Spezialarchitektur oder Standardlösung? Eine Fallstudie am Beispiel der linearen Optimierung

Bernd Schütz
Universität Hamburg, Fachbereich Informatik
Troplowitzstr. 7, 2000 Hamburg 54

Zusammenfassung

Die Entwicklung einer dedizierten Hardware verlangt in einem frühen Stadium des Entwurfsprozesses die Entscheidung, ob eine Spezialarchitektur entwickelt werden soll, oder ob auf eine Standardlösung zurückgegriffen werden kann. Anhand eines Beispiels zur Hardwarebeschleunigung für eine Aufgabe aus der linearen Optimierung werden verschiedene Fragen diskutiert, die helfen, diese Entwurfsentscheidung zu treffen.

Einführung

Rechnerarchitekten wird häufig die Aufgabe gestellt, dedizierte Rechnerarchitekturen, im folgenden Spezialarchitekturen genannt, zu entwickeln. Aufgabe derartiger Spezialarchitekturen ist es, bestimmte Algorithmen und Verfahren bzw. eine Klasse von Algorithmen besonders effizient auszuführen. Insbesondere bei der Konzeption von Hardwarebeschleunigern hat der Entwickler sich der Frage zu stellen, ob eine Spezialarchitektur entworfen werden sollte, oder ob mit einer auf dem Markt verfügbaren Standardlösung die Aufgabenspezifikation erfüllt werden kann. Wirtschaftliche Überlegungen sind ebenfalls zu berücksichtigen. Anhand einer Entwicklungsaufgabe eines Hardwarebeschleunigers für ein Verfahren der linearen Optimierung werden die verschiedenen Fragen und Ansätze zur Bestimmung dieser grundlegenden Entwurfsentscheidung diskutiert.

Das Fallbeispiel

Als Beispiel für eine Hardwareimplementation soll das Simplexverfahren dienen. Das Simplexverfahren wurde maßgeblich durch George. B. Dantzig 1947/48 [2] eingeführt und gehört heute zu den wichtigsten Verfahren der linearen Optimierung [1]. Bild 1 zeigt ein mathematisches Modell linearer Optimierungsprobleme, die mit dem Simplexverfahren gelöst werden können. Die Variablen $x_1, \ldots, x_n$ könnten z.B. die Produkte einer Fabrik darstellen. Die Nebenbedingungen stellen dann die Zusammenhänge zwischen Produkten und benötigten Ressourcen her. In die Zielfunktion $f(x_1, \ldots, x_n)$ könnten beispielsweise die Deckungsbeiträge der Produkte eingehen. Aufgabe ist es nun, eine zulässige Belegung der Variablen $x_1, \ldots, x_n$ zu bestimmen, so daß der Wert der Zielfunktion f maximal wird. Typische Anwendungsgebiete der linearen Optimierung und somit des Simplexverfahrens sind Transportprobleme, Mischungsprobleme, Aufgaben in Produktionsplanungssystemen, etc. (vgl. hierzu [5]). Auch in anderen Bereichen, beispielsweise Schedulingaufgaben in der Hardwaresynthese, finden sich Anwendungsfälle des Simplexverfahrens. Da es sich hier meist um gemischt ganzzahlige Probleme (MILP, Mixed Integer Linear Programming) handelt, wird zur Lösung in der Regel ein Branch-and-Bound Verfahren eingesetzt. Die Knoten repräsentieren dann lineare Optimierungsprobleme, die unabhängig voneinander (parallel) mit dem Simplexverfahren gelöst werden können.

Mit dem Beispiel für die Entwicklung des Hardwarebeschleunigers wurde eine Idee aufgegriffen, die einer Zusammenarbeit mit dem Fachbereich Wirtschaftswissenschaften der Universität Hamburg entstammt. Dafür stellte uns der Fachbereich Wirtschaftswissenschaften praxisgerechte LP-Aufgaben und einen Beispielalgorithmus zur Verfügung. Des weiteren war er bei der Festlegung der Randbedingungen behilflich. Hier wurden Problemgrößen, Genauigkeitsanforderungen und die zugrundeliegende Hardware definiert: Als Basishardware wurde ein PC/AT gefordert. Aufgabe war es also, für den IBM

$$\sum_{j=1}^{n} a_{ij} \cdot x_j \leq b_i, \quad i = 1, ..., m; \quad n \leq m,$$

$$\forall x_j : x_j \geq 0$$

$$f(x_1, \ldots, x_n) = \sum_{j=1}^{n} c_j \cdot x_j \quad \rightarrow \quad Max,$$

