Universität Hamburg
Fachbereich Informatik
Arbeitsbereich TECH
Vogt-Kölln-Straße 1
20200 Hamburg

Entwurf von Hochgeschwindigkeitsaddierern in dynamischer Schaltungstechnik

Dezember 1999

Arbeit von

cand. inf. Markus Böttger
Kempelbarg 16
22549 Hamburg

Betreut durch

Dipl. inf. Andreas Mäder

Abstract

This paper describes a dynamic circuit technique, using only a single-phase clock which is never inverted. Those circuits have the advantage of reduced clock lines and high speed. The development and expanding possibilities of two adder-circuits are described and discussed.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Einführung

Für die Geschwindigkeit integrierter Schaltungen sind viele verschiedene Faktoren maßgebend. Zu Beachten sind Bauteilgrößen, Schaltungsaufbau, Takt-Strategie, Takt-Verteilung usw. Der meist beachtete Faktor in der Suche nach der möglichst schnellen Architektur ist die Größe der MOS-Transistoren. „Die Geschwindigkeit einer CMOS-Schaltung ist umgekehrt proportional zu ihrem Maßstab, wenn alle ihre Dimensionen verkleinert werden ohne die physikalischen Eigenschaften zu verändern.“[1] Das resultiert aus der Tatsache, daß ein Transistor am schnellsten seine Ladung umkehren kann, wenn die Kanalweite der Ausgangslast angepaßt wird. Aber gerade was die Miniaturisierung der Bauteile anbetrifft, stoßen wir schnell an physikalische, geometrische und ökonomische Grenzen.

Durch die zunehmende Leistungsfähigkeit der integrierten Schaltungen werden Softwareaufgaben mehr und mehr der Hardware übertragen. Ein gutes Beispiel dafür ist die Entwicklung der Intel Pentium Prozessoren. Während bei der 486er und frühen Pentium-Baureihe die Beschleunigungsfunktionen der Grafik, Sound und Massenspeichersteuerung noch vollständig den jeweils vorhandenen Steckkarten und deren Softwaretreibern überlassen wurde, ging man bald anderer Wege. Der Intel Pentium MMX integrierte zusätzlich **M**ulti**m**edia-E**x**tentions, die einen großen Teil der o.g. Beschleunigung durch Hardware unterstützte. Diese Möglichkeiten forcieren natürlich die Entwicklung der Mikroelektronik insofern, daß schnellere Hardwareschaltungen kommerziell zunehmend interessanter werden. Je schneller Schaltungen realisierbar sind, umso größer ist der Druck, Softwareaufgaben durch Hardware ausführen zu lassen - Hardware ist schlichtweg schneller.

Die Mikroelektronik und ihre Leistung, Schaltungen zunehmend miniaturisieren zu können, ist ein wichtiger Entwicklungsschritt. Allerdings ist wie o.a. der Faktor der Abhängigkeit von der Ausgangslast der Schaltung nicht außer acht zu lassen. Durch intelligente Betrachtung der anzunehmenden Ladungsumladungen kann man die Schaltungen optimieren. Aus diesem Grund sollte man höheren Wert darauf legen, daß man geschickte Schaltungstechnik in bekannte Technologie einbezieht. So kann man zeigen, daß nicht die Architektur der Bauteile, sondern eher intelligente Schaltungstechnik das Maximum der Arbeitsfrequenz bestimmt.

Die angesprochene „intelligente“ Betrachtung der Ladungsumladung jedes Transistors einer komplexen Schaltung und die sich daraus entwickelnde zusätzliche Geschwindigkeit ist die Motivation für diese Arbeit.

[1] Yuan, Svensson [1]

1 Takt-Strategien

1.1 C^{2}MOS

Üblicherweise benutzt man in CMOS-Schaltungen statische oder dynamische CMOS-Logik und schaltet mehrere Logikstufen zu einer sog. Pipeline hintereinander. Wenn ein Teil dieser Pipeline von einem von außen kommenden Datensignal abhängt, welches vielleicht auch noch von einer anderen Pipeline stammt, so muß man spätestens hier für eine Synchronisation des gesamten Systems sorgen. Die meist verwendete Technik in diesem Fall ist die C^{2}MOS-Logik (Clocked CMOS logic), die einen nichtüberlappenden zweiphasen-Takt (nonoverlapping pseudo-two-phase-clock, NPTC) benutzt.

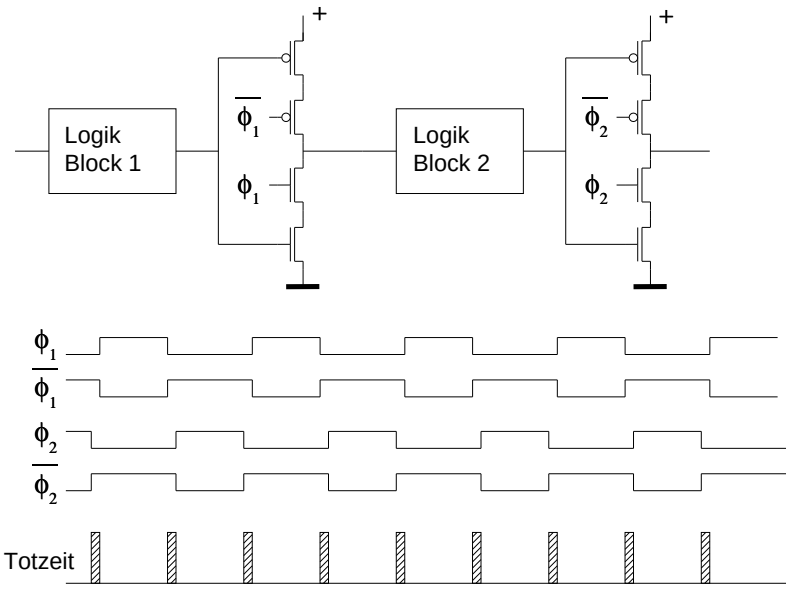


Abbildung 1: C^{2}MOS Logik

Die Verwendung dieser Taktstrategie führt dazu, daß man die Taktsignale normal und invertiert benötigt. Somit sind in einem solchen System vier verschiedene Taktsignale zu erzeugen und zu übertragen. Es ist leicht einzusehen, daß bei derartig kompliziertem Aufbau eine Signalverzögerung in einer der Taktleitungen zu Schaltfehlern führen kann. Deshalb muß man besonders darauf achten, daß die vier Signale zeitlich passend ihre Zustände einnehmen. Um die Flanken nicht zur Überschneidung zu bringen, fügt man eine sog. „Totzeit“ („dead time“) zwischen den Taktwechseln ein. Somit kommt es zwischen den Taktsignalen zwar zu keiner fehlerträchtigen Überlappung, es verlangsamt die Erzeugung des Taktsignals allerdings erheblich. Ein weiterer negativer Aspekt ist der hohe Verbrauch von kostbarer Chipfläche, den vier verschiedene Taktleitungen benötigen.

1.2 NORA

Die NORA (No RAce) Strategie bei dynamischen CMOS-Schaltungen benötigt ein zweiphasen-Taktsignal, welches normal (ϕ) und invertiert ($\overline{\phi}$) genutzt wird. Somit braucht man nur

zwei verschiedene Taktsignale und es verringert sich die Anfälligkeit und Komplexität bzgl. Signalverzögerungen gegenüber der C^2MOS-Strategie. Man kann bei der Generierung des Taktes auch auf Totzeiten verzichten, so daß mit der dynamischen NORA-C^2MOS-Technik schnellere Taktraten realisierbar sind als mit C^2MOS.

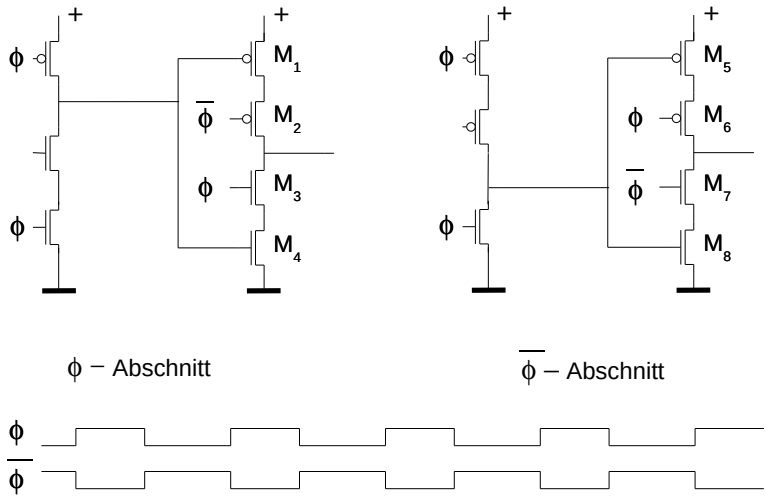


Abbildung 2: dynamische NORA-Technik

Beim Aufbau einer NORA-basierten Pipeline muß zwingend beachtet werden, daß zwischen zwei C^2MOS-Latches eine gerade Anzahl von Inversionsblöcken vorhanden ist. Zwei typische NORA-Konstrukte, die man als ϕ und $\overline{\phi}$-Abschnitte bezeichnet, enthalten einen N-precharge-Block im ϕ und einen P-precharge-Block im $\overline{\phi}$-Abschnitt. In Abbildung 2 ist zwischen den taktgesteuerten Transistoren beispielhaft ein P-, bzw. N-Transistor eingezeichnet. Diese stehen stellvertretend für einen beliebigen N-, bzw. P-Logik-Block.

1.3 TSPC

Eine moderne Clock-Strategie sollte mit möglichst einfachen und wenig verschiedenen Clock-Signalen auskommen. Die o.a. Argumente Chipfläche und Komplexität sprechen dafür.

Die dynamische True-Single-Phase-Clock CMOS-Strategie benötigt nur ein Taktsignal, daß auch nur normal und nicht invertiert gebraucht wird. Das hat den Vorteil, daß alle Vorkehrungen unnötig sind, eine Flankenüberlappung zu verhindern. Somit können maximale Taktfrequenzen möglich gemacht werden. Die TSPC-CMOS-Technik kann nicht nur für dynamische, sondern auch für statische Schaltungen benutzt werden und kann in den meisten Fällen die NORA Technik vollständig ersetzen.

1.4 CDPD

Eine leistungsfähige Technik zum Aufbau von mehrschichtiger Logik ist die „clock and data precharged dynamic CMOS technique“ (CDPD). CDPD ist eine Weiterentwicklung der „Domino-Technik“, die als langsam bzgl. ihrer maximal möglichen Taktrate gilt, und TSPC.

Die CDPD-Technik profitiert von Geschwindigkeitsgewinnen, die erzielt werden, wenn man bestimmte Bereiche der Schaltung „high“ oder „low“ vorlädt. Bei CDPD benutzt man zum Vorladen sowohl die Takt-, als auch die Dateneingangssignale.

2 Die TSPC-Technik

2.1 Basiselemente

Warum kann man nun auf das invertierte Taktsignal, daß in der NORA-Technik noch gebraucht wird, verzichten. Dazu betrachten wir die beiden Latches in Abb. 2, die von den Taktsignalen ϕ und $\overline{\phi}$ gesteuert werden. Die Notwendigkeit des Signals $\overline{\phi}$ liegt darin, daß es die Transistoren M_2 und M_7 steuert. Dieses kann man dadurch ersetzen, daß man das ϕ - Signal mit einem vorgeschalteten Inverter benutzt (s. Abb. 3). Betrachtet man die doppelte N-C^{2}MOS-Stufe beispielsweise, so erkennt man, daß der mit „x“-markierte P-Transistor durch den vorgeschalteten Inverter die Funktion des M_2-Transistors aus Abb. 2 mitübernimmt. Für ϕ=0 sperren in Abb. 3 die taktgesteuerten N-Transistoren; der „x“-markierte P-Transistor sperrt ebenfalls, da ein low-Signal durch den vorgeschalteten Inverter an seinem Gate niemals anliegen kann. Für ϕ=1 würde der M_2-Transistor in Abb. 2 sowieso leiten, so daß er abkommlich ist und somit die Schaltung in Abb. 3 als „Ersatzschaltung“ resultiert.

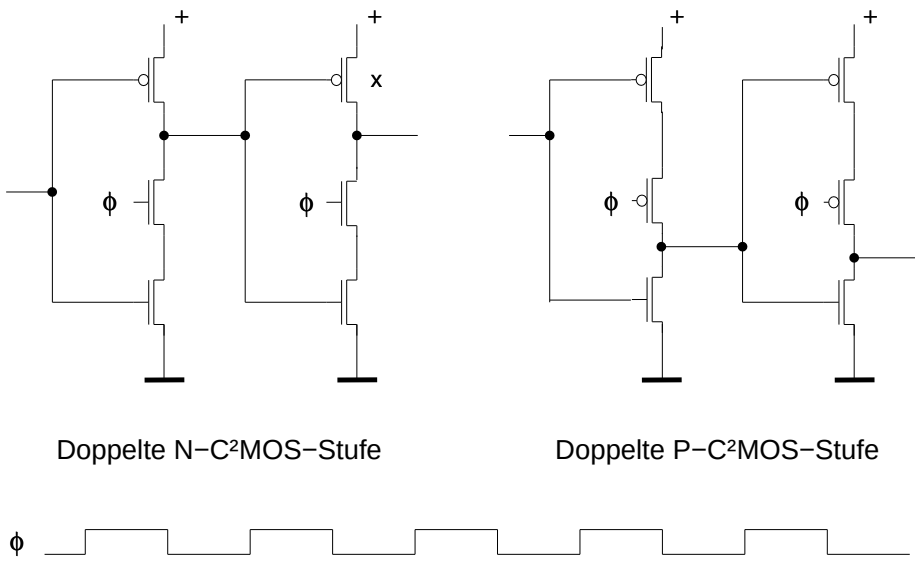


Abbildung 3: TSPC-Latch-Stufen

Durch diese Umformung entstehen symmetrische Schaltungen, von denen man die Hälfte des ϕ-Teils als N-C^{2}MOS-Stufe oder NC-Stufe, die Hälfte des $\overline{\phi}$-Teils als P-C^{2}MOS-Stufe oder PC-Stufe bezeichnet. Kommen zwei gleiche Halbstufen wiederum hintereinander vor, so ergeben sie ein N-Latch oder ein P-Latch. Stellt man die taktgesteuerten Transistoren nicht in die Mitte, sondern an das obere und untere Ende, so entstehen vorgeladene P-, bzw. N-Blöcke (precharged Blocks). Diese werden dann NC2- und PC2-Stufen genannt. Eine Übersicht der Grundbausteine von TSPC gibt die Abb. 4.

Beim korrekten Schaltungsaufbau folgt abwechselnd auf einen P-Abschnitt (P-Latch inkl. Logik-Block) ein N-Abschnitt (N-Latch inkl. Logik-Block). Als Logik-Blöcke sind

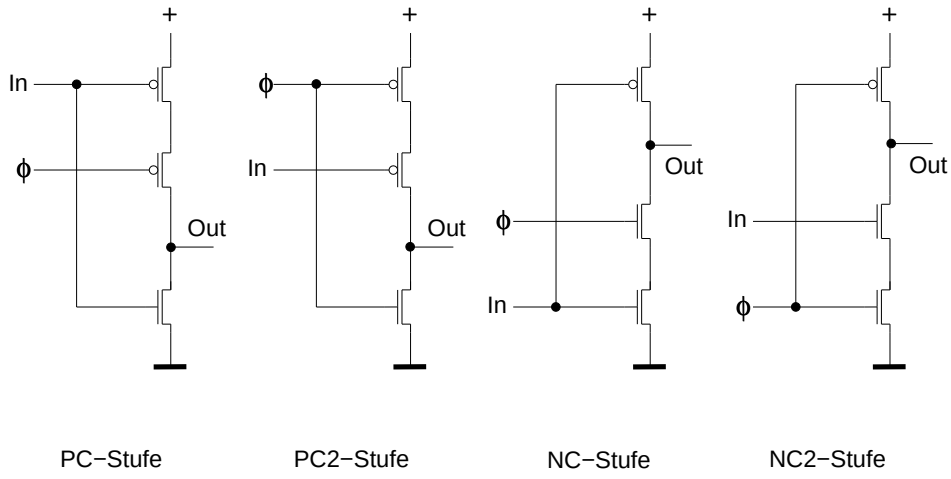


Abbildung 4: TSPC Grundstufen

statische sowie dynamische Varianten einsetzbar. „Solange die Taktverzögerung größer ist als die Schaltzeit der Gates, funktioniert dieses System.“[2] Die oben besprochenen Vorteile dieser Technik zeigen sich nun: man spart ein zusätzliches $\overline{\phi}$-Taktsignal, allerdings ist dafür jede Latch-Stufe um zwei Transistoren größer.

2.2 Funktionselemente und Pipelining

Aus den vier o.g. Grundstufen lassen sich viele verschiedene Funktionselemente einfach zusammensetzen: Natürlicherweise erreicht man die höchste Geschwindigkeit in einer

Stufen	Funktion
PC - PC	nichtvorgeladenes P-Latch
NC -NC	nichtvorgeladenes N-Latch
PC2 - PC	vorgeladenes P-Latch
NC2 - NC	vorgeladenes N-Latch
PC-PC-NC-NC	nichtvorgeladenes, vorderflankengest. Flip-Flop
NC-NC-PC-PC	nichtvorgeladenes, rückflankengest. Flip-Flop
PC2-PC-NC2-NC	vorgeladenes, vorderflankengest. Flip-Flop
NC2-NC-PC2-PC	vorgeladenes, rückflankengest. Flip-Flop
PC - NC2 - NC	minimiertes, vorgel., vorderflankengest. Flip-Flop
NC - PC2 - PC	minimiertes, vorgel., rückflankengest. Flip-Flop

Tabelle 1: TSPC-Funktionselemente

Schaltung mit möglichst geringen Stufen. Um diese dann miteinander zu einer TSPC-Pipeline zusammenzusetzen, gibt es zwei Möglichkeiten: eine Pipeline mit komplementärer- und eine Pipeline mit vorgeladener Logik (s. Abb. 5 und 6).

