

## 1. Chip Design

In jedem elektronischen Produkt sind sie zu finden, in Mobiltelefonen und Computern, in Waschmaschinen und Autos, in Flugzeugen und Herzschrittmachern. Die Rede ist von Mikrochips. Diese kleinen elektronischen Helfer übernehmen in den eingesetzten Geräten die unterschiedlichsten Aufgaben. Ebenso vielfältig wie die Einsatzgebiete der Mikrochips sind auch die Tätigkeitsbereiche bei deren Entwicklung. Auf dem Weg von der Idee bis hin zum fertigen Produkt müssen die Entwicklungsingenieurinnen und Entwicklungsingenieure eine Vielzahl interessanter Problemstellungen lösen. Diese betreffen den Entwurf der Schaltung, Schaltungsbeschreibung und Implementierung, Überprüfung auf korrektes Schaltungsverhalten (Verifikation), sowie Herstellung und Test der fertigen Mikrochips.

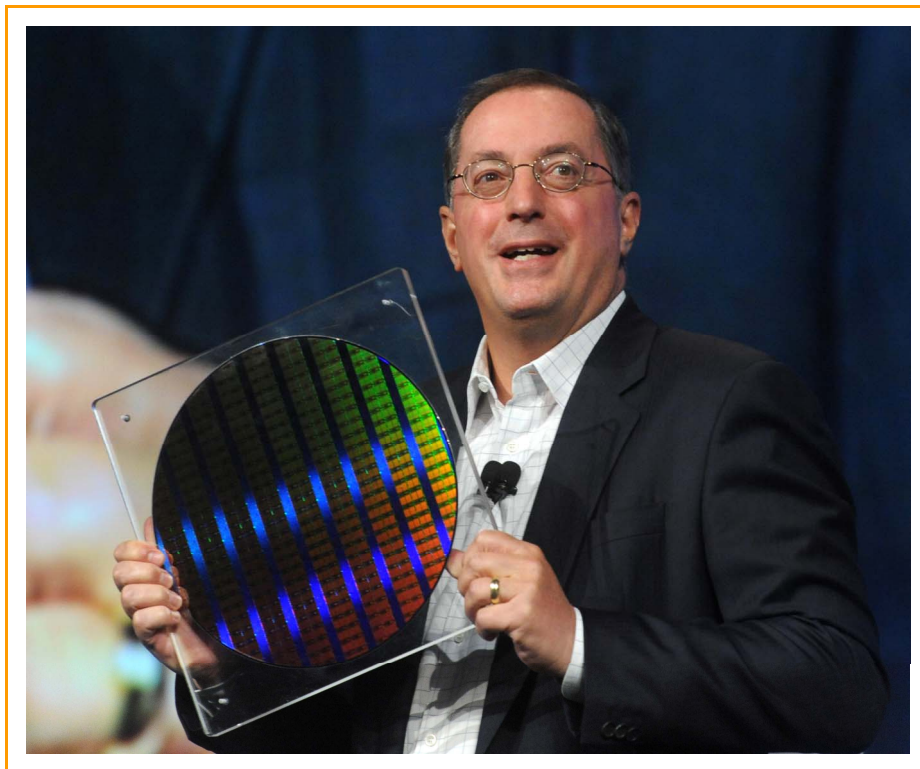


Abbildung 1 Moderne Mikrochips auf einem Wafer

In dieser Aufgabe wollen wir in Aufgabenteil 1.1 zunächst einen Einblick in das Thema Chip Design geben und einige Grundlagen vermitteln. Anschließend werden wir zwei wichtige Aspekte des Chip Design näher betrachten: die Verifikation einer Schaltung und den Test der fertigen Mikrochips. Im Aufgabenteil 1.2 soll eine vorgegebene Schaltung mit Hilfe eines digitalen Simulationsprogramms analysiert werden. In Aufgabenteil 1.3 ist ein einfaches Verfahren zur Identifizierung von Defekten in Mikrochips anzuwenden. Die Wichtigkeit von Verifikation und Test wird im abschließenden Aufgabenteil 1.4 deutlich, in dem es um die Wirtschaftlichkeit bei der Chip-Herstellung geht.

## 1.1.a Grundlagen

Der folgende Text soll ein paar Grundinformationen zum Thema Chip Design vermitteln. Füllt die offenen Lücken mit dem jeweils passenden Wort (bzw. Zahl).

Die Entwicklung eines Mikrochips, zum Beispiel eines Mikroprozessors für Computer, ist zweifellos eine der interessantesten und komplexesten Herausforderungen, denen man als Ingenieurin und Ingenieur begegnen kann. Der Ablauf bei der Entwicklung eines Mikrochips stellt sich in der Regel wie folgt dar.

Basierend auf einer Produktidee erarbeiten Chip Designer zunächst eine funktionale Beschreibung der Schaltung. Diese sog. Spezifikation beschreibt das funktionale Verhalten der zu entwickelnden Schaltung, meist in textueller Form, illustriert mit Bildern, Tabellen und Diagrammen. Große funktionale Schaltungsblöcke werden in [1] \_\_\_\_\_ Blöcke aufgeteilt um die Komplexität zu reduzieren.

