch ins externe RAM hinausgeht, kommt die Busbreite, Speichertyp und Cache-RAM Verbindung ins Spiel. Ob und wo ein Unterschied besteht, zeigen die nächsten Abschnitte.

C-Code Beispiel: "gute" und "schlechte" Programmierung bei Speicherzugriffen

Doch zunächst zu dem C-Code Beispiel, mit dem die Messung durchgeführt werden. Warum „gute“ und „schlechte“ Programmierung? Häufig wird einfach programmiert und das Ergebnis als Gott gegeben hingenommen. Mit ein wenig Verständnis, wie der Prozessor mit dem Cache und RAM arbeitet, können speicherintensive Funktionen um Faktoren schneller ablaufen.

Als Beispiel werden hier 2 fast gleiche C Funktionen benutzt.

```
long array[1024][1024];
for (i = 0; i < 1024; i++)
    for (j = 0; j < 1024; j++)
      b += array[j][i];
```

C-Source Code 1

```
long array[1024][1024];
for (i = 0; i < 1024; i++)
    for (j = 0; j < 1024; j++)
      b += array[i][j];
```

C-Source Code 2

Ergebnisse des C-Beispiels auf ARM, PowerPC und X86 Plattformen mit etwa gleich schnell getakteten CPUs

100 * die C Schleifen = 100 * 4MB lesen und aufaddieren

CPU	MHz	Vertikal (links) msec	Vertikal MB/sec	Horiz. (rechts) msec	Horiz. MB/sec	Faktor Verti./ Horiz.
ARM Cortex-A9 NXP i.MX6	800	19782	20	1209	338	**16**
ARM Cortex-A9 Xilinx Zynq	666	20077	20	1399	292	**14**
PowerPC QorIQ P2020	1200	41415	9	1351	303	**30**
Vortex86-DX	800	15676	26	5246	78	**3**
Intel i7	2100	1480	270	376	1080	**4**

Tabelle 1

In der Tabelle 1 sind die Ergebnisse exemplarisch für einige CPUs dargestellt. Andere CPUs mit anderen Speichern ergeben natürlich andere Werte! Wichtige Erkenntnisse liefert der Faktor der Beschleunigung (rechte Spalte, Tabelle 1). Dieser variiert zwischen 3 und 30. Eine „gute“ Programmierung kann damit oftmals von sehr viel größerer Bedeutung sein, als eine schnellere CPU.

Ursache der Unterschiede und was bedeutet das für die Gesamtperformance

Das Ausführen von C-Source 1 wird in Tabelle 2 dargestellt und zeigt das Lesen von Daten in vertikaler Reihenfolge. Das Ausführen von C-Source 2 wird in Tabelle 3 dargestellt und zeigt das Lesen von Daten in horizontaler Reihenfolge.

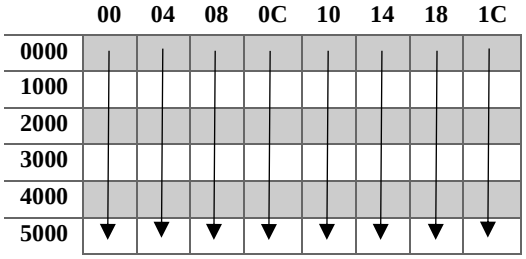

Tabelle 2 (zu C-Source 1)

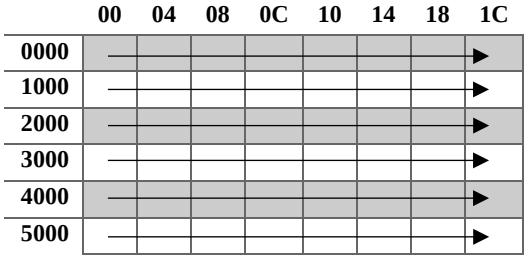

Tabelle 3 (zu C-Source 1)

Zunächst die Analyse von Tabelle 3. Es soll von Adresse $0000 ein 32bit Wert geladen werden. Der Wert befindet sich nicht im L1/L2 Cache, deswegen wird per Burst Read mit 32 oder 64 Byte (je nach Cache Line Size) das RAM gelesen. Dieses dauert im Vergleich zum Lesen aus dem Cache extrem lange. Die CPU wartet (stallt), bis der Wert im L1-Cache angekommen ist und gelesen werden kann. Die nächsten Zugriffe auf $4, $8, $C, $10… kommen dann nur noch aus dem L1-Cache, sind also extrem schnell.

