ГОДИШНИК на Минно-геоложкия университет "Св. Иван Рилски", Том 52, Св.III, Механизация, електрификация и автоматизация на мините, 2009
ANNUAL of the University of Mining and Geology "St. Ivan Rilski", Vol. 52, Part III, Mechanization, electrification and automation in mines, 2009

МЕТОДИ ЗА ПОВИШАВАНЕ НА БЪРЗОДЕЙСТВИЕТО ПРИ РЕАЛИЗАЦИЯ НА ИЗЧИСЛИТЕЛНИТЕ ПРОЦЕДУРИ В СПЕЦИАЛИЗИРАНИТЕ УСТРОЙСТВА ЗА ИЗМЕРВАНЕ И КОНТРОЛ

Ясен Видолов

Минно-геоложки университет "Св. Иван Рилски", 1700 София, e-mail: nre@abv.bg

РЕЗЮМЕ. Разгледани са съвременните подходи за увеличаване на бързодействието при реализация на изчислителните процедури – използване на специализирани приложни схеми, строго специализирани микроконтролери, конвейерна и многопроцесорна обработка на информацията, системи за реално време отговарящи на събития. На база на извършения анализ е предложено разработване на система, базирана на събития, която предлага голям набор от прекъсвания и осигурява възможност за паралелното им обслужване.

METHODS FOR INCREASED CALCULATION PERFORMANCE IN SYSTEMS FOR MEASUREMENT AND CONTROL

Jasen Vidolov
University of Mining and Geology "St. Ivan Rilski", 1700 Sofia, e-mail: nre@abv.bg

ABSTRACT. Modern aproaches are explored for inceasing the speed when implementing mathematical procedures, using specialized integrated circuits, strictly specialized microcontrollers, pipelines and multithreaded information processing, real-time event response systems. On behalf of the analysis shown, an event driven system is proposed. It offers a wide range of interrupts and provides capability of their simultaneous service routine execution.

Въведение

Увеличаващите се темпове на развитие на технологиите и нуждата от по-точна информация предоставяна през по-малък интервал на дискретизация, поставят все по-големи изисквания за качество и бързодействие пред интелигентните измерителните устройства. Решаването на проблема с увеличаване на бързодействието се налага и от факта, че те са неизменен елемент, влизащ в структурата на големите информационно-измервателни и управляващи системи и в значителна степен определят качеството на функционирането им.

Пътищата за повишаване на бързодействието са два, които условно можем да определим като хардуерен и софтуерен. При софтуерните подходи се търсят методи за обработка на информацията, които са с подобрени характеристики по отношение на скоростта на реализация на обработващите процедури, като например използване на табулирани функции, математични модели, рекурентни зависимости и др.

При хардуерният подход, който е обект на настоящия анализ, висока скорост на обработка на информацията при интелигентните измерителни устройства се постига чрез различни решения, зависещи от архитектурата на използваната изчислителна техника. Тя определя структурната организация на изчислителното устройство във вид на съвкупност от функционални модули, както и връзките между тях.

Методи за увеличаване на производителността на изчислителните устройства

Използване на CISC архитектури

Голяма част от микропроцесорните архитектури притежават съкратен набор команди (Reduced Instruction Set Code – RISC). Те се използват за малки по сложност математически и логически операции без необходимост от бързодействие. Когато времето за изпълнение на сложни команди е от значителна важност се прибягва до архитектури със сложен набор команди (Complex Instruction Set Code – CISC)[2]. При тях често използвани математически преобразования са имплементирани на хардуерно ниво, което ускорява тяхното изпълнение поради специфичната им реализация, съобразена с конкретните цели на приложение. Представители на архитектури със сложен набор команди са Цифровите Сигнални Процесори (DSP). Използването на CISC-архитектури позволява синтез на хардуерни решения, които осигуряват необходимата изчислителна точност и определена размерност на операндите, без да е

задължително тя да отговаря на размерността на машинната дума на процесора.

Например хардуерната имплементация на CORDIC-алгоритъма в микроконтролер ще опрости значително задача по моделиране на поведението на въртящо се магнитно поле, като предоставя възможност за последователното завъртане на вектор в полярна координатна система чрез предварително дефинирана стъпка. Без този алгоритъм, за изпълнението на задачата ще е необходимо използването на набор от тригонометрични функции, което ще отнемат относително голямо време за реализация.

От примера става ясно, че вграждането на специализирани функции е удачно при реализиране на задачи, които изскват честото им използване. Самзов този случай икономически е оправдано прилагането на CISC-архитектури.

Процесори за работа в реално време

Когато се говори за бърза реакция на управляващо устройство спрямо възникнали събития се прилагат процесори и операционни системи за обработка на информацията в реално време. При тях се поставя изискването за получаването на гарантиран отговор, не по-бавно от определено зададено времезакъснение [2]. За това при архитектурите с високи изисквания към бързодействието е важна детерминираността на времената за изпълнение на всяка една задача.