Im nächsten Entwicklungsschritt wird diese Beschreibung in eine für Computer verständliche Form übersetzt. Dies bezeichnet man als Schaltungsimplementierung und erfolgt mit Hilfe einer speziellen Programmiersprache zur Schaltungsbeschreibung (engl. hardware description language, HDL). Zwei der populärsten HDLs für den Entwurf digitaler Schaltungen sind VHDL und [2] \_\_\_\_\_. Die Schaltungsbeschreibung mittels HDL ermöglicht nicht nur eine Simulation der Schaltung sondern auch die automatische Umsetzung in eine sog. Netzliste. Dieser Vorgang wird [3] \_\_\_\_\_ genannt. Dabei beeinflusst die Art und Weise der HDL-Beschreibung maßgeblich die Eigenschaften der resultierenden Schaltung. Zu deren wesentlichen Merkmalen gehören im Fall einer digitalen Schaltung neben der maximal möglichen [4] \_\_\_\_\_ auch die benötigte Silizium-Fläche sowie die Verlustleistung.

Die bereits erwähnte Netzliste ist eine Datei, die eine Ansammlung von Logikgattern und Informationen über deren Verbindungen miteinander enthält. Sie stellt die Grundlage für den nächsten Entwicklungsschritt dar, die physikalische Implementierung der Schaltung (engl. physical layout). Dies ist ein komplexer Vorgang, auf den in der späteren Aufgabe 3 näher eingegangen wird. Das Resultat der physikalischen Implementierung ist wiederum eine Datei, die notwendige Informationen zur [5] \_\_\_\_\_ des Mikrochips in einer Fabrik enthält. Die Abfolge der aufeinander aufbauenden Entwicklungsschritte, die durch den Einsatz moderner EDA-Software ermöglicht bzw. bestimmt wird, bezeichnet man im Englischen als [6] \_\_\_\_\_.

Nach dem Abschluss der Entwicklung wird die physikalische Schaltungsbeschreibung zusammen mit den notwendigen Testmustern an die Fabrik übertragen, wo die Mikrochips gefertigt und getestet werden. Moderne Mikrochips wie der Intel Core i3-540 Mikroprozessor werden in CMOS Technologie mit Strukturbreiten von nur [7] \_\_\_\_\_ nm produziert. Beim Herstellungsprozess werden mehr als 200 einzelne Prozessschritte durchlaufen, bevor die feinen Chip-Strukturen in einer Vielzahl übereinanderliegender Schichten auf der Grundplatte aus hochreinem [8] \_\_\_\_\_, dem sog. [9] \_\_\_\_\_, aufgebracht sind. Für einen einzelnen Mikrochip dauert der gesamte Herstellungsprozess mehrere Wochen.

Ein wesentlicher Punkt bei der Entwicklung von Mikrochips ist das Erkennen von Fehlern. Entwurfsfehler, die bei jedem Schritt in der Entwicklungsphase entstehen können, sollen durch Verifikation erkannt werden, d.h. durch den Vergleich mit dem in der Spezifikation beschriebenen Verhalten. Produktionsfehler, die bei der Herstellung auftreten, werden hingegen durch das Testen der fertigen Mikrochips ermittelt. Je früher mögliche Entwurfsfehler erkannt werden, desto [10] \_\_\_\_\_ ist dies in Bezug auf die Entwicklungskosten. Daher versucht man Fehlerfreiheit unter anderem durch Simulationen nach jedem Entwicklungsschritt zu gewährleisten.

## 1.1.b Grundlagen

Beantwortet folgende Fragen zu den Grundlagen der Digitaltechnik:

- 1) Grundlegende Begriffe der Computertechnik sind „Bit“ und „Byte“.
  - Wie viele unterschiedliche Werte kann ein Bit annehmen?
  - Wie viele Bits sind in einem Byte enthalten?
  - Wie viele unterschiedliche Werte kann ein Byte annehmen?
- 2) Wandelt die beiden unten dargestellten, positiven Binärzahlen in eine Dezimalzahl um:
  - 10010110
  - 01011101
  - Beschreibt kurz eure Vorgehensweise (ca. 4-5 Sätze; max. 120 Worte).
- 3) Zeichnet eine sog. Wahrheitstabelle für die booleschen Logikfunktionen NAND (negiertes Und) und XOR (exklusiv Oder) mit jeweils 2 Eingängen.
- 4) Skizziert einen Halbaddierer, der maximal aus 4 Logikgattern besteht. Benutzt dafür die genormten grafischen Symbole für Logikgatter (DIN, vgl. Zusatzinformationsblatt) und gerade, horizontale bzw. vertikale Linien zur Verbindung der Gatteranschlüsse. Markiert deutlich, welche Anschlüsse Eingänge und welche Ausgänge sind.

Wir möchten einen 64-Bit-Multiplizierer testen. Die Schaltung hat also  $2 \cdot 64 = 128$  Eingänge. Die interne Schaltung ist dem Tester nicht bekannt, aber er weiß, dass es keine speichernden Elemente innerhalb der Schaltung gibt. Der Tester möchte ganz sicher sein, dass die Schaltung funktioniert und plant daher, alle möglichen Multiplikationen zu testen. Wie aufwändig wird der Test wohl? Beantwortet dazu die folgenden Fragen:

- 5) Wie viele mögliche Eingangskombinationen gibt es?

**Tipp:** Die Multiplikationen  $a \cdot b$  und  $b \cdot a$  sollen separat getestet werden.

- 6) Wie lange dauert ein vollständiger Test (in Sek.), wenn der Mikrochip mit einem Takt von 2 GHz betrieben wird und je eine Multiplikation pro Takt ausführen kann?

