Vorlesung 3

Teil 1 – kombinatorische Schaltungen

Folie 2

***

In dieser Vorlesung werden komplexere Digitalzellen beschreiben.

Es wird zwischen kombinatorischer und sequenzieller Logik unterschieden. Kombinatorische Logik wird nur aus den Schaltfunktionen (logische Gatter) aufgebaut. Der Ausgang hängt nur von den Eingängen ab. Für die sequenzielle Logik werden noch die Speicherzellen gebraucht. Sie werden am meistens mit den Flip-Flips (bzw. den Registern) realisiert.

Wir fangen mit den Aufbau der logischen Gattern an.

Folie 3

***

Die wichtigsten Booleschen Funktionen mit zwei Variablen sind NAND, NOR und EXOR. Da wir Inverter haben, können wir aus NAND, NOR und EXOR leicht AND, OR und die Äquivalenz (EXNOR) bauen.

Rein kombinatorisch gibt es 2^4 = 16 Booleschen Funktionen von zwei Variablen. (Die Länge der Ergebnistabelle ist 4 und für jede Zeile haben wir zwei Möglichkeiten.)

Wie kommen wir dann nur auf drei (bzw. sechs) Funktionen?

Folie 4

***

8 Booleschen Funktionen kann man aus den 16 durch Negation bekommen, wie AND aus NAND.

Zwei Funktionen (von 8) sind eigentlich keine Funktionen von zwei sondern nur einer Variable. Eine Funktion von 8 ist Konstante. Diese Funktionen werden in der Folie gezeigt.

Folie 5

***

Mit NAND, NOR, EXOR, A, B, und Null (und deren Negationen) fehlen uns noch 2 Funktionen, die in der Folie gezeigt werden. Diese können aus AND und OR mit invertierten Eingängen hergeleitet werden.

Das heißt NAND, NOR und EXOR sind ausreichend um alle Funktionen zu realisieren.

Folie 6

***

EXNOR kann man mit (N)AND, (N)OR und Inverter realisieren

Folien 7 - 9

***

Dazu kommt noch, dass es auch möglich ist NAND in NOR umzuwandeln. Also, streng genommen wäre z.B. NAND genug.

Folie 10

NAND und NOR als Schalter-Widerstand (RT-) Logik wurden bereits in Vorlesung 1 gezeigt. Es sind zwei Schalter in Serie oder Parallel geschaltet mit einem Pullup Widerstand.

Folie 11

Es wurde auch in Vorlesung 3 gezeigt, dass man einen Inverter als RT-Logik entweder mit einem NMOS und dem Pullup- oder mit einem PMOS und dem Pulldown Widerstand aufbauen kann.

Wenn man diese zwei Schaltungen kombiniert, bekommt man einen CMOS Inverter. Die Vorteile sind kein DC Strom und ein kleines Layout.

Man kann auf ähnliche Weise auch NAND und NOR als CMOS aufbauen.

Hier ist folgendes zu beachten.

Folie 12

***

Wenn der NMOS Teil für bestimmte Zeilen in der Ergebnisstabelle leitet (für die Zeilen wo wir null-Ergebnis haben), muss der PMOS Teil für genau andere Zeilen in der Tabelle leiten. Es darf nicht passieren, dass PMOS und NMOS Schaltnetz für manche Eingangskombinationen gleichzeitig leiten. Wir hätten dann einen großen „Querstrom“ und der Ausgang wäre undefiniert. PMOS und NMOS Teil sollen auch nie gleichzeitig offene Verbindungen sein. In dem Fall wäre der Ausgang von Versorgungslinien getrennt. Der logische Wert wäre ebenfalls undefiniert. Hier eine Bemerkung: Für ein Gate mit dem offenen Ausgang sagt man, dass es sich im hochohmigen Zustand befindet. Wir werden später auf die Anwendungen solcher Gatter eingehen. Im Moment werden wir solches Verhalten ausschließen.