Когато е необходимо осигуваряване на бърза реакция на изчислителното устройство на възникнали външни събития (например излизане на технологичен параметър извън зададени допустими граници), се използват процедури, прикачени към хардуерни прекъсвания. Тяхното бързо изпълнение зависи основно от два критерия – време за стартиране на обработката на прекъсването и времето за нейното изпълнение.

Времето за стартиране на обработката на прекъсването зависи от последната започната процесорна инструкция, като се изпълнява веднага след нейното завършване [3]. Това изчакване за завършване на започнатата инструкция е необходимо поради архитектурни ограничения, каквито са хардуерната липса на паралелни логически блокове, синхронизация на данните обуславяща се от атомарността на инструкциите, и др. При време-критичните процесорни архитектури намалено времезакъснение се постига чрез разделянето на бавните процесорни инструкции на последователност от няколко по-малки примитиви. Такъв е примерът с операцията "*делене*" при сигналният процесор ADSP2100 на Analog Devices, чиито делител-операнд представлява броят извикани инструкции DEVQ [4]. По този начин се съкращава максималното време за изпълнение на процесорните инструкции, което забавя в по-малка степен извикването на процедурата за обработката на прекъсванията.

При някои архитектури се прилага мултиплексиране на прекъсванията, когато разполагаме само с един вектор за тяхната обработка. Това допълнително забавя извикването на съответната обработваща процедура чрез условен код за определяне на конкретното възникнало прекъсване. Това може да се избегне чрез архитектурното осигуряване на отделен вектор за всяко налично прекъсване.

Обикновено едни и същи процесорни регистри се използват както в програмата, така и при обработката на прекъсванията. Това налага временното преместване на съдържанието им в стека при възникване на прекъсване, използването им в обработващата го процедура, както и възстановяването на съдържанието на регистрите след завършването й. Често за увеличаване скоростта на изпълнение на обработката на прекъсванията се предоставят дубликат регистри, като едните се използват в програмата, докато другите се ползват единствено в прекъсванията. Това спестява време по прехвърляне на контекста между програмата и възникналото прекъсване, което се изразява в съхранение съдържанието на процесорните регистри в стека. Такава възможност предоставя архитектурата ARM, в която са дефинирани 5 отделни стека (IRQ32, FIQ32, Abort32, Undef32 и SVC32). Недостатък е невъзможността за изпълнението на прекъсване по време на обработката на друго, без прехвърляне на регистрите през стека. Но този недостатък може да се избегне чрез осигуряването на отделен набор регистри към всяко прекъсване.

Методи за бърз достъп до функции

Извикването на функция често е съпроводено със загуба на време поради необходимостта по прехвърляне на данни от извикващата към виканата функция. Двете функции разменят информация помежду си с помощта на стек, където се записват стойностите на аргументите, точката на възвръщане на управлението и резултатът на изпълнената функция.

Една от възможностите за редуциране на това времезакъснение се реализира с предаване на данните посредством регистри, като се премахне нуждата от запис и извличане на данни от стека. Ограничение на този метод е невъзможността на употребата на рекурсия, понеже функциите трябва да комуникират само с глобални статични променливи, т.е. да са реентрантни.

Друга възможност е снабдяването на процесора с допълнителни регистри, които да приемат стойностите на върха на стека. Така употребата на стек се запазва, и извикването на функция се осъществява за един процесорен такт. Този подход е реализиран в процесорите SPARC на Sun Microsystems [13].

Всички разгледани методи могат да се реализират чрез архитектурни процесорни подобрения, при които за сметка на увеличеното количество регистри се постига по-голямо бързодействие на изпълнение на програмата.

Оптимизация на времезакъсненията в специализираните схеми

По-бърз подход би могъл да е изпълнение на хардуерно реализирана специализирана схема (ASIC). Тя може да бъде напълно оптимизирана за изпълнението на конкретна задача. Програмируемите логики предлагат именно такава хардуерна платформа, позволяваща гъвкава разработка на схеми с разнообразно предназначение. Те позволяват

изграждането и на паралелни програмни структури. При подобна разработка се взимат предвид времезакъсненията на всички участващи в изчисленията тригери и комбинационни схеми, които по време на проектиране биха могли да се оптимизират допълнително чрез разпределяне и групиране на хардуерните изграждащи блокове на физически местоположения вътре в самата интегрална схема, така че времезакъсненията на сигналите формиращи общи данни да се сведат до минимум чрез промяна на дължината на междублоковите връзки. Чрез тези подходи се постига намаляване на критичният път в системата и свеждането му до стойности, задоволяващи заданието на конкретната поставена задача.