### Form der Lösung

Teil 1.1.a: Tabelle mit den einzelnen Begriffen

Teil 1.1.b: Antworten auf die Fragen 1 - 6

## 1.2 Verifikation durch Simulation

In diesem Aufgabenteil steigt ihr in die Entwicklung und Simulation digitaler Schaltungen ein. Im Gegensatz zur Softwareentwicklung, bei der die Programmierer ihre Programme meist sofort ausprobieren können, wird das Verhalten der Schaltungen während der Entwicklung nur nachgebildet (simuliert).

Eine digitale Schaltung kann sowohl als Schaltplan als auch als textuelle Hardwarebeschreibungssprache (HDL, engl. Hardware Description Language) dargestellt werden. Schaltpläne sind oft einfacher für den Menschen zu lesen, eine HDL Datei kann besser von Computerprogrammen, wie z.B. von einem Simulator, verarbeitet werden.

Bei der Simulation werden üblicherweise die zu simulierende Schaltung sowie eine Testumgebung (engl. Testbench) als HDL Dateien in einen Simulator geladen. Dabei bestimmt die Testbench, welche Testwerte an die Eingänge der Schaltung angelegt werden sollen. Der Simulator berechnet dann die Werte an den Ausgängen der Schaltung und auch alle Werte auf internen Leitungen der Schaltung. Das Resultat kann man sich zum Beispiel in einem Pegeldiagramm (engl. "Waveform") anschauen (vgl. Abbildung 2). Dieses zeigt an, wie sich die Werte der beobachteten Signale im Verlauf der Zeit ändern.

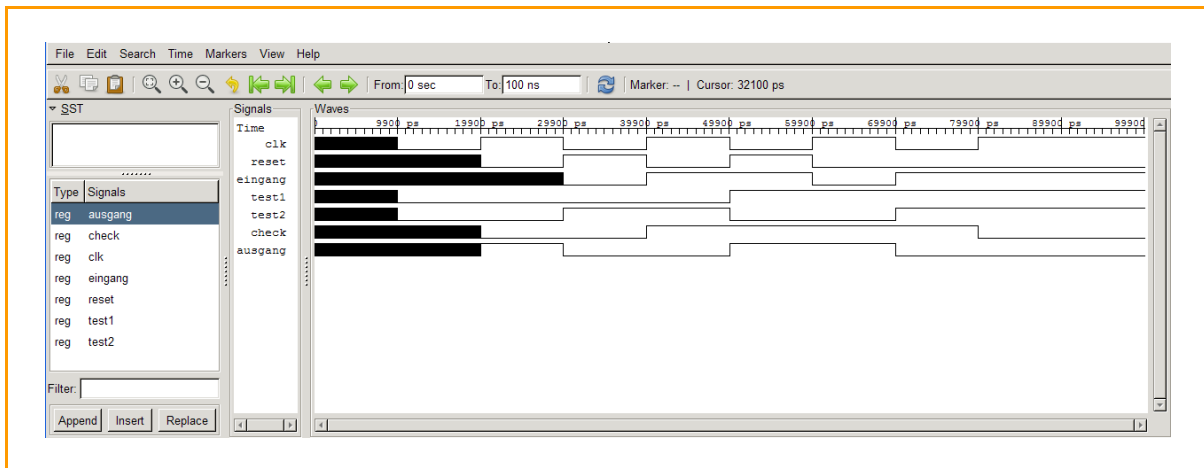


Abbildung 2 Beispiel eines Pegeldiagramms (Waveform)

Für diesen und den nächsten Aufgabenteil (1.2 und 1.3) werden der Simulator GHDL sowie der Viewer GtkWave verwendet, die in der ZIP-Datei beigefügt sind. Eine Kurzanleitung findet ihr im Zusatzinformationsblatt. Alle zu simulierenden Dateien sind in dem Unterverzeichnis „c:lilca1/hdl“ enthalten.

## 1.2.a Volladdierer

Zunächst betrachten wir eine einfache Schaltung, die drei Bit addiert - ein sogenannter Volladdierer (Abbildung 3a)). Genau diese Schaltung ist auch in der HDL Datei „full\_adder.vhdl“ beschrieben (Abbildung 3b)).

