

Abb. 5.39: Programmierbarer Gitterbaustein

5.4 CMOS Schaltungen und VLSI Design

Die Boolesche Algebra beginnt mit Elementarschaltern und konstruiert daraus durch Negation, Serien- und Parallelschaltung beliebige Schaltkreise. Als Elementarschalter werden in der Praxis *Transistoren* eingesetzt. Ein Transistor hat einen Eingang (*Source*), einen Ausgang (*Drain*) und einen Steuerungseingang (*Gate*). Legt man eine Spannung zwischen Source und Drain, so fließt nur dann Strom, falls auch eine Steuerspannung am Gate anliegt.

Wir beschränken uns hier auf die modernere und stromsparende CMOS-Technik (*complementary metal oxide semiconductor*), bei der sowohl p-MOS als auch n-MOS Feldeffekttransistoren (MOSFET) in zueinander komplementären Schaltkreisen eingesetzt werden. CMOS-Schaltungen verbrauchen im Gegensatz zu den älteren Schaltungen mit Bipolartransistoren nur wenig Strom was auch einen geringeren Aufwand zur Kühlung der Chips bedingt.

Die Schaltung wird mit einer positiven Versorgungsspannung V_{CC} (*voltage of the common collector*) betrieben, für die man z.B. 2,9V (oder 5V) wählen kann. Die logischen Werte 0 und 1 entsprechen dann idealerweise den Spannungspegeln 0 V und 2,9 V. In der Praxis kann man aber den Bereich 0-0,5 V als logisch „0“ und 2,4-2,9 V als logisch „1“ interpretieren.

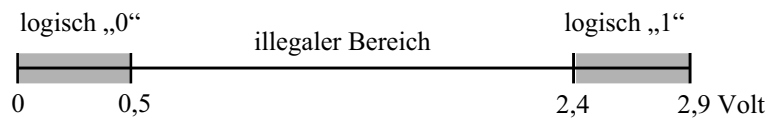


Abb. 5.40: Analoge und logische Werte

Wir hatten p-MOSFETs und n-MOSFETs als ideale Schalter eingeführt. In der Praxis unterscheiden sich ihre Schaltcharakteristiken je nachdem ob sie an $V_{CC}=1$ oder an $Gnd=0$ (*ground=Erde*) angeschlossen sind: p-MOS-Transistoren lassen das Signal 1 fast ungedämpft durch, während das 0-Signal gedämpft wird, bei n-MOS-Transistoren ist es genau umgekehrt. Daher bestehen CMOS-Schaltungen immer aus zwei Teilschaltungen - einer sogenannten *pull-up* Schaltung, die für das Ausgangssignal 1 zuständig ist und nur aus p-MOS Transistoren besteht sowie einer *pull-down* Schaltung aus n-MOS-Transistoren, die das Ausgangssignal 0 produziert.

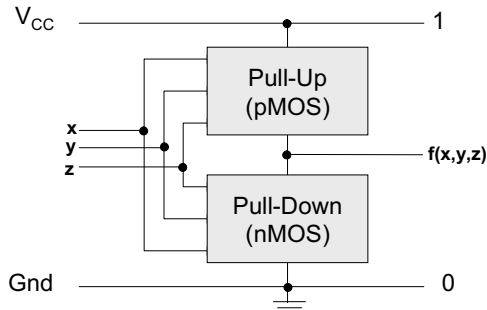


Abb. 5.41: Aufbau einer CMOS-Schaltung zur Realisierung einer booleschen Funktion $f(x,y,z)$

Selbstverständlich muss dafür gesorgt werden, dass der Ausgang nie gleichzeitig mit 1 (V_{CC}) und mit 0 (Gnd) verbunden sein kann. Dies hätte einen Kurzschluss zur Folge! Aus diesem Grund sind pull-up und pull-down Schaltung immer komplementär zueinander aufgebaut: Einer Parallelschaltung im pull-up Teil entspricht eine Serienschaltung im pull-down Kreis. Dies erklärt auch den Namen CMOS (complementary MOS).

5.4.1 Logikgatter in CMOS-Technik

Die einfachste CMOS-Schaltung ist der in der folgenden Figur gezeigte *CMOS-Inverter* der nur aus einem n-MOS und einem p-MOS besteht. Ist der Eingang $x = 1$, so sperrt der p-MOS Transistor, denn dessen Source und Gate liegen auf dem gleichen Spannungsniveau. Gleichzeitig ist am n-MOS-Transistor die Spannung zwischen Gate und Source maximal, so dass dieser öffnet und am Ausgang z das Spannungspegel $Gnd = 0$ liegt.

