

# Das Addieren von Dualzahlen

Von LOTHAR DILCHER

Wer als Gelegenheits-Informatiker sich in Kenntnis setzen will, wie ein moderner Computer addiert, wird überrascht sein, in neue, unbekannte Welten zu tauchen wie einst Alice in das Wunderland. (Das heutige Wunderland befindet sich selbstredend im Internet.)

Neben dem

seriellen Addierer, den alle kennen, dem

genialen von Neumann Addierer, dem

„Klapper“-Addierer (carry ripple adder), dem

carry select adder sowie dem

carry save adder und dem

Normalform-Addierer

gibt es nun etwa ein Dutzend Alternativen, deren Mehrzahl dem den Übertrag vorhersagenden Prinzip angehören, dem

carry preview adder.

Die Kombination unterschiedlicher Eigenschaften vergrößert wiederum die Addierer-Welt noch mehr.

Einige Möglichkeiten, die Übertrags-Vorausschau (Carry Preview) auszuführen, soll im Folgenden erörtert werden.

Der ganze Vorausschau-Aufwand dient dazu – wer hätte es nicht gehnt – den Additionsprozess zu beschleunigen. Man zählt die Zeit für die Addition zweier Dualzahlen entweder in Systemtaktten oder in Gatterlaufzeiten. Die Zuordnung zwischen beiden Verzugsmoden ist technologisch bedingt und wird selten beziffert ebenso wie die absolute Größe des Verzugs. Die Relativ-Werte dieser Größen (Takte, Gatterlaufzeiten) bewegen sich extremal zwischen: Anzahl der Stellen (N), dem  $\log_2(N)$  und dem Wert Eins.

## Einige Vereinbarungen:

Wir passen uns dem Schrifttum an und leihen die Booleschen Operatoren z.T. von der Arithmetik aus:

$$A \text{ UND } B \rightarrow A \cdot B \text{ oder } AB$$

$$A \text{ ODER } B \rightarrow A + B$$

$$\text{NICHT } A \rightarrow \bar{A}$$

$$A \text{ XOR } B = \bar{A}B + A\bar{B} = A \oplus B$$

### Zusammenhänge:

Übertrag (C) und Summe (S) errechnen sich für die Stelle einer Dualzahl (N=Stellenzahl) aus den Summanden  $A_i$  und  $B_i$  mit folgenden Booleschen Funktionen (Index EIN für EINGang und Index AUS für AUSgang):

$$\begin{aligned} S_i &= A_i \bar{B}_i \bar{C}_{i\text{EIN}} + \bar{A}_i B_i \bar{C}_{i\text{EIN}} + \bar{A}_i \bar{B}_i C_{i\text{EIN}} + A_i B_i C_{i\text{EIN}} \\ C_{i\text{AUS}} &= A_i B_i + (A_i + B_i) \cdot C_{i\text{EIN}} \\ C_{i\text{EIN}} &= C_{i-1\text{AUS}} \quad \text{oder} \quad C_{i+1\text{EIN}} = C_{i\text{AUS}} \end{aligned}$$

Es ist also immer zwischen  $C_{i\text{EIN}}$  und  $C_{i\text{AUS}}$  zu unterscheiden. Der an der Stelle  $i$  gebildete Übertrag geht in die Summen-/Übertrags-Bildung der Stelle  $i+1$  ein.

Für die Bestimmung von C werden meist folgende Zusatz-Signale gebildet:

$$\begin{aligned} G_i &= A_i B_i \text{ Erzeugung (engl. generate)} \\ K_i &= \bar{A}_i \bar{B}_i \text{ Vernichtung (engl. kill)} \\ \bar{K}_i &= A_i + B_i \\ P_i &= A_i \text{ XOR } B_i \text{ Durchschaltung (engl. propagate), mod 2 – Addition} \\ P \text{ und } \bar{K} &\text{ ergeben sich im vorliegenden Zusammenhang als gleichwertig} \end{aligned}$$

Summe und Übertrag können auch mit diesen abgeleiteten Größen ermittelt werden:

$$\begin{aligned} S_i &= G_i C_{i\text{EIN}} + K_i C_{i\text{EIN}} + \bar{G}_i \bar{K}_i \bar{C}_{i\text{EIN}} \\ C_{i\text{AUS}} &= G_i + \bar{K}_i \cdot C_{i\text{EIN}} \\ &= G_i + P_i \cdot C_{i\text{EIN}} \end{aligned}$$

### Summe und Übertrag

Additionen werden üblicherweise seriell durchgeführt-beginnend von rechts mit der niedrigsten Stelle bis nach links zur höchsten Stelle. Unter Berücksichtigung des Übertrags werden auf diese Weise Summe und Übertrag rekursiv entwickelt. Die Berechnung der Daten einer Stelle haben immer die Werte der vorhergehenden Stelle als Eingangsdaten. Diese Vorgehensweise entspricht den oben angegebenen rekursiven Formeln. Der klassische serielle Addierer und der Klapper-Addierer (Ripple Carry Adder) sind entsprechend dieser Vorgehensweise konstruiert.