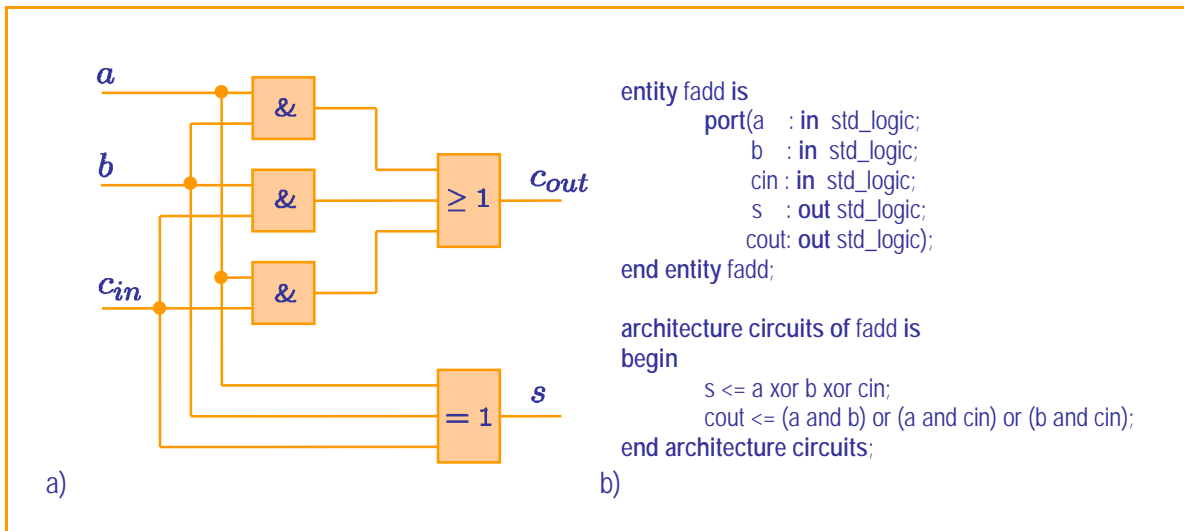


Abbildung 3 Volladdierer (Schaltbild und HDL-Code)

Simuliert die Schaltung „full\_adder.vhdl“ und die zugehörige Testumgebung „full\_adder\_test.vhdl“ mit dem Simulator GHDL (vgl. Anleitung). Öffnet die generierte Waveform mit GtkWave (vgl. Anleitung) und beobachtet, wie sich die Eingänge a, b, cin sowie die Ausgänge s und cout ändern. Ändert die Testumgebung „full\_adder\_test.vhdl“ so, dass alle 8 möglichen Kombinationen von a, b und cin simuliert werden. Macht einen Screenshot der Waveform für die abzugebende Lösung.

## 1.2.b Komplexe Schaltung

Betrachten wir nun eine komplexere Schaltung mit 4 Eingängen und 8 Ausgängen wie in Abbildung 4 dargestellt. Die Schaltung liegt als HDL-Datei „*schaltung.vhdl*“ vor. Eine Vorlage für eine Testumgebung ist in der Datei „*schaltung\_test.vhdl*“ gegeben.

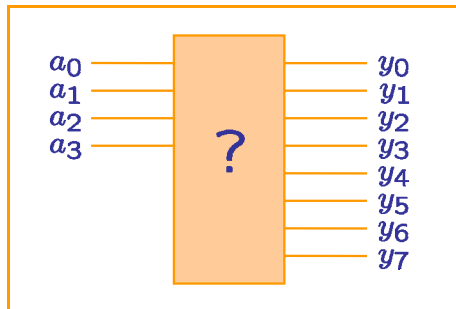


Abbildung 4 Komplexe Schaltung

Simuliert die Schaltung mit allen möglichen Werten für den Eingang  $a$  bestehend aus den Eingangsbits  $a_3$ ,  $a_2$ ,  $a_1$ ,  $a_0$ . Ergänzt dabei die Datei „*schaltung\_test.vhdl*“.

- 1) Macht einen Screenshot der Waveform.
- 2) Welche bekannte Funktion berechnet die Schaltung?

## 1.2.C Timing

Die Simulationen bisher waren idealisiert und haben keine Signalverzögerungen berücksichtigt. Tatsächlich verursachen logische Gatter eine gewisse Verzögerungszeit. In der Datei „*schaltung\_delay.vhdl*“ ist dieselbe Schaltung wie in 1.2.b beschrieben, jedoch mit einer Verzögerungszeit von einer Nanosekunde (ns) für jedes Gatter. Die Summe eines Addierers ist demnach nach einer Nanosekunde, das Übertragungssignal "Carry" erst nach zwei Nanosekunden berechnet.

Simuliert die Schaltung „*schaltung\_delay.vhdl*“ und beobachtet, wie der Ausgangsvektor **y** erst nach einiger Zeit den richtigen Wert annimmt.

- 1) Wie lange dauert es, wenn das Eingangssignal von **a**="0000" auf **a**="1011" umgeschaltet wird, bis am Ausgang das richtige Ergebnis anliegt? Nach welcher Zeit nehmen die einzelnen Bits **y**<sub>0</sub> bis **y**<sub>7</sub> jeweils den richtigen Ausgangswert an? Macht einen Screenshot von eurer Waveform, in der man die Wechsel von **a** und **y** bei diesem Übergang sieht.
- 2) Wie lange dauert es, wenn das Eingangssignal von **a**="1111" auf **a**="1011" umgeschaltet wird, bis am Ausgangsvektor das richtige Ergebnis anliegt? Nach welcher Zeit nehmen die einzelnen Bits **y**<sub>0</sub> bis **y**<sub>7</sub> jeweils den richtigen Ausgangswert an? Macht einen Screenshot von eurer Waveform, in der man die Wechsel von **a** und **y** bei diesem Übergang sieht.
- 3) Der Übergang von **a**="1111" auf **a**="1011" ist zugleich der langsamste Übergang. Mit welcher maximalen Taktfrequenz könnte man die Schaltung demnach betreiben?

