

1.1. Musterlösung

1.1.a Grundlagen

In dem Lückentext sind folgende Begriffe richtig:

- [1] Große funktionale Schaltungsblöcke werden in **kleinere/mehrere** Blöcke aufgeteilt um die Komplexität zu reduzieren.
- [2] Zwei der populärsten HDLs für den Entwurf digitaler Schaltungen sind VHDL und **Verilog/System Verilog**.
- [3] Die Schaltungsbeschreibung mittels HDL ermöglicht nicht nur eine Simulation der Schaltung sondern auch die automatische Umsetzung in eine sog. Netzliste. Dieser Vorgang wird **Synthese/Schaltungssynthese** genannt.
- [4] Zu deren wesentlichen Merkmalen gehören im Fall einer digitalen Schaltung neben der maximal möglichen **Taktfrequenz** auch die benötigte Silizium-Fläche sowie die Verlustleistung.
- [5] Das Resultat der physikalischen Implementierung ist wiederum eine Datei, die notwendige Informationen zur **Herstellung/Produktion/Fertigung** des Mikrochips in einer Fabrik enthält.
- [6] Die Abfolge der aufeinander aufbauenden Entwicklungsschritte, die durch den Einsatz moderner EDA-Software ermöglicht bzw. bestimmt wird, bezeichnet man im Englischen als **Design Flow**.
- [7] Moderne Mikrochips wie der Intel Core i3-540 Mikroprozessor werden in CMOS Technologie mit Strukturbreiten von nur **32 nm** produziert.
- [8] / [9] Beim Herstellungsprozess werden mehr als 200 einzelne Prozessschritte durchlaufen, bevor die feinen Chip-Strukturen in einer Vielzahl übereinanderliegender Schichten auf der Grundplatte aus hochreinem **Silizium**, dem sog. **Wafer**, aufgebracht sind.
- [10] Je früher mögliche Entwurfsfehler erkannt werden, desto **günstiger/besser** ist dies in Bezug auf die Entwicklungskosten.

1.1.b Grundlagen

- 1) Ein Bit kann genau **2** unterschiedliche Werte annehmen. Ein Byte enthält **8 Bits** und kann $2^8 = 256$ unterschiedliche Werte annehmen (00000000, 00000001, ..., 11111111).
- 2) Das *Binärzahlensystem* bzw. *Binärsystem* für die Darstellung von Zahlen ist wie das Dezimalsystem aufgebaut. Jede Zahl besteht aus Ziffern, die einen Wert und einen Stellenwert aufweisen. In diesem System kann jedoch jede *Ziffer* lediglich die Werte „0“ und „1“ annehmen. Die Stellenwerte sind die Potenzwerte der Basiszahl 2. Die Umwandlung wird somit wie folgt durchgeführt:

$$10010110_2 = 1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$$

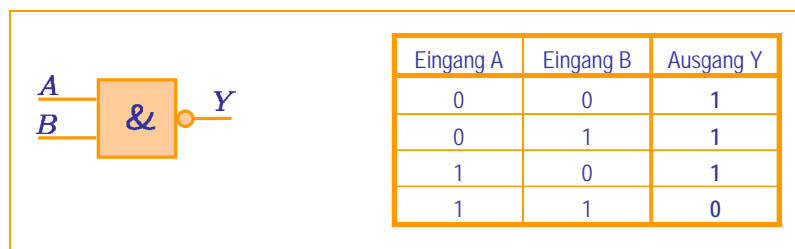
$$= 150_{10}$$

$$01011101_2 = 0 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$$

$$= 93_{10}$$

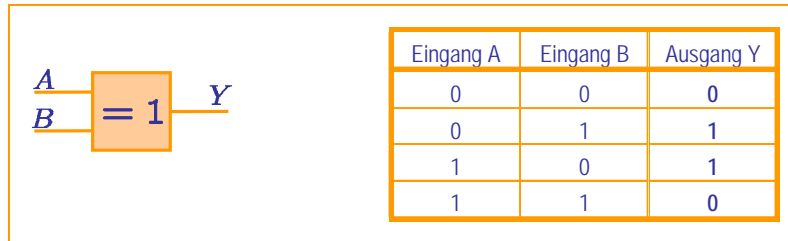
Hinweis: Als Beschreibung der Vorgehensweise bei der Umwandlung der Zahlen wird sowohl eine textuelle Beschreibung als auch die mathematische Berechnung (s. oben) akzeptiert.

