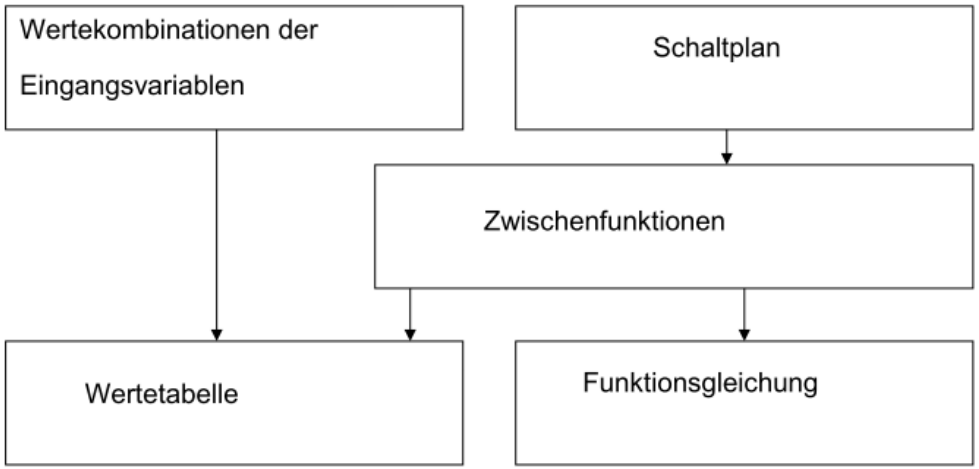
OSTBAYERISCHE
TECHNISCHE HOCHSCHULE
REGENSBURG

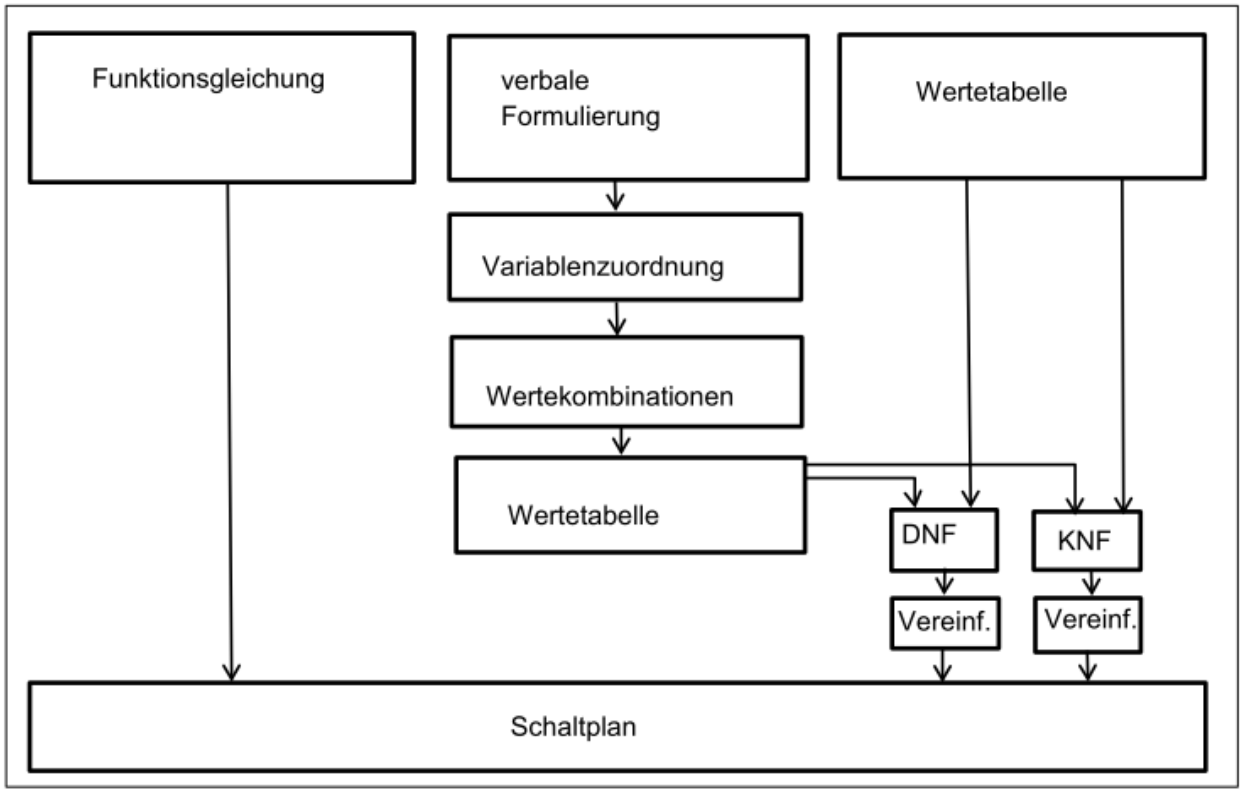
Kombinatorische Logik (comblogic)