Die Logikblöcke sollten optimalerweise nur wenige Eingänge besitzen. Ideal wäre eine vorgeladene Logik mit NAND-Operation im p-Block und NOR-Operation im n-Block.

[2] Yuan, Svensson [1]

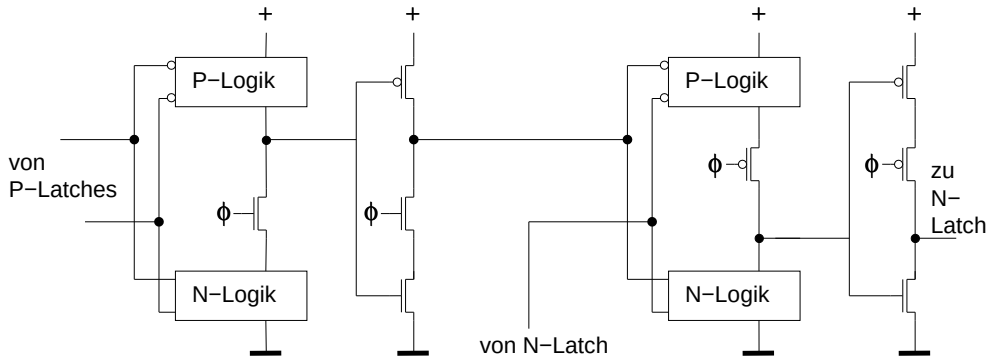


Abbildung 5: TSPC n-p Pipeline mit komplementärer Logik

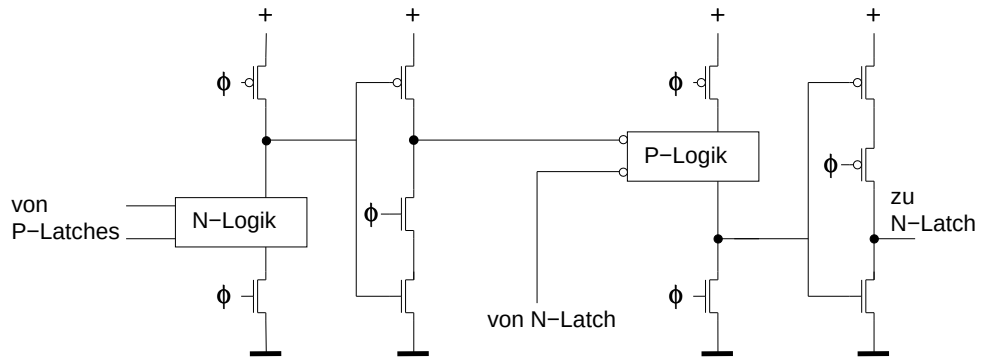


Abbildung 6: TSPC n-p Pipeline mit vorgeladener Logik

Diese Dekomposition der Logik ist nicht immer möglich, erhöht aber die Geschwindigkeit.

2.3 Taktgeschwindigkeit

Vergleicht man die drei verschiedenen Taktstrategien, so wird intuitiv klar, daß TSPC mit seinem einfachen Taktaufbau höhere Taktraten erlaubt. Bei genauerer Betrachtung (und Simulation) hat die höhere Taktfähigkeit mehrere Gründe:

Weil die Anzahl der Transistoren von vier bei NORA auf drei in einer N-C^{2}MOS oder P-C^{2}MOS-Stufe reduziert werden kann, da statt zwei nur noch ein taktgesteuerter Transistor benötigt wird, ist die Verzögerung dieser Stufen geringer. Besonders das Fehlen des P-Transistors in der NC-Stufe macht sich positiv bemerkbar, da die steigende Taktflanke die Zeitkritische für diese Stufe ist. Geeignete Simulation einer Schaltung im 3-μm zweilagen-Metall CMOS Prozeß mit SPICE brachte einen Geschwindigkeitsgewinn um den Faktor 1.8.[3] Außerdem erhält man durch das Fehlen des P-Transistors den Effekt des Vorladens. Dieser läßt die Stufen in der Initialisierungsphase aufladen und sorgt zur Evaluierungsphase für schnelleres Schalten.

In der Schaltkreistechnik hat sich vor allem die statische CMOS-Technik wegen ihrer geringen Anfälligkeit gegenüber Taktunregelmäßigkeiten durchgesetzt. Sie gilt als robuster als die dynamische Technik, so daß man dies auch bei TSPC untersuchen sollte. Ein Ansatz zur Untersuchung des Anfälligkeitsgrades ist die Setup-Time (S) und Hold-Time (H) in TSPC-Schaltungen, die in Abb. 7 skizziert sind. Die Setup- und Hold-Time bilden einen zeitlichen Bereich vor und nach dem Taktwechsel, in dem das Eingangssignal nicht wechseln darf. Gehen wir z.B. davon aus, daß der Eingang auf „low“ steht, dann wird der Ausgang „high“ einnehmen. Dies geschieht aber nicht unmittelbar, sondern erst nach einer gewissen Zeit - nämlich der Zeit, die nötig ist um den oberen P-Transistor vollstandig leitend werden zu lassen. Ist die Setup time nicht eingehalten worden und der Taktwechsel kommt zu früh, dann sperrt der getaktete P-Transistor den Anstieg des Ausgangs und der Ausgang bleibt anstelle von „high“ in einem undefinierten Spannungsbereich liegen.

Jiren Yuan gibt in [1] einen Überblick über die Größenordnung der Setup- und Hold-Time in Abhängigkeit von der Versorgungsspannung und der Zeit, die die Taktflanke bei einem Wechsel benötigt (s. Abb. 8). Dies sind Ergebnisse einer Standard 2μm CMOS Prozeß-Schaltung mittels SPICE.

Neben der Notwendigkeit, Setup- und Hold-Time einhalten zu müssen, ist die Steilheit der Taktflanke ein wichtiger Parameter. Die Gefahr einer zu flachen Taktflanke liegt darin, daß dann zu einem bestimmten Zeitpunkt die P- sowie auch die N-Transistoren gleichzeitig leiten.

Nach Yuan läßt sich aus Abb. 8 schließen, daß eine geringere Versorgungsspannung zu einer Entschärfung der Situation führt. Dies lasse sich damit begründen, daß das „Schalt-

[3] Yuan, Svensson [1]

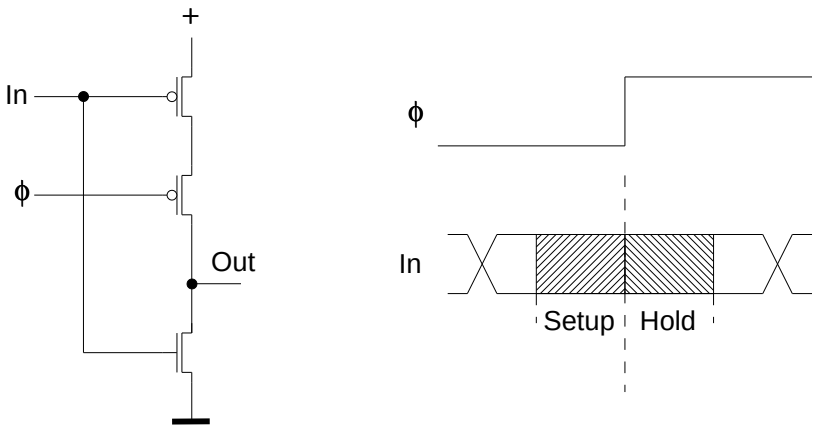


Abbildung 7: Setup-Time und Hold-Time in der TSPC-PC-Stufe

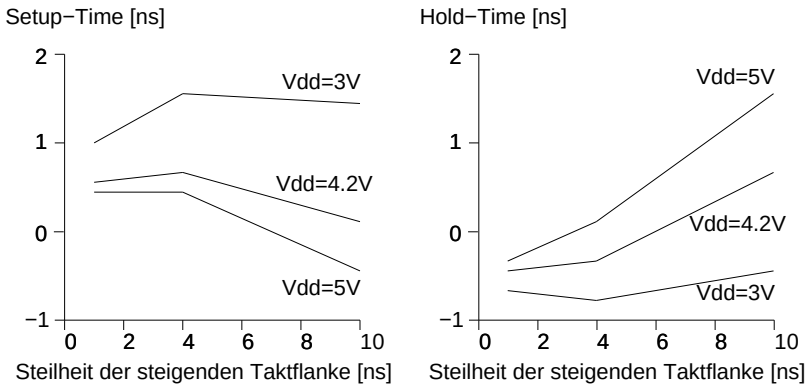


Abbildung 8: Setup-Time und Hold-Time einer PC-Stufe

fenster“ der P- und N-Transistoren proportional zu ihren Versorgungsspannungen kleiner wird. Außerdem spiele die Transistordimensionierung zusätzlich eine gewisse Rolle für deren Schaltgeschwindigkeit. Größer sei aber sicherlich der Einfluß der Prozeßtechnik auf die Schaltgeschwindigkeit einzuschätzen; bei einem kleineren Prozeß, z.B. bei 0.8 μm senkten sich die angegebenen Zeiten erheblich und die Schaltungen würden dadurch deutlich robuster.

Eine weitere Gefahr ist das Auftreten von Taktverzögerungen. Da es in TSPC nur ein Taktsignal gibt, können - im Unterschied zu NPTC und NORA - keine Verzögerungen zwischen einer ersten und zweiten - oder einem invertierten und nicht-invertierten Taktsignal auftreten. Zu einer Verzögerung kann es aber zwischen zwei logischen Blöcken kommen, die an einer Zusammenführung zweier bisher parallel verlaufenden Schaltwege liegen. In einer Schaltung mit einer einfachen „Pipeline-Struktur“ tritt dieses Problem nicht auf, da die Taktverzögerungen kleiner sind als die Schaltverzögerungen der Transistoren. Aber in einem größeren System muß man auf das passende Timing von Takt- und Datensignalen achten.

3 Die CDPD-Technik

3.1 Funktionsweise

Wie in Abschnitt 1.4 angesprochen, ist CDPD eine Weiterentwicklung der Domino- und TSPC-Technik. Eine typische Domino-Logikschaltung zeigt Abbildung 9a, in Abbildung 9b ist sie in CDPD überführt dargestellt. PH und PL bedeuten ein „precharged-to-high“

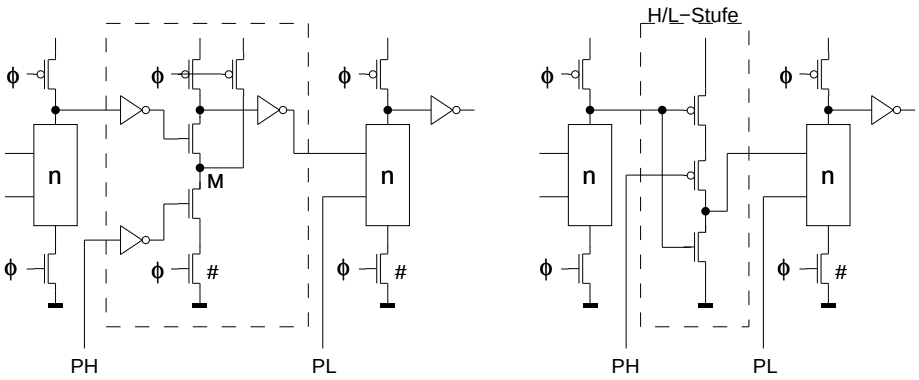
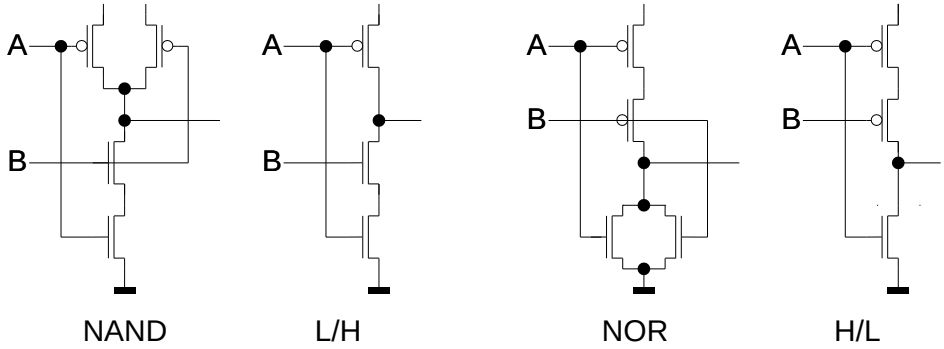


Abbildung 9: Domino- und CDPD-Logik

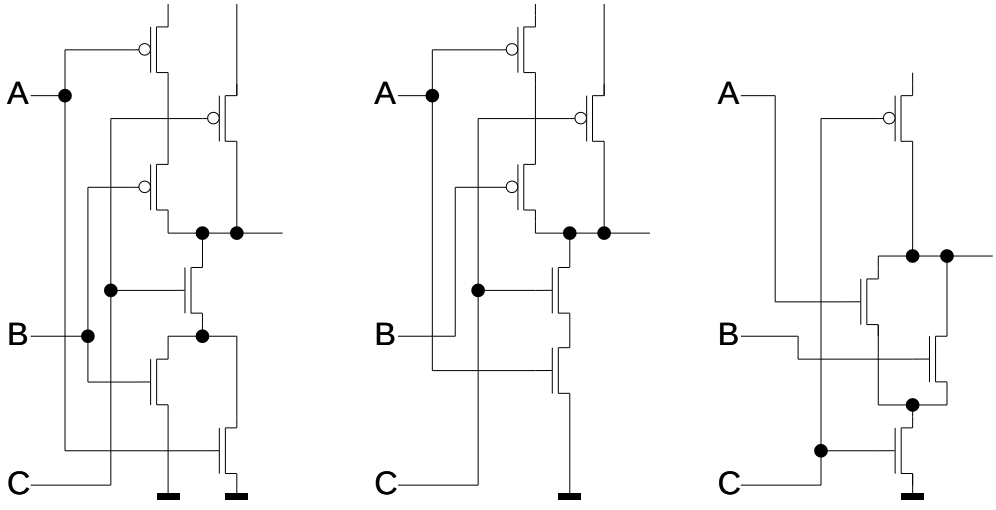
oder „precharged-to-low“ Signal. Bei genauer Betrachtung von Abb. 9a fällt auf, daß man zwischen den taktvorgeladenen Stufen Inverter einsetzen muß. Nach Yuan [2] sorgen diese zwar dafür, daß die Vorladung erhalten bleibt, verlangsamen die Schaltung allerdings. Zusätzlich müssen alle Knoten des Logikblockes im Falle eines taktgesteuerten Vorladens mitgeladen werden. Dies macht im Fall von Abbildung 9a einen zusätzlichen P-Transistor notwendig, um den Knoten M richtig zu laden. Nach Yuan [2] sind zusätzlich die mit # markierten Transistoren überflüssig. Die gesamte Schaltung im gestrichelten Kasten kann durch eine „data precharged H/L-Stufe“ mit nur drei Transistoren ersetzt werden. Eine H/L-Stufe ist eine Stufe, die durch die Daten- und Taktleitungen „high“-vorgeladene Eingänge und „low“-vorgeladene Ausgänge besitzt. Umgekehrt gibt es eine L/H-Stufe, deren Eingänge „low“- und Ausgänge „high“-vorgeladen sind.

Wie Gatter von statischer Bauweise in CDPD-Technik überführt werden können, zeigen die Abbildungen 10 und 11. Die Überführung der einfachen NAND- und NOR-Gatter ist einfach. Es muß nur darauf geachtet werden, daß niemals ein Kurzschluß zwischen Versorgungsspannung und Ground entsteht. In den Abbildungen 10a und b sieht man, daß das Signal A zwei Transistoren so steuert, daß sie nie gleichzeitig leiten können. Bei der Überführung einer beliebigen Schaltung mit mehreren Eingangssignalen (s. Abb. 11) muß man ebenso auf die Vermeidung einer direkten Verbindung zwischen Versorgungsspannung und Ground achten. Beispielsweise wird, wie aus Abb. 11b ersichtlich, für alle parallelen Stromzweige der P-Logik genau ein komplementärer Transistor im N-Logik-Teil angelegt. Diese sind dann seriell zu verschalten.



[a] NAND als L/H − Stufe [b] NOR als H/L − Stufe

Abbildung 10: NAND und NOR in CDPD



[a] statische Schaltung [b] H/L − Stufe [c] L/H − Stufe

Abbildung 11: Eine statische Schaltung als H/L- und L/H-Stufe

3.2 Pipelining

Die Anordnung der oben beschriebenen H/L- und L/H-Stufen in einer Pipeline findet zwischen zwei taktvorgeladenen Stufen statt. Um den Ausgangswert einer solchen Verkettung über den Zeitraum der nachfolgenden Vorladephase zu retten, kann man an das Ende einer solchen Kette eine TSPC-NC-Stufe stellen. Für eine Kette mit H/L-, L/H- und taktvorgeladenen Stufen gilt folgender Verschaltungsgrundsatz: zwischen zwei taktvorgeladenen Stufen muß die Anzahl der H/L- und L/H-Stufen ungerade sein, z.B. {taktvorgeladene Stufe} - {H/L} - {L/H} - {H/L} - {taktvorgeladene Stufe}. Zwischen einer taktvorgeladenen Stufe und einer TSPC-NC-Stufe muß die Anzahl der H/L- und L/H-Stufen gerade sein, z.B. {taktvorgeladene Stufe} - {H/L} - {L/H} - {TSPC-NC-Stufe}. Bei Beachtung dieser Verschaltungregel ist die Anzahl der in einer Kette enthaltenen Stufen unerheblich.

