Verabeitungsleistung, Stromaufnahme und Implementierungseffizienz

Prof. Dr. Wolfgang Matthes
Peukinger Weg 34
59423 Unna
http://www.realcomputerarchitecture.com

> Zum Philosophieren sind die zwei ersten Erfordernisse diese: erstlich, daß man den Mut habe, keine Frage auf dem Herzen zu behalten, und zweitens, daß man alles das, *was sich von selbst versteht*, sich zum deutlichen Bewußtsein bringe, um es als Problem aufzufassen.
>
> *Arthur Schopenhauer*

Vorgeschichte

Der im folgenden skizzierte Ansatz wurde in den 80er Jahren entwickelt, um objektive Bewertungskriterien für die Entwicklung von Spezialprozessoren zu gewinnen. Am Anfang stand die Überlegung, daß es sicherlich ohne weiteres möglich ist, für beliebige Algorithmen zweckgebundene Schaltungslösungen zu entwerfen. Hierbei geht allerdings die programmtechnische Flexibilität verloren, und es kann auch vorkommen, daß der Leistungsgewinn in keinem vernünftigen Verhältnis zu den Aufwendungen steht. Es ist also zweckmäßig, das Verhältnis von Leistungsgewinn und Aufwand genauer zu untersuchen.

Der Grundgedanke

Für einen einzelnen Algorithmus ist eigentlich nur die Ausführungszeit von Bedeutung, die Zeit, die die Maschine braucht, um die gewünschten Ergebnisse zu liefern.

Ein grundsätzliches Leistungsmaß

Es ist grundstzälich möglich, für den jeweils in Rede stehenden Algorithmus eine Einzweckmaschine zu entwerfen. Daß programmgesteuerte Universalmaschinen überhaupt verwendet werden, hat unter diesem Gesichtspunkt nur den Grund, daß es nicht durchführbar ist, für jeden Algorithmus eine Sondermaschine auch tatsächlich zu bauen. Die programmgesteuerte sequentielle Ausführung eines Algorithmus ist aus dieser Sicht nur ein Notbehelf. Man kann deshalb die Abarbeitung von Befehlen eher als Störfaktor auffassen und nur die Operanden und Resultate der Algorithmen betrachten. Diese Werte sind binär codiert, und sie werden in aufeinanderfolgenden Taktzyklen von Speichermitteln über Verbindungsleitungen und kombinatorische Netzwerke zu Speichermitteln bewegt.

Um ein Maß für die Verarbeitungsleistung zu gewinnen, ist es mithin ausreichend, zu zählen, wieviele Bits an Nutz-Information (Operanden und Resultate der jeweils betrachteten Algorithmen) in einer bestimmten Zeit verarbeitet und erzeugt werden.

$$PM = \frac{1}{t_x} \sum_{i=1}^{r} CARDB_i$$

(PM: Leistungsmaß (effektive Bits/s); t_x: Zeitintervall; CARDBi: Anzahl der transportierten Nutzbits im Maschinentakt i.)

Dieser Ansatz ist nicht nur zum Bewerten von Algorithmen und Schaltungslösungen nutzbar, wenn es um Leistungssteigerung geht. Er eignet sich vielmehr auch zum Bewerten der Energieeffizienz.

Die Stromaufnahme moderner Logikschaltungen

Hochintegrierte digitale Schaltkreise werden in CMOS-Technologien gefertigt. Im Innern der CMOS-Schaltungen gibt es keine ständigen Stromwege. Sind alle Verbindungen geschaltet, so können gar keine Stromflüsse auftreten (die Leckströme sind vernachlässigbar gering). Das gilt aber dann nicht, wenn sich der Schaltzustand ändert. Ein unendlich schnelles und perfekt gleichzeitiges Umschalten kann es nicht geben. Jeder Schaltvorgang ist vielmehr ein stetiger Übergang zwischen den beiden Bereichen der Logikpegel. Hierbei werden die bisher gesperrten Transistoren mehr und mehr leitend und die bisher leitenden Transistoren mehr und mehr gesperrt. Die zwischen Versorgungsspannung und Masse angeordneten Transistoren ergeben so zeitweise einen endlichen Widerstand, also eine Stromweg, über den ein Querstrom fließt. Die gesamte durch die Schaltvorgänge bedingte (dynamische) Stromaufnahme einer CMOS-Logikschaltung ergibt sich:

- infolge der Querströme von der positiven Speisespannung zur Masse (Abb. 1),
- durch das Umladen der interne Kapazitäten,
- durch das Umladen der Lastkapazitäten an den Ausgängen.