Übersicht Zusammenhänge Schaltungsentwurf I/III

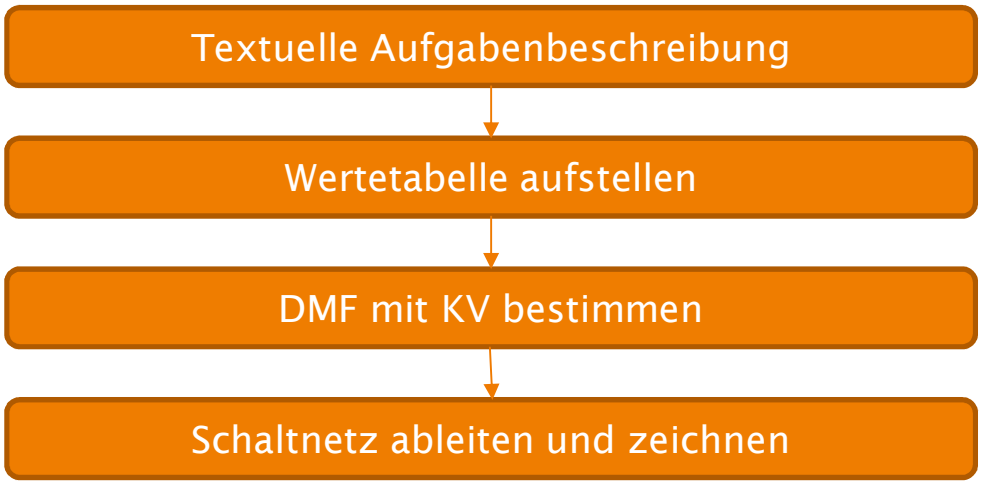
- Darstellungsmöglichkeiten:
  - Funktionsgleichung
  - Wertetabelle
  - KV-Diagramm
  - Schaltplan



Übersicht Zusammenhänge Schaltungsentwurf II/III



Übersicht Zusammenhänge Schaltungsentwurf III/III



Schaltnetze - Code-Umsetzer - Gray-Code decoder I/II

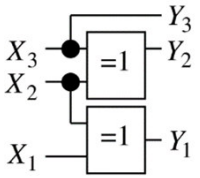
- Der Gray-Code decoder wandelt Binärcoder in Graycode um
- (bin -> Gray-Code)
- Beim Gray-Code unterscheiden sich benachbarte Codeworte nur in einem einzigen Bit
- Diese Eigenschaft wird verwendet um Übertragungsfehler bei sich kontinuierlich ändernden digitalen Signalen auf mehradrigen Leitungen zu verringern
- Eine Weitere Anwendung ist die Bestimmung der absoluten Position einer Scheibe oder Leiste[, die mit schwarzen und weißen Balken markiert ist, die mit Lichtschranken oder anderen Sensoren abgetastet wird.]
  - Diese Position wird dann zur Winkel- oder Drehgeschwindigkeitsmessung verwendet.

Schaltnetze - Code-Umsetzer - Gray-Code decoder II/II

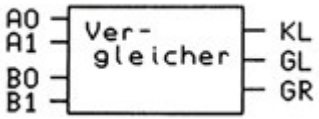
- Beispiel: 3bit Binär/Gray-Code Umsetzer (Gray-Code decoder)

X_3	X_2	X_1	Y_3	Y_2	Y_1
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

$$Y_3 = X_3$$
$$Y_2 = X_3 \oplus X_2$$
$$Y_1 = X_2 \oplus X_1$$

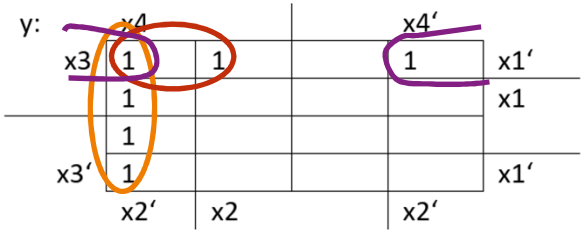


Schaltnetze - Komparator I/II



A1	A0	B1	B0	GR
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Mapping: a1=x4; a0=x3;
b1=x2; b0=x1; gr=y