In Abb. 1 werden einige Boolesche Variable als Ergebnisse der Addition der Zufallszahlen A und B gezeigt

Der serielle Addierer benötigt ca. N Systemtakte zur Erstellung der Summe. John von Neumann liegt im ungünstigen Fall ebenfalls bei N Systemtakt, im statistischen Mittel jedoch laut Literatur bei ca.  $\frac{1}{2} \log_2(N)$  Systemtakt. Der „Klapper“-Addierer braucht im ungünstigen Fall ca. N Gatterlaufzeiten

hierzu, wobei die Relation der Gatterlaufzeit zum Systemtakt von der verwendeten Technologie abhängt. In einen Systemtakt müssen immer einige Gatterlaufzeiten passen.

Jedenfalls sind die Addierer-Entwickler mit obigen Lösungen bei weitem nicht zufrieden und versuchen durch Vorausschau und andere Maßnahmen mehr Tempo zu machen.

| N     | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------|---|---|---|---|---|---|---|---|
| A     | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| B     | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| C     | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| Summe | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| P     | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| G     | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| K     | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |

Abb. 1 Boolesche Variable als Addier-Ergebnis von A und B (8 Stellen)

Wem das rein seriell-rekursive Vorgehen zu gemächlich ist, kann sich darauf besinnen, dass durch die Vorgabe der Summanden die Gesamt-Summe bereits bestimmt ist und damit auch der Übertrag jeder Stelle. Bei Kenntnis der Summanden besteht also die Möglichkeit der Vorausberechnung der Überträge und einer nachfolgenden jedoch gleichzeitigen Berechnung der Endsummen.

Der Übertrag  $C_i$  ist hiermit für alle  $A_i$  und  $B_i$  festgelegt. Von rechts nach links gehend, d.h. von der niedrigsten Stelle (LSB-Least Significant Bit) bis zur höchsten Stelle (MSB-Most Significant Bit) entsteht ein Übertrag  **$C=1$  wenn  $G=1$**  wird, und es bleibt bei  **$C=1$  solange  $K=0$**  ist. Er verschwindet mit  **$C=0$ , sobald  $K=1$**  wird. Solange dann  **$G=0$  ist, bleibt  $C=0$** . Erst wenn wieder  **$G=1$  wird, springt C auf den Wert 1**. G und K können nicht an der gleichen Stelle die Zahl 1 annehmen aber sehr wohl die Zahl 0.

Diese einfache Gesetzmäßigkeit wird besonders in Abb. 2 deutlich

Für eine Vorausschau des Übertrags ist diese Relation zu berechnen d.h. in eine logische Schaltung zu bringen. Der Addierer kann mit Kenntnis der Überträge die Summenbildung für alle Stellen der Summanden gleichzeitig in einem Systemtakt durchführen. Die Zeitdauer einer Addition wird bei dem Vorausschau-Verfahren fast nur noch durch die Zeiten zur Berechnung der Überträge bestimmt.

Der ungünstigste Fall (worst case) in Bezug auf die Laufzeit des Übertrags liegt vor, wenn die Stelle 0 einen Übertrag bildet, der dann seriell bis zur höchsten Stelle durchgeschoben wird, also z.B.:

A 0111111111111111

B 0000000000000001

C 0111111111111110

S 1000000000000000 Hier sind es ca. N Laufzeiten.