In Tabelle 2 hingegen wird $0 und dann $1000 gelesen. Es muss also gleich wieder gewartet werden, bis das RAM ausgelesen wurde, weil die nächsten Daten noch nicht im Cache vorliegen. Typischerweise sind die L1 Caches 32KB groß und beinhalten 1024 Daten Lines a 32 Byte (Cache Line Size). In dem Beispiel werden jetzt auch genau 1024 Zeilen gelesen, wodurch der Cache exakt genau gefüllt sind. Somit können die weiteren Daten bei $X004 / $X008 / $X00C bei der nächsten Runde nun aus dem Cache gelesen werden. Weiterhin spielt die MMU (Memory Management Unit) noch eine maßgebliche Rolle für die Systemleistung. Jede MMU Page ist typischerweise 4 KB groß (ARM kann auch 64KB). Ein Eintrag zeigt also auf genau eine Datenzeile von 1024*32bit. Da die Adresse $0 nicht in der MMU-Tabelle zu finden ist, wird eine Exception ausgelöst, um per Software den MMU-Eintrag für diese Adresse neu zu laden, was folglich für jeden Zugriff geschieht. Hier hat der PowerPC einen kleinen Nachteil, da er nicht wie bei ARM & X86 mit L1/L2 Tabellen arbeitet, sondern 512 Einträge a 4KB besitzt und somit in jedem Durchlauf eine Exception für die Zeile erzeugt, was den schlechteren Faktor 30 (Tabelle 1) für den PowerPC erklärt. Wenn das Beispiel etwas geändert wird (512*512 statt 1024*1024), dann hat der PowerPC nur noch den Faktor 8 während ARM immer noch bei 10-12 bleibt. Somit bleibt: Entwerfe und optimiere deinen Benchmark so, dass dein gewünschtes Ergebnis herauskommt!

Zusammenfassung zum Speicherzugriff: Es kommt maßgeblich auf die Architektur an, wie sich „schlechtes Programmieren“ auf die Performance auswirkt. Bei „guter Programmierung“ sind alle Prozessoren etwa gleich schnell (bei gleichen Speichertypen) und nur minimal abhängig von der CPU-Geschwindigkeit (666 / 800 / 1200 MHz im Beispiel). Die CPU-Geschwindigkeit kommt erst dann zum Zug, wenn viel gerechnet wird und die Daten größtenteils im L1-Cache vorliegen. Der Intel i7 Prozessor sticht heraus, weil der Speicher bei diesem Board mit 128Bit Dual Channel angebunden ist und damit in 2 Zyklen eine Cacheline füllt, für das die anderen Systeme 8 Zyklen benötigen.

Speicherarchitektur und Systemzuverlässigkeit

Hintergrundinformationen zu immer kleineren Chipstrukturen bei Speichern und verbundene Fehlerquellen

Moderne Halbleiterstrukturen werden immer kleiner und ermöglichen dabei immer mehr Speicherplatz. Das gilt für RAMs (DDR3/4) ebenso wie für NAND Flashes. Betrachten wir hinsichtlich der Systemsicherheit zunächst die RAMs.
Die aktuelle Größe der Chip-Struktur eines DDR4 RAMs beträgt 20 nm und wird mit einer Spannung von 1.2V betrieben, um die zu bewegende Ladung (< 20000 Elektronen) gering zu halten. Das ist notwendig, um den Stromverbrauch zu reduzieren und die Geschwindigkeit zu erhöhen. Damit besteht aber die Gefahr von Soft-Error-Störungen durch Höhen- und radioaktiver Strahlung, sowie durch energiereiche Strahlung von Elektromotoren und statischen Feldern. Diese Effekte bezeichnen wir als Single Event Upsets – SEU. Häufig sind auch die RAM Timings nicht genau berechnet und eingestellt. Da das Timing Temperaturabhängig ist, kann es bei hohen oder niedrigen Temperaturen zu Bit-Fehlern kommen kann. Ohne geeignete Gegenmaßnahmen kann das, hinsichtlich eines sicheren Systembetriebs, recht gefährlich werden! Hier ist für sicherheitskritische Anwendung eine geeignete Maßnahme der Einsatz von ECC-Speicher (Error Correction Code).