Паралелизъм

Постигането на паралелност е възможно в няколко направления – конвейерна обработка на командите, векторен паралелизъм и многопроцесорна платформа. Изразява се чрез едновременната обработка на няколко части на програмата.

Трябва да се отбележи, че бързодействието не е правопропорционално на броят използвани паралелни процесори поради необходимостта от допълнителна синхронизация помежду им при използването на общи ресурси. Обезпеченият брой процесори характеризират *мащабируемата* паралелна система, при която производителността нараства пропорционално с увеличаване на броя процесори.

Конвейерен паралелизъм

Конвейерната обработка на инструкциите принадлежи към паралелизъм на ниво инструкции. Тя представлява разделяне на подлежащата за изпълнение базова функция на подфункции и изпълнението им от отделни апаратни блокове. За всеки момент от време всеки апаратен блок е ангажиран с изпълнението на определена подфункция, в което се изразява и паралелизмът [1]. Максималната скорост за изпълнение на подфункциите съответства на скоростта за изпълнение на най-бавната от тях. За прилагане на конвейерна обработка е необходимо изчислението на базовата функция да е еквивалентно на последователността от изчисленията на подфункциите; величините явяващи се входни за дадена подфункция да се явяват изходни за предходната подфункция; действията, реализирани от тези апаратни блокове да отнемат приблизително едно и също време за изчисление.

Конвейерната обработка може да се прилага независимо от останалите методи за увеличаване на производителността на изчислителните устройства. Тя е удобна за реализация, когато имаме повтаряемост и детерминираност на последователни операции, които могат да се разделят на подоперации с близки времена на изпълнение. Детерминираността се обуславя от липсата на условни преходи, които затрудняват разделянето на програмният поток на порции за обработка чрез конвейера.

Паралелизъм на достъпа до паметта

Обикновено изпълнението на една инструкция се свежда до операция, приложена върху два операнда. За бързото й изпълнение е необходимо едновременното прочитане на инструкцията и операндите от паметта. Това налага използването на Харвард-архитектура, при която програмната памет е независима и отделена от паметта за данни, като всяка е снабдена със собствен набор от адресна шина и шина за данни [5]. Възможно е реализирането на система, снабдена с няколко памети за данни. Това ще повиши потока на данни към изчислителния блок и ще позволи реализацията на SIMD-архитектури. Неудобство представлява необходимостта от специално разделяне и форматиране на данните във всеки един блок памет.

Недостатък на такъв вид раздробяване на паметта на хардуерно независими блкове се изразява в ограничението на достъпа до дадена памет в даден момент от време.

Харвард-архитектурата е удобна за прилагане при паралелни независими помежду си изчисления, поради физически разделеният достъп до модулите памет.

Суперскаларна паралелност

Някои процесори могат да изпълняват по две или повече инструкции едновременно - такива са суперскаларните процесори. Подобно изпълнение е възможно само ако не съществува връзка между данните с които оперират паралелните инструкции. Те са удобни за изчисления на програмни цикли, работа с масиви и матрици от данни.
Представител на паралелно изчисление на ниво данни е векторният процесор. Той изпълнява една инструкция върху множество еднотипни данни.

Такъв вид обработка често се използва при необходимостта от презчисляване на координатите на множество точки от една координатна система в друга (например от полярни в декартови координати и обратно или барицентрична↔ декартова координатна система).

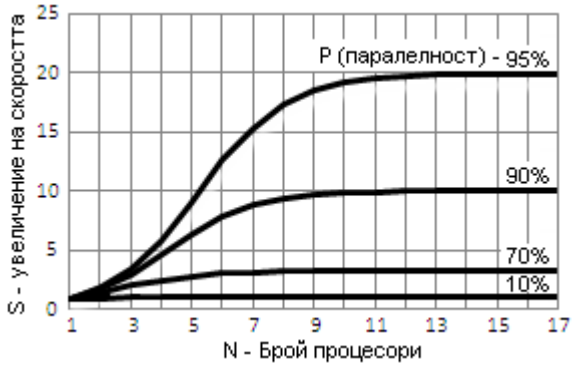
Многопроцесорни архитектури

Голяма част от съвременните специализирани процесори притежават междупроцесорна комуникация (Interprocessor communications) чрез външно изведена шина за данни и синхронизация, което позволява изграждането на многопроцесорни архитектури от еднотипни модули. Този подход за увеличаване на бързодействието на обработката на информация се прилага когато разполагаме с поток от данни, взаимно свързани помежду си. Такива възможности за разширяване притежават TMS320C40 на Texas Instruments [7] и ADSP21060 на Analog Devices [8]. Многопроцесорните и многоядрени компютри се състоят от множество изчислителни клетки, поместени в една машина. Когато процесорите са еднотипни се говори за симетрични многопроцесорни системи. Постигането на по-голяма производителност в зависимост от броя ядра или процесори се изразява чрез закона на Амдал (Amdahl's Law)[12]:

$$S = \frac{1}{(1-P) + \frac{P}{N}} \quad , \qquad (1)$$

където ‘P’ е тази част от програмата, която може да бъде подложена на паралелна обработка,