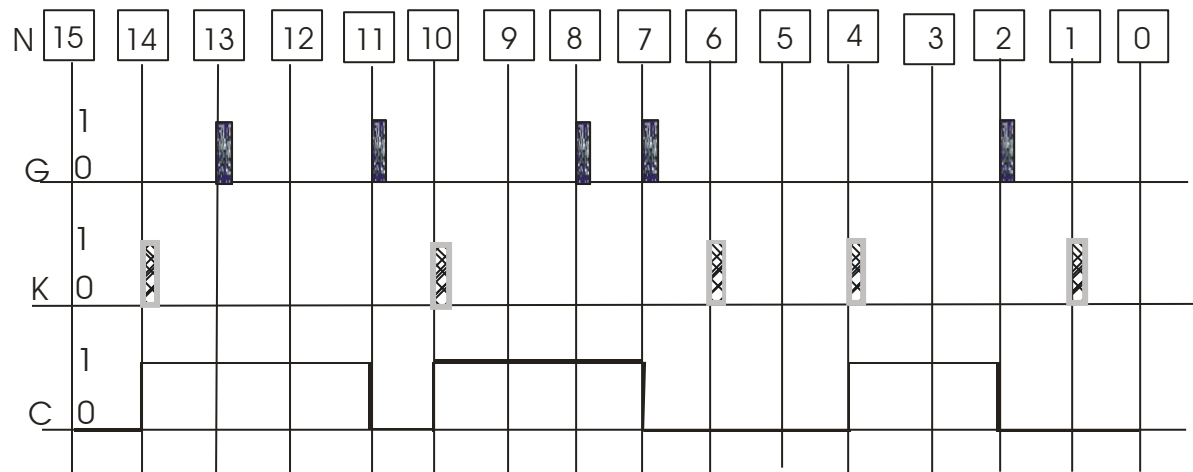


Abb. 2 Verlauf des Übertrags C in Abhängigkeit von G und K (letztere beliebig gewählt, 16 Stellen)

Die kürzesten Verzugszeiten ergeben sich zumindest theoretisch, wenn alle Überträge gleichzeitig ermittelt werden. Aus den Größen G und K (oder P) lassen sich z.B. Gleichungen für  $C_{AUS}$  für jede Stelle aufstellen. Es kann dann allerdings keine Rekursion mehr stattfinden, da diese ja aus einem Nacheinander mit Abwarten des vorherigen Ergebnisses bestünde. Das gleichzeitige Berechnen erfordert die gleichzeitige Bereitstellung der Eingangsgrößen der Rechnungen, also alle A und B bzw. G und K (oder P). Schaltungstechnisch bedeutet dies, dass die Ausgänge der zugehörigen Chips mit der Vielfältigung entsprechend hoch belastet werden. Dies führt zu einem ernsten „Fan-Out“-Problem.

Weiterhin ergeben sich durch die Parallelberechnung Anzahlen für die Eingangsgrößen der Chips, die sich ebenfalls im Bereich der Stellenzahl des Addierers befinden. Es liegt also auch ein „Fan-In“-Problem vor. Das Berechnungsbeispiel für Überträge im nächsten Abschnitt (Tabelle der Überträge) macht die Hardware-Explosion deutlich, die sich mit größerer Stellenzahl einstellt. Die Verkürzung der Laufzeit wird also mit Hardware bezahlt.

Die extremen Möglichkeiten zur Gestaltung der Vorausschau-Hardware sind nun die Reihenschaltung (s. Abb. 3) oder die Parallelschaltung (s. Abb. 4). Die beiden Extreme sind so zu charakterisieren, dass die serielle Variante ein Minimum an Bauteilen braucht aber viele Laufzeiten und die andere viele Baugruppen und wenig Laufzeit. Nun lassen sich die Varianten auch beliebig und nach Geschmack kombinieren.

Die Überträge müssen natürlich von jeder Anordnung oder Methode erfüllt werden. Sie lauten für 8 bzw. 16 Stellen (die Indizes  $i=0, 1, 2, \dots, N-1$  bezeichnen die zugehörige Stelle):

### Tabelle der Überträge $C_{AUS}$

$$C_0 = G_0$$

$$C_1 = G_1 + G_0 \bar{K}_1$$

$$C_2 = G_2 + G_1 \bar{K}_2 + G_0 \bar{K}_1 \bar{K}_2$$

$$C_3 = G_3 + G_2 \bar{K}_3 + G_1 \bar{K}_2 \bar{K}_3 + G_0 \bar{K}_1 \bar{K}_2 \bar{K}_3$$

$$C_4 = G_4 + G_3 \bar{K}_4 + G_2 \bar{K}_3 \bar{K}_4 + G_1 \bar{K}_2 \bar{K}_3 \bar{K}_4 + G_0 \bar{K}_1 \bar{K}_2 \bar{K}_3 \bar{K}_4$$

$$C_5 = G_5 + G_4 \bar{K}_5 + G_3 \bar{K}_4 \bar{K}_5 + G_2 \bar{K}_3 \bar{K}_4 \bar{K}_5 + G_1 \bar{K}_2 \bar{K}_3 \bar{K}_4 \bar{K}_5 + G_0 \bar{K}_1 \bar{K}_2 \bar{K}_3 \bar{K}_4 \bar{K}_5$$