Erhöhung der Zuverlässigkeit für CPU Plattformen und Speicher durch ECC

Um Speicherfehler zu erkennen und zu korrigieren, kommt in den meisten Fällen ein Hardware-ECC zum Einsatz. Dazu wird das 64 Bit breite RAM um 8 Bit auf 72 Bit erweitert (bei 32 Bit + 7 Bit). Damit lässt sich ein 1-Bit-Fehler direkt korrigieren und ein 2-Bit-Fehler erkennen, was natürlich vom Memory Controller unterstützt werden muss. Jedoch haben heute bei weitem nicht alle Prozessoren einen solchen ECC fähigen Memory Controller auf dem Chip (System On Chip – SOC). Fast alle PowerPC Prozessoren besitzen einen ECC fähigen Controller. Für ARM-CPUs ist das hingegen die Ausnahme und bei X86-CPUs sind es meist nur die größeren Server Chipsets und nur selten die Embedded SOCs, die mit dieser Funktion ausgestattet sind.

Wenn der SOC nun ECC unterstützt und auch die richtige Anzahl von RAMs vorhanden ist (dann sind typischerweise 5 oder 9 RAMs auf dem Board), dann sollte dieses Feature natürlich auch genutzt werden. Hierbei stellt sich gleich die Frage, wieviel zusätzliche Verarbeitungszeit das kostet. Aus unseren Erfahrungen mit PowerPC Boards und ARM Xilinx ZYNQ Systemen, sind das weniger als 5% Verluste der Memory Performance. Es ist sinnvoll, ECC Interrupts von dem Betriebssystem zu unterstützen, um die Fehler überhaupt sichtbar zu machen. Der Memory Controller kann bei einem ECC korrigierbaren Einzel-Bit-Fehler ein Interrupt auslösen. In der Interrupt-Behandlung wird nun die Speicherzelle ausgelesen und neu geschrieben, denn die gelesenen Daten sind ja (noch) korrekt. Bei einem Multi-Bit-Error geht das jedoch nicht mehr und es muss speziell entschieden werden, wie damit umgegangen werden soll -> Sicherer Zustand, Fehlermeldung, Reboot, … Das lässt sich auch sehr schön Testen, da jeder ECC fähige Controller auch die Möglichkeit einer Error Injection besitzt, um Fehler zu Simulieren.

Weshalb sind ECC-Speicher so wichtig für die Systemzuverlässigkeit? Ein Single-Bit-Fehler wird im Speicher, wodurch auch immer, verursacht. Mit ECC-Speicher wird eine Meldung ausgegeben und die Daten werden korrigiert zurückgeschrieben, alles gut! Ohne ECC-Speicher sind im besten Fall die Daten nur ein wenig falsch, im schlimmsten Fall ist das im Betriebssystem-Code passiert und ein Sprung geht nicht +100 Byte nach vorne, sondern -100 rückwärts. Na dann, Prost!

