Lehrstuhl für Netzwerktheorie und Signalverarbeitung
Technische Universität München

# Entwurfsmethoden für verlustarme integrierte Schaltungen

Christian Vinzenz Schimpfle

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs

genehmigten Dissertation.

Vorsitzende: Univ.-Prof. Dr. rer. nat. Doris Schmitt-Landsiedel
Prüfer der Dissertation:
1. Univ.-Prof. Dr. techn. Josef A. Nossek
2. Univ.-Prof. Dr.-Ing. Kurt Antreich

Die Dissertation wurde am 5. Mai 2000 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 28. Juli 2000 angenommen.

# Entwurfsmethoden für verlustarme integrierte Schaltungen

Christian Vinzenz Schimpfle

Lehrstuhl für Netzwerktheorie und Signalverarbeitung

# Danksagung

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Assistent am Lehrstuhl für Netzwerktheorie und Signalverarbeitung der Technischen Universität München.

Meinen herzlichen Dank möchte ich dem Lehrstuhlinhaber, Herrn Prof. Dr. techn. Josef A. Nossek aussprechen, der diese Arbeit ermöglicht, vielfach unterstützt und großzügig gefördert hat. Mein Dank gilt ausserdem dem Leiter der Arbeitsgruppe VLSI-Architekturen und Schaltungsentwurf, Herrn Prof. Dr.-Ing. habil. Walter Entenmann sowie Herrn Prof. Dr.-Ing. Kurt Antreich für die Übernahme des zweiten Berichts.

Bedanken möchte ich mich auch bei allen Kolleginnen und Kollegen am Lehrstuhl sowie bei den Diplomanden und studentischen Hilfskräften für die vielen hilfreichen Diskussionen und Anregungen und für die Schaffung eines freundschaftlichen Arbeitsklimas. Mein besonderer Dank gilt diesbezüglich Herrn Dr. Sven Simon, der diese Arbeit vielfach unterstützt hat, und durch dessen Inspirationen viele neue Erkenntnisse erst möglich wurden. Vielen Dank Herrn Dr. Rainer Pauli für die vielen interessanten Diskussionen und so manchen grundlegenden Hinweis, auch über das Themengebiet dieser Arbeit hinaus. Nicht zuletzt möchte ich mich bei Frau Lidmilla Barth bedanken, die den Großteil der Layouts mit unerschöpflicher Geduld erstellt hat.

Ein besonderes Anliegen ist es mir an dieser Stelle, meiner Familie zu danken, meiner Frau Birgit und meinen Kindern Lukas, Mathias und Luis, die mir in jeder Situation Stütze und Ausgleich sind. Sie haben, jeder auf seine Weise, entscheidend zur Entstehung dieser Arbeit beigetragen.

München, im Juni 2000

# Inhaltsverzeichnis

# 1. Einleitung

Die Forschungsaktivitäten im Bereich hochintegrierter digitaler Schaltungen waren in der Vergangenheit fast ausschließlich auf eine effiziente Nutzung der zur Verfügung stehenden Chip-Fläche, sowie auf die Maximierung der Schaltungsgeschwindigkeit hin ausgerichtet. Die rasche Entwicklung hin zu immer komplexeren Systemen mit hohen Integrationsdichten und hohen Taktraten hat jedoch zu Schaltungen mit immer höheren Verlustleistungen geführt. Beispielsweise beträgt die Leistungsaufnahme von Hochleistungsmikroprozessoren oftmals 20 Watt und mehr, bei Durchschnittsströmen von mehreren Ampere.

Dies bringt in vielerlei Hinsicht Probleme mit sich: zum einen ist die ausreichende Kühlung der Chips schwierig und kostenintensiv. Die Zuleitungen der Spannungsversorgung müssen entsprechend ausgelegt werden, Lüfter und Kühlkörper sind nötig. Zum anderen ergeben sich bei den immer stärker verbreiteten portablen, batteriebetriebenen Geräten Probleme mit der begrenzten Batteriekapazität und dem hohen Gewicht der Batterien pro gespeicherter Energie. Während sich Prognosen zufolge die Integrationsdichte der Chips etwa alle sieben Jahre verzehnfacht, wird sich die Batteriekapazität, bezogen auf das Gewicht der Batterie, in dieser Zeit kaum verdoppeln [90]. Ein weiteres wesentliches Problem ergibt sich aus ökologischer Sicht. Die große Anzahl von Arbeitsplatzrechnern und privaten Computern trägt heute bereits einen erheblichen Teil zum Gesamtverbrauch an elektrischer Energie bei. Eine amerikanische Studie schätzte den Beitrag der Arbeitsplatzrechner zum Gesamtstromverbrauch in den USA 1993 auf 5% und prognostiziert für das Jahr 2000 einen Anteil von 10% [2].

In den letzten Jahren hat sich deshalb der Forschungsschwerpunkt im Bereich hochintegrierter Schaltungen (*VLSI*-Schaltungen, *VLSI*: *V*ery *L*arge *S*cale *I*ntegration) verlagert, mit dem Ziel, die Verlustleistung der Schaltungen zu reduzieren. Dies ist in Anbetracht der oben genannten Probleme unumgänglich geworden. Im wesentlichen ergeben sich daraus zwei Teilaufgaben für die Forschung. Zum einen die Bereitstellung von Methoden zur Ermittlung der Verlustleistung um somit verschiedene Schaltungskonzepte vergleichen zu können und eine Lokalisierung von Quellen erhöhter Verlustleistung zu ermöglichen. Zum anderen die Entwicklung von Methoden zur Reduktion der Verlustleistung. In der einschlägigen Literatur zu diesem Thema finden sich diese beiden Teilgebiete zusammengefasst unter dem englischen Begriff *Low Power Design*.

In dieser Arbeit werden Methoden zur Ermittlung sowie zur Reduzierung der Verlustleistung digitaler Schaltungen vorgestellt und untersucht. Dabei wird für die Demonstrationsbeispiele zu den jeweiligen Verfahren eine CMOS-Technologie zugrunde gelegt. Die Verfahren an sich sind weitgehend unabhängig von der Technologiegeneration. Diese wirkt sich ledig-

lich dort auf das Schaltungsmodell aus, wo die Modellbildung basierend auf der untersten schaltungstechnischen Entwurfsebene, also der Bauteilebene erfolgt.

## 1.1 Die Leistungsaufnahme von digitalen CMOS-Schaltungen

Allgemein kann der Leistungsverbrauch eines Systems in zwei Arten unterteilt werden, wie man es auch in der einschlägigen Literatur zu diesem Thema findet: in *dynamischen* und *statischen* Leistungsverbrauch. Um die Wirkungsweisen verschiedener Verfahren zur Verlustleistungsreduzierung zu verstehen, ist es sinnvoll, zunächst die physikalischen Mechanismen zu untersuchen, die zum Leistungsverbrauch eines Systems führen. Ist im Folgenden von *Leistung* die Rede, so ist immer die elektrische Leistung gemeint, die aus einer Quelle entnommen wird. Wie im weiteren noch zu zeigen ist, wird in CMOS-Schaltungen prinzipiell die gesamte aus der Quelle entnommene Energie in der Schaltung in Wärme umgewandelt. Im Folgenden werden deshalb die Begriffe *Leistungsaufnahme*, *Verlustleistung*, *Leistungsverbrauch* und *dissipierte Leistung* synonym verwendet, da die einmal von der Schaltung aufgenommene Leistung für die Quelle gewissermaßen "verloren" ist und in der Schaltung komplett umgesetzt wird. Andere Schaltungsprinzipien wie sie in *adiabatischen* Schaltungen Anwendung finden, ermöglichen bis zu einem gewissen Grad die Wiederverwendung der in einer Schaltung gespeicherten Energie [3, 51, 70, 71, 49]. Diese Klasse von Schaltungen wird in der hier vorliegenden Arbeit nicht betrachtet.

### 1.1.1 Dynamische Leistungsaufnahme

Als dynamische Leistungsaufnahme einer Schaltung bezeichnet man allgemein die Leistungsaufnahme, die aufgrund eines logischen Umschaltvorganges also einer Zustandsänderung in der Schaltung auftritt. Diese Leistung lässt sich wiederum in zwei Anteile aufspalten: in die Leistung zum Umladen von in der Schaltung befindlichen Kapazitäten sowie in die Leistung, die infolge eines Quer- oder Kurzschlussstromes während des Umschaltvorganges in den Kanalwiderständen der Transistoren in Wärme umgewandelt wird. Dieser Querstrom tritt immer dann auf, wenn ein NMOS-Block und der dazu komplementäre PMOS-Block gleichzeitig leiten und damit einen direkten Stromfluss von der Quelle zur Masse ermögichen.

Die dynamische Leistungsaufnahme einer CMOS-Schaltung soll am Beispiel des CMOS-Inverters in Bild 1.1 veranschaulicht werden. Die Energie, die für $u_\mathrm{e}(t) = 0V$, $0 < t \leq T$ zum Laden der Kapazität $C_\mathrm{L}$ von $0V$ auf die Versorgungsspannung $U_\mathrm{B}$ während der Zeit $T$ der Quelle entnommen wird, berechnet sich zu

$$W_\mathrm{q} = \int_{t=0}^{T} U_\mathrm{B} i_\mathrm{c}(t)\, dt \quad . \tag{1.1}$$

Der Ladestrom der Kapazität ist durch die Gleichung

$$i_\mathrm{c}(t) = C_\mathrm{L} \frac{du_\mathrm{c}(t)}{dt} \tag{1.2}$$

gegeben. Unter der Annahme, dass die Kapazität zum Zeitpunkt $t = 0$ vollständig entladen war und zum Zeitpunkt $t = T$ auf $U_\mathrm{B}$ aufgeladen ist, lässt sich die Formel (1.1) auch schreiben als:

$$W_\mathrm{q} = \int_{0}^{U_\mathrm{B}} C_\mathrm{L} U_\mathrm{B}\, du_\mathrm{c} = C_\mathrm{L} U_\mathrm{B}^2 \quad . \tag{1.3}$$

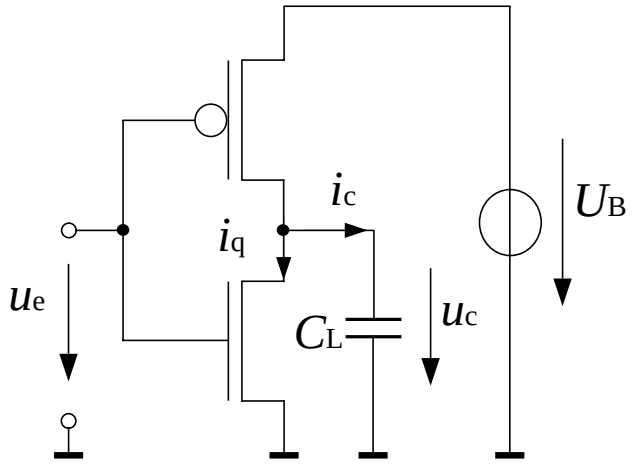



Bild 1.1. CMOS-Inverter mit Lastkapazität $C_\mathrm{L}$.

Die im Kondensator gespeicherte Energie ergibt sich zu:

$$W_\mathrm{c} = \int_{t=0}^{T} C_\mathrm{L} u_\mathrm{c} \frac{du_\mathrm{c}}{dt}\, dt = \int_0^{U_\mathrm{B}} C_\mathrm{L} u_\mathrm{c}\, du_\mathrm{c} = \frac{1}{2} C_\mathrm{L} U_\mathrm{B}^2 \quad , \tag{1.4}$$

d. h. die Hälfte der Energie aus der Quelle wurde im Kanalwiderstand des PMOS-Transistors in Wärme umgewandelt. Beim Entladen des Kondensators nach einer weiteren Periode $T$, also $u_\mathrm{e}(t) = U_\mathrm{B}$ für $T < t \leq 2T$, $u_\mathrm{c}(T) = U_\mathrm{B}$, $u_\mathrm{c}(2T) = 0$, wird die im Kondensator gespeicherte Energie $W_\mathrm{c} = \frac{1}{2} C_\mathrm{L} U_\mathrm{B}^2$ im Kanalwiderstand des NMOS-Transistors in Wärme umgewandelt.

Ändert sich also das Eingangssignal der Schaltung abwechselnd von logisch Null nach logisch Eins und von logisch Eins nach logisch Null mit einer Frequenz $f = \frac{1}{T}$, so wird in der Schaltung die Leistung

$$P_\mathrm{s} = \frac{1}{2} f C_\mathrm{L} U_\mathrm{B}^2 \tag{1.5}$$

dissipiert. Im weiteren wird $P_\mathrm{s}$ als *kapazitive Schaltleistung* bezeichnet. Betrachtet man das Eingangssignal allgemein als einen stochastischen Prozess, so lässt sich ein Maß $\alpha'$ definieren, das angibt, mit welcher Wahrscheinlichkeit das Eingangssignal innerhalb einer Periode $T$ seinen logischen Zustand ändert. Man bezeichnet $\alpha'$ als die *Aktivität* des Signals. Mit der Definition $\alpha = \frac{1}{2}\alpha'$ kann Gleichung (1.5) allgemein als

$$P_\mathrm{s} = \alpha f C_\mathrm{L} U_\mathrm{B}^2 \tag{1.6}$$

geschrieben werden. Eine Besonderheit hierbei ist, dass $\alpha$ auch Werte größer 1 annehmen kann. Dies rührt daher, dass es aufgrund von Laufzeitunterschieden zwischen den Signalen in der Schaltung zu ungewollten Schaltvorgängen, sog. *Glitches*, kommen kann. Bild 1.2 verdeutlicht die Entstehung eines Glitches. Durch die Verzögerungszeit des Inverters erreichen die ursprünglich synchronen Signalwechsel das NAND-Gatter zu unterschiedlichen Zeiten. Kurzzeitig liegt an beiden Eingängen des NAND-Gatters eine "1" wodurch der Glitch am Ausgang hervorgerufen wird. Natürlich muss die Periodendauer $T$ so gewählt werden, dass nach jeder vollen Periode die jeweils richtigen Zustände an den Ausgängen der logischen Gatter anliegen, die Schaltung also genügend Zeit hat, dass die Glitches abklingen können.

Betrachtet man also Gleichung (1.6), so ergeben sich offensichtlich verschiedene Ansatzpunkte zur Minimierung der kapazitiven Schaltleistung. Am effektivsten wirkt sich hierbei

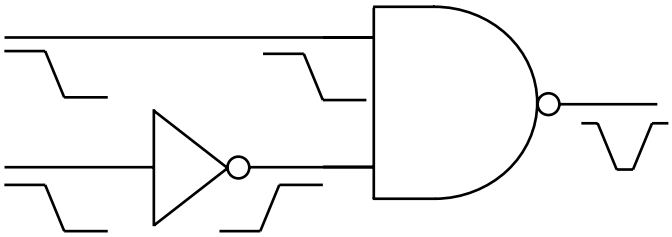


Bild 1.2. Entstehung eines ungewollten Schaltvorgangs (Glitch) aufgrund von unterschiedlich verzögerten Signalen am Eingang eines Gatters

die Reduzierung der Versorgungsspannung $U_\mathrm{B}$ aus, da diese quadratisch zur Leistungsaufnahme beiträgt. Eine Reduzierung der Versorgungsspannung führt jedoch inherent auch zu einer Reduzierung der Schaltgeschwindigkeit der CMOS-Logik. Dies wird in der folgenden Betrachtung für den Fall eines CMOS-Inverters verdeutlicht. Die Zeit zum Laden eines Kondensators $C_\mathrm{L}$ von der Spannung $U_1$ auf $U_2$ berechnet sich allgemein aus

$$t = \int_{U_1}^{U_2} \frac{C_\mathrm{L}}{i_\mathrm{C}(u_\mathrm{C})}\, du_\mathrm{C} \quad . \tag{1.7}$$

Näherungsweise kann der Ladestrom durch den Kondensator am Ausgang eines Inverters als konstant angesehen werden. Zur Vereinfachung kann weiterhin angenommen werden, dass sich der Transistor, über den der Kondensator geladen wird, während des gesamten Ladevorganges in Sättigung befindet. Als Verzögerungszeit eines Gatters definiert man die Zeit, nach der die Ausgangsspannung $\frac{U_\mathrm{B}}{2}$ erreicht hat. Somit berechnet sich die Verzögerungszeit des Inverters mit $U_1 = 0$ und $U_2 = \frac{U_\mathrm{B}}{2}$ zu

$$t = \frac{C_\mathrm{L} U_\mathrm{B}}{\beta (U_\mathrm{B} - U_\mathrm{th})^2} \tag{1.8}$$

wobei $\beta$ eine prozessabhängige Konstante und $U_\mathrm{th}$ die ebenfalls prozessabhängige Schwellenspannung des Transistors bezeichnen. Eine ausführliche Herleitung der Zusammenhänge findet sich in [50]. Hieraus wird ersichtlich, dass mit Verringerung der Differenz zwischen $U_\mathrm{B}$ und $U_\mathrm{th}$ die Verzögerungszeit zunimmt.

Desweiteren erhöht sich die Leistungsaufnahme linear mit der Frequenz und der zu treibenden Kapazität. Geschwindigkeitsnachteile, die durch die Verringerung der Versorgungsspannung oder der Frequenz entstehen, können durch entprechende Parallelisierung kompensiert werden, wobei jedoch eine Vergrößerung des Implementierungsaufwandes akzeptiert werden muss. Die Methoden, die in dieser Arbeit vorgestellt werden, zielen in erster Linie auf die Minimierung der Aktivität $\alpha$ und im weiteren auf die Reduzierung der Gesamtkapazität der Schaltung ab. Diese Methoden können jedoch mit anderen Verfahren zur Minimierung der kapazitiven Schaltleistung kombiniert werden. Im Allgemeinen hängt es immer vom Anwendungsgebiet der Schaltung ab, welche Methoden am besten geeignet sind. Beim Entwurf verlustarmer Schaltungen sollte jedoch nicht nur die Optimierung hinsichtlich eines bestimmten Faktors in Betracht kommen, also z. B. nur die Verringerung von $U_\mathrm{B}$ oder nur die Minimierung von $\alpha$, sondern alle zur Verfügung stehenden Möglichkeiten zur Reduzierung der Verlustleistung ausgereizt werden.

Im Allgemeinen verursacht das Laden und Entladen von Kapazitäten den weitaus größten Leistungsverbrauch in einer digitalen CMOS-Schaltung. Mit neuen Technologien nimmt je-

doch der Anteil des durch den Quer- oder Kurzschlussstrom verursachten Leistungsverbrauches der *dynamischen Kurzschlussleistung* zu. Dies liegt im wesentlichen an der Verringerung der Gesamtkapazität und somit der kapazitiven Schaltleistung durch immer kleinere Abmessungen, wovon die dynamische Kurzschlussleistung in geringerem Maße beeinflusst wird. Der Querstrom hängt von verschiedenen Technologieparametern, von der Temperatur, der Form des Eingangssignals und der Lastkapazität ab. Allgemein ist die dabei verursachte Verlustleistung schwieriger in eine geschlossene Formel zu fassen als dies für die kapazitive Schaltleistung der Fall ist. In [85] ist für einen unbelasteten, symmetrischen Inverter (gleiche Kanalwiderstände von PMOS- und NMOS-Transistor) eine Formel für die dynamische Kurzschlussleistung angegeben:

$$P_{\mathrm{k}} = \frac{\beta}{12}\tau f(U_{\mathrm{B}} - 2U_{\mathrm{th}})^3 \quad . \tag{1.9}$$

Dabei ist $\beta$ der Steilheitsfaktor von NMOS- bzw. PMOS-Transistor ($\beta_{\mathrm{n}} = \beta_{\mathrm{p}} = \beta$), $\tau$ die Anstiegszeit des Eingangssignals und $U_{\mathrm{th}}$ die Schwellenspannung ($U_{\mathrm{th,n}} = -U_{\mathrm{th,p}} = U_{\mathrm{th}}$). Formel (1.9) gibt die Zusammenhänge in erster Näherung wieder, genauere Werte lassen sich jedoch durch Schaltungssimulation mit SPICE ermitteln. Qualitativ können folgende Aussagen gemacht werden [85]:
Die dynamische Kurzschlussleistung ist umso kleiner

- je größer die Ausgangskapazität der Schaltung und
- je steiler die Flanke des Eingangssignals ist.

Die Diagramme in Bild 1.3 zeigen den Kurzschlussstrom und den Gesamtstrom in einem CMOS-Inverter bei unterschiedlicher Belastung des Ausgangs.

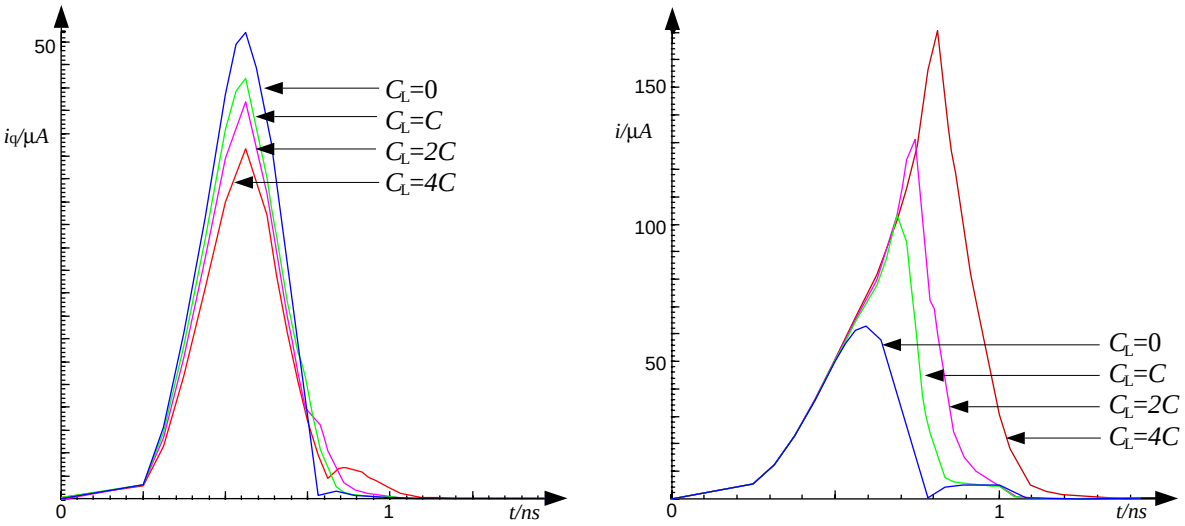



Bild 1.3. Kurzschlussstrom $i_{\mathrm{q}}$ und Gesamtstrom $i$ in einem CMOS-Inverter für unterschiedliche Belastung des Ausgangs.

### 1.1.2 Statische Leistungsaufnahme

Schaltungstechnisch bedingt fließt in einer CMOS-Logik kein statischer Strom, d.h. durch die Realisierung der Schaltung mit NMOS-Transistoren und dazu komplementären PMOS-Transistoren existiert kein permanenter Gleichstrompfad von der Quelle zur Masse. Statische

Verlustleistung in CMOS-Schaltungen resultiert lediglich aus Leckströmen. Diese sind zum einen die Sperrströme parasitärer Dioden, zum anderen aber auch Ströme, die bedingt durch den endlichen Kanalwiderstand der gesperrten Transistoren auftreten. Der Diodensperrstrom ist eine Funktion der Sperrspannung $U$ und der Temperaturspannung $U_\mathrm{T}$, wobei $U \leq 0$:

$$I_\mathrm{sperr} = I_\mathrm{S}(\mathrm{e}^{\frac{U}{U_\mathrm{T}}} - 1) \quad . \tag{1.10}$$

Der Sättigungssperrstrom $I_\mathrm{S}$ hängt wesentlich von der Temperatur und der Sperrschichtfläche ab, ist also vom Entwickler nur bedingt zu beeinflussen. In heutigen Schaltungen liegt der Sättigungssperrstrom in der Größenordnung um $1pA$ pro Transistor, ist also in jedem Fall im Vergleich zu den dynamischen Strömen vernachlässigbar.

Der durch den endlichen Kanalwiderstand gesperrter Transistoren ($U_\mathrm{ds} - U_\mathrm{th} < 0$) bedingte Leckstrom $I_\mathrm{sub}$ hängt ebenfalls wie der Diodensperrstrom vom Fertigungsprozess und der Temperatur ab. Desweiteren besteht auch eine Abhängigkeit von der Gate-Source-Spannung $U_\mathrm{gs}$ wobei

$$I_\mathrm{sub} \sim \mathrm{e}^{(U_\mathrm{gs} - U_\mathrm{th})} \quad . \tag{1.11}$$

Ebenso wie der Diodensperrstrom ist auch $I_\mathrm{sub}$ vernachlässigbar und aufgrund seiner starken Prozessabhängigkeit ohnehin vom Entwickler im Allgemeinen nicht beeinflussbar. Mit immer kleiner werdenden minimalen Transistorabmessungen zukünftiger Technologien und den damit immer kleineren Schwellenspannungen wird jedoch der Einfluss dieser statischen Leckströme auf den Gesamtleistungsverbrauch zunehmen und kann sich so zu einem nicht mehr vernachlässigbaren Anteil entwickeln.

## 1.2 Leistungsbetrachtungen auf verschiedenen Entwurfsebenen

Die rege Forschungstätigkeit im Bereich verlustleistungsarmer Schaltungen hat in den letzten Jahren eine Vielzahl von neuen allgemeinen Methoden zum Schaltungsentwurf hervorgebracht. Viele Publikationen präsentieren auch spezielle schaltungstechnische Realisierungen für bestimmte, arithmetische Funktionen [12, 16, 24, 74, 75]. Letztlich haben alle diese Methoden die Verringerung der Versorgungsspannung, der Taktfrequenz, der Kapazität, der Schaltaktivität oder einer Kombination aus diesen genannten Faktoren zum Ziel. Dabei ist die Einhaltung der vorgegebenen Anforderungen an die Schaltung (z.B. die minimale Datendurchsatzrate) eine Voraussetzung.

Allgemein ist man zu der Auffassung gelangt, dass die Anwendung von Methoden zur Verringerung der Verlustleistung auf allen Ebenen des Entwurfprozesses sinnvoll ist, um die optimale Schaltung zu erhalten. Bei den Entwurfsebenen unterscheidet man:

- die Systemebene als oberste Abstraktionsebene,
- die Algorithmenebene mit den algorithmischen Beschreibungen der einzelnen Funktionsblöcke,
- die Architekturebene, darin u. a. die Festlegung des Datenformates und des Taktschemas
- die Logikebene
- die Schaltkreisebene, hierzu gehören der Schaltplan mit den einzelnen Bauelementen (engl.: *schematic*) und schließlich das Maskenlayout.

Welche Methoden wann und wo sinnvoll eingesetzt werden können, darüber entscheidet auch der Entwurfsstil. Je nach Schaltungsstruktur erweist sich u. U. ein bestimmter Entwurfsstil als besonders vorteilhaft. Bei den Entwurfsstilen unterscheidet man zwischen

manuellem, vollkundenspezifischem Entwurf (*Full Custom Design*) und der automatisierten Synthese basierend auf einer Beschreibungshochsprache wie VHDL (*V*ery High Speed Integrated Circuit *H*ardware *D*escription *L*anguage). Die Entscheidung für einen bestimmten Entwurfsstil kann im Allgemeinen auf höherer Abstraktionsebene, typischerweise auf Architekturebene, gefällt werden, da sich hier bereits erkennen lässt, welche Schaltungsstruktur ein bestimmter Algorithmus erfordert.

Ein hoher Optimierungsgrad bezüglich der Chip-Fläche und der Geschwindigkeit kann bei regulären Strukturen, typischerweise bei Datenpfaden für die digitale Signalverarbeitung, durch vollständig manuellen Entwurf erreicht werden. Die Strukturinformation kann beim Entwurf von Hand optimal genutzt, das Layout mit maximaler Kompaktheit und minimalen Leitungslängen realisiert werden. Diese Klasse der regulären Schaltungen ermöglicht die Realisierung durch Kopieren und Aneinanderfügen von einfachen Grundzellen zu komplexeren Blöcken, welche wiederum zu größeren Blöcken zusammengefasst werden usw. bis schließlich das Gesamtlayout entsteht. Es existiert also ein hoher Grad an Modularität. Der Entwickler hat dabei Einfluss auf sämtliche Ebenen des Entwurfs, von der Systemspezifikation bis zum Layout der Grundzellen. Es steht hier die gesamte Palette der Methoden zur Verlustleistungsreduzierung zur Verfügung. Ein Teil dieser Arbeit beschäftigt sich damit, wie die Regularität und Modularität solcher Schaltungen dazu genutzt werden kann, die Verlustleistungsabschätzung einfach und schnell zu gestalten.

Im Gegensatz zum manuellen Entwurf hat der Entwickler beim automatisierten Entwurf mit einem Software-Werkzeug keinen oder nur eingeschränkten Einfluss auf die Struktur des Layouts. Die Eingabe der Schaltungsfunktion geschieht in Form einer Verhaltensbeschreibung oder in einer strukturellen, netzlistenähnlichen Form. Damit lassen sich Schaltungen ganz allgemeiner Art realisieren, ohne irgendwelche bevorzugten Struktureigenschaften. Besonders ist diese Art des Entwurfes jedoch für irreguläre Schaltungen wie z.B. Steuerlogiken geeigntet, wo durch die Komplexität des Problems auf niedrigeren Abstraktionsebenen kaum mehr ein Überblick gewonnen werden kann und ein manueller Eingriff deshalb nur schwer möglich ist. Zumeist bedient sich der Entwickler hier einer Standardzellenbibliothek mit vom Technologiehersteller garantierten Eigenschaften der einzelnen Zellen. Eine Optimierung der Schaltung hinsichtlich minimaler Verlustleistung ist also bei der rechnergestützten Schaltungssynthese in der Regel nur auf höherer Abstraktionsebene möglich. Im Sinne von kürzeren Entwicklungszeiten und damit geringeren Kosten ist es zudem wünschenswert, die Methoden zur Verlustleistungsreduzierung auf höheren und niedrigeren Entwurfsebenen weitgehend voneinander zu entkoppeln, sodass ihre Anwendung völlig unabhängig voneinander und ohne gegenseitige Beeinflussung geschehen kann. Dies bedeutet konkret im Fall des automatisierten Entwurfes, dass zunächst die Bereitstellung einer Standardzellenbibliothek mit auf unterster Schaltungsebene hinsichtlich minimaler Verlustleistung optimierten Zellen erfolgt. Der Entwickler dieser Bibliothek spezifiziert die Zellen lediglich durch Worst-Case Parameter, deren Einhaltung für die vorgesehenen Betriebssituationen garantiert ist. Der Entwurf einer komplexeren Schaltung aus diesen Zellen erfolgt i. a. von einem anderen Entwickler auf höherer Abstraktionsebene nur unter Kenntnis der spezifischen Parameter. Die hierfür vorgesehenen Methoden zur Verlustleistungsreduzierung sind im Idealfall unabhängig von der schaltungstechnischen Realisierung der Grundzellen anwendbar. D. h. auch eine erneute Synthese der Grundzellen ist nicht erforderlich wenn die Schaltung auf höherer Ebene verändert wird. Die optimale Lösung hinsichtlich minimaler Verlustleistung einer Schaltung ergibt sich also idealerweise genau dann, wenn in jeder einzelnen Entwurfsebene das jeweilige

Optimum erreicht wurde. Dieser Idealfall ist in der Praxis nicht generell garantiert, da die Anwendbarkeit der Methoden sehr wohl an bestimmte Voraussetzungen, die in anderen Entwurfsebenen festgelegt werden, gebunden sein können. Beispielsweise sind die Möglichkeiten zur Variationen der Versorgungsspannung von der verwendeten Prozesstechnologie abhängig.

Neben den unterschiedlichen Methoden bei der Synthese verlustleistungsarmer Schaltungen unterscheiden sich auch die Methoden der Verlustleistungsbestimmung in den verschiedenen Entwurfsebenen. Allgemein ist eine umso genauere Verlustleistungsbestimmung möglich, je niedriger der Grad der Abstraktion ist.

Auf System- und algorithmischer Ebene liegt weder Information über die eigentliche Realisierung der Schaltung noch über deren Architektur vor. Die Verlustleistung kann ist hier beispielsweise durch eine gewichtete Summe der nötigen Rechenoperationen abgeschätzt werden. Durch die Abschätzung der Differenz des Realisierungsaufwandes für den einen oder anderen Algorithmus kann in erster Näherung auf die Gesamtkapazität und die Verlustleistung geschlossen werden. Da auf höherer Entwurfsebene größere Leistungseinsparungen möglich sind, kann hier eine geringere absolute Genauigkeit der Leistungsabschätzung toleriert werden. Wichtig ist eine genaue relative Abschätzung, die einen Vergleich und damit die Wahl des günstigsten Algorithmus' ermöglicht.

Eine bereits weitaus genauere Verlustleistungsabschätzung ist auf der Architekturebene möglich. Hier ist die Datenwortlänge bereits festgelegt, der Realisierungsaufwand und somit die Gesamtkapazität der Schaltung kann einfacher ermittelt und für verschiedene Architekturen verglichen werden. Mit der Festlegung der Zahlendarstellung der Daten kann auch bereits eine Abschätzung der Schaltaktivitäten erfolgen. Beispielsweise ist die Aktivität in den Vorzeichenbits bei Zweierkomplementdarstellung abhängig von der zeitlichen Korrelation der Signale, d. h. von der Wahrscheinlichkeit, mit der ein negatives auf ein positives Signal folgt oder umgekehrt.

Für die Verlustleistungsbestimmung auf Logikebene stehen kommerzielle Logiksimulatoren zur Verfügung, sowie auch wahrscheinlichkeitsbasierte Ansätze zur analytischen Berechnung der Schaltaktivitäten. Mit der Information über die genaue Anzahl der Gatter, der Komplexität ihrer Funktion und der Schaltaktivitäten an ihren Ein- und Ausgängen, kann die Gesamtleistung durch Bilden einer mit den Knotenkapazitäten gewichteten Summe der Schaltaktivitäten abgeschätzt werden.

Die letztlich genaueste und am allgemeinsten anwendbare aber zugleich aufwendigste Verlustleistungsanalyse geschieht durch Schaltungssimulation auf Schaltkreis bzw. Layoutebene. Ein typischer kommerzieller Vertreter von Schaltungssimulatoren ist SPICE. Hierbei können Effekte wie Leckströme, Temperaturschwankungen oder parasitäre reaktive und resistive Elemente berücksichtigt werden. Für große Schaltungen erweist sich diese Vorgehensweise aufgrund des hohen Rechen- und Speicheraufwandes als nicht mehr praktikabel. Zudem ist für jede simulationsbasierte Analyse einer Schaltung bezüglich ihres zeitlichen Verhaltens und ihres Leistungsverbrauchs ein geeigneter Stimulus nötig, d. h. eine ausreichend große Anzahl von Testvektoren, um realistische Aussagen treffen zu können. Schaltungssimulationen eignen sich jedoch sehr gut zur genauen Spezifikation des funktionalen, zeitlichen und thermischen Verhaltens kleinerer Zellen und Blöcke und natürlich zur genauen Ermittlung von deren Verlustleistung.

In Tabelle 1.1 sind die verschiedenen Entwurfsebenen und Ansätze zur Bestimmung und Minimierung der Verlustleistung schematisch dargestellt.

| Entwurfsebene | Bestimmung der Verlustleistung | Minimierung der Verlustleistung |
|---|---|---|
| System- und Algorithmusebene | Abschätzung der Rechenoperationen | Abschaltung von Systemeinheiten, geeignete Einteilung in Systemblöcke, Reduzierung der Algorithmenkomplexität |
| Architekturebene | Abschätzung der Schaltaktivitäten | geeignete Wahl der Zahlendarstellung, Hardware-Sharing, Folding, Pipelining, Retiming |
| Logik-, Schaltkreis-Layoutebene | Logiksimulation, Schaltungssimulation | Logiktransformationen, Transistoroptimierung, Low Power Zellbibliotheken |

Tabelle 1.1. Beispiele für Ansätze zur Bestimmung und Reduzierung der Verlustleistung digitaler Schaltungen auf den verschiedenen Ebenen des Entwurfsprozesses.

In der vorliegenden Arbeit werden sowohl Methoden zur schnellen Bestimmung der Verlustleistung auf höherer Abstraktionsebene als auch Methoden zur Minimierung der Verlustleistung vorgestellt. Bei den Methoden zur Bestimmung der Verlustleistung handelt es sich zum einen um datenabhängige, also simulationsbasierte, zum anderen um ein datenunabhängiges Verfahren, bei dem der Leistungsverbrauch unter Verwendung spezieller Modelle analytisch bestimmt wird. Dieses analytische Verfahren ist besonders für Schaltungen geeignet, die mittels Retiming oder Pipelining hinsichtlich der Verlustleistung optimiert werden sollen.

Wie erwähnt, wird neben diesen Verfahren für spezielle Schaltungsstrukturen in einem weiteren Kapitel eine allgemein anwendbare Methode zur Verlustleistungsbestimmung auf höherer Abstraktionsebene vorgestellt. Diese Methode basiert auf der Bestimmung der für den Leistungsverbrauch relevanten Parameter auf unterster Layoutebene. In einem Modellierungsschritt werden diese Parameter dann in Zellenbeschreibungen eingebunden, die wiederum auf höherer Abstraktionsebene in Logiksimulationen komplexer Schaltungen als Modelle Verwendung finden. Mit dieser zweigeteilten Strategie, der layoutbasierten Modellierung von einfachen Grundzellen und der konventionellen Logiksimulation der Gesamtschaltung, gelingt es, schnelle und genaue Aussagen über den Leistungsverbrauch zu machen, ohne die eigentliche Schaltung auf Layoutebene zu realisieren. Für die Entwicklung eines solchen Verfahrens bedarf es also der Erforschung von Wechselwirkungen zwischen einzelnen Einflussfaktoren in verschiedenen Abstraktionsebenen.

Als Verfahren zur Verlustleistungsreduzierung auf Architekturebene werden die bereits erwähnten Pipelining- und Retimingmethoden diskutiert. Eine Methode die auf der Schaltkreisebene ansetzt ist die optimale Transistordimensionierung zur Reduzierung der Schalthäufigkeit. Beispielhaft für ein Verfahren auf algorithmischer Ebene wird der Entwurf einer verlustleistungsarmen Variante einer diskreten Kosinustransformation gezeigt. Hierbei findet auch eine Optimierung auf Architekturebene mittels Pipelining statt.

# 2. Prinzipielle Strategien zur Reduzierung der Verlustleistung

Beim Entwurf integrierter Schaltungen gibt es in jeder Abstraktionsebene verschiedene Ansätze zur Bestimmung und Verringerung der Verlustleistung. Unter Berücksichtigung zusätzlicher Nebenbedingungen wie z. B. bestimmter Anforderungen an die Datendurchsatzrate oder den Flächenbedarf einer Schaltung, erweisen sich jedoch bestimmte Methoden als besonders geeignet oder andere als nicht praktikabel. Zum Beispiel ist bei hohen Anforderungen an die Datendurchsatzrate und gleichzeitig geforderter minimaler Fläche die Verringerung der Versorgungspannung ein ungeeignetes Mittel zur Verlustleistungsreduzierung, da aufgrund der begrenzten Fläche eine Geschwindigkeitskompensation mittels Parallelisierung oder Übergang zu redundanten Zahlensystemen nicht möglich ist. Allgemein gilt, dass potentiell umso mehr Leistung eingespart werden kann, je höher die Abstraktionsebene ist, auf der entsprechende Überlegungen zur Minimierung der Leistung angestellt werden. Diese Tatsache erlaubt andererseits auch ein höheres Maß an Ungenauigkeit bei den Methoden zur Verlustleistungsbestimmung auf höheren Abstraktionsebenen. Solche Methoden sind vor allem daraufhin ausgerichtet, eine schnelle, qualitative Aussage über den Leistungsverbrauch einer Schaltung machen zu können. Dabei ist eine genauere quantitative Aussage nicht unbedingt nötig und oftmals auch nicht wünschenswert, da dies zumeist einen höheren Aufwand und damit eine längere Entwicklungszeit zur Folge hat. Es genügt oftmals verschiedene alternative Systeme, Algorithmen oder Architekturen miteinander vergleichen zu können. Die Genauigkeit der Verfahren zur Verlustleistungsabschätzung muss aber dennoch in einem Bereich liegen, in dem eine verlässliche Auswahl der für eine bestimmte Anwendung günstigsten Schaltung immer noch möglich ist. Im Folgenden werden einige prinzipielle Methoden zur Reduzierung der Verlustleistung auf verschiedenen Abstraktionsebenen vorgestellt und an Beispielen verdeutlicht.

## 2.1 Verfahren auf höheren Abstraktionsebenen

Auf höheren Abstraktionsebenen liegen die Schaltungen in Beschreibungsformen vor, bei denen noch keine feste Schaltungstopologie mit physikalischen Leitungsverbindungen vorgeschrieben ist. Der allgemeine Trend hin zu Schaltungsbeschreibungen auf höheren Abstraktionsebenen und zur High-Level-Synthese wird durch verschiedene Faktoren bestärkt. Erstens wird die Handhabung immer komplexerer Systeme dadurch erleichtert oder überhaupt erst möglich und zweitens werden kürzere Entwicklungszeiten erreicht. Ausserdem ist der Schaltungsentwurf mittels automatisierter Synthese weniger fehlerträchtig als der reine

Handentwurf. Diese Entwicklung erfordert auch geeignete Strategien zur Reduzierung der Verlustleistung auf höherer Abstraktionsebene.

### 2.1.1 Systemebene

Ein wichtiges Entwurfsziel auf Systemebene ist die optimale Verteilung und Steuerung der Spannungsversorgung für die verschiedenen Komponenten eines Systems. Eine Möglichkeit zur Verlustleistungsreduzierung besteht darin, die Betriebsspannung in bestimmten Teilen des Systems abzusenken. Damit lässt sich die kapazitive Schaltleistung umgekehrt proportional zum Quadrat der Spannung verringern, wie aus Gleichung (1.5) hervorgeht. Durch die verminderte Betriebsspannung erhöhen sich abergleichzeitig die Signallaufzeiten in den entsprechenden Teilsystemen. Diese Methode kann daher nur in zeitlich unkritischen Teilen des Systems angewendet werden. Wichtig ist in diesem Zusammenhang auch eine geeignete Partitionierung des Systems, d. h. dass Schaltungsteile, die aufgrund gegebener Geschwindigkeitsanforderungen mit einer bestimmten Spannung betrieben werden müssen, zu einem Teilsystem zusammengefasst werden. Bild 2.1 zeigt ein Beispiel für die Unterteilung eines Systems in Teilsysteme, in denen zeitkritische Aufgaben wie Berechnungen im Prozessor, Datentransfer über den Bus und Speicherzugriffe ausgeführt werden, sowie weniger zeitkritische Teilsysteme, die für Aufgaben wie Ein- und Ausgabe (Bildschirm oder Tastatur) zuständig sind. Aufgrund des Mehraufwandes für die einzelnen Spannungsquellen und die

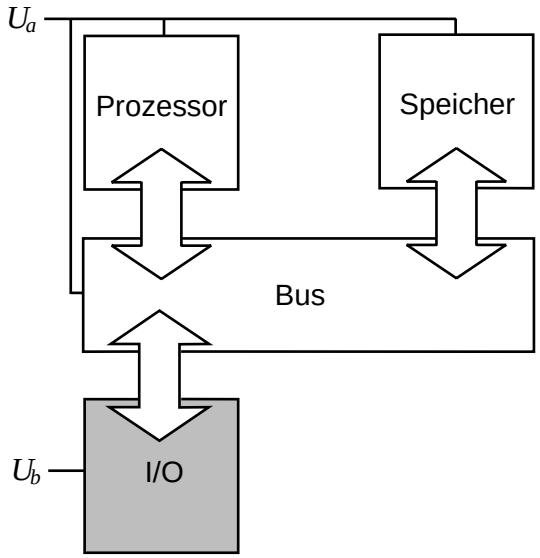



Bild 2.1. Beispiel für ein System mit mehreren Betriebsspannungen. Das nicht zeitkritische Ein-/Ausgabesystem wird mit niedrigerer Spannung betrieben, $U_b < U_a$.

entsprechenden Zuführungsleitungen zu den Teilsystemen, ist die Anzahl der unterschiedlichen Betriebsspannungen natürlich begrenzt. In [36] wird eine Methode zum Entwurf von verlustleistungsarmen Datenpfden mit verschiedenen Betriebsspannungen beschrieben.

Teilsysteme, die in bestimmten Betriebsphasen keine Funktion ausüben, können während dieser Zeit komplett abgeschaltet werden. Eine zusätzliche Kontrolleinheit muss in diesem Fall die Ruhephasen eines Teilsystemes erkennen und die Spannungsversorgung für dieses Teilsystem ab- und anschalten. Eine weitere Möglichkeit für die Optimierung des Leistungsverbrauches auf Systemebene besteht in der Einführung von *Standby-* oder *Sleep-Modes*.

Üblicherweise wird dies durch Reduzierung der Taktfrequenz oder komplettes Abschalten der Taktversorgung, dem sog. *Clock Gating*, in momentan nicht benötigten Systemteilen erreicht. Dadurch wird die effektive Gesamtkapazität und damit der Leistungsverbrauch des Systems verringert. Die Aufgabe einer entsprechenden Kontrolleinheit besteht darin, anhand logischer Zustände die Bedingungen für das Abschalten der Taktversorgung zu bestimmen. Bei der Modellierung von Teilsystemen mittels *Finite-State-Machines* entspricht dies dem Auffinden von Selbstschleifen im Zustands-Übergangs-Graphen, wie dies in Bild 2.2 dargestellt ist. Die

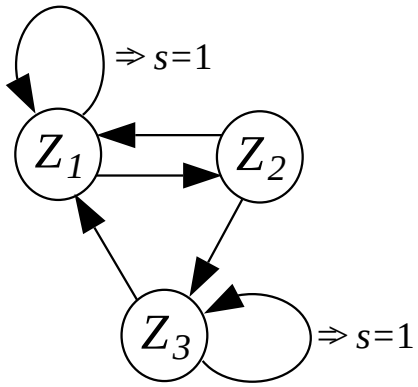



Bild 2.2. Zustands-Übergangs-Graph mit Selbstschleifen. Die Detektion einer Selbstschleife bewirkt das Abschalten des Taktes durch das Steuersignal $s = 1$, sonst $s = 0$. Die Zustände sind mit $Z_i$ bezeichnet. Die Symbole $x_i$ an den Kanten bezeichnen bestimmte Eingangswerte.

Detektion einer solchen Selbstschleife bedingt das Abschalten der Taktversorgung für den entsprechenden Systemteil. In Bild 2.3 ist das Clock-Gating Prinzip dargestellt. Es muss

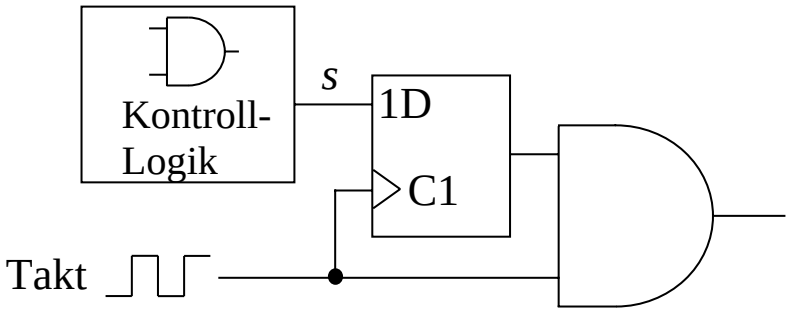



Bild 2.3. Clock-Gating.

sichergestellt werden, dass auf der Taktleitung durch das Clock-Gating keine Glitches, also ungewollte Schaltvorgänge, entstehen, die zu Fehlfunktionen der Schaltung führen würden. Die Kontrolleinheit kann jedoch die Bedingungen für Ab- oder Anschalten des Taktes in einer Taktperiode möglicherweise erst bestimmen, nachdem der entsprechende Takt bereits erfolgt ist. Als Beispiel signalisiere logisch "1" des Ausgangssignals $s$ der Kontrolleinheit das Einschalten des Taktes und logisch "0" entsprechend das Abschalten des Taktes. Das Signal $s$ ist um die Laufzeit der Steuerlogik verzögert und unter Umständen mit Glitches behaftet. Eine einfache UND-Verknüpfung von Taktsignal und Ausgangssignal der Steuerung erzeugt somit kein Taktsignal, das frei von Glitches ist, da eine logische "1" des ursprünglichen Taktes immer vor dem letztendlich gültigen Wert des Steuersignals $s$ am UND-Gatter

anliegt und deshalb alle Glitches das Gatter passieren können. Das Beispiel in Bild 2.3 ist eine Variante, bei der Glitches durch ein flankengetriggertes D-Flip-Flop eliminiert werden, da das Ausgangssignal der Steuereinheit nur im Augenblick einer ansteigenden Taktflanke übernommen wird.

Die für das Clock-Gating erforderliche Logik verbraucht zusätzliche Leistung, was je nach Realisierungsaufwand die Effektivität dieses Verfahrens erheblich reduzieren kann. In [6] wird die Kontroll-Logik dadurch vereinfacht, dass nur diejenigen Bedingungen für das Abschalten des Taktes betrachtet werden, die mit einer bestimmten Wahrscheinlichkeit auftreten. Die Leistungseinsparung, die mit solchen Verfahren erreicht werden kann, liegt je nach Schaltung im Bereich von 10 bis 60 Prozent.

### 2.1.2 Algorithmische Ebene

Die Algorithmen, die in den einzelnen Teilen eines Gesamtsystems ausgeführt werden sollen, können in rein mathematischer Form, also als geschlossene Formel, als Struktogramm oder in Form eines Programmcodes beschrieben werden. Grundsätzlich gibt es zwei Stufen der Leistungsoptimierung auf algorithmischer Ebene:

- die optimale Wahl eines Algorithmus' zur Erfüllung einer bestimmten Aufgabe und
- die Optimierung des gewählten Algorithmus' hinsichtlich des minimalen Leistungsverbrauchs der resultierenden Schaltung.

Beispielsweise kann für eine Signaltransformation zur Datenkommpression im ersten Schritt zwischen verschiedenen Transformationsarten, wie Fourier-, Kosinus- oder Wavelettransformation gewählt werden. Inwieweit es möglich ist, die Algorithmen selbst hinsichtlich minimaler Verlustleistung zu optimieren, hängt wiederum von der Art der Transformation ab.

Bei einer Leistungsoptimierung auf algorithmischer Ebene werden üblicherweise zwei Ziele angestrebt:

- Übergang auf eine niedrigere Versorgungsspannung. Die Wahl eines schnelleren Algorithmus' ermöglicht die Reduzierung der Versorgungsspannung und somit der kapazitiven Schaltleistung. Hier besteht ein direkter Zusammenhang zur Optimierung auf Systemebene mittels mehrerer Versorgungsspannungen, wie dies im Abschnitt 2.1.1 beschrieben wurde.
- Eine möglichst niedrige Anzahl von Schaltvorgängen.

Beide Ziele können sowohl bei der Auswahl sowie auch bei der Optimierung des Algorithmus' berücksichtigt werden.

Eine wichtige Rolle für den Leistungsverbrauch spielt die Algorithmenkomplexität. Die Reduzierung der Komplexität geht einher mit einer geringeren Anzahl an auszuführenden Operationen und kann so zu Leistungseinsparungen führen. Beispielsweise können komplexe Operationen wie Multiplikationen durch mehrere einfache Schiebe- und Addierschritte ersetzt werden. Mit der Anwendung des Distributivgesetzes gelingt es, die Anzahl der Rechenoperation zu verringern, wie dies in Bild 2.4 veranschaulicht wird. Bietet ein Algorithmus die Möglichkeit, mehrere Operationen parallel auszuführen, so kann dies zur Steigerung der Verarbeitungsgeschwindigkeit genutzt werden. Dadurch kann auch ein Geschwindigkeitsverlust kompensiert werden, der durch eventuelles Absenken der Versorgungsspannung entsteht. Reguläre Algorithmen haben den Vorteil, dass sie im Allgemeinen weniger Kontrollaufwand benötigen und deshalb mit geringerem Hardware-Überschuss für die Steuerung auskommen.

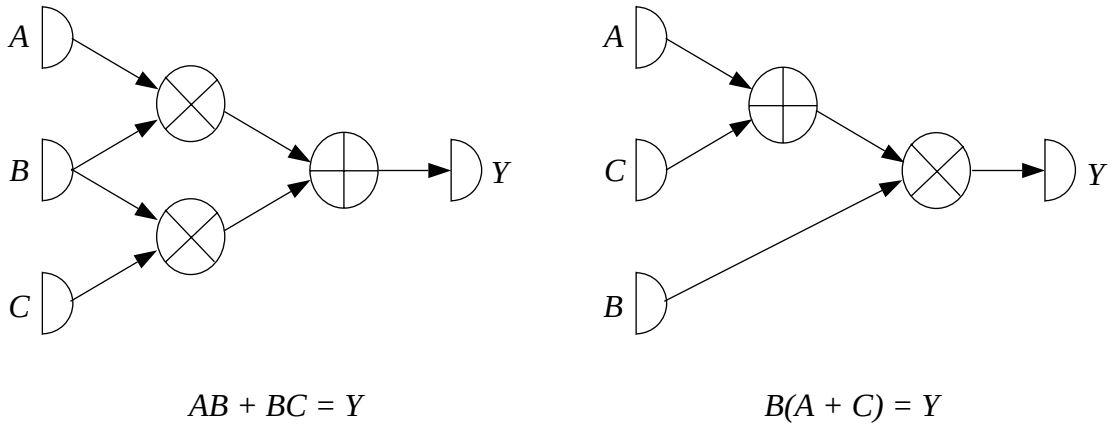



Bild 2.4. Reduzierung von Rechenoperationen.

In der Literatur finden sich Methoden zur Beurteilung von Algorithmen in [35, 73]. Eine Reihe algorithmischer Transformationen zur Erhöhung der Geschwindigkeit, Reduzierung der Operationen und der Gesamtkapazität beschreiben Chandrakasan et al. in [15]. In [52] schließlich wird ein Überblick über verschiedene Methoden zur High-Level-Synthese verlustleistungsarmer Schaltungen mittels Optimierungen auf algorithmischer Ebene gegeben.

### 2.1.3 Architekturebene

Die gebräuchlichste Form der Schaltungsbeschreibung auf höherer Abstraktionsebene ist die *Register-Transfer-Ebenen-Beschreibung* oder RTL-Beschreibung (von engl. *R*egister-*T*ransfer-*L*evel) mittels einer Hardware-Beschreibungssprache wie Verilog oder VHDL. Die Register-Transfer-Ebene wird allgemein der Architekturebene zugeordnet. Der Leistungsverbrauch einer Schaltung kann hier beispielsweise durch die Wahl der Zahlendarstellung, Retiming, Pipelining oder durch die Wahl der Architektur arithmetischer Einheiten wie Multiplizierer und Addierer beeinflusst werden.

Bei der Wahl der Zahlendarstellung spielen zweierlei Betrachtungen eine Rolle: erstens, die Geschwindigkeit, mit der arithmetische Operationen ausgeführt werden können und zweitens, der Realisierungsaufwand für diese Operationen. Diese beiden Faktoren sind letztlich mit entscheidend für die Gesamtkapazität einer Schaltung und die Möglichkeit zur Absenkung der Versorgungsspannung.

Die Geschwindigkeit einer Schaltung zur Addition wird wesentlich von den Laufzeiten der Überträge (Carry-Laufzeiten) bestimmt. Die weit verbreitete Carry-Ripple-Addition bei Zahlendarstellung im Zweierkomplement bietet den Vorteil der einfachen Implementierbarkeit, jedoch hängt die Carry-Laufzeit $t_c$ direkt proportional von der Datenwortlänge $m$ ab. Es gilt $t_c = O(m)$. Redundante Zahlendarstellungen wie Carry-Save-Darstellung [47] oder Signed-Digit-Number-Representation (SDNR) [4] erlauben Additionen mit $t_c = O(1)$, benötigen dabei aber einen mindestens doppelt so hohen Implementierungsaufwand. Als Alternative dazu bietet sich das Pipelining von Carry-Ripple-Addierern an, wodurch $t_c = O(1)$ erreicht werden kann. Dabei ist jedoch zusätzlicher Aufwand für Register und Taktversorgung nötig. Vergleiche von verschiedenen Multiplizierer- und Addiererarchitekturen unter gleichen Bedingungen ($U_B$, Wortlänge, Eingangsdatenrate) hinsichtlich Verzögerungszeit, Leistungsverbrauch und Fläche finden sich in [12] und [13]. In [5] wird der Leistungsverbrauch von Array-Multiplizierern durch datenabhängiges Konfigurieren der Hardware gesenkt. Dabei

| vor Retiming<br>$U_B = 5V$, $t_{c1} = 1.5ns$ | nach Retiming<br>$U_B = 1.8V$, $t_{c2} = 1.5ns$ |
|---|---|
| $109\mu W$ | $13.2\mu W$ |

Tabelle 2.1. Verlustleistung eines 8-Bit Carry-Ripple Addierers vor und nach Reduzierung der Versorgungsspannung und Geschwindigkeitskompensation durch Retiming.

werden die mit "0" zu multiplizierenden Zeilen eines Multipliziererfeldes überbrückt und tragen deshalb nicht zum dynamischen Leistungsverbrauch der Schaltung bei.

Retiming und Pipelining werden in verschiedenen Arbeiten [14, 45, 37] als Methoden zur Verlustleistungsreduzierung vorgestellt. Dabei wird zum einen der kritische Pfad einer Schaltung durch Registerumverteilung verkürzt, wodurch es möglich ist, die Schaltung schneller zu takten. Andererseits kann dadurch auch ein Geschwindigkeitsverlust, der durch Absenken der Betriebsspannung verursacht wurde, kompensiert werden. Bild 2.5 zeigt zwei Versionen eines Carry-Ripple-Addierers, wobei im ersten Fall die Register nur an den Eingängen der Schaltung liegen und im zweiten Fall durch Retiming eine vollsystolische Variante der Schaltung erreicht wurde. Für die Verzögerungszeiten $t_{c1}$ im ersten Fall und $t_{c2}$ nach dem Retiming

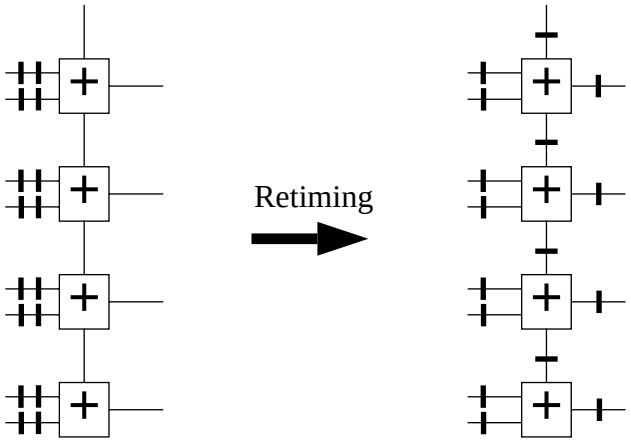



Bild 2.5. Carry-Ripple Addierer in kombinatorischer (links) und vollsystolischer Ausführung nach Retiming. Die Register sind durch schwarze Rechtecke symbolisiert.

gilt also: $t_{c1} = O(m)$, $t_{c2} = O(1)$. In einem Experiment wurde die Betriebsspannung eines 8-Bit Carry-Ripple Addierers nach dem Retiming soweit reduziert, dass $t_{c1} = t_{c2}$. Bei dem verwendeten $0.7\mu m$-Prozess erlaubt dies eine Absenkung der Spannung von $5V$ auf $1.8V$, wobei dann $t_{c1} = t_{c2} = 1.5ns$. Die Ergebnisse hinsichtlich der Verlustleistung sind in Tabelle 2.1.3 aufgeführt. Desweiteren können Register als *Glitch-Barrieren* eingesetzt werden. Mittels Retiming gelingt es, diese Register an Stellen besonders hoher Glitch-Aktivität zu platzieren und dadurch die Gesamtschaltaktivität und den Leistungsverbrauch zu reduzieren. Ein solches Verfahren wird in Kapitel 4 dieser Arbeit vorgestellt.

Ein weiteres Verfahren, das ebenfalls auf der Architekturebene ansetzt und auf einer Steuerung zur optimalen Hardware-Nutzung bei arithmetischen Operationen in redundanten Zahlensystemen basiert, wurde in [66] und [63] vorgestellt. Die Idee hierbei ist, Teile der Schaltung, die zeitweise inaktiv sind, vollständig von der Spannungversorgung abzuschalten. In den genannten Arbeiten wird diese Technik bei der Addition von Zahlen in SDNR-Zahlendarstellung angewendet. Besonders effektiv ist dieses Verfahren in CORDIC-

Prozessoren [20] einsetzbar, da sich hier bei SDNR-Zahlendarstellung in jeder der nacheinander auszuführenden CORDIC-Operationen die effektiv genutzte Hardware ändert. Durch eine geeignete Steuerung ist es möglich, die jeweils ungenutzten Hardware-Teile abzuschalten und somit Leistung zu sparen.

## 2.2 Verfahren auf Technologie- und schaltungstechnischer Ebene

Die Logik- und Schaltungsebene sind dem Entwickler nur bei vollkundenspezifischem Entwurf direkt zugänglich. Die Entwicklung von Low Power Zellbibliotheken für den Standardzellenentwurf findet ebenfalls auf diesen unteren Abstraktionsebenen statt. Im Allgemeinen hat der Schaltungsentwickler keinen Einfluss auf prozessbedingte physikalische Eigenschaften einer integrierten Schaltung. Methoden zur Optimierung der Prozessparameter hinsichtlich minimaler Verlustleistung finden ihre Anwendung seitens der Technologiehersteller. Jedoch eröffnen spezielle Technologien auch dem Entwickler neue schaltungstechnische Möglichkeiten wie dies z. B. bei der BiCMOS-Technologie deutlich wird.