4 Konkrete Schaltungen

4.1 Carry-Save-Addierer

4.1.1 Logik

Das erste Projekt dieser Arbeit beschäftigt sich mit einem Carry-Save-Addierer (CSA). Zunächst sei folgende Indexnotation für Dualzahlen definiert:
Gegeben seien Dualzahlen der Form
$a_m = \{a_{m,n}\ a_{m,n-1}\ a_{m,n-2}\ ...\ a_{m,1}\ a_{m,0}\}$, z.B.
$a_1 = \{a_{1,n}\ a_{1,n-1}\ a_{1,n-2}\ ...\ a_{1,1}\ a_{1,0}\}$ und
$a_2 = \{a_{2,n}\ a_{2,n-1}\ a_{2,n-2}\ ...\ a_{2,1}\ a_{2,0}\}$, so sei
$a_1 + a_2 = \{c_n\ s_n\ s_{n-1}\ s_{n-2}\ ...\ s_1\ s_0\}$.

Man addiert zwei 1-stellige duale Zahlen a_1 und a_2 derart, daß man zuerst das Bit $a_{1,0}$ in einen Volladierer eingibt, deren andere Eingänge zu Beginn auf Low gelegt sind. Der Volladdierer errechnet dann die erste Summe s_0 und den ersten (trivialen) Übertrag c_0. Die Summe s_0 sowie der Übertrag c_0 werden nun für einen Schritt in zwei Speichergliedern gehalten. Im zweiten Schritt legt man die zweite Zahl $a_{2,0}$ an den Volladdierer an, dessen übrige Eingänge von den Speichergliedern mit den Ergebnissen s_0 und c_0 des ersten Schrittes belegt sind. Damit liefert der Volladdierer das Ergebnis der Rechnung $a_1 + a_2 = \{c_0\ s_0\}$ nach zwei Schritten.

Addiert man nun zwei 2-stellige duale Zahlen $a_1 = \{a_{1,1}\ a_{1,0}\}$ und $a_2 = \{a_{2,1}\ a_{2,0}\}$, so benötigt man einen zusätzlichen Volladdierer und zwei weitere Speicherglieder. Die ersten Bits $a_{1,0}$ und $a_{1,1}$ werden zunächst parallel in die Volladdierer VA_0 und VA_1 eingegeben. Deren übrige Eingänge seien wieder zu Beginn auf Low gelegt worden. Es entstehen damit nach dem ersten Schritt s_0 und c_0, sowie s_1 und c_1. Im nächsten Schritt werden die nächsten Bits $a_{2,0}$ und $a_{2,1}$ an die Volladdierer angelegt. Wie im ersten Beispiel sind wieder die anderen Eingänge durch die Werte der Speicherglieder bestimmt. Nach dieser Berechnung ist u.U. an VA_1 ein Übertrag aufgetreten, der noch auf VA_2 übertragen werden muß. Somit benötigen wir noch einen weiteren Schritt, um den Übertrag „durchzurippeln“. Danach ist das Ergebnis berechnet: $\{a_{1,1}\ a_{1,0}\} + \{a_{2,1}\ a_{2,0}\} = \{c_1\ s_1\ s_0\}$.

Man benötigt also zur Addition zweier n-stelliger Dualzahlen n Volladdierer, 2n Speicherglieder und n+1 Zeitschritte. So betrachtet ist das ist kein sehr zeitoptimales Additionsverfahren, denn diese Anzahl Schritte benötigt auch ein normaler Ripple-Carry-Addierer.

Der Vorteil des Carry-Save-Addierers wird aber deutlich, wenn man mehrere Zahlen addiert. Zur Addition von m n-stelligen Dualzahlen benötigt er nämlich m Schritte zur Addition der Stellen und dann noch einmal n-1 Schritte zum „durchrippeln“ des letzten Übertrags von der niedrigsten bis zur höchsten Stelle; d.h. er benötigt dafür (m+n-1) Zeitschritte. Für ein geeignet großes m, also genügend viele zu addierende Zahlen ist dies

allerdings ein sehr günstiges Verfahren.

Den allgemeinen Aufbau eines solchen Addierers zeigt Abb. 12.

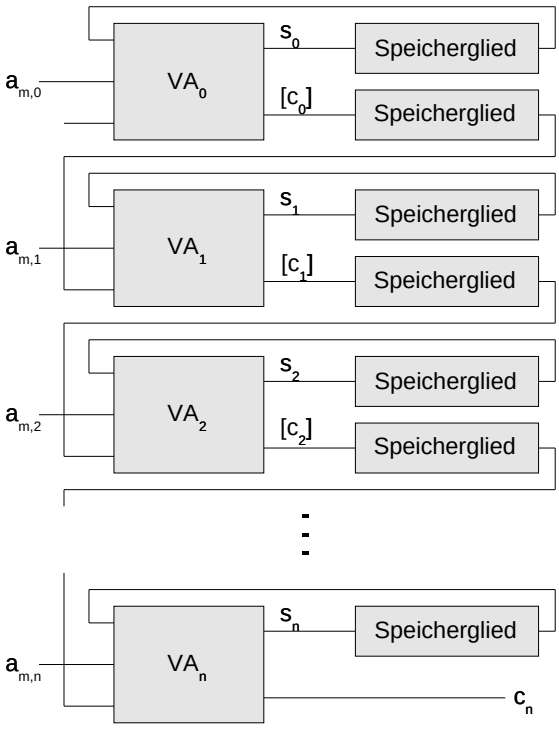


Abbildung 12: Carry-Save-Addierer (schematisch)

4.1.2 Aufbau in statischer Logik

Zum Aufbau eines Carry-Save-Addierers in statischer Technik benötigt man die Volladdierer, sowie je zwei Speicherglieder, die die nach jedem Additionsschritt entstandenen Überläufe (c_n, c_{n-1}, ... , c_1, c_0) zum Errechnen der nächten Bits zwischenspeichert. Der Aufbau eines Volladdierers enthält i.A. zwei verknüpfte Halbaddierer, die aus je einem XOR- und einem AND-Gatter bestehen, plus einem OR-Gatter (s. Abb. 13). Die Speicherglieder können unterschiedlich ausgelegt werden. Da hier ein Vergleich mit einer Hochgeschwindigkeitstechnik stattfindet, nehmen wir ein getaktetes D-Flip-Flop an, welches geringstmöglichen Zeitaufwand bedeutet. Außerdem wird in einer realen Schaltung sowieso eine Taktung bestehen, die das Anliegen der Eingangsbits zeitlich synchronisiert.

Die direkte Umsetzung der Logik liefert für einen Volladdierer ein dreistufiges Schaltnetz wie in Abb. 13 dargestellt.

4.1.3 Aufbau in TSPC-Logik

Wie im Abschnitt 2 deutlich gemacht wurde, bestimmt vor allem die logische Dekomposition der Schaltungen die Geschwindigkeit der einzelnen TSPC-Stufen. Daher ist es sinnvoll, möglichst geschickt die Logik in wenige, einfach abzuarbeitende TSPC-Stufen

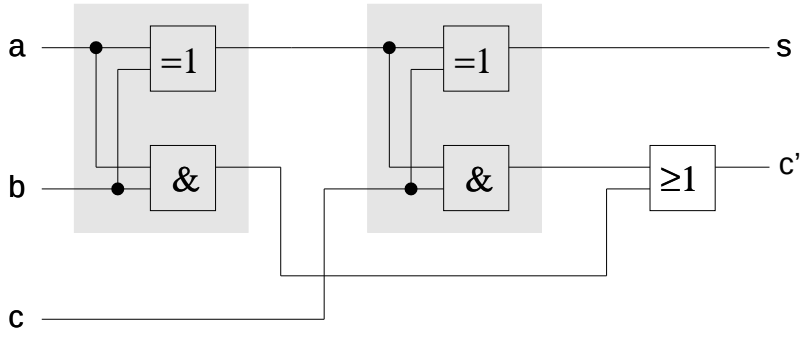


Abbildung 13: Volladdierer als dreistufiges Schaltnetz

einzusetzen. Die Abbildung einer VHDL-Beschreibung auf eine spezielle Bibliothek ist eine hilfreiche Methode. Es wurde mir eine Bibliothek zur Verfügung gestellt, die nur aus einfachen NAND, NOR und INV-Gattern, sowie bestimmten für CMOS-Schaltungen gut realisierbaren Komplexgattern besteht. Die Verwendung eines XOR wurde damit ausgeschlossen. Bildet man nun das Problem des Volladdierers auf diese Bibliothek ab, so erhält man eine Schaltung die für CMOS günstig, und für TSPC gut teilbar ist.

Anstelle der XOR-Gatter verwendet man besser ein NOA22 (ein NOR mit (zwei AND mit je zwei Eingängen)) und ein NAO22 (ein NAND mit (zwei OR mit je zwei Eingängen)), sowie zusätzliche Inverter. Das Ergebnis in Gatterform zeigt Abb. 14. Warum soll

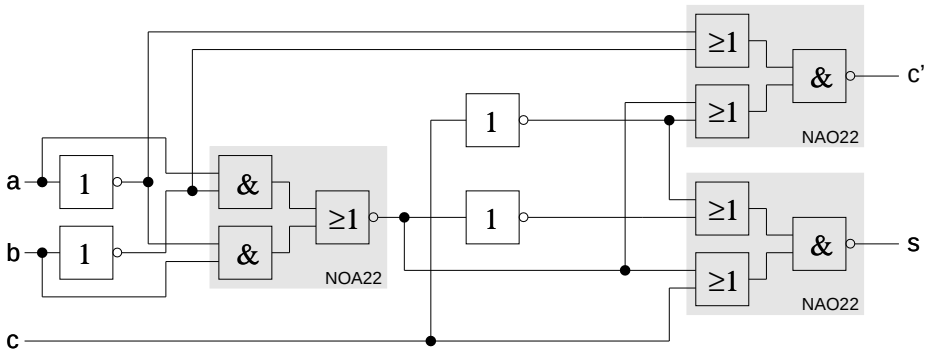


Abbildung 14: Gatterschaltung des optimierten Volladdierers

diese Schaltung „besser“ sein, als die in Abb. 13 gezeigte? Auf den ersten Blick hat diese Lösung mehr Stufen als zwei XORs plus OR, zusätzlich noch vier Inverter. Betrachtet man allerdings die resultierende Transistorschaltung (Abb. 15), so erkennt man, daß die Komplexgatter NAO22 und NOA22 nur eine Stufe bilden. Wie aus Abbildung 12 ersichtlich, folgen auf den Volladdierer noch zwei Speicherglieder. In der statischen Ausführung verwendet man dafür z.B. D-Flip-Flops. In der TSPC-Lösung verwenden wir hier ein vorgeladenes (precharged) N-Latch, daß (nach Tabelle 1) die Form (NC2 NC) hat.

Wichtig für die TSPC-Schaltung ist natürlich der zeitliche Ablauf. Dazu teilt man am Besten die Stufen auf, so daß ein gleichgewichteter Zeitaufwand zwischen der Taktvorderflanke und der Taktrückflanke entsteht. Natürlich gilt das nur, wenn wie hier der

Ablauf für einen einzigen Takt konzipiert werden soll. Ansonsten kann es günstiger sein, den Ablauf in vier Flanken einzuteilen, die vielleicht insgesamt schneller taktfähig sind.

In diesem Fall erscheint sinnvoll, die Übernahme der Eingänge a, b und c kurz vor der ersten Flanke - beispielsweise der Fallenden vorzunehmen und die ersten beiden Inverter, das NOA22-Komplexglied und die beiden zweiten Inverter abzuarbeiten. Deshalb sind diese auch ohne Taktung konzipiert. Nach der fallenden Taktflanke schalten dann die beiden NAO22-Gatter; die NC2-NC-Latches nehmen das Ergebnis dann zur steigenden Flanke auf. Damit entsteht ein möglichst gutes Zeitgleichgewicht zwischen den Taktflanken: mit fallender Flanke erfolgt die Übernahme der Eingangswerte, mit steigender Flanke nehmen die Speicherglieder die Summe s und den neuen Übertrag c' als Ergebnis für die Dauer eines Taktes auf. Das ist rechtzeitig genug, damit der „darunterliegenden Addiererzeile“ (vgl. Abb. 12) für den nächsten Takt der neue Summen- und Übertragswert bereitgestellt werden kann.

Die einzelnen Logikteile werden nun in die TSPC-Pipeline-Logik integriert. Der Teil, der bei Takt-Low aktiv ist, stellt den PC-Teil dar. Dementsprechend ist der Takt-high aktive Teil der NC-Teil. Nach Einsetzen der Logikelemente in die TSPC-Pipeline-Theorie von Abbildung 5 erhält man das gewünschte Transistorlayout für eine komplette Carry-Save-Addierer-Zeile von Abbildung 12 (s. Abb. 15). Zur ordnungsgemäßen Funktion der

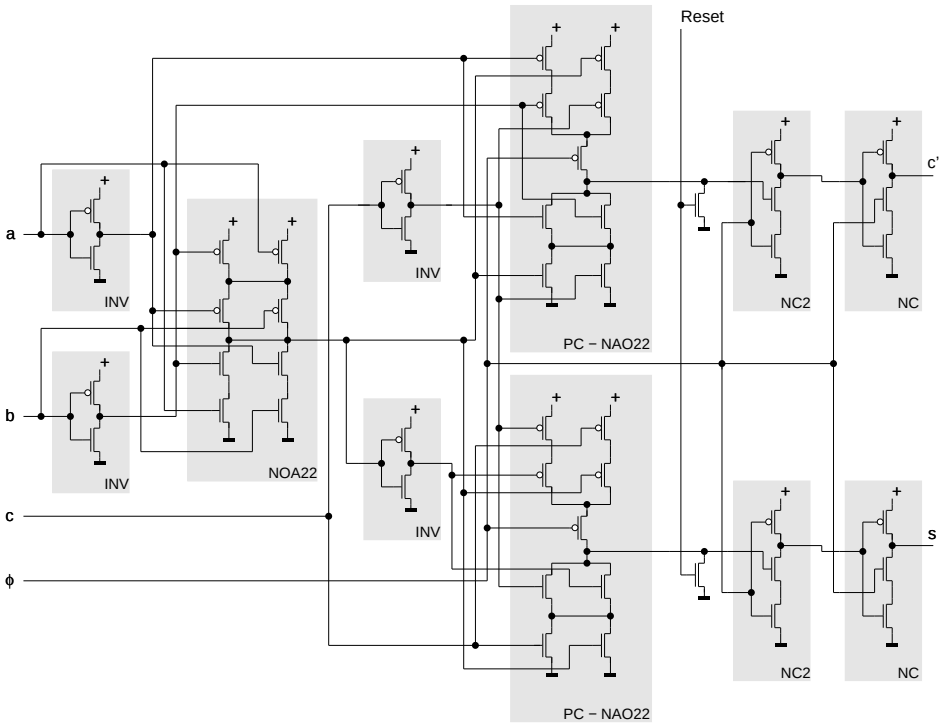


Abbildung 15: Transistorschaltung optimierter Volladdierer plus Latch

Schaltung muß man noch dafür sorgen, daß beim erstmaligen Anliegen der Eingangswerte a_i, bzw. b_i die nachfolgenden Spalten keine fehlerhaften Summen und Überträge bekommen. Dies kann nämlich dadurch passieren, daß bei Starten der Schaltung noch

undefinierte Werte in den Latches vorhanden sind. Deshalb legt man am Besten über ein Reset-Signal einmalig die Eingänge der Latches über einen N-Transistor auf Masse. Das sorgt dann dafür, daß die Latches keine ungewollten „high“ Werte mehr „gespeichert“ haben.

4.1.4 Leistungsbewertung

Die Schaltung ist mit CADENCE konstruiert worden, wobei ein 0.6μm zweilagen-Metall-CMOS-Prozeß benutzt wurde. Bestimmend für die Einschätzung, wie schnell der angelegte Takt maximal sein darf, ist die Beobachtung der Schaltzeiten der einzelnen TSPC-Stufen. Wie und wann diese nun schalten, erkennt man gut in Abbildung 16, da dort der Schaltkreis mit einem ausreichend langsamen Takt von 10ns, analog der Frequenz von 100MHz, betrieben wird. Vorteil dessen ist, daß man gut sieht, welche Schaltvorgänge nach den Taktflanken passieren. Die Abbildung 16 ist die Hardcopy einer CADENCE-ANALOG-ARTIST-Simulation eines 5-Bit Carry Save Addierers mit Schaltungen vom Aufbau wie in Abbildung 15 dargestellt.

Als Eingangswerte werden drei vierstellige Dualzahlen a_1 ... a_3 gewählt, mit a_1= {1011}, a_2={1101} und a_3={1111}. Als erstes wird für einen Bereich von 17 bis 23ns das RESET-Signal auf „high“ gelegt, um wie oben beschrieben die Latches in einen neutralen Zustand zu versetzen. Man sieht im Bereich von Simulationsstart (0ns) bis RESET-Start, daß die Eingangswerte des Simulators nicht auf low, sondern auf verschiedenen, zufälligen Werten liegen. Dies rechtfertigt in Bezug auf die Realität schon den Einsatz des RESET-Signals. Danach wird die erste Zahl {$a_{1,3}$ ... $a_{1,0}$} der Dualzahlen a_1 bis a_4 parallel ab Zeitpunkt 37ns in die Schaltung eingegeben. Dies reicht offensichtlich als Setup-time aus, um nach der bei 40ns fallenden Taktflanke ein Ergebnis zu produzieren.