$$C_6 = G_6 + G_5 \bar{K}_6 + G_4 \bar{K}_5 \bar{K}_6 + G_3 \bar{K}_4 \bar{K}_5 \bar{K}_6 + G_2 \bar{K}_3 \bar{K}_4 \bar{K}_5 \bar{K}_6 + G_1 \bar{K}_2 \bar{K}_3 \bar{K}_4 \bar{K}_5 \bar{K}_6 + G_0 \bar{K}_1 \bar{K}_2 \bar{K}_3 \bar{K}_4 \bar{K}_5 \bar{K}_6$$

$$C_7 = G_7 + G_6 \bar{K}_7 + G_5 \bar{K}_6 \bar{K}_7 + G_4 \bar{K}_5 \bar{K}_6 \bar{K}_7 + G_3 \bar{K}_4 \bar{K}_5 \bar{K}_6 \bar{K}_7 + G_2 \bar{K}_3 \bar{K}_4 \bar{K}_5 \bar{K}_6 \bar{K}_7 + G_1 \bar{K}_2 \bar{K}_3 \bar{K}_4 \bar{K}_5 \bar{K}_6 \bar{K}_7 + G_0 \bar{K}_1 \bar{K}_2 \bar{K}_3 \bar{K}_4 \bar{K}_5 \bar{K}_6 \bar{K}_7$$

- 
- 
- 

$$C_{15} = G_{15} + G_{14} \bar{K}_{15} + G_{13} \bar{K}_{14} \bar{K}_{15} + G_{12} \bar{K}_{13} \bar{K}_{14} \bar{K}_{15} + G_{11} \bar{K}_{12} \bar{K}_{13} \bar{K}_{14} \bar{K}_{15} + G_{10} \bar{K}_{11} \bar{K}_{12} \bar{K}_{13} \bar{K}_{14} \bar{K}_{15} + G_9 \bar{K}_{10} \bar{K}_{11} \bar{K}_{12} \bar{K}_{13} \bar{K}_{14} \bar{K}_{15} + G_8 \bar{K}_9 \bar{K}_{10} \bar{K}_{11} \bar{K}_{12} \bar{K}_{13} \bar{K}_{14} \bar{K}_{15} + G_7 \bar{K}_8 \bar{K}_9 \bar{K}_{10} \bar{K}_{11} \bar{K}_{12} \bar{K}_{13} \bar{K}_{14} \bar{K}_{15} + G_6 \bar{K}_7 \bar{K}_8 \bar{K}_9 \bar{K}_{10} \bar{K}_{11} \bar{K}_{12} \bar{K}_{13} \bar{K}_{14} \bar{K}_{15} + G_5 \bar{K}_6 \bar{K}_7 \bar{K}_8 \bar{K}_9 \bar{K}_{10} \bar{K}_{11} \bar{K}_{12} \bar{K}_{13} \bar{K}_{14} \bar{K}_{15} + G_4 \bar{K}_5 \bar{K}_6 \bar{K}_7 \bar{K}_8 \bar{K}_9 \bar{K}_{10} \bar{K}_{11} \bar{K}_{12} \bar{K}_{13} \bar{K}_{14} \bar{K}_{15} + G_3 \bar{K}_4 \bar{K}_5 \bar{K}_6 \bar{K}_7 \bar{K}_8 \bar{K}_9 \bar{K}_{10} \bar{K}_{11} \bar{K}_{12} \bar{K}_{13} \bar{K}_{14} \bar{K}_{15} + G_2 \bar{K}_3 \bar{K}_4 \bar{K}_5 \bar{K}_6 \bar{K}_7 \bar{K}_8 \bar{K}_9 \bar{K}_{10} \bar{K}_{11} \bar{K}_{12} \bar{K}_{13} \bar{K}_{14} \bar{K}_{15} + G_1 \bar{K}_2 \bar{K}_3 \bar{K}_4 \bar{K}_5 \bar{K}_6 \bar{K}_7 \bar{K}_8 \bar{K}_9 \bar{K}_{10} \bar{K}_{11} \bar{K}_{12} \bar{K}_{13} \bar{K}_{14} \bar{K}_{15} + G_0 \bar{K}_1 \bar{K}_2 \bar{K}_3 \bar{K}_4 \bar{K}_5 \bar{K}_6 \bar{K}_7 \bar{K}_8 \bar{K}_9 \bar{K}_{10} \bar{K}_{11} \bar{K}_{12} \bar{K}_{13} \bar{K}_{14} \bar{K}_{15}$$

### Reihenschaltung

Das Übertrag-Signal C entwickelt sich von rechts nach links mit  $i=0, 1, 2, \dots, N-1$  aus dem vorhergehenden Übertrag:

$$C_{i+1EIN} = C_{iAUS} = G_i + C_{iEIN} \bar{K}_i$$

Man bedient sich obiger Rekursionsformel zur Berechnung des Übertrags. Die Gatterlaufzeiten akkumulieren – es gibt also kaum einen Gewinn gegenüber dem „Klapper“-Addierer. Allerdings bleiben die gesamte Anzahl der Gatter bei N und die Zahl der Ein-/Aus-Gänge bei einem pro Variable.