- 3) Wahrheitstabelle eines NAND-Gatters

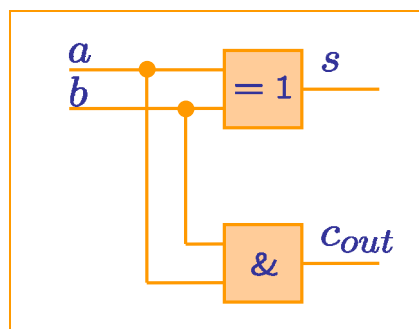


1.1. Musterlösung

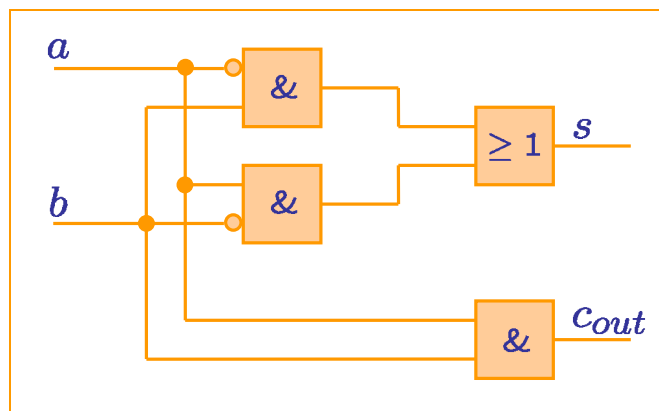
XOR-Gatter (Der Ausgang hat den Wert 1 wenn die Summe der Eingangswerte ebenfalls 1 ist)



4) Ein Halbaddierer kann mit zwei Gattern realisiert werden:



Es ist ebenfalls möglich den Halbaddierer mit vier Gattern zu realisieren:



- 5) Bei 128 Eingängen gibt es 2^{128} mögliche Kombinationen. Dies entspricht dem ungefähren Wert von $3,4 \cdot 10^{38}$.
- 6) Da jede Multiplikation innerhalb eines Taktes ausgeführt wird, werden für sie $0,5 \text{ ns}$ ($\frac{1}{2 \text{ GHz}} = \frac{1}{2 \cdot 10^9 \text{ Hz}}$) benötigt. Da 2^{128} Multiplikationen ausgeführt werden müssen, dauert der vollständige Test $1,7 \cdot 10^{29}$ sek.:

$$T = 2^{128} * 0,5 \text{ ns} = 2^{128} * 0,5 * 10^{-9} \text{ s} \approx 1,7 * 10^{29} \text{ s}$$

Diese Zeit entspricht ca. $5,4 \cdot 10^{21}$ Jahren. Im Vergleich dazu wird das Alter unseres Universums auf ca. $13 \cdot 10^9$ Jahre geschätzt. Dies bedeutet, dass wenn wir mit dem Test dieser Schaltung beim Urknall angefangen hätten, der vollständige Test noch nicht abgeschlossen wäre. Aus diesem Grund ist der vollständige Test für die meisten Schaltungen nicht möglich.

1.2. Musterlösung

1.2.a Volladdierer

In folgender Abbildung wird der Teil des VHDL-Codes dargestellt, der im Rahmen der Bearbeitung der Aufgabe ergänzt werden musste. Die vollständige VHDL-Datei kann gesondert heruntergeladen werden. Die ursprüngliche Testumgebung (Datei „full_adder_test.vhdl“) enthielt lediglich die Testmuster $\{abc_{in}\}=\{000\}$ und $\{abc_{in}\}=\{100\}$, so dass noch weitere sechs Testmuster ergänzt werden mussten.

```

...
begin
  a <= '0';
  b <= '0';
  cin <= '0';

  wait for 10 ns;
  a <= '1';
  b <= '0';
  cin <= '0';

  wait for 10 ns;
  a <= '0';
  b <= '1';
  cin <= '0';

  wait for 10 ns;
  a <= '1';
  b <= '1';
  cin <= '0';

  wait for 10 ns;
  a <= '0';
  b <= '0';
  cin <= '1';

  wait for 10 ns;
  a <= '1';
  b <= '0';
  cin <= '1';

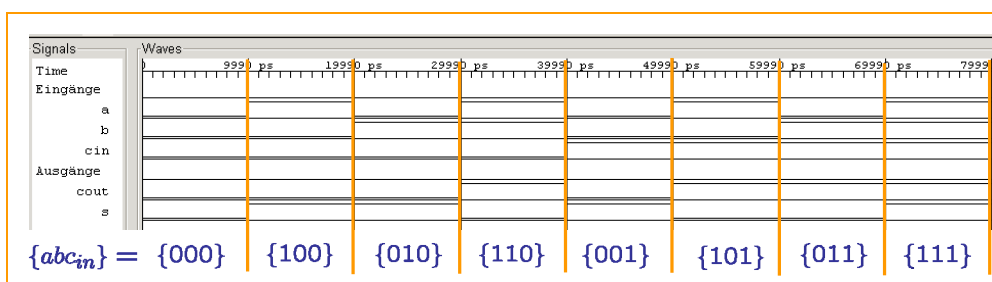
  wait for 10 ns;
  a <= '0';
  b <= '1';
  cin <= '1';

  wait for 10 ns;
  a <= '1';
  b <= '1';
  cin <= '1';

  assert false report "end of test" severity note;
...