### 2.2.1 Logik- und Schaltungsebene

Die Realisierung einer Architektur als Logikschaltung, die so genannte *Logiksynthese*, findet entweder, ausgehend von einer Hochsprachenbeschreibung auf RTL-Ebene, mittels Software-Werkzeugen automatisiert, oder in Form einer manuellen Schaltungseingabe (*schematic entry*) auf Gatterebene statt. Bei der automatischen Logiksynthese hängen die Möglichkeiten zur Verlustleistungsreduzierung neben der Programmiertechnik des Entwicklers auch von der Entwurfs-Software ab. Die Herausforderung liegt hier in der Entwicklung von Synthesewerkzeugen, die gezielt verlustleistungsoptimierte Logikschaltungen generieren können.

Das grundlegende Problem beim Logikentwurf ist die Abbildung von logischen Funktionen auf eine Anzahl gegebener Gatter. Dieses *Technology Mapping* [42] beinhaltet Freiheitsgrade für die Partitionierung der Schaltung, also wie Funktionen zu Gattern zusammengefasst, und welche Gattertypen zur Realisierung einer Funktion verwendet werden. Eine mögliche Strategie zur Minimierung der Verlustleistung auf dieser Ebene ist, Knoten mit hoher Schalthäufigkeit in das Innere von Gattern zu verlegen, wo sie geringere kapazitive Lasten zu treiben haben. Betrachtet man die Gatter als Knoten in einem Netzwerkgraphen, so entspricht dieses Verfahren der Minimierung von Knoten hoher Schaltaktivität. Dadurch wird die durchschnittlich pro Zeiteinheit umzuladende Kapazität verringert. Ein weiteres Ziel bei der Abbildung von Logikfunktionen auf Gatterschaltungen ist die Minimierung der Fläche, um die Gesamtkapazität der Schaltung möglichst klein zu halten.

Neben Fläche und Kapazität kann auch die Schaltaktivität innerhalb einer Schaltung beim Logikentwurf beeinflusst werden. Eine Methode zur Minimierung der Schaltaktivität ist die Wahl einer Baumstruktur bei der Zerlegung einer komplexen logischen Funktion in einfachere Einzelgatter, wie dies in Bild 2.6 für eine Schaltung mit vier Eingängen dargestellt ist. Es sei angenommen, dass alle drei Gatter vom gleichen Typ sind und für die Schaltaktivität am Ausgang $y$ eines einzelnen Gatters mit zwei Eingängen $x_1$ und $x_2$ aufgrund seiner logischen Funktion gelte: $\alpha(y) = \alpha(x_1)\alpha(x_2)$. Dabei werden Signale mit höherer Schaltaktivität erst mit zunehmender logischer Tiefe hinzugefügt. Mit den Schaltaktivitäten $\alpha(s)$ der Eingangssignale $s$, $s = 1 \ldots n$, berechnet sich die gesamte Schaltaktivität $\alpha_{\text{ges}}$ bei

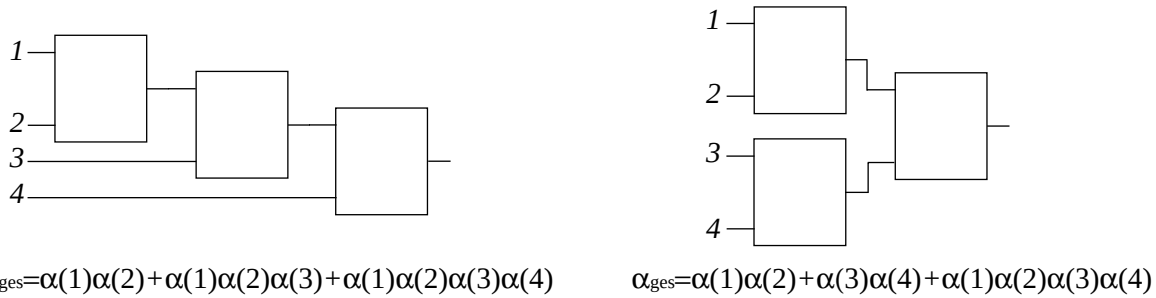



Bild 2.6. Logikzerlegung in eine Baumstruktur und eine balancierte Struktur. Unter der Annahme, dass für die Schaltaktivitäten der Eingänge $\alpha(1) < \alpha(2) < \alpha(3) < \alpha(4)$ mit $\alpha(i) \leq 1 \; \forall \, i$ gilt und die Gatter keine Verzögerungszeit aufweisen, ist die Schaltaktivität $\alpha_{\text{ges}}$ in der linken Schaltung geringer.

Verschaltung der Gatter in Baumstruktur gemäß Bild 2.6 zu

$$\alpha_{\text{ges}} = \sum_{m=2}^{n-1} \prod_{s=1}^{m} \alpha(s) \quad . \tag{2.1}$$

Für gegebene, feste Schaltaktivitäten $\alpha(s) \leq 1$ ist $\alpha_{\text{ges}}$ genau dann minimal, wenn $\alpha(s) \leq \alpha(s+1)$ für $s = 1 \ldots n-1$.

Gleichung (2.1) und die daraus abgeleitete Folgerung für minimale Schaltaktivität stimmen uneigeschränkt jedoch nur unter der Annahme, dass alle Eingangssignalwechsel gleichzeitig auftreten und die Gatter keine Verzögerungszeiten aufweisen. Unter reellen Bedingungen können durch Laufzeitunterschiede und die dadurch verursachten Glitches erheblich höhere Schalthäufigkeiten zwischen den Gattern auftreten, als dies durch die Berechnung der Wahrscheinlichkeiten ohne Berücksichtigung der Verzögerungszeiten ermittelt wurde. Eine *balancierte* Struktur, wie in Bild 2.6 rechts, bei der alle Signalpfade gleiche Länge haben, kann in diesem Fall effektiver sein.

Das Balancieren der Pfade als Methode zur Verringerung ungewollter Schaltaktivität kann auf unterschiedliche Weise erfolgen. Zum einen durch die Wahl einer geeigneten Logikstruktur, wie in diesem Abschnitt bereits angemerkt wurde. Zum anderen besteht die Möglichkeit des Einfügens zusätzlicher Verzögerungselemente [57] in nichtkritische Pfade, sodass die Verzögerungszeit der Gesamtschaltung unbeeinflusst bleibt. In Kapitel 5 der vorliegenden Arbeit, wird ein Verfahren vorgestellt, bei dem unterschiedliche Pfadlängen durch gezielte Variation der Transistorabmessungen in den einzelnen Gattern ausgeglichen werden. Schließlich kann auch Pipelining oder Retiming auf Logikebene angewandt werden, um durch das Platzieren von Registern Pfade unterschiedlicher Länge zu synchronisieren oder Glitch-Barrieren zu setzen.

### 2.2.2 Technologiebezogene Ansätze

Die einer integrierten Schaltung zugrundeliegende Prozesstechnologie mit ihrer charakteristischen Kenngröße (*Feature Size*) hat entscheidenden Einfluss auf die Verlustleistung der Schaltung. Immer kleinere Prozessabmessungen resultieren in Schaltungen mit geringerer Gesamtkapazität und folglich geringerer Verlustleistung. Wie aus Gleichung (1.8) ersichtlich ist, reduziert sich die Schaltgeschwingigkeit eines CMOS-Gatters mit kleiner werdender Differenz zwischen der Versorgungsspannung und der Schwellenspannung. Heutige Technologien mit Prozesskenngrößen weit unterhalb $1\mu m$ setzen jedoch aufgrund der geringeren

Spannungsfestigkeit niedrigere Versorgungsspannungen voraus. Der naheliegenste Weg um einem dabei entstehenden Geschwinigkeitsnachteil entgegenzuwirken, ist die Verringerung der Schwellenspannung auf Technologieebene. Dies führt allerdings unmittelbar zu einer Erhöhung der Leckströme gesperrter Transistoren und damit zur Erhöhung der statischen Verlustleistung.

Prozesse, bei denen durch die Wahl der Dotierung mehrere Schwellenspannungen auf einem Chip zur Verfügung stehen, bieten dem Entwickler die Möglichkeit, zeitkritische Schaltungsteile mit niedrigerer Schwellenspannung zu realisieren und gleichzeitig für die weniger zeitkritischen Schaltungsteile eine höhere Schwellenspannung vorzusehen, um die statische Verlustleistung zu reduzieren. Die Bereitstellung mehrerer Schwellenspannungen erfordert zusätzlichen Aufwand beim Herstellungsprozess, was die Anzahl der möglichen Schwellenspannungen aus Kostengründen einschränkt. In [87] findet sich eine detaillierte Beschreibung für die Verlustleistungsoptimierung von Schaltungen unter Verwendung von zwei verschiedenen Schwellenspannungen.

# 3. Abschätzung der Schaltaktivität in Datenpfadarchitekturen

Datenpfade bilden die Kernstücke in digitalen Signalverarbeitungssystemen. Der Begriff *digitaler Datenpfad* steht für eine Schaltungseinheit in der ein digitalisiertes Signal, beispielsweise ein Audio- oder Videosignal, auf bestimmte Weise manipuliert wird. Typische Datenpfadstrukturen finden sich in digitalen Filtern. Die häufigsten Grundfunktionen die hierbei ausgeführt werden sind Additionen, Multiplikationen, Schiebeoperationen und zeitliche Verzögerungen. Bei diesen Operationen ergibt sich eine Vielzahl unterschiedlich langer Signalpfade, wodurch in den realisierenden Schaltungen in hohem Maße ungewollte Schaltvorgänge auftreten können. Bild 3.1 zeigt einen *Array-* oder *Brown-Multiplizierer*, wie er zur Multiplikation bei vorzeichenfreier Zahlendarstellung verwendet wird. Die Balken neben den Addierern

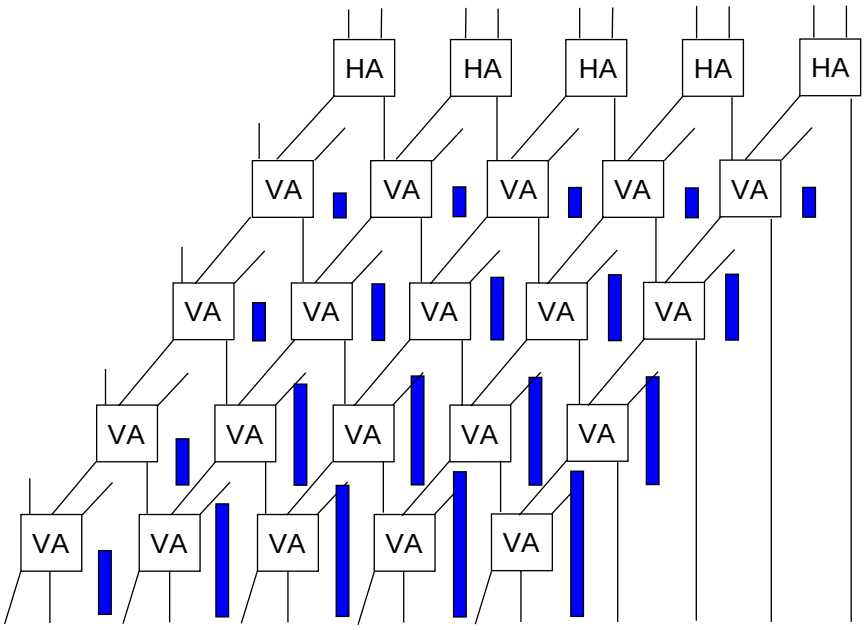



Bild 3.1. Schaltaktivität aufgrund von Glitches in einem Multiplizierer (symbolisiert durch die Balken). VA und HA in den Kästchen stehen für Volladdierer und Halbaddierer.

symbolisieren den Anteil an ungewollter Schaltaktivität an den Ausgängen des jeweiligen Addierers. Es wird deutlich, dass die ungewollte Schaltaktivität aufgrund von Glitches mit der logischen Tiefe der Schaltung zunimmt. Methoden um diese Glitches zu reduzieren, tragen deshalb in diesen Strukturen besonders effektiv zur Verringerung der Verlustleistung bei. Um solche Methoden sinnvoll anwenden zu können, ist die Lokalisierung und die Bestimmung

des Maßes an ungewollter Schaltaktivität aufgrund von Glitches (*Glitch-Aktivität*) nötig. In diesem Kapitel werden zwei Verfahren zur Abschätzung der Glitch-Aktivität vorgestellt, die für allgemeine Schaltungsstrukturen anwendbar sind. Dabei handelt es sich im einen Fall um ein simulationsbasieres, also datenabhängiges Verfahren, im zweiten Fall um ein datenunabhängiges Verfahren, das auf der Analyse der Booleschen Gleichungen einer Schaltung beruht.

## 3.1 Das Schaltungsmodell

Die folgenden Betrachtungen zur Schaltungsmodellierung erfolgen auf der Logikebene. Die Definitionen anhand des Signalflussgraphen sind jedoch unabhängig von der Abstraktionsebene. Eine Logikschaltung kann als ein Signalflussgraph bestehend aus Knoten und gerichteten Verbindungskanten abstrahiert werden. Auf Logikebene entsprechen die Knoten den Gattern, bzw. der Zusammenfassung mehrerer Gatter zu einem Knoten. Die Verbindungskanten entsprechen den physikalischen Leitungsverbindungen der Logikschaltung. Die Verwendung gerichteter Kanten ist hier ohne Einschränkung der Allgemeinheit möglich, da auf Logikebene Ein- und Ausgänge eindeutig definiert sind und sich daraus eine bestimmte Datenflussrichtung ergibt.

Für die Anwendung von Methoden zur Reduzierung der Glitch-Aktivität ist es sinnvoll, ein *Glitch-Gewicht* $g(v)$ für jeden Ausgang eines Knotens $v$ zu definieren. Dieses lässt sich anhand der Information über die Glitch-Aktivität an jedem Knotenausgang und der jeweiligen Ausgangslast bestimmen und bildet die zu minimierende Zielgröße bei der Optimierung der Schaltung.

Das Glitch-Gewicht ist ein Maß für die durch Glitches zusätzlich verbrauchte Leistung in einem Knoten. Eine genauere Definition erfolgt in Abschnitt 3.4 dieses Kapitels. Bild 3.2 zeigt ein Beispiel für einen Signalflussgraphen.

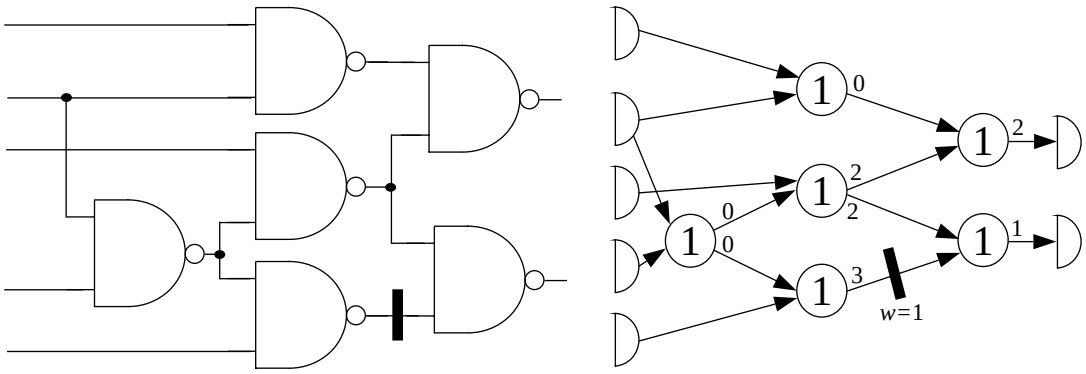



Bild 3.2. Logikschaltung und zugehöriger Signalflussgraph. Die Zahlen an den Ausgängen der Knoten im Signalflussgraphen symbolisieren das jeweilige Glitch-Gewicht, die Zahlen in den Knoten die durchschnittliche Verzögerungszeit.

Knoten mit mehreren Ausgängen unterschiedlichen Glitch-Gewichtes können dazu in mehrere Knoten mit jeweils einem Ausgang aufgespalten werden. In Bild 3.3 ist diese Methode für das Beispiel eines Volladdierers mit Summen- und Carry-Ausgang verdeutlicht. Summen- und Carrypfad des Addierers werden dabei als getrennte Knoten betrachtet.

Die Genauigkeit, mit der die Glitch-Aktivitäten in einer Logikschaltung bestimmt werden können, hängt vom Verzögerungszeitmodell ab. Nur ein Verzögerungszeitmodell, das es

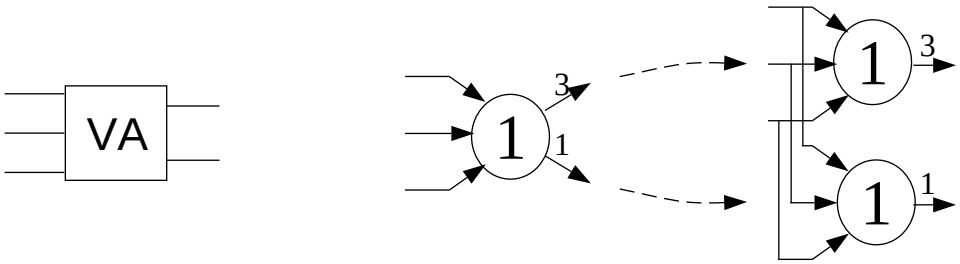



Bild 3.3. Aufspaltung eines Knotens mit mehreren Ausgängen in Knoten mit jeweils einem Ausgang am Beispiel eines Volladdierers mit der Verzögerungszeit 1 und unterschiedlichen Glitch-Gewichten für beide Ausgänge.

erlaubt, Laufzeitunterschiede in der Schaltung zu erfassen, ist für die Bestimmung ungewollter Schaltaktivitäten geeignet. Beim so genannten *Zero-Delay-Modell* wird für die Gatter eine Verzögerungszeit gleich Null angenommen, wodurch die durch Laufzeitunterschiede verursachte Glitch-Aktivität völlig unbeachtet bleibt. Ein solches Modell ist sinnvoll, wenn die Schaltaktivitäten verschiedener Logikstrukturen, die alle die gleiche Funktion erfüllen, verglichen werden sollen. In diesem Fall reduziert sich durch das einfache Verzögerungszeitmodell der Aufwand für statistische Berechnungen bzw. der Simulationsaufwand erheblich. Für die Ermittlung der laufzeitbedingten Schaltaktivität aufgrund von Glitches ist das Zero-Delay-Modell jedoch ungeeignet.

Ein *Worst-Case-Verzögerungszeitmodell* findet häufige Anwendung bei der Bestimmung der minimalen Taktperiode, mit der eine Schaltung betrieben werden kann. Hierbei wird die oberste Grenze für die Verzögerungszeit einer Schaltung betrachtet, um eine Fehlfunktion durch zu schnell aufeinander folgender Daten auszuschließen. Die Verzögerungszeit ist jedoch immer von den aktuellen Daten abhängig, d. h. der Worst-Case tritt nur bei bestimmten Daten tatsächlich auf. Für die Berücksichtigung von Laufzeitunterschieden bei der Ermittlung der Schaltaktivität ist es daher sinnvoller, von der durchschnittlichen Verzögerungszeit der Gatter auszugehen. Dieser Durchschnittswert ergibt sich als Mittelwert einer Verteilung der Verzögerungszeiten eines Gatters bei für die jeweilige Anwendung repräsentativen Eingangsdaten. Eine Vereinfachung der Verzögerungszeitmodelle lässt sich dadurch erreichen, dass für die Verzögerungszeiten der Gatter nur ganzzahlige Vielfache einer Einheitsverzögerungszeit zugelassen werden. Ein Sonderfall hiervon ist das *Unit-Delay-Model*, bei dem die Verzögerungszeiten aller Gatter als identisch angesehen werden.

Komplexere Verzögerungszeitmodelle berücksichtigen unterschiedliche Anstiegs- und Abfallszeiten eines Gatters wobei für die Ausgangswechsel $1 \rightarrow 0$ und $0 \rightarrow 1$ jeweils eine separate Verzögerungszeit verwendet wird. In der realen Schaltung ist die Verzögerungszeit der Gatter nicht allein durch die Zustandswechsel an deren Ausgängen charakterisiert, sondern ist vielmehr abhängig von den Zustandswechseln an den Eingängen. Eine Verfeinerung des Modells durch die Berücksichtigung aller möglichen Wechsel der Zustände an den Eingängen und der dafür jeweils charakteristischen Verzögerungszeiten der Gatter, erlaubt eine detaillierte Analyse des Zeitverhaltens der Schaltung und eine genauere Bestimmung der Anzahl ungewollter Schaltvorgänge an jedem Knoten. Die hohe Komplexität des Modells wirkt sich jedoch nachteilig auf die zur Schaltungsanalyse benötigte Rechenzeit aus.

## 3.2 Simulationsbasierte Bestimmung der Glitch-Aktivität

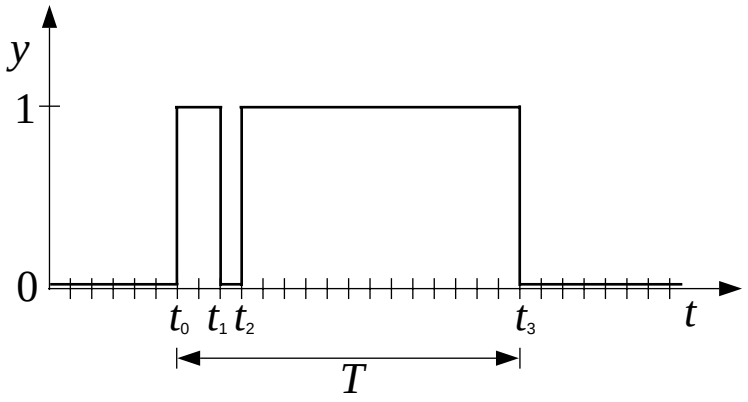
In diesem Abschnitt werden Verfahren zur Bestimmung der Glitch-Anzahl an den Ausgängen der Knoten des Signalflussgraphen behandelt, die auf Simulationen beruhen, d. h. mit einem Satz von Testdaten das Verhalten der Schaltung hinsichtlich ungewollter Schaltaktivität untersuchen. Ein Problem bei einer simulationsbasierten Analyse der Schaltung ist, eine hinreichend große Anzahl von geeigneten Testdaten in akzeptabler Zeit zu verarbeiten. Schaltungssimulatoren wie SPICE werten bei der Transientenanalyse im Zeitbereich die Spannungs- und Stromwerte in einer Schaltung zu jedem Zeitschritt aus. Für die Bestimmung der Schaltaktivität sind jedoch nur die diskreten Spannungswerte von Interesse, die den logischen Zuständen "1" oder "0" entsprechen. Konventionelle Simulatoren, die eine solche Diskretisierung vornehmen, sind Switch-Level- und Logik-Simulatoren. Im Falle der Switch-Level-Simulation sind die Grundelemente der Schaltung die Transistoren, die lediglich als Schalter mit Widerstand modelliert werden. Die Grundelemente der Logiksimulation sind die einzelnen Gatter. Im Folgenden wird zunächst diese Standardmethode zur Bestimmung der Schaltaktivität auf Logikebene behandelt. Weiterhin wird eine spezielle simulationsbasierte Methode zur Glitch-Bestimmung anhand der Booleschen Gleichungen der Schaltung präsentiert.

### 3.2.1 Bestimmung der Glitch-Aktivität durch Logiksimulation

Die Logiksimulation erlaubt eine wesentlich schnellere Bestimmung der Schaltaktivität als eine Schaltungssimulation auf Bauteilebene mit physikalischen Transistormodellen. Die Genauigkeit einer Logiksimulation hängt dabei wesentlich von der Modellierung der Grundelemente, also der Gatter ab. Für die Bestimmung der Anzahl von Glitches ist speziell das Verzögerungszeitmodell von Bedeutung. Von dessen Komplexität hängt jedoch auch die Simulationszeit ab. Allgemein bedeutet höhere Genauigkeit auch längere Simulationszeit. Bei der Schaltungssimulation erfolgt die genauere Analyse einer Schaltung im Zeitbereich in Zeitschritten mit einer einstellbaren minimalen Schrittweite. Für die Ermittlung der Schalthäufigkeit an einem Knoten ist es ausreichend, nur die Zustandsänderungen der Signale zu beobachten, also nicht die Zeitspannen, in der ein Signal konstant ober- oder unterhalb einer definierten Schaltschwelle liegt. Bei dieser ereignisgesteuerten (engl. *event-driven*) Simulation werden die Ereignisse, also die Zustandsänderungen in einer Ereignisliste vermerkt und in ihrer zeitlichen Reihenfolge abgearbeitet. Bild 3.4 zeigt ein Signal $y$ und die zugehörige Ereignisliste. Es sei angenommen, dass die Daten synchron mit einer Taktperiode $T$ verarbeitet werden. In diesem Beispiel tritt während einer Taktperiode für $t_0 \leq t < t_0 + T$ mehr als ein Ereignis auf. Die Taktperiode Nummer $i$ kann als die Zeitspanne $t_i \leq t < t_i + T$ definiert werden. Aus der Anzahl der Ereignisse $e_i$ in Taktperiode Nummer $i$ lässt sich die durchschnittliche Anzahl $a(y)$ der Glitches pro Taktperiode im Signal $y$ für eine Simulation über $N$ Taktperioden bestimmen.

$$a(y) = \frac{1}{N} \sum_{i=1}^{N} \lfloor \frac{e_i}{2} \rfloor \tag{3.1}$$

Ein Glitch ruft immer zwei Ereignisse hervor, nämlich die Signalwechsel $0 \rightarrow 1$ und $1 \rightarrow 0$ oder umgekehrt. Im Fall von Bild 3.4 wird also genau ein Glitch detektiert.




| Zeit | Ereignis |
|---|---|
| $t_0$ | '1' |
| $t_1$ | '0' |
| $t_2$ | '1' |
| $t_3$ | '0' |
| ⋮ | ⋮ |

Bild 3.4. Glitch-Behaftetes Signal und zugehörige Ereignisliste.

### 3.2.2 Bestimmung der Glitch-Aktivität durch Simulation von Booleschen Funktionen

Die Bestimmung der tatsächlich möglichen minimalen Taktperiode, mit der eine Schaltung betrieben werden kann, erfordert eine genauere Analyse des Zeitverhaltens als dies durch einfaches Auffinden des längsten Pfades mit einem der in Abschnitt 3.1 vorgestellten Verzögerungszeitmodelle der Fall ist. Der nach Aufaddieren aller Knotenverzögerungen längste Pfad einer Schaltung, wird u. U. aufgrund der logischen Funktion nie von einem Signal durchlaufen, d. h. ist nicht sensibilisierbar. Die Aufgabe ist es, alle nicht sensibilisierbare Pfade einer Schaltung, die sog. *falschen Pfade*, zu analysieren und so den längsten tatsächlich sensibilisierbaren Pfad zu finden. Die Analyse der falschen Pfade wird in [7, 18, 31, 32] genauer behandelt. In [31, 32] werden dazu zeitbehaftete Boolesche Funktionen verwendet. Diese können im weiteren zur Abschätzung überflüssiger Schaltaktivität verwendet werden.

#### 3.2.2.1 Zeitbehaftete Boolesche Funktionen

Das im Folgenden verwendete Schaltungsmodell beruht auf einem Verzögerungszeitmodell mit diskreten, ganzzahligen Werten für die Verzögerungszeiten der Gatter und gleicher Anstiegs- und Abfallszeit der Signale. Diese Vereinfachung dient lediglich zur übersichtlicheren Erläuterung und ist keinesfalls Voraussetzung. Zeitbehaftete Boolesche Funktionen beschreiben die logischen Verknüpfungen Boolescher Variablen, die zusätzlich mit einem Zeitindex versehen sind. Als Boolesche Variablen können dabei alle Signale einer Schaltung verwendet werden. Der Zeitindex einer Variablen gibt an, nach welcher kürzesten Zeit eine Änderung des zugehörigen Signals eine Wirkung am Ausgang des durch die zeitbehaftete Boolesche Funktion beschriebenen Schaltungsteils hervorrufen kann. Da das Ziel der Methode die Bestimmung der Anzahl von Glitches in einer Schaltung ist, ist es sinnvoll, nur rein kombinatorische Schaltungsteile zu betrachten, die zwischen Registerstufen liegen. Die zu betrachtenden Signale werden dabei als Funktionen der Eingangssignale eines Schaltungsteiles dargestellt. Diese Eingangssignale können entweder von den primären Eingängen der Gesamtschaltung oder von Registerausgängen stammen, werden also in jedem Fall als synchron angenommen. In Bild 3.5 ist ein Beispiel für eine kombinatorische Logik zwischen zwei Registerstufen dargestellt. Dabei ist für alle Gatter die gleiche Verzögerungszeit $d = 1$ angenommen. Die Rechenregeln der Booleschen Algebra sind ebenso für die zeitbehafteten Booleschen Funktionen anwendbar. Der Zeitindex wird dabei von der zugehörigen Variable mit

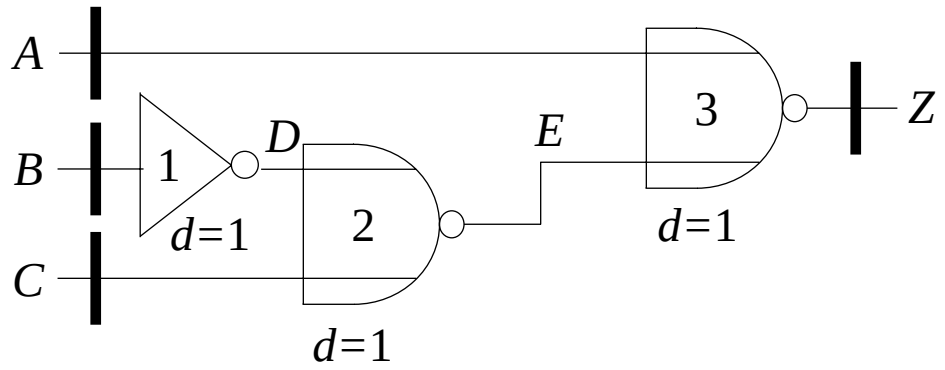



Bild 3.5. Kombinatorische Logik zwischen zwei Registerstufen. Die beschreibende zeitbehaftete Boolesche Funktion ist in Gleichung 3.2 gegeben. Die Zahlen in den Gattern dienen zur Nummerierung.

einem "," getrennt, Variable und Zeitindex werden in runden Klammern zusammengefasst und ergeben so eine zeitbehaftete Boolesche Variable. Der Zeitindex ist invariant gegenüber algebraischen Umformungen. Stellt man die logische UND-Verknüpfung als Produkt und die logische ODER-Verknüpfung als Summe dar, so lässt sich jede Boolesche Gleichung als eine Summe von Produkttermen (*P-Termen*) schreiben. Es können die folgenden Theoreme formuliert werden:

**Theorem 3.1** *Wird eine Variable durch einen zeitbehafteten Booleschen Ausdruck ersetzt, so wird zu jedem Zeitindex in diesem Ausdruck der Zeitindex der ursprünglichen Variable addiert.*

$$X_1 = (X_2, d_2) \quad , \quad X_3 = (X_1, d_1) \quad \text{dann} \quad X_3 = (X_2, d_1 + d_2)$$

**Theorem 3.2** *Die Gesamtverzögerungszeit einer UND-Verknüpfung zweier zeitbehafteter Boolescher Variablen ist gleich der maximalen Verzögerungszeit der beiden Einzelvariablen.*

$$(X_1, d_1) \cdot (X_2, d_2) = (X_1 \cdot X_2, \max(d_1, d_2))$$

**Beweis der Theoreme 3.1 und 3.2** siehe [32].

Für die Schaltung aus Bild 3.5 lautet somit die zeitbehaftete Boolesche Funktion für das Signal $D$ (Gatter 1)

$$D = (\overline{B}, 1)$$

und eingesetzt in die Gleichung für das Signal $E$ (Gatter 2)

$$E = \overline{(D, 1) + (C, 1)} = (B, 2) \cdot (\overline{C}, 1) \quad .$$

Für das Signal $B$ werden dabei die Verzögerungszeiten der Gatter 1 und 2 addiert. Nach dieser Methode ergibt sich schließlich für das Signal $Z$

$$\begin{aligned} Z &= \overline{(A, 1) + (E, 1)} = \overline{(A, 1) + \overline{(D, 2) + (C, 2)}} \\ &= \overline{(A, 1) + (B, 3) \cdot (\overline{C}, 2)} \\ &= (\overline{A}, 1) \cdot (\overline{B}, 3) + (\overline{A}, 1) \cdot (C, 2) \quad . \end{aligned} \tag{3.2}$$

Die letzte Zeile von (3.2) ist in der Form einer Summe aus P-Termen, wie sie im Folgenden zur Ermittlung der Glitches im Signal $Z$ benötigt wird.

#### 3.2.2.2 Simulation der zeitbehafteten Booleschen Funktionen

Der wesentliche Unterschied zwischen der Simulation der zeitbehafteten Booleschen Funktionen und einer Logiksimulation besteht darin, dass bei der Logiksimulation die Funktion jedes einzelnen Gatters analysiert wird, wogegen zur Simulation der zeitbehafteten Booleschen Funktionen lediglich die logische Beschreibung des Ausganges eines Logikblockes herangezogen wird. Bei der hier betrachteten Methode **GC** (**G**litch **C**ounting) [61, 67, 62] wird das Ausgangsverhalten entsprechend einer ereignisgesteuerten Simulation nur für Zustandsänderungen an den Eingängen betrachtet. Bei einem Übergang an den Eingängen des Logikblocks von einem Eingangsmuster $x_i$ zu einem Eingangsmuster $x_{i+1}$ wird zunächst der neue Ausgangszustand $y_{i+1}$ mit einem Zero-Delay-Modell im voraus berechnet. Dann erfolgt in Zeitschritten $t_k$, $k = 1, \ldots, K$, das Anlegen der Signalwerte des neuen Eingangszustandes an die Variablen der zeitbehafteten Booleschen Gleichung. $K$ bezeichnet die Anzahl der verschiedenen Zeitindizes in einer Gleichung. Die Schrittweite bestimmt sich aus den Zeitindizes der Variablen in der Gleichung. Startzeitpunkt $t_1$ ist dabei der kleinste vorkommende Zeitindex $d_{min}$, $t_1 = d_{min}$. Der nächste Zeitschritt entspricht dem nächstgrößten Zeitindex in der Gleichung bis schließlich der größte Zeitindex $d_{max}$ im letzten Schritt $t_K = d_{max}$ erreicht ist. Der Wert einer Eingangsvariablen wird nur dann in den neuen Wert umgeändert, wenn der Zeitindex der Variablen mit dem Zeitschritt übereinstimmt. Zu jedem Zeitschritt erfolgt ein Vergleich des aktuellen Gesamtwertes $y(t_k)$ der Gleichung mit dem im voraus berechneten Wert $y_{i+1}$. Gilt für den alten und den im voraus berechneten neuen Ausgangszustand

$$y_{i+1} \bigoplus y_i = 0 \quad , \tag{3.3}$$

so lautet die Bedingung für einen Glitch

$$\exists k \quad : \quad y(t_k) \bigoplus y_{i+1} = 1 \quad . \tag{3.4}$$

Gilt hingegen

$$y_{i+1} \bigoplus y_i = 1 \quad , \tag{3.5}$$

so lautet die Bedingung für einen Glitch

$$\exists k \quad : \quad (y(t_k) \bigoplus y_{i+1} = 0) \cdot (y(t_k + \delta) \bigoplus y_{i+1} = 1) \quad , \tag{3.6}$$

wobei $\delta$ die zeitliche Länge des Glitches ergibt.

Als Beispiel wird das Verhalten des Ausganges $Z$ für den Zustandsübergang $x_1 = (A, B, C) = (0, 1, 1)$ nach $x_2 = (0, 0, 0)$ am Eingang der Schaltung aus Bild 3.5 betrachtet. Gemäß Gleichung (3.2) ergibt sich für den Ausgangszustand $y_1 = (Z) = 1$ und $y_2 = 1$. Es liegt also der Fall aus Gleichung (3.3) vor. Aus (3.2) folgt weiter $d_{min} = 1$ und $d_{max} = 3$. Zum Startzeitpunkt $t_1 = d_{min} = 1$ kann nur eine Änderung des Signals $A$ auf den Ausgang wirken. An $A$ tritt beim jedoch Zustandübergang $x_1 \rightarrow x_2$ keine Änderung auf. Zeitschritt $t_1$ bleibt daher bei der Analyse unbeachtet. Zum Zeitschritt $t_2 = 2$ wirkt sich der Übergang des Signals $C$ von "1" nach "0" aus, der rechte P-Term in (3.2) und damit der Wert der gesamten Gleichung ergibt "0". Somit trifft nun die Bedingung (3.4) zu und ein Glitch wird detektiert. Erst im Zeitschritt $t_K = t_3 = d_{max}$ wirkt sich die Änderung des Signals $B$ von "1" nach "0" auf den Ausgang aus. Der linke P-Term in (3.2) und somit der Gesamtwert am Ausgang $Z$ ergibt nun "1". In der nachfolgenden Darstellung ist das Beispiel tabellarisch wiedergegeben. Der dabei auftretende unerwünschte Signalzustand von $Z$ ist mit $\square$ gekennzeichnet.

| $i$ | $x_i$ | $Z$ | $(\overline{A},1)$ | $(\overline{B},3)$ | $+$ | $(\overline{A},1)$ | $(C,2)$ |
|---|---|---|---|---|---|---|---|
| 0 | 011 | 1 | 1 | 0 | | 1 | 1 |
| 1 | 000 | 1 | 1 | 0 | | 1 | 1 |
| 2 | 000 | 0 | 1 | 0 | | 1 | 0 |
| 3 | 000 | 1 | 1 | 1 | | 1 | 0 |

**Anmerkung:** *Für die Bestimmung der Glitches anhand der zeitbehafteten Booleschen Funktionen sollte beim Aufstellen der Gleichungen das Theorem 3.2 nicht angewandt werden, da sonst möglicherweise bestimmte, in der Schaltung tatsächlich auftretende Glitches nicht detektiert werden können.*

Die Begründung erfolgt anhand eines Beispiels. Gegeben sei die zeitbehaftete Boolesche Funktion

$$Z = (A, d_1) \cdot (A, d_3) \cdot R_1 + (\overline{A}, d_2) \cdot R_2 \quad ,$$

wobei $R_1$ und $R_2$ zeitbehaftete Boolesche Restterme sind und $[A, \overline{A}] \notin R_1$, $[A, \overline{A}] \notin R_2$. Es gelte $d_1 < d_2 < d_3$. Die Funktion besteht aus den P-Termen $P_1 = (A, d_1) \cdot (A, d_3) \cdot R_1$ und $P_2 = (\overline{A}, d_2) \cdot R_2$. Zum Zeitpunkt $t_0$ sei $R_1 = 1, R_2 = 1, P_1 = 1, P_2 = 0$ und somit $Z = 1$, es erfolge ein Wechsel des Eingangssignals an $A$ der Form $1 \rightarrow 0$. Mit den obigen Festlegungen folgt somit für den Zeitpunkt $t_0 + d_3$: $R_1 = 1, R_2 = 1$ (unabhängig von $A$), $P_1 = 0, P_2 = 1$ und somit $Z = 1$. Jedoch gilt für den Zeitpunkt $t_0 + d_1$: $R_1 = 1, R_2 = 1$, $P_1 = 0, P_2 = 0$ (wegen $d_2 > d_1$) und somit $Z = 0$. Dieser ungewollte Schaltvorgang zum Zeitpunkt $t_0 + d_1$ kann bei Anwendung von Theorem 3.1 wegen des daraus folgenden Wegfalls von $(A, d_1)$ nicht detektiert werden.

Die Anzahl der detektierten Glitches für einen gesamten Satz von Eingangstestmustern wird in einer Variablen *count* abgelegt. Mit $N$ aufeinanderfolgenden Eingangstestmustern, also $N - 1$ Zustandsübergängen am Eingang, errechnet sich die Zahl der pro Übergang durchschnittlich auftretenden Glitches im Ausgangssignal $Z$ zu

$$a(Z) = \frac{count}{N-1} \quad . \tag{3.7}$$

Die Methode **GC** erlaubt es, die Glitch-Aktivität sehr genau zu erfassen. Im Folgenden dient sie als Richtlinie für die Genauigkeit anderer Verfahren. Sollen alle möglichen Zustandsübergänge am Eingang betrachtet werden, so ergibt sich für eine zeitbehaftete Boolesche Funktion mit $n$ Eingangsvariablen eine Zahl von $N = 2^{2n} - 2^n + 1$ Eingangsmustern. Damit können alle möglichen Glitches exakt erfasst werden. Für eine größere Anzahl von Eingangsvariablen ist dies jedoch nicht mehr praktikabel und es muss eine kleinere repräsentative Auswahl von Testmustern verwendet werden. Auf das Generieren von geeigneten Sätzen von Testmustern wird hier nicht näher eingegangen. Methoden dazu sind beispielsweise in [11] behandelt.

## 3.3 Bestimmung der Glitch-Aktivität durch direkte Analyse der zeitbehafteten Booleschen Funktionen

Mit der Simulation von zeitbehafteten Booleschen Funktionen oder durch Logiksimulation ist zwar eine genaue Erfassung der Glitches in einer Schaltung möglich, für bestimmte

Anforderungen genügt jedoch oftmals bereits eine Abschätzung der Größenordnung der ungewollten Schaltaktivität. Eine quantitative Erfassung der gesamten Schaltaktivität ist nur dann von Bedeutung, wenn der tatsächliche Leistungsverbrauch einer Schaltung bestimmt werden soll. Für einen Vergleich verschiedener Realisierungen hinsichtlich dem Maß an ungewollter Schaltaktivität ist eine qualitative Aussage ausreichend. In einer zeitbehafteten Booleschen Funktion steckt bereits Information, die zur Ermittlung von ungewollter Schaltaktivität nötig ist, nämlich die zeitliche Zuordnung der Signale zueinander und ihre Wirkung auf den Ausgang des jeweils betrachteten Gatters.

### 3.3.1 Bedingungen für das Auftreten von Glitches

Die Information über das Maß an ungewollter Schaltaktivität lässt sich direkt aus den zeitbehafteten Booleschen Funktionen gewinnen. Dazu werden alle in der Funktion vorkommenden Variablen daraufhin untersucht, mit welchem Vorzeichen (negiert oder nicht negiert) und mit welchem Zeitindex sie in den verschiedenen P-Termen vorkommen. Bei dieser Methode wird wieder vorausgesetzt, dass die Variablen für Signale stehen, die primäre Eingangssignale der Gesamtschaltung oder Ausgangssignale von Registern sind. Eine prinzipielle Erläuterung soll anhand eines Beispiels gegeben werden. Betrachtet wird die zeitbehaftete Boolesche Funktion

$$Z = (\overline{A}, d_1) \cdot (\overline{B}, d_2) + (A, d_2) \cdot (C, d_3) \quad . \tag{3.8}$$

Für die Zeitindizes der Variablen gelte $d_1 < d_2 < d_3$. In dieser Funktion kommt die Variable $A$ in beiden P-Termen vor, jedoch jeweils mit unterschiedlichem Zeitindex. Eine Änderung des Signalwertes von $A$ kann wegen $d_1 < d_2$ den Wert des linken P-Terms $(\overline{A}, d_1) \cdot (\overline{B}, d_2)$ früher verändern als den des rechten P-Terms $(A, d_2) \cdot (C, d_3)$. Es sei angenommen, dass zu Beginn der Betrachtungen $(\overline{A}, d_1) \cdot (\overline{B}, d_2) = 1$ und $(A, d_2) \cdot (C, d_3)$=0 und damit nach (3.8) $Z = 1$ gilt. Tritt nun zum Zeitpunkt $t_0$ am Eingang eine Änderung des Wertes von $A$ in der Form $0 \rightarrow 1$ auf, sodass zum Zeitpunkt $t_0 + d_3$ gilt $(\overline{A}, d_1) \cdot (\overline{B}, d_2) = 0$, $(A, d_2) \cdot (C, d_3)$=1 und $Z = 1$, so ist zwar der Zustand des Ausganges $Z$ nach $t_0 + d_3$ unverändert, der linke P-Term war jedoch bereits nach $t_0 + d_1$ auf dem Wert "0". Der Wert des rechten P-Terms ändert sich erst zum Zeitpunkt $t_0 + d_2$ nach "1". Bei dieser Änderung des Einganges ergibt sich also ein Glitch mit der Länge $d_2 - d_1$ am Ausgang $Z$ in Form eines Signalwechsels $1 \rightarrow 0 \rightarrow 1$. Nur aufgrund der Bedingungen $\overline{A} \in$ linker P-Term, $A \in$ rechter P-Term und $d_1 < d_2$ können in dem durch (3.8) beschriebenen Signal $Z$ vier mögliche Glitches vorhergesagt werden, nämlich für die Übergänge $(ABC) = (000) \rightarrow (101)$, $(000) \rightarrow (111)$, $(001) \rightarrow (101)$ und $(001) \rightarrow (111)$.

Die Bedingungen für das Auftreten von Glitches lassen sich allgemein formulieren. Der Algorithmus **DGA** (**D**irekte **G**litch **A**nalyse) zur Untersuchung von zeitbehafteten Booleschen Funktionen hinsichtlich der Glitch-Aktivität mittels dieser Bedingungen ist im Folgenden als Pseudo-Programmcode dargestellt. Durch Fallunterscheidungen wird dabei eine Funktion hinsichtlich der verschiedenen Bedingungen untersucht. Die Variablen in den P-Termen werden nacheinander abgearbeitet. Bereits abgearbeitete Variablen werden in eine "Erledigt"-Liste geschrieben. Die P-Terme werden ebenfalls nacheinander abgearbeitet und jeweils mit allen anderen P-Termen verglichen. Die ermittelten Glitches werden zur Gesamtanzahl der Glitches $G(Z)$ für die Funktion $Z$ addiert.

**Algorithmus DGA:**

**BEGINN** Betrachte die zeitbehaftete Boolesche Funktion $Z$.
$s = 0$, $n = 0$, $G(Z) = 0$
**FÜR** $i = 1$ **BIS** (Zahl der P-Terme in $Z$):
**FÜR** $j = 1$ **BIS** (Zahl der P-Terme in $Z$), $(i \neq j)$):
**FÜR** Alle Variablen $X$ in P-Term $i$:
**FALLS** ( $(X, d_1) \in$ P-Term $i$ & $(\overline{X}, d_2) \in$P-Term $j$ & $d_1 \neq d_2$):
& $X \notin$ "Erledigt"-Liste.
s=s+1.
schreibe $X$ in die "Erledigt"-Liste.
**ENDE** der Abfrage
**FALLS**
...
**ENDE** der Abfrage
**ENDE** der Schleife
**ENDE** der Schleife
$G(Z) = G(Z) + s$
**ENDE** der Schleife
**ENDE**

Normiert man dei Gesamtzahl $G(Z)$ der detektierten Glitches an $Z$ mit der Anzahl aller möglichen Zustandsänderungen an einem Eingang mit $n$ Einzelsignalen, so erhält man die durchschnittliche Glitch-Aktivität

$$a(Z) = \frac{G(Z)}{2^{2n} - 2^n} \quad . \tag{3.9}$$

Die Anzahl der Bedingungen, unter denen Glitches auftreten können, hängt mitunter von der Anzahl der Variablen und der Zusammensetzung der P-Terme ab. Es ist daher nicht möglich, alle Bedingungen in geschlossener Form aufzuführen. Im Folgenden sind einige Bedingungen formuliert, mit denen sich im allgemeinen ein großer Teil der Glitch-Aktivität bestimmen lässt. Dazu sind typische Gleichungskonstrukte und die dafür nötigen Bedingungen für das Auftreten von Glitches angegeben.

1) Gleichungskonstrukt von der Form

$$Z = (X_1, d_1) \cdot (X_2, d_2) \cdot \ldots + (\overline{X_1}, d_3) \cdot (X_2, d_4) \cdot \ldots + \ldots$$

**1. Bedingung:** $d_1 \neq d_2 \neq d_3 \neq d_4$.

2) Gleichungskonstrukt von der Form

$$Z = (\overline{X_1}, d_1) \cdot (\overline{X_1}, d_3) \cdot \ldots + (X_1, d_2) \cdot (X_2, d_4) \cdot \ldots + \ldots$$

**2. Bedingung:** $d_1 < d_2$.
**3. Bedingung:** $d_2 < d_3$.

3) Gleichungskonstrukt von der Form

$$\begin{aligned} Z &= (X_1, d_1) + (X_2, d_2) + \ldots \quad \text{oder} \quad Z = (\overline{X_1}, d_1) + (X_2, d_2) + \ldots \\ \text{oder} \quad Z &= (\overline{X_1}, d_1) + (\overline{X_2}, d_2) + \ldots \end{aligned}$$

**4. Bedingung:** $d_1 \neq d_2$.

4) Gleichungskonstrukt von der Form

$$\begin{aligned} Z &= (X_1, d_1) + (\overline{X_1}, d_2) \cdot (X_2, d_3) \cdot \ldots + \ldots \\ \text{oder} \quad Z &= (\overline{X_1}, d_1) + (X_1, d_2) \cdot (X_2, d_3) \cdot \ldots + \ldots \end{aligned}$$

**5. Bedingung:** $d_1 \neq d_2$.

Die absolute Genauigkeit der Methode **DGA** kann natürlich durch Hinzuziehen weiterer Bedingungen erhöht werden. Für bestimmte praktische Anwendungen reicht jedoch bereits die Untersuchung einer geringen Anzahl von Bedingungen aus, um eine ausreichende Genauigkeit zu erreichen. Dies wird im nächsten Abschnitt näher erläutert.

### 3.3.2 Anwendbarkeit der Methode

Mit dem Prinzip der direkten Analyse zeitbehafteter Boolescher Funktionen ist eine schnelle Abschätzung der Glitch-Aktivität an jedem Knoten einer Schaltung möglich. Der eigentliche Anwendungszweck dieser Methode ist die Bestimmung eines Glitch-Gewichtes für jeden Knoten, sodass Retiming-Verfahren zur Minimierung der Glitch-Aktivität eingesetzt werden können. Retiming im klassischen Sinne ist eine Methode zur Minimierung der Taktperiode einer Schaltung. Im Kapitel 4 wird Retiming als Methode zur Reduzierung der ungewollten Schaltvorgänge in einer Schaltung verwendet. Das Glitch-Gewicht $g(v)$ ist dabei die maßgebliche Größe, entsprechend der Knotenverzögerungszeit bei der klassischen Retiming-Anwendung. Das Ziel ist also die Summe aller Gewichte in einem Pfad $p$ durch geeignete Platzierung von Registern zu minimieren. In Bild 3.6 ist ein Graph mit vier Knoten $v_1 \ldots v_4$ dargestellt. Die Zahlen in den Knoten entsprichen einem Gewicht $z(v_\beta)$ und können

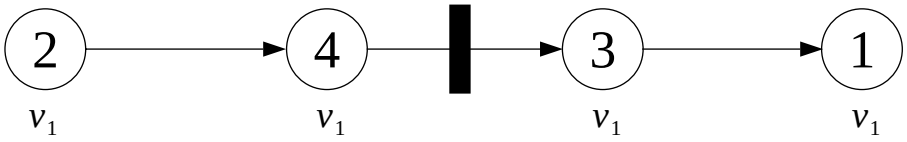



Bild 3.6. Graph mit optimaler Registerplatzierung. Die Zahlen in den Knoten werden als Gewichtsgrößen $z(v_\beta)$ interpretiert, wobei $z(v_1) = 2$, $z(v_2) = 4$, $z(v_3) = 3$, $z(v_4) = 1$

je nach Betrachtungsweise als Knotenverzögerungszeit $z(v_\beta) = d(v_\beta)$ oder Glitch-Gewicht $z(v_\beta) = g(v_\beta)$ interpretiert werden. Der Graph besitzt zwei registerfreie Teilpfade $p_1 : v_1 \rightarrow v_2$ und $p_2 : v_3 \rightarrow v_4$ Die Registerplatzierung in der Kante $e(v_2, v_3)$ ist bereits optimal getroffen, d. h. bei einer Verschiebung des Registers ergibt sich mindestens ein registerfreier Teilpfad, dessen Gewichtssumme größer ist als das Maximum der Gewichtssummen der beiden Teilpfade in der gezeigten Anordnung. Mit einem Register kann der Graph in maximal zwei Teilpfade aufgespalten werden. Das Optimierungsziel $\Delta_{min}$ lässt sich als die Minimierung des Maximums der Gewichtssummen der beiden Teilpfade formulieren:

$$\Delta_{min} = \min \max\left(\sum_{\beta=1}^{\mu} z(v_\beta), \sum_{\beta=\mu+1}^{N} z(v_\beta)\right) \quad .$$

Die Zahl $\mu$ gibt dabei die Nummer des Knotens an, nach dem das Register platziert ist. Entscheidend ist, dass die optimale Position $\mu$ direkt von den Größenverhältnissen der Gewichtsgrößen zueinander abhängt nicht jedoch vom absoluten Wert der $z(v_\beta)$. Für den Graphen

aus Bild 3.6 bedeutet dies, dass die optimale Position des Registers immer die gleiche ist, solange gilt: $z(v_4) < z(v_1) < z(v_3) < z(v_2)$. Wesentlich ist, ob das Gewicht an einem Knoten größer oder kleiner als das eines anderen Knotens ist. Für die Bestimmung der Gewichte genügt es also, Informationen über die Größenbeziehungen der einzelnen Gewichte zueinander zu gewinnen. Experimentell kann verifiziert werden, dass der Algorithmus **DGA** in der Lage ist dies zu leisten. Bild 3.7 zeigt einen Vergleich der **DGA**-Methode mit der Simulation der zeitbehafteten Booleschen Funktionen für vier verschiedene kombinatorische ISCAS89 Benchmark-Schaltungen. Für die **DGA**-Methode wurden aus der obigen Auflistung nur die Bedingungen 1, 4 und 5 berücksichtigt. Das Ergebnis der Simulation der zeitbehafteten Booleschen Funktionen ist dabei analog zum Ergebnis entsprechender Logiksimulationen. Zur Simulation wurden jeweils 1000 Testvektoren verwendet. Auf den $x$-Achsen der Grafik sind jeweils die einzelnen Knoten der Schaltungen aufgetragen. In $y$-Richtung ist die ermittelte Glitch-Aktivität an den Ausgängen der entsprechenden Knoten angegeben. Der Vorteil der **DGA**-Methode gegenüber einer Simulation zeigt sich deutlich in der Rechenzeit. In Tabelle 3.1 sind die Prozessorzeiten der beiden Methoden **DGA** und **GC** für kombinatorische ISCAS89 Benchmarks gegenübergestellt. Zur Simulation wurden dabei wieder 1000 Testvektoren verwendet.

| Schaltung | GC | DGA | Schaltung | GC | DGA |
|---|---|---|---|---|---|
| 9symml | 35.2 | 0.6 | count | 77.4 | 0.4 |
| C17 | 0.4 | 0.03 | decod | 1.5 | 0.01 |
| apex6 | 346.5 | 4.2 | example2 | 80.5 | 0.5 |
| b1 | 0.2 | 0.01 | i1 | 4.4 | 0.01 |
| b9 | 39.2 | 0.2 | i2 | 344.4 | 0.8 |
| cm138a | 1.2 | 0.02 | i4 | 42.6 | 0.3 |
| cm151a | 81.9 | 3.9 | i5 | 83.9 | 0.4 |
| cm162a | 9.1 | 0.05 | i7 | 33.5 | 0.3 |
| cm163a | 4.9 | 0.02 | k2 | 430.9 | 7.5 |
| cm42a | 1.0 | 0.01 | lal | 32.3 | 0.25 |
| cm82a | 0.8 | 0.02 | pcle | 6.7 | 0.02 |
| cm85a | 11.5 | 0.1 | sct | 12.5 | 0.07 |

Tabelle 3.1. Prozessorzeit in Sekunden für die Methoden **GC** (mit 1000 Testvektoren) und **DGA** für ISCAS89 Benchmark-Schaltungen.

Die Rechenzeit für eine Simulation steigt linear mit der Anzahl der Testvektoren an. Die Methode **DGA** ist datenunabhängig, es entfällt daher auch die Generierung von geeigneten Testdaten. Beide Methoden sind unabhängig von der Schaltungstopologie und sowohl für reguläre als auch für irreguläre Strukturen anwendbar.

## 3.4 Glitch-Gewicht

Anhand der in den Abschnitten 3.2.2.2 und 3.3 mit unterschiedlichen Methoden ermittelten Glitch-Aktivität kann ein Glitch-Gewicht für jeden Knoten des Signalflussgraphen definiert werden. Die Summe aller Glitch-Gewichte ist die zu minimierende Zielgröße für den im Kapitel 4 vorgestellten Optimierungsalgorithmus.

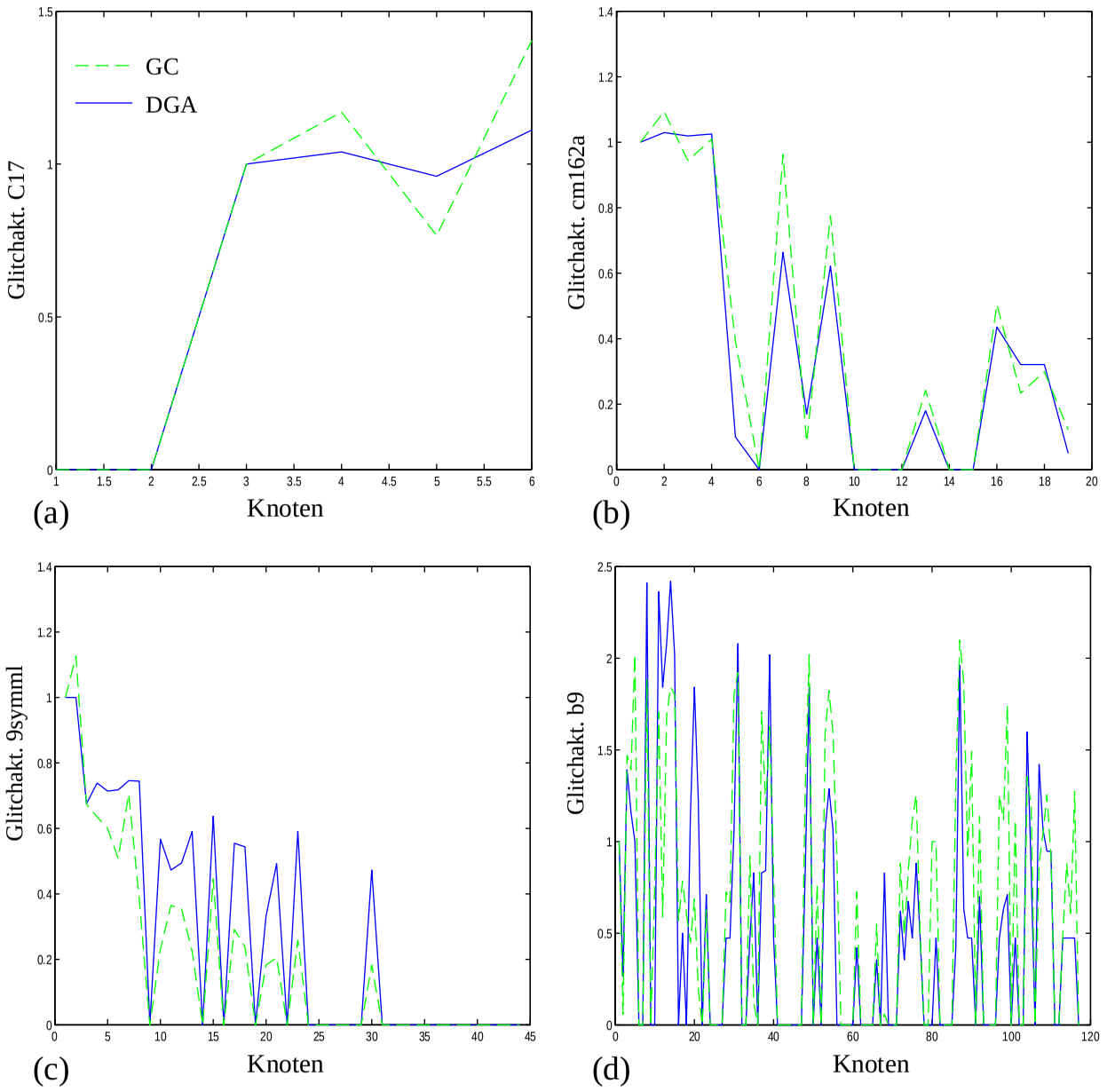



Bild 3.7. Glitch-Aktivität an jedem Knoten von vier kombinatorischen Benchmark-Schaltungen, jeweils ermittelt mit den Methoden **DGA** und **GC**. (a) C17, (b) cm162a, (c) 9symml, (d) b9.