Es sind Also keine „Fan-In“ oder „Fan-Out“ Probleme zu erwarten.

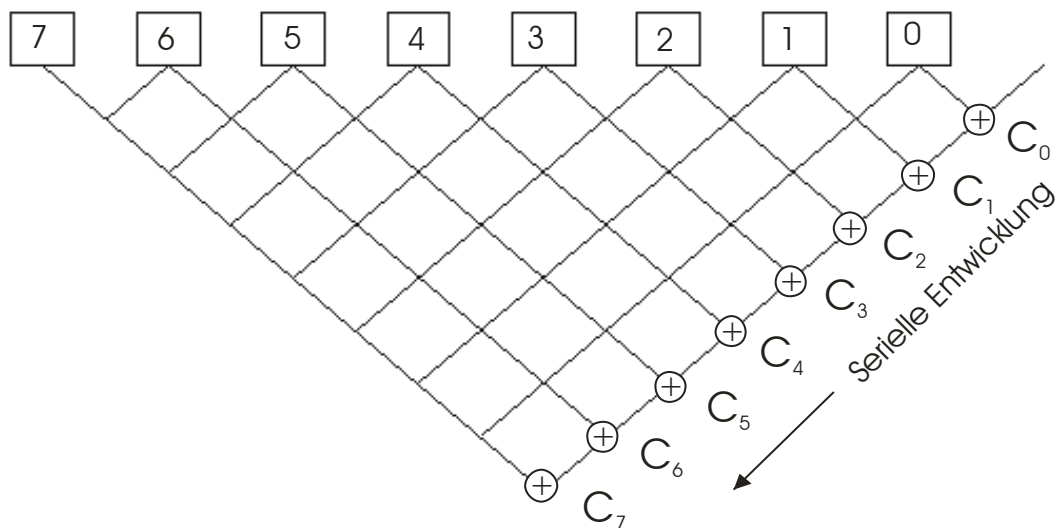


Abb. 3. Serielle Berechnung des Übertrags (Reihenschaltung, 8 Stellen).

### Parallelschaltung

Im Fall der parallelen Übertrags-Ermittlung wiederholt man die explizite Bestimmungsgleichung für  $C$  für jede Stelle. Es sind dann zwar wenig Gatter nacheinander zu durchlaufen, es stören jedoch offenbar die Gattergrößen mit bis zu ca.  $N$  Eingängen und auch die Belastung der Gatterausgänge mit ebenso vielen Abzweigen.

Die Elektronik muss diesen Anforderungen mit Hintereinanderschaltung von Gattern begegnen (Kaskadierung von Schaltern), was ebenfalls zu einer Erhöhung der Laufzeiten führt. Angenommen, es wäre möglich Gatter mit acht Eingängen und acht Ausgängen zu bauen, so müssten für einen 64-bit Addierer auch zwei elektronische Schalterebenen hintereinandergeschaltet werden. Es kann also aus technologischen Gründen die Laufzeit eins für die Parallelschaltung nicht perfekt verwirklicht werden.