### Form der Lösung

Teil 1.2.a: Waveform (Screenshot) und geänderte Testumgebung „*full\_adder\_test.vhdl*“

Teil 1.2.b: Waveform (Screenshot) und Antwort bezüglich der Funktion der Schaltung

Teil 1.2.c: Waveform (Screenshot) und Antworten auf die Fragen 1 - 3

### 1.3.a Test von Mikrochips - Fehlermodell

Sowohl beim Entwurf (Design) von Mikrochips als auch bei der Herstellung können sich Fehler „einschleichen“. Fehler im Entwurf können durch die in Aufgabenteil 1.2. durchgeführten Simulationen gefunden und vor der Herstellung repariert werden. Doch auch bei der Herstellung kann es zu Defekten bei einzelnen Mikrochips kommen. Diese Defekte werden meist durch kleinste Verunreinigungen verursacht, die sich z. B. in der Luft befinden. Da moderne Mikrochips sehr komplex sind, ist es gar nicht einfach, herauszufinden, ob ein hergestellter Mikrochip fehlerfrei ist. Um dieses „Testen“ nach der Herstellung geht es in diesem Aufgabenteil.

Die Fehler, die bei der Herstellung von Mikrochips auftreten, können sehr vielfältig sein. Um die Komplexität der verschiedenen Fehler zu reduzieren wird mit „Fehlermodellen“ gearbeitet. Das einfachste davon ist das „Hafffehlermodell“ (engl. stuck-at fault model). Dabei wird angenommen, dass ein internes digitales Signal fehlerhaft permanent auf 1 oder 0 gehalten wird. Es ist dabei notwendig die Wahrheitstabellen für alle Gatter zu erweitern, da sowohl der fehlerfreie als auch der fehlerbehaftete Fall betrachtet werden muss. Ein Beispiel für eine solche erweiterte Wahrheitstabelle könnt ihr Tabelle 1 entnehmen.

Tabelle 1 Erweiterte Wahrheitstabelle eines OR-Gatters (Unterschiede zum fehlerfreien Ausgangsverhalten sind orange markiert)

Fehlerfreier Eingang A	Fehlerfreier Eingang B	Ausgang C, wenn			
		A-fest-auf-0	A-fest-auf-1	C-fest-auf-0	C-fest-auf-1
0	0	0	1	0	1
0	1	1	1	0	1
1	0	0	1	0	1
1	1	1	1	0	1

In Aufgabenteil 1.1 habt ihr die Wahrheitstabellen eines NAND- und eines XOR-Gatters erstellt, die jeweils zwei Eingänge und einen Ausgang haben. Gebt nun jeweils die erweiterte Wahrheitstabellen dieser Gatter für den Fall an, dass einzelne Hafffehler an einem Eingang oder am Ausgang auftreten. Markiert die Unterschiede zu den Wahrheitstabellen aus dem Aufgabenteil 1.1.

**Tipp:** Verwendet die Tabelle 2 als Vorlage für die Lösung.

Tabelle 2 Vorlage für die Lösung der Aufgabe 1.3.a

Fehlerfreier Eingang A	Fehlerfreier Eingang B	Ausgang C, wenn			
		A-fest-auf-0	A-fest-auf-1	C-fest-auf-0	C-fest-auf-1

## 1.3.b D-Algorithmus

In Aufgabe 1.1.b habt ihr errechnet, wie lange ein Test (eines Multiplizierers) dauern kann, wenn man die interne Schaltung nicht kennt. Ein solcher Test ist aus zeitlichen Gründen in der Praxis nicht anwendbar. Da den Herstellern von Mikrochips die Funktion ihrer Schaltung bekannt ist, kann dieses Wissen beim Testen genutzt werden. Es wird daher immer ein sog. struktureller Test durchgeführt, bei dem das Wissen über die Funktion der Schaltung genutzt wird. Das Zusatzinformationsblatt erklärt in dem Abschnitt „Testmustererzeugung mit dem D-Algorithmus“ ein mögliches Verfahren für solch einen strukturellen Test.

Wir wollen nun die folgende (noch) fehlerfreie Schaltung testen, die in Abbildung 5 dargestellt ist. Die Schaltung besitzt die drei Eingänge A, B, C und zwei Ausgänge Y, Z. Dazu verbinden verschiedene interne Signale (d bis p) einfache Gatter.

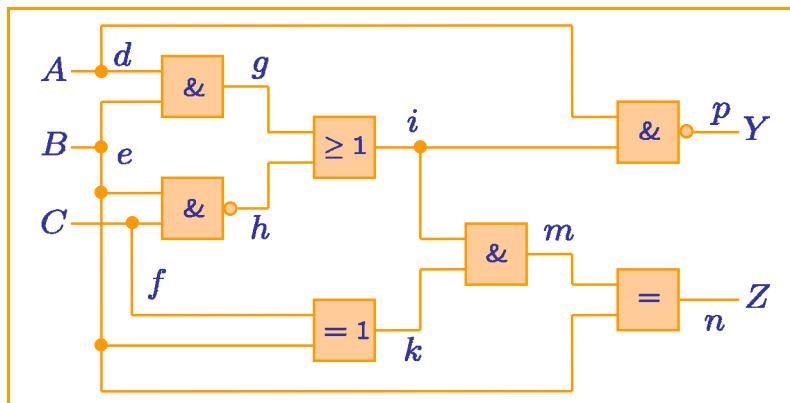


Abbildung 5 Zu testende Schaltung

Findet die folgenden angenommenen Fehler in der Schaltung. Benutzt für die ersten beiden Fragen den D-Algorithmus (vgl. Zusatzblatt).