‘(1 - P)’ е частта, неподлежаща на разпаралелване,
N е броят процесори, които извършват паралелната работа,
S – максимално увеличение на скоростта, което може да бъде постигнато с N-процесора



Фиг. 1. **Зависимост на скоростта от броят процесори**

Теоретично при $N \to \infty$ максималното увеличение на скоростта ще клони към 1/(1-p). Отношението увеличена производителност спрямо цената на процесорен модул намалява спрямо броят използвани процесори, което означава че внедряването им е изгодно само до определен брой (фиг. 1).

Клъстерите и мрежовите (grid) компютри използват няколко машини, свързани в мрежа. При тях паралелността на информацията се изразява в разпределяне на данните между няколко изчислителни единици, работещи едновременно.

Многоядрените, многопроцесорните, клъстерните и мрежовите изчисления прилагат паралелност на ниво задачи. Паралелизъм на ниво задачи се определя като едновременно изпълнение на набор от различни изчисления. За реализацията му се дефинира понятието нишка (thread), което представрява един набор от последователност от инструкции. Отделните нишки могат да боравят с общи (shared) или различни (distributed) данни. При първият вариант е необходим механизъм за синхронизация на достъпа до данните, което внася допълнителни времеви загуби в производителността на изчислителното устройство.

За софтуерното обслужване на паралелно изпълнение на задачи се използва предимно стандарта POSIX [9]. За многопроцесорно програмиране намира широко приложение комуникацията чрез съобщения (Message Passing Interface - MPI), които работят чрез FIFO структура за данни [10].

През 2009 г. бе създаден първият отворен стандарт за паралелен език с общо педназначение - OpenCL. Разработен от Кронос Група (Khronos Group), състояща се от водещи индустриални производители [11]. Той предоставя унифициран подход за изграждане на паралелни програми за микроконтролери, настолни компютри, както и многопроцесорни и разпределени системи и др.

Заключение

Всеки един от разгледаните методи за увеличаване на бързодействието на изчислителните машини може да бъде приложен за по-добра производителност. Удачна ще бъде разработката на система, базирана на събития, която да предлага голям набор от прекъсвания, обслужващи тези събития. Прекъсванията могат да работят в паралел чрез използване на методи за изграждане на паралелна обработка на данните. Производителността ще се повиши при редуциране на споделените ресурси за сметка на разпределените такива.

Литература

ADSP21xx, Programming Model, secondary registers, http://student.agh.edu.pl/~pmrosz/Motka/programming_model.pdf

ADSP-2100 Family DSP Microcomputers, http://www.analog.com/static/imported-files/data_sheets/ADSP-2101_2103_2105_2115.pdf

ADSP-2106x SHARC DSP Microcomputer Family, A multiprocessing system, Analog Devices, http://datasheet.digchip.com/041/041-4-000688-ADSP21060.pdf

Barney B., L.Livermore, POSIX Threads Programming, National Laboratory, https://computing.llnl.gov/tutorials/pthreads/

Barney B., L.Livermore, Message Passing Interface (MPI), https://computing.llnl.gov/tutorials/mpi/

Gambier A., Real-time Control Systems: A Tutorial, http://ascc2004.ee.mu.oz.au/proceedings/papers/P150.pdf

Introduction to DSP - DSP processors: memory architectures, http://www.bores.com/courses/intro/chips/6_mem.htm

Khronos Group, OpenCL - The open standard for parallel programming of heterogeneous systems, http://www.khronos.org/opencl/

Larson K., J. Patterson 1993, Designing with TMS320C40 Comm Ports, Digital Signal Processing Products, Semiconductor Group, Texas Instruments, http://focus.ti.com.cn/cn/lit/an/spra213/spra213.pdf

National Instruments Corporation, Real-Time Event Response, http://zone.ni.com/devzone/cda/tut/p/id/3938

Prabhu G. M., Computer architecture tutorial, http://www.cs.iastate.edu/~prabhu/Tutorial/title.html

Shi Y., Reevaluating Amdahl's Law and Gustafson's Law, http://www.cis.temple.edu/~shi/docs/amdahl/amdahl.html

SPARC, an extreme windowed RISC (Part II), http://www.cpushack.com/CPU/cpu4.html

Препоръчана за публикуване от кат. «Автоматизация на минното производство», МЕМФ