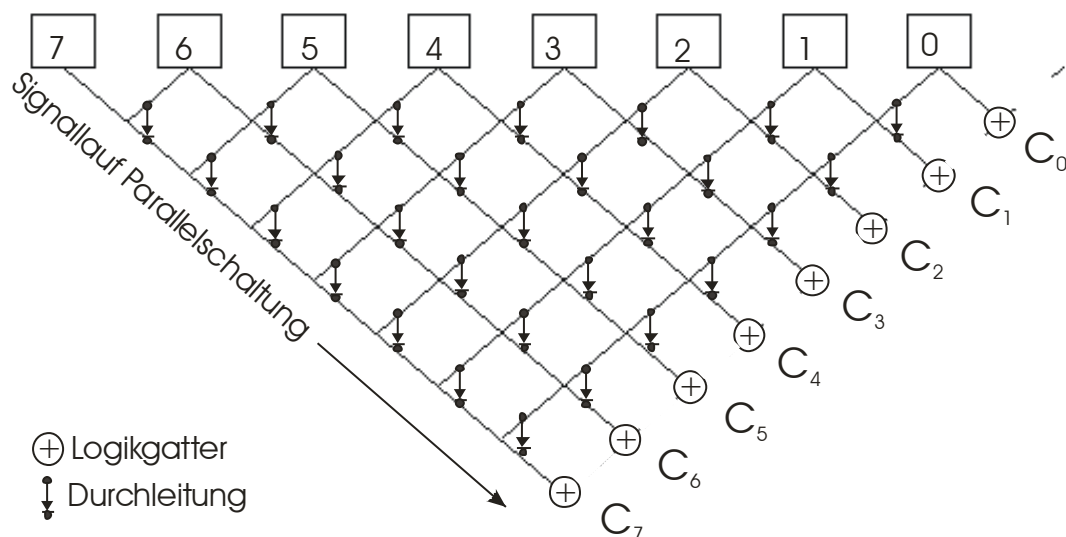


Abb. 4 Parallele Ermittlung der Überträge (8 Stellen).

### Parallel-Seriell

Die nächstliegende Übung könnte sein, eine seriell-parallele Kombination aufzubauen. In Abb. 5 soll ein 16stelliger Addierer in Bereiche zu je vier Stellen in Reihe und diese dann parallel geschaltet werden. Aus maximal 16 Gatterlaufzeiten würden sich dann 4 Laufzeiten pro Reihenelement und anschließend nochmal 1- 4 Laufzeiten zum Zusammenschalten der Reihenelemente, also maximal 5- 8 Laufzeiten aufaddieren. Man spart also vielleicht die Hälfte der Laufzeiten (oder etwas mehr) spendiert aber wie erwartet viele Bauteile.

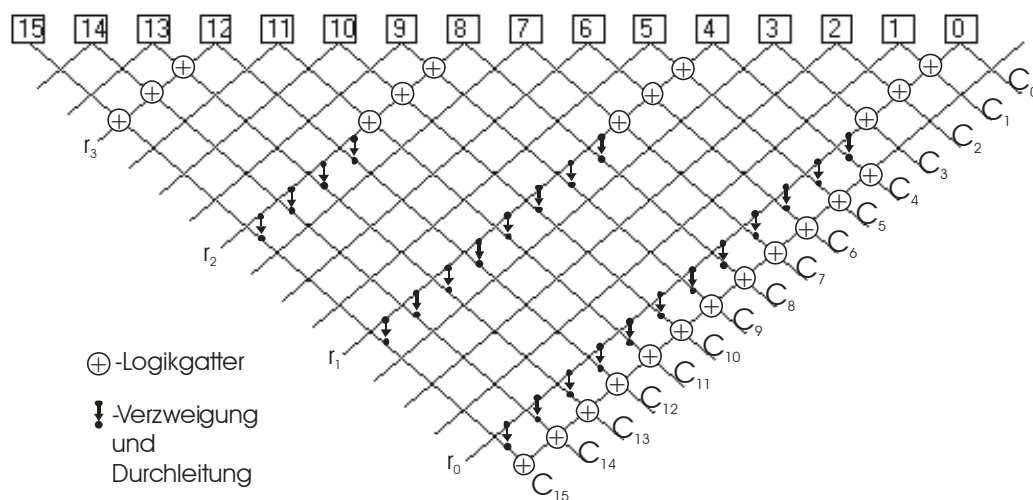


Abb. 5 Parallel-Serielle Anordnung (16 Stellen, N=16)

Die Booleschen Gleichungen müssen die identischen Übertragungswerte wie obige Tabelle ergeben, nur der Weg dahin ist unterschiedlich:

$$\begin{aligned}
 r_0 &= G_3 + G_2 \bar{K}_3 + G_1 \bar{K}_2 \bar{K} + G_0 \bar{K}_1 \bar{K}_2 \bar{K}_3 \\
 r_1 &= G_7 + G_6 \bar{K}_7 + G_5 \bar{K}_6 \bar{K}_7 + G_4 \bar{K}_5 \bar{K}_6 \bar{K}_7 \\
 r_2 &= G_{11} + G_{10} \bar{K}_{11} + G_9 \bar{K}_{10} \bar{K}_{11} + G_8 \bar{K}_9 \bar{K}_{10} \bar{K}_{11} \\
 r_3 &= G_{15} + G_{14} \bar{K}_{15} + G_{13} \bar{K}_{14} \bar{K}_{15} + G_{12} \bar{K}_{13} \bar{K}_{14} \bar{K}_{15} \\
 C_3 &= r_0 \\
 C_4 &= G_4 + r_0 \bar{K}_4 \\
 C_5 &= G_5 + G_4 \bar{K}_5 + r_0 \bar{K}_4 \bar{K}_5 \\
 C_6 &= G_6 + G_5 \bar{K}_6 + G_4 \bar{K}_5 \bar{K}_6 + r_0 \bar{K}_4 \bar{K}_5 \bar{K}_6 \\
 C_7 &= G_7 + G_6 \bar{K}_7 + G_5 \bar{K}_6 \bar{K}_7 + G_4 \bar{K}_5 \bar{K}_6 \bar{K}_7 + r_0 \bar{K}_4 \bar{K}_5 \bar{K}_6 \bar{K}_7 \\
 &\text{etc.}
 \end{aligned}$$