```

Das Ergebnis der vollständigen Simulation wird durch nachfolgende Abbildung dargestellt, wobei hier deutlich wird, dass an die Schaltung acht unterschiedliche Testmuster angelegt werden.



1.2. Musterlösung

1.2.b Komplexe Schaltung

- 1) In folgender Abbildung wird der Teil des VHDL-Codes dargestellt, der im Rahmen der Bearbeitung der Aufgabe ergänzt werden musste. Die vollständige VHDL-Datei kann gesondert heruntergeladen werden. Die ursprüngliche Testumgebung (Datei „*schaltung_test.vhdl*“) enthielt lediglich die Testmuster $a=\{0000\}$ und $a=\{0001\}$, sodass noch weitere 14 Testmuster ergänzt werden mussten (Hinweis: Die Testumgebung musste nicht abgegeben werden.).

```
...
a <= "0000";
wait for 20 ns;

a <= "0001";
wait for 20 ns;

a <= "0010";
wait for 20 ns;

a <= "0011";
wait for 20 ns;

a <= "0100";
wait for 20 ns;

a <= "0101";
wait for 20 ns;

a <= "0110";
wait for 20 ns;

a <= "0111";
wait for 20 ns;

a <= "1000";
wait for 20 ns;

a <= "1001";
wait for 20 ns;

a <= "1010";
wait for 20 ns;

a <= "1011";
wait for 20 ns;

a <= "1100";
wait for 20 ns;

a <= "1101";
wait for 20 ns;

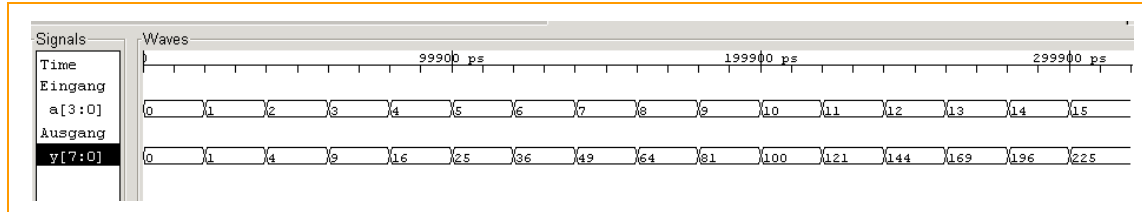
a <= "1110";
wait for 20 ns;

a <= "1111";
wait for 20 ns;

assert false report "end of test" severity note;
-- Wait forever; this will finish the simulation.
wait;
...
```

1.2. Musterlösung

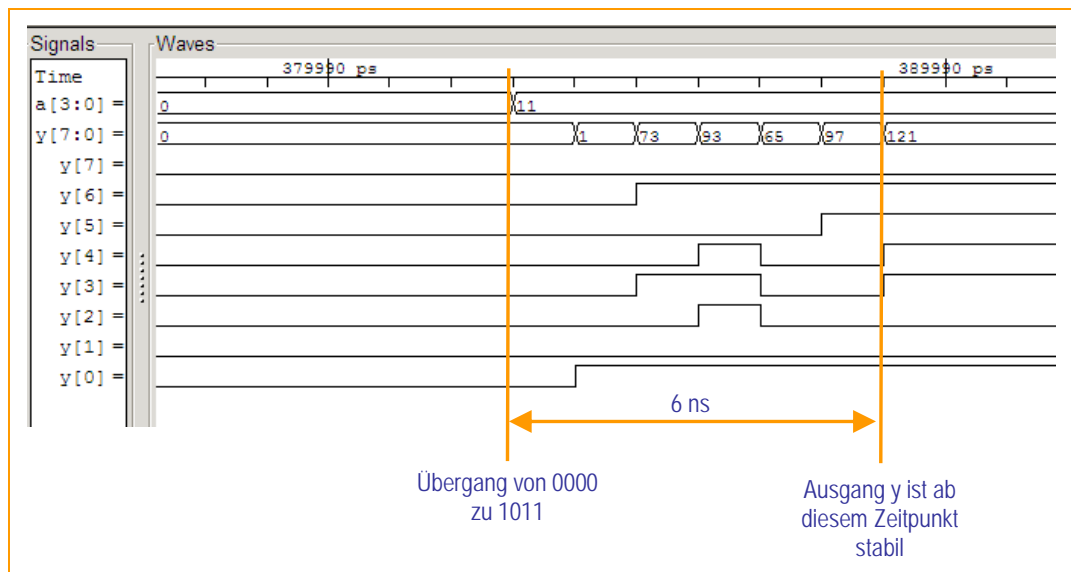
Die Simulation der komplexen Schaltung liefert folgendes Ergebnis:



- 2) Die Schaltung realisiert die quadratische Funktion $y = f(a) = a^2$. Dies kann man beispielsweise an beliebigen Stellen der Waveform ablesen (z. B. $a=5 \rightarrow y=25$; $a=12 \rightarrow y=144$).