Die durchschnittlich pro Schaltvorgang verbrauchte kapazitive Schaltleistung in einem Knoten $v$, der am Ausgang mit einer Kapazität $C_{\mathrm{L},v}$ belastet ist, ergibt sich aus Gleichung (1.6).

$$P_{\mathrm{s}}(v) = \alpha(v) f C_{\mathrm{L},v} U_{\mathrm{B}}^2$$

Die Aktivität $\alpha$ kann in zwei Anteile aufgespalten werden: in die von der logischen Funktion der Schaltung gegebene also gewollte Schaltaktivität $\alpha_{\mathrm{Logik}}(v)$ und in die ungewollte Schaltaktivität aufgrund von Glitches $a(v)$.

$$\alpha(v) = \alpha_{\mathrm{Logik}}(v) + a(v) \tag{3.10}$$

Die kapazitive Schaltleistung, die nur aufgrund ungewollter Schaltaktivität verbraucht wird, lässt sich somit angeben als

$$P_{\mathrm{Glitch}}(v) = a(v) f C_{\mathrm{L},v} U_{\mathrm{B}}^2 \quad . \tag{3.11}$$

Die Lastkapazität des Knotens $v$ setzt sich aus der Summe aller Eingangskapazitäten $C'_{\mathrm{E},q}$ der Knoten $q$ zusammen, die mit dem Ausgang von $v$ verbunden sind. Hinzu kommen die Kapazitäten $C_{\mathrm{W},q}$ der Verbindungsleitungen, die jedoch in die jeweiligen Eingangskapazitäten einbezogen werden können, sodass sich daraus die effektiven Eingangskapazitäten $C_{\mathrm{E},q} = C'_{\mathrm{E},q} + C_{\mathrm{W},q}$ ergeben. Die Lastkapazität des Knotens $v$ ergibt sich aus der Summe über alle Eingangskapazitäten der Knoten am Ausgang von $v$:

$$C_{\mathrm{L},v} = \sum_q C_{\mathrm{E},q} \quad . \tag{3.12}$$

Für eine Gewichtung der Glitches ist auch ihre zeitliche Ausdehnung von Bedeutung. Von der Länge eines Glitches hängt es unter anderem ab, über wie viele Stufen der Schaltung sich der Glitch fortpflanzen kann. Sehr kurze Glitches werden beim Durchlaufen von Gattern sowohl in der Länge als auch im Spannungshub reduziert [21]. Bei einer Kette von mehreren Gattern ist es sogar möglich, dass ein sehr kurzer Glitch nach einer bestimmten Anzahl von durchlaufenen Gattern vollständig verschwindet, wie dies in Bild 3.8 für eine Kette von UND-Gattern anhand einer SPICE-Simulation gezeigt ist. Der Effekt hängt mit der Trägheit der Gatter zusammen und tritt umso deutlicher auf, je größer die Verzögerungszeit eines Gatters ist. Um diesen Effekt bei der Gewichtung der Glitches zu berücksichtigen, wird ein *Trägheits-*

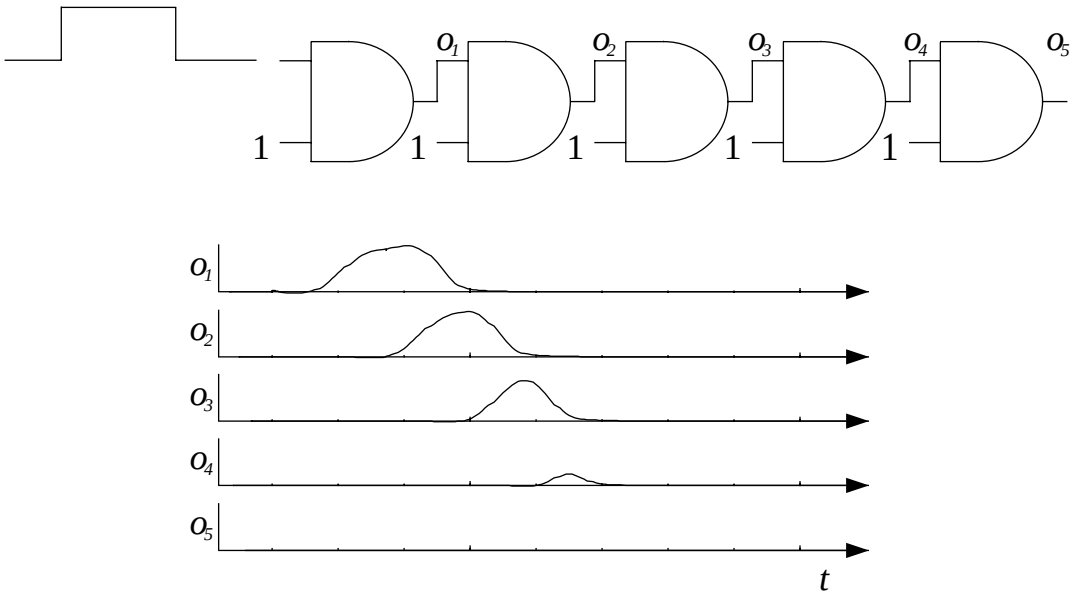



Bild 3.8. Verschwinden eines kurzen Glitches beim Durchlaufen einer Kette von Gattern.

*faktor* $\rho_{q,i}$ für jeden detektierten Glitch Nummer $i$ definiert, der am Eingang eines Knotens $q$ auftritt. Dabei gilt $0 \leq \rho_{q,i} \leq 1$. Der genaue Wert von $\rho_{q,i}$ kann für die verschiedenen Gatter experimentell ermittelt werden. In der Praxis erweist sich eine Diskretisierung von $\rho_{q,i}$ zur Reduzierung des Aufwands bei der Ermittlung des Glitch-Gewichtes als sinnvoll. Beispielsweise kann für einen Inverter mit der Verzögerungszeit $t_d$ die folgende Diskretisierung ausreichend genau sein: $\rho_i = 0$ für $t_{\mathrm{Glitch},i} < \frac{1}{2} t_d$, $\rho_{\mathrm{Inv},i} = 0.5$ für $\frac{1}{2} t_d \leq t_{\mathrm{Glitch},i} < t_d$ und

$\rho_{\mathrm{Inv},i} = 1$ für $t_{\mathrm{Glitch},i} \geq t_d$, wobei $t_{\mathrm{Glitch},i}$ die Länge des Glitches Nummer $i$ bezeichnet. Es sei $G$ die Anzahl der detektierten Glitches. Der Trägheitsfaktor $\rho(v)$ für einen Knoten $v$, der mit $Q$ Knoten am Ausgang belastet ist, wird somit definiert als

$$\rho(v) = \frac{1}{G \cdot Q} \sum_{q=1}^{Q} \sum_{i=1}^{G} \rho_{q,i} \quad . \tag{3.13}$$

Das Glitch-Gewicht $g(v)$wird damit als die mit $\rho(v)$ gewichtete Summe der kapazitive Schaltleistungen aufgrund von Glitches definiert.

$$g(v) = \rho(v) \cdot P_{\mathrm{Glitch}}(v) \tag{3.14}$$

Dieses Glitch-Gewicht wird jedem Knoten in der Schaltung zugeordnet, wobei für Knoten mit mehreren Ausgängen unterschiedlicher Gewichtung wieder die Zerlegung aus Abschnitt 3.1 verwendet werden kann.

# 4. Retiming und Pipelining als Methoden zur Reduzierung der Verlustleistung

In diesem Kapitel wird ein Verfahren zur Minimierung der ungewollten Schaltaktivität vorgestellt, das von den klassischen Verfahren zur Minimierung der Taktperiode einer Schaltung abgeleitet ist. Von dieser klassischen Anwendung stammt der Name *Retiming* und bedeutet die zeitliche Umorganisierung der Verarbeitungsschritte in einer Schaltung. Im Jahre 1983 erschien die erste Veröffentlichung zur Optimierung von Schaltungen mittels Retiming [39]. Seither gab es eine große Anzahl weiter Veröffentlichungen zu diesem Thema. In [45] wurde Retiming als ein Verfahren zur Reduzierung der Verlustleistung vorgestellt, bei dem durch die geeignete Platzierung von Registern als Glitch-Barrieren die Gesamtschaltaktivität der Schaltung verringert wird. Das in der vorliegenden Arbeit vorgestellte Verfahren unterscheidet sich von dem in [45] grundsätzlich zunächst dadurch, dass hier die Register nicht nur als Glitch-Barrieren, sondern in erster Linie zum Balancieren unterschiedlich langer Signalpfade verwendet werden. Desweiteren werden hier andere Verfahren zur Bestimmung der Glitch-Aktivität verwendet. Bild 4.1 verdeutlicht die Wirkung von Registern als Glitch-Barrieren. Die Glitch-Aktivitäten an den Ausgängen der Addierer sind durch Balken symbolisiert. Durch die Einführung einer Registerstufe nach der dritten Addiererreihe (rechtes Bild) wird die Glitch-Aktivität in der unteren Hälfte der Schaltung deutlich reduziert.

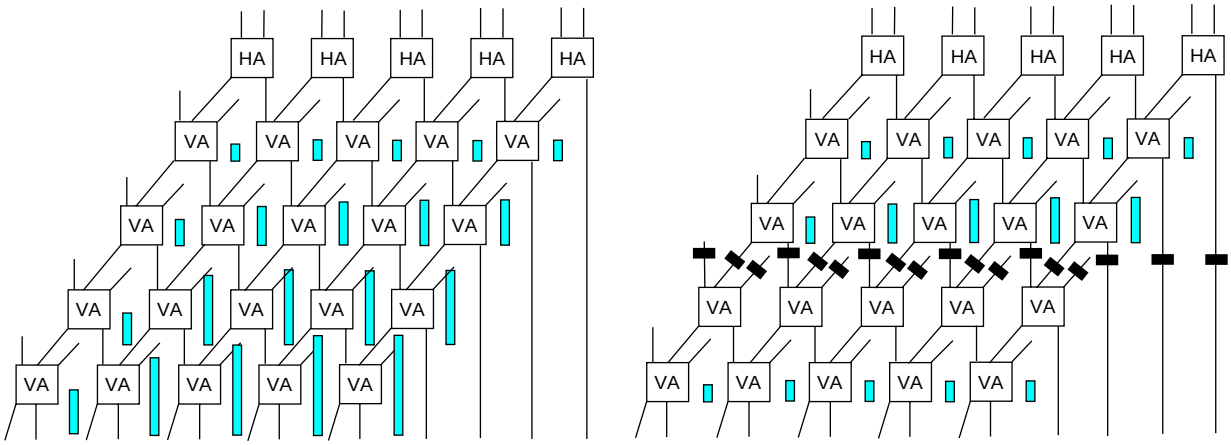



Bild 4.1. Register als Glitch-Barrieren. Durch Einfügen einer Registerstufe in den Multiplizierer wird die Glitch-Aktivität insgesamt deutlich verringert.

Pipelining unterscheidet sich von Retiming nicht grundsätzlich im Verfahren. Beim Pipelining lassen sich die gleichen Algorithmen und Modelle verwenden wie beim Retiming.

Dabei wird die Schaltung in Abschnitte mit kürzeren Laufzeiten zwischen den Registern unterteilt und somit eine höhere Taktrate ermöglicht. Dazu werden beim Pipelining neue Register in die Schaltung eingeführt. Dabei bleibt die Funktion der Schaltung bis auf eine zusätzliche Latenzzeit, die durch die eingefügten Register verursacht wird, unverändert. Ob eine Änderung der Latenzzeit akzeptiert werden kann hängt von der Anwendung der Schaltung ab. Beim Retiming im klassischen Sinne bleibt die Latenzzeit der Schaltung in jedem Fall unverändert. Bereits vorhandene Register werden lediglich umplatziert. Die Wirkung des Pipelinings auf die Verlustleistung einer Schaltung wurde in [37] untersucht.

Das hier vorgestellte Verfahren konzentriert sich auf ein grundsätzlich anderes Ziel als die ebenfalls auf Retiming basierende, in Abschnitt 2.1.3 vorgestellte Methode. Dort wurde das höhere Geschwindigkeitspotential zur Verringerung der Versorgungsspannung genutzt. Hier hingegen sollen mittels Retiming die Glitch-Aktivität in einer Schaltung reduziert werden. In diesem Kapitel erfolgt zunächst die Darlegung der Grundlagen zu den klassischen Retiming-Verfahren. Im weiteren wird das Problem der Minimierung der ungewollten Schaltaktivität auf ein klassisches Retiming-Verfahren übertragen. Es wird gezeigt, dass mit dieser Methode durch die Reduzierung der Glitch-Aktivität die Verlustleistung einer Schaltung verringert werden kann.

## 4.1 Grundlagen zu Retiming-Verfahren

Das hier zugrundegelegte Schaltungsmodell wurde bereits in Abschnitt 3.1 vorgestellt. Zunächst wird die Schaltung durch ihren Signalflussgraphen $G =< V, E, d, w >$ repräsentiert, ohne Berücksichtigung der Glitch-Gewichte an den einzelnen Knoten. Die Funktionalität der Knoten wird im Schaltungsmodell nicht betrachtet. Es wird wieder ein statisches Worst-Case-Verzögerungszeitmodell verwendet. Im Folgenden wird das für die späteren Betrachtungen zugrundegelegte Registermodell vorgestellt. Im Anschluss daran werden klassische Retiming-Verfahren behandelt, wobei für die grundlegenden Erläuterungen zunächst die ursprüngliche Anwendung der Verfahren zur Minimierung der Taktperiode betrachtet wird. Die später vorgestellte Anwendung zur Reduzierung der Glitch-Aktivität ist eine Modifikation der klassischen Verfahren mit anderer Zielfunktion, weshalb die prinzipiellen Betrachtungen ebenso am Beispiel der Geschwindigkeitsoptimierung von Schaltungen durchgeführt werden können.

### 4.1.1 Registermodell

Register sind Schaltungen, die logische Zustände zwischenspeichern können. Dabei wird der Eingangszustand mit einem Taktsignal zu einem definierten Zeitpunkt übernommen und solange am Ausgang gehalten, bis zu einem späteren Taktzeitpunkt ein anderer Eingangszustand anliegt, der dann an den Ausgang übernommen wird. Man unterscheidet zwischen flankengesteuerten und zustandsgesteuerten Registern, je nach dem, ob der Eingangszustand nur zum Zeitpunkt des Auftretens einer Taktflanke übernommen wird oder während der gesamten Dauer eines Taktimpulses. Eine weitere Unterteilung erfolgt in statische und dynamische Register. Statische Register halten einen gespeicherten Zustand für theoretisch unbegrenzte Zeit, unter der Voraussetzung, dass eine Versorgungsspannung anliegt. Als Ausführungsformen sind sowohl flanken- als auch zustandsgesteuerte statische Register möglich. Dynamische CMOS-Register hingegen funktionieren durch Ladungsspeicherung auf den Gates von Transistoren. Dabei wird die Ladung über einen elektronischen Schalter, ein sog. *Transmission-*

*Gate*, auf die Gates eines CMOS-Inverters geführt. Da das Transmission-Gate üblicherweise durch einen NMOS- und einen dazu parallelen PMOS-Transistor realisiert wird, sind zur Ansteuerung zwei Signale nötig: das nicht invertierte und das invertierte Steuersignal. Eine solche Transmission-Gate-Inverter-Einheit bildet ein *Latch*. Wird der Schalter geöffnet, so bleibt die Ladung für gewisse Zeit auf den Gates erhalten. Da die Gatekapazität in der Realität nicht ideal ist, fließt die Ladung mit der Zeit über parasitäre Widerstände ab, der gespeicherte Zustand muss also nach gewisser Zeit aufgefrischt werden. Üblicherweise kann ein Zustand über einige Millisekunden gespeichert werden, dann muss eine Auffrischung erfolgen. Diese Technik erfordert eine hohe Eingangsimpedanz der Transistoren, wie sie mit der CMOS-Technologie möglich ist. Um einen Zustand für eine volle Taktperiode zwischenspeichern zu können, sind für den Aufbau eines Registers zwei in Kette geschaltete Latches nötig, deren Transmission-Gates mit unterschiedlichen, nicht überlappenden Taktphasen angesteuert werden. Ein Nachteil dieses Registertyps ist die Notwendigkeit von vier Taktleitungen für die Zuführung des *Pseudo-Vierphasentaktes*. Ein solches Register ist immer zustandsgesteuert. Der große Vorteil dynamischer Register ist die im Allgemeinen einfachere Realisierung und der geringere Flächenbedarf als bei statischen Registern. Bild 4.2 zeigt das Layout und Taktschema eines dynamischen Registers. Dynamische Register werden häufig in Datenpfa-

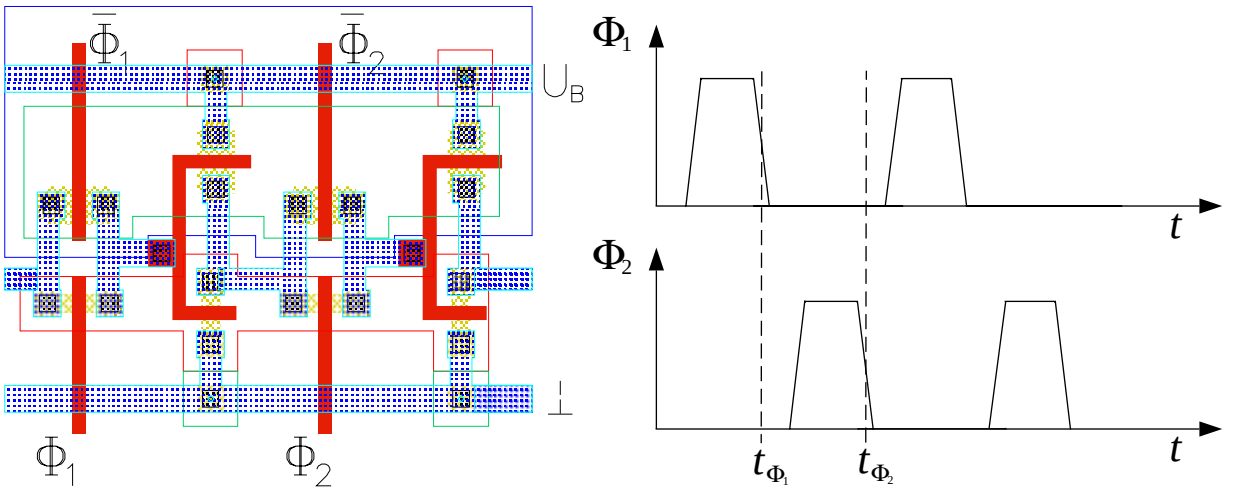



Bild 4.2. Layout und Taktschema eines dynamischen Registers.

den verwendet. In diesen Schaltungen wird der gespeicherte Zustand in jeder Taktperiode neu überschrieben, die Flüchtigkeit der Ladungsspeicher ist bei den üblichen Taktraten daher unproblematisch.

Bei der Umverteilung von dynamischen Registern ist es auch möglich, die Register jeweils in zwei Latches aufzuteilen und diese getrennt zu verschieben, wobei natürlich die Reihenfolge der Taktphasen eingehalten werden muss. Es sei $d_{\text{Logik}}$ die maximale Verzögerungszeit der Logikschaltung zwischen zwei Latches und $t_{\Phi_1}$ und $t_{\Phi_2}$ die diskreten Zeitpunkte zu denen die Übernahme der Daten durch die Takte $\Phi_1$ und $\Phi_2$ erfolgt. Für den Abstand der Taktflanken von $\Phi_1$ und $\Phi_2$ muss gelten

$$|t_{\Phi_2} - t_{\Phi_1}| > d_{\text{Logik}} \quad . \tag{4.1}$$

Die Funktion der Schaltung bleibt dabei unverändert. Dieses Prinzip ist in Bild 4.3 links dargestellt. Kommen dabei die Latches vor CMOS-Logikblöcken zu liegen, deren Eingänge die Gates von MOS-Transistoren bilden, so können deren Kapazitäten als Ladungsspeicher

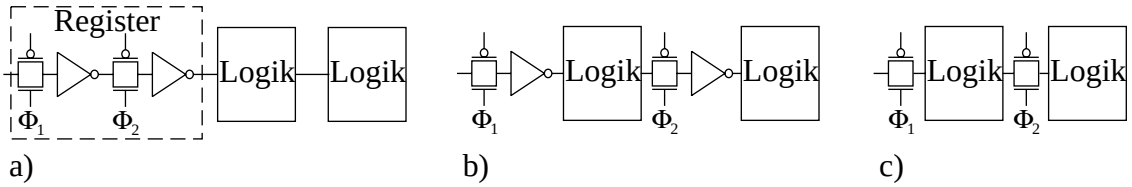



Bild 4.3. a) Ursprüngliche Schaltung mit dynamischem Register. b) Aufteilen des Registers in zwei Latches, die getrennt verschoben werden. c) Die Eingangskapazitäten der Logikblöcke hinter den Latches werden als Ladungsspeicher verwendet, die Inverter entfallen.

verwendet und die Inverter der Latches eingespart werden, wie in Bild 4.3 rechts gezeigt ist. In jedem Pfad, in dem Inverter auf diese Weise eingespart werden, muss natürlich immer eine geradzahlige Anzahl von Invertern entfernt werden, da sich ansonsten die Funktion der Schaltung ändert. Durch den Wegfall der Inverter reduziert sich der Leistungsverbrauch der Schaltung. Eine Retiming-Methode mit stückweise linearer Kostenfunktion, die von dieser Möglichkeit zur Leistungseinsparung Gebrauch macht, wurde in [82] vorgestellt.

Die Daten können von Registern nur zu fest definierten, äquidistanten Taktzeitpunkten übernommen werden. Ist das Signal vor einem Register mit Glitches behaftet, so darf das Taktsignal zur Übernahme des Eingangswertes erst dann erfolgen, wenn alle Glitches abgeklungen sind und der gültige Signalwert stabil anliegt. Nur so kann die korrekte Funktion der Schaltung gewährleistet werden. Mit der Einhaltung der Bedingung aus Gleichung (4.1) ist dies immer erfüllt. Bild 4.4 dient zur Erläuterung. Es existieren zwei unterschied-

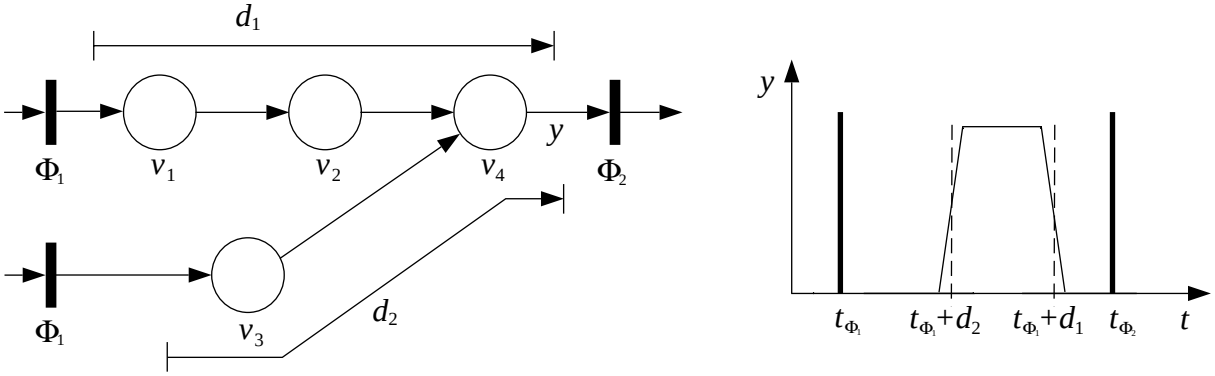



Bild 4.4. Bestimmung des minimalen Abstands der Taktphasen zur Gewährleistung korrekter Funktion bei Signalen die Glitches enthalten. Es muss gelten $t_{\Phi_2} > t_{\Phi_1} + d_2$.

lich lange Pfade von den Latches mit Taktphase $\Phi_1$ zum Latch mit der Taktphase $\Phi_2$: Pfad 1 $= v_1 \longrightarrow v_2 \longrightarrow v_4$ mit der Verzögerungszeit $d_1$ und Pfad 2 $= v_3 \longrightarrow v_4$ mit der Verzögerungszeit $d_2$, wobei $d_2 < d_1$. Die maximale Länge, die ein Glitch erreichen kann ist $t_{\text{Glitch}} = d_1 - d_2$. Der Zeitpunkt des Beginns eines solchen Glitches ist bei $t_{\Phi_1} + d_2$, der Endzeitpunkt ist $t_{\Phi_1} + d_1$. Die maximale Pfadverzögerung ist $\max(d_1, d_2) = d_1$. Nach (4.1) muss gelten $t_{\Phi_2} > t_{\Phi_1} + d_1$. Somit ist der Glitch sicher abgeklungen wenn die Daten mit $\Phi_2$ in das Latch am Ausgang übernommen werden. Latches und Register stellen deshalb Barrieren für Glitches dar. Die Glitch-Aktivität am Ausgang eines Latches oder Registers ist Null. War die Aktivität eines Signals am Eingang eines Latches oder Registers nach Gleichung (3.10) $\alpha = \alpha_{\text{Logik}} + a$, so ist die Aktivität am Ausgang lediglich noch $\alpha = \alpha_{\text{Logik}}$, also die gewünschte Aktivität.

Pfade mit ungleichen Verzögerungszeiten lassen sich durch Register oder Latches synchronisieren. Im Modell wird deshalb die Glitch-Aktivität am Ausgang eines Knotens zu Null angenommen, wenn an allen Eingängen des Knotens jeweils mindestens ein Latch sitzt, vorausgesetzt, dass alle unmittelbar an den Eingängen sitzenden Latches mit der gleichen Taktphase angesteuert werden. Mit Hilfe dieses Registermodells lässt sich die Wirkungsweise aller Retiming-Verfahren zur Geschwindigkeitsoptimierung und zur Verlustleistungsreduzierung erklären.

### 4.1.2 Schaltungsgraph

Das klassische Prinzip des Retiming basiert auf der Modellierung der Schaltung durch ihren Signalflussgraphen $G =< V, E, d, w >$. Dabei bezeichnet $V$ die Menge aller Knoten, $E$ die Menge aller Kanten, $d$ die Verzögerungszeiten der Knoten und $w$ die Kantengewichte. Das Kantengewicht ist gleichbedeutend mit der Anzahl der Register in einer Kante. Die folgenden Betrachtungen zu den klassischen Retiming-Verfahren beschränken sich auf Verwendung ganzer Register, wodurch die im vohergehenden Abschnitt erläuterte aufwendige Berücksichtigung einzelner Taktphasen und deren Abstand zueinander entfällt. Die Kantengewichte $w$ sind daher immer ganzzahlig. Die Verzögerungszeiten der Knoten werden dabei durch die Worst-Case-Verzögerungszeiten repräsentiert. Das Ziel dieser Verfahren ist es, die maximale Pfadverzögerungszeit zwischen zwei Registern zu minimieren. Bild 4.5 zeigt den Signalflussgraphen eines Teils eines digitalen Filters in Lattice-Struktur und die zugehörige Darstellung mit statischem Worst-Case-Verzögerungszeitmodell. Die Funktionalität der ein-

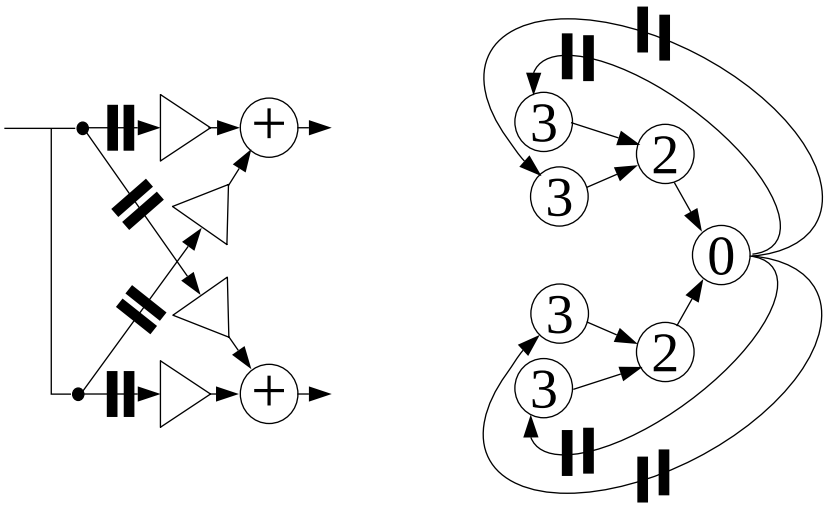



Bild 4.5. Signalflussgraph eines digitalen Filters und Darstellung ohne der Funktionalität der Knoten mit Worst-Case-Verzögerungszeiten.

zelnen Knoten wurde beim Signalflussgraphen weggelassen, für das Auffinden der minimalen Taktperiode ist lediglich die Verzögerungszeit der Knoten von Bedeutung. Eingänge und Ausgänge des Signalflussgraphen werden über einen zusätzlichen, virtuellen Knoten mit der Verzögerungszeit $d = 0$ verbunden. Dadurch ist sichergestellt, dass beim Verschieben von Registern über die Ein- oder Ausgänge der Schaltung hinaus die Latenzzeit unbeeinflusst bleibt.

### 4.1.3 Bestimmung der minimalen Taktperiode

Es wird vorausgesetzt, dass die Schaltung keine verzögerungsfreien Schleifen enthalten kann, d. h. eine gerichtete Masche im Signalflussgraphen muss mindestens ein Register enthalten. Entfernt man alle Kanten aus dem Signalflussgraphen $G$, für die gilt $w(e) > 0$ und die somit Register enthalten, dann ist mit der vorigen Bedingung sichergestellt, dass der reduzierte Signalflussgraph $G_0$ keine Schleifen enthält, also azyklisch ist. Die minimal mögliche Taktperiode ist gleich der maximalen Pfadverzögerung aller Pfade $p$ der reduzierten Schaltung

$$\Psi(G) = \max(d(p) : w(p) = 0) \quad . \tag{4.2}$$

Die maximale Pfadverzögerung kann systematisch durch topologisches Sortieren ermittelt werden. Die Knoten werden dabei nach folgender Regel sortiert: existiert eine Kante $e$ vom Knoten $u$ zum Knoten $v$, $u \xrightarrow{e} v$, dann ist $u$ ein unmittelbarer Vorgängerknoten von $v$. Die Komplexität des topologischen Sortierens ist proportional der Anzahl aller Kanten in der Menge $E$. Alle Knoten werden in der Reihenfolge der topologischen Sortierung einmal aufgesucht und die Verzögerungszeit $\Delta(v)$ vom Anfang eines Pfades bis zum betrachteten Knoten $v$ berechnet. Für die maximale Pfadverzögerung gilt dann: $\Psi(G) = \max_{v \in V}\{\Delta(v)\}$.

### 4.1.4 Schaltungstransformation durch Retiming

Ein manuelles Umplatzieren von Registern in einer Schaltung ist mit der so genannten *Cut-Set-Regel* möglich. Dabei wird eine geschlossene Hüllkurve um einen Schaltungsteil gelegt, die ausschließlich Verbindungskanten schneidet. Die von der Hüllkurve geschnittenen Kanten, die in den so umschlossenen Schaltungsteil (Cut-Set) zeigen, können als Eingänge, die aus diesem Schaltungsteil zeigenden Kanten als Ausgänge definiert werden. Fasst man alle Eingänge zu einem einzigen virtuellen Eingang und alle Ausgänge zu einem einzigen virtuellen Ausgang zusammen, so besagt die Cut-Set-Regel, dass die Summe aller über den (virtuellen) Eingang in die Schaltung geschobenen Register gleich der Summe der am (virtuellen) Ausgang herausgenommenen Register sein muss und umgekehrt. Mit anderen Worten: die Latenzzeit des von der Hüllkurve umschlossenen Schaltungsteiles muss konstant sein. Verzichtet man auf das Zusammenfassen zu einem virtuellen Ein- bzw. Ausgang, so lassen sich die folgenden Regeln formulieren:

- Wird an einem Eingang eines Cut-Sets ein Register entfernt oder hinzugefügt, so muss entsprechend an allen anderen Eingängen des Cut-Sets ein Register entfernt bzw. hinzugefügt werden.
- Wird an einem Ausgang eines Cut-Sets ein Register entfernt oder hinzugefügt, so muss entsprechend an allen anderen Ausgängen des Cut-Sets ein Register entfernt bzw. hinzugefügt werden.
- Wird an den Eingängen des Cut-Sets ein Register entfernt so muss an den Ausgängen ein Register hinzugefügt werden.
- Wird an den Eingängen des Cut-Sets ein Register hinzugefügt so muss an den Ausgängen ein Register entfernt werden.

Durch Anwendung der Cut-Set-Regel wie in Bild 4.6 gezeigt, kann die maximale Pfadverzögerung und somit die minimal möglichen Taktperiode in der Schaltung aus Bild 4.5 von 5 auf 3 Zeiteinheiten reduziert werden. Die Funktionalität der Schaltung bleibt davon unbeeinflusst.

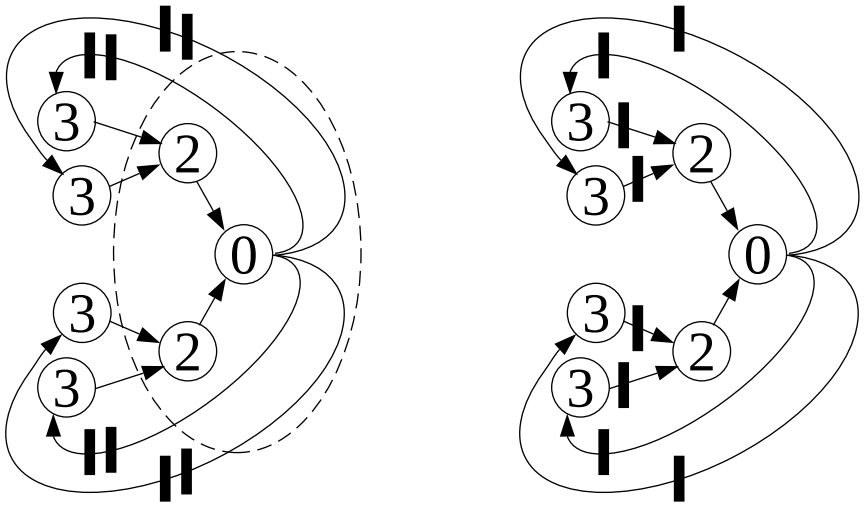



Bild 4.6. Anwendung der Cut-Set-Regel zur Verringerung der minimal möglichen Taktperiode von 5 auf 3 Zeiteinheiten.

Mit der Cut-Set-Regel lassen sich einfache Regeln für das Verschieben von Registern über einzelne Knoten ableiten, indem die Hüllkurve nur um den zu betrachtenden Knoten gelegt wird. Es gelten dann die gleichen Regeln wie oben. In einer realisierbaren Schaltung dürfen alle Kanten nur nicht negative Gewichte aufweisen. Daraus lässt sich weiter ableiten, dass ein Register von einer einlaufenden Kante das Knotens nur dann an eine auslaufende Kante verschoben werden kann, wenn an allen einlaufenden Kanten mindestens ein Register sitzt, sodass durch diesen Schritt keine negativen Register an den Eingängen entstehen.

Eine gesonderte Betrachtung verlangen Schaltungen, die Multiplexer für Zeitmultiplexverfahren enthalten oder allgmeiner, bei denen zusammenhängende Schaltungsteile mit unterschiedlicher Taktfrequenz betrieben werden. Fasst man einen Multiplexer oder Demultiplexer als einen Knoten auf, so ist die Datenrate auf den einlaufenden Kanten eine andere als auf den auslaufenden Kanten. Es sei $f_\mathrm{e}$ die Taktfrequenz an den einlaufenden Kanten, $f_\mathrm{a}$ die Taktfrequenz an den auslaufenden Kanten und $k = \frac{f_\mathrm{a}}{f_\mathrm{e}}$ das Verhältnis der Taktfrequenzen. Gemäß den Definitionen für Multiplexer und Demultiplexer ist $k$ dann gleich dem Verhältnis der Anzahl der auslaufenden Kanten zur Anzahl der einlaufenden Kanten. Für Multiplexer gilt $k$ ganzzahlig mit $k > 1$, für Demultiplexer gilt $k$ ganzzahlig mit $0 < k < 1$. Für das Verschieben von Registern über einzelne Multiplexer- oder Demultiplexer-Knoten müssen die Cut-Set-Regeln dementsprechend erweitert werden.

- Wird an den Eingängen eines Multiplexer- oder Demultiplexer-Knotens ein Register entfernt so müssen an den Ausgängen $k$ Register hinzugefügt werden.
- Wird an den Eingängen eines Multiplexer- oder Demultiplexer-Knotens ein Register hinzugefügt so müssen an den Ausgängen $k$ Register entfernt werden.

Zur Veranschaulichung dient Bild 4.7 In [59] wird dieses Verfahren für das Pipelining von Filterbänken für die diskrete Wavelettransformation vorgestellt, in denen Downsampling angewendet wird. In [78] ist ein algorithmisches Verfahren für das Retiming von Schaltungen angegeben, die Multiplexer enthalten.

Die Wahl der richtigen Cut-Sets zur Erlangung der optimalen Registerverteilung ist bei manueller Vorgehensweise in Schaltungen höherer Komplexität nichttrivial und die Optimalität der Lösung oftmals nicht nachprüfbar. Es ist daher wünschenswert, eine mathematische

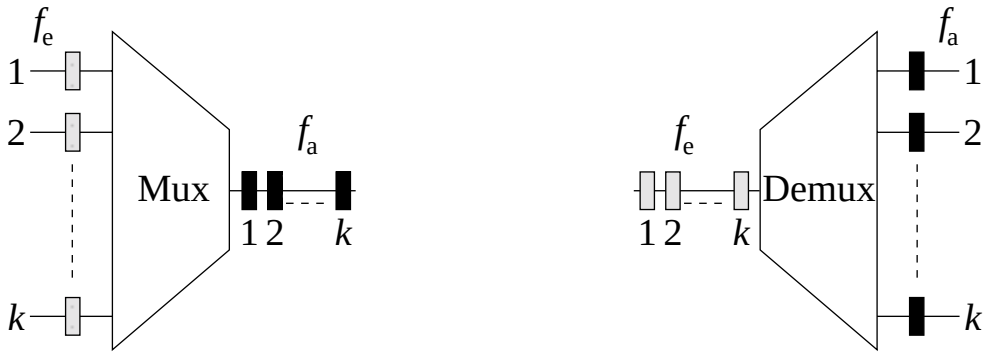



Bild 4.7. Verschieben von Registern über Multiplexer- und Demultiplexer-Knoten mit dem Verhältnis der Taktfrequenzen $k = \frac{f_a}{f_e}$. Die schwarzen und grauen Blöcke symbolisieren die Registerplatzierung jeweils vor und nach dem Retiming.

Formulierung des Problems abzuleiten und in einen geschlossenen Algorithmus zum Auffinden der optimalen Registerplatzierung einzubetten. In der einschlägigen Literatur zu diesem Thema wird Retiming als die Zuweisung ganzzahliger Werte $r(v)$ zu jedem Knoten definiert [38, 39, 40, 41]. Der Wert von $r(v)$ gibt an, wieviele Register jeweils von den Ausgängen eines Knotens $v$ zu den Eingängen verschoben werden. Dazu wird $r(v)$ um eins erhöht falls die Pfadverzögerungszeit $\Delta(v)$ zum Knoten $v$ größer ist als die gewünschte Taktperiode $c$,

$$r(v) = r(v) + 1 \quad \text{falls} \quad \Delta(v) > c \quad . \tag{4.3}$$

Anhand der Werte $r(v)$ kann für jede Kante $e$ das neue Kantengewicht $w_r(e)$ nach dem Retiming aus dem alten Kantengewicht $w(e)$ errechnet werden. Für eine Kante $e$ vom Knoten $u$ zum Knoten $v$, $u \xrightarrow{e} v$ errechnet sich das Kantengewicht nach dem Retiming zu

$$w_r(e) = w(e) + r(v) - r(u) \quad . \tag{4.4}$$

Dadurch wird der Graph $G =< V, E, d, w >$ in den Graphen $G_r =< V, E, d, w_r >$ transformiert. Für das Auffinden der minimal möglichen Taktperiode kann eine binäre Suche verwendet werden. Dabei wird das Zeitintervall, in dem die Lösung gesucht wird, in jedem Schritt halbiert. Zur Verifikation, ob für eine gegebene Taktperiode $c$ eine Retiming-Lösung existiert, geben Leiserson et al. in [41] den Algorithmus **FEAS** (**FEAS**ible clock-period test) an. Dieser Algorithmus ermittelt zugleich die Werte $r(v)$ für alle Knoten, sodass nach dem Auffinden der minimal möglichen Taktperiode die neuen Kantengewichte $w_r(e)$ und somit eine optimale Registerplatzierung berechnet werden können.

**Algorithmus FEAS:** Bestimmung der Werte $r(v)$ für eine vorgegebene Taktperiode $c$, falls diese durch Retiming erreicht werden kann.

**BEGINN**
$\forall i \in V$, setze $r(v) = 0$.
**FÜR** $v = 1$ **BIS** $|V| - 1$:
Bestimme den Graphen $G_r$.
$\forall v$: bestimme die maximale Pfadverzögerung $\Delta(v)$ durch topologisches Sortieren.
**FALLS** $\Delta(v) > c$ setze $r(v) := r(v) + 1$.
**ENDE** der Schleife

Bestimme $\Psi(G_r)$. Falls $\Psi(G_r) \leq c$, dann ist $G_r$ eine
Lösung.
**ENDE**

Ein Beweis für die Korrektheit des Verfahrens findet sich in [41]. Bei dieser Formulierung des Algorithmus' **FEAS** müssen für den Fall, dass für alle Knoten $\Delta(v) > c$ gilt, die maximalen Pfadverzögerungen in jedem Iterationsschritt $V$-mal neu berechnet werden. Dazu wird jedesmal topologisches Sortieren angewendet, dessen Komplexität, wie bereits erwähnt, proportional der Zahl der Kanten im Graphen ist, also von der Ordnung $O(|E|)$. Der Algorithmus **FEAS** ist somit von der Ordnung $O(|V||E|)$. Seit der Veröffentlichung dieses klassischen Retiming-Algorithmus', den man wohl als den Standard-Retiming-Algorithmus bezeichnen kann, gab es zahlreiche weitere Veröffentlichungen zu diesem Thema. Zum Teil stellen diese Publikationen Modifikationen des Standard-Algorithmus' dar mit anderen Modellierungen und geringerem Rechenaufwand [44]. Es ist hier anzumerken, dass der Algorithmus **FEAS** auch zum Pipelining verwendet werden kann. Die Register, die dabei zusätzlich in die Schaltung eingefügt werden sollen, werden dazu zunächst an den primären Eingängen der Schaltung platziert und dann durch Retiming in die Schaltung geschoben. Ein spezielles Retiming-Verfahren, das ausschliesslich für das Pipelining von Schaltungen verwendet werden kann, die ursprünglich keine Register enthalten haben, wird in [46] angegeben. Auch hier ergibt sich eine Rechenzeitersparnis gegenüber **FEAS**.

Als eine rechenzeiteffiziente Alternative zum Algorithmus **FEAS** wurde in [77, 79, 76] ein maschenorientiertes Verfahren zur Bestimmung der minimal möglichen Taktperiode vorgestellt. Dieses Verfahren basiert auf der Forderung für physikalisch realisierbare Systeme, dass die Summe der Register in einer gerichteten, geschlossenen Masche über die Ein- und Ausgänge des Signalflussgraphen größer oder gleich Null sein muss. Um diese Maschenumläufe zu ermöglichen werden Ein- und Ausgänge wieder über einen virtuellen Knoten mit einer Verzögerungszeit gleich Null verbunden. Weiterhin gilt, dass die Summe aller Register in einer gerichteten, geschlossenen Masche vor und nach dem Retiming gleich ist. Um zeitliche Nebenbedingungen berücksichtigen zu können, werden zusätzliche Kanten in den Signalflussgraphen $G =< V, E, d, w >$ eingefügt. Dadurch wird die Kantenmenge $E$ zur Kantenmenge $E_t$ ergänzt und es entsteht ein neuer Signalflussgraph $G_t =< V, E_t, d, w >$. Existieren im Graphen $G_t$ zwischen irgendwelchen Knoten $u$ und $v$ gerichtete Pfade, deren Verzögerungszeit $D(u, v)$ größer als das vorgegebene $c$ ist, so muss nach einem Retiming in diesen Pfaden mindestens ein Register zu liegen kommen. Die Anzahl der Register in diesem Pfad vor dem Retiming sei $W(u, v)$, und entsprechend nach dem Retiming $W_r(u, v)$. Die Bilanzgleichung für die Anzahl der Register in einzelnen Kanten nach (4.4) kann ebenso für die Summe der Register in einem Pfad formuliert werden.

$$W_r(u, v) = W(u, v) + r(v) - r(u) \tag{4.5}$$

Für die Anzahl der Register im Pfad von $u$ nach $v$ nach den Retiming muss gelten: $W_r(u, v) \geq 1$ falls $D(u, v) > c$. Daraus folgt

$$W(u, v) - 1 \geq r(u) - r(v) \quad \text{falls} \quad D(u, v) > c \quad . \tag{4.6}$$

D. h. dem Pfad von $u$ nach $v$ kann ein Gewicht $W(u, v) - 1$ vor dem Retiming zugeordnet werden. Dies geschieht durch Einfügen einer zusätzlichen Kante von $u$ nach $v$ mit dem Gewicht $W(u, v) - 1$, die diese zeitliche Nebenbedingung im Signalflussgraphen berücksichtigt. Zur

Überprüfung, ob eine vorgegebene Taktperiode durch Retiming erreicht werden kann, werden alle gerichteten Maschen über die Ein- und Ausgänge des Graphen $G_t$ durchlaufen und die Anzahl der Kantengewichte summiert. Ist für eine der Maschen die Summe aller Kantengewichte negativ, so ist die Taktperiode $c$ durch Retiming nicht erreichbar. In [76] wird für dieses Verfahren der Algorithmus **SMART** (**S**parse **MA**trix **R**e**T**iming) angegeben. Dabei wird gezeigt, dass die Überprüfung einer Taktperiode mit **SMART** in linearer Zeit mit der Ordnung $O(|E_t|)$ durchgeführt werden kann. Da im Allgemeinen $O(|E_t|) << O(|V||E|)$, ergibt sich gegenüber dem Algorithmus **FEAS** ein entsprechend hoher Gewinn an Rechenzeit.

## 4.2 Formulierung als klassisches Retiming Problem

Die Platzierung von Registern zur Reduzierung ungewollter Schaltaktivität soll im Folgenden nach dem Retiming-Prinzip vorgenommen werden. Dazu wird eine algorithmische Lösung des Problems vorgeschlagen. Das hier vorgestellte Verfahren ist eine modifizierte Form des Algorithmus' **FEAS**. Das Optimierungsziel ist nun nichtmehr die Minimierung der Taktperiode sondern die Minimierung der ungewollten Schaltaktivität aufgrund von Glitches.

### 4.2.1 Zielfunktion

Das Optimierungsziel beim klassischen Retiming kann als die Minimierung der maximalen Pfadverzögerung aller registerfreien Pfade aufgefasst werden. Mit der Formulierung der minimal möglichen Taktperiode $\Psi(G)$ gemäß Gleichung (4.2) ist das Optimierungsziel

$$\Psi_{\min}(p) = \min \max(d(p) : w(p) = 0) \quad . \tag{4.7}$$

Das Optimierungsziel beim Retiming einer Schaltung für minimale Glitchaktivität ist die Minimierung der Summe der Glitchaktivitäten aller Knoten in der Schaltung. Gesucht ist also eine Registerverteilung, sodass für die Summe aller Glitchgewichte in einer Schaltung gilt:

$$\sum_{v=1}^{|V|} g(v) = \min \quad . \tag{4.8}$$

Ein wesentlicher Unterschied zwischen Retiming für eine minimale Taktperiode und Retiming für minimale Glitch-Aktivität liegt in den unterschiedlichen Eigenschaften der zu optimierenden Größen. Während die Knotenverzögerungszeiten von den Registerpositionen in der Schaltung unabhängig sind und damit als eindeutig charakteristische Größen der Knoten angesehen werden können, ist dies für die Glitch-Gewichte nicht der Fall. Hierin liegt eines der Hauptprobleme bei der Minimierung der Glitch-Aktivität durch Retiming: die Glitch-Gewichte an den Knoten sind im Allgemeinen nicht unabhängig von den Registerpositionen in der Schaltung und sind somit nicht invariant gegenüber Retiming. Bild 4.8 verdeutlicht dies anhand eines Schaltungsbeispiels. Die Knoten $v_1$ bis $v_6$ enthalten Logik-Gatter deren Verzögerungszeiten als Zahlenwerte in den Knoten vermerkt sind. Durch das Verschieben der Register über die Knoten $v_2$ und $v_3$ (unteres Bild) ändern sich die Glitch-Gewichte. Die Zielfunktion der Optimierung ändert sich also mit jedem Retiming-Schritt, bei dem Register verschoben werden. Die Glitch-Gewichte der Knoten müssen deshalb nach jedem Retiming-Schritt neu bestimmt werden. Aufgrund dieser Eigenschaft besitzt Retiming mit dem Ziel der Minimierung der ungewollten Schaltaktivität eine höhere Rechenzeitkomplexität als bei der klassischen Anwendung zur Minimierung der Taktperiode. Die Rechenzeit

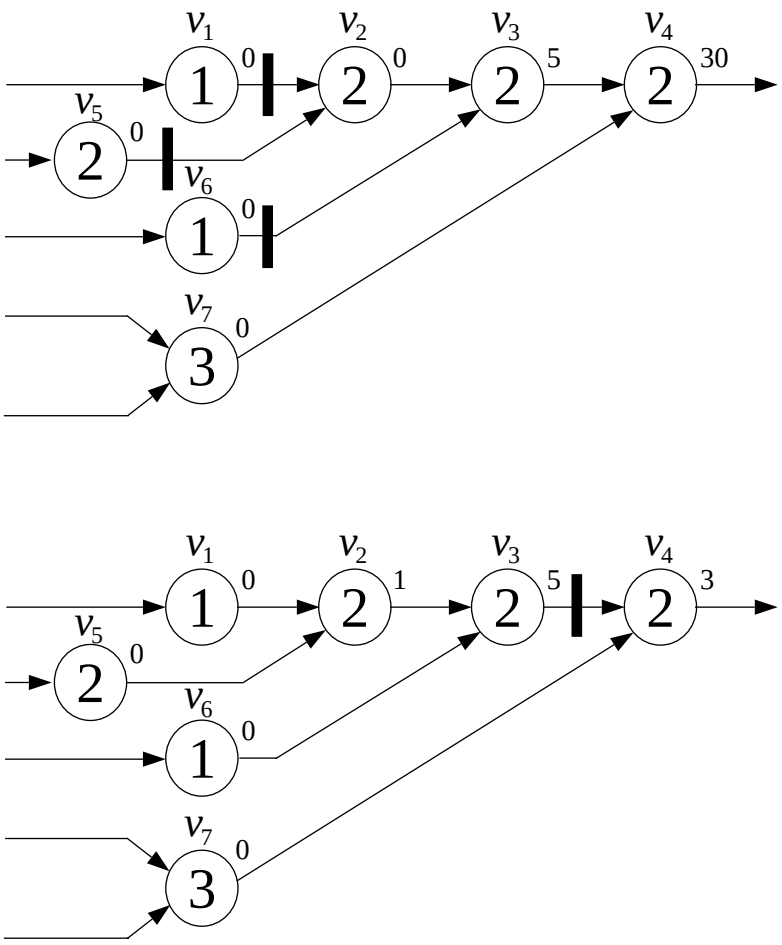



Bild 4.8. Die Glitch-Gewichte der Knoten sind von der Registerpositionierung abhängig. Die Glitch-Gewichte sind als Zahlenwerte an den Ausgängen der Knoten angegeben

für die Bestimmung der Glitch-Aktivität durch Glitch Count beispielsweise ist von der Ordnung $O(|V||X|)$, wobei $X$ hier die Menge der Testvektoren bezeichnet. Diese Rechenzeit ist in jedem Retiming-Schritt zusätzlich erforderlich.

### 4.2.2 Modifikation klassischer Retiming Verfahren

Es soll nun gezeigt werden, dass das Problem der idealen Registerplatzierung mit dem Ziel der Minimierung der Glitch-Aktivität und somit der Verlustleistung der Schaltung, als klassisches Retiming-Problem formuliert werden kann und damit den bekannten Algorithmen zur Lösung des Problems zugänglich ist. Wegen der veränderten Problemstellung beim Retiming zur Verlustleistungsreduzierung gegenüber der klassischen Anwendung, sind bestimmte Modifikationen des Verfahrens nötig. Zunächst betrifft dies die Zielfunktion, wie im vorigen Abschnitt erläutert. Die zweite notwendige Modifikation ist die Neubestimmung der Glitch-Gewichte nach jedem Retiming-Schritt und wurde ebenfalls bereits angesprochen. Neben diesen, aufgrund der anderen Zielsetzung notwendigen Modifikationen, sind die folgenden Veränderungen des Standard-Algorithmus' **FEAS** sinnvoll.

- Beim Algorithmus **FEAS** erfolgt in jedem Iterationsschritt die neue Berechnung der maximalen Pfadverzögerungen $\Delta(v)$ für jeden Knoten. Dem entspricht die Bestimmung der Glitch-Aktivität $g(v)$. Die Bestimmung von $g(v)$ erfolgt in topologischer Reihenfolge entlang eines registerfreien Pfades, dessen letzter Knoten $v$ ist. Es ist offensichtlich, dass $g(v)$ nicht in jedem Iterationsschritt für jeden Knoten neu bestimmt werden muss. Eine neue Bestimmung ist nur für solche Knoten notwendig, die – von den primären Eingängen der Schaltung her gesehen – nach einer Kante mit einem neu platzierten Register liegen, in der vor dem Retiming kein Register lag ($w(e) = 0$, $w_\mathrm{r}(e) > 0$), oder die nach einer Kante liegen, aus der ein Register entfernt wurde, sodass nunmehr kein Register in dieser Kante liegt ($w(e) > 0$, $w_\mathrm{r}(e) = 0$). Nur für solche Knoten kann sich das Glitch-Gewicht durch das Retiming überhaupt verändert haben. Die neue Bestimmung der $g(v)$ erfolgt dabei wieder in topologischer Reihenfolge bis eine Kante mit mindestens einem Register erreicht wird für die also gilt $w_\mathrm{r}(e) > 0$. Wird dabei ein Ausgang der Schaltung erreicht, so wird von allen Eingängen her mit der Neubestimmung der $g(v)$ fortgefahren bis eine Kante mit mindestens einem Register erreicht wird. Nur für diese "betroffenen" Knoten ist es auch nötig, das Glitch-Gewicht $g(v)$ neu zu bestimmen. Für alle anderen Knoten muss der Wert von $g(v)$ lediglich bis zum nächsten Iterationsschritt gespeichert werden. Dadurch ist eine große Zeitersparnis bei der Berechnung der Lösung des Retiming-Problems möglich. In der Schaltung aus Bild 4.8 beispielsweise ist nach dem Retiming im unteren Bild nur eine Neubestimmung von $g(v_2)$, $g(v_3)$, $g(v_4)$ nötig.

- Die Bedingung für das Erhöhen des Wertes $r(v)$ eines Knotens bei der klassischen Anwendung von Retiming zur Minimierung der Taktperiode lautet $\Delta(v) > c$, wobei $c$ die gewünschte Taktperiode ist. Der gewünschten Taktperiode $c$ entspricht bei der Anwendung von Retiming zur Minimierung der Glitch-Aktivität die maximal tolerierbare Glitch-Aktivität $c_\mathrm{G}$. Wie bei der klassischen Anwendung wird auch hier jedem Knoten ein Wert $r(v)$ zugewiesen, wobei gilt

$$r(v) = r(v) + 1 \quad \text{falls} \quad g(v) > c_\mathrm{G} \quad . \tag{4.9}$$

- Da mit jeder Registerumverteilung die Glitch-Gewichte der "betroffenen" Knoten neu bestimmt werden müssen, kann mit der Berechnung von $g(v)$ in einem Pfad abgebrochen werden, nachdem ein Wert $r(v)$ aufgrund $g(v) > c_\mathrm{G}$ an einem Knoten um eins erhöht wurde. Alle Werte $r$ der nachfolgenden Knoten dieses "Startknotens" $v$ werden ebenfalls um eins erhöht, solange bis eine Kante erreicht wird, die ein Register enthält ($w(e) > 0$). Wird dabei ein Ausgang der Schaltung über einen registerfreien Pfad erreicht, so wird beginnend von allen Eingängen der Schaltung der Wert $r$ aller Knoten um eins erhöht, solange bis eine Kante erreicht wird, die ein Register enthält. Wird dabei wieder der Startknoten $v$ über einen registerfreien Pfad erreicht, so ist natürlich kein Retiming der Pfade, die den Knoten $v$ enthalten, möglich, da offensichtlich keine Register zur Umverteilung zur Verfügung stehen. Durch dieses Vorgehen ist es nicht mehr nötig, die Ausgänge mit den Eingängen über einen virtuellen Knoten zu verbinden. Dennoch stellt das Verfahren sicher, dass neu zu platzierende Register von den Kanten abgezogen werden, die am nächsten zu den Eingängen der Schaltung liegen. D. h. diese Modifikation, nämlich auf den virtuellen Knoten zwischen Ein- und Ausgängen zu verzichten, wirkt sich nicht auf die Verteilung der Register aus.

Die beiden unterschiedlichen Anwendungen, Retiming zur Minimierung der Taktperiode und Retiming zur Minimierung der Glitch-Aktivität schließen einander nicht aus. Leistungs- und Geschindigkeitsoptimierung können durch die Verknüpfung der Ungleichungsbedingungen in (4.3) und (4.9) miteinander kombiniert werden.

$$r(v) = r(v) + 1 \quad \text{falls} \quad \Delta(v) > c \ \vee \ g(v) > c_{\mathrm{G}} \tag{4.10}$$

Es werden also zwei Optimierungsziele gleichzeitig angestrebt. Durch diese Verknüpfung der Bedingungen kann eine Kompromisslösung erreicht werden, d. h. eine Verbesserung der Lösung zugunsten eines Optimierungszieles hat eine Verschlechterung hinsichtlich des anderen Optimierungszieles zur Folge. Natürlich werden damit die Lösungsmengen der einzelnen Optimierungsziele eingeschränkt. In der Praxis ist es häufig wünschenswert, absolute Optimalität bezüglich eines Zieles zu erreichen und sich unter den so gegebenen Umständen dem anderen Optimierungsziel bestmöglich anzunähern. Oftmals gibt es mehrere alternative Registerverteilungen, die alle zur gleichen minimalen Taktperiode führen. Dieser Freiheitsgrad kann dazu genutzt werden, unter allen möglichen optimalen Lösungen diejenige zu wählen, bei der sich die geringste Glitch-Aktivität in der Schaltung ergibt. Die Optimierung kann dann in zwei Schritten erfolgen. Im ersten Schritt wird Retiming dazu verwendet, um die minimal mögliche Taktperiode $\Psi(G)$ zu ermitteln. Im zweiten Schritt erfolgt nun die Optimierung hinsichtlich der minimalen Glitch-Aktivität unter Einhaltung der Taktperiode $\Psi(G)$. Die Bestimmung der Werte $r(v)$ erfolgt also gemäß

$$r(v) = r(v) + 1 \quad \text{falls} \quad g(v) > c_{\mathrm{G}} \ \wedge \ \Delta(v) \leq \Psi(G) \quad . \tag{4.11}$$

In Bild 4.9 ist ein Beispiel gezeigt. Beide Schaltungen im Bild besitzen die gleiche minimale Taktperiode $\Psi(G) = 3$. In der obere Schaltung jedoch haben die Pfade $v_2 \longrightarrow v_4$ und $v_3 \longrightarrow v_4$ unterschiedliche Pfadverzögerungen, was am Ausgang des Knotens $v_4$ die Wahrscheinlichkeit für Glitches erhöht. Durch die Registerverteilung in der unteren Schaltung kann dies vermieden werden.

Für die Automatisierung der Retiming-Methode zur Minimierung der Glitch-Aktivität wird der Algorithmus **LPWR** (**L**ow **P**o**W**er **R**etiming) formuliert. Dieser Algorithmus bewirkt eine Umplatzierung der Register in einer Schaltung durch Retiming, sodass alle Glitch-Gewichte $g(v)$ kleiner oder gleich einer vorgegebenen, maximal tolerierbaren Glitch-Aktivität $c_{\mathrm{G}}$ sind, falls eine derartige Registerplatzierung gefunden werden kann. Zunächst wird dazu der Graph $G$ der Schaltung initialisiert, d. h. alle Werte $r(v)$ der Knoten werden zu Null gesetzt. Dann wird für alle Knoten in topologischer Reihenfolge $g(v)$ ermittelt. Dabei werden zwei Fälle unterschieden:

1) Falls noch kein Retiming erfolgt ist, der Graph also noch wie bei der Initialisierung vorliegt, werden beginnend an den Eingängen der Schaltung die Werte $g(v)$ in topologischer Reihenfolge ermittelt.
2) Falls bereits ein Retiming-Schritt erfolgt ist, so werden die Glitch-Aktivitäten $g(v)$ nur für jene Knoten neu bestimmt, bei denen sich diese aufgrund einer Registerumplatzierung verändert haben können.