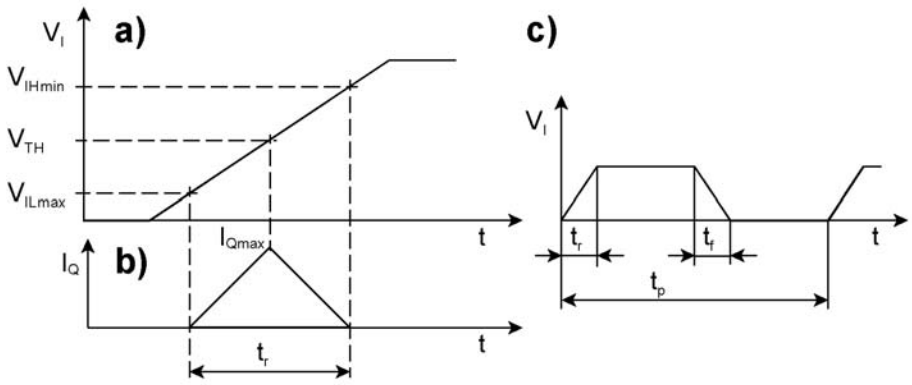

Abb. 1 Der Querstrom beim Umschalten. a) eine Signalflanke mit der Anstiegszeit t_r; b) der zeitliche Verlauf des Querstroms; c) eine Signalperiode. Jede Signalflanke bewirkt, dass ein Querstrom fließt.

Ein Querstrom fließt nur, wenn der Signalpegel den verbotenen Bereich durchläuft. Er hat dann sein Maximum, wenn der Signalpegel der Schwellenspannung entspricht. Maßgebend für die durch den Querstrom bedingte Verlustleistung ist das Zeitintegral des Querstroms, also die in Abb. 1b dargestellte Dreiecksfläche ½ I_{Qmax} • t_r. Der Effektivwert ergibt sich durch Mittelwertbildung in Bezug auf die Periodendauer der Schaltvorgänge. Gemäß Abb. 1c sei ein Ein- und Ausschalten mit der Anstiegszeit t_r, der Abfallzeit t_f und der Periodendauer t_p angenommen:

$$I_{leff} = \frac{1}{2} I_{Qmax} \cdot \frac{t_r + t_f}{t_p}$$

Damit ergibt sich die Verlustleistung P_I zu

$$P_I = \left(V_{IHmin} - V_{ILmax}\right) \cdot I_{Ieff} = \frac{1}{2}\left(V_{IHmin} - V_{ILmax}\right) \cdot I_{Qmax} \cdot \frac{t_r + t_f}{t_p} \quad (1)$$

mit $I_{Qmax} = \frac{V_{TH}}{R_{DSon}}$.

Da die Pegelwerte in (1) der Versorgungsspannung proportional sind, wächst dieser Verlustleistungsanteil mit dem Quadrat der Versorgungsspannung: $P_I = O(V_{CC}^2)$.

Bei hinreichend steilen Schaltflanken sind die Querströme im Vergleich zu den Strömen, die sich aus dem Umladen von Kapazitäten ergeben, typischerweise vernachlässigbar gering*.

*: Aber Achtung – bei offenen Eingängen oder zu flachen Flanken = zu langen Anstiegszeiten sind sie *nicht* zu vernachlässigen ...

Eine durch Umladevorgänge bedingte Verlustleistung P_C ergibt sich grundsätzlich zu

$$P_C = C \cdot f \cdot V_{DD}^2 \quad (2)$$

Sie ist also direkt proportional der umzuladenden Kapazität C, der Frequenz f der Schaltvorgänge und dem Quadrat der Versorgungsspannung V_{DD}. (Daraus folgt sofort die Notwendigkeit, die Versorgungsspannung so niedrig wie möglich zu halten.)