Mit  $r_0$  bis  $r_3$  seien die seriellen Kombinationen bezeichnet. Die Schaltung ist sicher noch optimierbar, indem einzelne Logikgatter und Durchleitungen gegeneinander ausgetauscht werden. Auf diese Weise könnten Fan-Out und Fan-In Probleme abgeglichen werden.

## Der ${}_2\text{LOG}(N)$

Wie kann man noch besser werden? Man begibt sich ins logarithmische Land, d.h. wir fassen die Terme der Booleschen Funktionen (Tabelle s. oben) für den Übertrag je paarweise zusammen:

$$G_{i,i+1} = G_i \bar{K}_i + G_{i+1}$$

$$\bar{K}_{i,i+1} = \bar{K}_i \bar{K}_{i+1}$$

Mit der Kombination von 2 Termen wird immer eine Reduktion der Variablen um den Faktor 2 bewirkt, gleichgültig ob sie direkt der Gleichung für den Übertrag entnommen sind oder schon ein- oder mehrmals kombiniert wurden. Es wird ein Entscheidungsbaum aufgebaut, dessen umgekehrte Version auch oft mit „node explosion“ bezeichnet wird. Man bewegt sich also auf dem Maßstab des  ${}_2\text{Log}$ . Die Anzahl der Berechnungsschritte für die Überträge können also im Grundsatz auf den  ${}_2\text{Log}$  der Stellenzahl reduziert werden, also bei z.B. 64 Stellen auf 6 Schritte (Stufen).

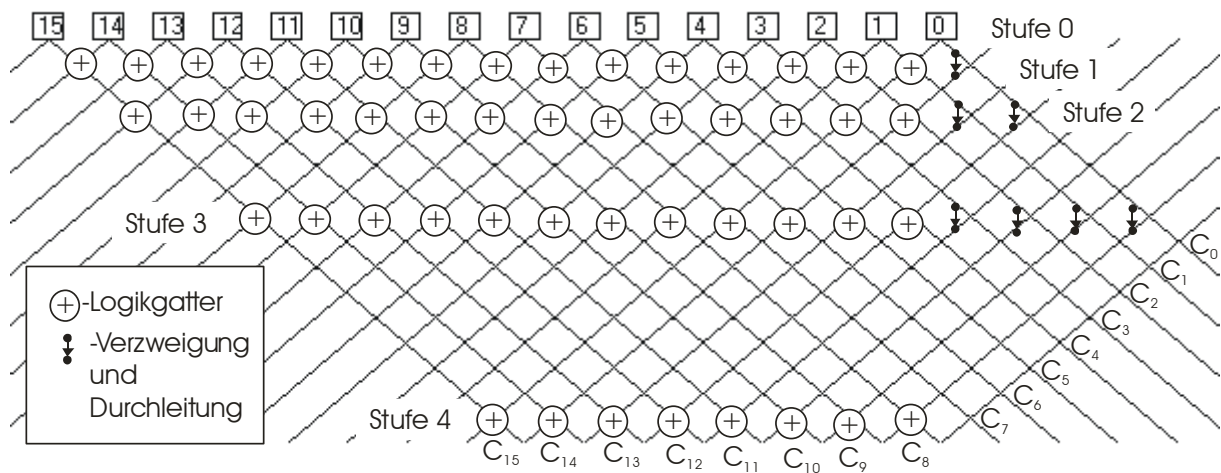


Abb.6 Anordnung nach **Kogge-Stone** (N=16)

(s.a. "Design of a 128-bit Kogge-Stone Low Power..." by **P. Annapurna Bai et al.**, August 2013)

Ich zitiere **robey** (How to add numbers; part 1 and 2, Internet 2012/2011): "Any time we do a recursive combination like this we're in log-scale country. This is the country where cowboys ride horses that go twice as far with each hoofstep. But seriously, it means we can compute the final carry in an 8-bit adder in 3 steps"

Kogge-Stone sind offenbar als erste diesen Weg konsequent gegangen (1973, IEEE-Veröffentlichung). Sie sind jedenfalls in der Literatur immer noch erster Sieger, obwohl sich eine starke Konkurrenz bereithält (s.a. **Muhammad Shoaib Bin Altaf**: Parallel Prefix Adders, Internet)

Nochmals **robey**: "In 1973, probably while listening to a Yes or King Crimson album, Kogge and Stone came up with the idea of parallel prefix computation Their paper was a description of how to generalize recursive linear functions into forms that can be quickly combined in an arbitrary order, but they were being coy in a way that math people do. What they are really getting at is that these G and P values can be combined before being used."