Falls $g(v)$ größer als die maximal tolerierbare Glitch-Aktivität ist, so erfolgt ein Retiming-Schritt mit der entsprechenden Zuweisung der Werte $r(v)$ und der Bestimmung der neuen Registerpositionen gemäß Gleichung (4.4). Da die in der Schaltung neu platzierten Register jeweils von den Kanten abgezogen werden, die am nächsten zu den Eingängen der Schaltung

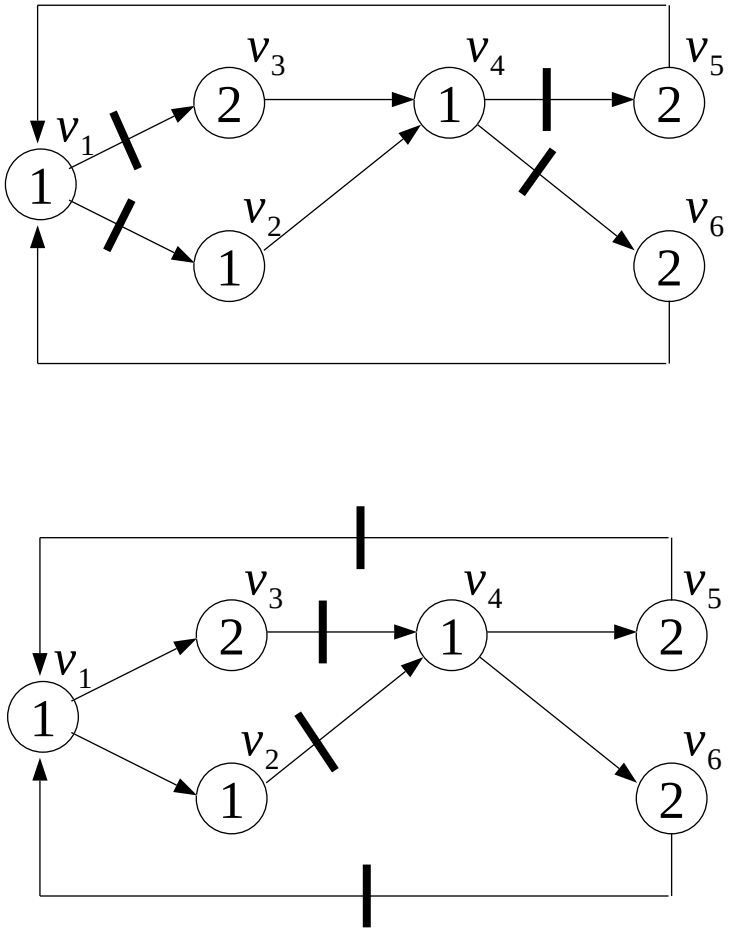



Bild 4.9. Minimierung der Glitch-Aktivität unter Beibehaltung der minimalen Taktperiode. Für beide Schaltungen ist $\Psi(G) = 3$, in der unteren Schaltung sind die ungleich langen Pfade $v_2 \longrightarrow v_4$ und $v_3 \longrightarrow v_4$ durch Register synchronisiert und dadurch die Wahrscheinlichkeit für Glitches am Ausgang von Knoten $v_4$ minimiert.

liegen, ist es möglich, dass nach einem Retiming mit Registerumplatzierung registerfreie Pfade $p$ existieren, die an den Eingängen der Schaltung entspringen, wobei für mindestens einen Knoten $v \in p$ gilt $g(v) > c_{\mathrm{G}}$. Dies führt dazu, dass im nächsten Schritt in diesem Pfad wieder ein Register platziert wird. Dieses Register wird von den Kanten mit $w(e) > 0$ abgezogen, die am nächsten zum Ausgang der Schaltung liegen. Wurden in diesen Kanten jedoch erst im vorhergehenden Schritt Register platziert, so wird durch den Abzug der Register die geforderte Reduzierung der Glitch-Aktivität wieder zerstört. Die Folge davon ist eine Oszillation des Algorithmus', bei der die Register alternierend zwischen zwei Positionen umplatziert werden. Um dies zu verhindern, werden Kanten, in denen bereits einmal Register zur Reduzierung der Glitch-Aktivität platziert wurden, mit einer Marke $m(e) = 1$ versehen. Ergibt sich für eine markierte Kante nach einem Retiming-Schritt $w_{\mathrm{r}}(e) = 0$, d. h. dass kein Register mehr in der Kante liegt, so wird der Algorithmus abgebrochen, $c_{\mathrm{G}}$ kann nicht eingehalten werden. Ansonsten terminiert der Algorithmus, wenn alle Knoten in topologischer Reihenfolge abgearbeitet wurden und gilt: $\forall v \in V : g(v) \leq c_{\mathrm{G}}$.

**Algorithmus LPWR:** Bestimmung der Werte $r(v)$ für eine vorgegebene maximal tolerierbare Glitch-Aktivität $c_{\mathrm{G}}$ falls eine Lösung gefunden werden kann.

**BEGINN**
Nummeriere alle Knoten in $G$ in topologischer Reihenfolge.
Initialisiere: $\forall e \in E$, setze $m(e) = 0$.
**FÜR** $v = 1$ **BIS** $|V|$:
$\forall v \in V$, setze $r(v) = 0$
Bestimme $g(v)$
**FALLS** $g(v) > c_{\mathrm{G}}$:
Setze $r(v) = r(v) + 1$ für den Knoten $v$ und alle Nachfolgerknoten bis zu Kanten mit $w(e) > 0$. Falls dabei ein Ausgang der Schaltung erreicht wird, beginne bei allen Eingängen und setze in topologischer Reihenfolge alle $r(v) = r(v) + 1$ bis zu Kanten mit $w(e) > 0$.
**ENDE** der 1. Abfrage
**FÜR** $e = 1$ **BIS** $|E|$
$w_{\mathrm{r}}(e) = w(e) + r(v) - r(u)$ .
**FALLS** ($w(e) = 0$ und $w_{\mathrm{r}}(e) > 0$):
$m(e) = 1$.
**ENDE** der 2. Abfrage
**FALLS** ($w(e) > 0$ und $w_{\mathrm{r}}(e) = 0$ und $m(e) = 1$ ):
Abbruch von Algorithmus **LPWR**.
**ENDE** der 3. Abfrage
**ENDE** der Schleife
$w(e) = w_r(e)$ .
**ENDE** der Schleife
**ENDE**

Der in dieser Form angegebene Algorithmus **LPWR** beinhaltet nur die Optimierung bezüglich der Glitch-Aktivität. Für die Kombination mit der Minimierung der Taktperiode muss die 2. Abfrage entsprechend um die Bedingung $\Delta(v) > c$ erweitert werden. Zur Verdeutlichung der Arbeitsweise des Algorithmus' ist in Bild 4.10 ein Beispiel für das Retiming einer Schaltung angegeben.

## 4.3 Minimierung der ungewollten Schaltaktivität durch Retiming

Im vorangegangenen Abschnitt wurde das maximal tolerierbare Glitch-Gewicht $c_{\mathrm{G}}$ fest vorgegeben. Im Folgenden soll nun der Algorithmus **LPWR** in ein Optimierungsverfahren eingebunden werden, welches das minimal mögliche Glitch-Gewicht zu erreichen versucht. Analog zu den Pfadverzögerungszeiten $D(u, v)$ zwischen Knoten $u$ und $v$ kann nun ein maximales Glitch-Gewicht $D_{\mathrm{G}}(u, v)$ aller Pfade zwischen den Knoten $u$ und $v$ definiert werden. Als Startwert für die zu überprüfende Glitch-Aktivität $c_{\mathrm{G}}$ wird $\max\{D_{\mathrm{G}}(u, v)\}$ gewählt. Der Algorithmus wird abgebrochen, sobald $c_{\mathrm{G}}$ um weniger als eine vorgegebene Schranke $\varepsilon$ von der tatsächlichen minimal möglichen Glitch-Aktivität abweicht. Zur Automatisierung der Optimierung einer Schaltung hinsichtlich minimaler Glitch-Aktivität durch Retiming wird der folgende Algorithmus **OPTPWR** (**OPT**imize **P**o**W**er by **R**etiming) formuliert.

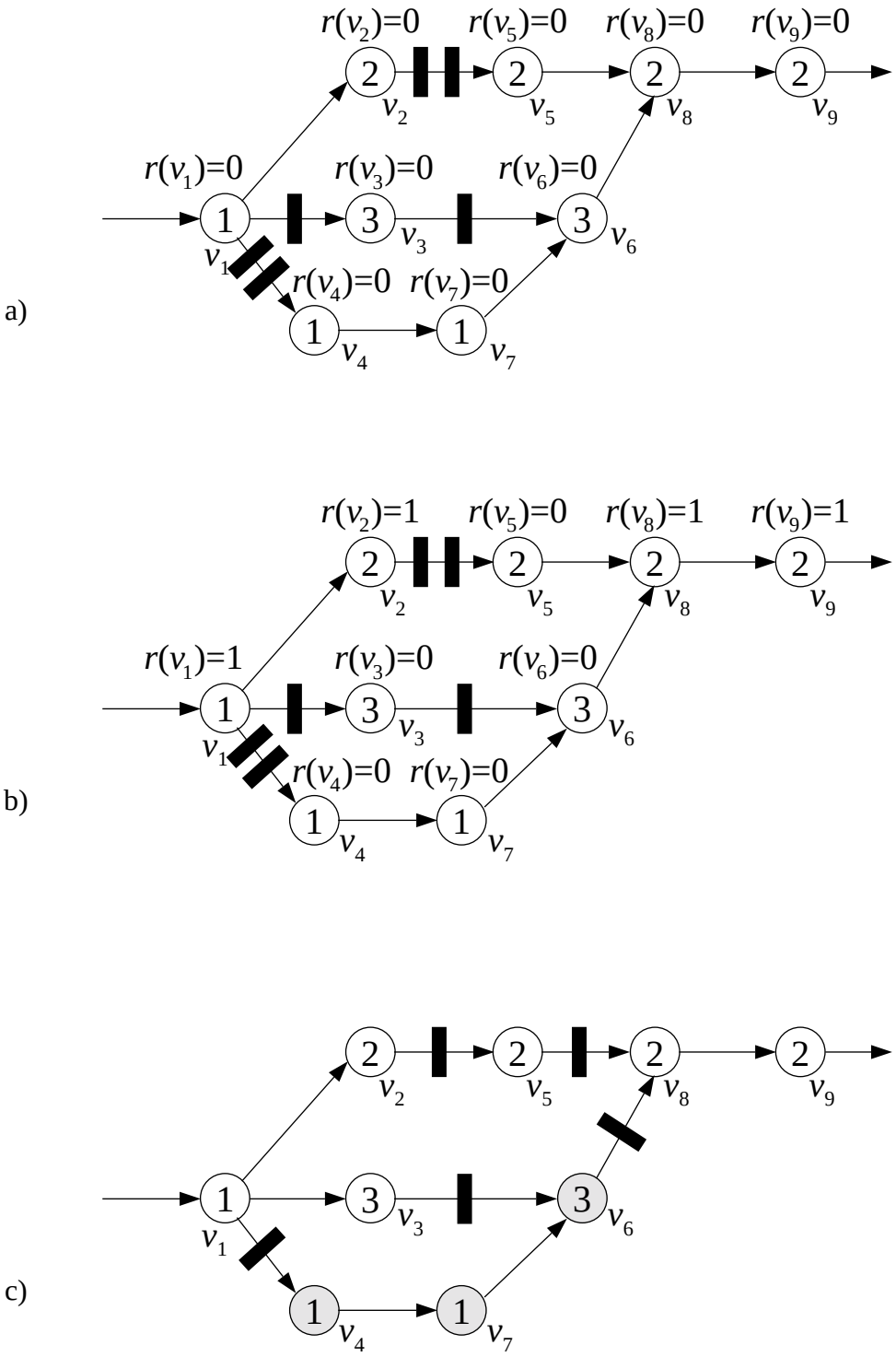



Bild 4.10. Beispiel für den Retiming mit dem Algorithmus **LPWR**. a) Initialisierung: setze alle $r(v_i) = 0$. Bestimme $g(v_i)$. b) Es sei $g(v_8) > c_G$. Setze $r(v_8 = 1$. Erhöhe $r(v_i)$ für alle nachfolgenden Knoten um eins, somit $r(v_9) = 1$. Hier ist der Ausgang der Schaltung erreicht. Beginne am Eingang und erhöhe alle $r(v_i)$ um eins bis zu Kanten, die Register enthalten, somit $r(v_1) = 1$, $r(v_2) = 1$. c) Bestimme die neuen Registerpositionen mit $w_r(e) = w(e) + r(v) - r(u)$ und markiere die Kanten $v_5 \longrightarrow v_8$ und $v_6 \longrightarrow v_8$. Im nächsten Schritt müssen die Glitch-Gewichte für die im Bild grau hervorgehobenen Knoten $v_4$, $v_6$ und $v_7$ nicht neu bestimmt werden.

**Algorithmus OPTPWR:** Minimierung der Glitch-Aktivität durch Retiming. Innerhalb einer Toleranz $\varepsilon$.

**BEGINN**
$g_{\mathrm{ges},i} = \infty$
Bestimme $D_{\mathrm{G}}(u,v)$ für alle $(u,v)$.
Setze $c_{\mathrm{G}} = \max\{D_{\mathrm{G}}(u,v)\}$, $c_{\mathrm{u}} = 0$, $c_{\mathrm{o}} = c_{\mathrm{G}}$.
**WIEDERHOLE** bis $c_{\mathrm{o}} - c_{\mathrm{u}} \leq \varepsilon$:
Starte Algorithmus **LPWR** mit $c_{\mathrm{G}}$ als maximal tolerierbares Glitch-Gewicht.
**FALLS** für $c_{\mathrm{G}}$ ein gültiges Retiming existiert:
Bestimme $g_{\mathrm{ges},i} = \sum_{v=1}^{|V|} g(v)$.
**FALLS** $g_{\mathrm{ges},i} > g_{\mathrm{ges},i-1}$:
Beende **OPTPWR**.
$c_{\mathrm{o}} = c_{\mathrm{G}}$.
$c_{\mathrm{G}} = \frac{c_{\mathrm{G}}+c_{\mathrm{u}}}{2}$.
**SONST**
$c_{\mathrm{u}} = c_{\mathrm{G}}$.
$c_{\mathrm{G}} = \frac{c_{\mathrm{G}}+c_{\mathrm{o}}}{2}$.
**ENDE** der Abfrage
Setze $g_{\mathrm{ges},i-1} := g_{\mathrm{ges},i}$.
**ENDE** der Schleife
**ENDE**

Wie bereits angesprochen, hängt das Glitch-Gewicht eines Knotens von den Registerpositionen in der Schaltung ab. Aufgrund dieser Eigenschaft kann im allgemeinen nicht garantiert werden, dass mit dem hier vorgestellten Retiming-Verfahren die tatsächlich optimale Registerpositionierung für minimale Glitch-Aktivität gefunden wird. Ausserdem muss nach dem Retiming überprüft werden, ob die gesamte Glitch-Aktivität in der Schaltung nach dem Retiming tatsächlich geringer ist als zuvor. Diese Überprüfung erfolgt im Algorithmus **OPTPWR** immer nachdem für ein bestimmtes $c_{\mathrm{G}}$ ein gültiges Retiming gefunden wurde. Hat sich die Glitch-Aktivität insgesamt erhöht, so bricht der Algorithmus ab. Als Lösung wird das vorherige Retiming genommen. In allen untersuchten Schaltungen ergab sich mit jedem gültigen Retiming immer eine Verringerung der gesamten Glitch-Aktivität und daher eine Verringerung der Verlustleistung.

Das Schaltungsbeispiel in Bild 4.11 verdeutlicht die Glitch-Minimierung mittels Retiming. Der kombinatorische Teil entspricht der ISCAS Benchmark-Schaltung C17. Die maximale Glitch-Aktivität an den Ausgängen der Gatter in der nichtoptimierten Schaltung beträgt 7% und wurde mit der Methode **GC** ermittelt. Das Ziel sei nun, die Register so in der Schaltung zu platzieren, dass die Glitch-Aktivitäten an allen Ausgängen der Gatter zu Null werden. Die vorgegebene maximal tolerierbare Glitch-Aktivität ist somit $c_{\mathrm{o}} = 0$. Nach dem ersten Retiming-Schritt wird eine Stufe von Registern in der Schaltung platziert. Die maximale Glitch-Aktivität wird dadurch auf 5% reduziert. Nach einem zweiten Retiming Schritt schließlich wird eine Registerpositionierung erreicht, bei der die Pfade zu den Eingängen aller Gatter die gleiche Länge besitzen. Die Glitch-Aktivität in der Schaltung ist

somit vollständig eliminiert. Natürlich muss auch der zusätzliche Leistungsverbrauch, den

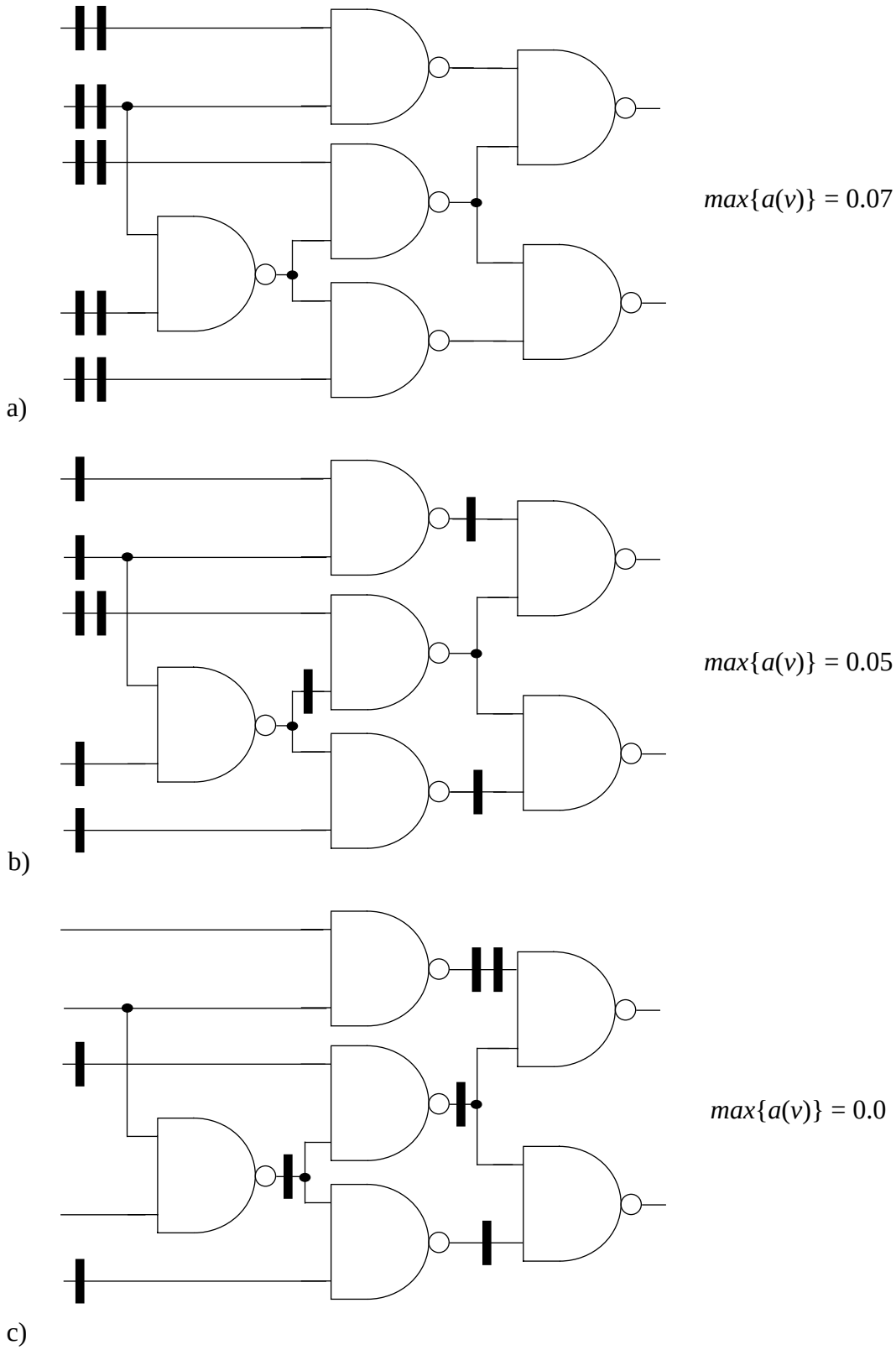



Bild 4.11. Retiming einer Schaltung mit **OPTPWR**. a) Ursprüngliche Schaltung. b) Nach dem ersten Retiming-Schritt. c) Nach dem zweiten Retiming-Schritt.

die Register selbst verursachen, in der Leistungsbilanz berücksichtigt werden. Wird Retiming im eigentlichen Sinne angewandt, d. h. es werden keine zusätzlichen Register künstlich in die Schaltung eingefügt, ist es dennoch möglich, dass die Summe der Register in der Schaltung vor und nach dem Retiming unterschiedlich ist. Dies ist zum Beispiel für jeden Retiming-Schritt in Bild 4.11 der Fall, wobei sich hier die Zahl der Register verringert, was sich positiv auf den Leistungsverbrauch auswirkt. Je nach Schaltung kann die Zahl der Register nach dem Retiming aber auch ansteigen. Im allgemeinen wird sich jedoch die Anzahl der Register vor und nach dem Retiming einer Schaltung weniger gravierend unterscheiden, als im Fall des Pipelinings, da dabei zusätzliche Register von aussen in die Schaltung gebracht werden. Hinzu kommt die stärkere Belastung des Taktsystems. Retiming und Pipelining zur Reduzierung von Glitches in einer Schaltung sind natürlich nur dann sinnvoll, wenn der zusätzliche Leistungsverbrauch der Register und des Taktsystems den Gewinn aus der Reduzierung der Glitches nicht übersteigt. Eine Möglichkeit, den zusätzlichen Leistungsverbrauch zu reduzieren, besteht in der Verwendung von Latches anstatt von Registern als getaktete Elemente. Wenn möglich sollten Register immer in Latches aufgespalten und diese getrennt verschoben werden, um eventuell Inverter einsparen zu können, wie dies in Abschnitt 4.1.1 gezeigt wurde.

Im Folgenden sind einige Ergebnisse angegeben. Die Algorithmen wurden dazu als C-Programme implementiert. In Tabelle 4.1 sind die Ergebnisse für einige ISCAS Benchmark-Schaltungen aufgelistet. Da es sich hierbei zunächst um rein kombinatorische Schaltungen handelt, wurden die Register vor dem Start des Optimierungsverfahrens gleichmäßig an den Eingängen der Schaltungen verteilt, d. h. gleich viele Register an jedem Eingang platziert. Es wurde versucht, die maximale Glitch-Aktivität $max\{a(v)\}$ an den Ausgängen der Gatter in den Schaltungen auf höchsten 5% zu drücken, d. h. bei 100 Schaltvorgängen am Ausgang eines Knotens durften maximal 5 ungewollte Schaltvorgänge auftreten. Die dazu jeweils nach dem Retiming nötige Registeranzahl ist in der vierten Spalte der Tabelle angegeben. Die Bestimmung der Glitch-Gewichte wurde mit der Methode **GC** mit jeweils 1000 Testvektoren durchgeführt. Die Tabelle zeigt die Summe der Glitch-Aktivitäten an den Knoten vor dem Retiming mit **OPTPWR** und die Summe der Glitch-Aktivitäten nach der Optimierung. Desweiteren sind die Prozessorzeiten für den Algorithmus **OPTPWR** auf einer SUN Ultra 10 Workstation angegeben.

| Schaltung | Anzahl der Gatter | Register vor Retiming | Register nach Retiming | Rechenzeit in Sekunden | $\sum_{v=1}^{\|V\|} g(v)$ vor Retiming | $\sum_{v=1}^{\|V\|} g(v)$ nach Retiming |
|---|---|---|---|---|---|---|
| sct | 91 | 19 | 26 | 2.20 | 0.44 | 0.15 |
| x4 | 389 | 282 | 108 | 3.92 | 2.38 | 0.44 |
| i6 | 340 | 138 | 404 | 20.13 | 17.64 | 0.00 |
| cmb | 41 | 32 | 22 | 0.80 | 0.21 | 0.10 |
| C17 | 6 | 5 | 4 | 0.02 | 0.20 | 0.17 |
| C1355 | 546 | 410 | 375 | 9384 | 50.47 | 31.21 |

Tabelle 4.1. Vergleich der maximalen Glitch-Aktivität an den Ausgängen der Gatter in ISCAS Benchmark-Schaltungen vor und nach Optimierung mit **OPTPWR**.

Tabelle 4.2 zeigt die Simulationsergebnisse hinsichtlich des Leistungsverbrauchs von vier verschiedenen Datenpfadkomponenten: einen $8 \times 8$ Wallace-Tree, einen $5 \times 5$ Brown-

Multiplizierer und zwei verschiedene Implementierungen einer Kaskade von drei CORDIC-Stufen, einmal in Carry-Ripple- und einmal in Carry-Save-Architektur. Die einzelnen Stufen bewirken Drehungen des Datenvektors um $45^o$, $26.57^o$ und $14.04^o$. Die Tabelle zeigt die Ergebnisse für die rein kombinatorischen und für die mit ein und zwei Pipelinestufen versehenen Versionen der Schaltungen. Dabei wurden anstatt von Registern einfache dynamische Latches als getaktete Elemente verwendet. Die Latches wurden vor Beginn der Optimierung an den Eingängen der Schaltungen platziert. Der zusätzlich durch die Latches verursachte Leistungsverbrauch wurde in den Simulationen ebenfalls berücksichtigt. Die Inverter der Latches wurden nach Möglichkeit weggelassen, sobald dies ohne Beeinträchtigung der Funktion der Schaltung möglich war (siehe Abschnitt 4.1.1 Bild 4.3). Die ursprünglichen, rein kombinatorischen Schaltungen wurden als Layouts in $1\mu m$ Technologie implementiert und anschließend unter Berücksichtigung aller parasitären Kapazitäten mit SPICE simuliert. Auf gleiche Weise wurden die durch Optimierung mit **OPTPWR** erhaltenen Graphen in Form von Layouts der zugehörigen Schaltungen realisiert und simuliert. An den Ergebnissen ist auffällig, dass die größte Leistungseinsparung bereits durch die Einführung einer Pipeline-Stufe bewirkt wird. Beim $8 \times 8$ Wallace-Tree steigt sogar die Verlustleistung bei zwei Pipeline-Stufen wieder etwas an. Der Grund dafür ist der zusätzliche Leistungsverbrauch der Latches, der den Gewinn durch die Reduzierung der Glitch-Aktivität in diesem Fall übersteigt. Dies liegt an der speziellen Wallace-Tree-Architektur. Im Gegensatz zum schichtweisen Aufbau eines Brown-Multiplizierers mit seiner typischen Addiereranordnung in einem regulären Feld, weist die Wallace-Tree-Architektur eine weniger reguläre Baumstruktur auf. Die höheren Bits der Zwischenergebnisse aus den Additionen in den verschiedenen Stufen des Wallace-Trees werden teilweise nicht zur unmittelbar nächsten, sondern gegebenenfalls über mehrere Stufen weitergereicht. Daher ergibt sich eine stark unterschiedliche logische Tiefe für nieder- und höherwertige Bits. Bei der Optimierung der Schaltung durch Retiming oder Pipelining kommen häufig Latches in den Leitungen der höherwertigen Bits hintereinander zu liegen, sodass die Inverter nicht eingespart werden können. Dies wirkt sich negativ auf den Leistungsverbrauch aus, wie die in Tabelle 4.2 für den $8 \times 8$ Wallace-Tree mit zwei Pipeline-Stufen dokumentiert ist.

Das vorgestellte Verfahren zum Retiming und Pipelining von Schaltungen zur Reduzierung der ungewollten Schaltaktivität durch Glitches wurde hier anhand von Beispielschaltungen mit azyklischen Signalflussgraphen dargelegt. Grundsätzlich ist das Verfahren auch für zyklische Graphen rückgekoppelter Schaltungen anwendbar. In physikalisch realisierbaren Schaltungen muss eine geschlossene Schleife mindestens ein Register enthalten. Für die Anwendung des Algorithmus' **LPWR** werden alle geschlossenen Schleifen des Graphen an jeweils einer Kante, die ein Register enthält, aufgetrennt. Das zum Register führende Stück der durchtrennten Kante wird als Ausgang, das vom Register wegführende Stück als Eingang der Schaltung betrachtet. Das Register selbst wird an diesen neuen Eingang gelegt. Somit ist der Graph wieder azyklisch und die Methode kann wie beschrieben angewendet werden. Natürlich müssen die durch das Auftrennen von Schleifen entstandenen Ein- und Ausgänge bei der eigentlichen Realisierung der Schaltung wieder verbunden werden.

Das Pipelining von geschlossenen Schleifen ist jedoch nicht möglich. Dies folgt aus der Forderung, dass die Anzahl der Register in einer Schleife invariant gegenüber Retiming ist. Um jedoch die Anzahl der in einer Masche zur Verfügung stehenden Register erhöhen zu können und damit mehr Möglichkeiten zur Reduzierung der Glitches zu haben, kann die Methode des sog. *Slowdown* angewendet werden. Leiserson und Saxe stellen diese Methode

| Schaltung | Latch-Stufen | Latches insgesamt | kap. Schaltleistung pro Eingangsübergang $mW$ |
|---|---|---|---|
| 8x8 Wallace- Tree | 0 | 0 | 3.169 |
| | 1 | 39 | 2.699 |
| | 2 | 57 | 2.789 |
| 5x5 Brown-Multiplizierer | 0 | 0 | 1.203 |
| | 1 | 17 | 0.914 |
| | 2 | 30 | 0.904 |
| CORDIC (Carry-Ripple) | 0 | 0 | 1.338 |
| | 1 | 8 | 1.085 |
| | 2 | 16 | 0.967 |
| CORDIC (Carry-Save) | 0 | 0 | 2.549 |
| | 1 | 18 | 2.082 |
| | 2 | 34 | 2.045 |

Tabelle 4.2. Kapazitive Schaltleistung von Datenpfad-Schaltungen mit unterschiedlich vielen Pipeline-Stufen. In den angegebenen Werten für die Verlustleistung pro Schaltvorgang ist auch die Verlustleistung der Latches enthalten.

in [41] vor. Dabei kann die Zahl der Register in einer Schleife um das $n$-fache erhöht werden, wenn alle Daten um den Faktor $n$ "verlangsamt" werden, d. h. bei gleichbleibender Taktrate muss jedes Datenwort $n$ Taktperioden lang am Eingang der Schleife anliegen. In [89] wird diese Methode verwendet, um die Zahl der zur Verfügung stehenden Register in Schleifen zu erhöhen und damit die Verlustleistung durch Glitches zu reduzieren. Die Automatisierung dieses Verfahrens zur Verlustleistungsreduzierung in Schaltungen, die Rückkopplungen enthalten, stellt eine Aufgabe für künftige Forschungsaktivitäten dar.

# 5. Optimale Transistordimensionierung zur Reduzierung der Schaltaktivität in digitalen Schaltungen

In diesem Kapitel wird eine neue Methode vorgestellt, durch die unterschiedliche Pfadverzögerungen in digitalen CMOS-Schaltungen auf Schaltkreisebene mittels Transistordimensionierung ausgeglichen werden können [68, 69, 88]. Dadurch gelingt es, die durch unterschiedliche Pfadverzögerungen verursachte ungewollte Schaltaktivität zu reduzieren. Wie anhand einiger Beispiele gezeigt wird, führt dieses Verfahren zur Verringerung der Verlustleistung digitaler Schaltungen. Wesentlich bei diesem Verfahren ist, dass die kritischen Pfade der Schaltung unverändert bleiben. Es werden lediglich "zu schnelle" Pfade verlangsamt. Dadurch bleibt die Geschwindigkeit der Schaltung insgesamt unbeeinflusst. Das Ausgleichen unterschiedlicher Pfadverzögerungen wurde bereits in [57] durch die Einführung zusätzlicher Verzögerungselemente in die unkritischen Pfade von kombinatorischen Array- und Wallace-Tree-Multiplizierern als Methode zur Minimierung von Glitches verwendet. Dabei kann jedoch der Leistungsverbrauch durch die zusätzlichen Verzögerungselemente (Buffer) erheblich sein. In [84] werden nach der Geschwindigkeitsoptimierung einer Schaltung die Transistorweiten in allen unkritischen Pfaden reduziert, um so die Gesamtkapazität der Schaltung zu verringern. Die Verringerung der Transistorweiten garantiert jedoch nicht das Ausgleichen unterschiedlicher Verzögerungszeiten und damit die Eliminierung von Glitches. Die Geschwindigkeit eines CMOS-Gatters hängt ausser von der Kanalbreite $W$ auch von der Kanallänge $L$ ab. In dem hier vorgestellten Verfahren wird das Balancieren von Pfadverzögerungen als ein Optimierungsproblem beschrieben, bei dem sowohl die Kanalweiten als auch die Kanallängen der Transistoren als Variablen betrachtet werden. Eine wichtige Aufgabe stellt die geeignete Modellierung der Schaltung dar. Insbesondere müssen dabei die Pfadverzögerungen als Funktionen der Transistordimensionen $W$ und $L$ dargestellt werden. Durch die Variation von $W$ und $L$ ändert sich jedoch auch die Eingangskapazität der Transistoren und damit auch die kapazitive Schaltleistung der gesamten Schaltung. Es besteht somit die Gefahr, dass eine durch Transistordimensionierung erhöhte kapazitive Schaltleistung die Leistungsreduzierung durch die Verringerung der ungewollten Schaltaktivität aufwiegt oder sogar übersteigt. Um dies zu verhindern, ist es bei diesem Verfahren notwendig, gleichzeitig mit den Pfadverzögerungen auch die Veränderung der kapazitiven Schaltleistung zu berücksichtigen. Bei der Modellierung muss deshalb auch die kapazitive Schaltleistung als Funktion der Transistordimensionen $W$ und $L$ dargestellt werden. Für die Behandlung solcher Probleme, bei denen zwei oder mehrere, unter Umständen widersprüchliche Optimierungsziele gleichzeitig betrachtet werden sollen, stellt die Mathematik so genannte *Mehrziel-* oder *Vek-*

*toroptimierungsverfahren* zur Verfügung. Die Optimierung von Transistordimensionen wurde in früher publizierten Arbeiten als Methode zur Erhöhung der Schaltgeschwindigkeit angewendet [8, 9, 25, 27, 28]. Als Nebenbedingungen werden dabei zumeist die Verlustleistung und/oder die Chipfläche betrachtet. In [28] und [29] wird ein Mehrzieloptimierungsverfahren zur Minimierung der Verzögerungszeit unter Berücksichtigung der Chipfläche und der kapazitiven Schaltleistung präsentiert. In dem hier vorgestellten Verfahren wird die Fläche bei der Optimierung zunächst nicht berücksichtigt. Eine Erweiterung des Verfahrens im Hinblick auf die Chipfläche ist jedoch auf einfache Weise möglich.

Das Verfahren wird in mehreren Schritten vorgestellt. Zunächst wird die Modellierung der Pfadverzögerungen und der kapazitiven Schaltleistung behandelt. Im weiteren folgt dann die Formulierung als Mehrzieloptimierungsproblem, und schließlich die Einbettung des Optimierungsverfahrens in einen Algorithmus zur automatischen Optimierung von Schaltungen hinsichtlich minimaler Verlustleistung durch Transistordimensionierung.

## 5.1 Modellierung der Verzögerungszeit und der Leistungsaufnahme von CMOS-Logikgattern

Für die Formulierung der Eigenschaften von Schaltungen in Abhängigkeit von Bauteilgrößen ist es zunächst naheliegend, eine Modellierung auf der Bauteilebene zu wählen. In den mathematischen Darstellungen der Verzögerungszeiten entlang der Pfade und der Verlustleistung der Schaltung kommen dann die $W$- und $L$-Werte aller Transistoren als Variablen vor. Die Ermittlung der günstigsten Werte für $W$ und $L$ stellt somit ein hochdimensionales Optimierungsproblem dar. Die Berechnungszeit für die Lösung steigt mit der zunehmenden Dimension des Problems. Zudem ist die individuelle Manipulation der einzelnen Transistoren nur bei einem vollständig manuellen Entwurf durchführbar. Obwohl das Verfahren selbst eine Modifikation auf Schaltkreisebene bewirkt, ist es für die Reduzierung der Problemdimension und für die Beibehaltung einer modularen Schaltungsstruktur dennoch sinnvoll, bei der Modellierung und der darauf basierenden Optimierung auf einer höheren Abstraktionsebene anzusetzen. Das hier beschriebene Verfahren basiert auf einer Modellierung auf Gatterebene. Man spricht in diesem Fall allgemein von einer sog. *Makromodellierung*. Dabei reduziert sich die große Anzahl lokaler Parameter für die einzelnen Transistoren auf einen Satz von repräsentativen Parametern für jedes Gatter. In dem hier diskutierten Fall beschränkt sich die Zahl der veränderlichen Parameter für jedes Gatter auf jeweils einen charakteristischen Wert für $W$ und $L$. Falls $W$ oder $L$ eines Gatters verändert werden, so ändern sich entsprechend alle Werte von $W$ und $L$ der einzelnen Transistoren in diesem Gatter um den gleichen Faktor. Im Folgenden werden die Modelle für die Verzögerungszeit und den Leistungsverbrauch auf Gatterebene detailliert behandelt.

### 5.1.1 Verzögerungszeitmodell

Die Verzögerungszeit eines einzelnen Gatters kann in zwei Anteile aufgespalten werden. Der erste Teil entspricht der Verzögerungszeit der Sprungantwort und ist von der Form des Eingangssignals unabhängig. Der zweite Teil berücksichtigt die Abhängigkeit der Verzögerungszeit von der endlichen Steilheit der Flanken des Eingangssignals. Betrachtet man einen Pfad mit mehreren Gattern, so kann im Allgemeinen jedes Gatter eine andere Verzögerungszeit aufweisen. Für ein Gatter an der Position $m$ gilt entsprechend eine Verzögerungszeit $\tau_m$.

Die Verzögerung der Sprungantwort wird mit $\tau_{s,m}$ bezeichnet, die von der Flankensteilheit der Eingangssignale abhängige Verzögerung mit $\tau_{in,m}$. Die Verzögerungszeit des Gatters ist somit

$$\tau_m = \tau_{s,m} + \tau_{in,m} \quad . \tag{5.1}$$

Die Nummerierung der Gatter in einem Pfad erfolgt in topologischer Reihenfolge. Die Anstiegs- bzw. Abfallzeit des Eingangssignals von Gatter $m$ ist damit gleich der Anstiegs- bzw. Abfallszeit des Ausgangssignals von Gatter $m-1$, $\tau_{r/f,m-1}$. Aus den oben genannten Definitionen folgt sofort, dass $\tau_m = \tau_{s,m}$ falls $\tau_{r/f,m-1} = 0$.

Die Verzögerung der Sprungantwort ist davon abhängig, welcher Übergang der Signale am Eingang auftritt. Mit Hilfe eines Beispiels kann dies einfach gezeigt werden: Für ein UND-Gatter in $0.25\mu m$-Technologie mit zwei Eingängen und einer Ausgangslast von $5fF$ ergibt sich beim Eingangsübergang $11 \longrightarrow 00$ eine Verzögerungszeit von $0.4ns$. Die Verzögerungszeit beim Eingangsübergang $00 \longrightarrow 11$ hingegen beträgt $0.75ns$. Dies stellt ein Problem bei der Modellierung der Gatter dar, da diese in mathematisch geschlossener Form und unabhängig von den Daten erfolgen soll. Das Ziel ist, durch ein Optimierungsverfahren die unterschiedlichen Verzögerungen der Signalpfade in einer Schaltung so auszugleichen, dass die Möglichkeit für das Auftreten von Glitches minimiert wird. Dies kann offensichtlich nur für bestimmte Signalwechsel exakt erreicht werden. In einem datenunabhängigen mathematischen Modell für die Verzögerungszeit eines Gatters kann nur ein bestimmter Fall einer Pfadsensibilisierung berücksichtigt werden. Nur für den Signalwechsel, bei dem der im Modell berücksichtigte Fall auftritt, kann auch die Verzögerungszeit durch Transistordimensionierung genau eingestellt werden. Ist die Signalstatistik bekannt, d. h. mit welcher Wahrscheinlichkeit bestimmte Übergänge an den Eingängen auftreten, so ist es sinnvoll, dies im Modell zu berücksichtigen. Es sollten dann die Signalpfade modelliert mit wahrscheinlichstem Übergang sensibilisiert werden. Falls über die Signalstatistik a priori keine Aussagen gemacht werden können, liegt es nahe zu versuchen, die durchschnittlichen Verzögerungszeiten der verschiedenen Signalpfade aneinander anzugleichen. In diesem Fall muss der Übergang modelliert werden, dessen Verzögerungszeit am nächsten zur durchschnittlichen Verzögerungszeit des Gatters liegt. In der Praxis erweist es sich jedoch als sinnvoll, die Worst-Case-Verzögerungszeiten der Gatter im Modell zu verwenden. Der Worst-Case kann mit einem $RC$-Verzögerungszeitmodell einfach und hinreichend genau ermittelt werden. Wie die späteren Experimente zeigen, kann durch den Angleich der Worst-Case-Verzögerungszeiten von Pfaden, die in einem Knoten zusammenlaufen, die ungewollte Schaltaktivität erheblich verringert werden. Im Vergleich dazu wurden durch den Angleich der durchschnittlichen Verzögerungszeiten in Experimenten schlechtere Ergebnisse erzielt.

Die Verzögerung der Sprungantwort $\tau_{s,m}$ wird im Folgenden mit dem Elmore-Delay-Modell [22, 50, 56] beschrieben. Die Modellbeschreibung erfolgt am Beispiel eines UND-Gatters, wie es in Bild 5.1 dargestellt ist. Für das Elmore-Delay-Modell werden die Transistoren der Schaltung durch lineare Kapazitäten und Widerstände ersetzt. Der Worst-Case tritt auf, wenn die parasitären Kapazitäten der Schaltung und die Lastkapazität nach dem Wechsel des Eingangszustandes umgeladen werden müssen. Im vorliegenden Beispiel erfolgt ein vollständiges Aufladen der inneren Knoten bei der Eingangsbelegung $A = 0$, $B = 0$. In diesem Fall werden die Drain-Source Kapazitäten am Knoten 1 auf die Betriebsspannung $U_B$ geladen. Die Lastkapazität und die Drain-Source Kapazitäten am Knoten 3 werden vollständig gegen Masse entladen, am Ausgang liegt eine logische "0". Es sei ausserdem angenommen, dass durch eine vorhergehende Eingangsbelegung $A = 0$, $B = 1$ die Drain-

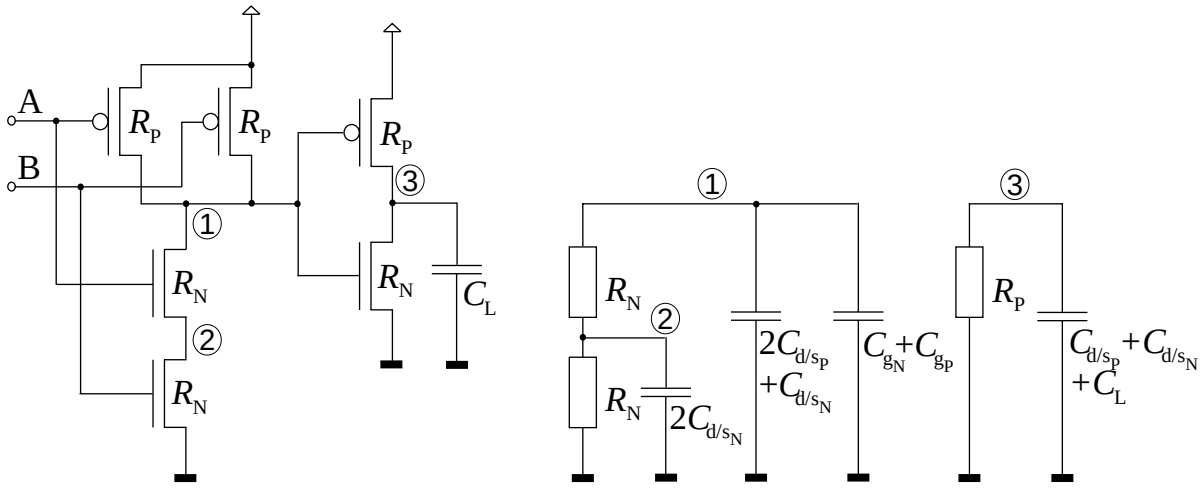



Bild 5.1. UND-Gatter und zugehöriges $RC$-Modell des längsten Pfades.

Source Kapazitäten der NMOS-Transistoren am Knoten 2 auf $U_\mathrm{B}$ geladen wurden. Erfolgt nun ein Wechsel der Eingangssignale $(A, B)$ : $(0,0) \longrightarrow (1,1)$, so müssen zunächst die Kapazitäten an den Knoten 1 und 2 entladen werden, um am Eingang des Inverters eine "0" zu erzeugen. Dadurch wird der PMOS-Transistor des Inverters am Ausgang leitend und die Lastkapazität sowie die Drain-Source Kapazitäten am Knoten 3 auf $U_\mathrm{B}$ geladen. Am Ausgang des UND-Gatters liegt damit der logische Zustand "1". In Tabelle 6.1 in Abschnitt 6.2.1 ist dieser Worst-Case für den Übergang $(0,0) \longrightarrow (1,1)$ erkennbar. Die entsprechende $RC$-Ersatzschaltung zur Modellierung des dabei auftretenden längsten Pfades ist im Bild 5.1 rechts dargestellt. Dabei wurden die leitenden Transistoren durch ihre endlichen Kanalwiderstände, die gesperrten Transistoren durch Leerläufe ersetzt.

Zur Minimierung der Rauschempfindlichkeit von CMOS-Gattern wird die Schaltschwelle in die Mitte zwischen $0V$ und $U_\mathrm{B}$ gelegt. Dazu muss für die Steilheitsfaktoren der Transistoren gelten, $\beta_\mathrm{N} = \beta_\mathrm{P}$. Dies wird dadurch erreicht, dass die Kanalwiderstände von N- und PMOS-Transistoren gleich gewählt werden, $R_{\mathrm{N},m} = R_{\mathrm{P},m} = R_m$. Damit dies erfüllt ist, muss bei gleicher Kanallänge $L_m$ ein festes Verhältnis zwischen den Kanalweiten eingehalten werden, d. h. die Kanalweiten von P- und NMOS-Transistor unterscheiden sich durch einen festen Faktor $s$, sodass gilt

$$\left(\frac{W_m}{L_m}\right)_\mathrm{P} = s\left(\frac{W_m}{L_m}\right)_\mathrm{N} \quad . \tag{5.2}$$

Die Drain- und Source-Kapazitäten der Transistoren sind proportional zu den Flächen von Drain- bzw. Source-Gebieten. Diese Flächen wiederum ändern sich proportional zu den Kanalweiten der Transistoren, sind jedoch konstant gegenüber den Kanallängen. Für die Drain- und Source-Kapazitäten von P- und NMOS-Transistoren gilt somit

$$C_{\mathrm{d/s_N},m} = \frac{1}{s} C_{\mathrm{d/s_P},m} = C_{\mathrm{d/s},m} \quad . \tag{5.3}$$

Die Gate-Kapazität eines Transistors ist proportional zur Fläche des Gates und somit proportional zum Produkt aus Kanallänge und Kanalweite. Da die Kanallängen von N- und PMOS-Transistoren gleich sind, ergibt sich für die Gate-Kapazitäten das gleiche Verhältnis wie für die Drain-/Source-Kapazitäten:

$$C_{\mathrm{g_N},m} = \frac{1}{s} C_{\mathrm{g_P},m} = C_{\mathrm{g},m} \quad . \tag{5.4}$$

Die Widerstände und Kapazitäten lassen sich damit in Abhängigkeit von den Transistordimensionen ausdrücken. Dabei gilt für die Kanalwiderstände

$$R_m = r\frac{L_m}{W_m} \quad , \tag{5.5}$$

für die Drain- und Source-Kapazitäten

$$C_{\mathrm{d/s},m} = c_{\mathrm{d/s}} W_m \tag{5.6}$$

und für die Gate-Kapazitäten

$$C_{\mathrm{g},m} = c_{\mathrm{g}} W_m L_m \quad . \tag{5.7}$$

Die Lastkapazität $C_{\mathrm{L},m}$ ist proportional der Fläche der Transistor-Gates der folgenden Gatter in Position $m+1$ und proportional der Anzahl der Gates, die an den Ausgang von Gatter $m$ angeschlossen sind. Da in der CMOS-Schaltungstechnik zu jedem NMOS-Transistor ein komplementärer PMOS-Transistor existiert ist es ausreichend, die Anzahl der Transistoren eines Typs zu bestimmen. Diese Zahl wird mit $\kappa_{m+1}$ bezeichnet. Mit den Gleichungen (5.4) und (5.7) folgt

$$C_{\mathrm{L},m} = \kappa_{m+1}(1+s) c_{\mathrm{g}} W_{m+1} L_{m+1} \quad . \tag{5.8}$$

Die Größen $r$, $c_{\mathrm{d/s}}$ und $c_{\mathrm{g}}$ sind materialspezifische Konstanten und nur von der jeweiligen Technologie abhängig.

Für die Schaltung aus Bild 5.1 ergibt sich gemäß dem Elmore-Delay-Modell die Zeit $\tau_{\mathrm{RC}}$, nach der die Ausgangsspannung bei sprunghafter Erregung am Eingang auf das $(1-\frac{1}{e})$-fache der Versorgungsspannung angestiegen ist zu

$$\begin{aligned} \tau_{\mathrm{RC}} \quad = \quad & R_{\mathrm{N},m} \cdot 2C_{\mathrm{d/s_N},m} + 2R_{\mathrm{N},m} \cdot (2C_{\mathrm{d/s_P},m} + C_{\mathrm{d/s_N},m} + C_{\mathrm{g_N},m} + C_{\mathrm{g_P},m}) \\ & + R_{\mathrm{P},m} \cdot (C_{\mathrm{d/s_P},m} + C_{\mathrm{d/s_N},m} + C_{\mathrm{L},m}) \quad . \end{aligned} \tag{5.9}$$

Die Verzögerungszeit eines Gatters wird üblicherweise als die Zeit definiert, nach der das Ausgangssignal $\frac{U_{\mathrm{B}}}{2}$ erreicht hat. Die Zeit $\tau_{\mathrm{RC}}$ muss deshalb noch mit einem konstanten Faktor von 0.8 multipliziert werden, um die eigentliche Verzögerung der Sprungantwort des Gatters zu erhalten.

$$\tau_{\mathrm{s},m} = 0.8 \cdot \tau_{\mathrm{RC}} \tag{5.10}$$

Ersetzt man in Gleichung (5.9) die Widerstände und Kapazitäten durch die entsprechenden Ausdrücke aus (5.3), (5.4), (5.5), (5.6), (5.7) und (5.8), so erhält man schließlich die Worst-Case-Verzögerung des Gatters bei sprunghafter Erregung als Funktion der Transistordimensionen $W$ und $L$.

$$\tau_{\mathrm{s},m} = 0.8 \cdot \left( 5rc_{\mathrm{d/s}}(1+s)L_m + 2rc_{\mathrm{g}}(1+s)L_m^2 + rc_{\mathrm{g}}\kappa_{m+1}(1+s)\frac{L_m}{W_m} W_{m+1} L_{m+1} \right) \tag{5.11}$$

**Anmerkung:** Das Elmore-Delay-Modell ist eine heuristische Methode zur Berechnung der dominanten Zeitkonstanten in RC-Schaltungen. In diesen Schaltungen ergeben sich natürlich gemäß dem Grad eine Reihe von Zeitkonstanten. Das Elmore-Delay-Modell ist also eine Näherung erster Ordnung.

Der zusätzliche Beitrag $\tau_{\mathrm{in},m}$ zur Verzögerungszeit eines Gatters in (5.1) dient zur Berücksichtigung der endlichen Flankensteilheit der Eingangssignale. In [27] und [28] wird dieser Beitrag in Abhängigkeit von der Anstiegs- bzw. Abfallszeit des Ausgangssignals des vorhergehenden Gatters, $\tau_{\mathrm{r/f},m-1}$, ausgedrückt. Zur Vereinfachung wird angenommen, dass die Anstiegs- und Abfallzeiten der Signalflanken gleich sind.

$$\tau_{\mathrm{in},m} = \Gamma \tau_{\mathrm{r/f},m-1} \left(1 + \frac{2U_{\mathrm{th}}}{U_{\mathrm{B}}}\right) \tag{5.12}$$

Dabei ist $\Gamma$ ein freier Parameter, der zum Abgleich technologiebedingter Schwankungen von $\tau_{\mathrm{in},m}$ dient. In erster Näherung kann der Verlauf der Signalflanken der Eingangsspannung $u_{\mathrm{in}}$ über der Zeit als linear angesehen werden. Dies gilt insbesondere im interessierenden Bereich $U_{\mathrm{th}} \leq u_{\mathrm{in}} \leq U_{\mathrm{B}} - U_{\mathrm{th}}$. In [28] ist eine Formel für die lineare Approximation $\tau_{\mathrm{eff},m-1}$ der Anstiegs- bzw. Abfallzeit $\tau_{\mathrm{r/f},m-1}$ angegeben:

$$\tau_{\mathrm{eff},m-1} = \frac{8}{3\ln 3}\tau_{\mathrm{s},m-1}\left(1 - 0.27\frac{U_{\mathrm{th}}}{U_{\mathrm{B}}}\right) \quad . \tag{5.13}$$

Die von der Eingangssignalflanke abhängige Verzögerungszeit kann damit als Funktion der Verzögerung der Sprungantwort des vorhergehenden Gatters ausgedrückt werden. Dazu wird in Gleichung (5.12) $\tau_{\mathrm{r/f},m-1}$ durch die lineare Approximation $\tau_{\mathrm{eff},m-1}$ ersetzt. Somit ergibt sich

$$\tau_{\mathrm{in},m} = \Gamma \frac{8}{3\ln 3}\left(1 + 1.73\frac{U_{\mathrm{th}}}{U_{\mathrm{B}}} - 0.54\left(\frac{U_{\mathrm{th}}}{U_{\mathrm{B}}}\right)^2\right) \cdot \tau_{\mathrm{s},m-1} \quad . \tag{5.14}$$

Fasst man alle technologieabhängigen Größen in der Konstanten $K$ zusammen, so erhält man

$$\tau_{\mathrm{in},m} = K \cdot \tau_{\mathrm{s},m-1} \quad . \tag{5.15}$$

Die gesamte Verzögerungszeit des Gatters berechnet sich nach Gleichung (5.1) aus der Summe der Verzögerung der Sprungantwort und dem zusätzlichen Beitrag aufgrund der endlichen Flankensteilheit. Aus (5.1) (5.11) und (5.15) folgt

$$\begin{aligned}
\tau_m &= K \cdot \left(5rc_{\mathrm{d/s}}(1+s)L_{m-1} + 2rc_{\mathrm{g}}(1+s)L_{m-1}^2 + rc_{\mathrm{g}}\kappa_m(1+s)\frac{L_{m-1}}{W_{m-1}}W_m L_m\right) \\
&\quad + 5rc_{\mathrm{d/s}}(1+s)L_m + 2rc_{\mathrm{g}}(1+s)L_m^2 + rc_{\mathrm{g}}\kappa_{m+1}(1+s)\frac{L_m}{W_m}W_{m+1}L_{m+1} \\
&= k_1 L_{m-1} + k_2 L_{m-1}^2 + k_3\frac{L_{m-1}}{W_{m-1}}L_m W_m + k_4 L_m + k_5 L_m^2 + k_6\frac{L_m}{W_m}L_{m+1}W_{m+1}.
\end{aligned} \tag{5.16}$$

Die technologieabhängigen Parameter sind hierbei in den Konstanten $k_1 \ldots k_6$ zusammengefasst.

Bislang wurde nur die Verzögerungszeit eines einzelnen UND-Gatters modelliert. Für alle anderen Gattertypen sowie auch für komplexere Blöcke wie Volladdierer, lässt sich die Verzögerungszeit nach dem gleichen Prinzip modellieren. Soll nun die Worst-Case-Verzögerungszeit entlang eines Pfades $p$ ermittelt werden, so ist dies durch einfache Summation aller Verzögerungszeiten der Einzelgatter in diesem Pfad möglich. Fasst man die Gatter wieder als Knoten eines Graphen auf, dann berechnet sich die Verzögerungszeit $\tau_p$ für einen Pfad $p : v_1 \longrightarrow v_N$ gemäß

$$\tau_p = \sum_{m=1}^{N} \tau_m \quad . \tag{5.17}$$

Sind die Eingänge des ersten Gatters ($m = 1$) primäre Eingänge der Schaltung, so ist $\tau_{\mathrm{r/f},0}$ in (5.12) durch die entsprechende Anstiegs- bzw. Abfallszeit des Eingangssignals zu ersetzen und diese Formel für die flankenabhängige Verzögerungszeit von Gatter Nummer 1 zu verwenden.

Das Verzögerungszeitmodell soll im Folgenden mit einer SPICE-Simulation überprüft werden. Für die SPICE-Simulation werden dazu die für die verwendete Technologie verfügbaren Transistormodelle eingesetzt. Die Beispielschaltung ist in Bild 5.2 gezeigt. Um die Er-

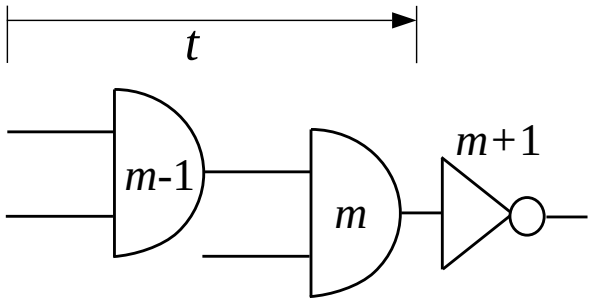



Bild 5.2. Beispielschaltung zum Vergleich des Verzögerungszeitmodells mit einer SPICE-Simulation.

gebnisse zweidimensional darstellen zu können wird hier eine einfache Beispielschaltung mit drei Gattern verwendet. Die Netzliste für die SPICE-Simulation ist aus dem Layout der Schaltung extrahiert ($0.25\mu m$-Technologie). Für diesen Vergleich werden die Längen- und Weitenverhältnisse der Gatter zueinander variiert und der Einfluss auf die Verzögerungszeit des Pfades entlang der ersten beiden Gatter bestimmt. Die Werte von $W_m$ und $L_m$ sind dabei konstant gehalten. Das Diagramm in Bild 5.3 zeigt das Ergebnis des Vergleichs zwischen der SPICE-Simulation und der Berechnung der Verzögerungszeiten mit dem oben hergeleiteten Modell.

Die mit Hilfe des Modells berechneten Verzögerungszeiten stimmen gut mit den Simulationsergebnissen überein. Für größere Verhältnisse $\sigma$ ergeben sich stärkere absolute Abweichungen. Große Unterschiede zwischen den Transistordimensionen benachbarter Gatter treten jedoch in der Praxis zumeist nicht auf. Zudem reicht für den Zweck der Ermittlung von Unterschieden in der Verzögerungszeit eine relative Genauigkeit aus. Diese ist mit dem vorgestellten Modell gegeben, wie Bild 5.3 zeigt: die Kurven für unterschiedliche $\sigma_{m+1}$ haben weitgehend immer den gleichen Abstand zueinander. Die Abweichung der mit dem Modell ermittelen Werte für die Verzögerungszeiten von den durch Simulation bestimmten Werten ist kleiner als der Fehler, der durch die Annahme einer datenunabhängigen Worst-Case-Verzögerung gemacht wurde. Für die hier beabsichtigte Anwendung des Modells, nämlich in einem Verfahren zur Glitch-Minimierung durch den Ausgleich unterschiedlicher Pfadverzögerungen, können diese Abweichungen toleriert werden, da zur Unterdrückung von Glitches die Pfade nicht unbedingt exakt abgeglichen werden müssen. Hier wirkt sich die Tatsache günstig aus, dass sehr kurze Glitches gedämpft oder gänzlich ausgefiltert werden (siehe Abschnitt 3.4). Eventuelle Modellungenauigkeiten haben daher geringeren Einfluss auf das eigentliche Entwurfsziel, nämlich die Reduzierung der Glitches.

Als ein weiteres Beispiel wird die Kaskadierung von fünf NAND-Gattern betrachtet, siehe Bild 5.4. Die Worst-Case-Verzögerung dieser Schaltung berechnet sich bei sprunghafter

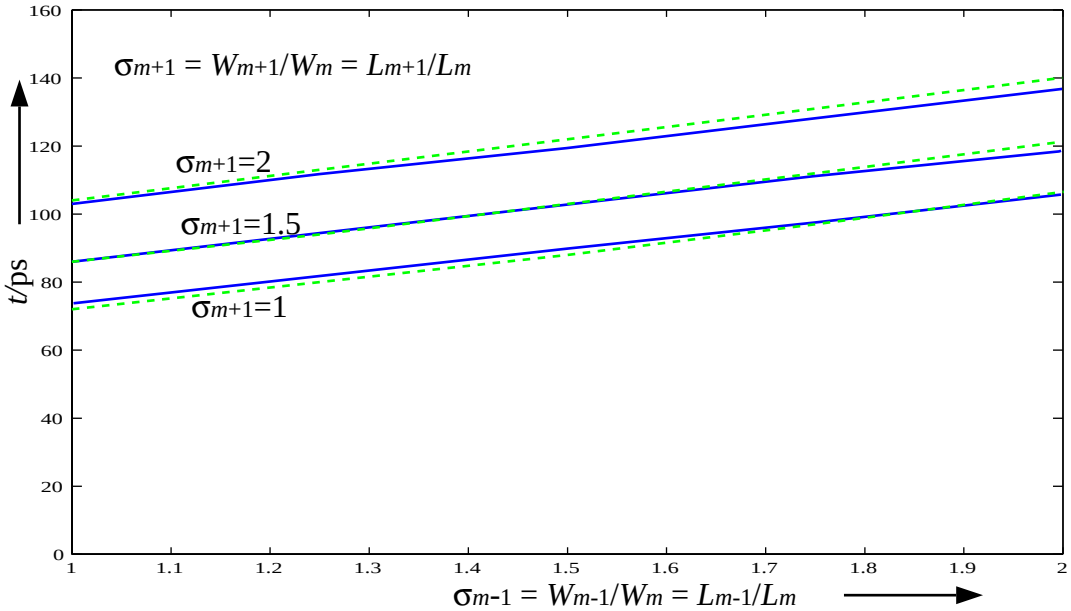



Bild 5.3. Verzögerungszeit $t$ über die Gatter $m-1$ und $m$ für verschiedene Längen- und Weitenverhältnisse $\sigma_{m-1} = \frac{W_{m-1}}{W_m} = \frac{L_{m-1}}{L_m}$ und $\sigma_{m+1} = \frac{W_{m+1}}{W_m} = \frac{L_{m+1}}{L_m}$. Die gestrichelte Linie zeigt das Ergebnis für die Berechnung mit dem hier vorgestellten Verzögerungszeitmodell, die durchgezogene Linie die Ergebnisse der SPICE-Simulationen.

