Wieviel Platz brauchen Algorithmen wirklich?

Multiplizieren und Dividieren mittels flacher Schaltkreise

Benedikt Rieger

13. August 2010

Zusammenfassung

Die Multiplikation und Division zweier Zahlen sind grundlegende Operationen, die täglich in fast allen Lebensbereichen gebraucht werden. Aus diesem Grund ist es wichtig, diese Operationen effizient durchzuführen. Folgend werden Algorithmen vorgestellt, um eine Multiplikation sowohl in der Klasse NC^2, als auch in NC^1 umzusetzen. Des Weiteren wird gezeigt, dass die Division in der Klasse NC^2 mit Hilfe der Newton-Iteration realisiert werden kann.

1 Multiplikation

Bei der Multiplikation werden zwei Zahlen der Länge n zu einer Zahl der Länge $2n$ multipliziert. Dabei soll der Schaltkreis für die Multiplikation möglichst flach sein. Ein NC^2-Algorithmus ist recht einfach zu implementieren. Die Berechnung in der Klasse NC^1 ist ein etwas schwierigeres, aber lösbares Problem.

1.1 NC^2-Algorithmus

Dieser Algorithmus wird häufig *Schulmethode* genannt, da die Vorgehensweise ähnlich derer ist, wie ein Schüler die Multiplikation in der Schule beigebracht bekommt. Die Multiplikation zweier Zahlen a und b wird durch die b-Fache Addition von a berechnet. Auf parallelen Systemen wird diese Methode wie folgt implementiert:

1. Ermittele n Zahlen, deren Summe das gesuchte Produkt ist
2. Die n Zahlen werden baumartig addiert, demnach hat der Additionsbaum die Tiefe $O(\log n)$
3. Zwei Zahlen lassen sich in einer Tiefe von $O(\log n)$ addieren
4. Die Gesamttiefe der Multiplikation beträgt $O(\log^2 n)$

Beispiel $1101 \cdot 0111 =$

$$\left.\begin{array}{l}\left.\begin{array}{l}00001101\\00011010\end{array}\right\} 00100111\\\left.\begin{array}{l}00110100\\00000000\end{array}\right\} 00110100\end{array}\right\} 01011011$$

1.2 NC^1-Algorithmus

Auch bei dem NC^1-Algorithmus werden n Zahlen ermittelt, die addiert werden. Problematisch ist jedoch, dass schon die Addition von zwei Zahlen eine Tiefe von $O(\log n)$ braucht. Daher muss eine Methode gefunden werden, Zahlen in der Tiefe $O(1)$ zu addieren. Zwei Zahlen können nicht in der Tiefe $O(1)$ addiert werden, jedoch ist dies bei der Addition von drei Zahlen zu zwei der Fall: $a + b + c = d + e$
Für jede Bitstelle wird unabhängig voneinander die Bits der Zahlen a, b und c addiert. Das Ergebnis ist jeweils eine zwei Bit lange Summe bestehend aus dem Übertrag und die Parität. Die Zahl d enthält genau die Paritätsbits, die Zahl e enthält die Folge dieser Überträge um eins nach links verschoben. Der Algorithmus wird nach folgendem Prinzip abgearbeitet:

1. Ermittele n Zahlen, deren Summe das gesuchte Produkt ist
2. Ersetzte jeden Block von drei Zahlen durch zwei Zahlen
3. In den verbleibenden $2/3 \cdot n$ Zahlen, ersetzte jeden Block von drei Zahlen durch zwei Zahlen
4. In den verbleibenden $4/9 \cdot n$ Zahlen, ersetzte jeden Block von drei Zahlen durch zwei Zahlen
5. Dieser Vorgang wird fortgesetzt, bis nur noch zwei Zahlen übrig sind
6. Diese werden in Tiefe $O(\log n)$ addiert

Beispiel einer Addition $a + b + c = d + e$

$$\left.\begin{array}{l} 0110100001011001\ (a) \\ 0000111011000000\ (b) \\ 0000001100000000\ (c) \end{array}\right\} \begin{array}{l} 0110010110011001\ (d) \\ 0001010010000000\ (e) \end{array}$$

2 Division

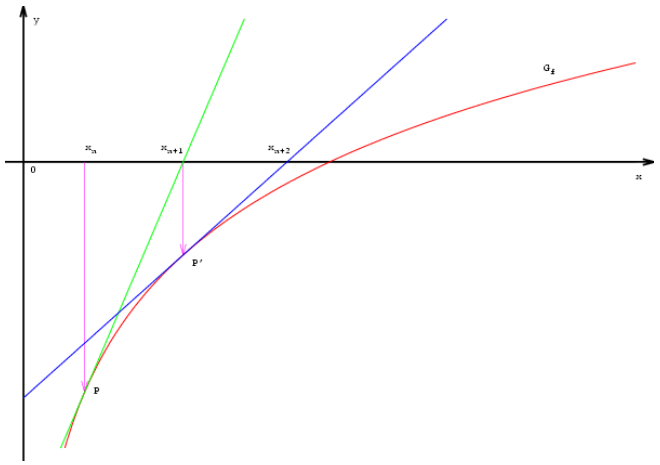
Zwei Zahlen der Länge n werden dividiert, das Ergebnis liefert eine Zahl der Länge n. Wie bei der Multiplikation soll auch hier der Schaltkreis möglichst flach sein. Im vorhergehenden Kapitel wurde gezeigt, dass eine Multiplikation zweier Zahlen in der Tiefe $O(\log n)$ berechnet werden kann. Das Ergebnis der Division c/d ist gleich der Multiplikation mit dem Kehrwert $c \cdot (1/d)$. So kann die Division in eine Multiplikation umgewandelt werden. Nun muss der Kehrwert $(1/d)$ berechnet werden. Dies kann mit einer Approximation $d' = 1/d$ erreicht werden.
Die Approximation wird mit der Newoton-Methode durchgeführt. Mit der Newton-Methode lassen sich bei einer gegebenen stetig differenzierbaren Funktion $f : \mathbb{R} \to \mathbb{R}$ Näherungswerte zur Lösungen der Gleichung f(x) = 0 finden.

Der Newton-Algorithmus wird wie folgt abgearbeitet:

1. Startwert raten
2. Ausgangspunkt linearisieren
3. Nullstelle der Tangente als verbesserte Näherung der Nullstelle der Funktion verwenden

Für das Newton-Verfahren soll eine sinnvolle Funktion f gefunden werden, deren Nullstelle der Kehrwert von d ist. Hierfür bietet sich Funktion

$$f(x) = d - 1/x$$



an. Aus der Newton-Methode folgt, dass sich mit der Gleichung

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{d - 1/x_n}{1/x_n^2} = x_n(2 - dx_n)$$

die n-Approximation berechnen lässt. Um die Approximation zu erleichtern, wird der Eingabewert d der Newton-Approximation auf das Intervall $1/2 \leq d \leq 1$ beschränkt. Der Startwert wird bei $x_0 = 1$ festgelegt.

Die Iteration $x_{n+1} = x_n(1 - dx_n)$ wird log n mal angewendet. Jeder dieser Iterationen benötigt die Tiefe $O(\log n)$. Das Ergebnis wird mit a multipliziert, das wiederum die Tiefe $O(\log n)$ hat. Demnach ist die gesamttiefe der Division mit dem Newton-Verfahren $O(\log^2 n)$