NAND (SLC, MLC, TLC) Flashes und deren Sicherung gegen Datenverlust

Richtig gruselig wird es jedoch bei den NAND-Flash-Speichern, die in allen gängigen Speicherkarten wie USB-Stick, CF, SD, µSD und SSD enthalten sind, wenn sie in zuverlässigen embedded Systemen zu Einsatz kommen sollen. Normalerweise würde man annehmen: 1 Speicherzelle = 1 Bit. Das stimmt grade mal bei den SLC-Flash-Bausteinen. SLC = Single Level Cells, MLC = Multi Level Cells, TLC = Three Level Cells. Besonders kritisch sind die MLC- und TLC-Technologien. Bei MLC (Schwarze Magie) werden pro Zelle 2 Bit gespeichert, also 0, 0.33, 0.66 und 1. Bei TLC (Alien Technologie) sind es sogar 3 Bit. Die Zelle wird also mit 8 unterschiedlichen Zuständen geladen.
Man kann sich vorstellen, was innerhalb der Chips für ein Aufwand getrieben werden muss, um diese minimalen Ladungsunterschiede noch voneinander unterscheiden zu können. Um die Daten korrekt zu halten wird der ECC in den OutOfBand (OOB) Daten gespeichert, was ein zusätzlicher Datenbereich zu den eigentlichen Daten ist (16Byte OOB / 512Byte Nutzdaten). Bei SLC typischerweise 3 Byte pro 512 Byte für den ECC genutzt, bei MLC und TLC wird fast der gesamte 16 Byte Bereich dazu benutzt. Die meisten NAND-Flash-Controller auf den SOCs benutzen die 3 Byte um eine 1 Bit Error Correction per Hardware zu erzeugen. Werden mehr ECC-Bits benötigt, dann muss das in der Treiber-Software gelöst werden. Da die Zellen nur eine gewisse Anzahl an Schreib-Lösch-Zyklen vertragen (SLC ~100.000, MLC ~10.000, TLC ~ 3 - 5.000), müssen die Daten mit einem Wear-Leveling gleichmäßig auf die NAND Blöcke verteilt werden. Diese Arbeit wird bei SD, µSD, CF, USB-Stick und SSD in dem eingebauten Flash-Controller geleistet. Bei eingelöteten NAND-Flashes organisiert das ein entsprechendes Filesystem wie UBIFS, JFFS2 oder YAFFS2.

Bereits vor einigen Jahren haben die NAND-Flashes eine Speicherdichte pro mm² erreicht, bei denen nach einigen Monaten einzelne Bits von selbst kippen und das sogar in Sektoren die täglich nur 1-2-mal beim Booten gelesen werden. Nach intensivem Nachfragen bei den Herstellern bekommt man irgendwann mal die Auskunft, dass das im Erwartungsbereich liegt. Und das für den Einsatz im industriellen Umfeld! In Datenblättern ist so etwas nirgends zu finden. Aufgefallen ist uns das dadurch, dass ein NAND-Flash nach 12 Monaten plötzlich das Linux nicht mehr lesen konnte, da in einem 512 Byte Sektor 2 Bits umgekippt waren, die dann nicht mehr mit dem 1-Bit-ECC korrigiert werden konnten. Abhilfe kann hier schaffen, regelmäßig (alle 1-2 Monate) einmal das gesamte NAND-Flash zu lesen und dabei den Fehler-Status vom NAND-Flash-Controller zu überprüfen. Meldet der Controller, dass er ein Bit korrigiert hat, kann man diesen Sektor einfach überschreiben und neu überprüfen. Wenn man diese Information vom Controller nicht bekommen kann, dann hilft nur ein regelmäßiges Scrubbing, bei dem die Sektoren gelesen und wieder überschrieben werden. Das sollten die externen Flash-Speicher-Medien idealerweise selbstständig machen. Nur, wie macht das z.B. eine SD-Karte, die als Backup bereits seit 3 Jahren im Schrank liegt?

Eine befreundete Firma hat in diesem Zusammenhang herausgefunden, dass z.B.: neuere SanDisk CF Karten (>= 2GB) scheinbar aus Geschwindigkeitsgründen das Wear-Leveling nicht mehr über den gesamten Bereich, sondern nur über den für den FAT32 benötigten Bereich durchführt, wie er typischerweise in Kameras benutzt wird. Somit sind fremde Filesysteme wie EXT2/3/4, OS-9, QNX u.a., die nicht in dem Wear-Leveling Bereich angelegt sind, tödlich und verweigern nach ein paar tausend Zugriffen die Funktion. In unseren Marktuntersuchungen fanden sich nur wenige CF Karten als industrietauglich, die mit SLC-Technologie und Wear-Leveling über den gesamten Bereich arbeiten. Vom Gebrauch von MLC- oder TLC-Technologien wird für den industriellen Einsatz deshalb ganz abgeraten. Wie das bei SD-Karten aussieht… keine Idee!

Jetzt fragen Sie sich sicherlich, wie verhält es sich nun in diesem Zusammenhang mit den zunehmend verbreiteten SSD-Festplatten?