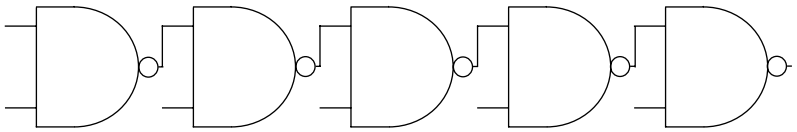


Bild 5.4. Kaskade von NAND-Gattern.

Erregung an den Eingängen des ersten Gatters zu

$$\begin{aligned}\tau_p &= k_3 L_1 + k_4 \frac{L_1}{W_1} W_2 L_2 + \left( \sum_{m=1}^{4} k_1 L_{m-1} + k_2 \frac{L_{m-1}}{W_{m-1}} W_m L_m + k_3 L_m + k_4 \frac{L_m}{W_m} W_{m+1} L_{m+1} \right) \\ &\quad + k_1 L_4 + k_2 \frac{L_4}{W_4} W_5 L_5 \quad . \end{aligned} \tag{5.18}$$

Die Längen aller Transistoren sind fest. Das Ziel ist nun, jenes Weitenverhältnis $\frac{W_{m+1}}{W_m}$, $m = 1 \ldots 5$, zu finden, bei dem $\tau_p$ minimal wird. Das errechnete optimale Weitenverhältnis ist $(\frac{W_{m+1}}{W_m})_{\text{opt}} = 1.516$. Bild 5.5 zeigt die mit SPICE ermittelten Worst-Case-Verzögerungszeiten der Schaltung für verschiedene Weitenverhältnisse. Das optimale Weitenverhältnis aus der Simulation stimmt fast exakt mit dem der Optimierung überein.

### 5.1.2 Ein Modell für die dynamische Verlustleistung von CMOS-Gattern

Mit der Änderung der Kanalweiten und -längen verändern sich auch die Flächen und damit die Kapazitäten der Gates sowie der Drain- und Source-Gebiete. Neben der kapazitiven Schaltleistung $P_{\text{s}}$ verändert sich dadurch auch die dynamische Kurzschlussleistung $P_{\text{k}}$ der Schaltung. Um den Einfluss dieser Änderungen auf den gesamten Leistungsverbrauch einer Schaltung bei der Optimierung der Transistordimensionen berücksichtigen zu können, muss

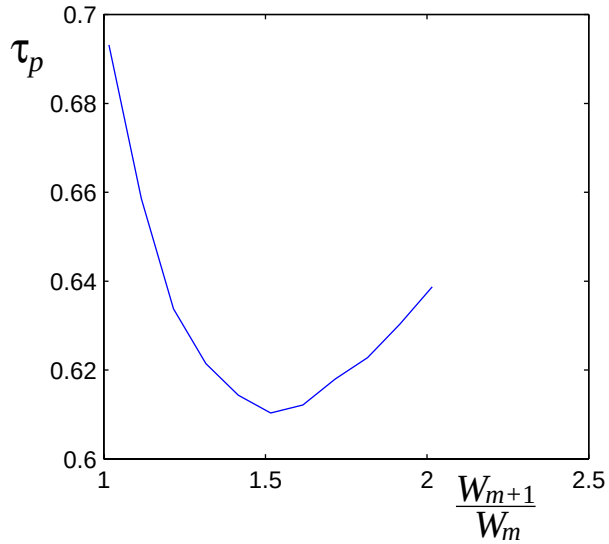



Bild 5.5. Mit SPICE ermittelte Worst-Case-Verzögerungszeiten der Schaltung aus Bild 5.4 für unterschiedliche Weitenverhältnisse.

auch die dynamische Verlustleistung als Funktion der Kanallänge $L$ und der Kanalweite $W$ dargestellt werden.

Die dynamische Verlustleistung eines CMOS-Gatters $m$ kann als Summe aus kapazitiver Schaltleistung und dynamischer Kurzschlussleistung aufgefasst werden:

$$P_m = P_{\mathrm{s},m} + P_{\mathrm{k},m} \quad . \tag{5.19}$$

Die kapazitive Schaltleistung des UND-Gatters aus Bild 5.1 kann nach Gleichung (1.6) aus den Drain-/Source- und Gate-Kapazitäten an den Knoten 1, 2 und 3 berechnet werden:

$$\begin{aligned} P_{\mathrm{s},m} &= \alpha_1 f U_{\mathrm{B}}^2 (C_{\mathrm{d/s_N},m} + 2C_{\mathrm{d/s_P},m} + C_{\mathrm{g_N},m} + C_{\mathrm{g_P},m}) + \alpha_2 f U_{\mathrm{B}}^2 2C_{\mathrm{d/s_N},m} \\ &\quad + \alpha_3 f U_{\mathrm{B}}^2 (C_{\mathrm{d/s_N},m} + C_{\mathrm{d/s_P},m}) + \alpha_3 f U_{\mathrm{B}}^2 C_{\mathrm{L}} \quad . \end{aligned} \tag{5.20}$$

Dabei sind $\alpha_1$, $\alpha_2$ und $\alpha_3$ die Schaltaktivitäten an den Knoten 1, 2 und 3. Diese sind über die logische Funktion der Schaltung fest mit den Schaltaktivitäten an den Eingängen verknüpft und unabhängig von den Transistordimensionen, $\alpha_i \neq f(L,W)\ \forall\ i$. Mit den Definitionen in (5.11) und (5.15) kann (5.20) als Funktion der Transistordimensionen formuliert werden.

$$\begin{aligned} P_{\mathrm{s},m} &= (1+2s)\alpha_1 f U_{\mathrm{B}}^2 c_{\mathrm{d/s}} W_m + \alpha_2 f U_{\mathrm{B}}^2 2c_{\mathrm{d/s}} W_m + (1+s)\alpha_1 f U_{\mathrm{B}}^2 c_{\mathrm{g}} W_m L_m \\ &\quad + (1+s)\alpha_3 f U_{\mathrm{B}}^2 c_{\mathrm{d/s}} W_m + (1+s)\alpha_3 f U_{\mathrm{B}}^2 \kappa_{m+1} c_{\mathrm{g}} W_{m+1} L_{m+1} \quad . \end{aligned} \tag{5.21}$$

Fasst man alle von den Transistordimensionen unabhängigen Größen in den Konstanten $c_1$, $c_2$ und $c_3$ zusammen, so erhält man einen einfachen Ausdruck für die kapazitive Schaltleistung eines UND-Gatters mit zwei Eingängen, in Abhängigkeit von den Kanallängen und -weiten des betrachteten Gatters und der unmittelbar nachfolgenden Gatter.

$$P_{\mathrm{s},m} = c_1 W_m + c_2 W_m L_m + c_3 W_{m+1} L_{m+1} \tag{5.22}$$

Neben der Leistung, die beim Laden von Kapazitäten aus der Quelle entnommen wird, bewirkt der Strom, der während eines Umschaltvorgangs direkt von der Quelle zur Masse

fließt, wenn NMOS- und PMOS-Blöcke gleichzeitig leiten, einen zusätzlichen Leistungsverbrauch. Bereits in der Einleitung wurde diese Kurzschlussleistung für einen unbelasteten Inverter angegeben:

$$P_{\mathrm{k},m} = \frac{\beta_m}{12}\tau_{\mathrm{eff},m-1} f (U_{\mathrm{B}} - 2U_{\mathrm{th}})^3 \quad . \tag{5.23}$$

Hier wurde die Anstiegs- bzw. Abfallzeit des Eingangssignals durch die in Gleichung (5.13) angegebene effektive Anstiegs- bzw. Abfallzeit des Ausgangssignals des vorhergehenden Gatters ersetzt. Wie an gleicher Stelle in der Einleitung gezeigt wurde, ist der Kurzschlussstrom umso größer, je kleiner die Lastkapazität am Ausgang ist. Das bedeutet, dass die obige Formel für den unbelasteten Fall eine obere Grenze für die Kurzschlussleistung darstellt. Der Steilheitsfaktor $\beta_m$ ist in dieser Formel für NMOS- und PMOS-Transistor gleich angenommen, wobei $\beta_m \sim \frac{W_m}{L_m}$. Die Gleichung (5.23) kann auch für allgemeine Gatter angewendet werden, wenn man die gleichzeitig leitenden NMOS- und PMOS-Blöcke als jeweils einen einzigen Transistor mit entsprechendem Steilheitsfaktor auffasst. Nimmt man eine reale Last ungleich null am Ausgang des Gatters an, so kann (5.23) dennoch zur Berechnung der Kurzschlussleistung angewendet werden, wenn durch die Wahl von $\beta_m$ sichergestellt wurde, dass die Anstiegs- bzw. Abfallzeiten der Eingangs- und Ausgangsflanken gleich sind. Wie in [85] empirisch ermittelt wurde, ist die Kurzschlussleistung in diesem Fall etwa die Hälfte des mit der angegebenen Formel berechneten Wertes. Die dafür nötige Voraussetzung gleicher Anstiegs- bzw. Abfallzeiten an Ein- und Ausgängen kann im Allgemeinen nicht garantiert werden, insbesondere dann nicht, wenn die Transistordimensionen im Hinblick auf ein anderes Optimierungsziel verändert werden. Für die Berücksichtigung der Kurzschlussleistung bei der Optimierung der Schaltung hinsichtlich gleich langer Pfade, ist jedoch die Abschätzbarkeit einer oberen Grenze bereits hilfreich. Man befindet sich so immer auf der sicheren Seite, wenn bei der Optimierung die dynamische Verlustleistung unterhalb einer bestimmten Schranke gehalten werden soll.

Das UND-Gatter in Bild 5.1 besteht aus zwei elementaren Grundgattern: einem NAND-Gatter und einem nachgeschalteten Inverter. Die Kurzschlussleistungen dieser beiden Gatter müssen separat betrachtet werden. Zur Bestimmung der Kurzschlussleistung des NAND-Gatters muss die Formel (5.23) mit der Schaltaktivität am Knoten 1, $\alpha_1$, multipliziert werden. Entsprechend muss für den Inverter eine Gewichtung mit der Aktivität $\alpha_3$ erfolgen. Bei dieser genaueren Betrachtung ergibt sich die obere Grenze der dynamischen Kurzschlussleistung des UND-Gatters an der Position $m$ aus der Summe der beiden Anteile:

$$P_{\mathrm{k},m} = P_{\mathrm{k_{Inv}},m} + P_{\mathrm{k_{NAND}},m} = \frac{1}{12}(\beta_{\mathrm{NAND},m} \cdot \tau_{\mathrm{eff},m-1} + \beta_{\mathrm{Inv},m} \cdot \tau_{\mathrm{eff},NAND}) f (U_{\mathrm{B}} - 2U_{\mathrm{th}})^3 \quad . \tag{5.24}$$

Inwieweit es sinnvoll ist, die Kurzschlussleistung bei der Optimierung der Schaltung hinsichtlich gleicher Pfadlängen zu berücksichtigen, hängt auch von der verwendeten Technologie ab. Bei den neueren Technologien mit immer kleineren minimalen Abmessungen nimmt der Anteil des Kurzschlussleistung zum Gesamtleistungsverbrauch tendenziell zu. Für die Beispiele in der vorliegenden Arbeit wurde zumeist ein $0.25\mu m$ Prozess mit einer Versorgungsspannung von $2.5V$ verwendet. In Experimenten hat sich hierbei gezeigt, dass eine Vernachlässigung der Kurzschlussleistung keine Nachteile für das Optimierungsergebnis zur Folge hat. Dies muss jedoch beim Übergang auf eine andere Technologie nicht mehr ohne weiters zutreffen. Im Allgemeinen muss also die Kurzschlussleistung berücksichtigt werden.

Die Optimierung der Schaltung für gleiche Pfadlängen geschieht immer stückweise, Pfad für Pfad. Es ist deshalb auch zweckmäßig, dynamische Verlustleistungen für komplette Pfade

zu definieren. Der dynamische Leistungsverbrauch eines Pfades $p$ mit $N$ Gattern ergibt sich zu

$$P_p = \sum_{m=1}^{N} (P_{\mathrm{s},m} + P_{\mathrm{k},m}) \quad . \tag{5.25}$$

$P_p$ ist bei gegebener Technologie und fest vorgeschriebener Frequenz und Versorgungsspannung nur eine Funktion der Transistorgrößen $W_m$ und $L_m$, $m = 1 \ldots N$.

## 5.2 Formulierung als Mehrzieloptimierungsproblem

Die grundsätzliche Schwierigkeit bei der Suche nach den optimalen Transistordimensionen ergibt sich aus der Tatsache, dass die beiden Zielkriterien, niedrige dynamische Verlustleistung und gleiche Pfadverzögerungen, u. U. widersprüchliche Forderungen an die Optimierungsvariablen $W$ und $L$ stellen. Die Verminderung der Verarbeitungsgeschwindigkeit in zu schnellen Pfaden erfordert eine Vergrößerung der Kanalwiderstände und somit gegebenenfalls eine Verlängerung der Kanäle, wenn die Verringerung der Kanalweiten bis zur Minimalweite nicht ausreicht. Mit der Verlängerung der Kanäle erhöhen sich auch die Gate-Kapazitäten und somit die dynamische Verlustleistung. Minimale dynamische Verlustleistung hingegen erfordert möglichst kleine Transistorabmessungen. Für die Behandlung von Problemstellungen dieser Art eignen sich Mehrziel- oder Vektoroptimierungsverfahren. Die Grundidee dieser Verfahren ist die Transformation des vektorwertigen Optimierungsproblems in ein skalarwertiges Optimierungsproblem. Für skalarwertige Optimierungsprobleme sind bereits ein Reihe von Lösungsverfahren bekannt. Durch die Lösung des skalarwertigen Ersatzproblems wird genau eine mögliche Lösung des Vektoroptimierungsproblems gefunden. Mit einer Parametrisierung des skalaren Problems lässt sich durch Variation des Parameters eine Vielzahl von Lösungen des vektorwertigen Problems ermitteln. Für die Parametrisierung gibt es unterschiedliche Möglichkeiten. Ein häufig verwendeter Ansatz ist die sog. *Methode der Beschränkungen* (engl.: *$\epsilon$-constraint method*) [10]. Dabei wird ein bestimmtes Zielkriterium ausgewählt, das optimiert werden soll. Für alle anderen Zielgrößen werden obere- bzw. untere Schranken festgelegt. Durch die Parametrisierung werden hier also die Auswahl der zu optimierenden Zielgröße sowie die Schranken der übrigen Zielgrößen festgelegt. Das Problem bei dieser Methode ist die Wahl der Schranken, da diese im Allgemeinen nicht a priori bekannt sind. Werden obere Schranken zu niedrig bzw. untere Schranken zu hoch gewählt, so existiert für das skalarwertige Ersatzproblem keine zulässige Lösung. Werden andererseits obere Schranken zu hoch bzw. untere Schranken zu niedrig gewählt, so gehen die in diesen Schranken festgelegten Zielgrößen nicht mehr in das Optimierungsproblem ein, haben also keinen Einfluss auf die Lösungen.

In der hier vorliegenden Arbeit wird ein anderer Ansatz zur Lösung des Mehrzieloptimierungsproblems verfolgt, die *Wichtungsmethode*. Dabei wird jeder einzelnen Zielfunktion ein Gewichtfaktor zugeordnet. Das skalarwertige Ersatzproblem lässt sich dann als die gewichtete Summe der einzelnen Zielfunktionen formulieren. In einer Schaltung existieren für jeden Pfad zwei zu minimierende Zielfunktionen: der dynamische Leistungsverbrauch nach Gleichung (5.25) und die Differenz zwischen den Pfadverzögerungen. Es werden nur rein kombinatorische Schaltungen betrachtet oder kombinatorische Teile von sequentiellen Schaltungen, also Schaltungsteile, die zwischen Registern liegen. Ist im Folgenden von "Schaltungen" die Rede, so sind damit immer rein kombinatorische Schaltungen oder kombinatorische

Schaltungsteile zwischen Registern gemeint. Durch das Balancieren der Pfade, also den Abgleich der Pfadverzögerungen, soll die Gesamtverzögerungszeit einer Schaltung unbeeinflusst bleiben. Das bedeutet, dass die Verzögerungszeit des kritischen Pfades der ursprünglichen Schaltung nicht verändert werden darf. Es wird hier angenommen, dass die Optimierung der Schaltung hinsichtlich der Geschwindigkeit bereits vorher separat durchgeführt wurde. Die Worst-Case-Verzögerung des kritischen Pfades, $\tau_{\text{krit}}$ ist somit fest vorgegeben. Das Balancieren der anderen (unkritischen) Pfade richtet sich nach dem kritischen Pfad. Die zugehörige Zielfunktion ist somit gegeben durch

$$f_{1,p}(\boldsymbol{W},\boldsymbol{L}) := \Delta\tau_p = |\sum_{m=1}^{N} \tau_m - \tau_{\text{krit}}| = |\tau_p - \tau_{\text{krit}}| \quad . \tag{5.26}$$

Die zweite Zielfunktion lautet

$$f_{2,p}(\boldsymbol{W},\boldsymbol{L}) := \sum_{m=1}^{N} (P_{\text{s},m} + P_{\text{k},m}) \quad , \quad p : v_1 \longrightarrow v_N \quad . \tag{5.27}$$

Die vektorwertige Zielfunktion lautet damit $\boldsymbol{f}(\boldsymbol{x}) = [f_{1,p}(\boldsymbol{x})\ f_{2,p}(\boldsymbol{x})]^{\text{T}}$ mit $\boldsymbol{x} = [\boldsymbol{W}\ \boldsymbol{L}]^T$. Die Optimierungsvariablen $W$ und $L$ können innerhalb eines zulässigen Bereichs R eingegrenzt werden, sodass gilt

$$\text{R} := \{(W_m, L_m) \in \mathbb{R}^2 : W_{\text{feat}} \leq W_m \leq W_{\max} \wedge L_{\text{feat}} \leq L_m \leq L_{\max}\} \quad . \tag{5.28}$$

Dabei sind $W_{\text{feat}}$ und $L_{\text{feat}}$ die durch die Technologie (Feature Size) gegebenen Minimalwerte, und $W_{\max}$ und $L_{\max}$ die vom Entwickler festzulegenden Maximalwerte. Die Funktionen $f_{1,p}(\boldsymbol{W},\boldsymbol{L})$ und $f_{2,p}(\boldsymbol{W},\boldsymbol{L})$ sind stets konvexe Funktionen in $\boldsymbol{W}$ und $\boldsymbol{L}$, unabhängig von den damit charakterisierten Gattertypen. Den einzelnen Zielfunktionen werden Gewichtsfaktoren $0 \leq w_i \leq 1$ zugeordnet. Die Werte $w_i$ können in einem Gewichtsvektor $\boldsymbol{w}$ mit der Dimension $k$ zusammengefasst werden, wobei $k$ der Anzahl der Zielkriterien entspricht. Üblicherweise wird eine Normierung der Gewichtsfaktoren durchgeführt, sodass die Summe aller Gewichte eins ergibt, $\sum_{i=1}^{k} w_i = 1$. Im hier vorliegenden zweidimensionalen Fall lässt sich der Gewichtsvektor folgendermaßen in Abhängigkeit von nur einer einzigen Gewichtsgröße $0 \leq w \leq 1$ darstellen:

$$\boldsymbol{w} = \begin{bmatrix} w \\ 1-w \end{bmatrix} \quad . \tag{5.29}$$

Das Vektoroptimierungsproblem kann mit dieser Parametrisierung auf das Ersatzproblem mit der skalarwertigen Zielfunktion $S(\boldsymbol{W},\boldsymbol{L}) = \boldsymbol{w}^{\text{T}}\boldsymbol{f}$ übergeführt werden. Die Optimierungsaufgabe lautet somit allgemein

$$\min_{(\boldsymbol{W},\boldsymbol{L})\in\text{R}} \left(S(\boldsymbol{W},\boldsymbol{L})\right) \quad . \tag{5.30}$$

Ein Problem bei der Differentiation dieser Funktion stellt der Betrag der Differenz aus den Verzögerungen des betrachteten Pfades und des kritischen Pfades nach Gleichung (5.26) dar. Um eine Fallunterscheidung an der Unstetigkeitsstelle $\tau_p = \tau_{\text{krit}}$ zu vermeiden, wird der Betrag der Differenz durch das Quadrat ersetzt. Dadurch verändert sich die Lösung des Optimierungsproblems nicht. Für den hier betrachteten Fall lautet das Optimierungsproblem somit

$$\min_{(\boldsymbol{W},\boldsymbol{L})\in\text{R}} \left(w(\tau_p - \tau_{\text{krit}})^2 + (1-w)P_p\right) \quad . \tag{5.31}$$

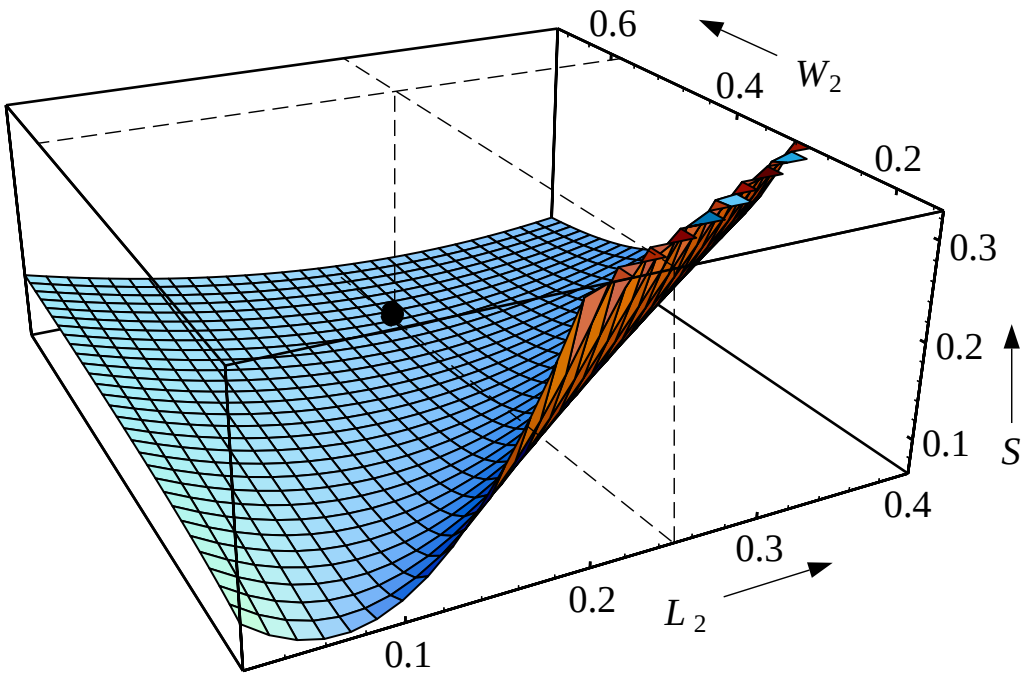



Bild 5.6. Graph der Funktion $S(W_2, L_2)$ für die Schaltung aus Bild 5.2. $N = 2$, $\boldsymbol{x} = [W_2\ L_2]^{\mathrm{T}}$, $w = 0.5$.

In Bild 5.6 ist der Verlauf von $S(\boldsymbol{W}, \boldsymbol{L})$ für die Schaltung aus Bild 5.2 entlang von $W_2$ und $L_2$ dreidimensional dargestellt. Das berechnete Optimum liegt bei $W_2^* = 0.58\mu m$, $L_2^* = 0.25\mu m$.

Durch die Variation des Parameters $w$ lässt sich die Menge der *effizienten Punkte* $S^*$ der Zielfunktion bestimmen, indem für jedes $w$ die Lösung von (5.31) ermittelt wird. Die effizienten Punkte der Zielfunktion nennt man auch *paretooptimale Punkte*. Die zugehörigen Urbilder $\boldsymbol{W}^*$ und $\boldsymbol{L}^*$ heissen ebenfalls paretooptimal. Allgemein lässt sich folgender Satz formulieren:

**Definition 6.1** Es sei $\boldsymbol{f}$ eine vektorwertige Zielfunktion mit

$$\boldsymbol{f} : \begin{cases} \mathbb{R}^{\mathrm{n}} \longrightarrow \mathbb{R}^{\mathrm{k}} \\ \boldsymbol{x} \longmapsto \boldsymbol{f}(\boldsymbol{x}) \end{cases}$$

und $\boldsymbol{x} \in \mathrm{R} \subset \mathbb{R}^{\mathrm{n}}$. Weiterhin sei $\mathrm{f(R)} \subset \mathbb{R}^{\mathrm{k}}$ die Bildmenge des zulässigen Bereichs R unter der vektorwertigen Zielfunktion $\boldsymbol{f}$. Ein Punkt $\boldsymbol{y}^* \in \mathrm{f(R)}$ heisst dann effizient oder paretooptimal, wenn es kein $\boldsymbol{y} \in \mathrm{f(R)}$ mit $\boldsymbol{y} \neq \boldsymbol{y}^*$ gibt, für das gilt $\boldsymbol{y} \leq \boldsymbol{y}^*$[1].

Bild 5.7 verdeutlicht die Menge der paretooptimalen Punkte für die Zielfunktion $\boldsymbol{f}(\mathrm{R})$ mit $k = 2$. Die Lösung des skalarwertige Ersatzproblems entspricht genau einem Punkt auf dem in der Zeichnung hervorgehobenen Kurvenstück.

Eine theoretische Begründung für die mit der Wichtungsmethode vorgenommene Transformation auf ein skalarwertiges Ersatzproblem liefert das Theorem von Kuhn und Tucker. Das vorliegende Optimierungsproblem kann als ein Vektoroptimierungsproblem mit Ungleichungsnebenbedingungen aufgefasst werden. Die Nebenbedingungen ergeben sich hier aus

[1]Das bedeutet hier: $\boldsymbol{y}^*$ ist bezüglich mindestens einer Komponente kleiner als $\boldsymbol{y}$ und bezüglich aller anderen Komponenten nicht größer als $\boldsymbol{y}$.

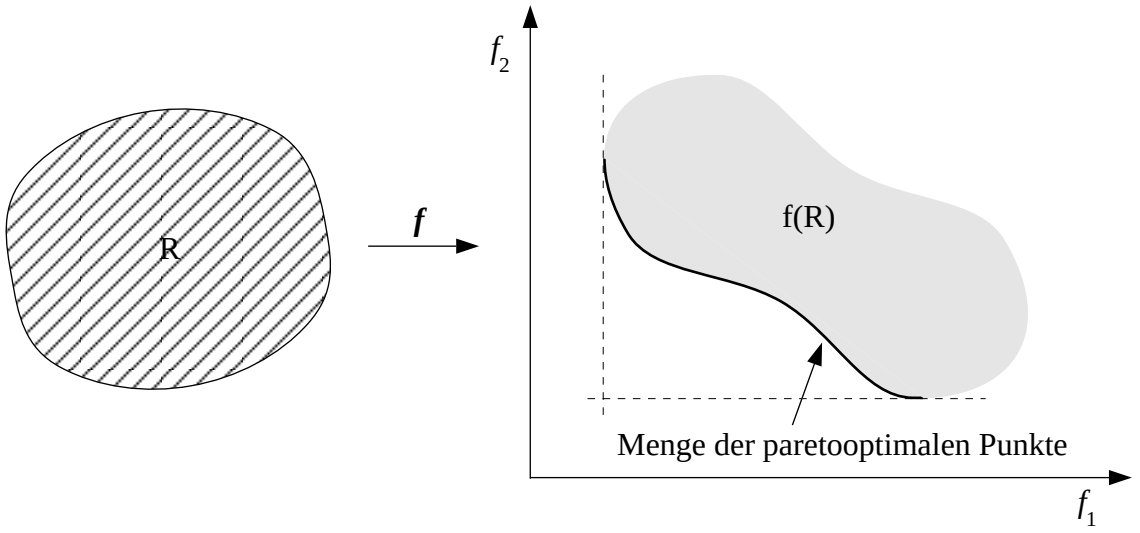



Bild 5.7. Menge der paretooptimalen Punkte einer vektorwertigen Zielfunktion $\boldsymbol{f}$.

der Eingrenzung des zulässigen Bereichs für die Optimierungsvariablen $\boldsymbol{W}$ und $\boldsymbol{L}$ innerhalb der durch die Feature Size der Technologie gegebenen minimalen und der vom Entwickler festgelegten maximalen Abmessungen. Die Anzahl der Ungleichungsnebenbedingungen sei $q$. Für die Optimierung der Transistordimensionen ergeben sich $q = 4$ Ungleichungsnebenbedingungen:

$$\begin{aligned} h_1 &= \frac{W_{\min}}{W_m} - 1 \leq 0 \quad , \\ h_2 &= \frac{W_m}{W_{\max}} - 1 \leq 0 \quad , \\ h_3 &= \frac{L_{\min}}{L_m} - 1 \leq 0 \quad , \\ h_4 &= \frac{L_m}{L_{\max}} - 1 \leq 0 \quad . \end{aligned} \tag{5.32}$$

Es sei $\boldsymbol{x}^* = [W_m^* \; L_m^*]^T$. Die Gradienten $\nabla h_i(\boldsymbol{x}^*)$ seien linear unabhängig für alle Nebenbedingungen $i$. Für die Paretooptimalität gilt der folgende Satz:

**Satz 6.2** Ein Punkt $\boldsymbol{x}^*$ ist paretooptimal wenn es einen Wichtungsvektor $\boldsymbol{w}$, $\sum_{i=1}^{k} w_i = 1, 0 \leq w_i \leq 1$ gibt, sodass für das skalarwertige Ersatzproblem $S(\boldsymbol{x})$ mindestens ein Vektor von Lagrangemultiplikatoren $\boldsymbol{\lambda} \in \mathbb{R}^q$ existiert, mit dem die folgenden Bedingungen erfüllt sind:

$$\begin{gathered} \nabla S(\boldsymbol{x}^*) + \sum_{i=1}^{q} \lambda_i \nabla h_i(\boldsymbol{x}^*) = 0 \\ \lambda_i \geq 0 \quad , \quad h_i(\boldsymbol{x}^*) \leq 0 \quad , \quad \lambda_i h_i(\boldsymbol{x}^*) = 0 \quad . \end{gathered} \tag{5.33}$$

Obige Bedingung heisst *Kuhn-Tucker Optimalitätsbedingung erster Ordnung.*

Ein konkretes Beispiel für den hier behandelten Fall der Optimierung von CMOS-Schaltungen hinsichtlich der Zielfunktionen (5.26) und (5.27) ist in Bild 5.8 dargestellt. Als

Beispielschaltung wurde wieder die Kaskade von fünf NAND-Gattern gemäß Bild 5.4 verwendet. Die linke Grafik zeigt die Verläufe von $P_p$ und $\Delta\tau_p$ über dem Wichtungsparameter

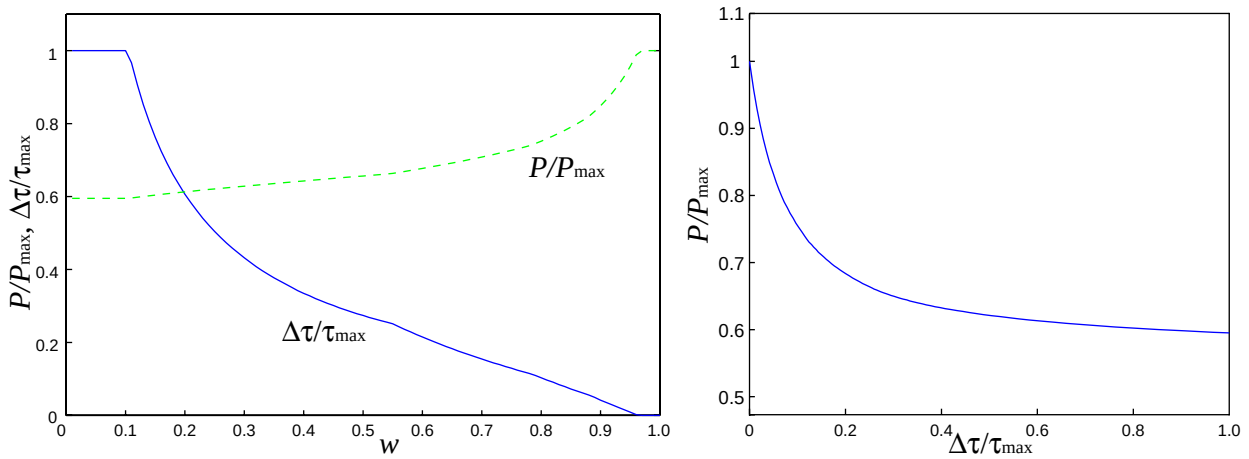



Bild 5.8. Links: Optimale Lösungen für die dynamische Verlustleistung $P$ und Differenz der Pfadverzögerungen $|\tau - \tau_{\text{krit}}|$ für die Schaltung aus Bild 5.4. Rechts: Menge der paretooptimalen Punkte des vektorwertigen Optimierungsproblems.

$w$. Die Werte sind jeweils auf die Maximalwerte $P_{\max}$ bzw. $\Delta\tau_{\max}$ normiert. Jeder Punkt auf diesen Kurven ist das errechnete Optimum für das jeweilige $w$. Die Bestimmung der Kurven erfolgte durch das Lösen des skalarwertigen Ersatzproblems $\min_{(\boldsymbol{W},\boldsymbol{L})\in\mathrm{R}} S(\boldsymbol{W},\boldsymbol{L})$ für jeweils feste Werte von $w$. Die Variation von $w$ wurde im Bereich $0 \leq w \leq 1$ mit einer Schrittweite von 0.01 durchgeführt. Aus den Kurven lassen sich also die paretooptimalen Punkte des vektorwertigen Optimierungsproblems ablesen. Die rechte Grafik in Bild 5.8 zeigt die Menge der paretooptimalen Punkte in der $\boldsymbol{f}$-Ebene für das konkrete Beispiel aus Bild 5.4. Jeder Punkt auf dieser Kurve stellt eine Schaltungsrealisierung mit einer bestimmten Transistordimensionierung dar. Im Bereich $0 \leq w \leq 0.1$ in der linken Grafik, in dem beide Kurven waagrecht verlaufen, überwiegt das Gewicht der dynamischen Schaltleistung, sodass keine Veränderung der Transistordimensionen erfolgt, da dies im vorliegenden Beispiel eine Vergrößerung der Transistoren und somit eine Vergrößerung der Kapazität zur Folge hätte. Mit zunehmender Gewichtung auf $\Delta\tau$ wird die Schaltung mehr und mehr balanciert. Für $w \approx 0.95$ ist der betrachtete Pfad schließlich exakt auf den kritischen Pfad abgeglichen. Ab diesem Punkt, bei dem $\Delta\tau = 0$ erreicht wird, ändert sich das Optimierungsergebnis für größere $w$ nicht mehr, was in den waagrechten Kurvenverläufen für $0.95 \leq w \leq 1.0$ zum Ausdruck kommt.

Durch das Hinzuziehen von zusätzlichen, übergeordneten Kriterien kann nun aus der Menge der paretooptimalen Punkte ein bestimmter Punkt ausgewählt werden. Ein solches übergeordnetes Kriterium ist beispielsweise die tatsächliche gesamte Verlustleistung der Schaltung. Dieser setzt sich zusammen aus der mit dem Modell berechenbaren dynamischen Verlustleistung, bei der das Auftreten von Glitches noch nicht berücksichtigt ist und aus der Verlustleistung aufgrund von Glitches, die mit der Differenz der Pfadverzögerungen in Zusammenhang steht. Ein Problem ist dabei, dass es kein direktes Maß für die Glitch-Aktivität in Abhängigkeit von der Differenz der Pfadverzögerungen gibt. Ein direkter Rückschluss auf den Leistungsverbrauch ist damit nicht möglich. Es kann innerhalb der Modellgenauigkeit lediglich garantiert werden, dass im Fall $\Delta\tau_p = 0\ \forall p$ die Glitch-Aktivität in der Schaltung auf Gatterebene zu null wird. In der Menge der paretooptimalen Punkte lässt

sich die tatsächlich optimale Lösung für die Schaltungsrealisierung mit der geringsten Verlustleistung weiter einkreisen, indem Schaltungsrealisierungen in der Nähe des vermuteten Optimums simuliert werden. Der Bereich, in dem das Optimum liegen kann, lässt sich dadurch stark eingrenzen. Die berechneten paretooptimalen Lösungen können in der Praxis nur bis zu einer bestimmten Genauigkeit realisiert werden. Der Fertigungsprozess erlaubt für alle Abmessungen nur eine bestimmte Auflösung und erfordert eine Quantisierung der errechneten Lösungen. Die Auflösung des hier verwendeten $0.25\mu m$-Prozesses beträgt $0.025\mu m$. Diese Quantisierung schränkt in der Praxis die Lösungen nicht ein, da die Abweichungen von berechneten zu realisierbaren Lösungen mit maximal einem Zwanzigstel ($\pm 0.0125\mu m$) der Feature Size kaum eine messbare Wirkung auf das Verhalten der Schaltung haben.

## 5.3 Ein Algorithmus zum Abgleich unterschiedlicher Pfadverzögerungen durch Transistordimensionierung

Die Optimierung der Transistordimensionen in einer Schaltung hinsichtlich der in (5.26) und (5.27) gegebenen Zielfunktionen erfolgt für jeden Pfad separat. Für jeden optimierbaren Pfad der Schaltung werden zunächst die Pfadverzögerung nach Gleichung (5.17) und die Leistung nach Gleichung (5.25) ermittelt. Bei den einfachen Beispielen im vorhergehenden Abschnitt konnte das Aufstellen der Funktionen für Verzögerung und Leistung noch manuell ausgeführt werden. Die Komplexität größerer Schaltungen erfordert eine Automatisierung des Verfahrens. Insbesondere für die Manipulationen an der Schaltung selbst, die nach der Bestimmung der optimalen Lösungen für die Transistorweiten und -längen eventuell durchgeführt werden müssen, ist es wünschenswert, programmunterstützt vorgehen zu können. Eine manuelle Ausführung dieser Aufgaben ist im Allgemeinen zu zeitaufwendig und zudem fehleranfällig. Die Manipulationen an den Transistordimensionen werden in der Netzliste der zu optimierenden Schaltung vorgenommen. Die Umsetzung auf das Layout kann in einem weiteren Schritt, beispielsweise durch Modulgeneratoren, ebenfalls automatisch erfolgen.

Die Eingabe für das Programm zur automatischen Optimierung beliebiger Schaltungen besteht aus der Netzliste der ursprünglichen Schaltung, den Funktionen für die Verzögerungszeiten und die dynamischen Verlustleistungen der einzelnen Gatter in Abhängigkeit von den Kanalweiten und Kanallängen der MOS-Transistoren sowie aus dem vom Entwickler vorgegebenen Wert für den Wichtungsparameter $w$. Eine eventuelle Optimierung der Schaltung bezüglich bestimmter Geschwindigkeitsanforderungen muss bereits vor dem Abgleichen der Pfade erfolgen. Die maximale Verzögerungszeit wird deshalb im weiteren als fest vorgegeben angenommen. Die folgenden Optimierungsschritte dürfen lediglich die Unterschiede der Pfadverzögerungen eliminieren, nicht jedoch die Gesamtgeschwindigkeit der Schaltung verändern. Es wird also angenommen, dass die Eingabenetzliste von einer bereits hinsichtlich Geschwindigkeit optimierten Schaltung stammt. Die Funktionen für $\tau_m$ und $P_m$ der Gatter werden in einer Bibliothek bereitgestellt. Für jeden Gattertyp sind darin die entsprechenden Funktionen abgelegt. Taucht der Name eines Gatters in der Netzliste auf, so wird die Bibliothek nach diesem Gatter durchsucht und die benötigten Funktionen werden ausgelesen. Die Ausgabe des Programms ist die Netzliste der optimierten Schaltung. Da die Optimierung auf Gatterebene erfolgt, ist es sinnvoll, eine modulare, hierarchische Netzlistenstruktur zu verwenden. Jeder Gattertyp wird darin nur einmal auf Transistorebene definiert und im weiteren lediglich durch Angabe des Gatternamens, der äusseren Anschlüsse und der entsprechenden Parameter $W$ und $L$ in der Netzlistenbeschreibung der Schaltung verwendet.

Die erste Aufgabe des Optimierungsalgorithmus' besteht darin, den kritischen Pfad zu finden und die maximale Verzögerungszeit der Schaltung zu ermitteln. Wie bereits erwähnt, werden dabei nur rein kombinatorische Schaltungsteile untersucht, in sequentiellen Schaltungen also nur die Logik zwischen zwei Registerstufen. Die Schaltung wird durch ihren Graphen repräsentiert, wobei jeder Knoten des Graphen einem Gatter der Schaltung entspricht. Zunächst wird eine Liste der Abhängigkeiten erstellt, in die jeder Knoten des Schaltungsgraphen und dessen Verzögerungszeit eingetragen wird. Zudem werden für jeden Knoten $n_v$ auch die unmittelbaren Vorgängerknoten $v$ und die maximalen Pfadverzögerungen $\delta_{\max}(v)$ vom Eingang bis zu diesen Knoten in einer Vorgängerliste $\mathbf{LV}_n$ abgespeichert. Die Verzögerungszeit der einzelnen Knoten wird dabei mit der Formel für $\tau_m$ des entsprechenden Gattertyps und den aktuellen Werten für die Parameter $W$ und $L$ berechnet. Bild 5.9 zeigt das

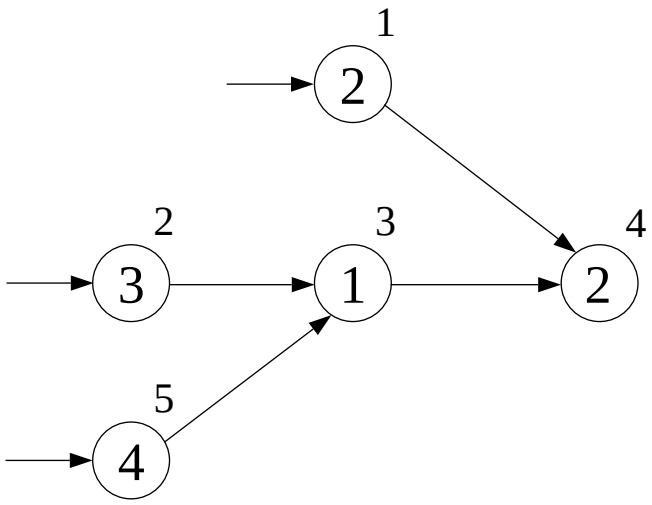



Bild 5.9. Schaltungsgraph.

Beispiel für einen Schaltungsgraphen mit den in die Knoten eingetragenen Werten für die Verzögerungszeiten. In Tabelle 5.1 ist die zugehörige Liste der Abhängigkeiten dargestellt. Für jeden Knoten $n_v$ lässt sich die maximale Pfadverzögerung $\Delta(n_v)$ seiner unmittelbaren

| Knoten $n_v$ | Delay $d(n_v)$ | Liste $\mathbf{LV}_n$ der Vorgänger $\delta_{\max}(v)$ |
|---|---|---|
| 1 | 2 | |
| 2 | 3 | |
| 3 | 1 | 4(5) |
| 4 | 2 | 2(1), 5(3) |
| 5 | 4 | |

Tabelle 5.1. Liste der Abhängigkeiten für den Schaltungsgraphen aus Bild 5.9.

Vorgängerknoten $v$ mit

$$\Delta(n_v) = \max_v(\delta_{\max}(v)) \tag{5.34}$$

bestimmen. Zum Beispiel ergeben sich aus Tabelle 5.1 für die Knoten 3 und 4 die Werte $\Delta(3) = 4$ und $\Delta(4) = 5$. Aus der Liste der Abhängigkeiten ist die maximale Pfadverzögerung

$\tau_{\text{krit}}$ dann leicht zu ermitteln. Dazu muss lediglich

$$\tau_{\text{krit}} = \max_{n_v} \left( d(n_v) + \Delta(n_v) \right) \tag{5.35}$$

gebildet werden. Alle Knoten des zugehörigen kritischen Pfades werden als "fest" markiert, d. h. die Transistordimensionen der entsprechenden Gatter bleiben von den folgenden Optimierungsschritten unbeeinflusst. Bei Schaltungen mit mehreren Ausgängen existiert zu jedem Ausgang ein kritischer Pfad. Geht man davon aus, dass alle Ausgänge in Register münden, so ist es sinnvoll, die Pfade zu den jeweiligen Ausgängen nur bezüglich des zugehörigen kritischen Pfades zu optimieren. Nach dem Erstellen der Liste der Abhängigkeiten beginnt das Programm von jedem Ausgang her die Schaltung abzuarbeiten. Dabei wird mit dem Ausgang begonnen, zu dem der längste kritische Pfad führt. Von jedem Knoten aus wird nun derjenige Vorgängerknoten aufgesucht, der gemäß der Liste der Abhängigkeiten die größte Pfadverzögerung aufweist. Auf diese Weise wird von jedem Ausgang aus die "längste Spur" bis zum Eingang zurückverfolgt. Die Funktionen für $\tau_p$ und $P_p$ dieses Pfades werden nach den Gleichungen (5.17) und (5.25) aufgestellt. Gilt für die Verzögerungszeit des Pfades $\tau_p < \tau_{\text{krit}}$, so erfolgt die Optimierung des Pfades gemäß (5.31). Das im Rahmen dieser Arbeit realisierte Programm verwendet dazu eine MATLAB-Routine [33]. Alle Gatter eines optimierten Pfades werden wiederum als "fest" markiert und bleiben in den weiteren Schritten unverändert. Dieser Vorgang wird für alle Pfade, die nichtmarkierte Knoten enthalten, solange wiederholt, bis alle Knoten in der Teilschaltung mit dem betrachteten Ausgang markiert sind. Die Abarbeitung der Ausgänge erfolgt bezüglich der Länge der zugehörigen kritischen Pfade in absteigender Reihenfolge. Schließlich erfolgt die Ausgabe der Netzliste der optimierten Schaltung. Im Folgenden ist eine strukturierte Formulierung des Algorithmus' **OpTRAN** (**Op**timiere **TRAN**sistordimensionen) angegeben.

**Algorithmus OpTRAN:**

**BEGINN**
Lies Netzliste und $w$ ein.
Lege alle Eingangsknoten der Schaltung in der Menge **P** ab.
Erstelle die Liste der Abhängigkeiten **LA**:
**FÜR** $v \in$ **P**
    wähle ein $n_v$.
    $\delta(n_v) = d(n_v) + \delta_{\max}(v)$.
    **FALLS** $n_v \notin$ **LA**:
        schreibe $n_v$, $d(n_v)$ in **LA**,
        schreibe $\delta_{\max}(v)$ in $\mathbf{LV}_n$.
    **SONST: FALLS** $v \notin \mathbf{LV}_n$:
        schreibe $\delta_{\max}(v)$ in $\mathbf{LV}_n$.
    Entferne $v$ aus **P** und schreibe alle $n_v$ in **P**, die noch nicht in **P** stehen.
Bestimme $\Delta(n_v)$ für alle $n_v \in$ **LA**.
Bestimme $\tau_{\text{krit}} = \max_{n_v} \left( d(n_v) + \Delta(n_v) \right)$ für alle Ausgänge der Schaltung.
Markiere alle Knoten in den längsten Pfaden.
**FÜR** alle Ausgänge in absteigender Reihenfolge bezüglich der Länge der kritischen Pfade:
    **WIEDERHOLE** solange nichtmarkierte Knoten existieren:
        Beginne am Ausgangsknoten, gehe jeweils zum nichtmarkierten

Vorgängerknoten mit dem größten $\delta(v)$.
Bestimme für diesen Pfad $p$ $\tau_p(\boldsymbol{W}, \boldsymbol{L})$ und $P_p(\boldsymbol{W}, \boldsymbol{L})$.
**FALLS** $\tau_p < \tau_{\text{krit}}$:
Optimiere Pfad $p$ gemäß
$\min_{(\boldsymbol{W},\boldsymbol{L})\in \mathrm{R}}\ (S(W,L) = w(\tau_p - \tau_{\text{krit}})^2 + (1-w)P_p)$.
Markiere alle Knoten im Pfad $p$.
Modifiziere die Netzliste und gib die Netzliste der optimierten Schaltung aus.
**ENDE**

Als Beispiel soll die Schaltung in Bild 5.10a so optimiert werden, dass alle Pfade die gleiche Länge haben. Der Wichtungsparameter sei $w = 1$. Zunächst wird die Liste der Abhängigkeiten erstellt, siehe Tabelle 5.2. Die Verzögerungszeit des kritischen Pfades ist $\tau_{\text{krit}} = \max_{n_v}(d(n_v) + \Delta(n_v)) = d(8) + \Delta(6) = 2 + 6 = 8$. Die Gatter im kritischen Pfad $1 \longrightarrow 3 \longrightarrow 6 \longrightarrow 8$ sind im Bild grau gezeichnet. Im nächsten Schritt erfolgt der Angleich der Verzögerungszeit des Pfades $2 \longrightarrow 4 \longrightarrow 6 \longrightarrow 8$. Das Ergebnis ist in Bild 5.10b dargestellt. Schließlich wird noch die Verzögerungszeit des Pfades $5 \longrightarrow 7 \longrightarrow 8$ angeglichen. Bild 5.10c zeigt die fertig optimierte Schaltung. Natürlich können durch diese Methode nicht alle sensibilisierbaren Pfade einer Schaltung ausgeglichen werden. Im gezeigten Beispiel können die Eingangssignale der Gatter 4 und 7 nicht synchronisiert werden, da nur jeweils in einem der einlaufenden Pfade zu diesen Gattern ein vorhergehendes Gatter vorhanden ist. Der jeweils andere Pfad kann also nicht durch Transistordimensionierung verzögert werden. In diesem Fall kann es sinnvoll sein, zusätzliche Verzögerungselemente einzuführen.

| Knoten $n_v$ | Delay $d(n_v)$ | Liste $\mathbf{LV}_n$ der Vorgänger $v[\Delta_{\max}(v)]$ |
|---|---|---|
| 1 | 2 | |
| 2 | 2 | |
| 3 | 2 | 2(1) |
| 4 | 1 | 2(2) |
| 5 | 2 | |
| 6 | 2 | 4(3), 3(4) |
| 7 | 1 | 2(5) |
| 8 | 2 | 6(6), 3(7) |

Tabelle 5.2. Liste der Abhängigkeiten für das Beispiel in Bild 5.10.

Anhand einiger Anwendungsbeispiele soll nun die Wirkung der hier vorgestellten Methode zur Reduzierung der Glitch-Aktivität mittels Transistordimensionierung untersucht werden. Um den Einfluss der Transistordimensionierung auf die Signallaufzeiten zu demonstrieren, wird zunächst ein konstruiertes Schaltungsbeispiel betrachtet. Bild 5.11 zeigt die dazu verwendete Testschaltung. Bei Verwendung eines Zero-Delay-Modells ist der Ausgang dieser Schaltung immer null, unabhängig vom Eingang. In der realen Schaltung treten jedoch am Ausgang Glitches auf, die durch unterschiedliche Laufzeiten im Innern der Schaltung verursacht werden. Das Ausgangssignal $Y$ eines UND-Gatters der nichtbalancierten Schaltung (alle Transistoren mit minimalen Abmessungen) ist im Bild 5.12a dargestellt. Den Verlauf des Signals $Y$ nach der Optimierung der Schaltung zeigt Bild 5.12b. Dabei wurde die Schaltung in beiden Fällen unter Verwendung der gleichen Eingangsdaten mit SPICE simuliert.

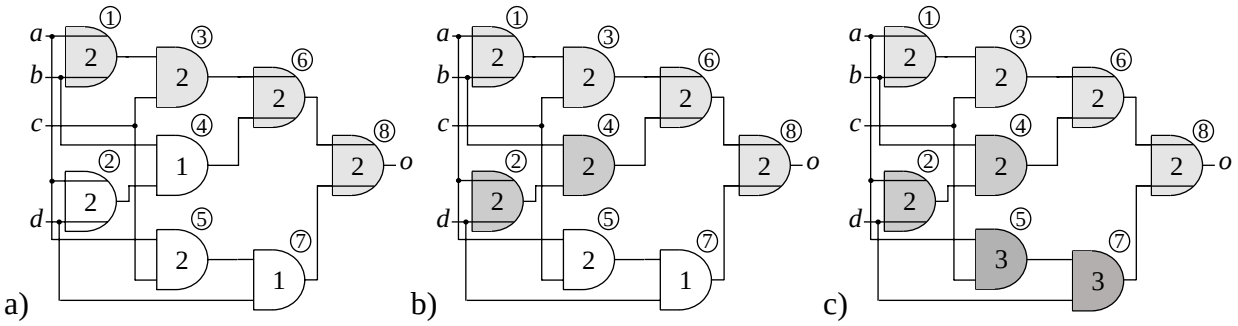



Bild 5.10. Beispiel für die Arbeitsweise von Algorithmus **OpTRAN**. a) Markiere den kritischen Pfad. b) Optimiere die nichtmarkierten Gatter in Pfad 2 ⟶ 4 ⟶ 6 ⟶ 8 und markiere die Gatter. c) Optimiere die nichtmarkierten Gatter in Pfad 5 ⟶ 7 ⟶ 8 und markiere die Gatter.

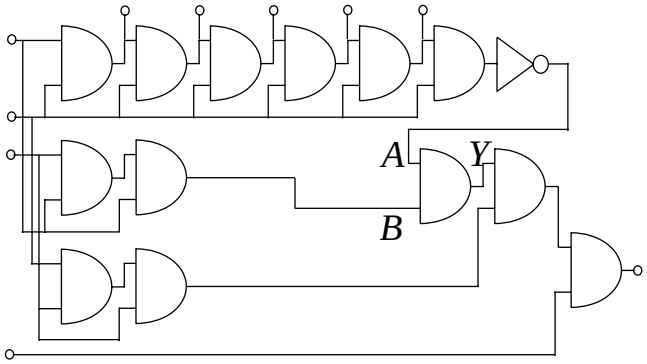



Bild 5.11. Testschaltung.

Wie Bild 5.12b zeigt, lassen sich durch die Optimierung die Glitches komplett eliminieren. Die Auswirkung auf den gesamten Leistungsverbrauch der Schaltung ist in der ersten Zeile von Tabelle 5.3 dokumentiert, wobei die Schaltung dort mit "Test" bezeichnet wurde. Die

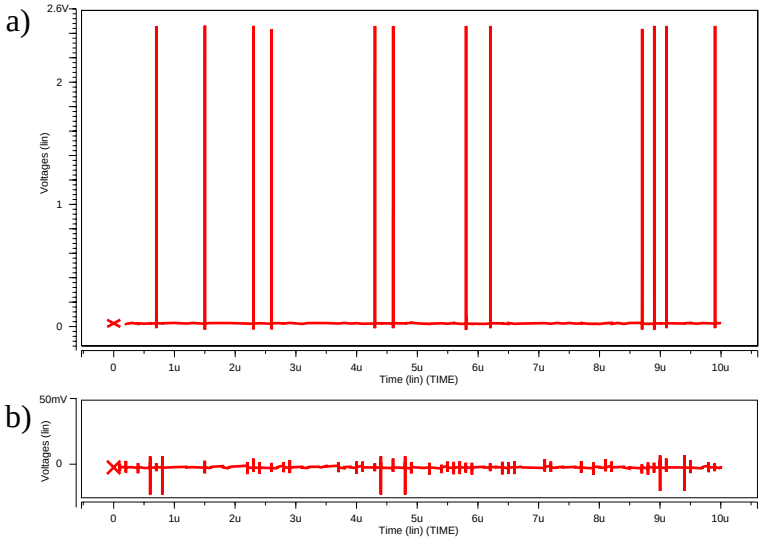



Bild 5.12. Verlauf des Signals $Y$ der Schaltung aus Bild 5.11. a) Ohne Balancieren der Pfade. b) Nach Balancieren der Pfade durch Transistordimensionierung mit **OpTRAN**. Die Achsen für die Spannung sind unterschiedlich skaliert.

Verläufe der Eingangssignale $A$ und $B$ des betrachteten UND-Gatters sind in Bild 5.13 dar-

gestellt. Der obere Bildteil zeigt einen Ausschnitt des Signalverlaufs vor dem Balancieren der Pfade. Im unteren Bildteil ist zu erkennen, wie das ursprünglich früher eintreffende Signal $B$ durch die Transistordimensionierung verzögert wurde. Die Flanken beider Signale treffen nun annähernd gleichzeitig am Eingang des UND-Gatters ein. Dadurch werden Glitches am Ausgang $Y$ vermieden.

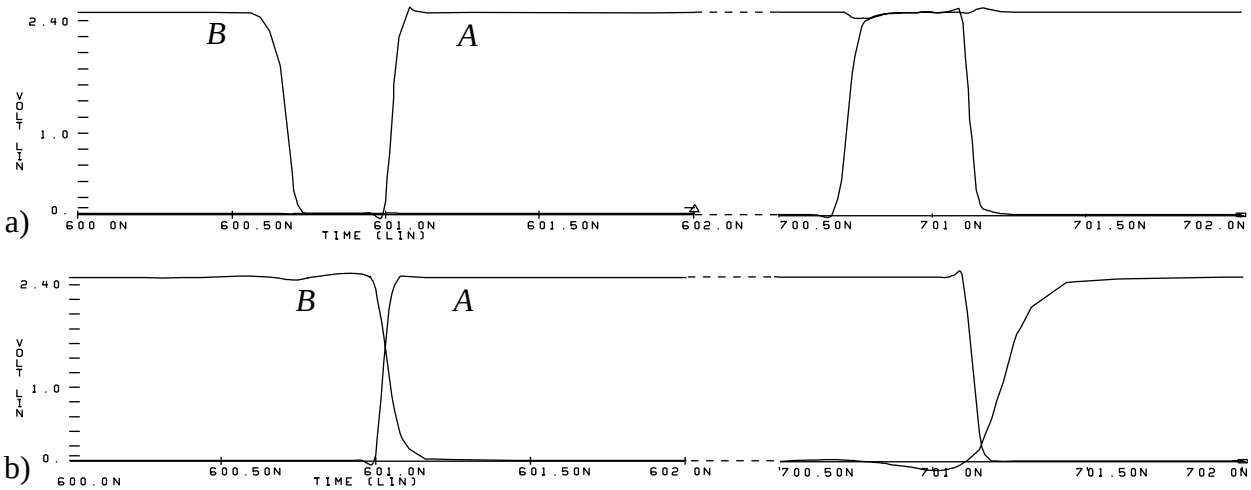



Bild 5.13. Verläufe der Signale $A$ und $B$ der Schaltung aus Bild 5.11. a) Nichtbalancierte Schaltung. b) Balancierte Schaltung.

In Tabelle 5.3 sind weitere Schaltungsbeispiele aufgeführt. Es handelt sich dabei um Array-Multiplizierer unterschiedlicher Größe. Alle in der Tabelle aufgelisteten Schaltungen wurden mit PowerMill [19] simuliert. Als Eingangsdaten dienten jeweils 10000 zufällige Testvektoren. Der Flächenzuwachs durch die Transistordimensionierung wurde an den einzelnen Grundgattern bestimmt und dann auf die Gesamtschaltung hochgerechnet. Die Simulationen der Schaltungen erfolgten unter Verwendung einer $0.25\mu m$ Zellbibliothek. Bei den verwendeten Multiplizierern wirkt sich eine Optimierung besonders stark aus, da in diesen Schaltungen viele unterschiedlich lange Pfade existieren und viele Glitches auftreten. Die Dauer der gesamten Optimierung für einen $16 \times 16$-Multiplizierer dauert auf einer Workstation (Sun Ultra Sparc 1) circa 7 Minuten.

| Schaltung | nicht balanciert | balanciert | Leistungs-einsparung | Flächen-zuwachs | maximale Kanallänge ($\mu m$) |
|---|---|---|---|---|---|
| Test | 0.187 | 0.172 | 8% | 5% | 1.15 |
| $4 \times 4$ Mult. | 0.914 | 0.620 | 32.0% | 15% | 1.67 |
| $8 \times 8$ Mult. | 4.12 | 2.37 | 42.4% | 19% | 2.87 |
| $16 \times 16$ Mult. | 19.29 | 10.49 | 45.6% | 31% | 4.5 |

Tabelle 5.3. Vergleich des Leistungsverbrauchs verschiedener Schaltungen vor und nach dem Balancieren der Pfade. Die Leistungsangaben sind in $mW$. Die Schaltungen wurden auf eine $0.25\mu m$-Technologie mit $2.5V$ Betriebsspannung abgebildet. In der Spalte ganz rechts sind die größten in der jeweiligen Schaltung vorkommenden Kanallängen angegeben. Die minimale Kanallänge ist $0.25\mu m$, alle Kanalweiten sind minimal, hier also $0.35\mu m$.