Wie macht ein CMOS Gate in Praxis? Jede Zeile mit dem Ergebnis 0 ist die Serienschaltung von zwei (oder mehreren) NMOS Transistoren die nur für die Eingangswerte dieser Zeile leiten. Man muss alle Eingänge, die Null sind, zuerst invertieren und dann an NMOS Gates anschließen. Das ganze NMOS Netzwerk ist die Parallelschaltung von allen Reihenschaltungen die allen Zeilen = 0 entsprechen.

Folie 13

***

PMOS Teil macht man dual.

Beachten wir, dass PMOS für niedriges Gate-Potential leitet

Man muss alle Eingänge = eins invertieren.

Folie 14

***

Folie 14 zeigt EXNOR, das nach dem obigen Rezept realisiert wurde.

Folien 15-17

***

Oft kann man die logische Funktion vereinfachen.

Folie 15 zeigt den Aufbau des CMOS NOR – Gates.

PMOS Netzwerk leitet für die Eingangskombination 00 – wir haben die Reihenschaltung. NMOS Netzwerk leitet immer außer für 00 – wir haben die Parallelschaltung.

Eigentlich, wenn man nach dem obigen Rezept vorgehen würde, wäre NAND zuerst sehr kompliziert, wie in der Folie dargestellt wird. Man kann aber die Absorptionsregeln verwenden, die zur minimalen Schaltung auf der Folie 17 führen.

Folie 18

***

Folie 18 zeigt NAND

Folie 19

***

Es ist leicht die NAND und NOR auf mehr als 2 Eingänge zu erweitern

Die Schaltpläne sind in Folien 19 gezeichnet.

Folie 20

***

Zusammenfassung:

NMOS Teil leitet für die Zeilen mit null-Ergebnis

PMOS Teil leitet für die Zeilen mit eins-Ergebnis

PMOS und NMOS Teile dürfen nie gleichzeitig leiten

Sonst hätten wir einen großen Querstrom und der Ausgang wäre undefiniert

PMOS und NMOS Teil sollen auch nie gleichzeitig offene Verbindungen sein. In dem Fall wäre der Ausgang von den Versorgungslinien getrennt. Der logische Wert wäre undefiniert

Gate mit offenem Ausgang befindet sich im Hochohmigen Zustand

Folie 21

***

Bauen wir schließlich ein EXNOR auf. Es gibt hier zwei Möglichkeiten. Entweder verwenden wir die Schaltung von Folie 14. Wir brauchen hier 4 Transistoren in zwei Invertern und 8 Transistoren im EXNOR Netzwerk.

Folie 22

***

Zweite Möglichkeit: wir schreiben wir die disjunktive Normalform und realisieren wir sie mit Invertern, AND und OR Gattern.

EXOR = (~A & ~B) | (A & B)

Hier bietet sich an, OR in (N)AND umzuwandeln und die Schaltung zu vereinfachen. Wir brauchen dann 3 NAND – Gattern und zwei Invertern. Das sind 4 Transistoren in zwei Invertern und 12 Transistoren in drei NANDs. Also die erste Realisierung verwendet weniger Transistoren.

Folie 23 - 24

***

Noch komplexere Gates mit drei Eingängen.

Vielleicht der wichtigste und vielseitigste Bauteil in Digitaltechnik ist der Multiplexer. Je nachdem ob der Select-Eingang null oder eins ist, ist der Ausgang X0 oder X1. Multiplexer hat auch eigene Notation im Verilog Code

Y = sel? X1 : X0.

Multiplexer kann wie folgend dargestellt werden: (disjunktive Normalform):

Y = !sel & X0 | sel & X1

Entsprechender Schaltplan ist in Folie 23/24 gezeigt.

Wir brauchen 3 NAND Gattern (Folie 24) und einen Inverter. Das sind 3 x 4 + 2 = 14 Transistoren.