Die gesamte dynamisch bedingte Leistungsaufnahme P_{dyn} hat zwei Anteile, die jeweils gemäß (2) gebildet werden:

- Die transiente Leistungsaufnahme P_T, die vor allem durch dasUmladen der Gatterkapazitäten bedingt ist.
- Die ausgangsseitige Leistungsaufnahme P_L, die sich aus dem Umladen der kapazitiven Lasten ergibt.

$$P_{dyn} = P_T + P_L \quad (3)$$

$$P_T = C_{pd} \cdot V_{DD}^2 \cdot f_I \cdot n_s \quad (4)$$

$$P_L = C_L \cdot V_{CC}^2 \cdot f_O \cdot n_s \quad (5)$$

(f_i, f_O - Schaltfrequenzen (an den Ein- und Ausgängen); n_s - Anzahl der schaltenden Bitpositionen.)

C_{pd} ist eine eigens zur Leistungsberechnung definierte Kapazitätsgröße (Power Dissipation Capacitance), die im Datenblatt angegeben ist. Gibt es mehrere unterschiedliche kapazitive Lasten, die mit unterschiedlichen Frequenzen geschaltet werden, ist P_L im Sinne eines Erwartungswertes zu bilden:

$$P_L = \sum_n \left(C_{Ln} \cdot f_{On}\right) \cdot V_{CC}^{\ 2} \qquad (6)$$

Die gesamte im Schaltkreis umgesetzte Verlustleistung P_{tot} ergibt sich als Summe von statischer und dynamischer Verlustleistung:

$$P_{tot} = P_{stat} + P_{dyn} \qquad (7)$$

Der Strombedarf steigt linear mit der Betriebsfrequenz. Alle schaltenden Signale führen zum Umladen von Kapazitäten und erhöhen so den Strombedarf, auch wenn sie funktionell nichts bewirken (z. B. Adressen an nicht ausgewählten Speicherschaltkreisen). Um die Stromaufnahme zu verringern (Stromsparen), sind also so viele Signale wie möglich auf einem zulässigen Logikpegel ruhigzustellen.

Mit anderen Worten: man kann sich zwar sehr viele Transistoren leisten, sollte aber danach trachten, daß sowenige wie möglich gleichzeitig schalten.

Wenn man sich viele Transistoren leisten kann, kann man es sich auch leisten, sie für viele vergleichsweise einfache (aber nicht allzu einfache) Verarbeitungsschaltungen einzusetzen. Hieraus ergibt sich die Möglichkeit, für jeden im Programm angewiesenen Verarbeitungsvorgang eine Verarbeitungsschaltung vorzusehen und die jeweils zu verarbeitenden Variablen direkt in den Speichermitteln (z. B. Registern) der Verarbeitungsschaltungen zu halten. Mit anderen Worten, die zu verarbeitenden Variablen residieren nicht in zentralen Speichermitteln, aus denen sie zwecks Verarbeitung immer wieder geholt werden müssen, sondern unmittelbar in den Verarbeitungsschaltungen, die sich die meiste Zeit über in Ruhe befinden und nur dann aktiviert werden, wenn die Verknüpfungsergebnisse benötigt werden (Abb. 2).

Implementierungseffizienz

Was ist eigentlich die ideale technische Vergegenständlichung eines Algorithmus? – Doch wohl eine Anordnung, die aus den Eingangsdaten (Operanden) die Resultate unmittelbar, sozusagen auf einen Schlag bildet (d. h. durch unmittelbare Zuordnung und nicht durch zeitliche Folgen aufeinanderfolgender Verarbeitungschritte). Abb. 3 zeigt das geradezu triviale Blockschaltbild. Offensichtlich läßt sich so etwas nur dann bauen, wenn die Anzahl der Bits (bzw. Signalleitungen) nicht allzu hoch ist und wenn die Verknüpfungen nicht allzu kompliziert sind. Ebenso offensichtlich ist ein Ausweg: die abschnittsweise Verknüpfung (Abb. 4). Eine solche Anordnung arbeitet taktgesteuert, wobei in jedem Takt aus den jeweils ausgewählten Operandenabschnitten die jeweiligen Ergebnisabschnitte gebildet werden. Die höchste überhaupt mögliche Verarbeitungsleistung einer solchen Anordnung wird dann erreicht, wenn in jedem Maschinentakt ein Abschnitt des Ergebnisses gebildet werden kann, dessen Bitanzahl der Verarbeitungsbreite der Anordnung entspricht bzw. wenn es ausreicht, zur Bildung des Ergebnisses auf jeden Operandenabschnitt genau einmal zuzugreifen. Ob dies überhaupt gelingen kann, ist letzten Endes eine Eigenschaft des Algorithmus.