Die hier vorgestellte Methode dient zur Verlustleistungsreduktion von kombinatorischen Schaltungen und Schaltungsteile auf Schaltkreisebene. Sie ist in Ergänzung zur Glitch-

Minimierung mittels Retiming und Pipelining einsetzbar. Ein Großteil der Glitch-Aktivität wird durch die Register als Glitch-Barrieren und durch Pfadsynchronisation eliminiert. Zusätzlich können mit der hier vorgestellten Methode nach dem Retiming die Schaltungsteile zwischen den Registern durch Transistordimensionierung balanciert, und somit noch vorhandene Glitches eliminiert werden.

Da durch die vorgestellte Methode einzelne Zellen gleichen Typs unterschiedlich dimensioniert werden, stellt sich die Frage, wie ein solches Verfahren in den Entwurfsablauf einbezogen werden kann. Für einen Standardzellenentwurf kommt das Verfahren nicht in Betracht, da der Entwickler nicht auf die innere Struktur der Standardzellen zugreifen kann. Generell ist die Methode geeignet, um bei manuellem Schaltungsentwurf eine leistungsoptimierte Lösung zu erzielen. Allerdings ist der erhöhte Realisierungsaufwand, der durch das manuelle Verändern der Transistorabmessungen verursacht wird, nicht immer zu rechtfertigen. Als ideal erweisen sich deshalb parametrisierbare Grundzellen in Verbindung mit Modulgeneratoren. Bei geeigneten Zellenlayouts ist es immer möglich, Parameter für die Transistoren einzuführen, mit denen die Kanalweiten und -längen entsprechend gedehnt oder gestaucht werden können. Normiert man dabei alle Längen und Weiten auf jeweils ein Bezugsmaß, so ist es möglich, mit nur zwei Parametern, $W$ und $L$, die Zelle entsprechend den Forderungen aus der Optimierung zu verändern. Ein Modulgenerator hat dann anschließend die Aufgabe, die veränderte Zelle wieder in die Gesamtschaltung einzupassen. Moderne CAD-Werkzeuge stellen dafür spezielle Programmiersprachen wie z. B. SKILL (Cadence) zur Verfügung. Mit Hilfe dieser Software-Unterstützung ist die Veränderung der Transistordimensionen im Layout der Schaltung mit erheblich geringerem Zeitaufwand durchführbar.

# 6. Ein VHDL-Modell zur schnellen Bestimmung der Verlustleistung auf höherer Abstraktionsebene

In diesem Kapitel wird eine Methode zur Bestimmung der Verlustleistung vorgestellt, die auf der detaillierten Nachbildung von Eigenschaften der einzelnen Gatter in digitalen CMOS-Schaltungen basiert. Die Bestimmung der Modellparameter erfolgt auf der Schaltkreisebene anhand von Simulationen der Layouts der einzelnen Gatter. Die Gatter werden im Folgenden auch als *Basiszellen* bezeichnet. Beim High-Level-Schaltungsentwurf bilden diese Basiszellen in Form von Standardzellen immer die kleinste dem Schaltungsentwickler verfügbare Einheit. Auch beim manuellen Entwurf wird zumeist auf einen modularen Aufbau geachtet, bei dem einmal entworfene Basiszellen immer wieder verwendet werden. Das Platzieren der einzelnen Elemente im Layout einer Schaltung geschieht also auch hier auf Gatterebene. Da die Gatter einer Schaltung als Basiszellen in einer Bibliothek abgelegt sind und im Allgemeinen nicht verändert werden, ist es naheliegend, bei der Modellierung die innere Struktur der Gatter als gegeben anzusehen und nur das äussere Klemmenverhalten nachzubilden. Dies ermöglicht bei der Verifikation der Schaltung mittels Simulation den Übergang von der Schaltkreisebene auf die Gatterebene.

Im Folgenden werden die Motivation für eine Methode zur Verlustleistungsabschätzung auf höherer Abstraktionsebene und die Ziele eines solchen Verfahrens dargelegt. Im weiteren werden die einzelnen Schritte für die Modellierung der Basiszellen beschrieben, insbesondere die Extraktion der für die Verlusteistungsanalyse wichtigen Modellparameter und die Formulierung des Modells in der Hardware-Beschreibungssprache VHDL. Schließlich wird das so gewonnene Modell der Basiszellen für die Bestimmung der Verlustleistung größerer Schaltungen verwendet.

## 6.1 Motivation

Am genauesten kann die Verlustleistung mittels einer Simulation auf Schaltkreisebene unter Verwendung entsprechend detaillierter Modelle für die einzelnen Bauteile bestimmt werden. Die Schaltungssimulation, beispielsweise mit SPICE, gilt daher nach wie vor als der Anhaltspunkt für die Genauigkeit anderer Methoden zur Verlustleistungsbestimmung. Die Transientenanalyse bei einem Schaltungssimulator erfolgt durch Berechnung der aktuellen Strom- und Spannungswerte zu jedem Zeitschritt und für jeden Knoten in der Schaltung. Je kleiner die Schrittweite gewählt wird, umso größer ist die Genauigkeit der Ergebnisse. Dies geht jedoch auf Kosten der Simulationszeit. Für Standard-Logikgatter, die eine relativ geringe Anzahl von Transistoren aufweisen, beansprucht eine Schaltungssimulation mit den

heutigen Rechnerleistungen, selbst bei sehr kleinen Schrittweiten im Bereich von Pikosekunden, nur wenige Sekunden Rechenzeit. Dies gilt für die vollständige Analyse eines Gatters, d.h. für die Simulation aller möglichen Eingangskombinationen. Die Simulationszeit für die Verlustleistungsbestimmung von größeren Schaltungen, etwa Multiplizierern, liegt bei einer dafür angemessenen Anzahl von Eingangsdaten bereits im Bereich von Tagen und Wochen und ist daher in den meisten Fällen nicht mehr vertretbar. Um die Verlustleistung größerer Teilschaltungen oder gar Gesamtsysteme verlässlich bestimmen zu können, ist es notwendig, nach neuen Verfahren zu forschen, die weniger Rechenzeit und Speicherplatz beanspruchen.

Bereits in der Einleitung wurden einige Ansätze für die Verlustleistungsbestimmung auf höheren Abstraktionsebenen angesprochen. Dabei unterscheidet man allgemein zwischen simulationsbasierten und wahrscheinlichkeitsbasierten Methoden. Der Vorteil der wahrscheinlichkeitsbasierten Methoden liegt in der im Allgemeinen geringeren Rechenzeit. Der entscheidende Nachteil dieser Verfahren ist allerdings der Mangel an Genauigkeit. Die Genauigkeit hängt hier zudem oftmals von der zu untersuchenden Schaltung ab [90]. Eine im Allgemeinen höhere Genauigkeit und universelle Einsetzbarkeit bieten hingegen simulationsbasierte Methoden. Hier kann mehr Rechenzeit zur Ermittlung der Verlustleistung nötig sein als bei den wahrscheinlichkeitsbasierten Methoden. Im Einzelfall muss der Anwender die Vor- und Nachteile beider Verfahren abwägen. Häufig kann jedoch eine etwas längere Simulationszeit für eine höhere Genauigkeit in Kauf genommen werden, da die Anzahl der zu vergleichenden alternativen Schaltungsvarianten begrenzt ist, und somit die Simulationsdauer nur einen geringen Teil zur Gesamtdauer der Systementwicklung beiträgt.

Ein Großteil des modernen Schaltungsentwurfs erfolgt mittels High-Level-Synthese. Im Gegensatz zum *Full Custom Design* bei manuellem Entwurf spricht man hier vom *Semi Custom Design*. Im allgemeinen Sprachgebrauch bezeichnet man den Entwurfsablauf in diesem Fall als *Semi Custom Design Flow*. Hierbei kann im Gegensatz zum Full Custom Design die innere Beschaffenheit der Basiszellen grundsätzlich nicht vom Systementwickler beeinflusst werden. Geht man von einem solchen Semi Custom Design Flow aus, so liegt die Schaltungsbeschreibung in Form einer Hardware-Beschreibungssprache vor. Bild 6.1 zeigt eine schematische Darstellung des Entwurfsablaufs. Die Entwurfsschritte in Algorithmen- und Architekturebene sind unter dem Begriff "Architekturentwicklung" zusammengefasst. Für die gebräuchlichsten Hardwarebeschreibungssprachen Verilog und VHDL gibt es spezielle Simulatoren, die von den Herstellern der Synthesewerkzeuge angeboten werden. Die Beschreibungen der Basiszellen, die für die Verilog- oder VHDL-Simulation einer Schaltung benötigt werden, liegen in standardisierter Form in Bibliotheken vor. Entsprechende Bibliotheken werden von den Technologieherstellern für die Standardzellen geliefert. Die Simulation einer Schaltung, basierend auf diese Zellenbeschreibungen, erlaubt nur eine sehr grobe Abschätzung der Verlustleistung.

Wünschenswert wäre es, eine genauere Modellierung der Basiszellen zu haben und mit den herkömmlichen, kommerziellen Simulationswerkzeugen für VHDL oder Verilog eine verlässliche Aussage über die Verlustleistung der fertigen Schaltung machen zu können. Die Motivation für ein neues Verfahren zur Verlustleistungsbestimmung kann also in den folgenden Punkten zusammengefasst werden:

- Hochsprachen wie VHDL ermöglichen eine detaillierte Beschreibung der Basiszellen mit allen für die Verlustleistungsbestimmung nötigen Parametern. Die Zellbibliothek kann deshalb aus einzelnen VHDL- oder Verilog-Modulen bestehen und muss nicht das spezielle Bibliotheksformat des Syntheseprogrammherstellers aufweisen. Die Modelle sollen

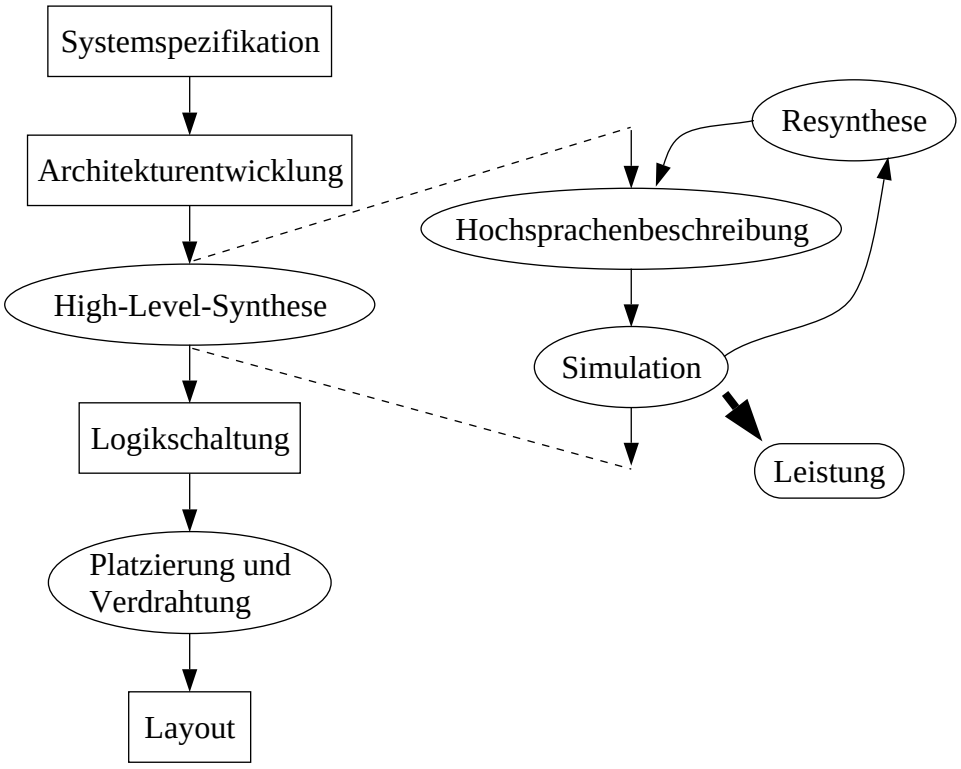



Bild 6.1. Entwurfsablauf.

universell und nicht systemspezifisch für bestimmte CAD-Werkzeuge anwendbar sein. VHDL und Verilog sind standardisiert und erlauben deshalb eine universelle Modellierung.

- Basiszellen können sehr genau durch Schaltungssimulation analysiert werden. Die Simulationsdauer für eine vollständige Analyse ist relativ kurz. Die eigentliche Gesamtschaltung soll dagegen nur durch Logiksimulation auf die Verlustleistung hin untersucht werden.
- Der Entwurf der Schaltung erfolgt durch High-Level-Synthese. Die Schaltung liegt in einer Hochsprachenbeschreibung vor, für die fertige Simulatoren bereits existieren. Deshalb ist es wünschenswert, eine genaue Aussage über die Verlustleistung einer Schaltung durch Simulation mit den herkömmlichen Simulatoren treffen zu können, ohne ein neues Programmwerkzeug zu benötigen.

Durch dieses Verfahren soll es möglich sein, verschiedene Realisierungen von Architekturen in Form von Hochsprachenbeschreibungen noch vor der Logiksynthese auf den Leistungsverbrauch hin zu untersuchen. Die günstigste Realisierung kann somit ermittelt werden, ohne die Schaltung als Gatterlogik aufzubauen. Eventuelle Resyntheseschritte können am Hochsprachen-Code noch vor der Synthese der Logikschaltung durchgeführt werden. In Bild 6.1 ist dies grafisch dargestellt.

Ein wesentliches Merkmal der hier vorgestellten Methode ist die Unterteilung in zwei Phasen:

**Phase 1:** Charakterisierungsphase. Diese beinhaltet die Extraktion der Modellparameter aus Schaltungssimulationen der einzelnen Basiszellen, die Modellierung der Zellumgebung (Lastabhängigkeit der Modellparameter) sowie die Modellbeschreibung jeder Basiszelle durch eine Hardwarebeschreibungssprache.

**Phase 2:** Ausführungsphase. Simulation der Schaltung mit einem Standard-Logiksimulator unter Verwendung der in Phase 1 erstellten Modelle.

Die Charakterisierungsphase muss für alle benötigten Zellen nur einmal durchlaufen werden, d. h. aufwendige Schaltungssimulationen und Modellierungen werden nur einmal pro Zellentyp durchgeführt. Die eigentliche Ermittlung der Verlustleistung beliebiger Schaltungen erfolgt davon getrennt durch einfache Logiksimulation.

Dieser zweigeteilte Ansatz wurde auch schon bei anderen, bereits publizierten Verfahren zur Verlustleistungsbestimmung verwendet. In [72] wird eine Methode vorgestellt, bei der Schaltungsteile zunächst durch Simulation mit Zufallsstimuli charakterisiert werden. Die Bestimmung der Verlustleistung erfolgt dann durch die Summation aller Schaltaktivitäten an den Knoten der Schaltung, gewichtet mit dem jeweiligen durchschnittlichen Leistungsverbrauch an diesen Knoten. Die Genauigkeit dieses Verfahrens hängt stark davon ab, inwiefern die Annahme gerechtfertigt ist, dass im normalen Betrieb der Schaltung die Signalwechsel an den Gattern die gleiche Zufallsverteilung aufweisen wie dies bei der Charakterisierung angenommen wurde. Ein detaillierterer Ansatz wird in [43] verfolgt, wobei hier der Leistungsverbrauch für jede Ein-/Ausgangskombination der Signale durch Schaltungssimulation ermittelt wird. Nach einer anschließenden Logiksimulation werden dann die Verlustleistungen für alle an den Gattern vorkommenden Ein-/Ausgangskombinationen addiert. Dabei findet ein einfaches Unit-Delay-Modell Anwendung. Eine weitere hierarchische Methode wird in [58] präsentiert. Auch hier wird die gegebene Gesamtschaltung zunächst in kleinere Teilschaltungen partitioniert. Die Teilschaltungen werden dann mit Hilfe von Zustandsübergangsdiagrammen modelliert. Die Bestimmung der Leistungswerte für jeden Zustandswechsel erfolgt durch SPICE-Simulationen (Charakterisierungsphase). Allerdings wird hierbei nicht berücksichtigt, dass die Zustandswechsel durch verschiedene Eingangssignalwechsel hervorgerufen werden können, für die sich auch der Leistungsverbrauch der Teilschaltung unterscheidet. Die Bestimmung der Schaltaktivitäten erfolgt durch statistische Methoden. Schließlich wird die Gesamtleistung der Schaltung durch Addition der Leistungen der Teilschaltungen ermittelt.

Alle diese Verfahren benötigen eine eigene spezielle Simulations-Software für die Ausführungsphase. Der Anwender muss sich also mit einem neuen CAD-Werkzeug vertraut machen und besitzt zudem bei keinem dieser Verfahren die Möglichkeit, die Schaltung bereits anhand des VHDL- oder Verilog-Codes auf die Verlustleistung hin zu untersuchen. Das hier vorgestellte Verfahren soll dagegen dem Entwickler die Möglichkeit bieten, die Ermittlung der Verlustleistung innerhalb seiner gewohnten Entwurfsumgebung durchführen zu können.

## 6.2 Bestimmung der Modellparameter

Das Modell soll die Eigenschaften, die für die Bestimmung der Verlustleistung relevant sind, für jede Basiszelle nachbilden. Diese Eigenschaften werden durch Modellparameter beschrieben. Ein Modellparameter der dabei berücksichtigt werden muss, ist natürlich die Verlustleistung der Basiszelle. Aber auch weitere Parameter sind von Bedeutung. Um die Schaltaktivität einschließlich der Glitches durch eine Logiksimulation der Gesamtschaltung möglichst genau bestimmen zu können, müssen auch die Verzögerungszeiten der Gatter berücksichtigt werden. Bei komplexeren Basiszellen, wie z. B. Volladdierern, kann es sogar zu ungewollten Signalwechseln am Ausgang kommen, obwohl am Eingang alle Signalflanken gleichzeitig ankommen. Dieser Effekt ist eine Folge von unterschiedlich langen Signalpfaden innerhalb der Zelle. Glitches, die dadurch verursacht werden, sollen ebenfalls in der Modellierung berück-

sichtigt werden. Schließlich ist es für den Leistungsverbrauch auch entscheidend, welche Last vom Ausgang einer Zelle getrieben werden muss. Die Ausgangslast einer Zelle hängt immer von der Eingangskapazität der folgenden Zellen ab. Es ist deshalb nötig, ein Maß für die Eingangskapazität einzuführen und diesen Parameter für die vorhergehende, treibende Zelle zugänglich zu machen. In den folgenden Abschnitten werden die einzelnen Modellparameter im Detail behandelt.

### 6.2.1 Verzögerungszeitmodell

Die Verzögerungszeit eines CMOS-Gatters hängt von verschiedenen Faktoren ab. In Kapitel 5 wurde die Verzögerungszeit in Abhängigkeit von den Transistorabmessungen berechnet. Bei der quantitativen Bestimmung der Verlustleistung soll jedoch nur eine bestimmte Schaltung mit einer festen Dimensionierung betrachtet werden. Die Transistorabmessungen sind hier nicht variabel und als Modellparameter in diesem Fall nicht interessant. Die Faktoren, die bei der Zellenmodellierung für die Verlustleistungsbestimmung auf höherer Abstraktionsebene von Bedeutung sind, ergeben sich aus der äusseren Beschaltung der Zellen. Als Zellen oder Basiszellen werden hier immer einzelne Logikgatter oder Volladdierer bezeichnet. Die Verzögerungszeit einer Zelle hängt vom Zustandswechsel am Eingang und von der kapazitiven Last am Ausgang ab. Die Bestimmung der kapazitiven Last kann jedoch nicht für jede Zelle isoliert erfolgen, sondern erst im eingebauten Zustand, d. h. wenn die Umgebung des Gatters in der Gesamtschaltung bekannt ist. Die Abhängigkeit vom Zustandswechsel am Eingang kann für jede Zelle getrennt modelliert werden. Je nach Eingangsübergang kann die Verzögerungszeit einer Zelle stark variieren. Die Zahl $M$ der möglichen Eingangsübergänge an einer Zelle mit $n$ Eingängen ergibt sich zu

$$M = 2^{2n} - 2^n \quad . \tag{6.1}$$

Tabelle 6.1 zeigt die Verzögerungszeiten für die verschiedenen Eingangsübergänge an einem UND-Gatter mit zwei Eingängen bei Belastung mit einem minimal dimensionierten Volladdierer. Die Zahl der möglichen Eingangsübergänge ist in diesem Fall ($m = 2$) $2^4 - 2^2 = 12$. Die Zahl $M$ aller möglichen Eingangsübergänge setzt sich zusammen aus der Zahl $M_0$ der Eingangsübergänge, bei denen sich der Ausgang der betrachteten Basiszelle nicht ändert und der Zahl $M_\mathrm{T}$ der Eingangsübergänge, bei denen sich der Ausgang ändert. Für die Bestimmung der Verzögerungszeit müssen nur diejenigen Übergänge beachtet werden, bei denen sich auch der Ausgang ändert. Dies reduziert im Fall des UND-Gatters die Anzahl der zu betrachtenden Übergänge von $M = 12$ auf $M_T = 6$. Die Durchschnittsverzögerungszeit des UND-Gatters beträgt $0.67ns$. Die prozentuale Schwankung beträgt nach oben 25.4%, nach unten 31.3%. Hieraus wird ersichtlich, dass die Verwendung des Durchschnittswertes als repräsentative Verzögerungszeit nur eine relativ ungenaue Modellierung zulässt. In dem Verzögerungszeitmodell der Zellen werden deshalb alle möglichen Eingangsübergänge separat berücksichtigt, bei denen am Ausgang eine Änderung des Zustandes auftritt. Für Basiszellen mit einer großen Anzahl von Eingängen ist diese Vorgehensweise problematisch, da sich nach Gleichung 6.1 ein exponentieller Anstieg der Anzahl der möglichen Eingangsübergänge ergibt. Die Grafik in Bild 6.2 veranschaulicht diese Tatsache. Wird bei der Modellierung für jeden Übergang der zugehörige Wert der Verzögerungszeit gepeichert, so ergeben sich im Fall einer größeren Anzahl von Eingängen sehr komplexe Zellenmodelle, die schwierig handhabbar sind und ausserdem viel Speicherplatz benötigen. Typischerweise besitzt jedoch

| $u_1$ | $u_0$ | y | Verzögerung $(ns)$ |
|---|---|---|---|
| 0 | 0 | 0 | - |
| 1 | 1 | 1 | 0.84 |
| 1 | 0 | 0 | 0.65 |
| 1 | 1 | 1 | 0.72 |
| 0 | 1 | 0 | 0.63 |
| 1 | 1 | 1 | 0.74 |
| 0 | 0 | 0 | 0.46 |

Tabelle 6.1. Verzögerungszeiten eines UND-Gatters mit den Eingängen $u_0$ und $u_1$ und dem Ausgang $y$ für verschiedene Eingangsübergänge. Der Ausgang des UND-Gatters ist mit einem Volladdierer belastet. Prozessdaten: $0.25\mu m$, $2.5V$.

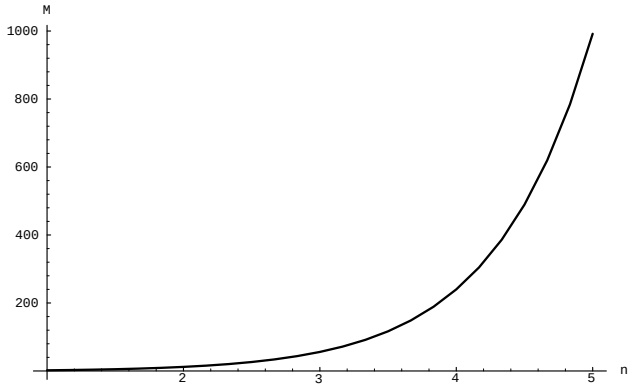



Bild 6.2. Anzahl der möglichen Eingangsübergänge $M$ in Abhängigkeit von der Anzahl $n$ der Eingänge einer Zelle.

eine Basiszelle zwischen zwei und fünf Eingänge. Dabei bleibt die Anzahl der möglichen Eingangsübergänge immer noch handhabbar. Im Fall einer größeren Anzahl von Eingängen können Übergänge, bei denen sich die Verzögerungszeit der Zelle nur um einen kleineren Betrag unterscheidet, zu einer Gruppe von Übergängen zusammengefasst werden. Für diese Gruppe muss nur eine einzige repräsentative Verzögerungszeit zur Modellierung gespeichert werden.

Die Verzögerungszeit von CMOS-Gattern hängt ausser von den Eingangsdaten auch von der Ausgangsbelastung ab. Der Ausgang eines Gatters kann mit den Eingängen mehrerer nachfolgender Gatter verbunden sein. Die Eingänge von Standard-CMOS-Gattern werden von den Gates der inneren Transistoren gebildet, wenn man von Pass-Transistor-Logiken absieht, bei denen auch Transistorkanäle im Signalpfad liegen können. Die Gates von MOS-Transistoren lassen sich als Kapazitäten auffassen, der Eingangswiderstand der Transistoren ist im Idealfall unendlich. Hinzu kommen noch die parasitären Kapazitäten der Verbindungsleitungen zwischen den Gattern. Es ist also gerechtfertigt, die Ausgangslast eines Gatters als rein kapazitiv anzusehen. Um die Modellierung möglichst allgemein zu halten, muss bei der Charakterisierung der Basiszellen die Abhängigkeit der Verzögerungszeit von der Lastsituation ermittelt und im Modell berücksichtigt werden. Dabei kann ohne Einschränkung der Allgemeinheit die kapazitive Ausgangslast zunächst auf die minimal mögliche Last normiert werden. Als minimal mögliche Last wird die Gatekapazität eines NMOS-Transistors mit minimaler Kanalweite und -länge angenommen. Die Gatekapazität $C_{\mathrm{g}}$ ist direkt pro-

portional der Fläche $A_g$ und einer technologieabhängigen Konstanten $c_g$, $C_g = c_g \cdot A_g$. Die Konstante $c_g$ ist für alle Transistoren der Schaltung gleich, die Normierung erfolgt daher lediglich auf die Fläche der Gates. Die normierte Fläche $a_g$ ergibt sich aus dem Quotienten der Gate-Fläche $A_g$ und der minimalen Fläche $A_{g,min} = L_{min} \cdot W_{min}$, $a_g = \frac{A_g}{A_{g,min}}$. Durch diese Normierung vereinfacht sich die analytische Berechnung der lastabhängigen Verzögerungszeit der Basiszellen.

Bild 6.3 verdeutlicht die Lastabhängigkeit der Verzögerungszeit für das Beispiel eines Volladdierers. Die Verzögerungszeiten für die verschiedenen Lastsituationen wurden mit SPICE-Simulationen ermittelt. Anhand der Grafik von Bild 6.3 ist ersichtlich, dass zu ei-

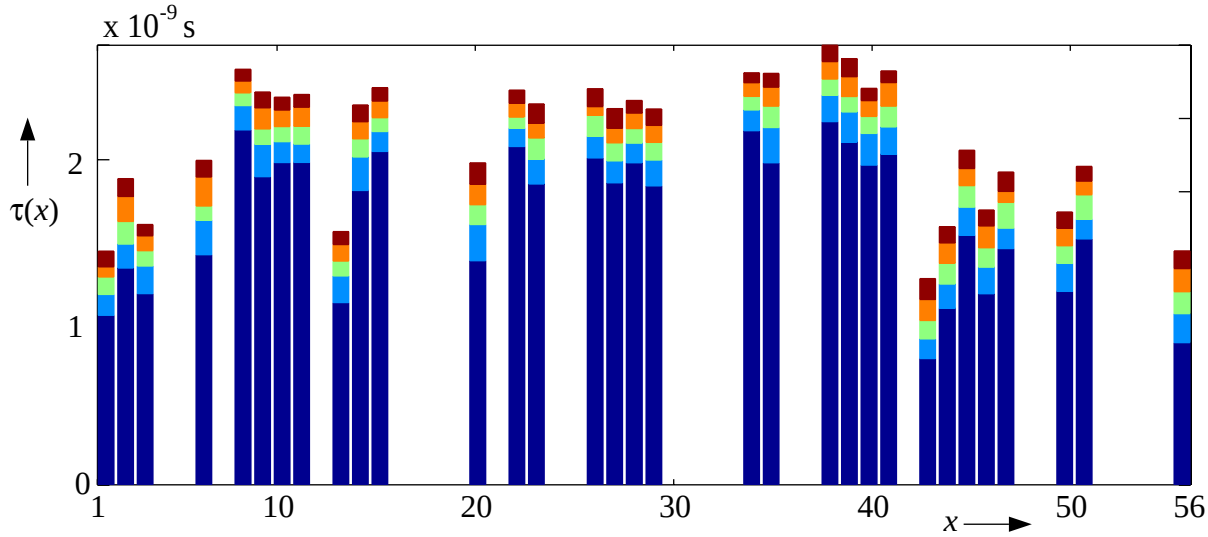



Bild 6.3. Verzögerungszeiten $\tau(x)$ des Summenpfades einer Volladdiererzelle bei unterschiedlicher Belastung. Auf der x-Achse sind alle möglichen Eingangsübergänge nummeriert von $x = 1 \ldots 56$ aufgetragen. Die unterste Balkenreihe zeigt die Verzögerungszeiten ohne Last. Darüber angetragen sind jeweils die Werte für Belastung mit einem, zwei, drei und vier Volladdierern parallel am Ausgang.

ner Grundverzögerungszeit ohne Last für jeden weiteren parallelen Volladdierer am Ausgang ein gleich großer Betrag hinzukommt. Allgemein steigt dieser additive Anteil linear mit der Ausgangslast. Anhand des Elmore-Delay-Modells ist dieser Zusammenhang leicht ersichtlich. Die Verzögerungszeit eines NAND-Gatters mit einer Lastkapazität $C_L$ berechnet sich zu (vergleiche Bild 6.4):

$$\tau = 2C_{d/s_N}R + 2R(C_{d/s_N} + 2C_{d/s_P}) + 2RC_L = 8RC_{d/s} + 2RC_L \quad , \tag{6.2}$$

wobei hier angenommen wurde, dass $R_N = R_P = R$ und $C_{d/s_N} = C_{d/s_P} = C_{d/s}$. Der erste Term in (6.2) ist lastunabhängig, der zweite Term hängt linear von der Lastkapazität ab. Allgemein kann die Verzögerungszeit $\tau$ einer Basiszelle aus zwei Anteilen, einem lastunabhängigen und einem lastabhängigen Teil zusammengesetzt werden. Dabei ist es wiederum sinnvoll, jeden Eingangsübergang $x$ separat zu behandeln. Es seien $y(k)$ und $y(k+1)$ jeweils die Ausgangszustände der Basiszelle vor und nach einem Wechsel des Eingangszustandes. Für einen Eingangsübergang $x$ ergibt sich die Verzögerungszeit

$$\tau(x) = \begin{cases} \tau_0(x) + K \cdot \tau_L & \text{falls} \quad y(k+1) \neq y(k) \\ 0 & \text{falls} \quad y(k+1) = y(k) \end{cases} \quad . \tag{6.3}$$

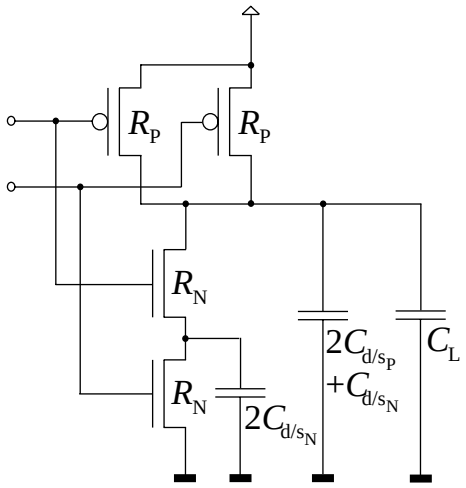



Bild 6.4. NAND-Gatter mit parasitären Kapazitäten.

Dabei ist $\tau_L$ die Verzögerungszeit, die durch Belastung des Ausgangs mit einer "Normlast" zur Gesamtverzögerungszeit hinzukommt. $K$ ist die Anzahl der Normlasten, die der tatsächlichen kapazitiven Belastung des Ausganges entsprechen. Die Definition der Normlast ist grundsätzlich beliebig. Beispielsweise kann ein minimal dimensioniertes Transistorgate, die kleinste vorkommende Lasteinheit, als Normlast gewählt werden. Bei CMOS-Schaltungen ist es jedoch meist sinnvoller, die Kombination eines NMOS-Transistors mit dem dazu komplementären PMOS-Transistor als Normlast zu wählen, also einen minimal dimensionierten Inverter. In diesem Fall entspricht der Faktor $K$ der Anzahl der minimal dimensionierten Inverter, die parallel zum Ausgang angeschlossen die gleiche Last für die Basiszelle darstellen würden, wie die tatsächlich angeschlossenen Gatter. Die Bestimmung von $K$ kann erst erfolgen, wenn die äussere Umgebung der Basiszelle in der Gesamtschaltung bekannt ist. Bereits in der Charakterisierungsphase können für dieses Verzögerungszeitmodell die folgenden Parameter bestimmt werden:

- die Grundverzögerungszeit $\tau_0(x)$ der Basiszelle für alle möglichen Eingangskombinationen,
- das Zeitinkrement $\tau_L$ bei Belastung des Ausganges mit einer Normlast.

Das Zeitinkrement der belasteten Basiszelle ist weitgehend unabhängig vom Eingangsübergang. Dies geht auch aus den Kurven in Bild 6.3 hervor. Bei der Modellierung gilt deshalb die Annahme $\tau_L \neq f(x)$. Die Bestimmung des Zeitinkrements kann auf unterschiedliche Weise erfolgen. Da das Zeitinkrement als von den Eingangsübergängen unabhängig betrachtet werden kann, ist eine Möglichkeit zur Bestimmung von $\tau_L$ die Simulation der Basiszelle mit einem einzigen Eingangsübergang, bei dem sich der Ausgangszustand ändert. Als Ausgangsbelastung wird dabei eine Normlast gewählt. Die Differenz der dabei ermittelten Verzögerungszeit zur Verzögerungszeit ohne Last ergibt $\tau_L$. Eine aufwendigere Methode zur Bestimmung von $\tau_L$ ist die Berechnung eines Durchschnittswertes aus den Ergebnissen von Simulationen mit $B$ verschiedenen Lasten. Die Lasten sind dabei wieder ganzzahlige Vielfache einer Normlast. Für jeden Belastungsfall werden alle $M_T$ Eingangsübergänge simuliert, bei denen sich der Ausgangszustand ändert. Das Zeitinkrement $\tau_L$ berechnet sich gemäß Gleichung 6.4.

$$\tau_L = \frac{1}{M_T(B-1)} \sum_{i=1}^{B-1} \sum_{x=1}^{M_T} (\tau_{i+1}(x) - \tau_i(x)) \tag{6.4}$$

Dabei unterscheidet sich der Belastungsfall $i$ vom Belastungsfall $i+1$ um genau eine Normlast. In Tabelle 6.2 sind die Werte für minimales, maximales und durchschnittliches Zeitinkrement eines minimal dimensionierten Volladdierers aufgelistet. Die Ergebnisse stammen aus der gleichen Simulation wie die Werte für die absoluten Verzögerungszeiten in Bild 6.3. Die Normlast ist in diesem Fall ebenfalls ein minimal dimensionierter Volladdierer.

| | $\tau_{\mathrm{L}}$ in $ps$ |
|---|---|
| minimal | 24.3 |
| maximal | 34.2 |
| durchschnittlich | 31.0 |

Tabelle 6.2. Minimaler, maximaler und durchschnittlicher Betrag, um den die Verzögerungszeit eines Volladdierers bei Belastung mit einem identischen Volladdierer zunimmt.

Die wesentlichen Merkmale dieses Verzögerungszeitmodells lassen sich in den folgenden vier Punkten zusammenfassen:

- Die kleinsten Schaltungseinheiten, die modelliert werden, sind Basiszellen. Sie sind die Grundmodule der Schaltung.
- Alle Simulationen zur Charakterisierung der Basiszellen erfolgen auf Layoutebene.
- Alle möglichen Zustandsübergänge am Eingang einer Basiszelle, für die sich unterschiedliche Verzögerungszeiten ergeben, werden separat betrachtet. Die einzelnen Verzögerungszeiten werden durch Simulation bestimmt und in der Modellbeschreibung den entsprechenden Eingangsübergängen zugeordnet.
- Die Lastabhängigkeit der Verzögerungszeiten wird berücksichtigt. Dazu wird die durch Belastung verursachte zusätzliche Verzögerungszeit in einer Simulation bestimmt und für jeden Eingangsübergang zur Grundverzögerungszeit addiert.

Durch die detaillierte Nachbildung des zeitlichen Verhaltens der Basiszellen ist es möglich, die Schaltaktivität in Schaltungen, die aus diesen Basiszellen bestehen, sehr genau zu erfassen. Glitches, die aufgrund unterschiedlicher Signallaufzeiten in der Schaltung entstehen, können dadurch berücksichtigt werden. Darüberhinaus können jedoch auch Glitches an den Ausgängen von Basiszellen auftreten, obwohl alle Signalwechsel an den Eingängen gleichzeitig erfolgen. Um diese Glitches erfassen zu können, ist eine Betrachtung auf Schaltungsebene notwendig. Dieses Verhalten muss also gegebenenfalls in der Modellbeschreibung der einzelnen Basiszellen berücksichtigt werden.

### 6.2.2 Modellierung von Glitches

Beim Wechsel der Eingangssignale ändert sich in CMOS-Schaltungen gegebenenfalls auch der Pfad, über den der Ausgang auf dem Potential der Masse oder der Versorgungsspannung liegt. Der logische Zustand "0" am Ausgang tritt auf, wenn ein Pfad im NMOS-Block der Schaltung leitend ist, d.h. der Ausgang über die Kanäle der leitenden NMOS-Transistoren auf Massepotential liegt. Entsprechend gilt für den logischen Zustand "1", dass ein Pfad im PMOS-Block der Schaltung leitend ist und der Ausgang auf dem Potential der Versorgungsspannung liegt. Die Pfade innerhalb von NMOS- oder PMOS-Block können unterschiedlich lang sein. Entsprechend können sich unterschiedliche Signallaufzeiten ergeben. Diese internen Laufzeitunterschiede einer Basiszelle können zu Glitches am Ausgang führen, obwohl die

Flanken der Eingangssignale gleichzeitig auftreten. Ein solches Verhalten zeigen insbesondere komplexere Basiszellen, wie beispielsweise bestimmte Typen von Volladdierern. Die Länge der dabei entstehenden Glitches, die gewissermaßen von der Zelle selbst "erzeugt" werden, hängt wieder vom Eingangsübergang und der Ausgangslast ab. In Bild 6.5 ist ein Volladdierer gezeigt, bei dem für bestimmte synchrone Eingangsübergänge am Ausgang Glitches auftreten. Der hier dargestellte Volladdierer ist in [48] näher beschrieben. Die zusätzliche

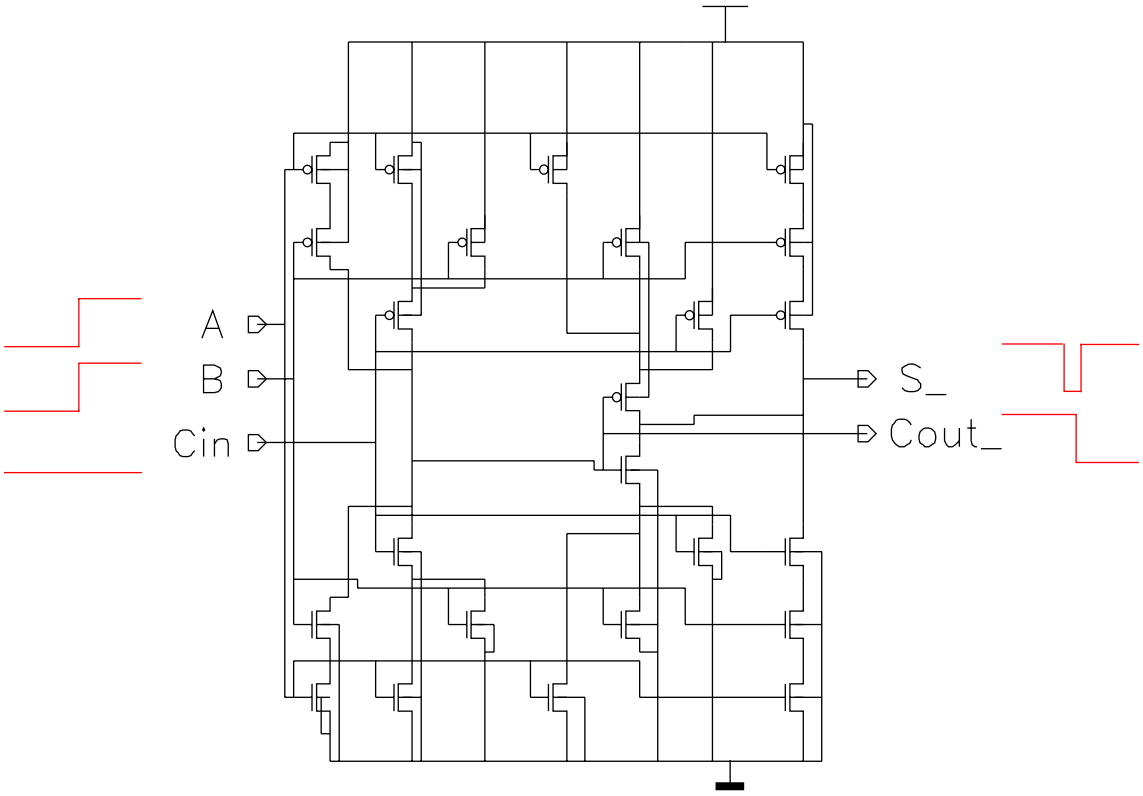



Bild 6.5. Entstehung eines Glitches am Ausgang einer Basiszelle bei synchron einlaufenden Eingangssignalen. Die gezeigte Basiszelle ist ein Volladdierer. Beim Eingangsübergang 000 $\longrightarrow$ 110 tritt am Summenausgang des Volladdierers ein ungewollter Schaltvorgang auf.

Schaltaktivität bei bestimmten Eingangsübergängen kann im Modell mit VHDL beschrieben werden. Dazu wird beim Auftreten des entsprechenden Eingangsübergangs dem Ausgang zunächst der logische Wert des Glitches zugewiesen. Nach der durch Simulation ermittelten Dauer des Glitches folgt schließlich die Zuweisung des korrekten, aus der aktuellen Eingangsbelegung resultierenden Wertes. Bild 6.6 zeigt die Simulation eines Volladdierers mit allen 56 möglichen Eingangsübergängen. Die oberen drei Diagramme zeigen die Verläufe der Eingangssignale A, B und $C_{in}$. Darunter ist der mit SPICE ermittelte Signalverlauf $S_{SPICE}$ des Summenausgangs dargestellt. Das unterste Diagramm zeigt das Signal $S_{model}$ des Summenausgangs für die Logiksimulation des VHDL-Modells der Schaltung. Aus dieser Darstellung ist ersichtlich, dass Glitches, deren Spannungswert in der SPICE-Simulation eine bestimmte Schwelle betragsmäßig überschreitet, im Modell als volle Impulse mit maximalem Spannungshub auftauchen, während betragsmäßig kleinere Glitches unberücksichtigt

bleiben. Die Beschreibung einer Schaltung mit VHDL erlaubt nur volle Spannungspegel, also $0V$ oder $U_B$, jedoch keine Zwischenwerte. Die Schwellenspannung, ab der ein Glitch im Modell berücksichtigt wird, muss wiederum experimentell ermittelt werden. Als sinnvoll erweist es sich, hierfür die Schwellenspannungen $U_{\text{th}}$ der Transistoren zu wählen, da Glitches, die betragsmäßig größer als $U_{\text{th}}$ sind, einen Schaltvorgang in der nachfolgenden Stufe bewirken können.

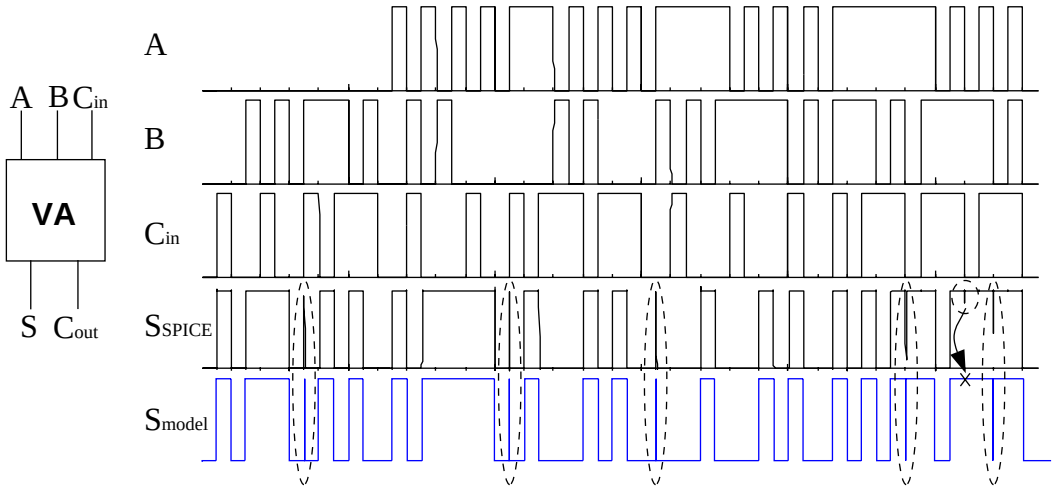



Bild 6.6. Summensignal am Ausgang eines Volladdierers ermittelt durch SPICE-Simulation des Layouts ($S_{SPICE}$) und durch Logiksimulation des VHDL-Modells ($S_{model}$). Die Stellen, an denen Glitches aufgrund des SPICE-Ergebnisses durch das Modell nachgebildet worden sind, sind im Bild markiert. Es ist ausserdem zu erkennen, dass Glitches mit kleiner Amplitude nicht nachgebildet werden.

Im Abschnitt 3.4 wurde gezeigt, dass sehr kurze Eingangsimpulse ein Gatter u. U. nicht oder nur gedämpft passieren können. Die Simulationen verschiedener Gatter haben jedoch gezeigt, dass durchaus auch Impulse, die kürzer als die Verzögerungszeit eines Gatters sind, am Ausgang ungedämpft auftreten. Der von den derzeit aktuellen Synthesewerkzeugen unterstützte VHDL-Standard VHDL'87 beinhaltet nur zwei Modelle, mir der die Verzögerungszeit eines Gatters berücksichtigt werden kann: das sog. *Inertial-* und das sog. *Transport-*Delay-Modell. Beim Inertial-Delay-Modell bleiben solche Eingangsimpulse unberücksichtigt, die kürzer als die Verzögerungszeit des Gatters sind. Gegebenenfalls resultiert dies in einer Unterschätzung der Schaltaktivität bei der VHDL-Logiksimulation. Im Gegensatz dazu können sich bei Verwendung des Transport-Delay-Modells Eingangsimpulse jeder beliebigen Länge auf den Ausgang auswirken. Dies kann zu einer Überschätzung der Schaltaktivität führen, da dieses Modell den Effekt, dass sehr kurze Glitches aufgefiltert werden, nicht berücksichtigt. Für eine möglichst realistische Nachbildung des Verhaltens der Basiszellen ist es wünschenswert, die minimal Länge der noch zu berücksichtigenden Glitches mittels eines Parameters für jede Basiszelle einstellen zu können. Durch eine geeignete VHDL-Programmierung lässt sich dies erreichen. Eine spezielle Programmkonstruktion bewirkt, dass bei der ereignisgesteuerten VHDL-Simulation beim Auftreten eines Ereignisses am Eingang der Basiszelle zunächst eine Zeit $t_{\text{min}}$ gewartet wird, bevor die Übernahme dieses Ereignisses in die Ereignisliste erfolgt. Tritt vor Ablauf dieser Zeit $t_{\text{min}}$ ein neues Ereignis am Eingang auf, so wird das erste Ereignis verworfen und nicht in die Ereignisliste aufgenommen. Dieses Ereignis macht sich somit nicht im Simulationsergebnis bemerkbar. Die Zeit $t_{\text{min}}$

entspricht somit der minimalen Länge, die ein Glitch aufweisen muss, um am Ausgang der Basiszelle eine Wirkung hervorrufen zu können. Der Wert von $t_{\min}$ muss für jede Basiszelle einmal mittels Schaltungssimulation ermittelt werden.

Im Folgenden soll anhand eines Beispiels gezeigt werden, dass mit der hier vorgestellten Modellierung der Glitches eine bessere Abschätzung der Schaltaktivität in einer Schaltung möglich ist, als mit den Standardmodellen. Als Beispielschaltung dient ein $5 \times 5$-Bit Multiplizierer. Die schaltungstechnische Realisierung der verwendeten Volladdierer entspricht der Schaltung in Bild 6.5. Insgesamt sind zum Aufbau des Multiplizierers 16 Volladdierer nötig (ohne der abschließenden Summation des Endproduktes). Nach Gleichung (1.6) ist der Leistungsverbrauch eines Gatters oder einer Zelle, bei Vernachlässigung der Leckströme, proportional zur Schaltaktivität. Das bedeutet, dass, bei geeigneter Modellierung, ein linearer Zusammenhang zwischen den durch Schaltungssimulation ermittelten Leistungsaufnahmen und den durch VHDL-Simulation ermittelten Schaltaktivitäten der einzelnen Volladdierer bestehen muss. Um dies für die verschiedenen Modelle zu überprüfen, wurde das Layout des Multiplizierers mit SPICE simuliert und daraus die Leistungsaufnahme jedes einzelnen Volladdierers ermittelt. Ausserdem wurden VHDL-Simulationen mit den verschiedenen Verzögerungszeitmodellen durchgeführt und damit die Schaltaktivitäten an den einzelnen Volladdierern bestimmt. In Bild 6.7 sind die Schaltaktivitäten über die Leistungen für das hier beschriebene Modell sowie für Inertial- und Transport-Modell aufgetragen. Jeder Punkt in der Graphik entspricht dabei einem Volladdierer an einer bestimmten Position innerhalb des Multiplizierers. Die Ergebnisse zeigen, dass durch die detailliertere Modellierung des Glitch-Verhaltens der Basiszellen genauere Rückschlüsse von der Schaltaktivität auf die tatsächliche Verlustleistung möglich sind. Dies zeigt der näherungsweise lineare Zusammenhang zwischen diesen beiden Größen, der sich bei dieser Modellierung ergibt. Deutlich zu erkennen sind auch die Abweichungen bei Verwendung der Standardmodelle. Die Verwendung des Transport-Modells führt offensichtlich zu einer Überschätzung, die Verwendung des Inertial-Modell dagegen zu einer Unterschätzung der Schaltaktivitäten. Die Abweichungen für eine bestimmte Zelle sind umso größer, je größer die Logiktiefe bis zu dieser Zelle ist. Die Berücksichtigung aller auftretenden Glitches beim Transport-Modell führt zu einem lawinenartigen Anstieg der ermittelten Schaltaktivität mit steigender Logiktiefe. Beim Inertial-Modell wirkt sich das Vernachlässigen kurzer Glitches, die dennoch die Zellen passieren und weitere Glitches hervorrufen können, ebenfalls in größerer Logiktiefe stärker aus.

### 6.2.3 Verlustleistungsmodell

Die detaillierte Modellierung der Verzögerungszeiten und des Glitch-Verhaltens der Basiszellen erlaubt eine genaue Bestimmung der Schaltaktivitäten. Damit ist die erste Voraussetzung für eine zuverlässige Bestimmung der Verlustleistung einer Schaltung geschaffen. Die zweite wichtige Voraussetzung ist die Zuweisung eines genauen Leistungswertes zu jedem Schaltvorgang. Ebenso wie die Verzögerungszeit hängt auch die Verlustleistung einer Basiszelle von den Eingangsdaten ab. Jeder der $2^{2n} - 2^n$ möglichen Eingangsübergänge hat eine bestimmte, im Allgemeinen unterschiedliche Leistungsaufnahme zur Folge. Die Bestimmung der einzelnen Leistungswerte kann gleichzeitig mit der Bestimmung der Verzögerungszeiten durch Schaltungssimulation der Basiszellen erfolgen. Ebenso wie bei der Verzögerungszeit ist es auch hier möglich, die Verlustleistung einer Basiszelle als die Summe aus zwei voneinander unabhängigen Anteilen zu betrachten. Ein Teil der Leistung wird beim Umladen

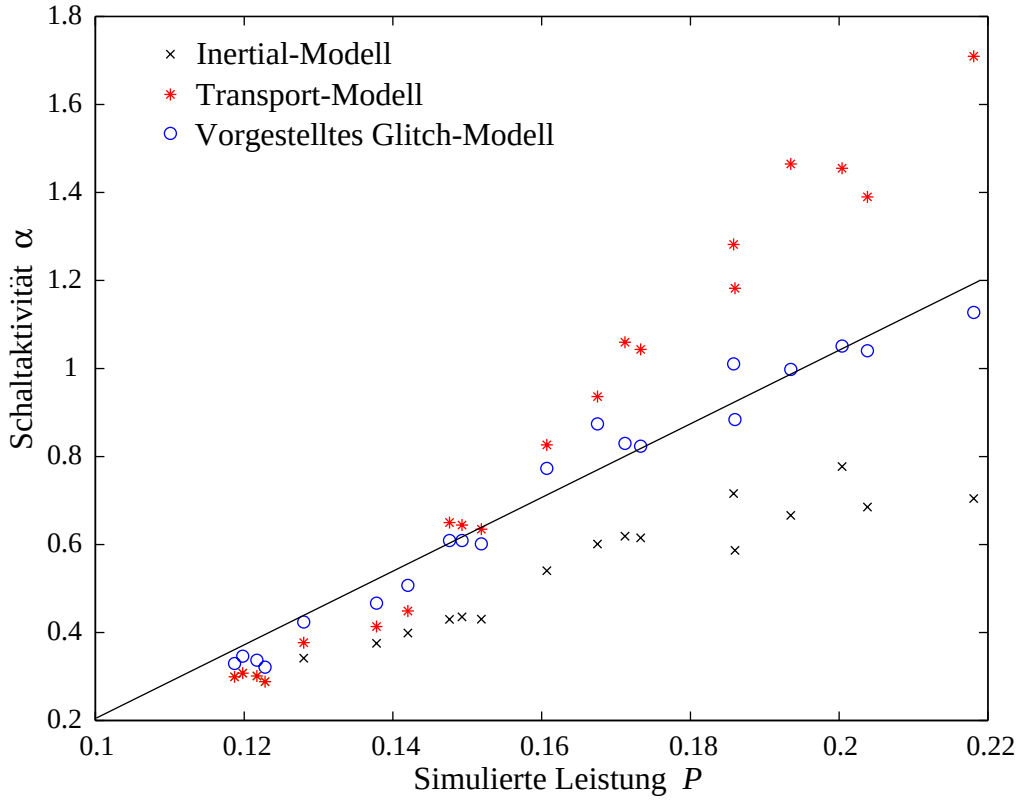



Bild 6.7. Der Zusammenhang zwischen Verlustleistung und Schaltaktivität der einzelnen Volladdierer in einem 5 × 5-Bit Multiplizierer. Die Schaltaktivitäten wurden mit den Standard-VHDL-Verzögerungszeitmodellen und einer genaueren Glitch-Modellierung ermittelt. Im Idealfall ergibt sich ein linearer Zusammenhang zwischen der Verlustleistung und der Schaltaktivität (durchgezogene Linie).

der parasitären Kapazitäten an den inneren Knoten der Basiszelle verbraucht. Betrachtet man den Leistungsverbrauch wieder für alle möglichen Eingangsübergänge getrennt, so kann dieser lastunabhängige Anteil für einen bestimmten Übergang $x$ formuliert werden als

$$P_0(x) = f C_0(x) U_\mathrm{B}^2 \quad . \tag{6.5}$$

Dabei ist $C_0(x)$ die effektive Kapazität, die beim Eingangsübergang $x$ geladen werden muss. Die effektive Kapazität unterscheidet sich im Allgemeinen für verschiedene Eingangsübergänge, ist also abhängig davon, welche Transistoren gerade leitend bzw. gesperrt sind und welche parasitären Kapazitäten dadurch eine Last für die Quelle darstellen. Unabhängig davon kann der Leistungsverbrauch, den eine Last am Ausgang der Zelle zur Folge hat, als zusätzlicher Anteil zum Gesamtleistungsverbrauch angesehen werden. Dieser lastabhängige Anteil lässt sich nach folgender Gleichung berechnen:

$$P_\mathrm{L}(x) = f C_\mathrm{L}(x) U_\mathrm{B}^2 \quad . \tag{6.6}$$

Bei dieser Definition wird auch die Lastkapazität $C_\mathrm{L}(x)$ als eine Funktion der Eingangsdaten angesehen. Somit schließt die Definition den Fall mit ein, dass die Lastkapazität nur im Fall eines Wechsels des Ausgangszustandes umgeladen wird und nur dann Leistung aus der Quelle entnommen wird, wenn der Ausgang vom Zustand "0" nach "1" wechselt. Andernfalls kann

die Lastkapazität als identisch null betrachtet werden, wodurch der lastabhängige Anteil zum Gesamtleistungsverbrauch wegfällt. Eine gleichbedeutende Definition lässt sich auch angeben, wenn die Lastkapazität und entsprechend der lastabhängige Teil der Verlustleistung als unabhängig von den Eingangsdaten betrachtet werden und eine Fallunterscheidung aufgrund des Ausgangszustandes erfolgt. Es seien $y(k)$ und $y(k+1)$ jeweils die Ausgangszustände vor und nach dem Wechsel des Eingangszustandes. Der lastabhängige Teil der Verlustleistung kann somit auch folgendermaßen formuliert werden:

$$P_{\mathrm{L}} = \begin{cases} f C_{\mathrm{L}} U_{\mathrm{B}}^2 & \text{falls} \quad y(k+1) \neq y(k) \\ 0 & \text{falls} \quad y(k+1) = y(k) \end{cases} \quad . \tag{6.7}$$

Gemäß den Definitionen für den lastunabhängigen und den lastabhängigen Anteil nach (6.5) und (6.7), errechnet sich die gesamte Leistungsaufnahme einer Basiszelle für einen bestimmten Eingangsübergang $x$ zu

$$P(x) = P_0(x) + P_{\mathrm{L}} \quad . \tag{6.8}$$

Der lastabhängige Anteil ist, wie bei der Verzögerungszeit, additiv. Verdeutlicht wird dies durch die Simulation einer Basiszelle mit verschiedenen Ausgangslasten. Als Beispiel dient wieder die SPICE-Simulation eines Volladdierers zunächst ohne und dann mit Belastung. Als Last wurden, wie auch schon in Abschnitt 6.2.1 bei der Ermittlung der Lastabhängigkeit der Verzögerungszeit, minimal dimensionierte Volladdierer gleichen Typs verwendet. Bild 6.8 zeigt die entsprechenden Simulationsergebnisse. Bei Eingangsübergängen, die nicht zu einer

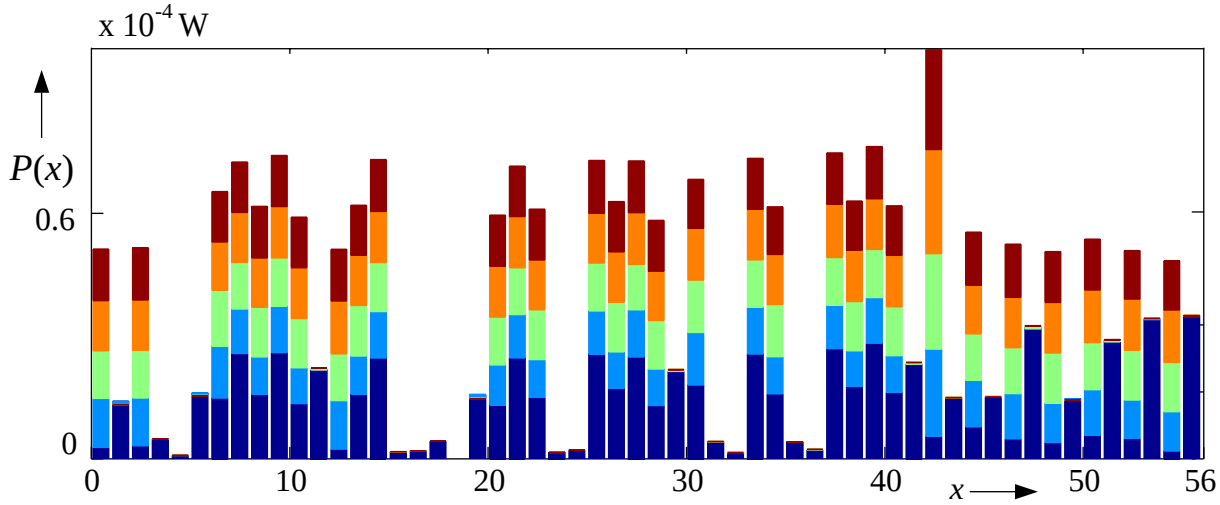



Bild 6.8. Leistungsaufnahme einer Volladdiererzelle bei unterschiedlicher Belastung für alle möglichen Eingangsübergänge $x$ $(1 \leq x \leq 56)$. Die unterste Balkenreihe zeigt den Verlauf ohne Belastung. Darüber angetragen sind jeweils die Werte für Belastung mit einem, zwei, drei und vier Volladdierern parallel zum Summenausgang.