Folie 25

***

Warum ist ein Multiplexer so wichtig? Jede logische Funktion kann mit Multiplexern, Invertern und logischen Konstanten realisiert werden.

Nehmen wir als Beispiel AND.

Folie 26

***

AND ist null wenn die Variable A null ist, unabhängig von B. Wir können also einen Multiplexer verwenden und A an Select anschließen. An Eingang X0 schließen wir die logische 0. Wenn A eins ist (Select = 1), hängt das Ergebnis von Variable B ab.

AND = A ? B : 0

Variable B wird an Eingang X1 angeschlossen.

Dementsprechend kann man AND mithilfe eines Multiplexer wie in Folie 26 darstellen.

Folie 27

***

Ähnlich kann man auch ENXOR (Äquivalenz) realisieren

EXOR = A ? B : ~B

Folie 28 - 29

***

Multiplexer kann auch einfacher als in der Folie 28 gezeigt realisiert werden.

Beachten wir, dass wir bis jetzt nie zwei Gate-Ausgänge kurzgeschlossen haben. (Wir haben auch erwähnt, dass sich, im Fall von Standardgattern, ein Ausgang nie in einem Hochohmigen Zustand befinden darf.) Wenn wir zwei Ausgänge kurzschließen und wenn sie verschiede Logische Niveaus haben, würden die PMOS und NMOS Transistoren gegeneinander wirken. Der PMOS-Teil würde den Ausgangsknoten mit VDD kurzschließen und NMOS-Teil mit GND. Wir hätten einen Querstrom (Folie 29).

Folie 30 - 31

***

Wir können aber ein Gate so erweitern, dass es sich auch in einem hochohmigen Zustand befinden kann. Die Ausgänge solcher Gates können auch kurzgeschlossen werden. Die Bedingung ist es, dass sich in einem Moment nur ein Gate im niederohmeigen Zustand befinden darf.

Das einfachste Beispiel eines Bauteils mit dem zusätzlichen hochohmigen Zustand ist ein, so genannter, Gated-Inverter.

Wenn der „Enable“ Eingang eins ist, funktioniert der Inverter wie ein ganz gewöhnlicher Inverter. Mit Enable = null, ist der Ausgang von VDD und GND „abgeklemmt“ - er schwebt („floatet“) im hochohmigen (high impedance) Zustand.

Folien 32 – 35

***

Diese Folien zeigen die Schaltpläne vom Gated-Inverter von Grundidee bis zur endgültigen Schaltung.

Die Gated-Inverter werden oft benutzt. Zum Beispiel, man kann mit zwei Gated-Invertern einen Multiplexer realisieren.

Folie 36

***

Die Schaltung ist in Folie 36 gezeigt. Wir brauchen zwei normale- und zwei Gated-Invertern – es sind insgesamt 2 x 2 + 2 x 4 = 12 Transistoren, zwei weniger als auf Folie 28.

Folie 37

***

Oft werden auch mehrfache Multiplexer verwendet. Ein Anwendungsbeispiel ist es, wenn man die digitalen Signale von mehreren Quellen über eine Leitung übertragen soll.

Hier könnten wir mehrere Gated-Inverter verwenden. Deren Ausgänge sind kurzgeschlossen. Als Select haben wir normalerweise eine binäre Zahl die der Kanalnummer (Eingangsnummer) entspricht. Wir betrachten in diesem Fall Select als Bus: Sel = Sel(1:0). Z.B. für Sel(1:0) = 11 wird der Inverter X3 (bzw. Kanal X3) an den Ausgang geschlossen und alle andere Inverter sind im High-Impedance Zustand. Die Select Eingänge können mithilfe von AND-Gattern realisiert werden. Z.B. der Select-Eingang von X3 ist die logische Funktion:

Sel1 & Sel0,

weil diese AND Verknüpfung nur für Sel(1:0) = 11 eins ist.