- 1) Findet ein Testmuster für den Fehler, dass das Signal fehlerhaft **k-fest-auf-1** gehalten wird! (Ein sogenannter 1-Haftfehler, oder stuck-at-1 Fehler.) Welche Werte müssen an den Eingängen anliegen? Welche Werte seht ihr an den Ausgängen im fehlerfreien Zustand?
- 2) Findet ein Testmuster für den Fehler, dass das Signal fehlerhaft **i-fest-auf-0** gehalten wird. Welche Werte müssen an den Eingängen anliegen? Welche Werte seht ihr an den Ausgängen im fehlerfreien Zustand?

Schickt uns zu den folgenden Fragen jeweils einen Screenshot der Waveform.

- 3) Simuliert mit GHDL/GTKWave (siehe Aufgabe 1.2) die Schaltung, die in Abbildung 5 dargestellt ist. Die Schaltung und die zugehörige Testumgebung findet ihr als „KombSchaltung.vhdl“ und „KombSchaltung\_tb.vhdl“ im Unterverzeichnis „c:lilca1\hdl“.
- 4) Nun editiert die Schaltung derart, dass das Signal **k-fest-auf-1** gehalten wird. Überprüft, ob das Ergebnis der Waveform mit dem Ergebnis aus Teil 1) übereinstimmt. Markiert die Stelle in der Waveform, an der eure Testmuster angelegt werden.
- 5) Nehmt dann erneut die fehlerfreie Schaltung, setzt **i-fest-auf-0** und überprüft wieder, ob das Waveform mit dem Ergebnis aus Teil 2) übereinstimmt. Markiert die Stelle in der Waveform, an der eure Testmuster angelegt werden.

## 1.3.C Fehlersuche

Neben dem Detektieren, dass ein Fehler überhaupt existiert, ist es oft auch wichtig, herauszufinden, wo genau der Fehler aufgetreten ist. Dieses ist notwendig, um eventuelle Herstellungsprobleme finden zu können. In der beiliegenden Waveform „test\_output\_diagnosis.vcd“ im Unterverzeichnis „c:lilca1\hdl“ sowie in Abbildung 6 ist das Ergebnis eines vollständigen Tests der obigen Schaltung dargestellt, wobei das Adjektiv „vollständig“ auf die Tatsache hindeutet, dass alle möglichen Eingangskombinationen bei dem Test verwendet werden. Es ist bekannt, dass nur genau ein Haftfehler aufgetreten ist. Die gegebene Waveform stimmt also nicht mit der simulierten fehlerfreien Schaltung (vgl. vorangegangene Simulation) überein.

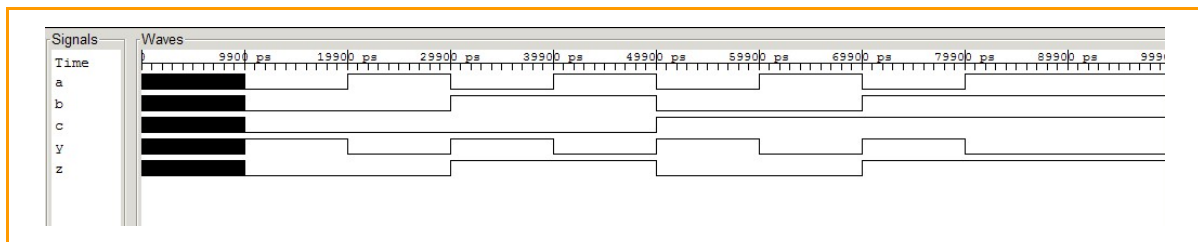


Abbildung 6 Ergebnis eines vollständigen Tests

Welches Signal (d,e,f,g,h,i,k,m,n,p) hatte einen Haftfehler und welcher Art war der Haftfehler? Erläutert in 2-3 Sätzen euren Lösungsweg.

**Form der Lösung:**

Teil 1.3.a: Jeweils eine erweiterte Wahrheitstabelle des NAND- und XOR-Gatters

Teil 1.3.b: Für die Fragen 1 - 2 Auflistung der Werte der Signale A, B, C, Y, Z, d, e, f, g, h, i, k, m, n, p

Für die Fragen 3 - 5 Screenshot der Waveform

Teil 1.3.c: Antworten auf die Frage

## 1.4.a Ausbeute

Da bei der Herstellung eines Mikrochips Defekte auftreten können, muss der Mikrochip getestet werden. Allerdings ist der Test auch mit Kosten verbunden. Dabei sind fixe Kosten für die Testgeräte genauso zu berücksichtigen wie „Kosten pro Mikrochip“. Des Weiteren ist wichtig, wie die Kosten aller hergestellten Mikrochips auf die Preise für die verkauften, funktionierenden Mikrochips verteilt werden können.

In der Abbildung unten ist schematisch ein runder Silizium-Wafer dargestellt. Aus diesem Wafer werden quadratische Mikrochips gefertigt, wie durch das gegebene Raster angedeutet. Bei der Herstellung sind einzelne Defekte aufgetreten, die durch blaue Punkte markiert sind.