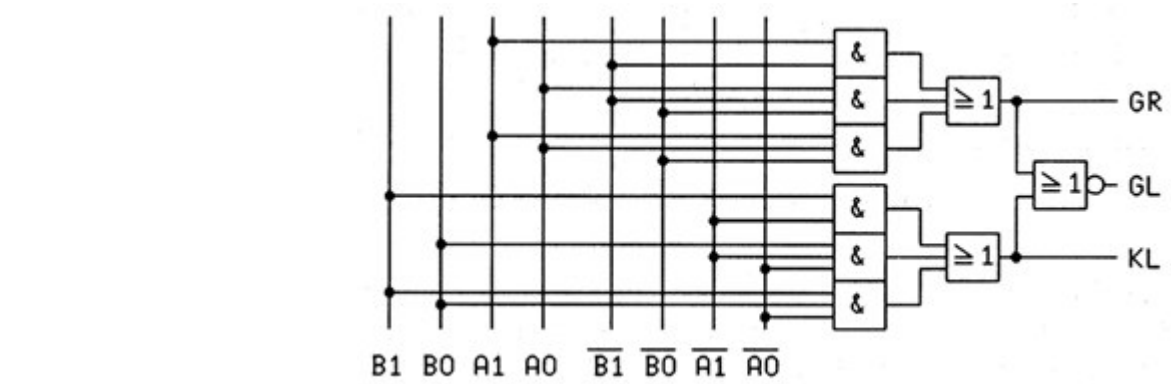


x4x2‘
x4x3x1‘
x3x2‘x1‘

-> y= x4x2‘+ x4x3x1‘+ x3x2‘x1‘
->gr=a1b1‘+ a1a0b0‘+ a0b1‘b0‘
-kl: nehme gr und tausche ai durch bi
kl=b1a1‘+ b1b0a0‘+ b0a1‘a0‘
-gl: gl=(gr+kl)‘

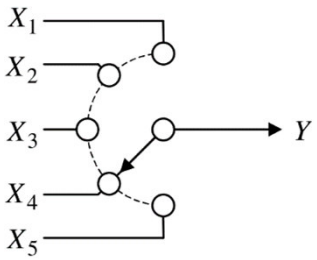
Schaltnetze - Komparator II/II

-gr=a1b1‘+ a1a0b0‘+ a0b1‘b0‘
-kl=b1a1‘+ b1b0a0‘+ b0a1‘a0‘
-gl=(gr+kl)‘



Schaltnetze - Multiplexer - Prinzip

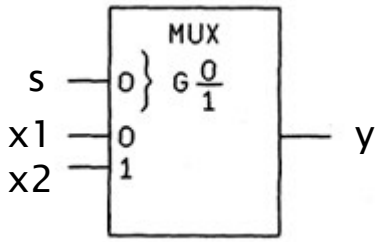
- Allgemeine Funktionsweise eines Multiplexers:
  - Ein Multiplexer schaltet einen Eingang in Abhängigkeit des Steuereingangs auf den Ausgang



  - Ein Multiplexer ist also ein gesteuerter Umschalter
- Multiplexer gibt es in verschiedenen Größen, man spricht dann von N:1 Multiplexern

Schaltnetze - Multiplexer 2:1 I/II - Übung

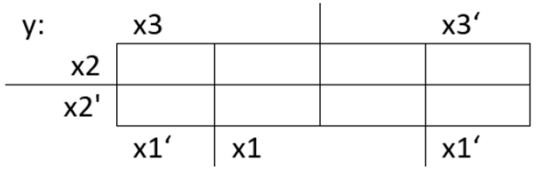
- Beispiel: mux 2:1



Nr	s	x2	x1	y
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

Minimierung

Mapping: s=x3; x2=x2; x1=x1

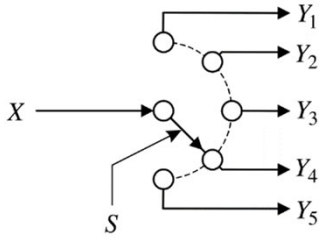


-> y=

Schaltplan

Schaltnetze - Demultiplexer - Prinzip

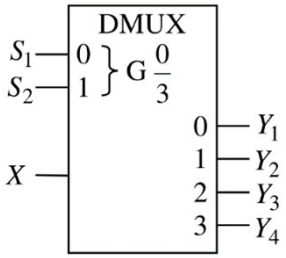
- Allgemeine Funktionsweise eines Demultiplexers:
  - Ein Multiplexer schaltet den Eingang in Abhängigkeit des Steuereingangs auf einen Ausgang