Konkret erzeugen also die Werte $a_{1,3}$=1, $a_{1,2}$=0, $a_{1,1}$=1 und $a_{1,0}$=1 nach fallender Taktflake bei 40ns wie erwartet die Ausgangssummen s_3=1, s_2=0, s_1=1 und s_0=1. Diese Werte werden mit der folgenden steigenden Flanke bei 45ns in die Speicherglieder übernommen. Das ist auch notwendig, da mit diesem Takt-„high“ die Signale der nächsten Bits $a_{2,3}$=1, $a_{2,2}$=1, $a_{2,1}$=0 und $a_{2,0}$=1 anliegen. Wie man in Abb. 16 gut erkennen kann, reagiert die Schaltung darauf mit s_3=0, s_2=1, s_1=1 und s_0=0 nach der gefallenen Flanke bei 50ns. Auf die dritte Zahl folgt dann das Ergebnis bei 60ns. Von jetzt ab liegt auch kein neues Eingangssignal an einer der a-Leitungen mehr an. Das Summationsergebnis, welches bei 70ns entsteht, ist das Ergebnis des ersten „Durchrippelns“ von Überträgen. Da die Zahlen möglichst „spektakulär“ gewählt wurden, entsteht zu guter Letzt noch ein Übertrag bei 80ns. Wie an der Bezeichnung der Ausgangssignalleitungen zu erkennen ist, wurde hier in ANALOG-ARTIST ein Addierer simuliert, der über fünf Volladdierer und neun Speicherglieder verfügte.

Zur Einschätzung, welche nun die maximale Grenzfrequenz für diese Schaltung darstellt, lassen wir dieselbe Schaltung mit einem Takt von 2ns arbeiten; das entspricht einer Taktfrequenz von 500MHz. Die Schaltvorgänge sieht man in Abbildung 17.

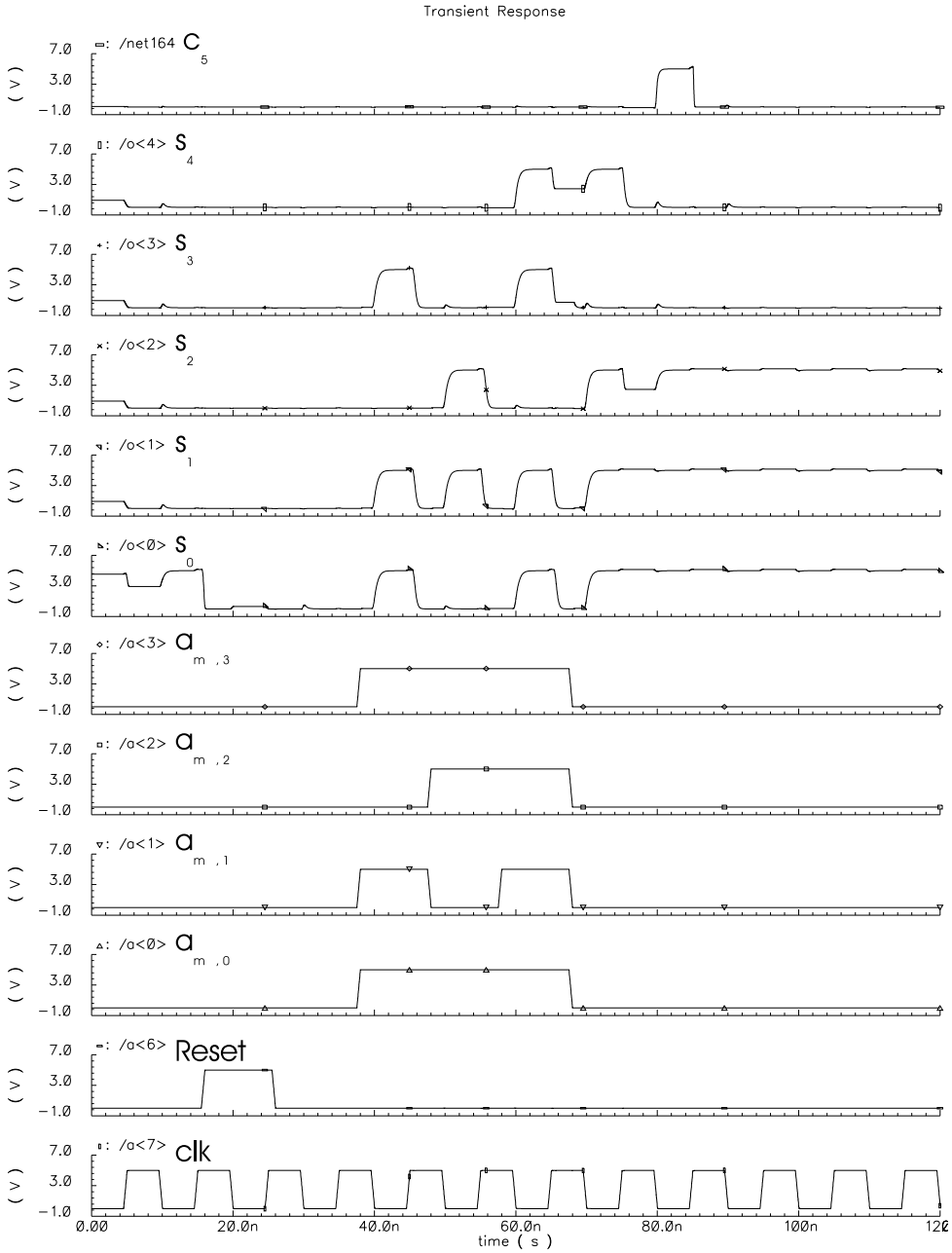


Abbildung 16: Carry-Save-Addierer bei 100MHz

Der Einfachheit halber sind es diesmal nur zwei Summanden mit je zwei Bit Länge: $a_1 = \{01\}$, $a_2 = \{11\}$. Anders als in Abbildung 16 erscheinen nun die Summensignale

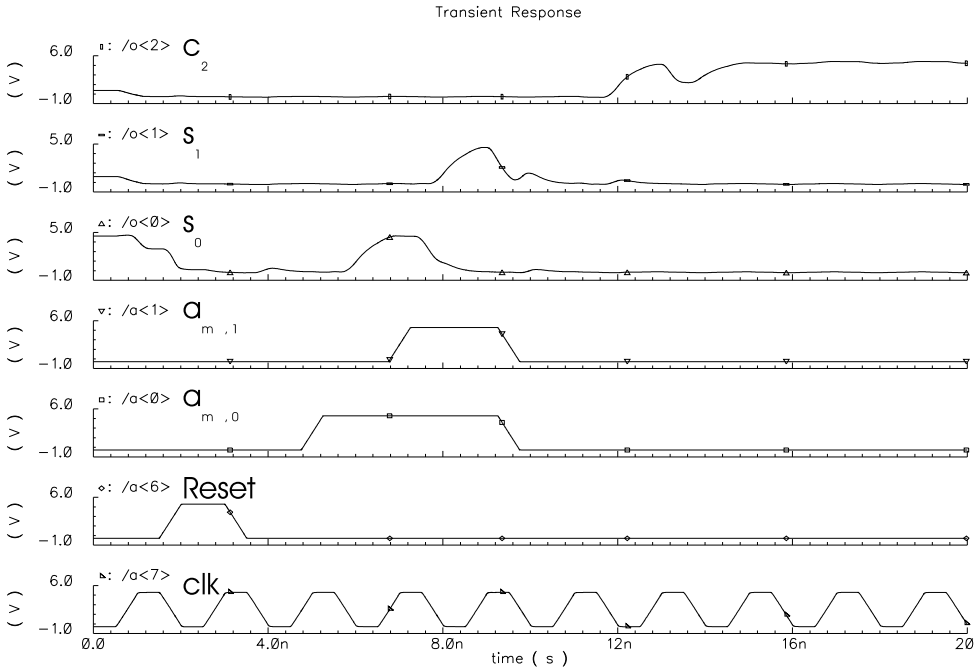


Abbildung 17: Carry-Save-Addierer bei 500MHz

deutlich langsamer ansteigend, d.h. mit einem niedrigeren Anstiegswinkel. Dies zeigt, wie schnell - oder eher wie langsam - die P-Transistoren, die für das Leiten gegenüber der Versorgungsspannung verantwortlich sind, zu Schalten vermögen. Dabei ist wichtig, wie groß die externen Lasten sind, die zur Simulation angelegt werden. Geht man davon aus, daß keine externe Last von der Schaltung getrieben werden muß, so kann man ihre Arbeitsfrequenz unrealistisch schnell angeben. Damit hier eine möglichst realistische Simulation erfolgt, wird jeder Ausgang über ein RC-Glied mit R=50Ω und C=100fF getrieben. Die Flanken des Taktsignals sind hier mit einer Steilheit von 0.5ns eingestellt. Bei der Simulation mit 1GHz beträgt die Steilheit 0.4ns.

Abbildung 17 zeigt, daß der Schaltkreis bei einem Takt von 2ns, was einer Frequenz von 500 MHz entspricht, noch hinreichende Ergebnisse liefert. Auf der Suche nach der maximalen Grenzfrequenz der Schaltung wird diese noch einmal verdoppelt, d.h. der Takt wird von 2ns auf 1ns reduziert. Das Ergebnis der ANALOG-ARTIST-SIMULATION zeigt Abbildung 18. Die Setup- und Hold-Zeiten liegen offenbar unter den Schaltzeiten der Komplexgatter. Das Signal der Summe s_0 schafft es nicht, innerhalb einer Nanosekunde den „high“-Level einzunehmen. Das s_1-Signal kommt durch diesen „undefinierten“ Zustand des s_0-Signals nicht einmal über den Wert von 1V hinaus (man beachte die Ordinate in Abb. 18 bzgl. Signal s_1). Dies führt zu dem Schluß, daß die maximale Taktfrequenz x der Schaltung im Intervall $]1ns < x < 2ns[$ liegt.

Bei statischen oder auch dynamischen Schaltungen ist die Dimensionierung der Transistoren von emminenter Bedeutung. Yuan zeigt in [4], daß auch bei TSPC-Schaltungen die Größe der Transistoren über die Schaltgeschwindigkeit der gesamten Schaltung ent-

scheiden kann. Grundsätzlich gilt, daß je größer der Transistor ist, die Schaltgeschwindigkeit zunimmt. In Schaltungen, wo jedoch Transistoren parallel, seriell oder beides gleichzeitig schalten müssen, kann eine optimierte Dimensionierung zu schnellerem Verhalten führen. So kann man durch Simulation nachweisen, daß Transistoren, die von der Versorgungsspannung oder Ground „weiter entfernt“ in Reihe liegen, mit geringerer Kanalweite realisiert werden können und somit die Schaltung schneller wird. Generell sind durch die unterschiedlichen Schaltverhalten der dotierten Bereiche der Transistoren die P-Transistoren um den Faktor 3 größer anzulegen, als die N-Transistoren.

Dies habe ich ebenfalls untersucht. Ich habe versucht, am Beispiel der oben angeführten NOA22- und NAO22-Komplexgatter einen Vorteil der Schaltgeschwindigkeit durch unterschiedliche Transistorgrößen zu erreichen. Das Ergebnis ist, daß die jeweils maximale Größe zum schnellsten Schalten führt. Das hat meiner Meinung nach folgenden Grund: erstens sind die von mir zum Test gewählten Schaltungen zu wenig komplex, um einen meßbaren Erfolg zu erreichen, und zweitens sind sie vorgeladen. Offenbar ist bei vorgeladenen Schaltkreisen der Geschwindigkeitsvorteil durch das Vorladen höher, als das Schalten von optimal dimensionierten Transistoren in einer statischen Schaltung.

Die Größe der Transistoren ist bei allen hier dargestellten Ergebnissen folgende: 12μm für die Kanalweite der P- und 4μm für die der N-Transistoren. Es ist wohl unnötig zu erwähnen, daß die Dimensionierung der Transistoren neben einer Geschwindigkeitsverbesserung auch zu einer Verkleinerung der gesamten Schaltung - sollte sie denn im Full-Custom-Design erstellt werden - führen kann.

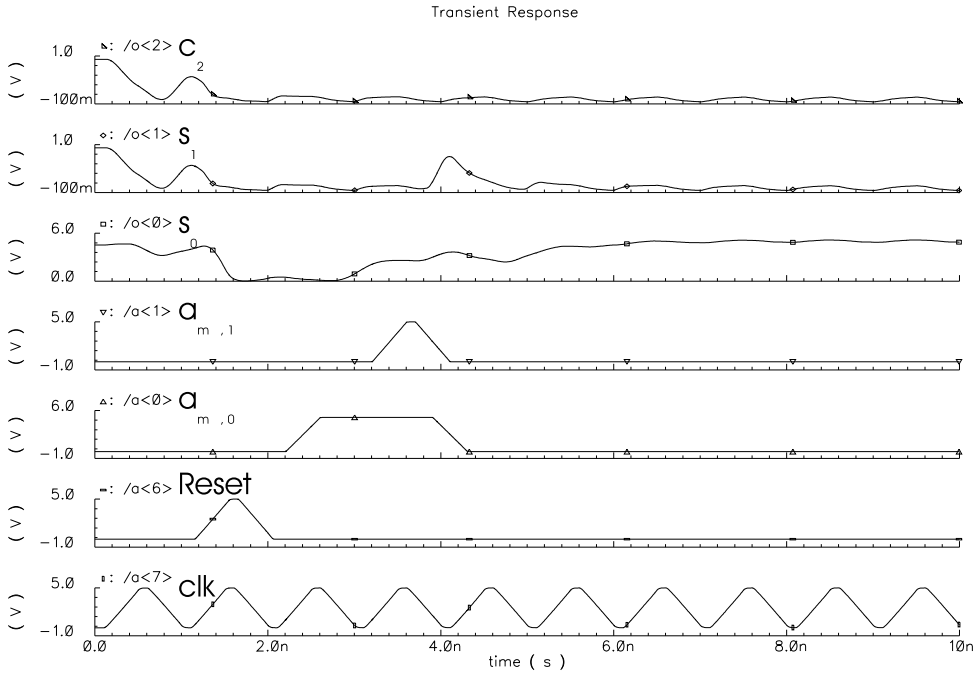


Abbildung 18: Carry-Save-Addierer bei 1GHz

4.2 Carry-look-ahead-Addierer

4.2.1 Logik

Neben der Technik des „Ripple-Carry-Adders“ und dem im vorstehenden Kapitel vorgestellten „Carry-Save-Addierer“ verwendet man häufig den sogenannten „Carry-Look-Ahead-Addierer“ (CLA). Im Unterschied zu den o.g. Addierervarianten müssen beim CLA die Volladdierer nicht auf die Überlaufergebnisse der voranstehenden Volladierer warten. Ein eigenständiges Netz, der sog. „Carry-Look-Ahead-Generator“ (ClaGen) berechnet die Überläufe für die Volladdierer im Voraus.

Gegeben seien wieder Dualzahlen der Form wie im Abschnitt 4.1.1 auf Seite 18. Die gewöhnliche Art, die Summe s_i zu berechnen, ist zuerst den Überlauf der vorherigen Addition c_{i-1} zu bestimmen und dann auf Volladdierer-Art zu verknüpfen:

$$c_0 = 0,$$
$$c_i = (a_{1,i} \wedge a_{2,i}) \vee (a_{1,i} \wedge c_{i-1}) \vee (a_{2,1} \wedge c_{i-1}),$$
$$s_i = (a_{1,i} \oplus a_{2,i} \oplus c_{i-1}), \text{ für i = 1, ... ,n ,}$$
$$s_{n+1} = c_n.$$

Zur Bestimmung des Überlaufs wird bei Carry-Look-Ahead-Addierern das bekannte Schema mit den „generate-“ und „propagate-Funktionen“ benutzt:

$$c_0 = 0,$$
$$c_i = g_i \vee (p_i \wedge c_{i-1}), \text{ mit}$$
$$g_i = a_{1,i} \wedge a_{2,i} \quad \text{„carry generate“}$$
$$p_i = a_{1,i} \oplus a_{2,i} \quad \text{„carry propagate“}$$
$$\text{für i = 1, ... ,n .}$$

Daraus wird ersichtlich, daß der Überlauf c_i entweder von $a_{1,i}$ und $a_{2,i}$ „generiert“, oder vom vorhergehenden Überlauf c_{i-1} „propagiert“ wird. Diese Erkenntnis führt zu einer „Überlauf-Kette“ in Abbildung 19. Aus der nun jeweils vorhandenen Information

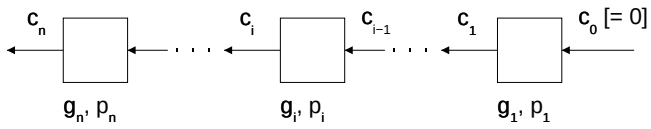


Abbildung 19: Überlauf-Kette

über die Werte g_i und p_i läßt sich die Summe s_i folgendermaßen berechnen:

$$s_i = p_i \oplus c_{i-1}, \quad \text{für i = 1, ... ,n.}$$

R. Brent und H. Kung stellen 1982 in [3] eine Methode auf, mit der man eine Schaltung zur parallelen Ermittlung der Überläufe, sowie nachfolgend die Summen aufbauen kann. Abbildung 20 zeigt ein abstraktes Schema für einen CLA-Paralleladdierer, basierend auf der Grundlage der Überlauf-Kette sowie Brent und Kung’s Methode. Der untere Kasten in Abbildung 20 enthält die Logik zur Berechnung der generate- und propagate-Funktion. Im oberen Kasten erfolgt die Berechnung der Überläufe . Für diese Ermittlung

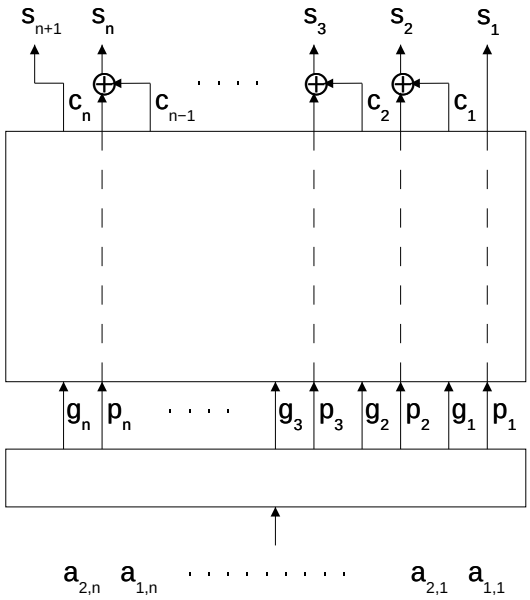


Abbildung 20: CLA-Paralleladdierer-Schema

geben die beiden o.g. Autoren ein günstiges Verfahren an, das unter dem Namen „Brent und Kung's Binary-Lookahead-Carry (BLC) Tree" bekannt ist.