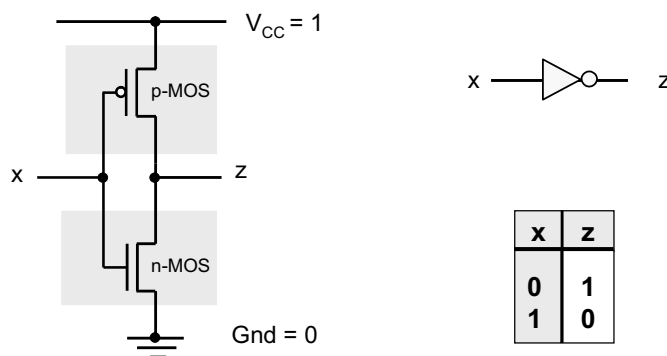


Abb. 5.42: CMOS-Inverter, Gattersymbol und Schalttabelle

Genau umgekehrt sind die Verhältnisse im Fall $x=0$. Jetzt liegen Gate und Source des n-MOS auf gleichem Niveau, so dass dieser sperrt. Dagegen ist die Spannung zwischen Gate und Source des p-MOS maximal, so dass dieser öffnet und dem Ausgang z das gleiche Span-

nungsniveau beschert wie dem Source des p-MOS also logisch 1. Insgesamt hat also z immer den entgegengesetzten logischen Wert von x , weshalb die gezeigte Schaltung der Negation entspricht.

Da im Allgemeinen p-MOS Transistoren nur im pull-up Teil verwendet werden und n-MOS nur im pull-down Teil, ist in diesen Fällen eine Spannung zwischen Gate und Source eines p-MOS gleichbedeutend mit dem Signal 0 am Gatter. Das heißt, dass in einer CMOS-Schaltung ein p-MOS Transistor leitend ist, wenn logisch 0 am Gatter liegt und analog ein n-MOS Transistor, wenn logisch 1 am Gatter liegt. Dies erklärt den Kreis im Schaltbild des p-MOS Transistors.

Vor diesem Hintergrund sind die folgenden Schaltungen auch leichter zu verstehen. Die erste Abbildung zeigt die CMOS-Schaltung für NOR, das Gattersymbol und die Wertetabelle. Man sieht wie die Serienschaltung im pull-up Teil einer Parallelschaltung im pull-down-Kreis entspricht. Nur wenn $x = 0$ und $y = 0$ sind, ist der Ausgang z mit 1 (V_{CC}) verbunden. Gleichzeitig sind beide n-MOS Transistoren gesperrt. Falls $x = 1$ oder $y = 1$ ist die Verbindung von z zu 0 (Gnd) hergestellt.

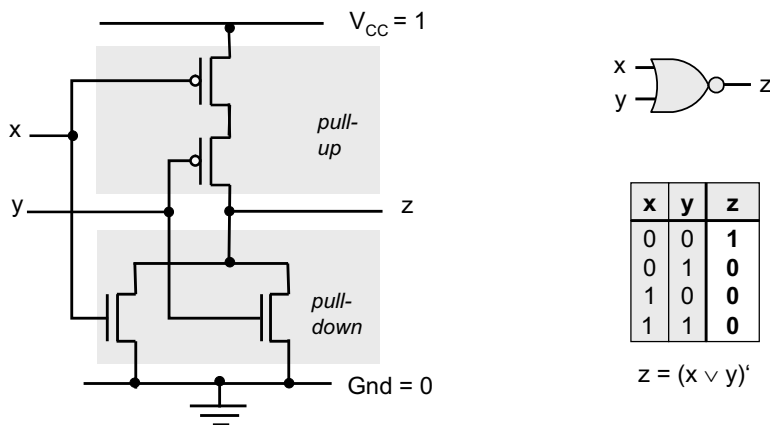


Abb. 5.43: CMOS Schaltung für NOR, Gattersymbol und Schalttabelle

Die CMOS-Schaltung für NAND ist dual zur Schaltung für NOR. Die Dualität drückt sich darin aus, dass das pull-up Netz der einen dem pull-down Netz der anderen Schaltung entspricht, wobei selbstverständlich im pull-up Teil stets nur p-MOS und im pull-down Teil nur n-MOS Transistoren verwendet werden.