  - Ein Demultiplexer schaltet den Eingang in Abhängigkeit des Steuereingangs auf einen der Ausgänge
  - Ein Demultiplexer ist also ein gesteuerter Umschalter für den Ausgang
- Demultiplexer gibt es in verschiedenen Größen, man spricht dann von 1:N Demultiplexern

Schaltnetze - Demultiplexer 1:4

- Beispiel: demux 1:4

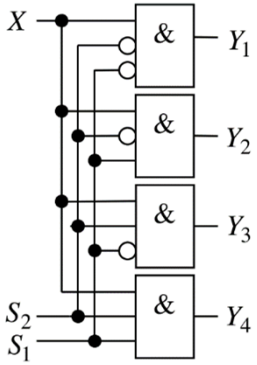


S_2	S_1	Y_1	Y_2	Y_3	Y_4
0	0	X	0	0	0
0	1	0	X	0	0
1	0	0	0	X	0
1	1	0	0	0	X

Gleichung

$$Y_1 = X \wedge \overline{S_2} \wedge \overline{S_1}$$
$$Y_2 = X \wedge \overline{S_2} \wedge S_1$$
$$Y_3 = X \wedge S_2 \wedge \overline{S_1}$$
$$Y_4 = X \wedge S_2 \wedge S_1$$

Schaltplan



Schaltnetze - Rechenschaltungen - Addierer - Halbaddierer I/II

- Addition von Ganzzahlen - Addition einer einzelnen Stelle (Halbaddierer) - Wertetabelle

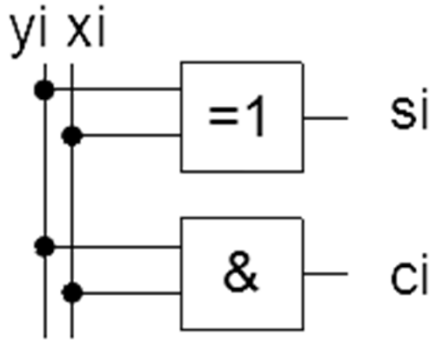
yi	xi	si=yi + xi	ci
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

- Minimierung -> Schaltfunktion -> Umsetzung als Gatter!

Schaltnetze - Rechenschaltungen - Addierer - Halbaddierer II/II

- Addition von Ganzzahlen - Addition einer einzelnen Stelle (Halbaddierer) - Gleichung und Schaltplan

si= yi‘xi + yixi‘
  = yi $\oplus$ xi
ci= yixi



Schaltnetze - Rechenschaltungen - Addierer - Volladdierer I/III

- Addition von Ganzzahlen
  - Addition einer einzelnen Stelle mit Berücksichtigung des Übertrags der Vorgängerstelle (Volladdierer) - Wertetabelle

c_{i-1}	yi	xi	si= yi+ xi + c_{i-1}	ci
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Eigentliche Addition braucht Übertrag der Vorgängerstelle

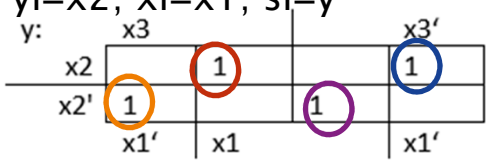
```
  0111 ( 7)
+ 0010 ( 2)
  11
= 1001 ( 9)
```

Schaltnetze - Rechenschaltungen - Addierer - Volladdierer II/III

- Addition von Ganzzahlen
  - Volladdierer - Minimierung

c_{i-1}	yi	xi	si= yi+ xi+ c_{i-1}	ci
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Mapping für si: c_{i-1}=x3; yi=x2; xi=x1; si=y