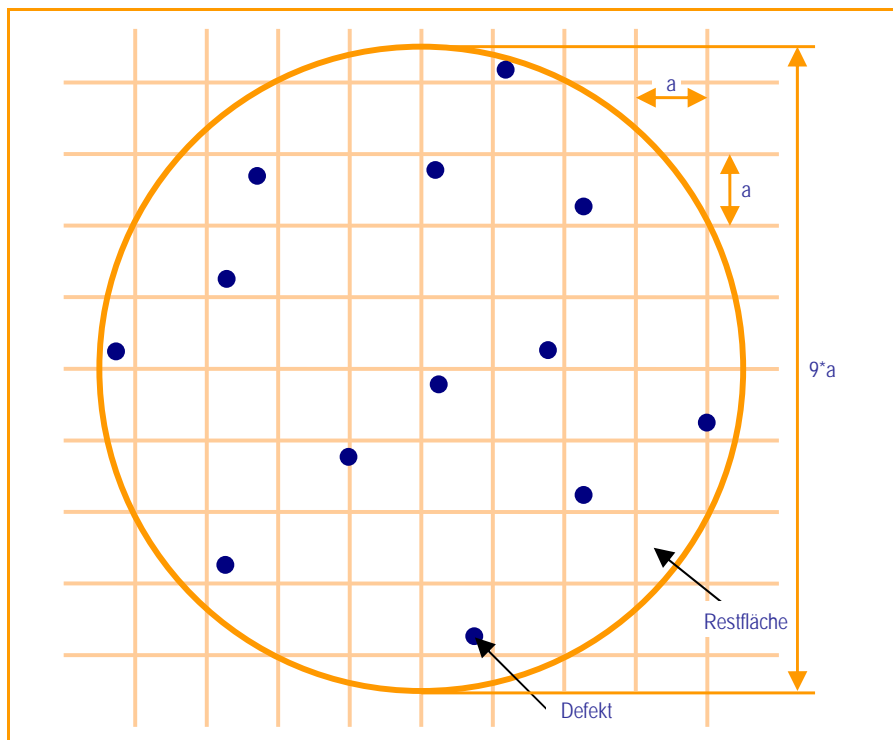


Abbildung 7 Wafer mit Defekten

- 1) Wie viele funktionierende Mikrochips erhält man? (Beachtet, dass Teilflächen keine funktionierenden Mikrochips bedeuten!)
- 2) Ermittelt, wie viele funktionierende Mikrochips ihr erhaltet, wenn die Chipfläche geviertelt wird (z.B. durch einen neuen Halbleiterprozess). Dazu könnt ihr einfach in das gegebene Raster zusätzliche vertikale und horizontale Linien einzeichnen. Wie viele funktionierende Mikrochips erhält man für diese verkleinerte Chip-Fläche?
- 3) Berechnet für beide Fälle den Anteil der Fläche der funktionierenden Mikrochips zur Gesamtfläche des Wafers in Prozent. Der Durchmesser des Wafers ist, wie dargestellt, genau 9-mal die Länge eines Mikrochips aus dem Teil 1) ( $9 \cdot a$ ).

## 1.4.b Bewertung des Tests

Bei einem fiktiv angenommenen Herstellungsprozess sind 20% der hergestellten Mikrochips defekt, z.B. aufgrund von Verunreinigungen. Angenommen die Qualität des ersten Tests, bei dem die Mikrochips noch auf dem Wafer getestet werden, ist 90%, d.h. 10% der getesteten Mikrochips werden falsch getestet.

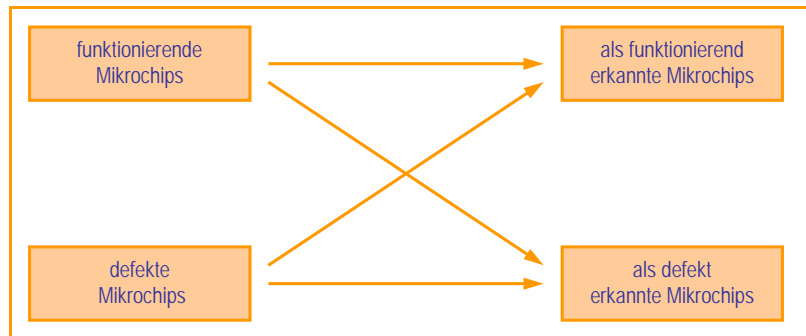


Abbildung 8 Ergebnis des Tests

- 1) Gebt die Wahrscheinlichkeit für jeden der vier Übergänge an, die in Abbildung 8 durch die Pfeile symbolisiert sind.
- 2) Wie viel Prozent der hergestellten Mikrochips werden in diesem ersten Test als "fehlerfrei" markiert?
- 3) Wie viel Prozent der hergestellten Mikrochips sind eigentlich defekt und werden zunächst weiterverarbeitet?
- 4) Wie viel Prozent der hergestellten Mikrochips sind eigentlich funktionierend, werden aber in diesem ersten Test als "defekt" erkannt und aussortiert?