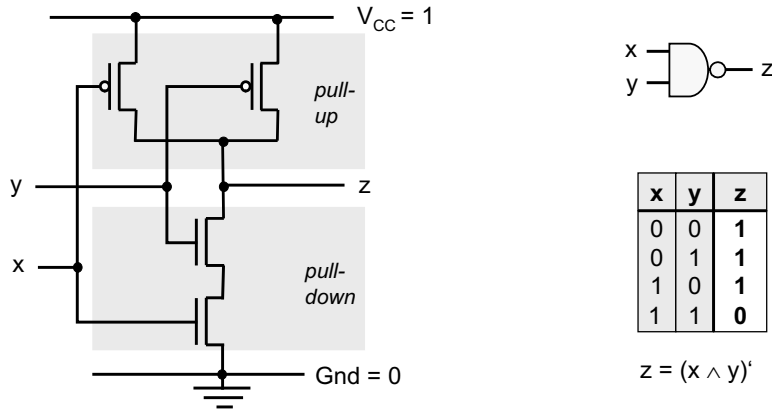


Abb. 5.44: CMOS Schaltung für NAND, Gattersymbol und Schalttable

Die Schaltglieder für AND und OR werden durch nachgelagerte Inverter realisiert, wie in der folgenden Figur am Beispiel von AND gezeigt wird.

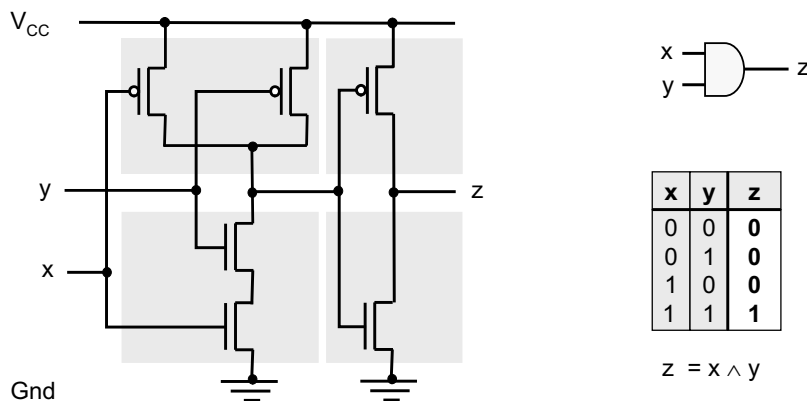


Abb. 5.45: CMOS-Implementierung von AND

5.4.2 CMOS-Entwurf

Es ist nun einfach festzustellen, wie eine beliebige Schaltung in CMOS entworfen werden kann. Sei $f(x_1, \dots, x_n)$ der Boolesche Term. Da die pull-down Schaltung genau dann das Ergebnis mit *Gnd* verbinden soll, wenn $f(x_1, \dots, x_n) = 0$ ist, negieren wir den Ausgangsterm zu $f(x_1, \dots, x_n)'$, vereinfachen diesen, und interpretieren dann jedes Literal als n-MOS Transistor, jedes + als Parallelschaltung, jedes • als Serienschaltung.

Im pull-up Teil leitet ein p-MOS-Transistor genau dann, wenn sein Gate 0 ist. Daher negieren wir alle Literale von $f(x_1, \dots, x_n)$, und bauen die Schaltung, die $f(x_1', \dots, x_n')$ entspricht. Es folgt, dass die pull-up Schaltung und die pull-down Schaltung zueinander dual sind.

Zur Illustration betrachten wir den Term $f(x, y, z) = (x' \bullet y) + z'$. Für die pull-up-Schaltung invertieren wir die Literale und erhalten $f(x', y', z') = x \bullet y' + z$, was einer Parallelschaltung von z mit der Serienschaltung von x und y' entspricht. Für die pull-down-Schaltung vereinfachen wir $f(x, y, z)' = ((x' \bullet y) + z)'$ zu $(x + y') \bullet z$, erhalten also eine Reihenschaltung von z mit der Parallelschaltung von x und y' . Eigentlich muss man jetzt noch die Negation y' von y bereitstellen. In der Praxis hat man zu jeder Eingangsvariablen oft schon an anderer Stelle auch deren Negation „vorrätig“, so dass man diese einfach abgreifen kann.

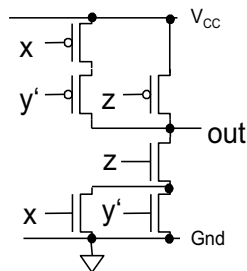


Abb. 5.46: CMOS-Entwurf - Beispiel

5.4.3 Entwurf von CMOS Chips

Die gezeigten CMOS Schaltungen erwecken den Eindruck, als müsse man alle Logikgatter bzw. sogar die Transistoren einzeln bauen und diese dann entsprechend verbinden. In Wirklichkeit werden ganze Schaltungen nach dem logischen und dem CMOS-Entwurf anhand von sogenannten Zellbibliotheken entworfen. Es beginnt mit dem Entwurf der Schaltung in einer modularen Hardwarebeschreibungssprache, z.B. *VHDL* oder *Verilog* oder *SystemC*. In solchen Sprachen kann man, aufgrund ihrer Modularität, beliebig komplexe Schaltungen spezifizieren, simulieren und testen, bevor man die teure und aufwendige Herstellung des Chips in Angriff nimmt. Eine vollständige Verilog-Implementierung einer CPU ist in dem Buch von K. Stroetmann: *Computerarchitektur* (s. Literaturverzeichnis) angegeben und genau erklärt.