## Einige Boolesche Ausdrücke aus den Kombinationsstufen

Stufe 0:

$$G_0, G_1, G_2, \dots, G_{15}$$

$$\bar{K}_1, \bar{K}_2, \dots, \bar{K}_{15}$$

$\bar{K}_0$  würde nur bei einem externen  $C_{EIN}$  verwendet

Stufe 1:

$$G_{(0,1)} = G_0 \bar{K}_1 + G_1$$

$$G_{(1,2)} = G_1 \bar{K}_2 + G_2$$

⋮

$$G_{(14,15)} = G_{14} \bar{K}_{15} + G_{15}$$

$$\bar{K}_{(1,2)} = \bar{K}_1 \bar{K}_2$$

⋮

$$\bar{K}_{(14,15)} = \bar{K}_{14} \bar{K}_{15}$$

Stufe 2:

$$G_{((0),(1,2))} = G_0 \bar{K}_{(1,2)} + G_{(1,2)}$$

$$G_{((01),(2,3))} = G_{(0,1)} \bar{K}_{(2,3)} + G_{(2,3)}$$

$$G_{((1,2),(3,4))} = G_{(1,2)} \bar{K}_{(3,4)} + G_{(3,4)}$$

⋮

$$G_{((12,13),(14,15))} = G_{(12,13)} \bar{K}_{(14,15)} + G_{(14,15)}$$

⋮

$$\bar{K}_{((1,2),(3,4))} = \bar{K}_{(1,2)} \bar{K}_{(3,4)}$$

Stufe 3:

$$G_{((0),(1,2),(3,4))} = G_0 \bar{K}_{((1,2),(3,4))} + G_{((1,2),(3,4))}$$

$$G_{((0,1),(2,3),(4,5))} = G_{(0,1)} \bar{K}_{((2,3),(4,5))} + G_{((2,3),(4,5))}$$

$$G_{((8,9),(10,11),(12,13),(14,15))} = G_{((8,9),(10,11))} \bar{K}_{((10,11),(12,13),(14,15))} + G_{((12,13),(14,15))}$$

: etc.

Jedes Carry-Signal hat seinen eigenen Entscheidungsbaum. Ab Stufe 0 hat der Rechenpfad einer C-Berechnung mit einem geradzahligen Index keinen gemeinsamen Punkt mit dem einer Berechnung des Carry-Signals mit einem ungeradzahligen Index. Der Entscheidungsbaum für ein Carry-Signal enthält also nur die Hälfte aller Kombinationen der Stufe 1. Im Addierer-Entwurf von **Brent-Kung** findet

man einen Entscheidungsbaum für das höchste C, das von der Stufe 1 nur die Hälfte der Kombinationen benutzt.

In der Literatur wird die komplizierte Verdrahtung beklagt. In dem oben zitierten Artikel von **Annapurna** wird z.B. erwähnt, dass die Verbindungen nicht vollständig aufgeführt wurden, wegen „cumsiness“. Als schwierigen Punkt der Verdrahtung könnte man sich auch die Summenbildung vorstellen, da z.B. für die Summe  $S_{10}$  die Größen  $A_{10}$ ,  $B_{10}$  und  $C_{9AUS}$  bzw.  $G_{10}$ ,  $K_{10}$  und  $C_{9AUS}$  benötigt werden.

***Die Einfachheit der Zuordnung von C zu G und K, wie sie in Abb. 2 gezeigt wird, lässt sich mit den angewendeten Methoden nicht widerspiegeln. Vielleicht gibt es einen Genieblitz, der die ganze Mühsal auf wenige Operationen verdampft.***

Zum Schluss noch einmal Zitat **robey**:

**„Adding numbers: Proof that humans can make anything complicated, if they try hard enough.“**

Und unsere Schlussfolgerung:

**Es muss nicht immer Kogge-Stone sein, eine geeignete Kombination von seriell und parallel kann durchaus ein angepasstes Optimum erzeugen.**