Abbildung 1: Standardaufgabe der linearen Optimierung

PC/AT eine preisgünstige Hardwareerweiterung bereitzustellen, die das Simplexverfahren in seiner Ausführungszeit beschleunigt.

Voruntersuchungen

Die Beantwortung der Frage Spezialarchitektur oder Standardlösung läßt sich nur durch eine eingehende Analyse des Problemkreises, also Verfahren, Daten, Kosten, herbeiführen. Begonnen werden soll mit der Untersuchung der Eingabedaten.

Analyse der Eingabedaten: Die Untersuchung der Eingabedaten zeigte, daß die Eingabematrizen sehr schwach besetzt sind. Die Füllung liegt typischerweise deutlich unter 5%. Des weiteren wurde beobachtet, daß etwa 40% – 60% der Nichtnullelemente die Werte -1 oder $+1$ aufweisen. Diese Beobachtungen geben zwar keinen Hinweis für die Klärung der übergeordneten Entwurfsentscheidung, sie lassen jedoch erkennen, daß dieser Umstand in einer Spezialhardware berücksichtigt werden sollte. Er könnte sich z.B. in einer Architektur mit *Tag-Bit* Mechanismen niederschlagen.

Untersuchung des Verfahrens/Algorithmus: In Anbetracht einer Hardwareimplementierung sollte das vorliegende Verfahren bzw. der Algorithmus eingehend untersucht werden. Zentraler Punkt ist hier, ob eine Änderung der Anforderungen an das Verfahren während der Benutzungsdauer wahrscheinlich ist. Ist dieses nicht ausgeschlossen, muß der Umfang der möglichen Änderung bewertet werden. Bei tiefgreifenden Änderungsmöglichkeiten (Änderung des Verfahrens/anderes Verfahren) scheidet eine Spezialarchitektur zwar nicht generell aus, büßt aber deutlich an Vorzügen ein (teilweiser Verlust der Vorteile der Hardwareimplementation).

Parallelverarbeitung: Unter der Zielsetzung einer hohen Verarbeitungsleistung ist der Einsatz von Parallelverarbeitung erforderlich. Dabei ist zwischen externer und interner Parallelisierbarkeit zu unterscheiden. Externe Parallelität ist beispielsweise bei Problemen der gemischt ganzzahligen Optimierung gegeben. Interne Parallelität bezeichnet die parallele Ausführung von Teilen des Algorithmus unter Berücksichtigung der Datenabhängigkeiten.

Für die gegebene Aufgabenstellung läßt sich die externe Parallelverarbeitung dadurch verwirklichen, daß mehrere Hardwarebeschleunigerkarten an das Grundgerät IBM PC/AT angeschlossen werden. Die internen Parallelisierungsmöglichkeiten können sich nur in der Architektur des Hardwarebeschleunigers niederschlagen.

Arithmetiken: Deuten sich im Bereich der Numerik Schwierigkeiten an, wie auch beim vorliegenden Beispiel, so ist zu prüfen, ob das Verfahren entsprechend modifiziert werden kann, so daß es sich über der konventionellen IEEE-Floatingpoint-Arithmetik numerisch stabil verhält. Sind hier keine befriedigenden Ergebnisse zu erzielen, so ist der Einsatz alternativer Rechnerarithmetiken (Intervallmethoden, X/Y-Arithmetiken, Residuen-Arithmetiken, etc.) zu erwägen. Diese alternativen Arithmetiken erfordern dann allerdings in der Regel eine dedizierte Hardware, da sie auf konventionellen Rechnerarchitekturen nicht effizient ausführbar sind. Andererseits ist zu bedenken, daß sich bei ungenügender numerischer Stabilität des Verfahrens eine Hardwareimplementation verbietet. Bei einer Softwarelösung kann bei Problemen in bestimmten Konstellationen schnell partiell nachgebessert werden; bei einer Hardwareimplementation ist dies nicht möglich! Im vorliegenden Fall wurde aus pragmatischen Erwägungen die IEEE-Floatingpoint-Arithmetik gewählt. Durch Hinzufügen von Epsilontiken und Kriterien für die Nullelementerkennung wurde eine ausreichende numerische Stabilität des Verfahrens erreicht. Diese Zusatzfunktionen lassen sich besonders effizient in Hardware lösen; ihr Aufwand bei Implementierung auf einer konventionellen Hardware bleibt allerdings auch in vertretbaren Grenzen.