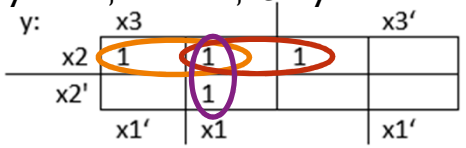


x3x2‘x1‘
x3x2x1
x3‘x2‘x1
x3‘x2x1‘

-> y= x3x2‘x1‘+ x3x2x1+ x3‘x2‘x1+ x3‘x2x1‘

->si= c_{i-1}yi‘xi‘+ c_{i-1}yixi+ c_{i-1}‘yi‘xi+ c_{i-1}‘yixi‘

Mapping für ci: c_{i-1}=x3; yi=x2; xi=x1; ci=y



x3x2
x2x1
x3x1

-> y= x3x2+ x2x1+ x3x1

->ci= c_{i-1}yi+ yixi+ c_{i-1}xi

Schaltnetze - Rechenschaltungen - Addierer – Volladdierer III/III - Übung (R)

- Addition von Ganzzahlen - Volladdierer - Schaltplan

ci=

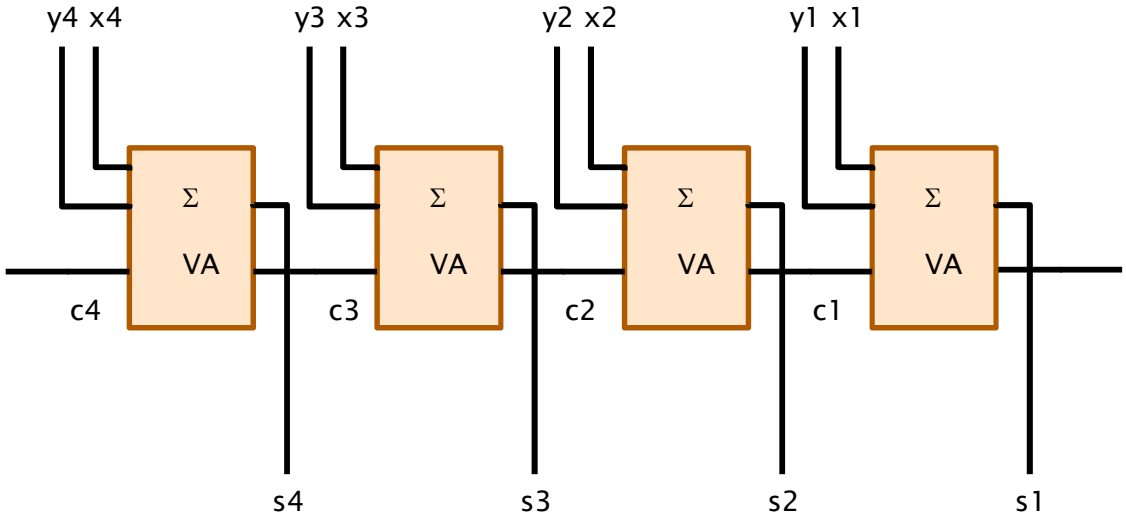
$= xiyi + xi\ c_{i-1} + yi\ c_{i-1}$

si=

$= xiyi'c_{i-1}' + xi'yi\ c_{i-1}' + xi'yi'c_{i-1} + xiyi\ c_{i-1}$

Schaltnetze - Rechenschaltungen - Addierer - RCA

- Addition von Ganzzahlen
  - Hintereinander-Schaltung von Volladdieren (Carry Ripple Adder (RCA)
    - Beispiel 4 Bit

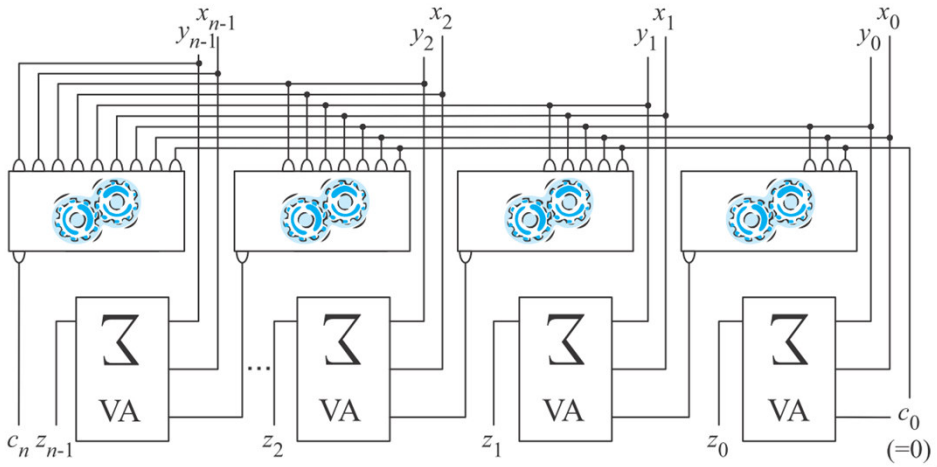


  - Einfach, wenig Bauelemente, aber langsam (Rechenzeit ~ Bitbreite)

OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG

Schaltnetze - Rechenschaltungen - Addierer - CLA - Motivation und Prinzip

- Der Ripple-Carry-Adder (RCA) ist recht Ressourcen schonend, aber sehr langsam
  - Das Carry muss von Stelle zu Stelle rutschen, bis es an der obersten Stelle ankommt
- Die Abhilfe hierfür ist eine beschleunigte Berechnung der Überträge(carry)
- Der Carry-Look-Ahead-Adder (CLA) berechnet alle Übertragsbits (carry) in einem extra Schaltnetz parallel



Schaltnetze - Rechenschaltungen - Addierer - CLA - Herleitung I/II (R)

- Startpunkt ist der Volladdierer
  - ci= c_{i-1}yi+ yixi+ c_{i-1}xi und zi=si= c_{i-1}yi‘xi‘+ c_{i-1}yixi+ c_{i-1}‘yi‘xi+ c_{i-1}‘yixi‘
- Dies kann man nun umschreiben für c_{i+1} und z_{i+1}:
  - c_{i+1}= ciyi+ yixi+ cixi und z_{i+1}=yi‘xi‘+ ciyixi+ ci‘yi‘xi+ ci‘yixi‘
- Dies kann man durch bool’sche Umformung umformen in:
  - c_{i+1}= ((xi xor yi)*ci)+ (xiyi) und z_{i+1}= ci xor (xi xor yi)
- Nun kann man eine weiter Abkürzung einführen:
  - mit gi=xiyi und pi=xi xor yi
  - c_{i+1}= (pi*ci)+gi = gi +(ci * pi)
  - z_{i+1}= ci xor pi

Schaltnetze - Rechenschaltungen - Addierer - CLA - Herleitung II/II

- Nun kann man für ci einsprechend rekursiv einsetzen und „ausrollen“ (c_{i+1}= gi +(ci * pi)):