Zustandsänderung des Ausgangs führen, ergeben sich für alle Belastungsfälle die gleichen Verlustleistungen, da in diesem Fall nur der lastunabhängige Anteil $P_0(x)$ zum Leistungsverbrauch beiträgt. Wie aus den Simulationsergebnissen hervorgeht, bewirkt die Belastung des Ausgangs der Zelle einen zusätzlichen Leistungsverbrauch der sich additiv auf den Gesamtleistungsverbrauch auswirkt. Es ist daher sinnvoll, wie im Fall der Verzögerungszeit, eine Normlast zu definieren, deren Beitrag zum Leistungsverbrauch für jede Basiszelle durch

Simulation bestimmt wird. Jede beliebige Last kann als ein Vielfaches $K$ der Normlast ausgedrückt werden.

Geht man von einer Belastung mit $K$ Normlasten aus, so lässt sich die Verlustleistung als die Summe aus der lastunabhängigen Verlustleistung $P_0(x)$ und dem lastabhängigen, auf die Normalast normierten Leistungsanteil $P_\mathrm{L}$, multipliziert mit der Anzahl $K$ der Normlasten darstellen, gemäß

$$P(x) = P_0(x) + KP_\mathrm{L} \quad . \tag{6.9}$$

Die bei der Schaltungssimulation der Basiszelle ermittelten Leistungswerte gelten zunächst nur für jene bestimmte Frequenz, die bei der Simulation als Taktfrequenz verwendet wurde. Die Taktfrequenz, bei der die Leistungswerte ermittelt wurden, ist im Folgenden mit $f_0$ bezeichnet. Zur Ermittlung der Verlustleistung bei beliebigen Frequenzen $f$ müssen die Werte mit einem Faktor $\frac{f}{f_0}$ skaliert werden. Wegen der linearen Abhängigkeit der Verlustleistung von der Frequenz ist diese einfache Skalierung immer möglich. Ein anderer Ansatz basiert auf der Verwendung von Energiewerten anstelle von Leistungswerten. Die der Quelle entnommene Energie pro Schaltvorgang ist natürlich unabhängig von der Frequenz. Die Schaltungssimulation erlaubt es zunächst nur, auf direktem Weg die Leistungen $P(x)$ zu bestimmen. Entsprechend ergeben sich die Energiewerte für die Einzelnen Eingangsübergänge aus: $W_0(x) = \frac{1}{f_0}P_0(x)$ und $W_\mathrm{L} = \frac{1}{f_0}P_\mathrm{L}$. Zur Ermittlung der gesamten Verlustleistung einer Schaltung müssen bei diesem Ansatz nur die einzelnen Energiewerte aufsummiert und mit der Taktfrequenz multipliziert werden.

## 6.3 Modellbeschreibung in VHDL

Nach der Bestimmung der Modellparameter durch Schaltungssimulationen der Basiszellen müssen die gewonnenen Informationen in geeigneter Weise in eine Modellbeschreibung für die Basiszellen eingebettet werden. Für diese Beschreibung eignet sich die Hardwarebeschreibungssprache VHDL in besonderer Weise. VHDL ist ein allgemeiner, international akzeptierter Standard zur Dokumentation, Simulation und zum Datenaustausch beim Entwurf digitaler Schaltungen. Die mit VHDL erstellten Modelle sind somit portierbar und ohne jede weitere Aufbereitung von jedem Entwickler anwendbar. Darüberhinaus erlaubt es die Struktur der Sprache VHDL, neben der funktionalen Beschreibung auch zusätzliche physikalische Eigenschaften einzelner Zellen, Module oder ganzer Schaltungen zu berücksichtigen. Eigenschaften dieser Art können beispielsweise die Verzögerungszeiten und Verlustleistungen sein. Die Aufgabe bei der Modellbeschreibung in VHDL besteht darin, die funktionale Beschreibung der Basiszellen zu erstellen und darüberhinaus die Parameter, die nur durch Betrachtungen auf unterster Schaltkreisebene zugänglich sind, mit in das abstrakte Modell auf höherer Entwurfsebene einzubeziehen. Zusätzlich muss sichergestellt sein, dass während der abschließenden VHDL-Simulation Informationen zwischen den einzelnen Zellen einer Schaltung ausgetauscht werden können. Jede Zelle muss der vorhergehenden Zelle den Wert ihrer Eingangskapazität übergeben, womit dann in der Vorgängerzelle der Wert der Ausgangslast bestimmt wird. Auf diese Weise ist es möglich, die Gültigkeit der Modelle für die Basiszellen allgemein zu halten, sodass auch unterschiedlichste Belastungssituationen, die sich aus dem Einbau der Zellen in eine Schaltung ergeben, von der Modellierung abgedeckt werden können.

VHDL bietet die Möglichkeit, zwischen verschiedenen Beschreibungsformen zu wählen:

- einer Verhaltensbeschreibung,

- einer datenflussorientierten Beschreibung,
- einer Strukturbeschreibung,
- einer Kombination der oben genannten Beschreibungsformen.

Bei der Verhaltensbeschreibung wird das Verhalten der Schaltung durch einen Satz von sequentiellen, also nacheinander auszuführenden, Anweisungen beschrieben. Es wird also keinerlei Struktur vorgeschrieben, sondern lediglich die Funktionalität. Wegen der sequentiellen Abarbeitung der Anweisungen eignet sich diese Beschreibungsform nicht für die Modellierung von Basiszellen, in denen die Signale aufgrund unterschiedlicher Laufzeiten gegeneinander "konkurrieren" können und kein sequentieller Ablauf von Ereignissen vorliegt. Datenabhängige Laufzeiten sind jedoch eine wesentliche Eigenschaft der zu beschreibenden Basiszellen. Die datenflussorientierte Beschreibung hingegen besteht aus konkurrierenden Anweisungen, die nach einer definierbaren Verzögerungszeit ausgeführt werden, ohne Rücksicht auf eine bestimmte Reihenfolge. Diese Art der Beschreibung eignet sich ideal für die gewünschte Modellierung der Zellen, da hier für bestimmte Ereignisse (z. B. Eingangsübergänge) individuelle Verzögerungszeiten angegeben werden können und zudem die Möglichkeit besteht, Abfragen einzuführen, d. h. Fallunterscheidungen zwischen verschiedenen Ereignissen zu machen. Bei einer Strukturbeschreibung wird eine Schaltung als ein Satz von miteinander verbundenen Komponenten modelliert. Diese Art der Modellierung entspricht einer Netzlistenbeschreibung, bei der die Verbindungen zwischen den Ein- und Ausgängen der Komponenten festgelegt wird. Die Strukturbeschreibung eignet sich besonders für Schaltungen, die eine charakteristische Struktur aufweisen, wie beispielsweise reguläre Datenpfade. Da Basiszellen keine weiteren Komponenten enthalten, sondern die kleinste Einheit auf der betrachteten Abstraktionsebene bilden, ist die Strukturbeschreibung für die Modellierung nicht geeignet. Für die Beschreibung der aus den Basiszellen aufgebauten Gesamtschaltungen ist diese Form der Beschreibung jedoch von Vorteil, da der Austausch von Informationen zwischen den Zellen hiermit sehr gut beschrieben werden kann.

Für die Modellierung der Basiszellen wird eine datenflussorientierte VHDL-Beschreibung gewählt. Die folgenden Modellparameter, die durch Schaltungssimulation ermittelt wurden, sollen dabei in die Funktionsbeschreibung einfließen:

- Grundverzögerungszeit ohne Last für alle möglichen Eingangsübergänge,
- Verlustleistung ohne Last für alle möglichen Eingangsübergänge,
- Zeitinkrement zur Gesamtverzögerungszeit bei Belastung mit einer Normlast,
- Leistungsinkrement zur gesamten Verlustleistung bei Belastung mit einer Normlast,
- minimale Zeitdauer, die ein Eingangsimpuls haben muss, um am Ausgang eine Wirkung hervorzurufen,
- durch Laufzeitunterschiede innerhalb der Basiszelle hervorgerufene Glitches.

Die möglichen Fälle von Eingangsübergängen werden durch Abfragen voneinander unterschieden. Dies ist in der VHDL-Beschreibung, wie auch bei Programmiersprachen wie C oder Pascal, durch eine If-Then-Else-Struktur möglich. Da Eingangsübergänge betrachtet werden sollen, ist es nötig, ausser dem aktuellen Eingangsdatenvektor $q_i$ auch den vorhergehenden Eingangsdatenvektor $q_{i-1}$ zu kennen, wobei $i$ den aktuellen Taktzeitpunkt beschreibt. Der Eingangübergang $x_i$ bezeichne das Ereignis $q_{i-1} \longrightarrow q_i$. Die Menge aller Übergänge, die am Eingang einer Basiszelle möglich sind, sei X, mit $x_i \in$ X, $i = 1, \ldots, 2^{2n} - 2^n$. Im VHDL-Modell der Basiszellen werden die Verzögerungszeiten und Verlustleistungen für alle möglichen Eingangskombinationen $m_j$ abgelegt. Die Menge der in der Modellbeschreibung abgelegten Eingangskombinationen sei M, wobei $m_j \in$ M und $j = 1, \ldots, 2^{2n} - 2^n$. Im Fall

der Übereinstimmung zwischen dem aktuellen Eingangsübergang $x_i$ und einer in der Modellbeschreibung abgelegten Eingangskombination $m_j$, werden die entsprechenden Anweisungen für diesen Fall ausgeführt. Dabei wird die für diesen Fall charakteristische Verzögerungszeit berücksichtigt und der entsprechende Wert der Verlustleistung zur Gesamtverlustleistung addiert. Schließlich wird $q_{i-1} = q_i$ gesetzt und somit der aktuelle Eingangsdatenvektor als Vorgänger für den folgenden Eingangsdatenvektor gespeichert. Die Zuweisung des neuen Ausgangswertes erfolgt durch eine Transport-Anweisung. Der Wert der Verzögerungszeit, der sich für den aktuellen Eingangsübergang ergibt, wird erst beim eigentlichen Aufruf der Transport-Anweisung aus den charakteristischen Werten $\tau_0(x)$ und $\tau_\mathrm{L}$ für den aktuellen Belastungsfall berechnet.

Jede VHDL-Beschreibung einer Zelle besteht im wesentlichen aus zwei Blöcken, *Entity* und *Architecture* . Die Entity beschreibt die äusseren Anschlüsse und ist gewissermaßen eine Black-Box-Beschreibung der Zelle. Der Architecture-Block beschreibt die eigentliche Funktion. Im Folgenden ist der Architecture-Block des VHDL-Modells einer Basiszelle angegeben. Der Name der Basiszelle ist dabei "$\mathrm{Z}_\mu$".

**ARCHITECTURE** $\mathrm{Z}_\mu$_Arc OF $\mathrm{Z}_\mu$ **IS**
**BEGIN**
  **PROCESS**
  **BEGIN**
    **WAIT ON** $q_i$;
    **WAIT ON** $q_i$ FOR $t_{\min}$;
    $x_i := q_{i-1} \longrightarrow q_i$;
    **IF** $x_i = m_j$ **THEN**
        $y <=$ **TRANSPORT** $B$ **AFTER** $(t(m_j) = t_0(m_j) + K \cdot t_\mathrm{L})$;
        $P_{\mu,j} := P_0(m_j) + K \cdot P_\mathrm{L}$;
    **ELSIF** $x_i = m_{j+1}$ **THEN**
            ...
    **END IF;**
    $P_\mu := P_\mu + P_{\mu,j}$;
    $q_{i-1} := q_i$;
  **END PROCESS;**
**END** $\mathrm{Z}_\mu$_Arc;

Die Anweisung "**WAIT ON** $q_i$" bedeutet das Warten auf das nächste Ereignis am Eingang. Durch die darauffolgende Anweisung "**WAIT ON** $q_i$ FOR $t_{\min}$" wird erreicht, dass für die Zeitdauer von $t_{\min}$ auf das übernächste Ereignis am Eingang gewartet wird. Erfolgt dieses neue Ereignis vor Ablauf der Zeit $t_{\min}$, so wird das vorhergehende Ereignis verworfen und das neue Ereignis als aktuell übernommen. Durch diesen Programmkonstrukt ist es möglich, kürzere Glitches als $t_{\min}$ zu eliminieren. Mit der Transport-Anweisung wird dem Ausgang $y$ nach Ablauf der Zeit $t(m_j) = t_0(m_j) + K \cdot t_\mathrm{L}$ der Wert $B$ zugewiesen, $B \in \{0, 1\}$. Durch spezielle Ein- und Ausgangsvariablen muss sichergestellt werden, dass ein Informationsaustausch zwischen den Zellen innerhalb einer Schaltung möglich ist. Dazu werden in der Entity entsprechende Ein- und Ausgänge definiert, die keine Signale darstellen, sondern reellwertige Variablen, entsprechend den Ein- und Ausgangsvariablen in Funktionen und Prozeduren anderer Programmiersprachen, wie z. B. C. Für das hier vorgestellte Modell wird eine Ausgangsvariable $c_\mathrm{E}$ definiert, die die Größe der Eingangskapazität in Form des Faktors

beinhaltet, der multipliziert mit der Kapazität der Normlast den eigentlichen Kapazitätswert ergibt. Der Wert von $c_{\mathrm{E}}$ ist eine charakteristische Größe der Basiszelle und konstant. Eine vektorwertige Eingangsvariable **k** dient zum Einlesen der Ausgangslast. Jede Komponente dieses Vektors beinhaltet den Wert der Eingangskapazität einer nachfolgenden Zelle oder null. Die Dimension des Vektors **k** muss ausreichend groß gewählt werden, entsprechend der maximalen Anzahl der am Ausgang angeschlossenen Zellen. Die gesamte Ausgangsbelastung der Basiszelle wird innerhalb des Architecture-Blocks der Modellbeschreibung berechnet. Dies geschieht durch Addition der Komponenten des Vektors **k**. Somit ist $K = \sum_{\nu=1}^{N} k_\nu$, wobei $N$ die Dimension des Vektors **k** bezeichnet. Eine weitere Ausgangsvariable $P_\mu$ ist nötig, um den Wert der Verlustleistung in die Ergebnisdatei "DAT"ausgeben zu können. In Bild 6.9 ist der Informationsaustausch zwischen den Zellen mit Hilfe der zusätzlichen Ein- und Ausgangsvariablen dargestellt.

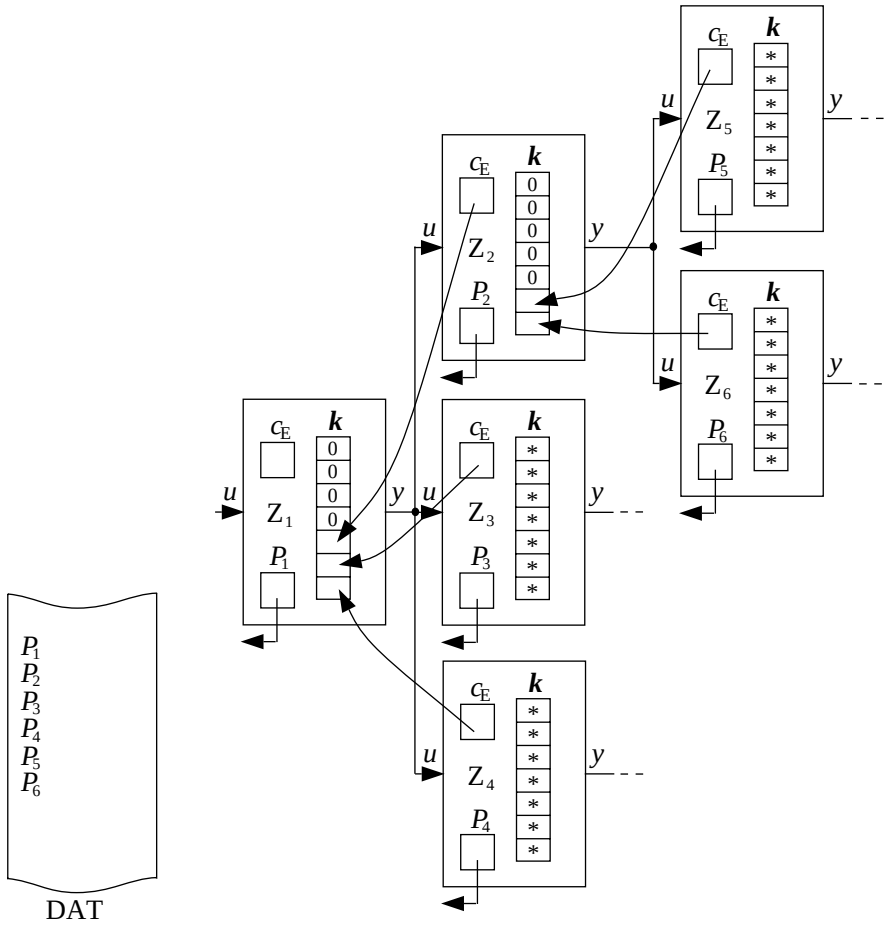



Bild 6.9. Schaltung aus mehreren Basiszellen. Neben den Leitungsverbindungen für die Signale besteht ein Informationsaustausch über die vektorwertige Eingangsvariable **k** und die Ausgangsvariable $c_{\mathrm{E}}$. Die Verlustleistung jeder Zelle wird in die Ergebnisdatei "DAT" ausgegeben.

Mit der Erstellung der VHDL-Modelle aller Basiszellen ist die Charakterisierungsphase abgeschlossen. Diese Phase muss für jede Basiszelle nur einmal durchlaufen werden. Die daraus gewonnenen VHDL-Modelle sind dann universell in verschiedenen Schaltungen einsetzbar. Der Zeitaufwand für die Charakterisierung einer Basiszelle setzt sich aus der Zeit für die Schaltungssimulation zur Ermittlung der Parameter und der VHDL-Programmierung

zusammen. Die Simulationsdauer zur vollständigen Bestimmung der Parameter eines Volladdierers mit SPICE beträgt auf einer Workstation ca. 8 Minuten, die VHDL-Beschreibung umfasst 310 Zeilen Code, wobei 56 If-Then-Else-Anweisungen mit jeweils 4 Zeilen Code enthalten sind, die sich nur in den Werten der Parameter für die Verlustleistung und die Verzögerungszeit unterscheiden und daher durch einfaches Kopieren und Verändern der Parameterwerte gewonnen werden können. Das Programmieren des VHDL-Modells eines Volladdierers lässt sich daher in weniger als einer Stunde bewerkstelligen. Diese Zeit ließe sich durch Verwendung eines Code-Generators erheblich verkürzen. Die Programmierung eines Code-Generators, der aus den Ergebnissen der SPICE-Simulation den VHDL-Code für die Basiszellen automatisch generiert, ist Gegenstand zukünftiger Arbeiten.

Es bleibt anzumerken, dass eine Schaltung unter Verwendung der hier eingeführten Zellenmodelle im Allgemeinen nicht synthetisierbar ist, d. h. die derzeitigen Entwicklungswerkzeuge sind nicht in der Lage, anhand dieser Beschreibungen eine Schaltung zu erzeugen. Diese Form der Zellenbeschreibung dient lediglich für die Simulation von Schaltungen auf höherer Abstraktionsebene zur schnellen Ermittlung der Verlustleistung. Um eine syntetisierbare VHDL-Beschreibung für die Gesamtschaltung zu erhalten, müssen jedoch nur andere, synthetisierbare Architecture-Blöcke für die Basiszellen verwendet werden. Die Entities als Klemmenbeschreibungen bleiben unverändert. In VHDL ist es möglich, für eine Entity mehrere Architecture-Blöcke zu definieren. In einem Konfigurationsteil der Gesamtschaltung muss dann lediglich angegeben werden, welche Architecture-Blöcke für die jeweiligen Entities verwendet werden sollen. Je nachdem, ob die Schaltung synthetisiert oder die Verlustleistung bestimmt werden soll, müssen deshalb nur kleine Änderungen in einem Konfigurationsteil der Schaltungsbeschreibung vorgenommen werden.

## 6.4 Bestimmung der Verlustleistung auf höherer Abstraktionsebene

Die in der Charakterisierungsphase erstellten Modelle für die Basiszellen werden im Folgenden dazu verwendet, die Verlustleistung komplexerer Schaltungen zu ermitteln. Voraussetzung dafür ist natürlich, dass die Schaltungen aus den vorher charakterisierten Basiszellen aufgebaut sind. Als Beschreibungsform eignet sich eine strukturelle Logikbeschreibung. In dieser Beschreibung tauchen die Basiszellen als Komponenten auf. Die VHDL-Beschreibungen der Basiszellen mit der in Abschnitt 6.3 vorgestellten Modellstruktur finden sich in einer eigenen VHDL-Zellbibliothek. Ein konventioneller VHDL-Logiksimulator dient nun in der zweiten Phase der Verlustleistungsbestimmung dazu, die Funktion der Schaltung unter Verwendung von Testdatenvektoren zu überprüfen. Dabei werden für die Komponenten der Schaltung die VHDL-Beschreibungen der entsprechenden Basiszellen aus der VHDL-Zellbibliothek verwendet. Gleichzeitig werden bei dieser ereignisgesteuerten Simulation die Leistungsaufnahmen für die einzelnen Ereignisse in allen Basiszellen in einer Ergebnisdatei abgelegt. Dies geschieht lediglich durch die spezielle Beschaffenheit der VHDL-Beschreibungen der Basiszellen und ist keine Option des Simulators. Ebenso ist die Genauigkeit des Ergebnisses für die Gesamtverlustleistung der simulierten Schaltung nur aufgrund der detaillierten Modelle gegeben. Der VHDL-Logiksimulator hat hierbei lediglich die Aufgabe, die Daten zu den entsprechenden Zeitpunkten an den einzelnen Zellen der Schaltung anzulegen. Der Vorteil dieser Methode ist, dass kein neues Programmwerkzeug für die Verlustleistungsbestimmung benötigt wird. Durch den zweiphasigen Aufbau ist

die aufwendigere, jedoch für jede Basiszelle nur einmal durchzuführende Modellierung von der simulationsbasierten Ermittlung der Verlustleistung größerer Schaltungen getrennt. Die grafische Darstellung in Bild 6.10 zeigt den prinzipiellen Aufbau der beiden Phasen der Methode, die im Folgenden als *DCM*-Methode (*D*evice-Level-Based-*C*ell *M*odelling-Methode) [64, 65] bezeichnet wird. Nach der Simulation steht für jede Zelle der simulierten Schaltung

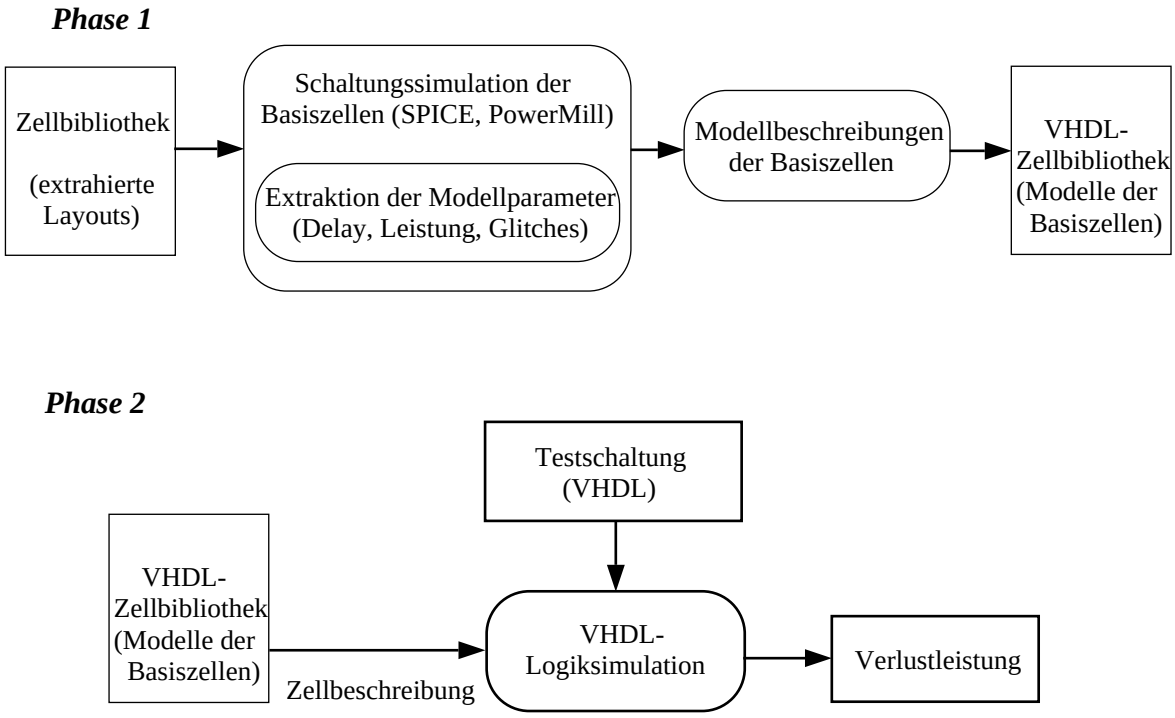



Bild 6.10. Zweiphasiger Aufbau der DCM-Methode zur Bestimmung der Verlustleistung.

genau ein Eintrag in der Ergebnisdatei. Jeder Eintrag ist die Summe der Verlustleistungen aller Schaltvorgänge am Eingang einer Zelle. Die Summe über alle Einträge in der Datei dividiert durch die Anzahl der Taktperioden während der Simulation ergibt die gesamte Verlustleistung der Schaltung.

$$P = \frac{1}{N}\frac{f}{f_0}\sum_{\mu=1}^{M}\sum_{j=1}^{S_\mu} P_{\mu,j} = \frac{1}{N}\frac{f}{f_0}\sum_{\mu=1}^{M} P_\mu \tag{6.10}$$

In dieser Formel bezeichnet $N$ die Anzahl der Taktperioden, $M$ die Anzahl der Zellen in der Schaltung, $S_\mu$ die Anzahl der Schaltaktionen am Eingang der Zelle $\mu$ und $P_{\mu,j}$ die Verlustleistung der Zelle $\mu$, die durch die Eingangskombination $m_j$ hervorgerufen wird. Die Frequenz, mit der die Schaltung betrieben wird ist $f$, $f_0$ bezeichnet die Frequenz, für die die Werte $P_{\mu,j}$ durch Schaltungssimulation der Basiszellen ermittelt wurden. Die innere der beiden Summen, $\sum_{j=1}^{S_\mu} P_{\mu,j}$, wird bereits innerhalb der Basiszellen bestimmt.

Die Überprüfung der hier vorgestellten DCM-Methode hinsichtlich der Genauigkeit bei der Ermittlung der Verlustleistung soll anhand einiger Beispielschaltungen durchgeführt werden. Bei den im Folgenden betrachteten Beispielen handelt es sich um Multiplizierer- und Addiererschaltungen, also typischen Datenpfadkomponenten, wie sie in der digitalen Signalverarbeitung Anwendung finden. Die hier verwendeten Schaltungen sind $5 \times 5$-, $8 \times 8$- und

16 × 16-Bit Array-Multiplizierer, 8 × 8-Bit Wallace-Tree-Multiplizierer sowie 16- und 32-Bit Carry-Lookahead-Addierer. Jede Multipliziererarchitektur wurde dabei mit den folgenden Volladdierern realisiert: einem Standard-CMOS-Volladdierer (vgl. Bild 6.5), einem auf Transmission-Gates basierenden Volladdierer [23] und einem flächeneffizienten, aus speziellen XOR- und XNOR-Gattern aufgebauten Volladdierer, der lediglich aus 14 Transistoren besteht [75, 86]. Tabelle 6.3 zeigt die mit SPICE ermittelten Verzögerungszeiten und Verlustleistungen der drei verschiedenen Addierertypen. Die Addierer wurden dazu als Layouts in $0.25\mu m$-Technologie realisiert. Für alle drei Addierer wurde die gleiche Ausgangslast angenommen. Der Standard-CMOS-Addierer ist im Folgenden kurz als "CMOS", der Transmission-Gate-Addierer als "TG" und der flächeneffiziente, aus 14 Transistoren aufgebaute Addierer als "T14" bezeichnet.

| Addierer | Leistung in $(\mu W)$ | Verzögerung in $(ns)$ |
|---|---|---|
| CMOS | 3.39 | 0.89 |
| TG | 5.23 | 0.84 |
| T14 | 4.07 | 0.78 |

Tabelle 6.3. Verlustleistungen und Verzögerungszeiten von drei verschiedenen Volladdierern.

Im Fall der 5 × 5-Bit Multiplizierer erfolgte die Realisierung in Form von Layouts in $0.25\mu m$-Technologie. Alle anderen Schaltungen wurden nach VHDL-Beschreibungen synthetisiert und auf eine $0.7\mu m$-Technologie abgebildet. Für den Vergleich zwischen der Schaltungssimulation mittels SPICE und den Ergebnissen aus der DCM-Methode wurden nur die 5 × 5-Bit Multiplizierer als Beispielschaltungen herangezogen. Die Simulation der größeren Schaltungen ist mit SPICE nicht mehr in akzeptabler Zeit durchführbar. PowerMill erlaubt eine bis zu eintausendfach kürzere Simulationszeit als SPICE. Für die größeren Schaltungen wurde deshalb der Vergleich mit PowerMill durchgeführt. In diesen Fällen wurde auch die Charakterisierung der Basiszellen mit PowerMill durchgeführt, um die Genauigkeit der DCM-Methode im Vergleich mit einer Schaltungssimulation zu zeigen. Die Verwendung eines genaueren Simulators wie SPICE zur Charakterisierung würde das Ergebnis dieses Vergleichs zugunsten der DCM-Methode verzerren.

Für die SPICE-Simulationen der 5 × 5-Bit Multiplizierer wurden aus den handentworfenen Layouts der Schaltungen Netzlisten in einem für SPICE verständlichen Format extrahiert. Die Extraktion der Netzlisten erfolgte unter Berücksichtigung aller parasitären Elemente, einschließlich der Drain-/Source-, Gate- und Leitungskapazitäten. Anschließend wurden die Schaltungen unter Verwendung von 1000 zufälligen Eingangsvektoren simuliert und die Verlustleistung bestimmt. Die Verlustleistungsbestimmung mit der DCM-Methode erfolgte anhand der VHDL-Beschreibungen der Schaltungen. Die Modellparameter der Basiszellen wurden dabei aus den entsprechenden Layouts der Zellen extrahiert, die auch in den Gesamtlayouts der Multiplizierer Anwendung fanden. In Tabelle 6.4 sind die Ergebnisse für die 5 × 5-Bit Multiplizierer aufgelistet. Dabei sind sowohl die ermittelten Verlustleistungen als auch die Simulationszeiten für SPICE und die DCM-Methode aufgeführt. Die Abweichungen der Werte nach der DCM-Methode von den SPICE-Ergebnissen sind in der Spalte $\Delta$ in Prozent angegeben. Die maximale Abweichung bei den ermittelten Leistungswerten beträgt 4%. Alle Simulationen wurden auf einer SUN Ultra Sparc 10 Workstation durchgeführt. Die Ergebnisse aus der Tabelle zeigen, dass die DCM-Methode für die untersuchten Schaltungen eine bis zu vier Größenordnungen schnellere Ermittlung der Verlustleistung er-

| Schaltung | Leistung ($mW$) | | $\Delta$ | Simulationsdauer ($s$) | |
|---|---|---|---|---|---|
| | SPICE | DCM | | SPICE | DCM |
| $5 \times 5$ (CMOS) | 0.081 | 0.081 | 0.0% | 38398 | 5 |
| $5 \times 5$ (TG) | 0.144 | 0.150 | 4.0% | 25354 | 5 |
| $5 \times 5$ (T14) | 0.141 | 0.145 | 2.7% | 16855 | 5 |

Tabelle 6.4. Leistungsverbrauch von $5 \times 5$-Bit Array-Multiplizierern gleicher Struktur aber mit unterschiedlichen Volladdierern. Vergleich zwischen SPICE und der DCM-Methode.

laubt als SPICE. Da die Rechenzeit von SPICE exponentiell mit der Anzahl der Knoten ansteigt, wirkt sich der Rechenzeitunterschied bei größeren Schaltungen noch gravierender aus. Die Simulation von $16 \times 16$-Bit Multiplizierern mit 1000 Eingangsvektoren ist mit SPICE nicht mehr praktikabel. Tabelle 6.4 zeigt die Ergebnisse für die Simulationen der verschiedenen $16 \times 16$-Bit Multiplizierer und der Carry-Lookahead-Addierer mit PowerMill. Auch hier sind zum Vergleich die Ergebnisse nach der DCM-Methode aufgeführt. Ein Vergleich der Rechenzeiten zeigt, dass die DCM-Methode eine ein bis zwei Größenordnungen schnellere Ermittlung der Verlustleistung erlaubt. Die maximale Abweichung bei den Werten für die Verlustleistung beträgt ca. 8%. Beim Vergleich der Rechenzeiten fällt auf, dass PowerMill für Schaltungen mit gleicher Architektur bei Verwendung des T14-Addierers erheblich mehr Zeit benötigt als bei den Realisierungen mit anderen Addierertypen. Dies liegt möglicherweise daran, dass in diesem Addierertyp Knoten existieren, die nicht voll auf $U_{\mathrm{B}}$ aufgeladen, bzw. nicht vollständig entladen werden, woraus sich für PowerMill Konvergenzprobleme ergeben können. Allgemein ist die Rechenzeit bei der DCM-Methode weit weniger von der Art der verwendeten Grundzellen abhängig als bei PowerMill.

| Schaltung | Leistung ($mW$) | | $\Delta$ | Simulationsdauer ($s$) | |
|---|---|---|---|---|---|
| | PowerMill | DCM | | PowerMill | DCM |
| $8 \times 8$-Mult. (CMOS) | 1.13 | 1.16 | 2.1 | 214 | 14 |
| $8 \times 8$-Mult. (TG) | 1.80 | 1.85 | 2.9 | 323 | 15 |
| $8 \times 8$-Mult. (T14) | 1.55 | 1.64 | 5.7 | 1643 | 14 |
| $16 \times 16$-Mult. (CMOS) | 7.56 | 7.76 | 2.6 | 1537 | 94 |
| $16 \times 16$-Mult. (TG) | 12.01 | 12.66 | 5.4 | 3002 | 109 |
| $16 \times 16$-Mult. (T14) | 8.28 | 8.94 | 7.9 | 42616 | 2105 |
| Wallace-Mult. (CMOS) | 0.91 | 0.92 | 1.5 | 167 | 11 |
| Wallace-Mult. (TG) | 1.43 | 1.49 | 4.8 | 218 | 11 |
| Wallace-Mult. (T14) | 1.65 | 1.73 | 4.8 | 1592 | 12 |
| CLA-Addierer 16-Bit | 1.25 | 1.29 | 3.2 | 157 | 18 |
| CLA-Addierer 32-Bit | 4.95 | 5.18 | 4.6 | 278 | 64 |

Tabelle 6.5. Leistungsverbrauch verschiedener Schaltungen. Vergleich zwischen PowerMill und der DCM-Methode.

Eine interessante Frage ist, inwieweit die Rechenzeit für die Simulation mit dem VHDL-Logiksimulator durch die aufwendige Art der Modellierung der Basiszellen bei der DCM-Methode beeinflusst wird. Relevant ist in diesem Zusammenhang der Unterschied in der Simulationszeit bei genauer Modellierung und bei reiner Verhaltensbeschreibung der Basiszellen. Es ist offensichtlich, dass die Genauigkeit bei der Modellierung mit einem größeren

Rechenaufwand erkauft werden muss. Dies erklärt sich bereits aus der Tatsache, dass beispielsweise die Verhaltensbeschreibung eines Volladdierers gerade zwei Anweisungen enthält (je eine für Summen- und Carry-Pfad), wohingegen bei der Modellierung nach Abschnitt 6.3 im ungünstigsten Fall zunächst 56 Abfragen durchgeführt, und dann die entsprechenden Anweisungen für Carry- und Summenpfad ausgeführt werden müssen. Hinzu kommt noch der zusätzliche Aufwand für die Berechnung der aktuellen Leistungen und Verzögerungszeiten entsprechend der Ausgangslast, sowie für das Aufsummieren der einzelnen Verlustleistungen. In Tabelle 6.6 sind die Simulationszeiten bei reiner Verhaltensbeschreibung und bei genauer Modellierung der Basiszellen von Array-Multiplizierern unterschiedlicher Größe aufgelistet. Durch die genauere Modellierung verlangsamt sich die VHDL-Simulation in den hier unter-

| Schaltung | DCM-Modellierung der Basiszellen | Verhaltensbeschreibung der Basiszellen |
|---|---|---|
| $8 \times 8$-Mult. (CMOS) | $14s$ | $5s$ |
| $16 \times 16$-Mult. (CMOS) | $94s$ | $38s$ |
| $32 \times 32$-Mult. (CMOS) | $675s$ | $332s$ |

Tabelle 6.6. Vergleich der Rechenzeiten für die VHDL-Simulationen von Array-Multiplizierern bei Modellierung der Basiszellen nach der DCM-Methode und bei reiner Verhaltensbeschreibung der Basiszellen.

suchten Fällen etwa um einen Faktor 2 bis 3. Dieser Faktor ist jedoch auch von der Komplexität der verwendeten Basiszellen abhängig. Komplexere Zellen erfordern eine aufwendigere Modellierung. Die Verhaltensbeschreibung hingegen kann zumeist auch für komplexere Zellen mit wenigen Anweisungen erstellt werden. Dementsprechend kann sich ein deutlicherer Unterschied in den Simulationszeiten ergeben.

Zusammenfassend lässt sich feststellen, dass sich das hier vorgestellte, zweiphasige Verfahren zur Ermittlung der Verlustleistung durch eine für diese Abstraktionsebene (Logikebene oder Register-Transferebene) hohe Genauigkeit auszeichnet. Die Charakterisierung der Basiszellen erfolgt auf Schaltkreisebene und muss nur einmal pro Zelle ausgeführt werden. In der eigentlichen Ausführungsphase wird die Verlustleistung einer Schaltung mittels eines konventionellen VHDL-Simulators ermittelt. Es ist keinerlei neue Software nötig. Das Verfahren erlaubt eine zwei bis vier Größenordnungen schnellere Bestimmung der Verlustleistung als PowerMill respektive SPICE. Dadurch ist ein schneller Vergleich verschiedener Schaltungsrealisierungen hinsichtlich der Verlustleitung möglich.

# 7. Effiziente Implementierung einer Signaltransformation

In diesem Kapitel sollen am Beispiel einer diskreten 8-Punkte Kosinustransformation (DCT) [30, 83] zwei verschiedene Varianten von Schaltungsarchitekturen bezüglich Realisierungaufwand und Leistungsverbrauch verglichen werden. Die DCT findet breite Anwendung auf verschiedensten Gebieten der Bild- und Signalverarbeitung [34]. Bei Bildverarbeitungssystemen wird dabei das Bild zumeist in Blöcke von $8 \times 8$ oder $16 \times 16$ Pixel aufgeteilt, die dann einzeln transformiert werden. Die transformierten Blöcke werden zunächst quantisiert und schließlich kodiert versendet. Am Empfänger wird eine inverse Kosinustransformation (IDCT) durchgeführt, um wieder das ursprüngliche Bild zu erhalten. Durch die Anwendung der DCT zur Bildcodierung lassen sich hohe Kompressionsraten erreichen.

Von besonderer Wichtigkeit im Hinblick auf portable Systeme ist die Realisierung der DCT durch eine Schaltung mit geringer Verlustleistung. Für die Realisierung werden in diesem Kapitel zwei Architekturen näher betrachtet, die auf unterschiedlichen Quantisierungsverfahren basieren. Als Richtlinie für die numerischen Eigenschaften der verschiedenen Implementierungen dient dabei der IEEE-Standard für die diskrete $8 \times 8$ Kosinustransformation [1]. Bei einer der beiden betrachteten Architekturen erfolgt eine Quantisierung und damit Approximation der Winkel, um die die Datenvektoren im Zustandsraum durch die Transformation gedreht werden. Um diese Rotationswinkel zu approximieren, werden einfach zu implementierende Teilrotationen verwendet, mit denen sich jeder Winkel bis zu einer beliebigen Genauigkeit nachbilden lässt. Die Methode der Winkelapproximation wurde in anderen Arbeiten bereits für diskrete Single- und Multiwavelettransformationen angewendet. In [54, 55, 80, 81] wurde gezeigt, dass mit diesem Verfahren sehr effiziente VLSI-Implementierungen von Wavelettransformationen möglich sind. Allerdings erfolgte in diesen Arbeiten keine Orientierung an einen gegebenen Standard. Interessant ist die Frage, inwieweit sich die Winkelquantisierung für standardisierte Signaltransformationen wie der DCT eignet. In [60] wurde mit der Methode der Winkelquantisierung eine verlustleistungsarme und flächensparende Lösung für die Implementierung der DCT vorgestellt. Zum Vergleich diente in dieser Arbeit eine Architektur, bei der nicht die Rotationswinkel, sondern, wie bei Standard-Realisierungen von digitalen Filtern üblich, die Multipliziererkoeffizienten quantisiert wurden. In diesem Kapitel wird eine gegenüber [60] günstigere Koeffizientenquantisierung vorgestellt und mit der Winkelquantisierung verglichen.

Das Kapitel gliedert sich in vier Abschnitte. Zunächst wird der zugrundegelegte DCT-Algorithmus vorgestellt. Der Signalflussgraph der Schaltung lässt sich dabei direkt aus einer Zerlegung der Transformationsmatrix in $2 \times 2$ Blöcke ableiten. Anschließend wird näher

auf die Quantisierung der Rotationswinkel eingegangen. Ein besonderes Augenmerk wird dabei auf die dazu verwendeten Klassen von Teilrotationen gerichtet. In einem weiteren Abschnitt folgt die Betrachtung der Koeffizientenquantisierung und der dabei möglichen Optimierungen. Die abschließende Realisierung beider Architekturen als Layouts ermöglicht einen quantitativen Vergleich des Flächenaufwandes. Mittels Simulation und der in Kapitel 6 vorgestellten DCM-Methode werden schließlich beide Realisierungen hinsichtlich der Verlustleistung untersucht.

## 7.1 Der Transformationsalgorithmus

Ein strukturierter, schneller Algorithmus zur Berechnung der DCT wurde von Chen et al. in [17] vorgestellt. Durch die spezielle Faktorisierung der Transformationsmatrix $\mathbf{C}_N$ müssen hierbei nur reelle Operationen ausgeführt werden. Die allgemeine Formulierung der DCT lautet

$$\mathbf{X} = \sqrt{\frac{2}{N}}\mathbf{C}_N\mathbf{x} \quad , \tag{7.1}$$

mit dem $N$-dimensionalen Datenvektor $\mathbf{x}$. Im Fall der hier speziell betrachteten 8-Punkte DCT wird $N = 8$ gewählt. Für die inverse Transformationsmatrix $\mathbf{C}_N^{-1}$ gilt aufgrund der Orthogonalität:

$$\mathbf{C}_N^{-1} = \mathbf{C}_N^{\mathrm{T}} \quad .$$

Die Rücktransformation ergibt sich entsprechend zu

$$\mathbf{x} = \sqrt{\frac{2}{N}}\mathbf{C}_N^{\mathrm{T}}\mathbf{X} \quad . \tag{7.2}$$

Die $N \times N$-dimensionale Transformationsmatrix lautet allgemein

$$\mathbf{C}_N = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \cdots & \frac{1}{\sqrt{2}} \\ \cos\left(\frac{\pi}{2N}\right) & \cos\left(\frac{3\pi}{2N}\right) & \cdots & \cos\left(\frac{(2N-1)\pi}{2N}\right) \\ \cos\left(\frac{2\pi}{2N}\right) & \cos\left(\frac{6\pi}{2N}\right) & \cdots & \cos\left(\frac{(2N-1)2\pi}{2N}\right) \\ \vdots & \vdots & \ddots & \vdots \\ \cos\left(\frac{(N-1)\pi}{2N}\right) & \cos\left(\frac{3(N-1)\pi}{2N}\right) & \cdots & \cos\left(\frac{(2N-1)(N-1)\pi}{2N}\right) \end{pmatrix} \quad . \tag{7.3}$$

Die Transformationsmatrix kann in rekursiver Form dargestellt werden. Die $N$-dimensionale Matrix berechnet sich hierbei aus $\frac{N}{2}$-dimensionalen Teilmatrizen gemäß

$$\mathbf{C}_N = \mathbf{P}_N \begin{pmatrix} \mathbf{C}_{\frac{N}{2}} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{\frac{N}{2}} \end{pmatrix} \mathbf{B}_N \quad . \tag{7.4}$$

Die Matrix $\mathbf{B_N}$ ist dabei wie folgt definiert:

$$\mathbf{B}_N = \begin{pmatrix} \mathbf{I}_{\frac{N}{2}} & \mathbf{J}_{\frac{N}{2}} \\ \mathbf{J}_{\frac{N}{2}} & -\mathbf{I}_{\frac{N}{2}} \end{pmatrix} \mathbf{B}_N \quad . \tag{7.5}$$

Dabei ist $\mathbf{I}_{\frac{N}{2}}$ eine Einheitsmatrix und $\mathbf{J}_{\frac{N}{2}}$ eine Koidentitätsmatrix, also eine Matrix mit ausschließlich Einsen auf der Antidiagonalen, alle anderen Einträge sind null. Die Permutationsmatrix $\mathbf{P}_N$ dient dazu, die Elemente eines Vektors so zu sortieren, dass sie wieder in

aufsteigender Reihenfolge angeordnet sind.

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ x_N \end{pmatrix} = \mathbf{P_N} \begin{pmatrix} x_1 \\ x_N \\ x_2 \\ x_{N-2} \\ \vdots \\ x_{N-\frac{N}{2}+1} \end{pmatrix} \tag{7.6}$$

Die Matrix $\mathbf{R}_{\frac{N}{2}}$ kann nun in orthogonale $2 \times 2$ Blöcke zerlegt werden. Für eine 8-Punkte DCT ($N = 8$) ergibt sich somit die folgende Matrix:

$$\begin{aligned} \mathbf{R}_4 = & \begin{pmatrix} \cos\left(\frac{-7\pi}{16}\right) & 0 & 0 & -\sin\left(\frac{-7\pi}{16}\right) \\ 0 & \cos\left(\frac{-3\pi}{16}\right) & -\sin\left(\frac{-3\pi}{16}\right) & 0 \\ 0 & \sin\left(\frac{-3\pi}{16}\right) & \cos\left(\frac{-3\pi}{16}\right) & 0 \\ \sin\left(\frac{-7\pi}{16}\right) & 0 & 0 & \cos\left(\frac{-7\pi}{16}\right) \end{pmatrix} \\ & \cdot \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -\sin\left(\frac{\pi}{4}\right) & \cos\left(\frac{\pi}{4}\right) & 0 \\ 0 & \cos\left(\frac{\pi}{4}\right) & \sin\left(\frac{\pi}{4}\right) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad . \end{aligned} \tag{7.7}$$

Die Rekursion startet mit der 2-Punkte DCT

$$\mathbf{C}_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad . \tag{7.8}$$

Aus dieser Matrixfaktorisierung lässt sich schließlich der Signalflussgraph für die Schaltungsrealisierung ableiten. Im Falle der 8-Punkte DCT ergibt sich der Signalflussgraph nach Bild 7.1. Aus dem Signalflussgraphen ist ersichtlich, dass nur orthogonale $2 \times 2$ Rotationen ausgeführt werden müssen. Diese lassen sich als eine Sequenz von einfach realisierbaren Teilrotationen darstellen. Jede Teilrotation führt eine Drehung um einen bestimmten Winkel aus. Damit können beliebige Winkel approximiert werden. Eine Quantisierung erfolgt dadurch, dass die Winkel nur bis zu einer gewissen Genauigkeit approximiert werden sollen, um die Anzahl der zu realisierenden Teilrotationen möglichst gering zu halten. Alternativ dazu können auch die Multipliziererkoeffizienten approximiert werden. Die Koeffizienten entsprechen den Sinus- bzw. Kosinuswerten der Rotationswinkel. Im Signalflussgraphen in Bild 7.1 sind die Koeffizienten für die 8-Punkte DCT eingetragen. Um auch hier den Realisierungsaufwand zu begrenzen, ist eine Quantisierung der Koeffizienten nötig. Die mit jeder Quantisierung verbundenen Fehler bei der Transformation müssen innerhalb der vom Standard vorgeschriebenen Grenzen liegen. Im Folgenden soll untersucht werden, welche Realisierung – mit Winkel- oder Koeffizientenquantisierung – die Schaltung mit der geringeren Verlustleistung ergibt.

## 7.2 Winkelapproximation

Durch die Matrixfaktorisierung nach Chen et al. [17] kann eine DCT unter ausschließlicher Verwendung orthogonaler $2 \times 2$-Rotationen realisiert werden. Orthogonale $2 \times 2$ Rotationen lassen sich allgemein durch eine Rotationsmatrix $\mathbf{G}$ beschreiben:

$$\mathbf{G} = \begin{pmatrix} \cos(\Phi) & -\sin(\Phi) \\ \sin(\Phi) & \cos(\Phi) \end{pmatrix} \quad . \tag{7.9}$$

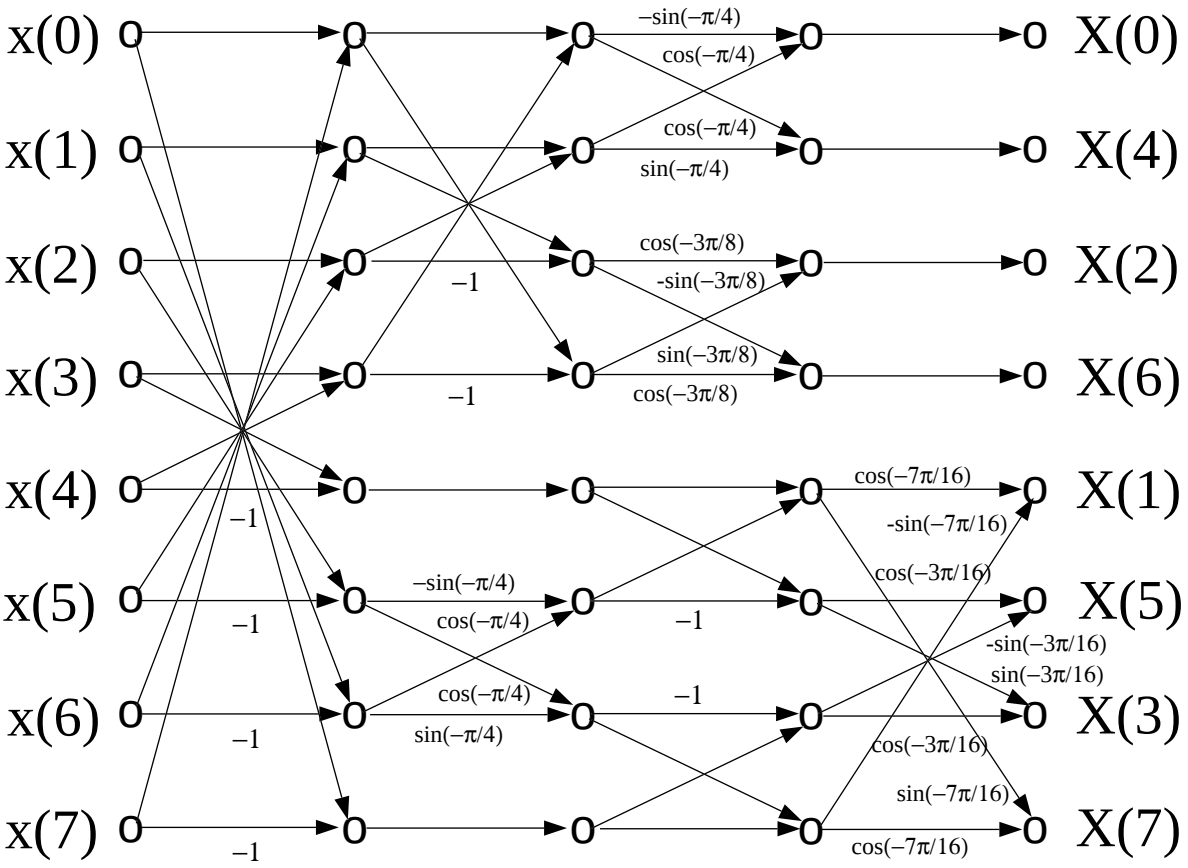



Bild 7.1. Signalflussgraph einer 8-Punkte DCT.

Dabei ist $\Phi$ der Rotationswinkel. Die Rotationsmatrix $\mathbf{G}$ kann in ein Produkt von Teilrotationen aufgespalten werden, die sich durch einfache Schiebe- und Addieroperationen ausführen lassen. In der Literatur werden die Teilrotationen auch als $\mu$-Rotationen bezeichnet. Es existieren verschiedene Klassen von $\mu$-Rotationen [26, 53], mit denen jeweils unterschiedliche Winkel realisiert werden können. Für die hier vorgestellte 8-Punkte DCT sind lediglich zwei Klassen von $\mu$-Rotationen nötig, um eine standardkonforme Implementierung zu erzielen. Klasse I der $\mu$-Rotationen entspricht den Stufen einer CORDIC-Sequenz [20]. Diese Klasse ist durch

$$\mathbf{G_i^I} = \frac{1}{\sqrt{1+2^{-2i}}} \cdot \begin{pmatrix} 1 & \mp 2^{-i} \\ \pm 2^{-i} & 1 \end{pmatrix} \quad . \tag{7.10}$$

definiert. Durch eine einzelne Rotation dieser Klasse lassen sich die Winkel $\alpha_i^{\mathrm{I}} = \pm \arctan\left(2^{-i}\right)$ realisieren. Um eine feinere Abstufung der Winkel zu erreichen, wird eine zweite Klasse von $\mu$-Rotationen benötigt:

$$\mathbf{G_i^{II}} = \frac{1}{\sqrt{1+2^{-4i-2}}} \cdot \begin{pmatrix} 1-2^{-2i-1} & \mp 2^{-i} \\ \pm 2^{-i} & 1-2^{-2i-1} \end{pmatrix} \quad . \tag{7.11}$$

Die möglichen Rotationswinkel für eine $\mu$-Rotation der Klasse II sind $\alpha_i^{\mathrm{II}} = \pm \arctan\left(\frac{2^{-1}}{1-2^{-2i-1}}\right)$. Aus den Definitionen der Teilrotationen wird ersichtlich, dass bei binärer Zahlendarstellung nur einfache Schiebe- und Addierschritte zur Realisierung nötig sind, da in den Gleichungen nur Potenzen zur Basis 2 auftreten. Ein wesentlicher Vorteil dieser Art

der Realisierung liegt darin, dass durch die ausschließliche Verwendung orthogonaler Teilrotationen die gesamte approximierte Transformation ebenfalls orthogonal bleibt. Dadurch wird die exakte Rekonstruierbarkeit des Signals nach der Rücktransformation garantiert.

Mit den zwei Klassen von $\mu$-Rotationen sollen im Folgenden alle Rotationen im Signalflussgraphen aus Bild 7.1 approximiert werden. Für die Approximation der Rotationswinkel der hier betrachteten DCT ist es nötig, die Rotationswinkel schrittweise soweit zu verfeinern, dass die gesamte Transformation den Anforderungen des IEEE-Standards genügt. Der Standard sieht für die Pixel eines zu kodierenden Bildes eine Auflösung von 9 Bit vor. Anhand der Gleichungen (7.1) und (7.3) ist nachvollziehbar, dass die Eingangsdaten durch die Transformation höchstens mit einem Faktor $\sqrt{\frac{2}{N}} \cdot \frac{N}{\sqrt{2}} = \sqrt{N}$ multipliziert werden. Für $N = 8$ ergibt sich demnach ein maximaler Faktor von 2.83. Um einen dabei möglichen numerischen Überlauf zu verhindern, müssen zwei zusätzliche Bit am MSB (Most Significant Bit) angefügt werden. Bei einer Schiebeoperation entsprechend einer Multiplikation der Daten mit dem Faktor $2^i$ werden alle Bit um $i$ Positionen in Richtung der niederwertigeren Stellen verschoben. Daher müssen zusätzliche Bit an den niederwertigsten Stellen vorgesehen werden, um die Quantisierungsfehler, die aufgrund der begrenzten Datenwortbreite auftreten können, innerhalb der vom Standard geforderten Grenzen zu halten.

Der Entwurf des Signalflussgraphen der zu realisierenden Schaltung mit approximierten Rotationen erfolgt iterativ. Zunächst wird dabei von einer unendlichen Datenwortbreite ausgegangen und Quantisierungsfehler somit nicht berücksichtigt. Für die Startlösung wird jeder Winkel nur durch eine $\mu$-Rotation der Klasse $I$ approximiert. Die daraus erhaltene Schaltung genügt jedoch noch nicht den Anforderungen des Standards. Es folgt eine schrittweise Verfeinerung der Approximation, bis die Transformation den Standard erfüllt. Im letzten Schritt wird die Datenwortbreite soweit verringert, bis der maximal tolerierbare Quantisierungsfehler gerade noch innerhalb der geforderten Grenzen liegt. Nach diesem Verfahren ergibt sich der Signalflussgraph nach Bild 7.2. Es ist eine Datenwortbreite von 16 Bit erforderlich, die sich aus der 9-Bit-Darstellung für die Pixel, 2 Bit für Überlauf und 5 Bit für die Reduzierung der Quantisierungsfehler zusammensetzt. In Tabelle 7.1 sind die exakten und approximierten Winkel gegenübergestellt. Die Tabelle zeigt ausserdem die Zusammensetzung der approximierten Winkel aus den $\mu$-Rotationen der beiden Klassen. Der Skalierungsfaktor der $45^o$

| exakter Winkel $\alpha_{\text{ex}}$ | approximierter Winkel $\alpha_{\text{approx}}$ | Anzahl der $\mu$-Rotationen |
|---|---|---|
| $33.75^o$ | $33.7342^o$ | 4<br>$\alpha_{\text{approx}} = \alpha_1^{II} + \alpha_3^I - \alpha_4^{II} + \alpha_7^I$ |
| $45^o$ | $45^o$ | 1<br>$\alpha_{\text{approx}} = \alpha_0^I$ |
| $67.5^o$ | $67.508^o$ | 3<br>$\alpha_{\text{approx}} = \alpha_0^I + \alpha_1^{II} - \alpha_3^I - \alpha_9^I$ |
| $78.75^o$ | $78.7885^o$ | 3<br>$\alpha_{\text{approx}} = 90^o - (\alpha_3^{II} + \alpha_7^I + \alpha_4^{II})$ |

Tabelle 7.1. Exakte und approximierte Rotationswinkel für die 8-Punkte DCT. Unter der nötigen Anzahl von $\mu$-Rotationen ist angegeben, welche $\mu$-Winkel jeweils zum approximierten Gesamtwinkel führen.

Rotation beträgt $\frac{1}{\sqrt{1+2^{-2\cdot 0}}} = \frac{1}{\sqrt{2}}$. Entwickelt man diesen mit der erforderlichen Genauigkeit

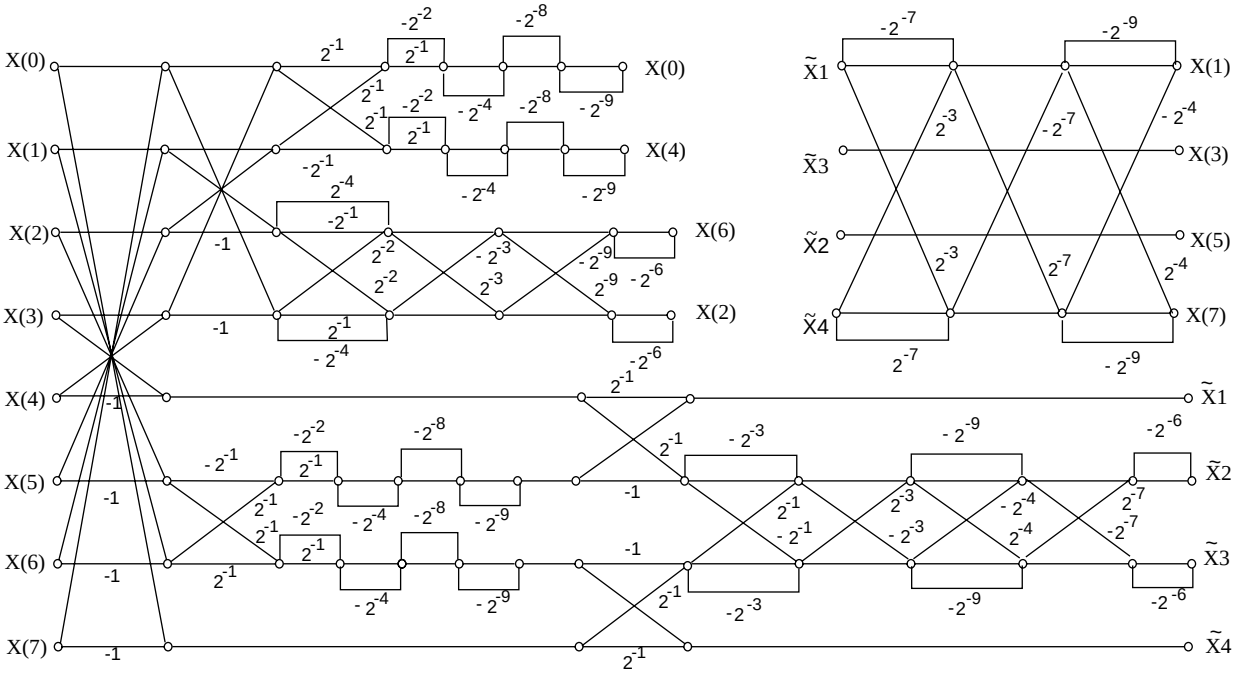



Bild 7.2. Signalflussgraph einer 8-Punkte DCT mit approximierten Rotationen.

in eine Potenzreihe, so ergibt sich:

$$\frac{1}{\sqrt{2}} \approx 2^{-1} + 2^{-3} + 2^{-4} + 2^{-6} + 2^{-8} \quad . \tag{7.12}$$

Alternativ dazu kann der Skalierungsfaktor auch in Produktform dargestellt werden. Mit der erforderlichen Genauigkeit ergibt sich dabei:

$$\frac{1}{\sqrt{2}} \approx (2^{-1} + 2^{-1})(1 - 2^{-4})(1 + 2^{-8})(1 + 2^{-9}) \quad . \tag{7.13}$$

Beide Darstellungen erfordern bei der Realisierung 4 Schiebe- und Addierschritte. Jedoch ergibt sich für die Produktdarstellung nach (7.13) eine günstigere Verdrahtungsstruktur. Diese Darstellung wurde auch zur Realisierung des Signalflussgraphen aus Bild 7.2 angewendet. Bei einer Realisierung nach der Reihendarstellung gemäß (7.12) muss das zu skalierende Signal $x$ an jeden Addierer geführt werden. Das skalierte Signal $\hat{x}$ ergibt sich bei Skalierung mit $\frac{1}{\sqrt{2}}$ entsprechend zu $\hat{x} = 2^{-1}x + 2^{-3}x + 2^{-4}x + 2^{-6}x + 2^{-8}x$. Die Leitungen müssen dazu im Layout um die jeweils vorhergehenden Addierer herum- oder unter Verwendung einer eigenen Metalllage darüberhinweggeführt werden. Bei gleicher Anzahl von Schiebe- und Addierschritten ist deshalb eine Realisierung des Skalierungsfaktors in Produktdarstellung vorzuziehen.