Folie 38

***

Folie 38 zeigt ein weitere wichtige Schaltung - Decoder. Hier haben wir Z.B. einen 8-bit Eingang D(7:0), eine binäre Zahl, und 2^8 Ausgänge.

Wenn der binäre Eingang m ist (binär Kodiert) ist der m-te Ausgang 1. Alle anderen Ausgänge sind null.

Folie 39

***

Den Decoder können wir auf mindestens zwei Weisen realisieren, die erste ist in Folie 39 gezeigt.

Wir verwenden 2^8 AND Gattern mit n Eingängen. Das Prinzip ist das gleiche wie bei dem Multiplexer. Wenn z.B. das AND-Gate dem Ausgang 5 gehört, sollte es „1“ für die binäre Zahl

D(7:0) = 0000_1001 erzeugen.

Daraus folgt:

Y5 = !D7 & !D6 & !D5 & !D4 & D3 & !D2 & !D1 & D0

Alle Variablen, die null sind, werden negiert.

In solcher Realisierung brauchen wir 256 ANDs mit 8 Eingängen und 8 Invertern. Das sind insgesamt 256 x 16 + 8 x 2 ~ 4000 Transistoren.

Folie 40

***

Decoder kann als Teil eines Multiplexers verwendet werden, das die Gated Invertern ein- und ausschaltet. Folie 40 zeigt den 256 -> 1 Multiplexer.

Für einen solchen Multiplexer werden etwa 5500 Transistoren benötigt.

Folien 41 und 42

***

Diese Folien zeigen als Vergleich einen Analogmultiplexer. Hier werden statt Gated-Invertern die Schaltern (Transmission-Gates) benutzt. Ein Analogmultiplexer kann verschiedene analoge Signale über eine Leitung übertragen. Im Unterschied zum Digitalmultiplexer, ist der analoge Bi-Direktional. Er kann also die Signale in beide Richtungen leiten. Das wird in Folie 42 gezeigt. Wenn man in einem Analogmultiplexer die Eingänge und Ausgänge „vertauscht“ bekommt man einen Demultiplexer.

Folie 43

***

Diese Folie zeigt den Aufbau des Digtal-Demultiplexers mit 256 Ausgängen. Diese Schaltung soll ein Eingangssignal an einen von vielen Ausgängen verteilen. Die unbenutzten Ausgänge sollen null sein.

Für die Realisierung dieser Schaltung werden etwa 5000 Transistoren gebraucht. Zuerst brauchen wir 256 8 Fach ANDs mit jeweils 10 Transistoren.

Wir brauchen auch, für jenen Select Eingang, einen Inverter (2 Transistoren) um die Negation zu erzeugen.

Bemerken wir, dass man aus einem Demultiplexer einen Decoder bekommt, wenn man Eingangssignal an logische Eins anschließt.

Folie 44

***

Zusammenfassung

Ein n-Fach Demultiplexer verteilt ein Digitalsignal an 2^n Ausgänge. Wenn man den Eingang des Demultiplexers an „1“ anschließt, bekommt man einen Decoder. Den Decoder kann man in einem n -> 2^n Multiplexer verwenden.

Folien 45 und 46

***

Ein Multiplexer kann auch mit weniger Transistoren realisiert werden, wenn eine Baumstruktur verwendet wird. In jedem Knoten verwenden wir jeweils einen (2->1) Multiplexer. Der Select Eingang von Multiplexer der ersten Stufe wird an Sel0 angeschlossen, von der zweiten an Sel1, usw. Diese Schaltung ist etwas langsamer als die Standardschaltung (Folie 40), benötigt aber um Faktor 5.5 weniger Transistoren.

Folie 47

***

Auch ein Demultiplexer (und Decoder) können als Baumstruktur realisiert werden.

In jedem Knoten verwenden wir jeweils einen (1->2) Demultiplexer. Der Select Eingang der ersten Stufe wird an Sel3 angeschlossen, der nächsten an Sel2 usw. Die Zahl der Transistoren, im Falle eines 1->256 Demultiplexers ist:

(128 + 64 + … + 1) x 12 (2xAND in jedem 1->2 Demux) + 8 x 2T (Negation für jeden Select Eingang) ~ 3000 Transistoren.