Zwischenbilanz: Bisher wurden einige grundlegende Punkte angesprochen, die helfen können, die Frage nach Spezialarchitektur oder Standardlösung zu entscheiden. Im Falle der Beispielaufgabe ist aber hiernach keine eindeutige Entscheidung zu treffen. Auch weitere, detailliertere und hier nicht im einzelnen angesprochene Untersuchungen brachten keine Klärung. Die Realisierung mit anschließendem Vergleich beider Möglichkeiten soll nun das Für und Wider zu den Alternative aufzeigen.

Die Spezialarchitektur

Bild 2 gibt einen groben Überblick der entworfenen Spezialarchitektur wieder [4]. Der LPA (Linear Programming Accelerator) gliedert sich grob in Master Control Unit (MCU), in 1 – 8 parallel angeordnete Processing Units (PUs) und den globalen Systemspeicher. Die MCU übernimmt das Task-Scheduling und verwaltet die globalen Systemressourcen. Die PUs sind identisch aufgebaut und enthalten neben einer lokalen Kontrolleinheit (LCU) die Floatingpoint-Einheiten. Die Architektur zeichnet sich durch folgende Punkte aus:

+ Flexible Konfigurierbarkeit
+ Parallelität auf mehreren Ebenen (zwischen Host & LPA, zwischen PUs und zwischen PU & FPU)
+ Vermeidung redundanter Berechnungen durch Sonderbehandlung von Nulloperanden
+ spezielle Adreßberechnungshardware für strukturierte Datentypen auf Maschinenprogrammebene
− globaler Speicher als Systemengpaß
− veraltete FPUs mit aufwendiger Ansteuerung

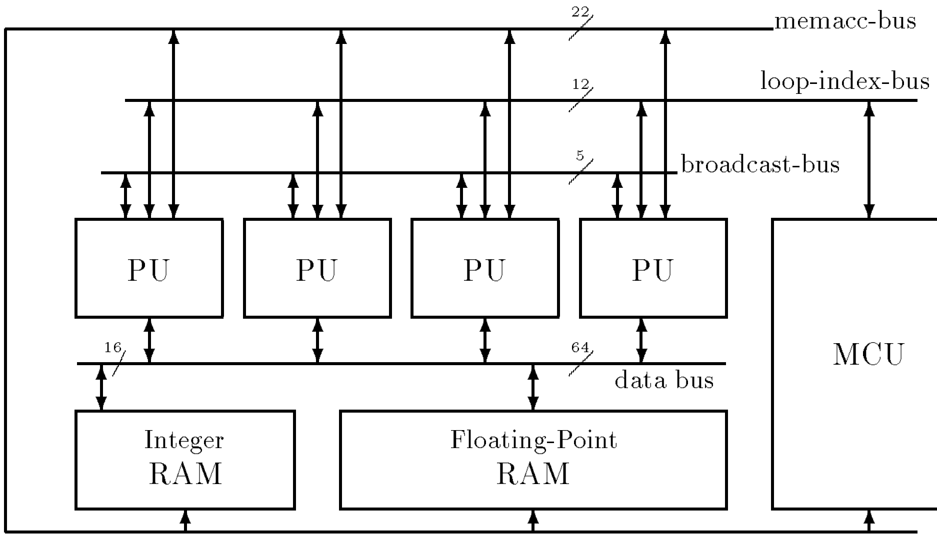

Abbildung 2: Überblick der LPA-Architektur

Die Standardlösung

Als Standardlösung wird eine kommerziell verfügbare Zusatzkarte mit Intel i860 Prozessor und on-Board Speicher gewählt. Das als C-Programm vorliegende Simplexverfahren wurde zunächst unverändert mit dem verfügbaren C-Compiler zu übersetzt; ggf. können noch einige Handoptimierungen im i860-Code durchgeführt werden. Der Intel i860 ist aufgrund seiner Architektur und seiner hohen Floatingpoint-Leistung für die vorliegende Aufgabe sehr gut geeeignet [3].