Im nächsten Abschnitt wird gezeigt, dass, je nach Faktor, entweder die Produktdarstellung oder die Reihendarstellung eine geringere Anzahl von Schiebe- und Addierschritten erfordert. Durch den Übergang auf eine andere Darstellungsform kann gegebenenfalls eine günstigere Realisierung erreicht werden.

## 7.3 Koeffizientenapproximation

Eine direkte Realisierung des Signalflussgraphen aus Bild 7.1 erhält man, indem man die Sinus- und Kosinuswerte der Rotationswinkel als Multipliziererkoeffizienten auffasst. Da die Koeffizienten $c_i$ mit der Architektur der DCT nach der Matrixfaktorisierung in (7.7) festgelegt sind, können sie als fest verdrahtete Multiplikationen realisiert werden. Der Wertebereich R der Koeffizienten ist $\mathrm{R} = [-1, 1]$. Bei binärer Kodierung erfolgt die Darstellung als Summe aus Potenzen zur Basis 2:

$$c_i = \sigma_i \sum_{k=0}^{\infty} \alpha_{i,k} 2^{-k} \quad , \tag{7.14}$$

wobei $\sigma_i \in \{-1, 1\}$ und $\alpha_{i,k} \in \{0, 1\}$. Ein *Signed-Digit*-Darstellung (SDNR-Darstellung) erhält man, wenn man die Menge der zulässigen Werte für $\alpha_{i,k}$ erweitert, sodass auch negative Wertigkeiten zugelassen sind. Es gilt dann $\alpha_{i,k} \in \{-1, 0, 1\}$. Dadurch ist es möglich die Anzahl der von Null verschiedenen Stellen in der binären Darstellung zu reduzieren. Eine *kanonische* SDNR-Darstellung der Koeffizienten enthält die minimal mögliche Anzahl von Stellen ungleich Null. Da nur die von Null verschiedenen Summanden in (7.14) realisiert werden müssen, ergibt sich für die kanonische SDNR-Darstellung ein im Allgemeinen geringerer Realisierungsaufwand als bei einfacher binärer Kodierung. Um den Realisierungsaufwand weiter zu verringern, ist es nötig, ein Quantisierung der Koeffizienten durchzuführen, da diese natürlich nur innerhalb einer bestimmten Genauigkeit dargestellt werden müssen. Die erforderliche Genauigkeit ergibt sich wieder aus den Anforderungen des IEEE-Standards für die DCT. Das Vorgehen ist dabei ähnlich wie bei der Winkelquantisierung. Die Auflösung der Pixel beträgt 9 Bit. Zwei zusätzliche Bit an den höchstwertigen Stellen sind nötig, um einen Überlauf innerhalb der Transformation zu verhindern. Zunächst wird wieder eine unendliche Anzahl von Daten-Bit an den niederwertigsten Stellen angenommen, d.h. es erfolgt keine Quantisierung der Daten. Nun wird die Darstellung der Koeffizienten beginnend mit nur einer Stelle hinter dem Komma soweit verfeinert, bis die Anforderungen des Standards von der Gesamtransformation erfüllt werden. Erst dann erfolgt die Ermittlung der nötigen Datenwortbreite, indem die Datenworte nach jeder Schiebeoperation von den niederwertigsten Stellen her durch Zweierkomplement-Endschneiden auf eine bestimmte Anzahl von Bit quantisiert werden. Die Anzahl der nach jeder Schiebeoperation abgeschnittenen Bit wird schrittweise solange erhöht, bis die Anforderungen des Standards gerade noch erfüllt werden.

Als Beispiel für die Quantisierung wird im Folgenden die Darstellung des Koeffizienten $c_1 = \sin(\frac{3\pi}{8})$ gezeigt. Die binäre Darstellung dieses Koeffizienten ergibt bei einer Genauigkeit von neun Nachkommastellen:

$$c_1 = 0.111011001_2 \quad ,$$

die kanonische SDNR-Darstellung (CSD) mit der minimalen Anzahl von Elementen ungleich Null lautet

$$c_1 = 1.000\bar{1}0\bar{1}001_{CSD} \quad .$$

Diese Darstellung ist äquivalent mit

$$c_1 = 1 - 2^{-4} - 2^{-6} + 2^{-9} \quad .$$

Zur Realisierung in dieser Darstellungsform sind drei Schiebe- und Addierschritte nötig. Subtraktionen werden durch Komplementbildung auf Additionen übergeführt. Die gesamte

Transformation bleibt jedoch standardkonform, wenn der Koeffizient in faktorisierter Form realisiert wird:

$$c_1 = (1 - 2^{-4})(1 - 2^{-6}) \quad .$$

Bei dieser Darstellung sind lediglich zwei Schiebe- und Addierschritte nötig. Zudem gestaltet sich die Verdrahtung bei der Produktdarstellung einfacher, wie anhand von Bild 7.3 deutlich wird. Welche Art der Koeffizientendarstellung für die Realisierung am günstigsten ist, hängt

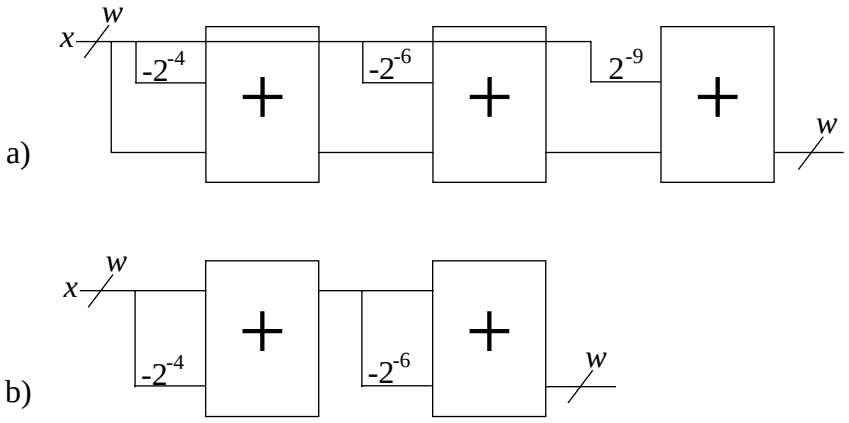



Bild 7.3. Realisierung des Koeffizienten $c_1 = \sin(\frac{3\pi}{8})$. a) Summe aus Potenzen zur Basis 2: $c_1 = 1-2^{-4}-2^{-6}+2^9$, b) Produktdarstellung $c_1 = (1-2^{-4})(1-2^{-6})$. $w$ bezeichnet die Datenwortbreite.

vom jeweiligen Koeffizienten selbst ab und muss von Fall zu Fall entschieden werden. In Tabelle 7.2 sind die Summen- und Produktdarstellungen für alle Koeffizienten der 8-Punkte DCT aufgeführt. Die Koeffizienten sind dabei so quantisiert, dass die Gesamttransformation gerade noch die Anforderungen des Standards erfüllt. Für die letztendliche Realisierung wird immer die Darstellung gewählt, die die wenigsten Schiebe- und Addierschritte benötigt. Um die Quantisierungsfehler nach jeder Schiebeoperation innerhalb der durch den Standard

| Koeffizient | Summenform | Produktform |
|---|---|---|
| $\sin(\frac{\pi}{4})$ | $2^{-1}+2^{-3}+2^{-4}+2^{-6}+2^{-8}$ | • $(2^{-1}+2^{-2})(1-2^{-4})(1+2^{-8})(1+2^{-9})$ |
| $\cos(\frac{\pi}{4})$ | $2^{-1}+2^{-3}+2^{-4}+2^{-6}+2^{-8}$ | • $(2^{-1}+2^{-2})(1-2^{-4})(1+2^{-8})(1+2^{-9})$ |
| $\sin(\frac{3\pi}{8})$ | $1-2^{-4}-2^{-6}+2^{-9}$ | • $(1-2^{-4})(1-2^{-6})$ |
| $\cos(\frac{3\pi}{8})$ | • $2^{-2}+2^{-3}+2^{-7}$ | $(2^{-2}+2^{-3})(1+2^{-5})(1+2^{-8})$ |
| $\sin(\frac{3\pi}{16})$ | • $2^{-1}+2^{-4}-2^{-7}$ | $(2^{-1}+2^{-4})(1-2^{-3})(1-2^{-8})$ |
| $\cos(\frac{3\pi}{16})$ | $1-2^{-3}-2^{-4}+2^{-6}+2^{-8}$ | • $(1-2^{-3})(1-2^{-5})(1-2^{-6})(1-2^{-8})$ |
| $\sin(\frac{7\pi}{16})$ | $1-2^{-6}-2^{-8}$ | • $(1-2^{-6})(1-2^{-8})$ |
| $\cos(\frac{7\pi}{16})$ | • $2^{-3}+2^{-4}+2^{-7}$ | $(2^{-3}+2^{-4})(1+2^{-5})(1+2^{-7})$ |

Tabelle 7.2. Koeffizientendarstellung in Summen- und Produktform, jeweils bis zu einer Genauigkeit, bei der der IEEE-Standard für die 8-Punkte DCT gerade noch eingehalten wird. Die für die VLSI-Realisierung ausgewählten Darstellungen sind mit • gekennzeichnet.

gegebenen Grenzen zu halten, müssen an den niederwertigsten Stellen der Daten zusätzliche Bit zur Verfügung gestellt werden. Bei der Quantisierung der Koeffizienten wird zunächst wieder von einer unendlichen Ausdehnung der Datenworte in Richtung niederwertiger Bits ausgegangen. Für die Realisierung muss die Datenwortbreite auf das minimal mögliche Maß

beschränkt werden, um den Hardware-Aufwand gering zu halten. Bei Anwendung der oben gezeigten Koeffizientenquantisierung ergibt sich eine minimal nötige Datenwortbreite von 15 Bit. Diese setzt sich aus den 9 Bit für die Darstellung der Pixel, 2 Bit für Überlauf und weiteren 4 Bit für die Kompensation von Quantisierungseffekten zusammen. Zum Vergleich: bei der Winkelapproximation war eine Datenwortbreite von 16 Bit erforderlich. Offensichtlich ist die Realisierung mit quantisierten Koeffizienten weniger empfindlich gegenüber einer Datenquantisierung als die Realisierung mit quantisierten Rotationswinkeln. In Tabelle 7.3 sind die nötigen Erweiterungen der Datenwortbreite für beide Realisierungen dargestellt.

| Grund | Position | | Winkel- | Koeffizienten- |
|---|---|---|---|---|
| | MSB | LSB | approximation | approximation |
| Überlauf | x | | 2 | 2 |
| Quantisierung der Daten | | x | 5 | 4 |

Tabelle 7.3. Erweiterungen der Datenwortbreite für Überlauf und Reduzierung von Quantisierungsfehlern.

## 7.4 VLSI-Realisierung

Nach dem Entwurf der DCT auf Architekturebene wird in diesem Abschnitt die Realisierung der Schaltung in Form eines Maskenlayouts vorgestellt. Ein Vergleich der beiden unterschiedlichen Architekturen mit Winkel- und Koeffizientenapproximation zeigt zunächst, dass die direkte Realisierung des Signalflussgraphen der DCT nach Bild 7.1 mit approximierten Koeffizienten einen offensichtlich geringeren Aufwand erfordert. Der Grund dafür ist zum einen die bei dieser Architektur geringere Datenwortbreite. Zum anderen zeigt sich aber auch, dass darüberhinaus im Vergleich zur Winkelapproximation weniger Schiebe- und Addierschritte nötig sind, um die gesamte Transformation auszuführen. Insgesamt sind bei der Version mit den approximierten Winkeln 74 Schiebe- und Addierschritte nötig, gegenüber 72 Schiebe- und Addierschritten bei der Version mit den approximierten Koeffizienten.

Um die Datendurchsatzrate zu erhöhen und gleichzeitig die Anzahl unerwünschter Schaltvorgänge aufgrund von Laufzeitunterschieden zu verringern, sollen mittels Pipelining Registerstufen in die Schaltung eingeführt werden. Dabei ist es sinnvoll, anstatt ganzer Register nur Latches, bestehend aus einem Transmission-Gate und einem Inverter, zu verwenden. Da die Eingangskapazitäten der Gatter und Volladdierer in der Schaltung zur Ladungsspeicherung genutzt werden können, lassen sich die Inverter der Latches einsparen, wie dies bereits im Abschnitt 4.1.1 erläutert wurde. Das Pipelining soll so erfolgen, dass nach jeder Schiebe- und Addieroperation eine Register- oder Latch-Stufe platziert wird. Damit wird eine auf Wortebene vollsystolische Realisierung erreicht. Die minimal mögliche Taktperiode hängt in diesem Fall nur noch von der Datenwortbreite $w$ ab. Bezeichnet $d_{\mathrm{A}}$ die Verzögerungszeit eines Volladdierers, so ergibt sich die minimal mögliche Taktperiode zu $\Psi = w \cdot d_{\mathrm{A}}$. Bei vollsystolischer Realisierung ergibt sich für die Architekturen mit Winkel- und Koeffizientenapproximation eine unterschiedliche Anzahl von benötigten Pipeline-Stufen. Dies liegt an der unterschiedlichen Anzahl von hintereinander auszuführenden Schiebe- und Addierschritten. Für die DCT mit approximierten Rotationswinkeln sind bei vollsystolischer Realisierung 19 Pipeline-Stufen nötig. Das Pipelining der DCT mit approximierten Koeffizienten erfordert

dagegen nur 11 Pipeline-Stufen. Dies lässt einen geringeren Flächenbedarf bei der Realisierung des Layouts erwarten. In Bild 7.4 sind die Taktschemen für das Pipelining der beiden unterschiedlichen Realisierungen mit Latches als getakteten Elementen dargestellt.

Werden die Floorpläne der Schaltungen so gestaltet, dass die beiden Additionen am Ende jedes Rotors gemäß den Signalflußgraphen übereinander angeordnet sind, so ergeben sich aufgrund der Überkreuzungen der Datenleitungen im Layout Verdrahtungskanäle, deren Breite durch die Datenwortbreite $w$ bestimmt ist. Durch eine wortweise Verschachtelung der

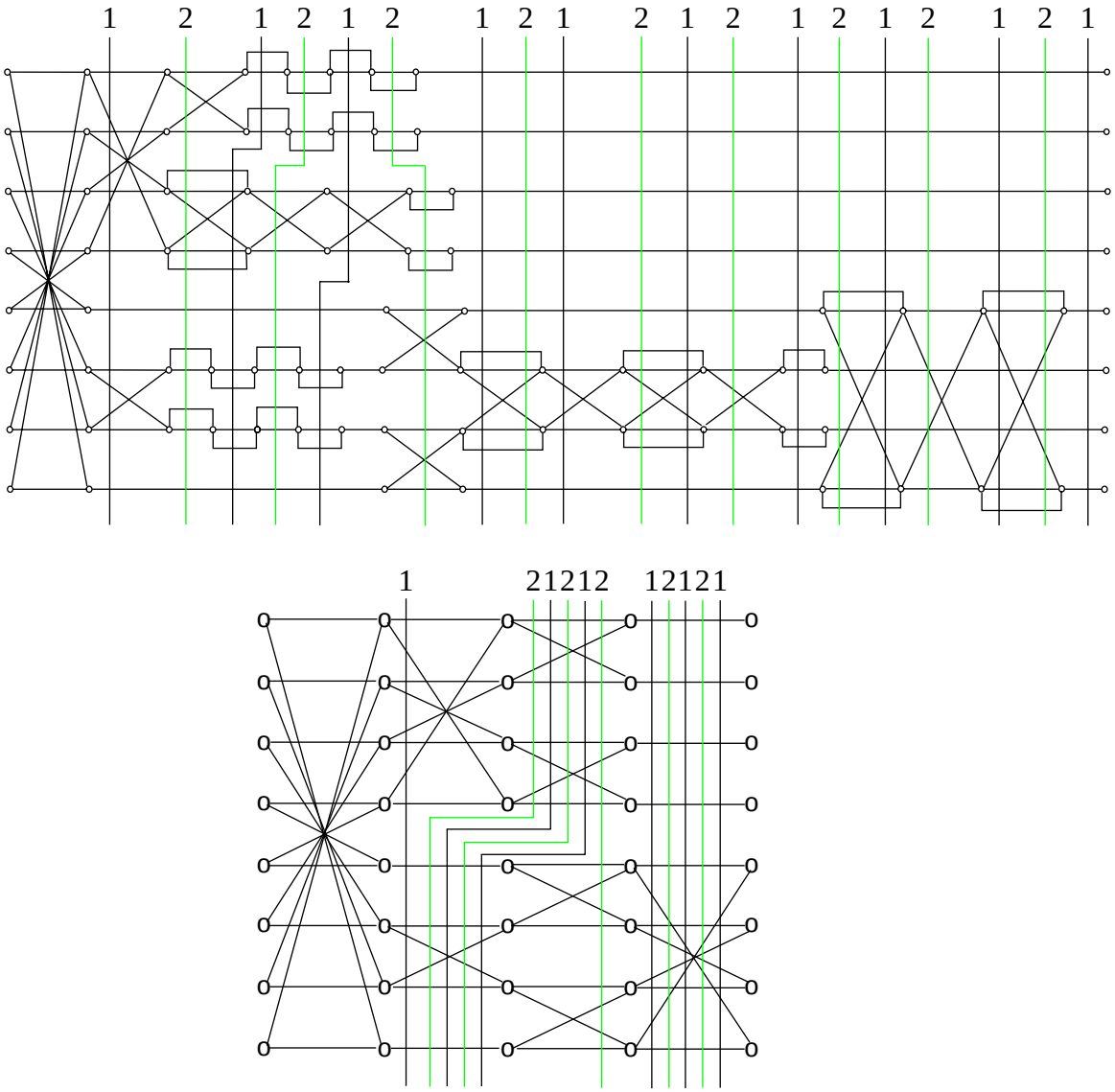



Bild 7.4. Pipelining bei vollsystolischer Realisierung der DCT mit Winkel- und Koeffizientenapproximation und Verwendung von Latches als getakteten Elementen. Die beiden unterschiedlichen Taktphasen zur Ansteuerung der Latches sind mit "1" und "2" gekennzeichnet.

Addierer ist es möglich, den Flächenbedarf für die Verdrahtung zu reduzieren. In Bild 7.5 ist das Prinzip für die wortweise Verschachtelung eines Rotors dargestellt. Der Aufbau der Layouts erfolgt in einer Bit-Slice-Struktur, d. h. die gesamte DCT wird zunächst nur für eine Datenwortbreite von einem Bit realisiert. Für die nötigen Schiebeverdrahtungen werden spezielle Verdrahtungszellen verwendet, die in [81] und [76] detailliert beschrieben sind. Die

Gesamtlayouts der Schaltungen entstehen durch paralleles Aneinanderfügen einer der Datenwortbreite entsprechenden Anzahl von Bit-Slices. Die benötigte Chip-Fläche für das Ge-

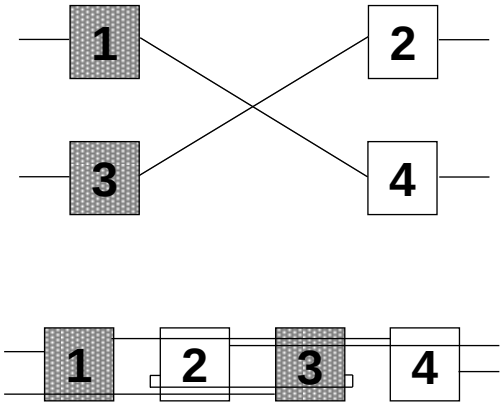



Bild 7.5. Wortweise Verschachtelung eines Rotors.

samtlayout der 8-Punkte DCT mit approximierten Rotationswinkeln in $0.25\mu m$-Technologie beträgt $0.59mm^2$, die Anzahl der Transistoren ist 43456. Die benötigte Chip-Fläche für die 8-Punkte DCT mit approximierten Koeffizienten beträgt $0.52mm^2$, die Anzahl der Transistoren ist hier 37740.

Eine Simulation der beiden Realisierungen mit PowerMill zeigt den geringeren Leistungsverbrauch der DCT mit den approximierten Koeffizienten. Die Verlustleistungsbestimmung mit der in Kapitel 6 vorgestellten DCM-Methode bestätigt dieses Ergebnis. In Tabelle 7.4 sind die Ergebnisse aus den beiden DCT-Realisierungen zusammengefasst. Die in der Ta-

| | DCT mit approximierten Rotationswinkeln | DCT mit approximierten Multipliziererkoeffizienten |
|---|---|---|
| Anzahl der Schiebe- und Addierschritte | 74 | 72 |
| Anzahl der Volladdierer | 1184 | 1080 |
| Latenzzeit (Taktperioden) | 19 | 11 |
| Anzahl der Transistoren | 43456 | 37740 |
| Chip-Fläche ($mm^2$) | 0.59 | 0.52 |
| Leistung in $mW$ (PowerMill) | 0.91 | 0.76 |
| Leistung in $mW$ (DCM) | 0.86 | 0.74 |

Tabelle 7.4. Vergleich der Realisierungen einer 8-Punkte DCT mit Winkel- und Koeffizientenapproximation jeweils in $0.25\mu m$-Technologie.

belle angegebenen Latenzzeiten gelten bei Verwendung ganzer Register für das Pipelining.

Werden, wie in der eigentlichen Realisierung nach den Signalflussgraphen in Bild 7.4, nur Transmission-Gates als getaktete Elemente verwendet, so halbieren sich die Latenzzeiten.

Das Fazit aus den hier gezeigten Ergebnissen ist, dass die Approximation der Rotationswinkel zwar eine effizientere Implementierung von Wavelettransformationen erlaubt, wie dies aus den Arbeiten [54, 55, 80] und [81] hervorgeht, für die standardkonforme Realisierung einer DCT bezüglich Verlustleistung und Flächenbedarf jedoch nicht die günstigste Wahl darstellt. Verantwortlich dafür ist mitunter auch der höhere Aufwand für das Pipelining und die daraus resultierende größere Belastung der Taktversorgung.

Das Gesamtlayout für die 8-Punkte DCT mit approximierten Rotationswinkeln ist in Bild 7.6 dargestellt. Bild 7.7 zeigt das Gesamtlayout für die 8-Punkte DCT mit approximierten Multipliziererkoeffizienten. Beide Layouts wurden nach der Full Custom Design-Methode realisiert.

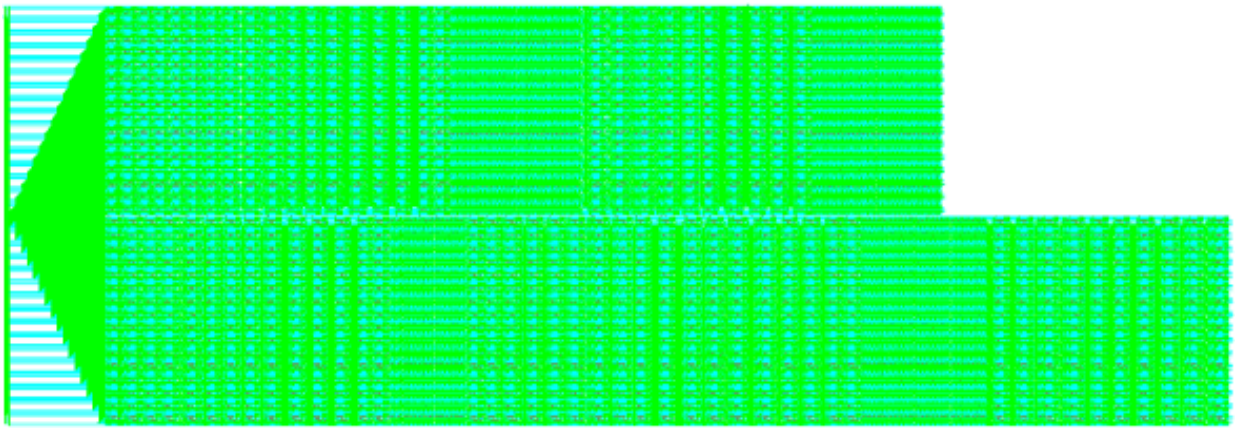


Bild 7.6. Gesamtlayout der 8-Punkte DCT mit approximierten Rotationswinkeln.

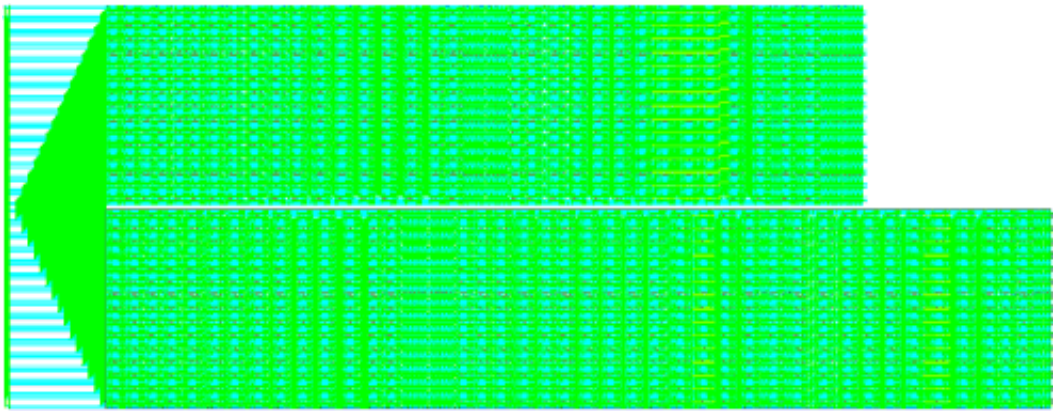


Bild 7.7. Gesamtlayout der 8-Punkte DCT mit approximierten Koeffizienten.

# 8. Zusammenfassung

Das Thema der vorliegenden Arbeit ist der Entwurf integrierbarer digitaler CMOS-Schaltungen mit niedriger Verlustleistung. Einen besonderen Schwerpunkt bilden dabei Schaltungen, die typischerweise in digitalen Signalverarbeitungssystemen zum Einsatz kommen. Es wurden zum einen allgemein anwendbare Methoden und Verfahrensweisen vorgestellt, zum anderen deren praktischer Einsatz anhand von Beispielen gezeigt und die Effektivität der Verfahren untersucht. Die allgemein unter dem Begriff *Low-Power-Design* zusammengefassten Entwurfsmethoden lassen sich in zwei große Teilbereiche einordnen:

1) Methoden zur Bestimmung der Verlustleistung und
2) Methoden zur Verringerung der Verlustleistung elektrischer Schaltungen.

In dieser Arbeit wurden sowohl Methoden zur Bestimmung, als auch solche zur Verringerung der Verlustleistung erarbeitet. Ihre Anwendbarkeit ist nicht auf einen bestimmten Entwurfsstil beschränkt; sie sind gleichermaßen für die High-Level-Synthese als auch für den Full-Custom-Entwurf einsetzbar.

Um ein hinsichtlich minimaler Verlustleistung optimales Resultat zu erhalten, ist es notwendig, beim Top-Down-Entwurf einer Schaltung auf allen Entwurfsebenen die dabei zur Verfügung stehenden Methoden zur Verringerung der Verlustleistung anzuwenden. Es wurden neue Verfahren sowohl für höhere als auch für niedrigere Entwurfsebenen entwickelt und in dieser Arbeit vorgestellt. Auf der Logikebene kann durch die *direkte Glitch-Analyse* zeitbehafteter Boolescher Funktionen die durch Glitches verursachte unerwünschte Schaltaktivität schnell abgeschätzt werden. Für die zeitbehafteten Booleschen Funktionen wurden Bedingungen angegeben, unter denen Glitches auftreten können. Gegebene zeitbehaftete Boolesche Funktionen können programmunterstützt auf diese Bedingungen hin untersucht, und dadurch die Glitchaktivität der zugehörigen Schaltungen ermittelt werden. Die Genauigkeit dieses Verfahrens ist für die Bestimmung von Knotengewichten zur Optimierung der Schaltung hinsichtlich der Glitch-Aktivität mittels Retiming ausreichend.

Retiming und Pipelining wurden als Methoden zur Reduzierung der Verlustleistung vorgestellt. Im Gegensatz zu klassischen Retiming-Verfahren ist hier nicht die Minimierung der Taktperiode sondern die Minimierung der Glitches das Ziel. Als Gewichte an den Knoten des Signalflussgraphen einer Schaltung dienen hier nicht, wie bei der klassischen Anwendung, die Verzögerungszeiten, sondern die Glitch-Aktivitäten an den Knotenausgängen. Zur Bestimmung dieser *Glitch-Gewichte* können wieder die oben erwähnten Verfahren angewendet werden. Es wurde gezeigt, dass durch die Verwendung von Registern als Barrieren für Glitches und zur Synchronisation unterschiedlicher Signallaufzeiten, die wiederum zu Glitches führen können, die Verlustleistung von Schaltungen reduziert werden kann. Die gezielte

Umverteilung der Register durch Retiming ergab bei den untersuchten Datenpfaden eine Verringerung der Verlustleistung um bis zu 40%.

Als ein Verfahren zur Verringerung der Verlustleistung auf Transistorebene wurde die Dimensionierung von Transistoren zum Abgleich unterschiedlicher Signallaufzeiten vorgestellt. Dieses neue Verfahren basiert auf einer Modellierung der Schaltung auf der höher liegenden Gatterebene, wodurch sich die Anzahl der freien Parameter für die Optimierung erheblich verringern lässt. Optimierungsvariablen sind die Kanalweiten und -längen der Transistoren. Die zu minimierende Zielfunktion enthält neben dem Laufzeitunterschied der Signale auch die durch parasitäre Kapazitäten hervorgerufene Verlustleistung als Optimierungsziel. Das resultierende Optimierungsproblem gehört zur Klasse der Mehrziel- oder Vektoroptimierungsprobleme. Durch die Einführung eines Parameters, mit dem beide Optimierungsziele gewichtet werden, gelingt es, das Vektoroptimierungsproblem auf ein skalares Optimierungsproblem überzuführen. Zur Lösung dieses skalaren Optimierungsproblems stehen dann eine Reihe von Standardmethoden zur Verfügung. Das Optimierungsverfahren wurde in einen Algorithmus eingebunden, der einen Abgleich der Verzögerungszeiten aller Pfade in einer Schaltung vornimmt. Der kritische Pfad der Schaltung bleibt dabei unverändert, d. h. die zeitlichen Anforderungen an die Gesamtschaltung werden auch nach der Anwendung des Verfahrens eingehalten. Die Ergebnisse zeigen eine erhebliche Verringerung der Verlustleistung bei einem in geringerem Maße größeren Flächenbedarf. In Anbetracht der vielversprechenden Ergebnisse erscheint die weitere Erforschung dieses Themas als lohnend. Ein neuer Gesichtspunkt diesbezüglich ist z. B. das Einfügen von Transistoren in die Gatter, die dann lediglich als Verzögerungselemente dienen, ohne die eigentliche Funktion zu beeinflussen. Anders als beim bisherigen Ansatz ließe sich dadurch vermeiden, dass sich die Ausgangsbelastung vorhergehender Gatter durch die neue Dimensionierung der Transistoren verändert.

Speziell für den High-Level-Syntheseprozess wurde ein Verfahren zur Bestimmung der Verlustleistung präsentiert, das auf der genauen Modellierung der Grundelemente einer Schaltung, so genannten *Basiszellen* basiert. Die Beschreibung der Schaltung erfolgt auf höherer Abstraktionsebene mit VHDL. Das gesamte Verfahren ist in zwei Phasen unterteilt. In einer ersten Phase werden alle Basiszellen charakterisiert. Die Charakterisierung erfolgt durch Simulation der einzelnen Basiszellen auf Transistorebene. Die daraus gewonnen Parameter fließen dann in eine VHDL-Beschreibungen der Basiszellen ein. Wie gezeigt wurde, gelingt es somit, den Leistungsverbrauch und das zeitliche Verhalten der Zellen sehr genau auf höherer Abstraktionsebene nachzubilden. Die erste Phase des Verfahrens muss für jede Zelle nur einmal ausgeführt werden. In der zweiten Phase, der eigentlichen Ausführungsphase, erfolgt die Simulation von beliebigen Schaltungen, die aus den vorher charakterisierten Basiszellen bestehen, mittels eines konventionellen VHDL-Logiksimulators. Die neue Modellierungsmethode erlaubt es, eine genaue Bestimmung der Verlustleistung mit dem Rechenaufwand einer einfachen Logiksimulation durchzuführen. Mit diesem Verfahren können – nur anhand der VHDL-Beschreibungen und bereits vor der eigentlichen Synthese – verschiedene Schaltungskonzepte hinsichtlich des Leistungsverbrauchs verglichen, und so eine frühe Auswahl der günstigsten Realisierung getroffen werden.

Schließlich wurden auf Architekturebene Überlegungen zur effizienten Realisierung von Lattice-Filterstrukturen durchgeführt. Als Beispiel wurde dazu eine zum IEEE-Standard konforme diskrete Kosinustransformation (DCT) gewählt, die sich mit Lattice-Filtern realisieren lässt. Es wurde ein Vergleich hinsichtlich des Hardware-Aufwands und des Leistungsverbrauchs zwischen einer Realisierung mit quantisierten Rotationswinkeln und einer Rea-

lisierung mit quantisierten Multipliziererkoeffizienten angestellt. Für die Realisierung von Wavelettransformationen hat sich schon in früheren Arbeiten die Quantisierung der Rotationswinkel als die effizientere Methode herausgestellt. Für die DCT wurde diese Methode erstmals im Rahmen der vorliegenden Arbeit untersucht. Multiplikationen mit festen Koeffizienten können durch Schiebe- und Addieroperationen realisiert werden. Die Darstellung der Koeffizienten kann als Summe von Potenzen zur Basis 2 oder als Produkt aus Termen, die nur Potenzen zur Basis 2 enthalten, erfolgen. Je nach Koeffizient ist die Summendarstellung oder die Produktdarstellung günstiger bezüglich der Anzahl der Schiebe- und Addierschritte. Durch die individuelle Auswahl der jeweils günstigsten Darstellung für die einzelnen Koeffizienten kann eine sehr effiziente Realisierung der Gesamttransformation erreicht werden. Simulationen haben gezeigt, dass für die Realisierung einer standardkonformen DCT die Quantisierung der Multipliziererkoeffizienten der Quantisierung der Rotationswinkel im Hinblick auf den Leistungsverbrauch und den Realisierungsaufwand vorzuziehen ist.

# Literaturverzeichnis


[1] *IEEE Standard Specifications for the Implementations of 8x8 Inverse Discrete Cosine Transform, IEEE-Std 1180-1990.* The Institute of Electrical and Electronics Engineers, New York, 1991.

[2] Energy Star Computers, Introducing the Energy Star Computers Program. *United States Environmental Protection Agency, EPA 430-F93-049*, November 1993.

[3] W. C. Athas, L. Svensson, J. G. Koller, N. Tzartzanis, and E. Y.-C. Chou. Low-Power Digital Systems Based on Adiabatic-Switching Principles. *IEEE Transactions on Very Large Scale Integration Systems*, 2(4):398–407, Dezember 1994.

[4] A. Avenzienis. Signed Digit Number Representations for Fast Parallel Arithmetic. *IRE Transactions on Electronic Computers*, Seiten 389–400, 1961.

[5] V. A. Bartlett and E. Grass. A Low-Power Concurrent Multiplier-Accumulator Using Conditional Evaluation. $6^{th}$ *IEEE Int. Conference on Electronics, Circuits and Systems, ICECS*, II:629–632, September 1999.

[6] L. Benini and G. De Micheli. Automatic Synthesis of Gated-Clock Sequential Circuits. *IEEE Transactions on Computer-Aided Design*, 15(6):630–643, Juni 1996.

[7] J. Benkoski, E. Vanden Mersch, L. J. M. Claesen, and H. De Man. Timing Verification Using Statically Sensitizable Paths. *IEEE Transactions on Computer-Aided Design*, 9(10):1073–1084, Oktober 1990.

[8] M. Borah, M. J. Irwin, and R. M. Owens. Minimizing Power Consumption of Static CMOS Circuits by Transistor Sizing and Input Reordering. *Proc. Int. Conf. on VLSI Design*, 1995.

[9] M. Borah, R. M. Owens, and M. J. Irwin. Transistor Sizing for Low Power CMOS Circuits. *IEEE Trans. on Computer-Aided Design*, 15(6):665–671, Juni 1996.

[10] R. K. Brayton, G. D. Hachtel, and A. L. Sangiovanni-Vincentelli. A Survey of Optimization Techniques for Integrated Circuit Design. *IEEE Proceedings*, 69(10):1334–1362, Oktober 1981.

[11] R. Burch, F. Najm, P.Yang, and T. Trick. A Monte Carlo Approach for Power Estimation. *IEEE Transactions on VLSI Systems*, 1(1):63–71, März 1993.

[12] T. K. Callaway and Jr. E. E. Swartzlander. Optimizing Arithmetic Elements for Signal Processing. *VLSI Signal Processing*, V:91–100, 1992.

[13] T. K. Callaway and Jr. E. E. Swartzlander. Estimating the Power Consumption of CMOS Adders. *Proc.* $11^{th}$ *Symposium on Computer Arithmetic*, Seiten 210–216, Juni 1993.

[14] A. P. Chandrakasan, R. Allmon, A. Stratakos, and R. W. Brodersen. Design of Portable Systems. *Proc. CICC'94, San Diego*, Seiten 12.1.1.–12.1.8., 1994.

[15] A. P. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey, and R. W. Brodersen. Optimizing Power Using Transformations. *IEEE Transactions on Computer-Aided Design*, 14(1):12–31, Januar 1995.

[16] C.-R. Chang and J.S. Wang. A New High-Speed/Low-Power Dynamic CMOS Logic and Its Application to the Design of an AOI-type ROM. *Proc. IEEE Int. Symp. on Circuits and Systems, ISCAS'99, Orlando*, Mai 1999.

[17] W. H. Chen, C. Harrison Smith, and S. C. Fralick. A Fast Computational Algorithm for the Discrete Cosine Transform. *IEEE Transactions on Communications*, 25(9):1004–1008, September 1977.

[18] L. Claesen, J.-P. Schupp, P. Das, P. Johannes, S. Perremans, and H. De Man. Efficient false path elimination algorithms for timing verification by event graph preprocessing. *Integration, the VLSI journal*, 8:173–187, 1989.

[19] A.-C. Deng, X. Huang, S. Napper, j. Tuan, and J. Benkoski. Simulation Algorithms, Power Estimation and Diagnostics in PowerMill. *Proceedings of PATMOS'95 Workshop*, Seiten 399–410, Oktober 1995.

[20] E. F. Deprettere, A. A. J. De Lange, and P. Dewilde. The Synthesis and Implementation of Signal Processing Application Specific VLSI CORDIC Arrays. *Proc. Int. Conf. on Acoustics, Speech and Signal Processing*, Seiten 974–977, 1990.

[21] M. Eisele and J. Berthold. Dynamic Gate Delay Modelling for Accurate Estimation of Glitch Power at Logic Level. *Proceedings of PATMOS'95 Workshop*, Seiten 190–201, Oktober 1995.

[22] E. Elmore. The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers. *Journal of Applied Physics*, Seiten 55–63, Januar 1948.

[23] M. S. Elrabaa, I. S. Abu-Kather, and M. I. Elmasry. Advanced Low-Power Digital Circuit Techniques. *Kluwer Academic Publishers*, 1997.

[24] A. Fahim and M. Elmasry. A Low-Voltage High-Performance Differential Static Logic (LVDSL) Family. *Proc. IEEE Int. Symp. on Circuits and Systems, ISCAS'99, Orlando*, Mai 1999.

[25] J. P. Fishburn and A. E. Dunlop. TILOS: A Posynomial Programming Approach to Transistor Sizing. *Proc. ICCAD*, Seiten 326–328, 1985.

[26] J. Götze and G. J. Hekstra. Adaptive Approximate Rotations for Computing the EVD. In M.Moonen and F. Cathoor, editors. *Algorithms and Parallel VLSI Architectures*, 1994.

[27] N. Hedenstierna and K. O. Jeppson. CMOS Circuit Speed and Buffer Optimization. *IEEE Transactions on Computer Aided Design*, CAD-6(2):270–281, März 1987.

[28] B. Hoppe. Mathematical Techniques for Low-Cost optimization of Digital MOS Circuits. *IEE Proceedings*, 137, Pt. G(5):340–344, Oktober 1990.

[29] B. Hoppe, G. Neuendorf, D. Schmitt-Landsiedel, and W. Specks. Optimization of High-Speed CMOS Logic Circuits with Analytical Models for Signal Delay, Chip Area, and Dynamic Power Dissipation. *IEEE Trans. on Computer-Aided Design*, 9(3):236–247, März 1990.

[30] H. S. Hou. A Fast Recursive Algorithm for the Discrete Cosine Transform. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-35(10):1455–1461, Oktober 1987.

[31] S.-T. Huang, T.-M. Parng, and J.-M. Shyu. A New Approach to Solving False Path Problem in Timing Analysis. *Proc. IEEE Int. Conference on Computer Aided Design, ICCAD*, Seiten 216–219, 1991.

[32] S.-T. Huang, T.-M. Parng, and J.-M. Shyu. Timed Boolean Calculus and Its Applications in Timing Analysis. *IEEE Trans. on Computer Aided Design*, 13(3):318–337, März 1994.

[33] The MathWorks Inc. MATLAB® Optimization Toolbox User's Guide. Version 5, Dezember 1996.

[34] A. K. Jain. *Fundamentals of Digital Image Processing.* Prentice Hall, Englewood Cliffs, 1989.

[35] L. Jamieson, D. Gannon, and R. Douglass. *The Characteristics of Parallel Algorithms.* MIT Press, Cambridge, 1987.

[36] M. C. Johnson and K. Roy. Scheduling and Optimal Voltage Selection for Low Power Multi-Voltage DSP Datapaths. *Proc. IEEE Int. Symp. on Circuits and Systems, ISCAS'97, Hong Kong*, Seiten 2152–2155, Juni 1997.

[37] J. Leijten, J. van Meerbergen, and J. Jess. Analysis and Reduction of Glitches in Synchronous Networks. *Proceedings of the ED&TC*, Seiten 398–403, 1995.

[38] C. E. Leiserson, F. M. Rose, and J. B. Saxe. Optimizing Synchronous Circuitry by Retiming. *Proceedeings of the 3rd Calctech Conference on VLSI*, Seiten 87–116, 1983.

[39] C. E. Leiserson and J. B. Saxe. Optimizing Synchronous Systems. *Journal of VLSI and Computer Systems*, Seiten 41–67, 1983.

[40] C. E. Leiserson and J. B. Saxe. A Mixed-Integer Linear Programming Problem Which Is Efficiently Solvable. *Journal of Algorithms*, 9, 1986.

[41] C. E. Leiserson and J. B. Saxe. Retiming Synchronous Circuitry. *Algorithmica*, Seiten 5–35, 1991.

[42] B. Lin and H. De Man. Low-Power driven Technology Mapping Under Timing Constraints. *Proc. International Conference on Computer Design*, Seiten 421–427, 1993.

[43] L. E. Lucke, J. Lee, and B. Vinnakota. Power Estimation Using Input/Output Transition Analysis (IOTA). *Proceedings Int. Symposium on Circuits and Systems*, 6:49–52, Juni 1998.

[44] G. De Micheli. Synchronous Logic Synthesis: Algorithms for Cyce-Time Minimization. *IEEE Transactions on Computer-Aided Design*, 10(1):63–73, Januar 1991.

[45] J. Monteiro, S. Devadas, and A. Ghosh. Retiming Sequential Circuits for Low Power. *Proceedings of the International Conference on CAD*, 1993.

[46] A. Muenzer and G. Hemme. Converting Combinational Circuits into Pipelined Data Paths. *Proceedings of the International Conference on CAD*, Seiten 368–371, 1991.

[47] T. G. Noll. Carry-Save Architectures for High-Speed Digital Signal Processing. *Journal of VLSI Signal Processing*, 3:121–140, 1991.

[48] T. G. Noll. VLSI Design. *Unterlagen zur Vorlesung an der RWTH Aachen*, Seiten 1–150, 1991.

[49] J. A. Nossek and A. Schlaffer. Adiabatic Circuits for Low Power VLSI (Invited Paper). In *43rd International Scientific Colloquium '98 Proceedings.*

[50] J. M. Rabaey. *Digital Integrated Circuits: A Design Perspective.* Prentice Hall, New Jersey, 1996.

[51] J. M. Rabaey and M. Pedram (Ediors). *Low Power Design Methodologies.* Kluwer Academic Publishers, 1996.

[52] A. Raghunathan, N. K. Jha, and S. Dey. *High-Level Power Analysis and Optimization.* Kluwer Academic Publishers, 1998.

[53] P. Rieder, J. Götze, M. Sauer, and J. A. Nossek. Orthogonal Approximation of the Discrete Cosine Transform. *Proc. ECCTD'95*, Seiten 1003–1006, 1995.

[54] P. Rieder, C. V. Schimpfle, and J. A. Nossek. Realization of Multiwavelet–Based Transform Kernels for Image Coding. *Proc. IEEE Int. Symp. on Circuits and Systems, ISCAS'98, Monterey, California, USA*, V:538–541, 1998.

[55] P. Rieder, S. Simon, and C. V. Schimpfle. Application Specific Efficient VLSI Architectures for Orthogonal Single- and Multiwavelet Transforms. *Journal of VLSI Signal Processing Systems*, 21:77–90, 1999.

[56] J. Rubinstein, P. Penfield Jr., and M. A. Horowitz. Signal Delay in RC Tree Networks. *IEEE Transactions on Computer-Aided Design*, 2(3):202–211, Juli 1983.

[57] T. Sakuta, W. Lee, and P. T. Balsara. Delay Balanced Multipliers for Low Power/Low Voltage DSP Core. *IEEE Symp. on Low Power Electronics*, Seiten 36–37, Oktober 1995.

[58] J. H. Satyanarayana and K. K. Parhi. HEAT: Hierarchical energy analysis tool. *33rd ACM/IEE Design Automation Conference*, Seiten 9–14, Juni 1996.

[59] C. Schimpfle. Implementierung digitaler Filter mit parametrisiertem Datenformat. *Diplomarbeit, Technische Universität München*, Mai 1995.

[60] C.V. Schimpfle, P. Rieder, and J.A. Nossek. A Power Efficient Implementation of the Discrete Cosine Transform. *Proc. 31st Asilomar Conference on Signals, Systems and Computers. Pacific Grove*, Seiten 729–733, November 1997.

[61] C.V. Schimpfle and S. Simon. Estimation and Reduction of Glitches in Data Paths. *Technical Report No.: TUM-LNS-TR-97-4, Munich University of Technology*, Februar 1997.

[62] C.V. Schimpfle and S. Simon. Estimation and Reduction of Spurious Switching Activities in Static CMOS Circuits. *International Journal of Electronics and Communications, AEÜ*, 51(6):296–303, November 1997.

[63] C.V. Schimpfle and S. Simon. Signed Digit CORDIC Implementation for Low Power Applications. *Technical Report No.: TUM-LNS-TR-97-2, Munich University of Technology*, Februar 1997.

[64] C.V. Schimpfle, S. Simon, and J. A. Nossek. Device Level Based Cell Modeling for Fast Power Estimation. *Proc. IEEE Int. Symp. on Circuits and Systems, ISCAS'99, Orlando*, Mai 1999.

[65] C.V. Schimpfle, S. Simon, and J. A. Nossek. High-Level Circuit Modeling for Power Estimation. *Proc.* $6^{th}$ *Int. Conference on Electronics, Circuits and Systems, ICECS'99, Pafos, Cyprus*, September 1999.

[66] C.V. Schimpfle, S. Simon, and J.A. Nossek. Low Power CORDIC Implementation Using Redundant Number Representation. *Proc. IEEE Int. Conf. on Application Specific Systems, Architectures and Processors, ASAP'97, Zürich*, Seiten 154–161, Juli 1997.

[67] C.V. Schimpfle, S. Simon, and J.A. Nossek. Optimizing Sequential Circuits for Minimum Glitching Activity. *Proc. European Conf. on Circuit Theory and Design, ECCTD'97, Budapest*, Seiten 1332–1336, 1997.

[68] C.V. Schimpfle, A. Wróblewski, and J. A. Nossek. Transistor Optimization for Minimizing Switching Power in CMOS Circuits. *Technical Report No.: TUM-LNS-TR-99-4, Munich University of Technology*, Juli 1999.

[69] C.V. Schimpfle, A. Wróblewski, and J. A. Nossek. Transistor Sizing for Switching Activity Reduction in Digital Circuits. *Proc. European Conf. on Circuit Theory and Design, ECCTD'99, Stresa, Italy*, August 1999.

[70] A. Schlaffer. An adiabatic Full-Adder using Pass-Transistors and Precharging. Technical Report TUM-LNS-TR-96-6, Munich University of Technology, 1996.

[71] A. Schlaffer and J. A. Nossek. Enhanced Prediction of Energy Losses during adiabatic Charging. In *ISLPED '97 Proceedings*, 1997.

[72] A. E. Schlegel and T. G. Noll. Switching Activity Optimization in Digital CMOS Circuits. *Proceedings of the Int. Symposium on Low Power Electronics and Design*, 1996.

[73] H. Schmeck. *Analyse von VLSI-Algorithmen*. Spektrum Akademischer Verlag, 1995.

[74] R. Secareanu and E. Friedman. A Universal CMOS Voltage Interface Circuit. *Proc. IEEE Int. Symp. on Circuits and Systems, ISCAS'99, Orlando*, Mai 1999.

[75] A. M. Shams and M. A. Bayoumi. A Novel Low-Power Building Block CMOS Cell For Adders. *Proc. IEEE Int. Symp. on Circuits and Systems, ISCAS'98, Monterey*, II:153–156, 1998.

[76] S. Simon. Entwurf von Datenpfaden in schnellen integrierten Schaltungen. *Dissertation, Technische Universität München*, Shaker Verlag, Aachen, 1997.

[77] S. Simon and J. Hofner. The Application of Fast SP-Algorithms to Retiming. *Technical Report TUM-LNS-TR-94-11, Institute for Network Theory and Signal Processing, Munich University of Technology*, 1994.

[78] S. Simon and J. Hofner. Retiming Algorithms for Multiplexer Circuits. *Technical Report TUM-LNS-TR-94-8, Institute for Network Theory and Signal Processing, Munich University of Technology*, 1994.

[79] S. Simon, J. Hofner, and J. A. Nossek. A Fast Retiming Algorithm for Digital Circuits with Acyclic Circuit Graphs. *Proceedings of the European Conference on Circuit Theory and Design, ECCTD'95*, 2:1047–1050, 1995.

[80] S. Simon, P. Rieder, and J. A. Nossek. Efficient VLSI Suited Architectures for the Discrete Wavelet Transform. *IEEE Workshop on VLSI Signal Processing*, 1996.

[81] S. Simon, P. Rieder, C. Schimpfle, and J. A. Nossek. CORDIC-Based Architectures for the Efficient Implementation of Discete Wavelet Transforms. *Proc. IEEE Int. Symp. on Circuits and Systems, ISCAS96, Atlanta, GA, USA*, IV:77–80, 1996.

[82] S. Simon, C. V. Schimpfle, M. Wróblewski, and J. A. Nossek. Retiming of Latches for Power Reduction of DSP Designs. *Proc. IEEE Int. Symp. on Circuits and Systems, ISCAS'97, Hong Kong*, Seiten 2168–2171, Juni 1997.

[83] S. K. Tang, S. C. Chan, K. L. Ho, and F. K. Lam. Implementation of the Fast Cosine Transform on the Motorola DSP 96002 Digital Signal Processor. *IEEE Int. Symposium on Circuits and Systems*, Seiten 73–76, April 1991.

[84] S. Trimberger. Automated Performance Optimization of Custom Integrated Circuits. *Proc. Int. Symp. on Circuits and Systems*, Seiten 194–197, 1983.

[85] H. J. M. Veendrick. Short-Circuit Dissipation of Static CMOS Circuitry and Its Impact on the Design of Buffer Circuits. *IEEE Journal of Solid State Circuits*, SC-19(4):194–197, August 1984.

[86] J.-M. Wang, S.-C. Fang, and W.-S. Feng. New Efficient Designs for XOR and XNOR Functions on the Transistor Level. *IEEE Journal of Solid-State Circuits*, 29(7):780–786, Juli 1994.

[87] L. Wei, Z. Chen, K. Roy, M. C. Johnson, Y. Ye, and V. K. De. Design and Optimization of Dual-Threshold Circuits for Low-Power Applications. *IEEE Transactions on Very Large Scale Integration Systems*, 7(1):16–24, März 1999.

[88] A. Wróblewski. Optimierung von Transistordimensionen zur Reduktion der Schalthäufigkeit in digitalen CMOS-Schaltungen. *Diplomarbeit, Technische Universität München*, April 1999.

[89] M. Wróblewski, S. Simon, and J. A. Nossek. Low Power Transformation of Datapath Architectures with Cyclic SFGs. *Proc. IEEE Int. Symp. on Circuits and Systems, ISCAS 2000, Geneva*, Mai 2000.

[90] G. K. Yeap. *Practical Low Power Digital VLSI Design*. Kluwer Academic Publishers, 1998.

# Anhang

# A. Symbole und binäre Operatoren

## A1 Liste häufig verwendeter Symbole

| | |
|---|---|
| $\alpha$ | Schaltaktivität |
| $a$ | Glitch-Aktivität |
| $\beta$, $\beta_{\mathrm{n}}$, $\beta_{\mathrm{p}}$ | Steilheitsfaktor, bzw. Steilheitsfaktor von NMOS- und PMOS-Transistor |
| $C_{\mathrm{E}}$ | Eingangskapazität |
| $C_{\mathrm{L}}$ | Lastkapazität |
| $C_{\mathrm{W}}$ | Leitungskapazität |
| $c$ | Wunschgröße für die minimale Taktperiode |
| $c_{\mathrm{d/s}}$ | Drain-Source-Kapazität |
| $c_{\mathrm{G}}$ | Wunschgröße für die minimale Glitch-Aktivität |
| $c_{\mathrm{g}}$ | Gate-Kapazität |
| $D(u, v)$ | Verzögerungszeit zwischen den Knoten $u$ und $v$ |
| $d(v)$ | Verzögerungszeit des Knotens $v$ |
| $\Delta(v)$ | maximale Pfadverzögerungszeit zum Knoten $v$ |
| $\Delta_{\mathrm{G}}(v)$ | Summe der Glitch-Gewichte zum Knoten $v$ |
| $e$ | Kante eines Schaltungsgraphen |
| $f$ | (Takt)frequenz |
| $g$ | Glitch-Gewicht |
| $G =< V, E, d, w >$ | Schaltungsgraph mit der Knotenmenge $V$, der Kantenmenge $E$, den Knotenverzögerungszeiten $d$ und den Kantengewichten (Registerzahlen) $w$ |
| $i_{\mathrm{q}}$ | Querstrom bei gleichzeitig leitenden NMOS- und PMOS-Transistoren von der Quelle zur Masse |
| $i_{\mathrm{C}}$ | Ladestrom eines Kondensators |
| $I_{\mathrm{S}}$ | Sättigungssperrsrom |
| $I_{\mathrm{sperr}}$ | Sperrstrom |
| $I_{\mathrm{sub}}$ | Leckstrom (sub-threshold current) |
| $L$ | Kanallänge eines MOS-Transistors |
| $M_{\mathrm{T}}$ | Menge aller möglichen Zustandsübergänge am Eingang eines Gatters |
| $P_{\mathrm{Glitch}}$ | kapazitive Schaltleistung aufgrund unnötiger Schaltvorgänge |
| $P_{\mathrm{L}}$ | Leistungsanteil, der durch Belastung eines Gatters mit einer normierten Last zusätzlich zur Leistung bei Leerlauf verbraucht wird |
| $P_{\mathrm{s}}$ | kapazitive Schaltleistung |
| $P_{\mathrm{k}}$ | dynamische Kurzschlussleistung |
| $\Psi(G)$ | Maximale Pfadverzögerung in einem Graphen $G$ |
| $\Psi_{\mathrm{G}}(G)$ | Maximales Pfad-Glitch-Gewicht in einem Graphen $G$ |
| $r(v)$ | Knotenindex, gibt an, wieviele Register nach Retiming über den Knoten $v$ geschoben werden müssen |
| $\rho$ | Trägheitsfaktor zur Modellierung des Verschwindens kurzer Impulse in einer Kette von Logikgattern |

| | |
|---|---|
| $s$ | Verhältnis der Kanalweiten von N- zu PMOS-Transistor |
| $T$ | Zeitspanne bestimmter Länge |
| $t$ | Zeitvariable |
| $\tau_{\mathrm{L}}$ | Verzögerungszeit, die durch Belastung eines Gatters mit einer normierten Last zusätzlich zur Verzögerungszeit bei Leerlauf auftritt |
| $\tau_{\mathrm{eff}}$ | lineare Approximation der Anstiegs- bzw. Abfallzeit einer Signalflanke |
| $\tau_m$ | Verzögerungszeit des Gatters Nummer $m$ |
| $\tau_{\mathrm{r/f}}$ | Anstiegs- bzw. Abfallzeit einer Signalflanke |
| $\tau_{\mathrm{s},m}$ | Verzögerungszeit des Gatters Nummer $m$ bei sprunghafter Erregung |
| $\tau_{\mathrm{in},m}$ | Verzögerungszeit des Gatters Nummer $m$ aufgrund endlicher Flankensteilheit des Eingangssignals |
| $U_{\mathrm{B}}$ | Versorgungs- oder Batteriespannung |
| $U_{\mathrm{T}}$ | Temperaturspannung |
| $U_{\mathrm{th}}$ | Schwellenspannung |
| $U_{\mathrm{ds}}$ | Drain-Source-Spannung |
| $U_{\mathrm{gs}}$ | Gate-Source-Spannung |
| $u_{\mathrm{e}}$ | Eingangsspannung |
| $u_{\mathrm{C}}$ | Spannung an einem Kondensator |
| $v$ | Knoten eines Schaltungsgraphen |
| $W$ | Kanalweite eines MOS-Transistors |
| $W_{\mathrm{C}}$ | in einem Kondensator gespeicherte Energie |
| $W_{\mathrm{q}}$ | sus einer Quelle entnommene Energie |
| $W(u,v)$ | Summe aller Kantengewichte (Register) zwischen den Knoten $u$ und $v$ |
| $w$ | Wichtungsfaktor zur Parametrisierung eines Vektoroptimierungsproblems |
| $w(e)$ | Anzahl der Register in einer Kante $e$ |

## A2 Liste der verwendeten binären Operatoren

| | |
|---|---|
| $\cdot$ | Und-Verknüpfung |
| $+$ | Oder-Verknüpfung |
| $\oplus$ | exklusive Oder-Verknüpfung, Antivalenz |

# B. Abkürzungen

| | |
|---|---|
| **GC** | **G**litch **C**ounting |
| DCM | *D*evice Level Based *C*ell *M*odelling |
| **DGA** | **D**irekte **G**litch-**A**nalyse |
| **FEAS** | **FEAS**ible Clock period; klassischer Retiming-Algorithmus |
| **LPWR** | **L**ow **P**o**W**er **R**etiming |
| **OPTPWR** | **OPT**imize **P**o**W**er by **R**etiming |
| **OpTRAN** | **Op**timiere **TRAN**sistordimensionen |
| SDNR | Signed Digit Number Representation |
| **SMART** | **S**parse **MA**trix **R**e**T**iming |
| VHDL | Very High Speed Integrated Circuit Hardware Description Language |