c1= g0+ c0p0

c2= g1+ **c1**p1

= g1+ (g0+ c0p0)p1

= g1+ g0p1+ c0p0p1

c3= g2+ **c2**p2

= g2+ (g1+ g0p1+ c0p0p1)p2

= g2+ g1p2+ g0p1p2+ c0p0p1p2

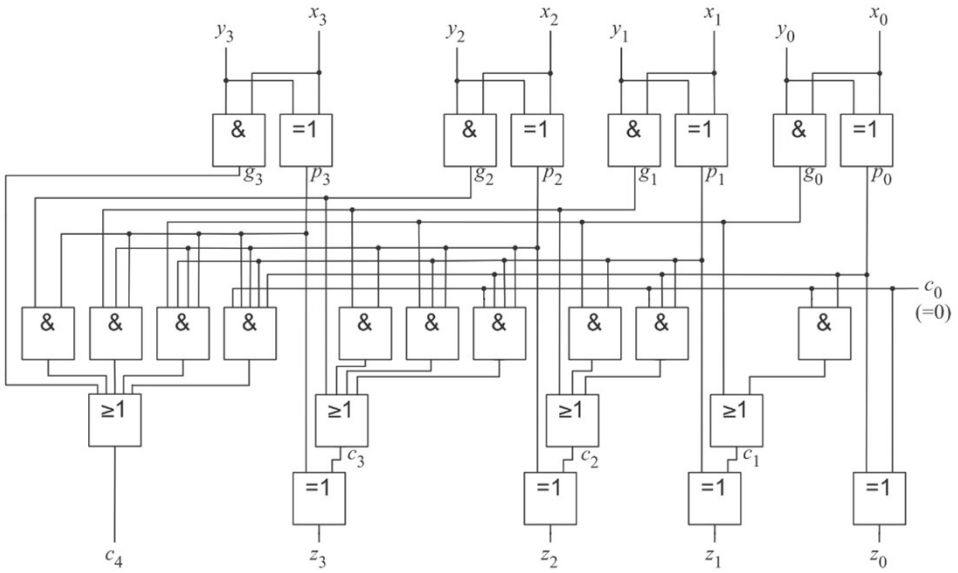
c4= g3+ **c3**p3

= g3+ (g2+ g1p2+ g0p1p2+ c0p0p1p2)p3

= g3+ g2p3+ g1p2p3+ g0p1p2p3+ c0p0p1p2p3

Schaltnetze - Rechenschaltungen - Addierer - CLA - Schaltung

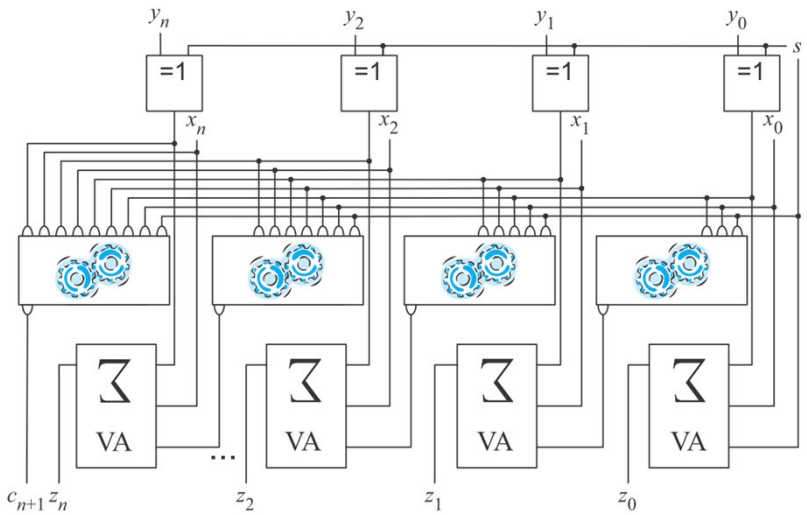
```
c1= g0+ c0p0
c2= g1+ g0p1+ c0p0p1
c3= g2+ g1p2+ g0p1p2+
    c0p0p1p2
c4= g3+ g2p3+ g1p2p3+
    g0p1p2p3+ c0p0p1p2p3
z0= c0 xor p0
z1= c1 xor p1
z2= c2 xor p2
z3= c3 xor p3
```



- An dieser Schaltung kann man gut erkennen, das die Tiefe(=Laufzeit) unabhängig von der bitbreite ist
- Aber die Anzahl an Logikgattern nimmt pro bit kontinuierlich zu

Schaltnetze - Rechenschaltungen - Subtrahierer

- Die Subtraktion kann man auf die Addition des Zweierkomplements zurückführen
  - x-y= x+(-y)
- D.h. jeder Addierer lässt sich durch eine „Zweierkomplementsstufe“ (Negation +1) zu einem Subtrahierer machen
- (Beispiel CLA-Adder)



- Wenn s=0 -> CLA-Adder
- Wenn s=1 -> yi wird negiert und 1 dazu addiert

Schaltnetze - Rechenschaltungen - Multiplizierer I/II

- Zur Wiederholung: Die binäre Multiplikation funktioniert wie die dezimale Multiplikation
- Der Unterschied ist lediglich, dass es nur 0 und 1 gibt

multiplicand	0101
multiplier	× 0111
partial products	0101
	0101
	0101
	+ 0000
result	0100011

$5 \times 7 = 35$

- Die Bildung des Partialprodukts ist recht einfach: entweder den Multiplikant (=Mal 1) hinschreiben oder 0 (= Mal 0) hinschreiben
- Das Ergebnis ist dann (die gehiftete) Addition der Partialprodukte

Schaltnetze - Rechenschaltungen - Multiplizierer II/II

- Generell multipliziert ein NxN Multiplizierer zwei N-bit Zahlen und erzeugt ein 2N-bit Ergebnis
- Multiplikation von 1-bit Zahlen ist äquivalent zu UND-Operationen -> UND-Gatter
- Jedes einzelne Partialprodukt ist eine einzelne bitweise UND-operation eines Multiplikator-Bit (B3, B2, B,1 B0) mit einem Multiplikant-Bit (A3, A2, A1, A0)
- Bei N-bit Operanden gibt es N Partialprodukte, N-1 1-bit Addierstufen