Trick: Man kann statt AND-Gattern in den Demultiplexern, stufenweise abwechselnd NANDs und NORs verwenden und so die Schaltung weiter vereinfachen.

Vorlesung 4

Teil 2 – sequentielle Schaltungen

Folie 48

***

In der ersten Vorlesung wurden der Flip-Flop und das Latch eigeführt. Als Speicherelement haben wir einen Kondensator angenommen. Das Latch speichert ein Eingangsniveau auf einem Kondensator, wenn Load Signal 1 ist. Wenn Load = 0, bleibt der Zustand erhalten.

Ein Flip-Flop besteht im Prinzip aus zwei Latch-es in Reihe. Das Load-Signal des ersten Latch ist an CkB angeschlossen und das Load-Signal des zweiten Latch an Ck. Flip-Flop Ausgang ist der Ausgang des zweien Latch. In dieser Konfiguration wird im Flip-Flop, vereinfach gesagt der Eingangswert D im Moment der steigenden Talkflanke gespeichert. Spätere Änderungen am D-Eingang haben keine Wirkung auf den Ausgang bis zur nächsten Taktflanke.

Folie 49

***

Latch

Die Funktionsweise vom Latch erinnert an ein System mit einer Schleuse.

Folie 50

***

Flip-Flop

Die Funktionsweise vom Flip-Flop erinnert an ein System mit zwei Schleusen.

Der Eingangssignal D = 5 wird im Moment Ck 0->1 (zweites Bild, erste Reihe) am Ausgang sichtbar und gespeichert.

Eine weitere Änderung am Eingang (D=2) wird erst bei der nächsten steigenden Taktflanke (drittes Bild, zweite Reihe) gespeichert und an Ausgang übertragen.

Wichtig ist, dass bei Änderung Ck 1->0, zuerst beide Schleusen geschlossen werden und dann die erste geöffnet. So passiert keine Änderung am Ausgang bei der fallenden Taktflanke.

Folie 51

***

Das wird in diesem Beispiel nicht gewährleistet, und der „Flip-Flop“ funktioniert nicht richtig.

Folie 52

***

Der Nachteil einer Latch Schaltung mit Kondensator ist es, dass sie den Zustand nicht beliebig lange halten kann. Der Kondensator wird langsam entladen. Solche Schaltungen nennt man dynamische Logik.

Folie 53

***

Die nächsten Folien zeigen wie man ein Latch baut, das den Zustand so lange hält, bis die Spannungsversorgung ausgeschaltet wird. Solche Speicherzellen nennt man statisch.

Statische Speicherzellen basieren auf einer Mitkopplung (einer positiven Rückkopplung).

Betrachten wir zwei in Reihe geschalteten Invertern. Ihre Kennlinie sieht wie in Folie 53 aus.

Folie 54

***

Wenn wir nun den Ausgang des zweiten Inverters mit dem Eingang des ersten verbinden, haben wir zunächst Vin = Vout. Der Zustand der Schaltung liegt also im Schnittpunkt der Kennlinie Vout = f(Vin) und der Gerade Vout = Vin.

Folie 55

***

Wir sehen dass es drei Schnittpunkte gibt: Vout/Vin = 0 (logische 0), Vout/Voin = VDD (logische eins) und Vout=Vin ~ VDD/2 (undefiniert).

Folie 56

***

Die ersten zwei Arbeitspunkte sind stabil.

Beweis:

Nehmen wir an, wir haben eine kleine Störung welche folgendes bewirkt: Vin = Vout – Delta.

Wir sehen, dass diese Vin-Änderung, den Vout in ersten zwei Arbeitspunkten nicht beeinflusst.