Ja, SSD-Platten basieren auf NAND-Flashes. Auch hier gibt es SLC-, MLC- und TLC-Festplatten. Je lauter die Werbung schrillt: „bahnbrechende 3D-NAND-Technik“, desto vorsichtiger sollte man sein, denn damit wird meist MLC oder TLC beworben. Günstig sind sie schon, aber wie steht es um deren langfristige Zuverlässigkeit? SSDs mit SLC hingegen muss man speziell suchen und die sind dann auch keine Billigheimer mehr, aber für den industriellen Einsatz sicherlich geeignet. Ich habe bei den Recherchen zum dem Artikel im E4You-Verein (Zusammenschluss von mehr als 30 embedded Lösungsanbietern) nachgefragt, ob jemand besondere Erfahrungen mit SSD-Technik gemacht hat. Und prompt kam zurück, „wir haben in 10 Rechnern seit 1 ½ Jahren SSD-Platten im Entwicklungseinsatz und innerhalb von 2 Wochen sind 3 SSD-Platten so ausgefallen, dass gar nichts mehr ging. Sie ließen sich überhaupt nicht mehr lesen. Zum Glück hatten wir bei 2 davon ein Backup der nicht älter als 1 Woche war.“

Ich möchte noch auf einen Test hinweisen, bei dem versucht wird 1 Petabyte Daten zyklisch auf 250GB SSD-Platten zu speichern (1 Petabyte = 1024 Terabyte). Hierbei konnten 3 SSD-Platten von insgesamt 6 nur knapp über 700 Terabyte beschrieben werden, bevor sie komplett ausfielen. Ausführliche Informationen über diese Untersuchungen sind auf nachfolgender Webseite zu finden:

techreport.com/review/26523/the-ssd-endurance-experiment-casualties-on-the-way-to-a-petabyte

Zusammenfassung & Diskussion

Durch die zunehmend kleiner werdenden Strukturen und die eingesetzten Alien Technologien werden Sicherungsmaßnahmen durch den SOC, das Betriebssystem und den Filesystemen immer wichtiger. Bei industriellem Einsatz unter extremen Bedingungen, wie Temperatur und Störfeldern, sollte **immer** versucht werden ECC-Speicher zu nutzen, da sonst die Folgen desaströs sein können. Der minimale Verlust an Performance wird durch massive Verbesserung der Stabilität und Systemzuverlässigkeit mehr als wettgemacht. Da ECC schon lange Standard bei PowerPC Prozessoren ist, sind diese Prozessoren zumindest in unserem Haus für weiterhin die erste Wahl für Applikationen mit besonderen Anforderungen.

Die meisten ARM-Prozessoren hingegen sind für Commercial Use gebaut und haben nur selten einen ECC fähigen Speicher-Controller. Bei den NAND-Flashes sind im industriellen Einsatz eigentlich nur Flashes mit SLC-Technologie guten Gewissens zu empfehlen. Alles andere wäre an der falschen Stelle gespart. Wie zu Anfang beschrieben, ist Performance durch nichts zu ersetzen als durch noch mehr Performance. Die bekommt man aber nicht alleine von der Hardware, sondern auch zum großen Teil von gut geschriebenen Programmen.

Quellen

DDR RAM: de.wikipedia.org/wiki/Dynamic_Random_Access_Memory
Soft Error: de.wikipedia.org/wiki/Soft_Error
ECC: de.wikipedia.org/wiki/Fehlerkorrekturverfahren
Heavy Ion sensitivity of 16/32-Gbit NAND-Flash and 4-Gbit DDR3 SDRAM, ESA 3.2.2012
SLC, MLC, TLC: www.speedguide.net/faq/slc-mlc-or-tlc-nand-for-solid-state-drives-406
CF Karten Problem: Bericht von ESD
SSD Petabyte Club: techreport.com/review/26523/the-ssd-endurance-experiment-casualties-on-the-way-to-a-petabyte

Autor

Dipl.Ing. (FH) Kei Thomsen verfügt über 29 Jahre Erfahrung im Bereich der Embedded RTOS Programmierung. Seit 1997 ist er als Entwickler, Support und Trainer für das Betriebssystem OS-9 zuständig. Ein Schwerpunkt dabei sind hardwarenahe Entwicklungen für kundenspezifische Produkte basierend auf PowerPC, ARM und X86.

Kontakt

Internet: www.microsys.de
Email: thomsen@microsys.de

MicroSys Electronics GmbH
Mühlweg 1
D-82054 Sauerlach