Der BLC-Tree basiert auf einer generellen, zerlegbaren Formel für jeden Überlauf:

$$C_i = g_i + p_i\ g_{i-1} + p_i\ p_{i-1}\ g_{i-2} + ... + p_i\ ...\ p_0\ C_{in}$$

C_{in} bezeichnet dabei den Übertrag, der an Bit 0 anliegt. Der kann nur einen Wert ungleich 0 annehmen, wenn der Block einen Teil einer Kaskade darstellt, d.h. es existiert vor diesem Block noch einer, der einen Übertrag in der höchsten Stelle besitzt. Yuan gibt in [4] eine grafische Darstellung dieser Formel an. Diese zeigt einen 16-Bit Binary-Lookahead-Carry tree um die Überläufe C_0 - C_{15} zu berechnen (s. Abb. 21). Brent und Kung berücksichtigen in ihrer Arbeit, daß bei der Entwicklung der Überläufe C_7 - C_{14} immer das Signal g_{0-7} benötigt wird. Dies führt dazu, daß der das Signal g_{0-7} bereitstellende Schaltkreis eine hohe Ausgangslast zu treiben hat. Aus diesem rein schaltungstechnischen Grund führten sie einen inversen Baum ein, der diese Last verringerte.

Yuan gibt 1995 in [4] eine verbesserte Version dieses BLC-Tree an und nennt diesen „Distributed BLC-Tree", DBLC-Tree. Während der Aufwand des BLC-Trees noch $2 \log_2$ n - 1 Stufen zur Errechnung der Überläufe benötigt, kommt der DBLC-Tree mit einem Aufwand von $\log_2$ n aus.

4.2.2 Aufbau in CDPD-Logik

Der gesamte 16 Bit CLA Addierer besteht aus dem oben beschriebenen DBLC-Tree zur Berechnung der Überläufe zuzüglich 16 Volladdierern. Aus schaltungstechnischen Gründen sind die Volladdierer in zwei Zellen aufgeteilt realisiert, nämlich der Zelle (Sum-1) = $\overline{a\ xor\ b}$ und der Zelle (Sum-2) = (Sum-1) *xor* c_{i-1}.

Der DBLC-Tree ist nach dem Schema in Abb. 21 stufenweise aufgebaut. Die dazu nö-

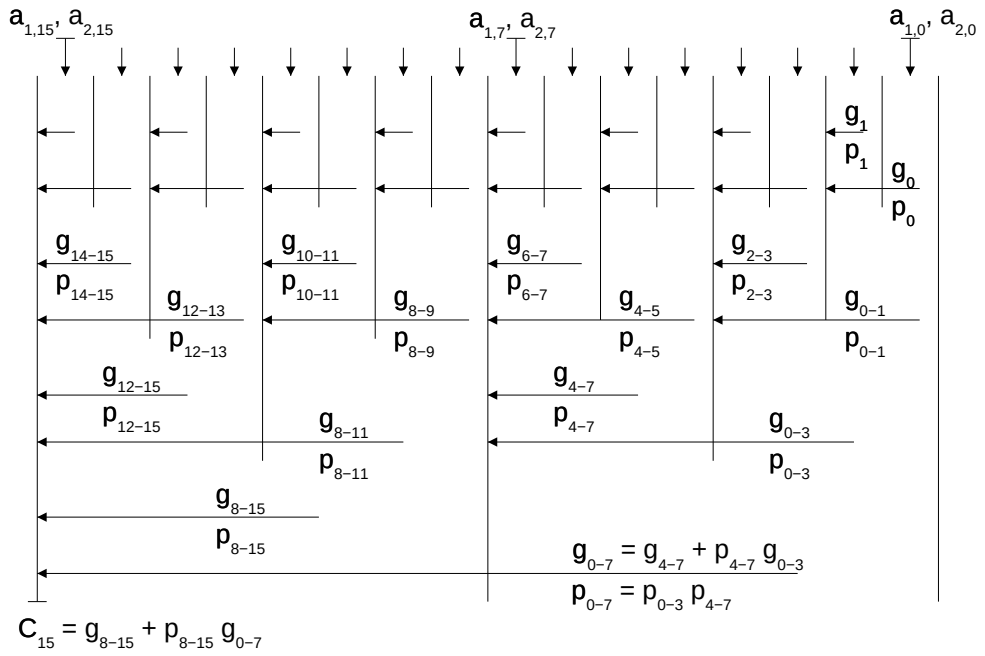


Abbildung 21: BLC-Tree für 16 Bit

tigen Zellen bestehen aus den Gleichungen, die die generate- und propagate-Funktionen über die verschiedenen in Abb. 21 angegebenen Intervalle liefern. Eine Übersicht der verwendeten Zellen und deren Ein- und Ausgabesignale gibt Abb. 22. Die Aufteilung des

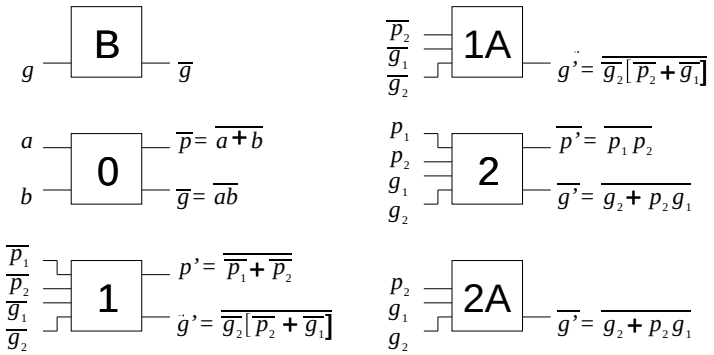


Abbildung 22: Zellen des DBLC-Tree

Gesamtproblems in kleine Zellen hat mehrere Vorteile. Bedingt durch die CMOS-Technik liefern die Zellen invertierte Ergebnisse, so daß man normalerweise in die Zellen Inverter einplanen muß. Da wir im vorliegenden Fall allerdings wissen, daß der gesamte Baum eine Breite von sechs Zellen hat, bekommen wir am Ende ein invertiertes Signal als Ergebnis des Überlaufs heraus. Dieses wird dann noch mit dem Ergebnis der Zelle (Sum-1), das ebenfalls invertiert vorliegt, verknüpft. Das ergibt ein nicht invertiertes Endergebnis, wenn jede Zelle ihre Ausgabewerte invertiert liefert.

Durch die so gesparten Inverter ist die Zeit zur Ermittlung eines Übertrages um die sechsfache Schaltzeit eines Inverters gesunken. Desweiteren lassen sich die Funktionen

in dieser Form gut in CMOS-Schaltungen überführen, wie am Beispiel der ersten Zelle, der Zelle 0 in Abb. 23 gezeigt wird.

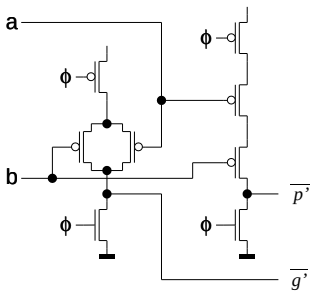


Abbildung 23: Schaltung der Zelle 0 in dynamischer Logik

Wozu ist die Zelle 0 denn nun getaktet ausgeführt? Dazu betrachten wir die komplette Schaltung des DBLC-Tree in CDPD-Technik in Abbildung 24. In der Abbildung 24 ist nur der Carry-Look-Ahead-Baum dargestellt, die beiden Zellen (Sum-1) und (Sum-2) einer jeden Zeile sind der Übersicht wegen weggelassen worden. (Sum-1) befindet sich in der Gesamtschaltung jeweils parallel zur Zelle 0 und ist ebenfalls als getaktete Zelle aufgebaut. Das Ergebnis wird zur Zelle (Sum-2) geführt, die jeweils parallel ganz am Ende jeder Zeile die Überträge mit dem (Sum-1)-Ergebnis verknüpft. Da die (Sum-2)-Zellen mit keinen anderen Zellen parallel abgearbeitet werden können, stellen sie die „sechste Spalte“ des Addierers dar und führen nebenbei die bisher invertierten Signale nach der Verknüpfung in nicht invertierte über.

Die Bedeutung der Taktung in Zelle 0 führt wieder zu der Frage, wie eine solche Zeile am geschicktesten als Pipeline aufzubauen ist. Dazu wird, wie bei dem Aufbau einer TSPC-Schaltung, die Zeile wieder in zeitlich möglichst gleich aufwendige Teile aufgeteilt. In diesem Fall sind es wieder zwei. Der erste Teil besteht aus den Spalten eins bis drei, der zweite aus den Spalten vier bis sechs. Das bedeutet, daß zu Anfang eines jeden Teils mit einer bestimmten Taktflanke die Abarbeitung beginnt und bis zum Ende dieses Teils durchläuft. Hier sind die Zellen 0 als getaktete Stufen konstruiert; die Evaluierungsphase findet bei fallender Flanke statt. Die Zellen der beiden darauffolgenden Stufen zwei und drei sind in CDPD ausgeführt. Die vierte Spalte ist der Anfang des zweiten Teils und wird somit wie die Zellen 0 als taktgesteuerte Spalte ausgelegt. Die Zellen der vierten Spalte sind nicht bei fallender Flanke, bzw. bei Takt-low aktiv, sondern bei Takt-high. Die fünfte Spalte ist wieder in CDPD aufgebaut, während der zweite Teil des Volladdierers in Spalte sechs (bis auf den Übertrag c_{16}) eine getaktete Stufe darstellt. Die Summensignale werden für einen Takt bereitgehalten.

Welche Zellen für diese Schaltung benötigt werden, ist schon in Abbildung 22 dargestellt worden. Die Zelle 0 entwickelt zu allererst aus den zwei Eingangsbits die generate- und propagate-Funktionen. Da diese Zelle stets am Anfang des Baums steht, kommt man um deren Taktabhängigkeit nicht herum. Wann und wo sonst taktabhängige Stufen eingesetzt werden, ist völlig frei wählbar. In diesem Beispiel, wo der zweite Teil mit Spalte

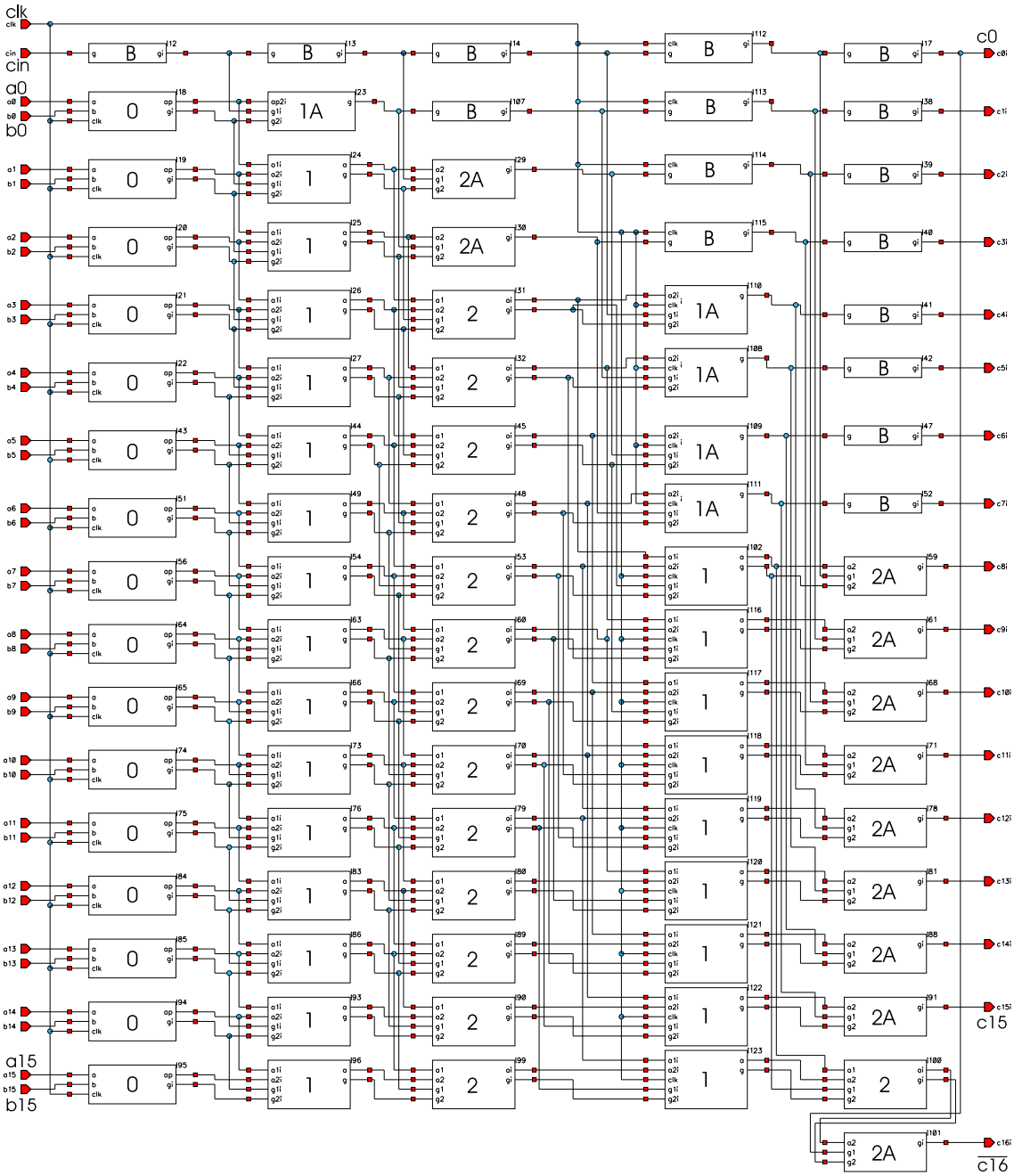
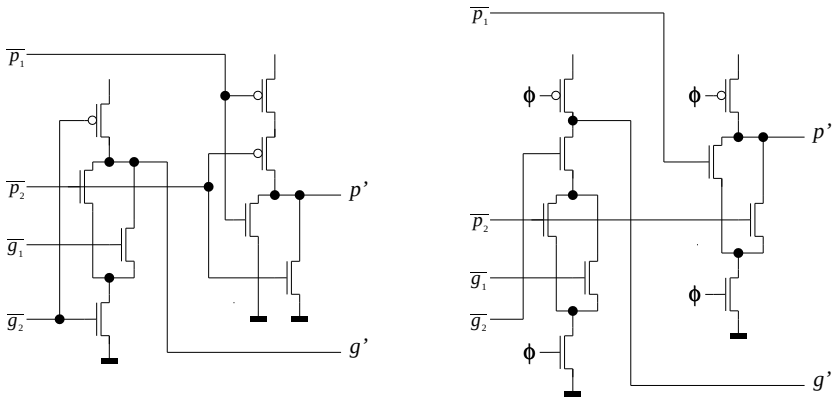


Abbildung 24: DBLC-Tree in CADENCE

vier beginnt, müssen die sich dort befindlichen Zellen B, 1A und 1 taktabhängig gemacht, d.h. in getaktete L/H-Stufen umgesetzt werden. Dadurch ergeben sich dann neue Zellen, die Yuan in [4] folgendermaßen benamt: Zelle B als getaktete Stufe wird z.B. Zelle B-PC; Zelle 1A als getaktete Stufe wird z.B. Zelle 3A und Zelle 1 als getaktete Stufe wird z.B. Zelle 3. Der Übergang von Zelle 1 zur getakteten Stufe Zelle 3 ist in Abbildung 25 verdeutlicht. Zur Verdrahtung eines solchen Systems sei nur noch bemerkt, daß alle



[a] Zelle 1 [b] Zelle 3

Abbildung 25: Übergang von Zelle 1 zu Zelle 3

Überlegungen hinsichtlich Pipelineaufbau, invertierter und nicht-invertierter Signale nur dann funktionieren, wenn jede Spalte ausschließlich von der vorherigen abhängt. Die Grundidee der Schaltungsaufteilung in Taktflanken und somit in bestimmte taktabhängige Spalten ist nicht realisierbar, wenn z.B. Zellen der Spalte fünf von Signalen der Spalte eins abhängen.

4.2.3 Leistungsbewertung

Der gesamte CLA-Addierer mit 16 Bit Breite ist in einer mehrstufigen Hierarchie mittels CADENCE konstruiert worden. Dabei bildeten die Zellen des DBLC-Baumes die unterste Hierarchieebene. Danach kam das „Routing“ der Zellen und die Kontruktion eines davon getrennten Systems für die zwei Teile der Volladdierer. Dies ist auch der Grund für die ausschließliche Darstellung des DBLC-Baums in Abb. 24 ohne die dazugehörigen Volladdiererteile, da die bei dieser Konstruktion ein eigenes System bilden. Zur Verifikation der Ergebnisse hinsichtlich logischer, als auch zeitlicher Art erwies sich dies im Nachhinein jedoch als sinnvoll. Die Simulation erfolgte wieder mittels ANALOG-ARTIST.