1. Beispiel (wo es klappt): die Bildung des Skalarprodukts zweier Vektoren.

2. Beispiel (wo es im allgemeinen Fall nicht klappt): Sortierabläufe. Hierbei ist es praktisch nicht zu vermeiden, auf die einzelnen Operandenabschnitte mehrmals zuzugreifen.

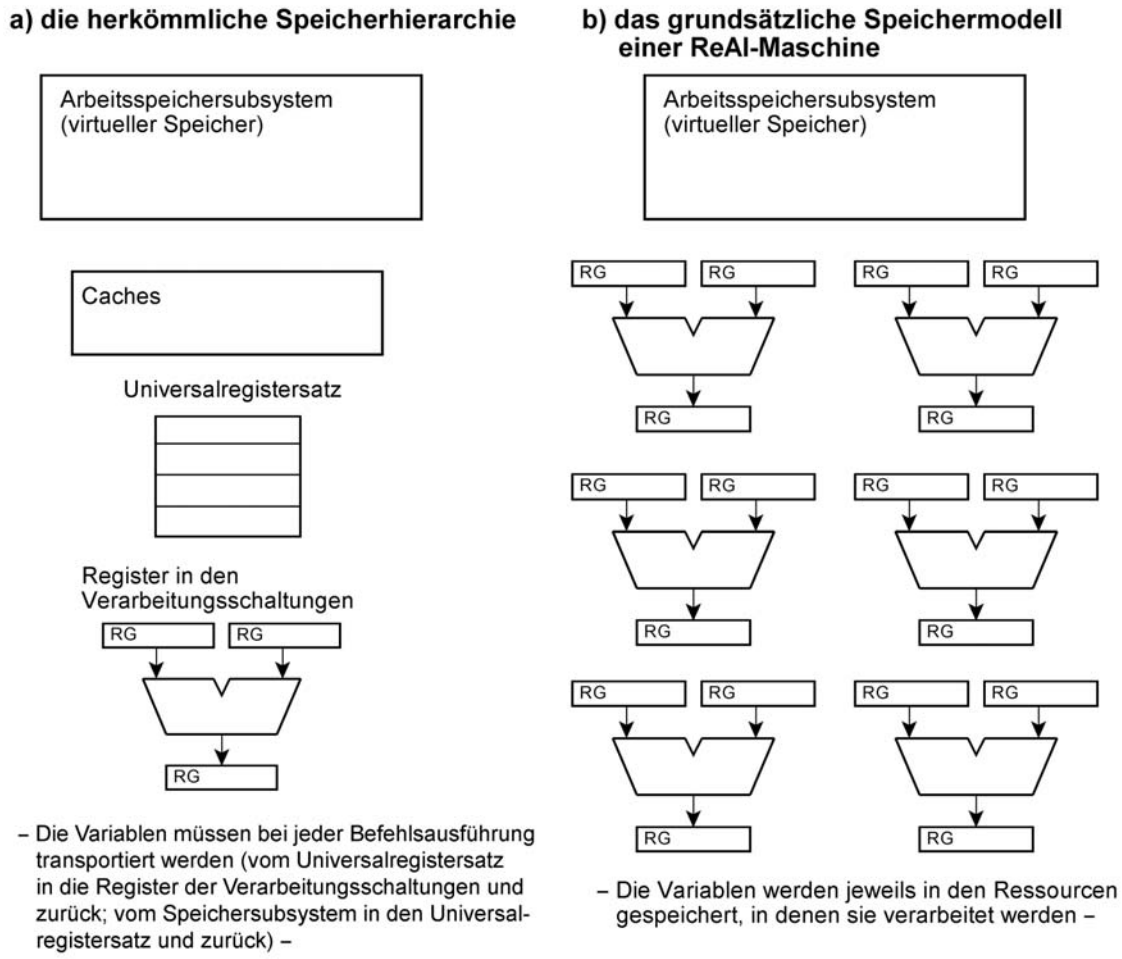

Abb. 2 Speichermodelle im Vergleich. Die herkömmmliche Speicherorganisation führt während des Verabeitungsvorgangs zu vielen Informationstransporten und damit Schaltvorgängen. In einer ReAl-Maschine finden hingegen nur die im Verarbeitungsvorgang unumgänglich notwendigen Informationstransporte statt (Datenflußprinzip).