Die Beschreibung eines Volladdierers in Verilog könnte folgendermaßen beginnen:

```
module fulladder(input a,b,c, output s, cout);
    sum s1(a,b,c,s);
    carry c1(a,b,c,cout);
endmodule
```

Das Modul *fulladder* bezieht sich auf zwei Untermodule, *sum* und *carry*, von denen wir das letztere schon direkt boolesch beschreiben können.

```
module carry(input a,b,c, output cout)
    assign cout = (a&b) | (a&c) | (b&c);
endmodule
```

Die *carry*-Schaltung berechnet also einfach den logischen Ausdruck $a \bullet b + a \bullet c + b \bullet c$, der zu $a \bullet b + c \bullet (a + b)$ vereinfacht werden kann. Die Summe wird analog als $a \oplus b \oplus c$ spezifiziert.

Aus der Verilog-Beschreibung kann automatisch die *Netzliste*, d.h. die Teileliste mit ihren Verbindungen, somit auch der Schaltplan der CMOS-Schaltung gewonnen werden.

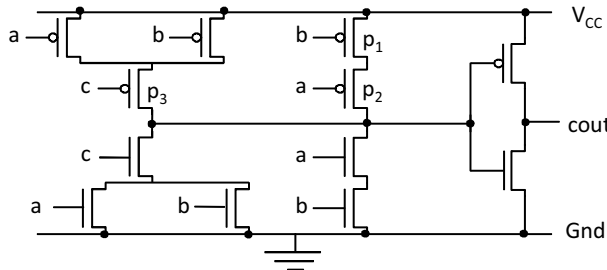


Abb. 5.47: CMOS-Schaltung zur Carry-Berechnung

Rechts in der Abbildung erkennt man die typische Inverter-Schaltung. Links daneben wird zunächst das Komplement von $a \cdot b + c \cdot (a + b)$ berechnet, das dann im Inverter wieder invertiert wird.

Diese Schaltung sieht auf den ersten Blick außergewöhnlich aus, da die Komplementarität von pull-up und pull-down Teil nicht unmittelbar ersichtlich ist. Eigentlich müssten im pull-up Teil die in Serie geschalteten p-MOS Transistoren p_1 und p_2 mit p_3 parallel geschaltet sein. Dies kann man aber offensichtlich zu der gezeigten Schaltung vereinfachen, die den Vorteil hat, dass nirgends mehr als 2 Transistoren in Reihe geschaltet sind.

Wir wollen $a \cdot b + c \cdot (a + b)$ implementieren. Weil wir uns die Invertierung jedes der Eingangssignale sparen wollen, entschließen wir uns, das Komplement $f(a,b,c) = [a \cdot b + c \cdot (a + b)]'$ zu implementieren und dieses anschließend zu invertieren.

Für die Pull-up Schaltung erhalten wir:

$$f(a',b',c') = [a' \cdot b' + c' \cdot (a' + b')] = (a + b) \cdot (c + a \cdot b) = (a + b) \cdot c + (a + b) \cdot (a \cdot b) = (a + b) \cdot c + (a \cdot b) = (a \cdot b) + c \cdot (a + b).$$

Für die Pull-down Schaltung erhalten wir: $f(a,b,c)' = [a \cdot b + c \cdot (a + b)]'' = a \cdot b + c \cdot (a + b)$ also die identische Schaltung. Damit haben wir gezeigt, dass der Ausgangsterm *selbstdual*, also identisch zu seinem dualen ist:

$$f(x_1, \dots, x_n)' = f(x_1', \dots, x_n').$$

Nach der Erstellung der *Netzliste*, d.h. der Liste aller Schaltglieder mit ihren Verbindungen, werden die Bestandteile der Schaltung in Zellbibliotheken gesucht, die das layout der p- und n-dotierte Bereiche, Gates, Isolierung Kontakte etc. bestimmen, aus denen dann die Masken für das Belichten, Ätzen und dotieren bestimmt werden.

5.4.4 VLSI-Werkzeuge

Die Konstruktion komplexer CMOS-Chips kann heute nur mit umfangreicher Werkzeugunterstützung gelingen. Der gezeigte Bildschirmabzug zeigt das freie VLSI-Entwurfswerkzeug *Electric* von *Static Free Software*. Mit diesem System haben wir eine CMOS-Schaltung,