Wie auch im o.a. Fall des Carry-Save-Addierers liegt das Hauptaugenmerk auf dem Zeitverbrauch nach den Taktflanken. Davon hängt schließlich maßgeblich ab, wie weit diese zusammengerückt werden können - also wie schnell die Schaltung taktbar ist.

Normalerweise ist von einem ClaGen zu erwarten, daß das „Look-Ahead“ eines Überlaufs des hochwertigen, z.B. 15. Bits länger dauern muß, als das des dritten Bits, beispiels-

weise. Nach Abbildung 24 erkennt man allerdings, daß je höherwertig der zu ermittelnde Überlauf ist, nicht unbedingt mehr generate- und propagate-Verknüpfungen notwendig sind als bei niederwertigen. Somit liegt der zeitliche „worst-case“ nur deshalb in der Ermittlung des höchstwertigen Übertrages, da hier komplexere Zellen in den letzten Spalten durchlaufen werden müssen. Da Überlauf 16 am Ende nicht mehr durch einen Volladdierer mit der Summe eines darauffolgenden Eingangsbitpaares verknüpft wird, liegt er zum Schluß invertiert vor. Allerdings erfordert die Ermittlung des 16. Carries zudem auch noch eine zusätzliche 2A-Zelle, die somit in der sechsten Spalte liegt - sozusagen als Ersatz des sonst üblichen zweiten Teils des Volladdierers.

Die Simulation der Schaltung kann hier aus Gründen der Übersicht nicht mit der vollen 16 Bit Breite als Eingangs- und Ausgangsgraph dargestellt werden. Es wurde deshalb ein festes Bitmuster durch Verdrahtung in der CADENCE-Testumgebung des Addierers an die Schaltung angelegt, wie in Abbildung 26 gezeigt. Dieses Bitmuster sorgt für eine

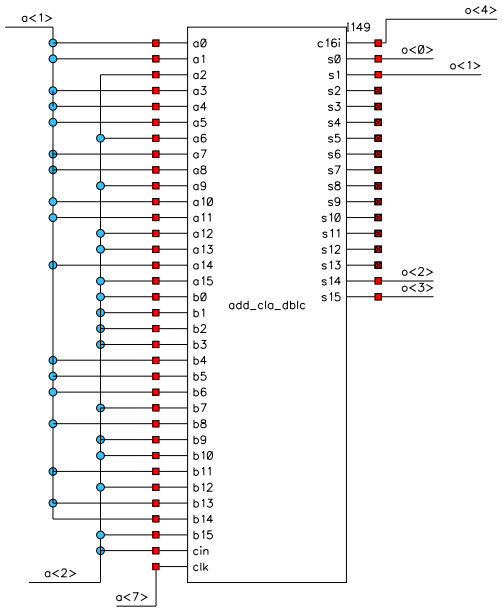


Abbildung 26: Testumgebung 16 Bit CLA-Addierer in CADENCE

Addition des nachfolgenden Schemas. Das freie Feld neben c_i im Schema ist das Feld c_0, welches ja immer low einnimmt, wenn dieser Baum nicht mindestens zweiter Teil einer Kaskade ist. Somit berechnet sich $a_{1,0} + a_{2,0} + c_0 = c_1\ s_0 \Rightarrow 1 + 0 + 0 = 0\ 1$. Dieses Verfahren ist ja schon im Logik-Abschnitt dargelegt worden.

	15					10					5					0	i
	0	1	0	0	1	1	0	1	1	0	1	1	1	0	1	1	$a_{1,i}$
+	0	1	1	0	1	0	0	1	0	1	1	1	0	0	0	0	$a_{2,i}$
0	1	0	0	1	0	0	1	1	1	1	1	0	0	0	0		c_i
0	1	0	1	1	0	1	1	1	0	0	1	0	1	0	1	1	s_i

Wie verhält sich nun der DBLC-Baum in der ANALOG-ARTIST-Simulation? Bei Anlegen einer Taktfrequenz von < 100 MHz zeigt - außer einem korrekten Schaltverhalten

- nichts interessantes. Das Verhalten bei einer Taktverzögerung von 2 ns, das entspricht 500 MHz bringt aber schon die wachsende Ausgangsverzögerung der einzelnen Spalten zum Vorschein (s. Abb. 27). Wie schon im vorherigen Abschnitt erwähnt, liegen in Abb. 27 bis auf c_{16} alle Übertäge invertiert vor. Das unterste Signal stellt den Takt dar, die Signale der einzelnen Überträge sind darüber aufgeführt.

Der interessante Bereich liegt zwischen der ersten und zweiten fallenden Flanke, also zwischen ca. 1.8 und 3.8 ns. Die erste fallende Flanke repräsentiert den Beginn des Schaltvorganges der ersten drei Spalten, deren Schaltende für alle Signale bei ca. 2.8 ns erreicht ist. Bereits im Takt-high-Bereich vor der fallenden Flanke profitieren die Spalten von der geeigneten Vorladung der Zellen. Man sieht, daß die meisten Signale schon vor der fallenden Flanke, nämlich bei ca. 1.2 ns beginnen, einen Vorladezustand einzunehmen.

Die darauffolgende steigende Flanke bei ca. 2.9 ns sorgt für das Schalten der vierten und fünften Spalte. Interpretiert man unter Zuhilfenahme von Abbildung 24, so sieht man, daß für die Überträge c_0 bis c_3 nur zwei Inverter nötig sind. Deshalb nehmen diese extrem schnell bei ca. 2.9 ns ihren Endwert an. Die Signale c_4 bis c_7 benötigen die Zelle 1A zzgl. einem Inverter. Das läßt die Signale etwas langsamer zum Endzustand bei ca. 3.2 ns kommen. Die Überträge c_8 bis c_{15} müssen durch die Zellen 1 und 2A bestimmt werden. Das bedeutet den höchsten Zeitaufwand und dieses Schalten ist bestimmt durch eine flach ansteigende Signalflanke mit Erreichen des maximalen Spannungspotentials bei ca. 3.9 ns. Mit der fallenden Flanke bei ca. 3.9 ns ist der eine Taktdurchlauf, der zum kompletten Schaltungsdurchgang angesetzt wurde, beendet und das Ergebnis kann übernommen werden. Die neuen Eingangswerte könnten nun schon für die nächste Berechnung anliegen.

Kamen die Überträge nun frühzeitig genug bei den zweiten Volladdiererteilen an, um bei Takt-high zur richtigen Summe s_i zu führen? Dazu betrachten wir erst einmal die Komplettschaltung in der Simulation mit einer Taktverzögerung von 10ns in Abbildung 28. Abbildung 28 zeigt den Signalverlauf für den Takt, sowie den resultierenden Signalen s_0, s_1, s_{14}, s_{15} und s_{16}. Wie beschrieben, entstehen die Summen aus zwei Zellen (Sum-1) und (Sum-2). (Sum-1) ist eine H/L-Stufe und schaltet gleichzeitig mit Spalte eins, (Sum-2) ist eine L/H-Stufe und schaltet gleichzeitig mit Spalte vier. (Sum-1) hat dieselben Eingangsbits wie Spalte eins, also existieren keine schaltinternen Abhängigkeiten. An (Sum-2) liegt der Ausgang von (Sum-1) an, sowie je ein Übertrag, was bedeutet, (Sum-2) schaltet erst richtig im Takt-high-Bereich, wenn die Überträge anliegen.

Auch in Abbildung 28 interessiert der Bereich zwischen der ersten und zweiten fallenden Taktflanke, also der Zeitbereich von 9.5 bis 19.5 ns. Hier ist wie in Abb. 27 zu erkennen, daß das Vorladen der Zellen schon zu einem Signalresultat bei Takt-high führt. Mit der fallenden Flanke bei 9.5 ns schalten die ersten drei Spalten. Dies führt durch die geeignete Vorladung zu kaum einer Veränderung der s_i-Signale. Mit steigender Flanke bei 14.5 ns erkennt man anhand s_0, s_1 und s_{14}, daß die Zellen (Sum-2) ohne die Signale der

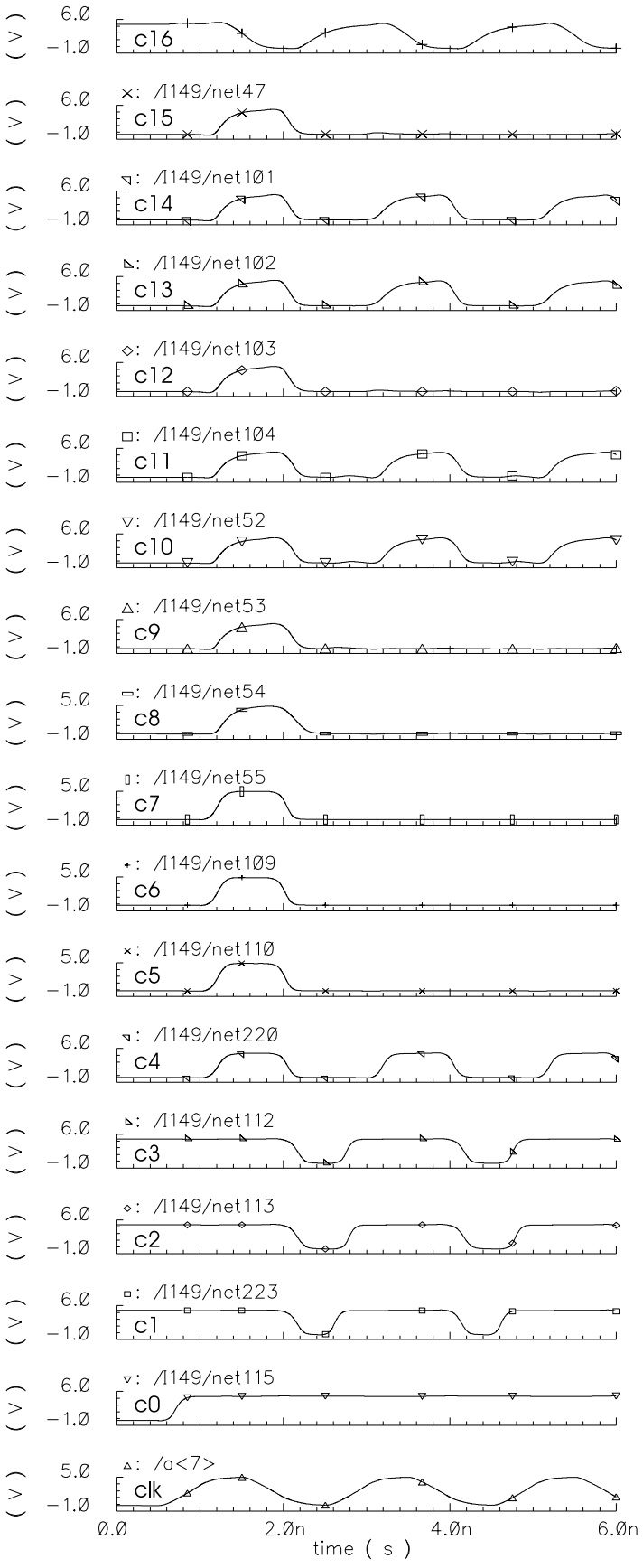


Abbildung 27: Signalgraph der DBLC-Baum-Überträge bei 500 MHz

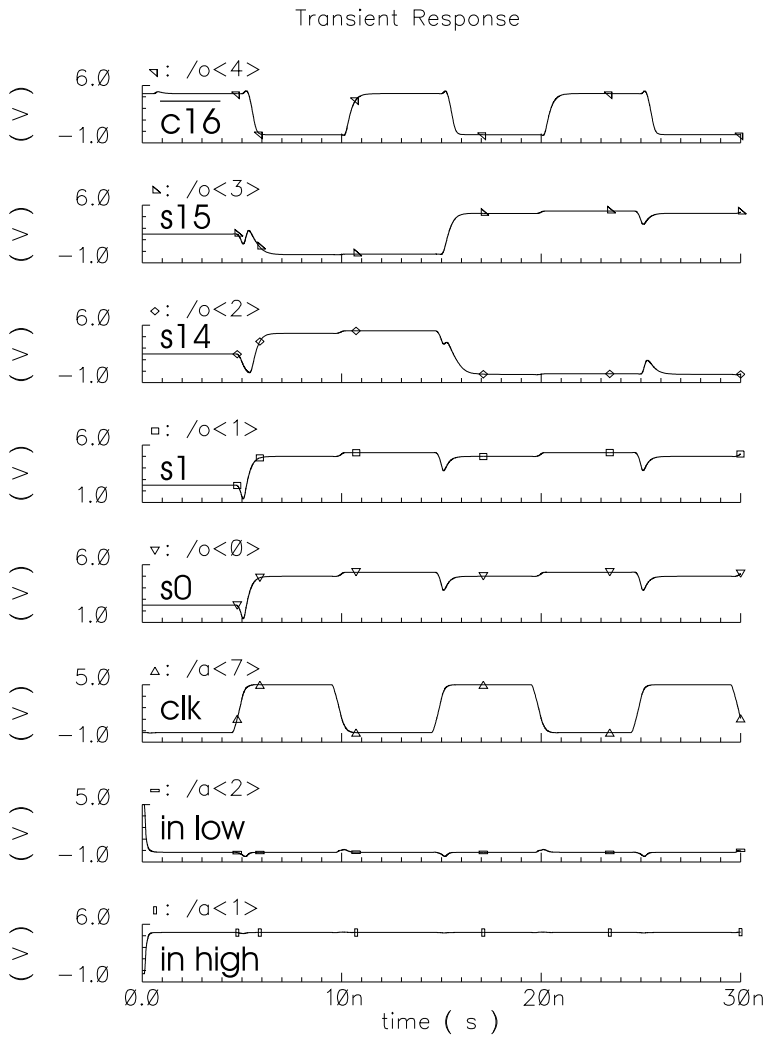


Abbildung 28: CLA-Addierer mit 100MHz

vorherigen Überträge schalten. Diese liegen im Moment der Taktflanke noch nicht vor, da mit der Flanke erst die Spalten vier und fünf zur Ermittlung der Überträge schalten. Da, wie o.a. dazu aber nur ein kurzer Schaltweg nötig ist, ändert sich das Signal nach Vorliegen des Übertrags bei ca. 15 ns sofort. Diese ca. 0.3 bis 0.5 ns haben die Spalten vier und fünf gebraucht, um die (Sum-2)-Zellen mit dem Übertrag zu versorgen. Anhand der Signale s_0 und s_1 sieht man, daß ohne die letztendlich doch noch eingetroffenen Überträge, die ja den Wert 0 haben (c_0=0, c_1=0, s. Abb. 27), beide den Wert low anstrebten. Nur durch die kurz später berechneten Überträge änderte die Zelle (Sum-2) ihren Endwert auf high. Der umgekehrte Fall liegt bei s_{15} vor. Hier schaltet (Sum-2) nach steigender Flanke sofort nach high, das kurze Zeit später kommende Übertragssignal c_{15}=1 ändert daran nichts. Man sieht, daß der Addierer bei 16 ns das Ergebnis errechnet hat. Dieser Wert wird durch die speichernde Eigenschaft der L/H-Zelle (Sum-2) bis zur nächsten steigenden Taktflanke erhalten bleiben.

Wesentlich zur Ermittlung der maximalen Taktfrequenz ist offensichtlich die Frage, wie schnell nun genau die Spalten schalten können, und wie langsam die Übertragwerte höchstens an die (Sum-2)-Zellen gelangen dürfen um noch zu einem faktisch richtigen s_i-Signal zu führen. Zur weiteren Untersuchung wird die Taktfrequenz auf 500 MHz, also einer Taktverzögerung von 2 ns, erhöht. Abbildung 29 zeigt das Ergebnis einer entsprechenden ANALOG-ARTIST-Simulation. Wie auch in der vorangegangenen Abb. 28 wurde hier ebenfalls die festverdrahtete Addition aus Abb. 26 benutzt. Auch die Reihenfolge der Signale dieser Abb. 29 stimmt mit der in Abb. 28 überein.

Man erkennt deutlich den Effekt des Vorladens vor der ersten fallenden Taktflanke. Dieser sorgt dafür, daß die Zellen im Augenblick der fallenden Flanke ein high-Potential aufgebaut haben, daß groß genug ist, um den Transistoren der vierten und fünften Spalte Grund zu geben, sinnvoll zu schalten. Den o.g. Effekt des späteren Eintreffens der Überläufe an den (Sum-2)-Zellen findet man sehr deutlich wieder, insbesondere an den Signalen s_0, s_1 und s_{14}, die dadurch umgeladen werden. Trotz der geringen Taktverzögerung von 2 ns reicht offensichtlich die Zeit aus, um bei der fallenden Flanke bei ca. 3.9 ns eindeutige Signalunterscheidungen hinsichtlich „low“ oder „high“ treffen zu können. Es scheint aber ratsam, bis zur Vollendung des Takt-low abzuwarten, da die Signale dann bei ca. 4.2 bis 4.4 ns deutlicher interpretierbar ausfallen.

Als nächstes wird die Schaltung bei einer Taktfrequenz von 1 GHz beobachtet. Die Ausgangssignale bei dieser Taktperiode von 1 ns Länge sind in Abb. 30 zu sehen. Vergleicht man das Signalverhalten mit dem in Abb. 29, so erkennt man z.B. bei Signal s_{14} sehr deutlich, daß die P-Transistoren bei dieser Taktgeschwindigkeit nicht mehr in der Lage sind, dem Signal einen deutlichen high-Pegel zwischen 0.7 und 1.5 ns zu verschaffen. Es kommt hier zu einem eindeutig falschen Ausgangswert für die Summe s_{14}. Eine Taktfrequenz von 1 GHz ist für diese Schaltung mit Realisierung in einem 0.6 μm zweilagen-Metall-Prozeß zu hoch.