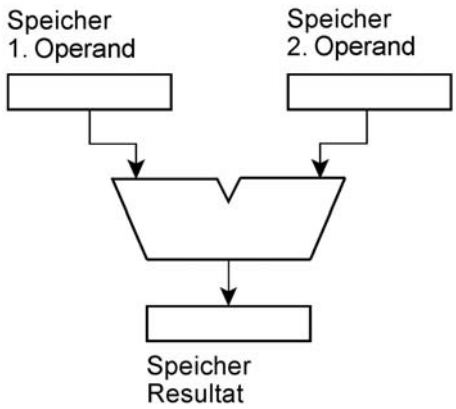

Abb. 3 Ausführung eines Algorithmus durch direkte Zuordnung.

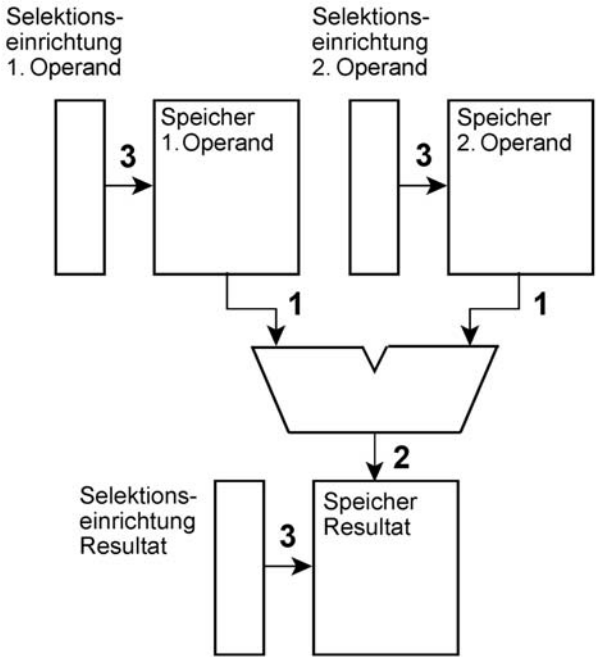

Abb. 4 Abschnittsweise Ausführung eines Algorithmus. 1 - Informationsleitungen für Argumentabschnitte; 2 - desgl. für Resultatabschnitt; 3 - Auswahlleitungen (Speicheradressen).

Diese Eigenschaft des Algorithmus läßt sich durch eine Zahlenangabe ausdrücken, die wir als *Implementierungseffizienz* e_i bezeichnen wollen.

$$e_i = \frac{\sum_{i=1}^{n} CARDB(A_i) + \sum_{j=1}^{m} CARDB(R_j)}{z \cdot (ARG\_LINES + RES\_LINES)}$$

Zur Bedeutung der Symbole:

- CARDB (A_i), CARDB (R_j): Anzahl der Bits, mit denen die Argumente und Resultate repräsentiert (bzw. codiert) werden.
- ARG-LINES, RES-LINES: Anzahl der Signalleitungen, die zum Transportieren der Argument- und Resultatabschnitte zur Verfügung stehen (Stichwort: Verarbeitungsbreite),
- z: Anzahl der Maschinenzustände (Taktzyklen), die benötigt werden, um alle Resultate zu bilden ($z \geq 1$).

Ist eine technisch gegebene Leitungszahl größer als die Anzahl der Bits des betreffenden Abschnittes, ist die Abschnittslänge anstelle der Leitungszahl einzusetzen.

Die Implementierungseffizienz e_i ist somit eine dimensionslose Zahl im Intervall $0 < e_i \leq 1$. e_i ist gleich 1, wenn die beschriebene zweckmäßigste Implementierung tatsächlich möglich ist, d. h., wenn es gelingt, eine Anordnung zu bauen, die in jedem Taktzyklus entsprechend ihrer Verarbeitungsbreite zum gewünschten Endergebnis beiträgt. Dies kann nur dann erreicht werden, wenn für jeden Abschnitt des

Resultats gilt, daß dieser durch Zuordnung (im Sinne von Abb. 3) aus den jeweils ausgewählten Abschnitten der Argumente gebildet werden kann, wobei in diese Zuordnung höchstens noch Zustands-Angaben einbezogen werden, die eindeutig aus den zuvor verarbeiteten Abschnitten bestimmt worden sind (Abb. 5).