# NB	# Var	CPU-Zeit/Pivotschritt [ms] LPA-4	i860	Apollo DN3000	SUN SS2
61	69	11.7	9.2	318	8.4
118	52	13.9	18.3	625	20.5
88	116	12.1	13.0	524	14.1
166	77	67.2	39.3	1899	59.3
292	251	201.7	68.3	3285	230.8
94	88	32.0	23.2	692	38.0

Abbildung 3: Laufzeiten einiger typischer LP-Probleme

Vergleich der Ergebnisse

Aus der Analyse des Verfahrens und der Eingabedaten ließ sich nicht eindeutig entscheiden, ob eine Standardlösung oder eine Spezialarchitektur gewählt werden sollte. Aus diesem Grunde wurden beide Varianten realisiert, um Auskunft über die jeweilige Leistungsfähigkeit zu erhalten.

Abb. 3 zeigt die Laufzeiten von einigen typischen LP-Problemen jeweils auf unserer Spezialarchitektur LPA, auf der Standardlösung mit Intel i860 und zum Vergleich noch mit Apollo DN3000 und Sun SS2. Das i860-Board zeigt sich im Vergleich als leistungsstärkere Alternative, was vor allem in der hohen Floatingpoint-Leistung begründet ist.

Schlußbetrachtung

Die Voruntersuchungen führten zu keinem klaren Votum. Die Realisierung beider Varianten zeigte dann, daß in diesem Falle die Standardlösung die bessere Wahl ist. Der Grund hierfür ist in der um Größenordnungen besseren Floatingpoint-Leistung des i860 gegenüber den Leistungswerten der in der Entwicklung des LPA verwendeten Floatingpoint-Einheiten zu sehen. Trotzdem zeigt der Vergleich, daß auch bei Verwendung relativ langsamer Floatingpoint-Prozessoren durch die Entwicklung dedizierter Architekturen eine vergleichbare Verarbeitungsgeschwindigkeit erreicht werden kann.

Obwohl die Verwendung eines leistungsfähigen Standardprozessors auch unter Einbeziehung von Kostenüberlegungen hier sicherlich die günstigere Entscheidung ist, wäre es allerdings falsch, Spezialarchitekturen grundsätzlich abzulehnen. Diese Tatsache verdeutlicht bereits das vorliegende Beispiel. Aus pragmatischen Gründen wurden kommerzielle Floatingpointeinheiten des Types M68881/M68882 gewählt und die erforderlichen ASICs in Standardzelltechnik entworfen. Die daraus resultierende Betriebsfrequenz von 12MHz wie die eingesetzten Floatingpoint-Einheiten entsprechen in etwa der Technik einer APOLLO DN3000 Workstation. Abbildung 3 verdeutlicht eindrucksvoll, wie hoch hier der Gewinn ausfällt, der durch eine Spezialarchitektur erreicht werden kann.

Literatur

[1] K. H. Borgward. *The Simplex Method - A Probabilistic Analysis.* Springer-Verlag, Berlin, Heidelberg, New York, London, Paris, Tokyo, 1987.

[2] G. B. Dantzig. *Linear Programming and Extensions.* Princeton University Press, Princeton, 1963.

[3] Intel. *i860 64-Bit Microprocessor Programmers's Reference Manual.* Santa Clara, 1989.

[4] A. Klindworth. *Entwurf, Simulation und Realisierung einer parallelen Rechnerarchitektur für ein revidiertes Simplex-Verfahren.* Diplomarbeit, Fachbereich Informatik, Universität Hamburg, 1992

[5] H. Müller-Merbach. *Operations Research.* Vahlen, München, 1973.