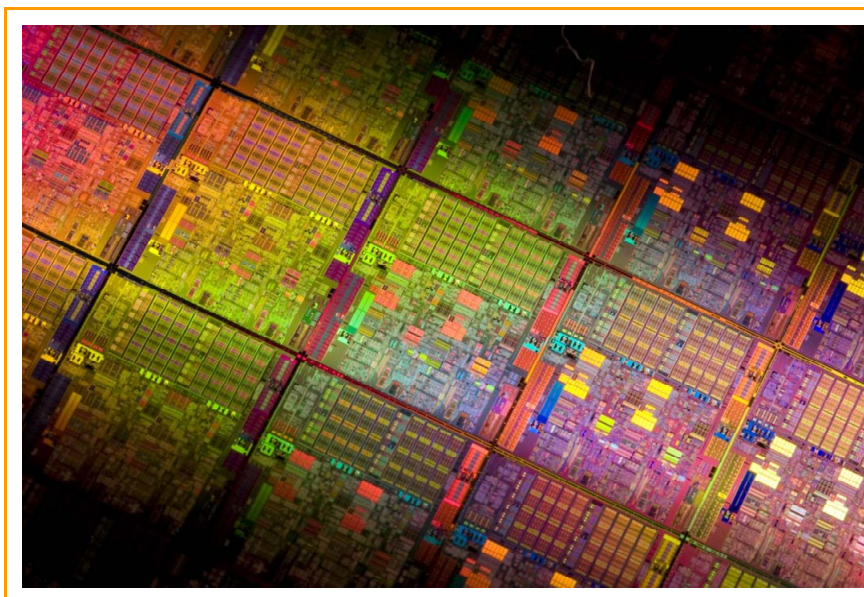


Abbildung 9 Blick auf den neuen Intel-Mehrkernprozessor Core i3. Teilbereiche dieses Mikrochips wurden von Intel-Mitarbeitern in Braunschweig verifiziert.

## 1.4.C Verbesserung des Tests

Nehmen wir an, die Herstellung jedes Mikrochips aus Teil 1.4.b (unabhängig, ob defekt oder nicht) kostet 10,00 EUR. Der erste Test des Mikrochips hat eine Qualität von 90% und kostet 1,00 EUR je Mikrochip. Die als gut getesteten Mikrochips werden dann in ein Produkt (z.B. Mainboard, Kaffeemaschine, DVD-Player, ...) eingesetzt. Die als schlecht getesteten Mikrochips werden entsorgt. (Leider werden auch einige Mikrochips entsorgt, die funktionieren, aber fälschlicherweise als defekt getestet werden. Es gelten die Wahrscheinlichkeiten aus Teil 1.4.b). Das fertige Produkt wird dann durch einen finalen Produkttest geprüft. Die Herstellung des Produkts sowie der Produkttest sind perfekt (d.h. es entstehen bei der Produktherstellung keine weiteren Defekte, alle defekten Produkte werden durch den Produkttest gefunden und es fallen keine weiteren Testkosten pro Produkt an). Allerdings entstehen bei defekt gefundenen Produkten, die erst im Produkttest erkannt werden, zusätzliche Kosten von 30,00 EUR.

Beantwortet folgende Fragen

- 1) Die Kosten pro hergestellten Mikrochip betragen 11,00 EUR (10,00 für Herstellung und 1,00 EUR für den ersten Test). Wie groß sind die Kosten pro hergestellten Mikrochip, wenn man die zusätzlichen Kosten beim Produkttest einberechnet?
- 2) Wie groß sind die Kosten pro funktionierendes Produkt, das verkauft wird? Bedenkt, dass alle Kosten auf die funktionierenden Produkte umgelegt werden und dass nur die Mikrochips letztlich im Produkt verkauft werden, die den ersten Mikrochiptest und den Produkttest bestanden haben.
- 3) Der Hersteller überlegt, ob er Geld investieren sollte, um die Testqualität des ersten Tests von 90% auf 95% zu erhöhen. Einmalige Kosten für neue Testgeräte wären 2 Millionen EUR. Die Testkosten pro Mikrochip würden sich auf 1,50 EUR erhöhen. Nach wie vielen produzierten Mikrochips rentiert sich die Investition?

**Form der Lösung:**

Teil 1.4.a: Antworten auf die Fragen 1 - 3

Teil 1.4.b: Antworten auf die Fragen 1 - 4

Teil 1.4.c: Antworten auf die Fragen 1 - 3 inkl. Rechenweg

## Wichtige Informationen

Falls ihr Fragen zu den Aufgaben habt oder eine Hilfestellung benötigt, so schaut doch einfach in unser Forum:  
<http://www.intel-leibniz-challenge.de/forum/>

Abgabe der Lösungen

**Wo:** [www.intel-leibniz-challenge.de/portal](http://www.intel-leibniz-challenge.de/portal)

**Wie:** **Zulässige Dateiformate:**

Textformate: PDF mit eingebetteten Bildern, txt

Bildformate: jpg, bmp, png

Videoformate: flv, avi, mpg

### **Dateigrößen und Dateibenennung**

Die Dateien sollten nicht größer als 7,5 MB sein! (Die Dateien können gezippt sein.)  
Bitte gebt in der Datei (nicht im Dateinamen) auch euren Teamnamen, die Namen der Gruppenmitglieder sowie deren Schulen an.

Bitte benennt eure Dateien mit dem Teamnamen und dem Aufgabenteil.

**Wann:** Bis zum **28.02.2010** um **23:59** Uhr

**Hinweis:** Um sicher zu gehen, dass eure Dateien wirklich fehlerfrei und für die Korrektoren zu öffnen sind, solltet ihr eure Zip-Dateien etc. nochmals von eurem Account runterladen und öffnen. **Dateien, die sich nicht öffnen lassen, können nicht bewertet werden!**

Die Teilnahmebedingungen und weitere Informationen: [www.intel-leibniz-challenge.de](http://www.intel-leibniz-challenge.de)  
Der Rechtsweg ist ausgeschlossen!