Folien 57 und 58

***

Wenn wir aber den dritten Arbeitspunkt betrachten, sehen wir, dass eine kleine Verringerung von Vin (z.B. wegen dem Rauschen), wegen der steilen Kennlinie zu noch größerer Verringerung von Vout führt, usw. Die Schaltung kommt aus dem instabilen Arbeitspunkt heraus, und landet entweder in den Arbeitspunkt Vout/Vin = 0 oder in den Arbeitspunkt Vout/Voin = VDD. Analogie wäre eine Münze die fast nie auf ihrem Rand stehen bleibt, oder eine Kugel die auf der Spitze eines Bergs nicht lange bleiben will.

Folie 59

***

Die Schaltung ist die Basis einer SRAM Zelle. Die zwei Schaltern werden benutzt, und die RAM Zelle zu beschreiben oder aus ihr zu lesen.

Folie 60

***

Ein Latch basiert normalerweise aus einer modifizierten Version der Speicherzelle. Multiplexer wird benutzt. Der Select Eingang ist an Load Signal angeschlossen. Eingang X1 ist der Latch Eingang D. X0 wird an den Ausgang des Multiplexers angeschlossen.

Folie 61

***

Wenn Load = 1, ist das Latch „transparent“ – der Eingang ist direkt am Ausgang sichtbar.

Folie 62

***

Wenn Load = 0, haben wir im Prinzip die gleiche Schaltung wie in einer RAM Zelle. Der Multiplexer behält den Zustand den wir vorher beim Load = 1 hatten.

Folie 63

***

Einen Flip-Flop bilden wir aus zwei Latches. Es soll dabei unbedingt vermieden werden, dass beide Latches gleichzeitig transparent werden, vor allem wenn wich Ck von 1 auf 0 ändert (inaktive Flanke).

Folie 64

***

Das könnte passieren wenn z.B. die Takt-Invertern zu langsam sind. Wir haben gesehen, dass große kapazitive Last die CMOS Schaltungen verlangsamt. Stellen wir uns die folgende Schaltung vor.

Wir bilden aus 1024 Flip-Flops ein 1kBit Register. Um Platz zu sparen, verwenden wir zwei kleine Inverter für alle Flip-Flops. Das wäre schlecht, da jeder Inverter große kapazitive Last hätte. Die Taktflanken wären verlangsamt und es könnte passieren, dass beide Latches im Flip-Flop kurze Zeit Load = 1 haben.

Folien 65 und 66

***

Um solche Probleme zu vermeiden, werden die Takt-Inverter in der Regel in den Flip-Flops eingebaut.

Folie 67

***

Nach dem Einschalten der Spannungsversorgung befindet sich ein Flip-Flop, genauso wie eine RAM Zelle, in einem unbekannten logischen Zustand. Wir könnten uns vorstellen, dass alle Flip-Flops zuerst in den astabilen Zustand kommen und dann, je nachdem wie sie aufgebaut sind, in logisch Eins oder Null Zustand kommen.

Um einen unbekannten Anfangszustand zu vermeiden, werden die Flip-Flops oft so erweitert, dass sie ein asynchrones Reset Signal haben.

Die Schaltung eines Flip-Flops mit dem asynchronen Reset ist in Folie 67 gezeigt.

Folien 68 und 69

***

Betrachten wir einen erweiterten Flip-Flop im Load = 0 Zustand. Reset = 1 (aktiv hoch) erzwingt logische Null am Ausgang, sie wird „rückgekoppelt“. Auf diese Weise bleibt Null gespeichert auch wenn Reset wieder inaktiv wird (Folie 69).

Folie 70

***

Im Flip-Flop ist immer wenigstens ein Latch im Speicherzustand, so dass ein Reset immer möglich ist wenn beide Latches die Reset Logik enthalten.

Asynchron Reset ist *stärker* als der Takteingang. Sobald Reset = 1 wird, wird der Flip-Flop Ausgang null, unabhängig von D und Ck Eingängen.