Die exakte, maximale Taktfrequenz zu bestimmen ist hier sicher ansatzweise möglich,

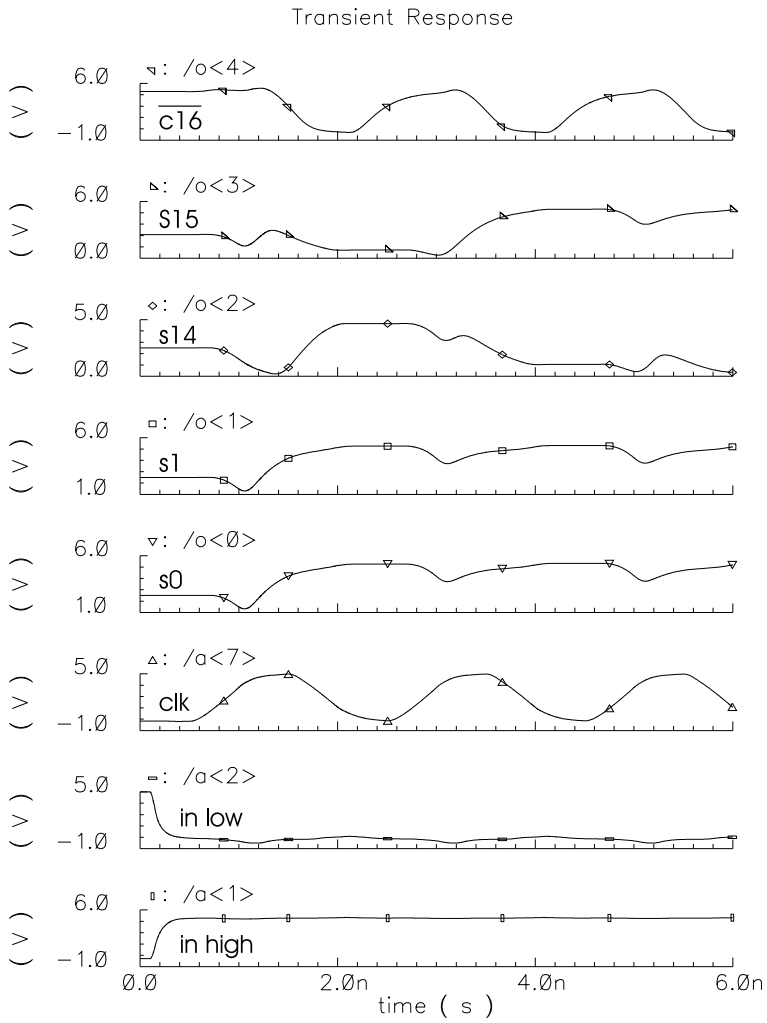


Abbildung 29: CLA-Addierer mit 500MHz

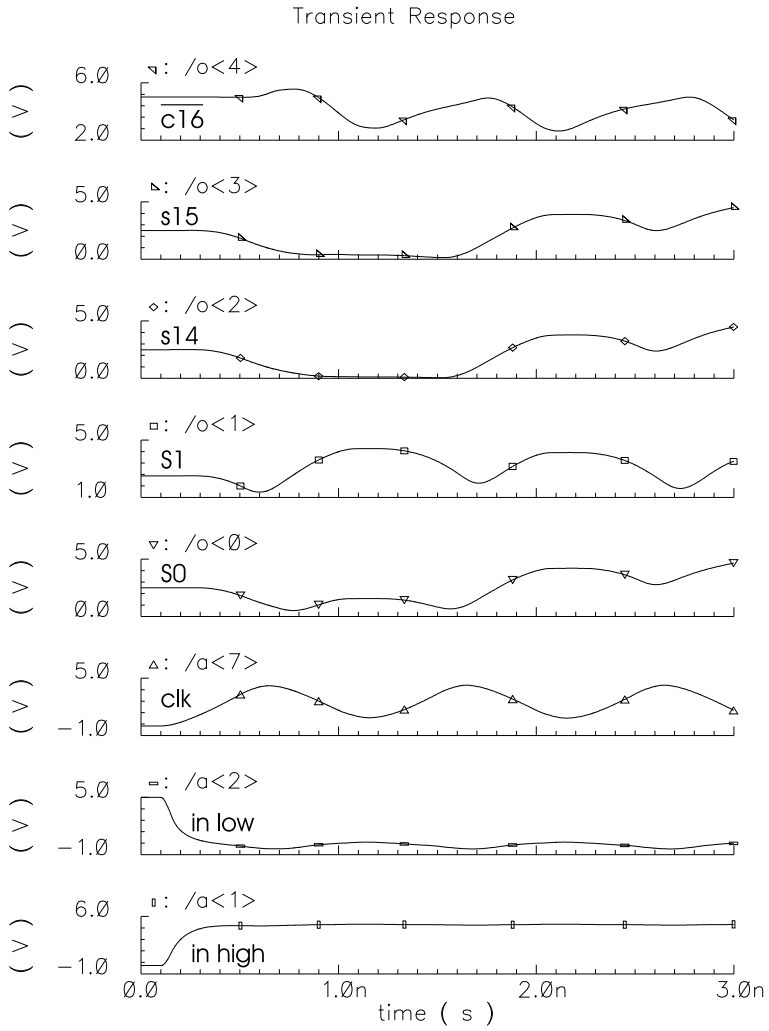


Abbildung 30: CLA-Addierer mit 1 GHz

aber über den Sinn ließe sich streiten. Zum einen ist der hier simulierte Fertigungsprozeß nicht mehr zeitgemäß, zum anderen läßt sich die korrekte Funktion der Schaltung nur bei weiteren, hintereinander ausgeführten Additionen testen. Auch die einzelnen Lasten, die an den Signalen hängen müßten variiert und getestet werden.

Die Simulation mit einer Taktperiode von 1.8 ns zeigte noch ein korrektes Schaltverhalten, bei 1.6 ns Taktperiode kam die erreichte Spannung der high-Pegel schon nur noch auf kritische Werte von ca 4.0 V.

5 Entwurfsvorgehen

5.1 Techniken

Bei Standardbausteinen wie Speicher und Mikroprozessoren bedient man eine sehr große Nachfrage des Marktes. Die Abnahme großer Stückzahlen eines Bauteils rechtfertigt einen hohen Entwicklungs- und Entwurfsaufwand. Bei kleinen Stückzahlen ist dieser Aufwand jedoch unwirtschaftlich. Man versucht den Entwicklungsaufwand und damit die Kosten zu minimieren, in dem man die Anzahl der Prozeßschritte reduziert. Dies kann man dadurch erreichen, daß man auf gegebene Chipstrukturen zurückgreift. Der Nachteil dabei ist, daß die erreichbare Schaltungsdichte bei der Verwendung von vorgegebenen Zellen im allgemeinen geringer ist als bei voll-kundenspezifischen Schaltungen.[4]

Man unterscheidet nach [8] vier Gruppen von Entwurfstechniken:

- Voll-kundenspezifischer Entwurf („full custom design“):
  Es wird keinerlei Beschränkung beim Entwurf gemacht, um so für jede Schaltung („kundenspezifisch“) das optimale Layout zu erhalten.

- Zellenentwürfe („cell design“):
  Hierbei wird auf eine Bibliothek bestehender Zellen zurückgegriffen. Das Layout besteht darin, diese Zellen zu plazieren und zu verdrahten.

- Entwurf mit Arrays („array logic“):
  Man geht von einer vorgefertigten, regelmäßigen Anordnung von Transistoren aus. Das Layout beschränkt sich auf eine oder mehrere Verdrahtungsebenen. Entwurf mit Zellen und mit Arrays wird oft unter dem Begriff semi-kundenspezifischer Entwurf („semi custom design“) zusammengefaßt.

- Entwurf mit programmierbaren Schaltungen („programmable logic“):
  In diesem Fall werden Schaltungen verwendet, die extern durch den Anwender programmiert werden können. Der Entwurf beschränkt sich nur noch auf die Programmierung der Schaltung.

Die Minimierung der Kosten durch die Vermeidung eines full custom Entwurfs ist im Falle von TSPC oder CDPD leider unmöglich, da TSPC von einer Minimierung der Leitungen und Transistoren profitiert. Bestehende Zellen, Arrays oder programmierbare Schaltungen bieten diese Möglichkeiten leider nicht an. Aus diesem Grund liegt die Realisierung von TSPC- oder CDPD-Schaltungen im Full-Custom-Design. Das bedeutet auf der einen Seite grenzenlose Freiheit und Innovation bzgl. Place und Route, auf der anderen Seite hohe Verantwortung und Logik im Sinne von Kompaktheit und Ökonomie.

[4] Rosenstiel, Camposano [8]

5.2 Full-Custom-Entwurf

Eine der Aufgaben dieser Arbeit ist der Entwurf einer speziellen Schaltung im Full-Custom-Design. Im Folgenden wird der Entwurf des Carry-Look-Ahead Addierers aus Abschnitt 4.2 beschrieben.

5.2.1 Entwurf der Zellen

Für den Entwurf stehen hier die Mittel eines 0.6μm zweilagen-Metall-Prozesses von der Firma AMS zur Verfügung. Die einzelnen Zellen des Addierers sind in ihren Transistorschaltungen bereits entworfen und getestet, also als sog. „Schematics“ bekannt.

Zu diesen Schematics bildet man nun nach den Grundsätzen der CMOS-Technik ein Layout. Dabei hat natürlich die Platzminimierung oberste Priorität. Hier wird mit dem CADENCE-VIRTUOSO-Layout-Editor das Layout entworfen. Dazu stehen im Editor einige fertige Komponenten zur Einbindung in das eigene Layout zur Verfügung, wie z.B. frei dimensionierbare N- und P-Transistoren, Kontaktpunkte (VIA's) zwischen den verschiedenen Metall- und Polysiliziumschichten, Kontakte für die N- und P-Wannen, etc.. Diese vorgefertigten Komponenten erleichtern die Arbeit deshalb, da man bei der Konstruktion von einem der vielen Transistoren nicht jedesmal von neuem die exakten Design-Regeln der einzelnen Bestandteile nachprüfen muß, sondern nur noch die Entfernung zu anderen Bauteilen.

Wichtigstes Kriterium bei dieser Art des Entwurfs ist die Einhaltung der sog. „Design-Rules“, die man über den Editor prüfen lassen kann. Dieser „Design-Rule-Check“ (DRC) ermittelt alle nicht realisierbaren Strukturen in Abhängigkeit vom verwendeten Prozeß. Dazu gehören z.B. die Mindestbreiten der Leiterbahnen oder die Mindestabstände zwischen N- und P-Wannen.

Vor der Konstruktion der Zellen ist die Planung des Gesamtlayouts wichtig. Dazu gehört vor allem die Überlegung, wie man die Leiterbahnen durch das Layout zieht. In dem hier verwendeten Prozeß stehen drei Typen zur Verfügung: Polysilizium für kurze Verbindungen, da dieses einen sehr hohen Bahnwiderstand besitzt, Metall-I und Metall-II für weitere Strecken. Metall-I wird hier vor allem zur Spannungsversorgung eingesetzt. Vdd und Ground sind horizontal am oberen, bzw. unteren Rand jeder Zelle durchgeführt. Dies erfordert von jeder Zelle einer Reihe dieselbe Höhe. Metall-II dient den vertikalen Verbindungen der Zellen, die für die Datensignale benötigt werden. Um das spätere Verbinden der Zellen in vertikaler Richtung möglichst einfach zu halten, werden die Zellen auch in der Breite ausgerichtet angeordnet, d.h. jede Spalte des Layouts ist durch die jeweils breiteste Zelle festgelegt.

Nach diesen Grundüberlegungen können die einzelnen Zellen konstruiert werden. Da jede Zelle durch ihre jeweilige Breite und Höhe das Gesamtlayout bestimmen, versucht man diese möglichst klein zu halten. Am Beispiel der Zelle 0 in Abbildung 31 kann man erkennen, daß nicht für jeden im Schematic (vgl. Abb. 23) vorgesehenen Transistor der

gesamte Platz eines solchen benötigt wird. Vielmehr werden funktional passende Sources oder Drains zusammengelegt, bzw. passende Transistorseiten ineinandergefügt.

Am oberen und unteren Ende der Zelle befinden sich die Leitungen zur Spannungsversorgung mit einer Breite von 3 μm, wobei unten Vdd und oben Ground anliegt. Entsprechend den o.g. Grundüberlegungen sind sie aus Metall-I, dies ist mit einer Schraffur von links unten nach rechts oben dargestellt. Die Gates aller Transistoren, sowie einige kurze Leitungen sind aus Polysilizium und haben hier ein geordnetes, gepunktetes Muster. In der linken unteren Ecke, knapp oberhalb der Vdd-Leitung liegt ein Kontaktpunkt, der Metall-I mit Polysilizium verbindet. Diese VIAs sind mit einem schwarzen Quadrat dargestellt. Auf der rechten Seite des Zellenlayouts erkennt man eine Leitung mit Schraffur von links oben nach rechts unten; dies ist Metall-II. Diese Leitung hat keinen Kontakt zur Zelle 0, d.h. sie führt nur ein Signal durch dieses Gebiet. Dieses Signal ist das Ergebnis der Zelle (Sum-1), daß durch den gesamten Chip zur allerletzten Reihe geführt werden muß, um dort einen Eingang der Zelle (Sum-2) zu bilden. Wo es der Platz nicht mehr erlaubte, Polysilizium- oder Metall-I-Bahnen zu benutzen, wurde auf kurzen Wegen auch mit Metall-II gearbeitet. So beipielsweise von der Ecke links oben, wo eine Metall-II-Bahn das Taktsignal aus der Reihe 0 heranführt. Diese führt nach unten und nach rechts zu einem VIA, welches einen Kontakt zu Polysilizium herstellt. Diese Poly-Bahnen sind die Gates beider obenliegenden N-Transistoren. Von diesem Kontaktpunkt geht ebenfalls eine Poly-Bahn nach unten, teilt sich und stellt die Gates zweier der untenliegenden P-Transistoren dar.

Im Folgenden noch ein paar Design-Regeln zum Entwurf eines Layouts mit diesem Prozeß: Metall-x Bahnen-Breite: 0.8 μm; Metall-x - Metall-x Abstand: 0.8 μm; Poly Bahnen-Breite: 0.5 μm; Poly - Poly Abstand: 0.5 μm.

5.2.2 Place and Route

Nachdem alle verschiedenen Zellen einmal erstellt sind, kann man sie mittels „Kopieren und Einfügen“ zu einem Gesamtschaltbild zusammenfügen. Dabei ist zu beachten, daß die Vdd- und Ground-Leitungen in Metall-I auch richtig zusammenliegen und nebeneinanderliegende Zellen die Mindestabstände bzgl. DRC einhalten. Ist eine Reihe fertiggestellt, beginnt man mit der Verdrahtung der Signalleitungen zur nächsten Reihe. Dabei zeigt sich der Vorteil der Vorüberlegungen, die Zellen auch vertikal nach o.g. Raster auszurichten. Nachteilig ist der Umstand, daß die breiteste Zelle die minimale Spaltenbreite definiert. Da die Zellen Sum(1) und 0 die breitesten darstellen, ergeben sich im Mittelteil große „Lücken“ mit ungenutzter Chipfläche. Die hier angegebene homogene Anordnung der Zellen ist bzgl. der Verdrahtung optimal und laßt eine nachträgliche Veränderung der Struktur vollkommen zu. Fraglich bleibt, ob eine nachträgliche Verdichtung der Zellen nicht zu einer Vergrößerung der Verdrahtungsfläche führen würde.