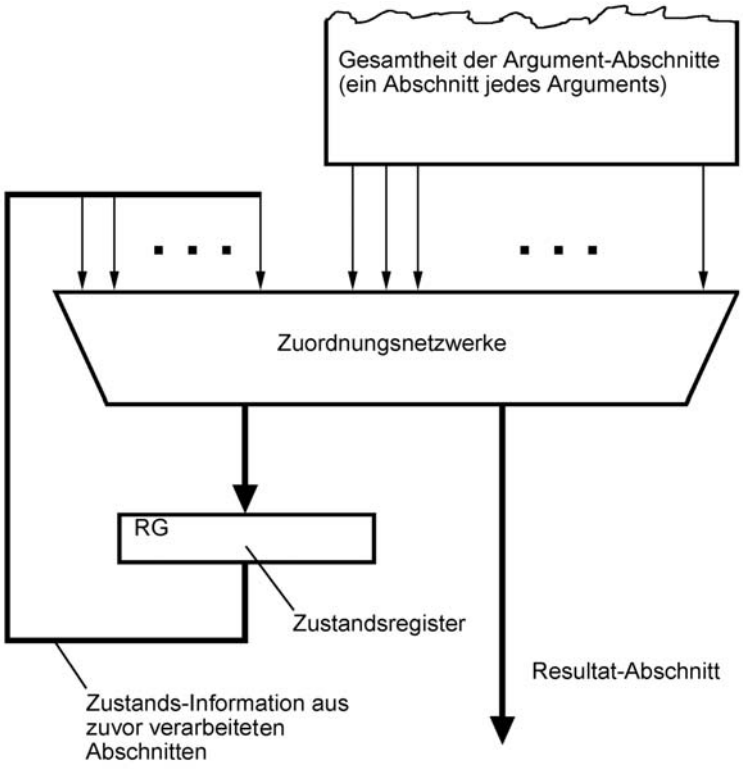

Abb. 5 Allgemeines Schema der Ausführung eines Algorithmus mit $e_i = 1$.

Es ist möglich, diesen Ansatz bis in die Feinheiten hinein zu formalisieren.

Die Implementierungseffizienz ist < 1, wenn

- die abschnittsweise Zuordnung technisch nicht verwirklicht werden kann (Aufwand, Kompliziertheit), so daß Folgen mehrerer Taktzyklen (Maschinenzustände) nötig sind, um jeweils einen Resultatabschnitt zu bilden,
- Resultatbits von Argumentbits abhängen, die sich in verschiedenen Abschnitten befinden können, so daß Zugriffe auf mehrere Argumentabschnitte notwendig sind, um diese Resultatbits zu bestimmen,
- gewisse Argumentbelegungen zur Folge haben, daß Teile bereits gebildeter Resultatabschnitte geändert werden müssen (dies erfordert, erneut auf diese Abschnitte zuzugreifen) bzw. daß man das Ergebnis gar nicht in abschnittsweise liefern kann, sondern daß zunächst Zwischenergebnisse zu bilden sind.

Ein Sachverhalt nach Punkt 1 ist nicht immer ein unüberwindliches Hindernis – es ist letztlich eine Ermessensfrage, was man als zu aufwendig oder zu kompliziert ansieht. Hingegen bezeichnen die Punkte 2 und 3 objektive Grenzen. Solche Algorithmen können nie mit $e_i = 1$ implementiert werden (auch dann

nicht, wenn der Aufwand keine Rolle spielt). Ein plausibles Beispiel ist das Umordnen von Bits in einem größeren Bitfeld (z. B. von Pixeln in einem Bildspeicher) gemäß den Angaben einer Indexliste. Jedes Argumentbit kann hierbei grundsätzlich in jede Resultatposition transportiert werden, so daß es notwendig sein kann, bereits gebildete Resultatabschnitte nochmals aufzurufen, um neue Werte einzufügen.