Abbildung 32 zeigt das Layout des kompletten Addierers. Im Unterschied zu Abbildung 24 auf Seite 30 sind hier in Reihe 0 und 6 die Zellen (Sum-1) und (Sum-2) vorhan-

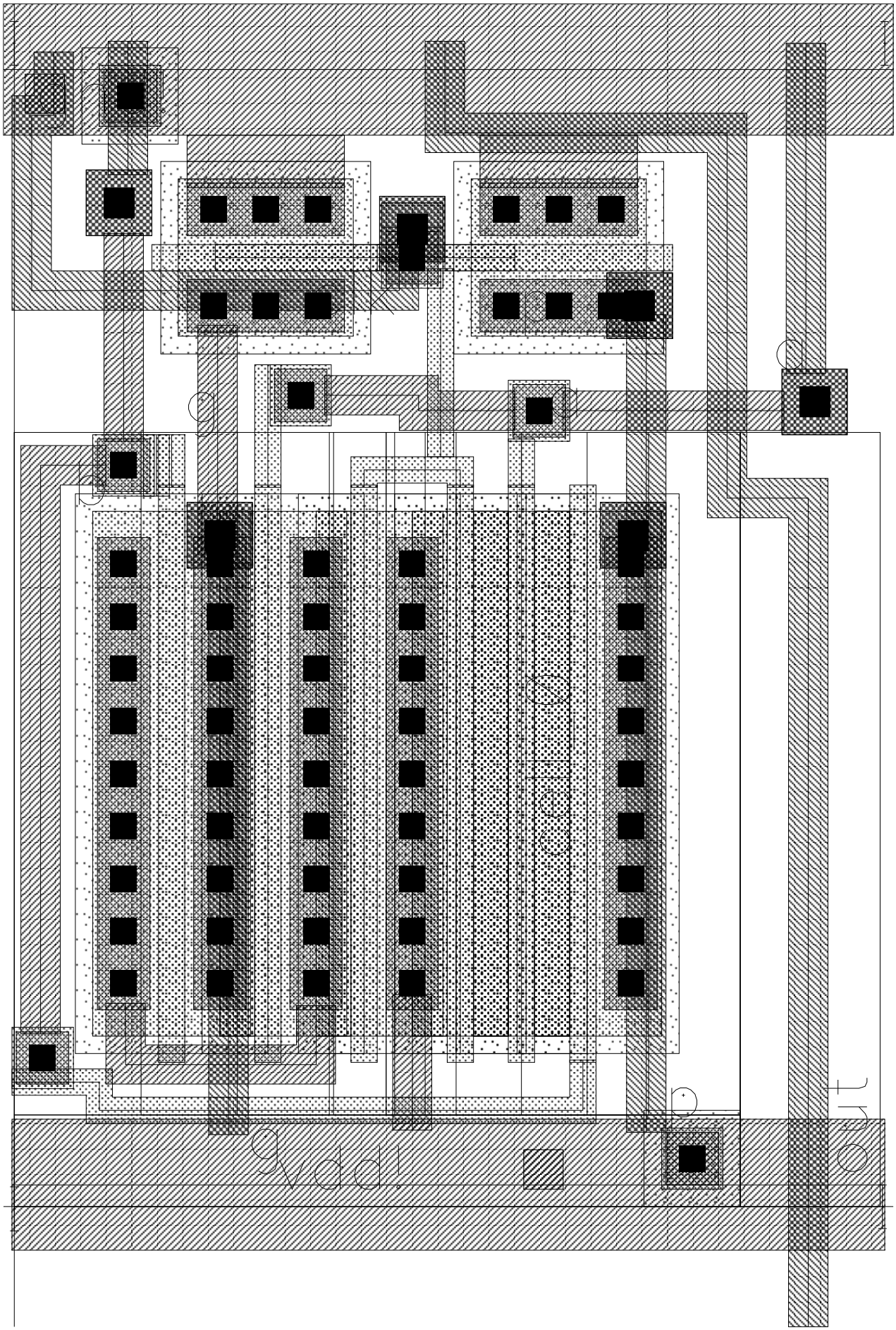


Abbildung 31: CMOS-Layout der Zelle 0

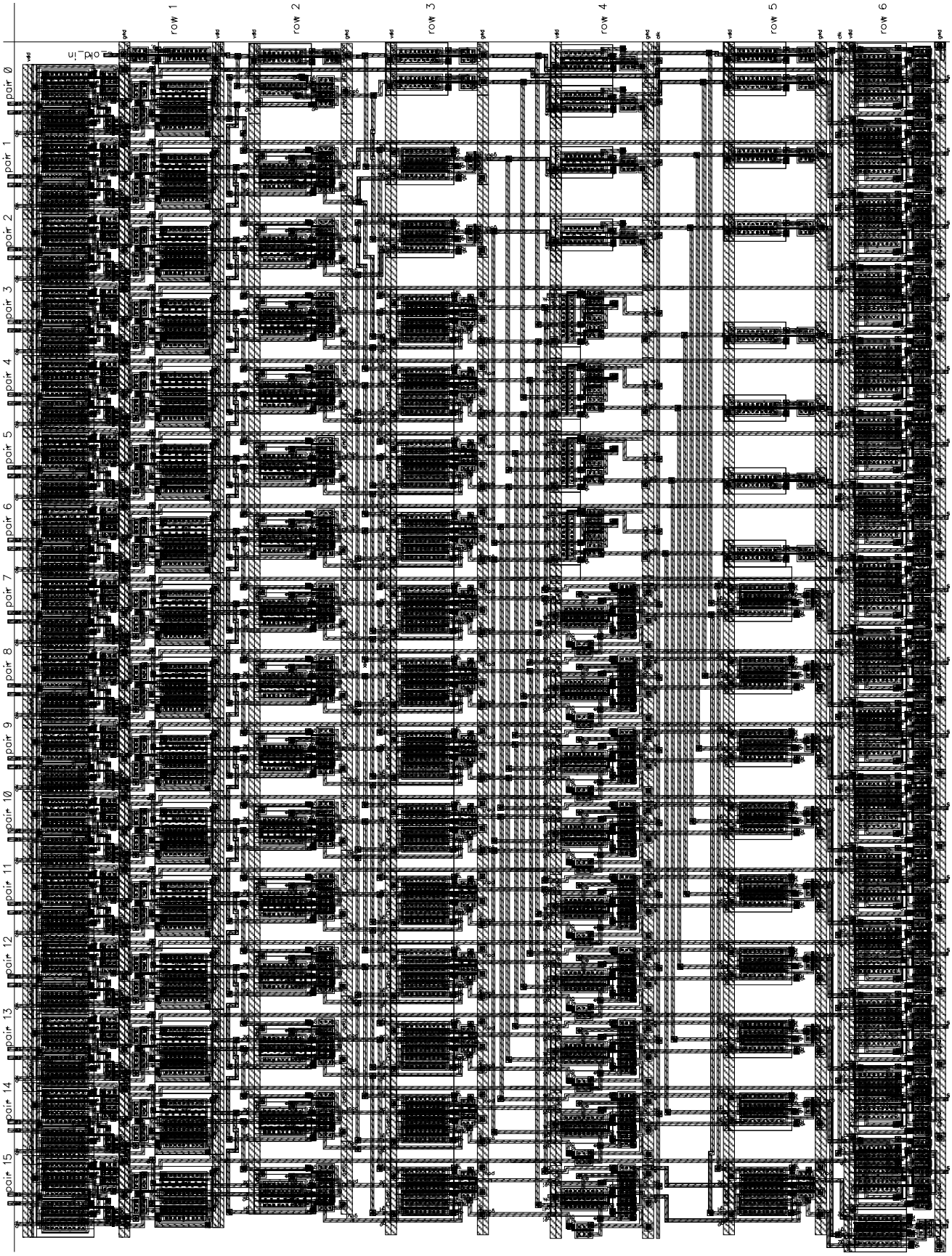


Abbildung 32: CMOS-Layout des 16-Bit CLA-Addierers

den. Ansonsten sind die beiden Abbildungen schematisch identisch, nur die Zelle 2A, die das Signal $\overline{c_{16}}$ liefert ist in die letzte Zeile 6 angehängt worden. Auf der linken Seite des Layouts befinden sich die Eingänge a_0, b_0 bis a_{15}, b_{15} beschriftet mit „pair 0“ bis „pair 15“. Jeweils links neben den beiden Eingangsleitungen für jedes pair befindet sich noch ein Eingang für das Taktsignal. Am oberen Ende des Schaltung, bei der Beschriftung der Reihen, befinden sich die Anschlüsse von Vdd und Ground, zwischen den Reihen vier, fünf und sechs zusätzlich jeweils ein Takteingang. Gut erkennbar ist das Schema, daß die Zellen reihenweise dieselbe Höhe aufweisen um die Versorgungsspannung und Ground durchkontaktieren zu konnen. Die Zelle 0 stellt die gesamte Reihe 1 dar, bis auf die erste Zelle.

5.2.3 Verifikation

Ist das gesamte Layout fertiggestellt, so stellt sich die Frage, ob sich dieses exakt so verhält, wie es das Schematic getan hatte. Dazu verbindet man zuerst die einzelnen, horizontal verlaufenden Vdd-, Ground- und Taktleitungen. An diesen und den Dateneingangs- und Ausgangsleitungen definiert man „Pins“, die CADENCE mitteilen, welche Funktion sie haben. Zum automatischen Vergleich müssen die Bezeichnungen der Pins im Layout mit denen im Schematic identisch sein.

Über die CADENCE-Funktion „Layout versus Schematic“ (LVS) werden die beiden Entwürfe miteinander verglichen. Dazu erzeugt CADENCE zwei ungerichtete Graphen und versucht u.a. anhand der Benamung die Übereinstimmung zu prüfen. Abbildung 33 zeigt das positive Ergebnis dieser LVS-Prüfung, nachdem sämtliche Fehler, wie falsch verdrahtete Transistoren oder Zellen und durch Zusammenführung unterbrochene Leitungen korrigiert worden sind. Man sieht, daß die Schaltung 53 Anschlußterminals besitzt, sowie 434 N- und 463 P-Transistoren. Zum Vergleich fand CADENCE 53 gleichnamige Terminals; unter der Angabe deren Namen findet man die erlösende Meldung „The netlists match“.

```
@(#)$CDS: LVS version 4.4.2 06/11/98 15:55 (cds10067) $

Like matching is enabled.
Net swapping is enabled.
Creating /home/tech_2/5boettge/AMS/LVS/xref.out file.
Fixed device checking is enabled.
Using terminal names as correspondence points.

   Netlist summary for /home/tech_2/5boettge/AMS/LVS/layout/netlist
     Count
       525          nets
       53           terminals
       434          nmos4
       463          pmos4

   Netlist summary for /home/tech_2/5boettge/AMS/LVS/schematic/netlist
     Count
       525          nets
       53           terminals
       434          nmos4
       463          pmos4


   Terminal correspondence points
         1     a0
         2     a1
         3     a10
         4     a11
         5     a12
         6     a13
         7     a14
         8     a15
         9     a2
         10    a3
         11    a4
         12    a5
         13    a6
         14    a7
         15    a8
         16    a9
         17    b0
         18    b1
         19    b10
         20    b11
         21    b12
         22    b13
         23    b14
         24    b15
         25    b2
         26    b3
         27    b4
         28    b5
         29    b6
         30    b7
         31    b8
         32    b9
         33    c16i
         34    cin
         35    clk
         36    gnd!
         37    s0
         38    s1
         39    s10
         40    s11
         41    s12
         42    s13
         43    s14
         44    s15
         45    s2
         46    s3
         47    s4
         48    s5
         49    s6
         50    s7
         51    s8
         52    s9
         53    vdd!

The netlists match.

                          layout  schematic
                             Instances
       unmatched          0       0
       rewired                    0       0
       size errors        0       0
       pruned                     0       0
       active             897     897
       total              897     897

                              Nets
       unmatched          0       0
       merged                     0       0
       pruned                     0       0
       active             525     525
       total              525     525

                          Terminals
       unmatched          0       0
       matched but
       different type     0       0
       total              53      53
```

Abbildung 33: LVS-Ergebnis CLA-Layout vs. CLA-Schematic

6 Ausblick

Die in den vorstehenden Kapiteln dargelegten Schaltungen sind bewußt modular beschrieben worden. Dies läßt die Freiheit, diese Konzepte in zukünftigen Anwendungen mit mehr oder weniger großen Bitbreiten schnell und anpassungsfähig zu gestalten oder in einen Automatismus einzuarbeiten.

6.1 Erweiterungen der Bitbreite

Die Erweiterung des Carry-Save-Addierers auf eine größere Bitbreite sehr einfach durch Anfügen von Volladdiererstufen und Speichergliedern nach dem Schema von Abbildung 12 möglich.

Etwas schwieriger ist die Erweiterung des Carry-Look-Ahead-Addierers in der hier vorgestellten Form mittels eines DBLC-Trees. Bei Betrachtung der Abbildung 22 fällt auf, daß die Zellen 1A und 2A nur Vereinfachungen der Zellen 1 und 2 sind. Aus dem 16 Bit Schema von Abbildung 21 läßt sich bei Vernachlässigung der Vereinfachungen sehr leicht ein 32 Bit Schema erstellen. Dies ist in Abbildung 34 und 35 mit den einzelnen Zellfunktionen dargestellt.

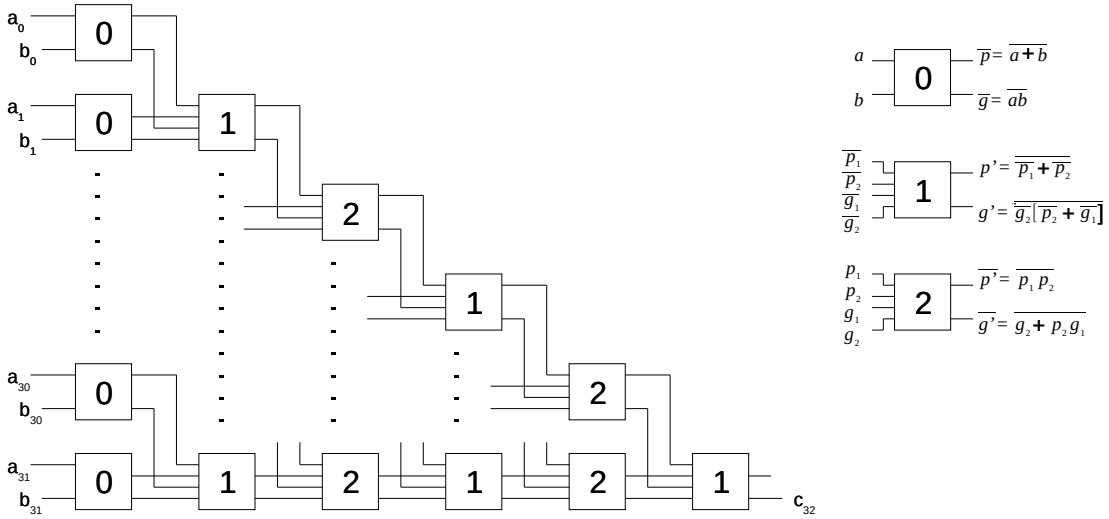


Abbildung 34: 32 Bit Überlauf-Kette

Jeder Überlauf läßt sich durch die o.a. Art und Weise einfach zusammenfügen.

6.2 Einbindung in Blockgeneratoren

Der Vorteil der in Abschnitt 5.2.2 angesprochenen homogenen Konzeption der Zellen zeigt sich bei der Entwicklung eines Automatismusses von CDPD-Schaltungen. Gleichförmigen Zellen sind dafür optimal geeignet. Sind die Zellen für z.B. einen CLA-Addierer erst einmal mittels Full-Custom-Layout erstellt, so läßt sich daraus ein Addierer beliebiger Bitbreite nach Schema in Abb. 35 automatisch erstellen.

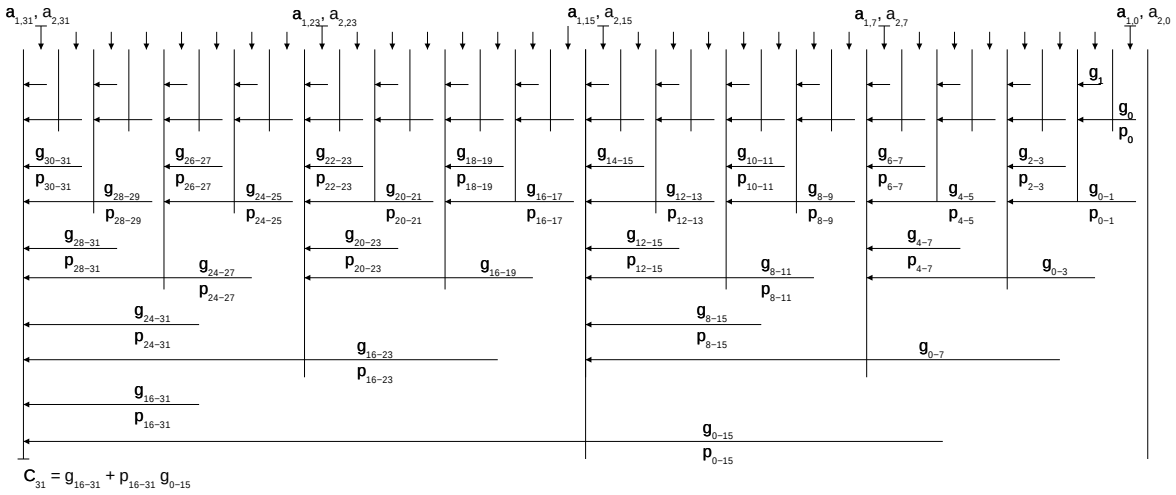


Abbildung 35: BLC-Tree für 32 Bit

Einen ersten Ansatz dafür sieht man schon in Abbildung 32. Alle Zellen haben die gleiche Höhe und Breite - nur eine räumliche Festlegung der Ein- und Ausgangsleitungen fehlt. Gibt man die Lage dieser Leitungen vor, so kann man durch einen Blockgenerator jegliche Schaltungsstruktur generieren.

Abschließend sei angemerkt, daß die in dieser Arbeit dargelegten Addiererschaltungen natürlich nur Anwendungsbeispiele für die CDPD-Technik sind. Es lassen sich auch alle anderen arithmetischen Operationen, wie z.B. Subtraktion oder Multiplikation realisieren.

Literatur

[1] J. Yuan, C. Svensson, „High-Speed CMOS Circuit Technique“, IEEE Journal Of Solid-State Circuits, vol. 24, pp. 62-70, 1989

[2] J. Yuan, „High Speed Silicon Design Course“, Lecture 8, University of Ulm/Germany, 1995

[3] R. Brent, H. T. Kung, „A Regular Layout for Parallel Adders“, IEEE Transactions on Computers, vol. C-31, No. 3, pp. 260-264, 1982

[4] J. Yuan, „High Speed Silicon Design Course“, Lecture 10, University of Ulm/Germany, 1995

[5] J. Yuan, C. Svensson, „A 700-MHz Pipelined Accumulator [...]“, IEEE Journal Of Solid-State Circuits, vol. 28, pp. 878-885, 1993

[6] J. Yuan, M. Afghahi, „Double Edge-Triggered D-Flip-Flops For High-Speed CMOS Circuits“, IEEE Journal Of Solid-State Circuits, vol. 26, pp. 1168-1170, 1991

[7] J. Yuan, I. Karlsson, C. Svensson, „A TSPC Dynamic CMOS Circuit Technique“, IEEE Journal Of Solid-State Circuits, vol. SC-22, No. 5, pp. 899-901, 1987

[8] W. Rosenstiel, R. Camposano, „Rechnergestützter Entwurf hochintegrierter MOS-Schaltungen“, Springer-Verlag, 1989

Index