1.2.c Timing

- 1) Der Übergang des Eingangssignals von $a="0000"$ (dezimal „0“) auf $a="1011"$ (dezimal „11“) verursacht eine Veränderung des Ausgangssignals y von $y="00000000"$ (dezimal: „0“) auf $y="01111001"$ (dezimal „121“). Die letzte Veränderung des Ausgangssignals y (y_3 und y_4) erfolgt **6 ns** nach der Änderung des Eingangssignals.

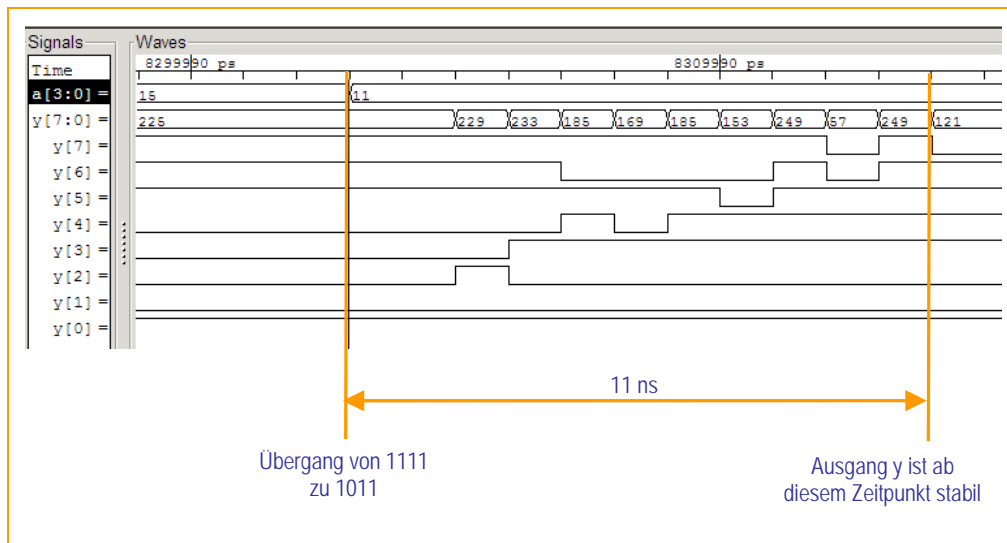


Die einzelnen Bits nehmen nach folgenden Zeiten ihren richtigen Ausgangswert an:

Ausgangsbit	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7
Zeit	1ns	0ns	4ns	6ns	6ns	5ns	2ns	0ns

- 2) Der Übergang des Eingangssignals von $a="1111"$ (dezimal „15“) auf $a="1011"$ (dezimal „11“) verursacht eine Veränderung des Ausgangssignals y von $y="11100001"$ (dezimal „225“) auf $y="10011110"$ (dezimal „121“). Die letzte Veränderung des Ausgangssignals y (y_7) erfolgt **11 ns** nach der Änderung des Eingangssignals.

1.2. Musterlösung



Die einzelnen Bits nehmen nach folgenden Zeiten ihren richtigen Ausgangswert an:

Ausgangsbit	y ₀	y ₁	y ₂	y ₃	y ₄	y ₅	y ₆	y ₇
Zeit	0ns	0ns	3ns	3ns	6ns	8ns	10ns	11ns

- 3) Die maximale Taktfrequenz einer Schaltung wird durch den langsamsten Übergang bestimmt, da die Werte nur dann übernommen werden können, wenn die Berechnung vollständig abgeschlossen ist. Die Werte werden dabei bei der steigenden (oder fallenden) Flanke des Taktes der Schaltung zugefügt und von der Schaltung entnommen. Die maximale Taktfrequenz beträgt somit **90,9 MHz**:

$$f_{Takt,max} = \frac{1}{11ns}$$

1.3. Musterlösung

1.3.a Fehlermodell

Für das NAND-Gatter ergibt sich folgende erweiterte Wahrheitstabelle:

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	1	1	0	1
0	1	1	0	0	1
1	0	1	1	0	1
1	1	1	0	0	1

Für das XOR-Gatter ergibt sich folgende erweiterte Wahrheitstabelle:

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	0	0	1
1	0	0	1	0	1
1	1	1	0	0	1

1.3.b D-Algorithmus

- 1) Der Fehler k-fest-auf-1 kann durch drei Testmuster entdeckt werden. Es handelt sich dabei um $\{ABC\} = \{000\}, \{100\}, \{111\}$. Die gesamte Wahrheitstabelle, die in der Aufgabe nicht gefordert war, bestätigt diese Aussagen, wobei hier die drei richtigen Antworten fett dargestellt sind:

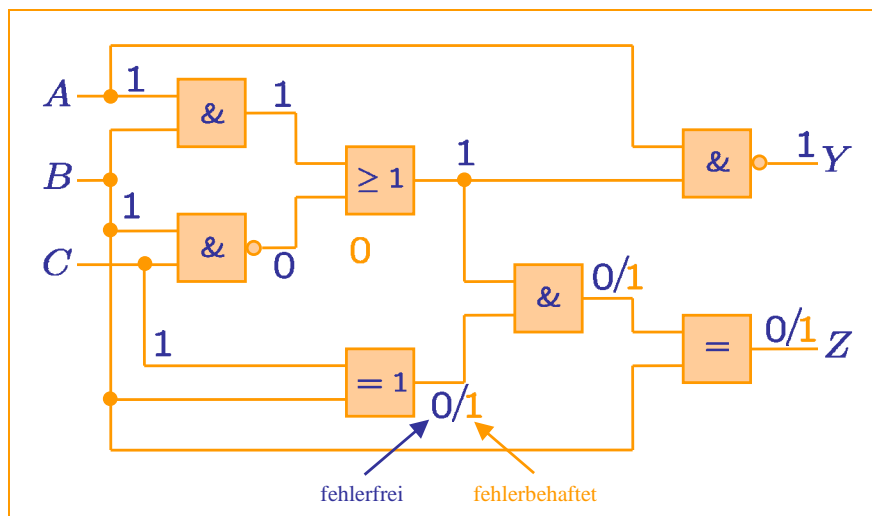
Eingänge			Ausgänge (fehlerfreier Zustand der Schaltung)		Ausgänge (fehlerbehafteter Zustand der Schaltung)	
A	B	C	Y	Z	Y	Z
0	0	0	1	1	1	0
0	0	1	1	0	1	0
0	1	0	1	1	1	1
0	1	1	1	0	1	0
1	0	0	0	1	0	0
1	0	1	0	0	0	0
1	1	0	0	1	0	1
1	1	1	0	0	0	1

Die Testmuster werden mithilfe des D-Algorithmus wie folgt bestimmt, wobei diese Beschreibung nicht abgeben werden musste:

- Um den Fehler am Ausgang Z beobachten zu können, muss zunächst das Gatter mit dem Ausgang m „durchgeschaltet“ werden. Da es sich um ein UND-Gatter handelt, muss das Signal i den Wert 1 erhalten.
- Dies ist der Fall, wenn entweder Signal g ODER h den Wert 1 haben. Aufgrund dieser Notwendigkeit ergeben sich für die Eingänge $\{ABC\}$ folgende mögliche Testmuster: $\{110\}, \{111\}, \{000\}, \{100\}, \{101\}, \{001\}, \{010\}$.
- Nun muss das Gatter mit dem Ausgang n „durchgeschaltet“ werden. Hier ist es mit jeder beliebigen Belegung der Eingänge, den Fehler sichtbar zu machen, sodass hier keine einschränkende Auswahl der o. g. Testmuster vorgenommen werden muss.

1.3. Musterlösung

- iv) Mit den vorausgehenden Schritten stellen wir sicher, dass der Fehler am Ausgang beobachtbar ist. Jetzt muss der Fehler eingestellt werden. Dies bedeutet, dass das Signal k einen zum Fehler entgegengesetzten Wert erhalten muss, damit im Fehlerfall der Fehler sich bemerkbar machen kann. Da es sich bei dem Fehler um k -fest-auf-1 handelt, muss das Signal k den Wert 0 erhalten. Dies ist gegeben, wenn beide Eingänge des Exklusiv-ODER-Gatters identische Werte aufweisen, sodass folgende Testmuster in Frage kommen: {000}, {100}, {011}, {111}.
- v) Da der Fehler sowohl einstellbar als auch beobachtbar sein muss, werden schließlich aus den in Punkt ii) und iv) gefundenen Testmuster die gemeinsamen Testmuster ausgewählt: {000}, {100}, {111}. Folgende Abbildung stellt die Werte der einzelnen Signale für das Testmuster $\{ABC\} = \{111\}$ dar:



- 2) Der Fehler i -fest-auf-0 kann durch sechs Testmuster entdeckt werden. Es handelt sich dabei um $\{ABC\} = \{001\}, \{010\}, \{100\}, \{101\}, \{110\}, \{111\}$. Die gesamte Wahrheitstabelle, die in der Aufgabe nicht gefordert war, bestätigt diese Aussagen, wobei hier die richtigen Antworten fett dargestellt sind:

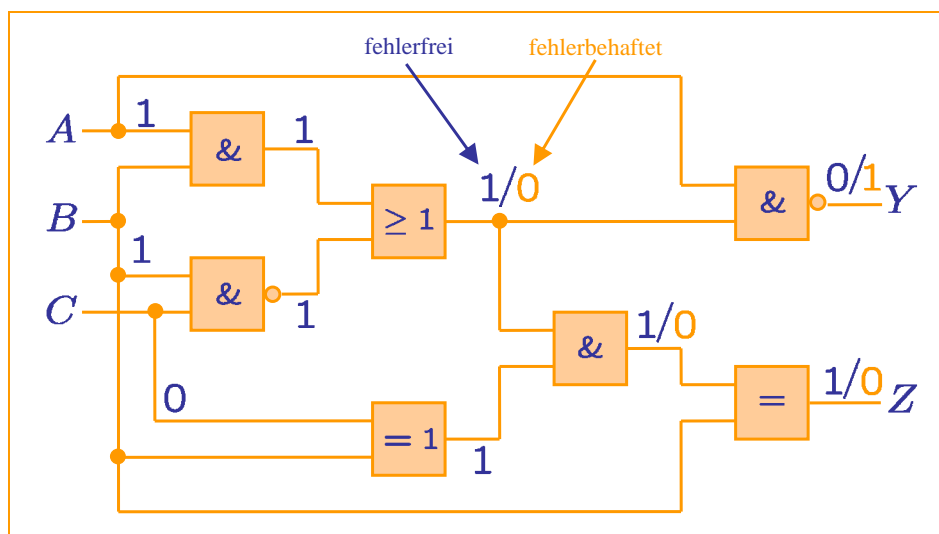
Eingänge			Ausgänge (fehlerfreier Zustand der Schaltung)		Ausgänge (fehlerbehafteter Zustand der Schaltung)	
A	B	C	Y	Z	Y	Z
0	0	0	1	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	1	0
0	1	1	1	0	1	0
1	0	0	0	1	1	1
1	0	1	0	0	1	1
1	1	0	0	1	1	0
1	1	1	0	0	1	0

Die Testmuster werden mithilfe des D-Algorithmus wie folgt bestimmt, wobei diese Beschreibung nicht abgegeben werden musste:

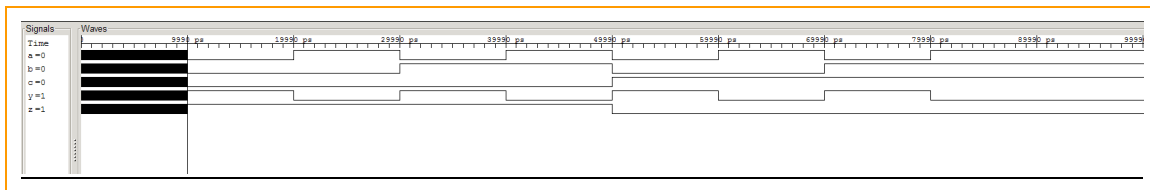
- i) Um den Fehler am Ausgang Y beobachten zu können, muss zunächst das Gatter mit dem Ausgang p „durchgeschaltet“ werden. Da es sich um ein NAND-Gatter handelt, muss das Signal d den Wert 1 erhalten.
- ii) Der Eingang A muss somit den Wert 1 haben, sodass folgende Testmuster in Frage kommen: {100}, {101}, {110}, {111}

1.3. Musterlösung

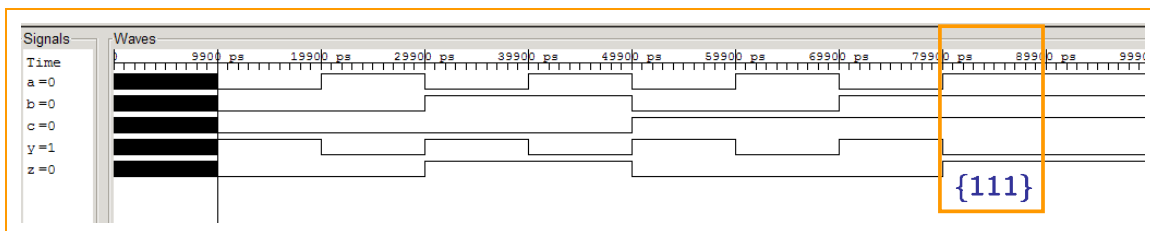
- iii) Mit den vorausgehenden Schritten stellen wir sicher, dass der Fehler am Ausgang Y beobachtbar ist. Jetzt muss der Fehler eingestellt werden. Dies bedeutet, dass das Signal i einen zum Fehler entgegengesetzten Wert erhalten muss, damit im Fehlerfall der Fehler sich bemerkbar machen kann. Da es sich bei dem Fehler um i-fest-auf-0 handelt, muss das Signal i den Wert 1 erhalten. Dies ist gegeben, wenn mindestens ein Eingang des ODER-Getters den Wert 1 hat, sodass folgende Testmuster in Frage kommen: $\{110\}$, $\{111\}$, $\{000\}$, $\{100\}$, $\{001\}$, $\{101\}$.
- iv) Da der Fehler sowohl einstellbar als auch beobachtbar sein muss, werden schließlich aus den in Punkt ii) und iii) gefundenen Testmuster die gemeinsamen Testmuster ausgewählt: $\{110\}$, $\{111\}$, $\{100\}$, $\{101\}$. Man kann selbstverständlich den Fehler am Ausgang Z beobachtbar machen, sodass sich weitere Testmuster ergeben. Folgende Abbildung stellt die Werte der einzelnen Signale für das Testmuster $\{ABC\} = \{110\}$ dar:



- 3) Die Simulation der fehlerfreien Schaltung liefert folgende Ergebnisse:

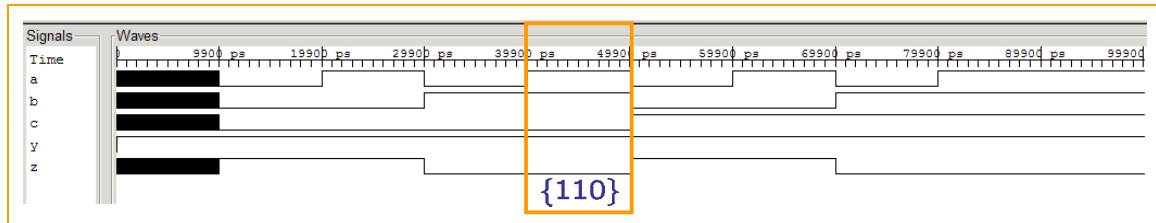


- 4) Um den Fehler „k-fest-auf-1“ in die Schaltung einzubauen muss in der Datei „KombSchaltung.vhdl“ die Zeile „k <= f xor e;“ durch „k <= '1';“ ersetzt werden. Die Simulation bestätigt, dass der Fehler bei den o. g. Testmuster am Ausgang sichtbar ist, wobei das Testmuster $\{ABC\} = \{111\}$ markiert ist.



1.3. Musterlösung

- 5) Um den Fehler „i-fest-auf-0“ in die Schaltung einzubauen muss in der Datei „KombSchaltung.vhdl“ die Zeile „i <= g or h;“ durch „i <= '0';“ ersetzt werden. Die Simulation bestätigt, dass der Fehler bei den o. g. Testmuster am Ausgang sichtbar ist, wobei das Testmuster {ABC} = {110} markiert ist.



1.3.c Fehlersuche

Innerhalb der Schaltung ist der Fehler **m-fest-auf-1** eingebaut. Es ist möglich diesen Fehler durch unterschiedliche Betrachtungen zu entdecken, sodass hier zwei Möglichkeiten vorgestellt werden:

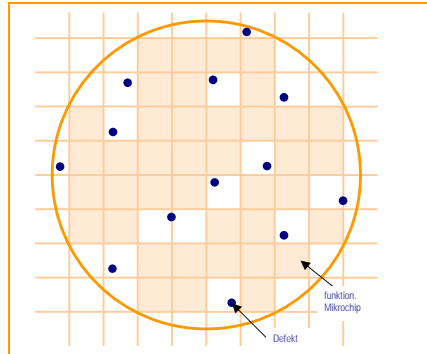
Der Fehler kann beispielsweise wie folgt bestimmt werden: Ausgang Y ist nicht vom Fehler betroffen. Daher sind Signale *p* und *d* nicht fehlerhaft. Signale *i, g, h, f, e, d* sind auch nicht möglich, da sie auch auf den Ausgang Y Auswirkungen hätten. Es bleiben mit *k, m, n* nur wenig Möglichkeiten. *n* scheidet aus, da dieses Signal nicht für alle Kombinationen konstant 0 oder 1 ist. Überdies kann man erkennen, dass der Ausgang Z immer identisch dem Eingang B ist, sodass bei einem XNOR-Gatter der eine Eingang fest-auf-1 sein muss.

Es wäre natürlich auch denkbar, alle Fehler nacheinander in die Schaltung einzubauen und durch eine Simulation die Auswirkungen zu überprüfen. Bereits in dieser Schaltung gibt es nach unserem Ansatz (Signal-fest-auf-0 oder Signal-fest-auf-1) 20 mögliche Fehlerursachen. In einer komplexeren Schaltung wäre eine solche Vorgehensweise nicht mehr möglich.

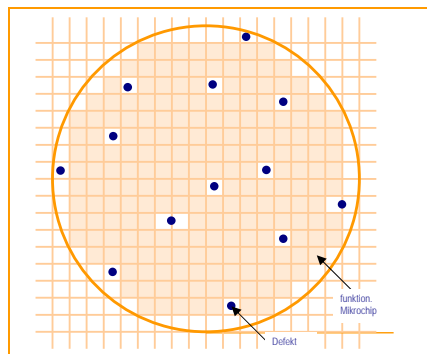
1.4. Musterlösung

1.4.a Ausbeute

- 1) Aufgrund der dargestellten Defekte sind noch **40** Mikrochips funktionierend:



- 2) Aufgrund der dargestellten Defekte und kleiner Abmessungen der Mikrochips sind noch **204** Mikrochips funktionierend:



- 3) Die gesamte Fläche des Wafers beträgt:

$$A_{Waf\text{er}} = \pi \left(\frac{9}{2}a\right)^2 = \frac{81}{4}\pi a^2$$

Bei den größeren Mikrochips mit der Fläche a^2 ist der Anteil der funktionierenden Mikrochips **62,88%**:

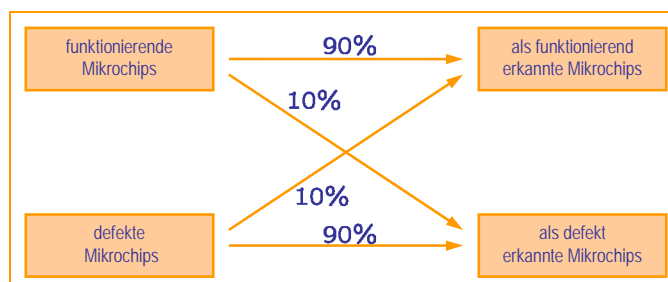
$$\frac{A_{Mikrochips}}{A_{Waf\text{er}}} = \frac{40a^2}{\frac{81}{4}\pi a^2} = \frac{160}{81\pi} \approx 0,6288$$

Bei den kleineren Mikrochips mit der Fläche $0,25a^2$ ist der Anteil der funktionierenden Mikrochips **80,17%**:

$$\frac{A_{Mikrochips}}{A_{Waf\text{er}}} = \frac{204\left(\frac{1}{2}a\right)^2}{\frac{81}{4}\pi a^2} = \frac{51a^2}{81\pi a^2} = \frac{68}{27\pi} \approx 0,8017$$

1.4.b Bewertung des Tests

- 1) Die folgende Abbildung zeigt die Wahrscheinlichkeiten für die einzelnen Übergänge:



1.4. Musterlösung

- 2) Es werden 90% der tatsächlich fehlerfreien Mikrochips (80% der hergestellten Mikrochips) und 10% der defekten Mikrochips (20% der hergestellten Mikrochips) als fehlerfrei markiert, sodass insgesamt **74%** der hergestellten Mikrochips als fehlerfrei markiert werden:

$$n_{\text{fehlerfrei}} = 0,9 * 0,8 + 0,1 * 0,2 = 0,74$$

- 3) Es werden 10% der defekten Mikrochips (20% der hergestellten Mikrochips) als fehlerfrei markiert und weiterverarbeitet, sodass **2%** der hergestellten Mikrochips als fehlerfrei markiert werden, obwohl sie defekt sind:

$$n_{\text{Defekt, fehlerfrei}} = 0,1 * 0,2 = 0,02$$

- 4) Es werden 10% von den funktionierenden Mikrochips (80% der hergestellten Mikrochips) als defekt markiert, sodass **8%** der hergestellten Mikrochips unnötigerweise als defekt markiert werden:

$$n_{\text{fehlerfrei, Defekt}} = 0,1 * 0,8 = 0,08$$

1.4.c Verbesserung des Tests

- 1) Die gesamten Kosten pro Mikrochip ergeben sich aus den Herstellungskosten, Testkosten und Produkttestkosten, wobei die Produktkosten lediglich für die fälschlicherweise als fehlerfrei erkannten Mikrochips entstehen. Aufgrund dieser Zusammensetzung der Kosten kostet bei der Herstellung von n Mikrochips jeder Mikrochip **11,60 EUR**:

$$K_{\text{Mikrochip}} = \frac{n * 10,00 \text{ EUR} + n * 1,00 \text{ EUR} + 0,02 * n * 30,00 \text{ EUR}}{n}$$

- 2) Da ausschließlich fehlerfreie Mikrochips verkauft werden, werden die Gesamtkosten nicht auf alle sondern nur auf die verkauften Mikrochips verteilt. Es ergeben sich als Kosten für jeden verkauften Mikrochip von **16,11 EUR**:

$$K_{\text{Mikrochip}} = \frac{n * 10,00 \text{ EUR} + n * 1,00 \text{ EUR} + 0,02 * n * 30,00 \text{ EUR}}{0,8 * 0,9 * n}$$

- 3) Durch den neuen Test werden 95% der fehlerfreien hergestellten Chips (80%) als fehlerfrei markiert, sodass **76%** der hergestellten Mikrochips verkauft werden. Es ergeben sich somit Kosten für jeden verkauften Mikrochip von **15,53 EUR**:

$$K_{\text{Neu, Mikrochip}} = \frac{n * 10,00 \text{ EUR} + n * 1,50 \text{ EUR} + 0,01 * n * 30,00 \text{ EUR}}{0,8 * 0,95 * n}$$

Nun müssen natürlich auch die Investitionskosten auf die verkauften Mikrochips umgelegt werden, sodass die Kosten steigen:

$$K_{\text{Neu, Mikrochip}} = 15,53 \text{ EUR} + \frac{2000000, - \text{EUR}}{0,8 * 0,95 * n}$$

Die Investition hat sich rentiert, wenn die alten und neuen Kosten identisch sind:

$$K_{\text{Mikrochip}} * n = K_{\text{Neu, Mikrochip}} * n$$

$$16,11 \text{ EUR} * n = 15,53 \text{ EUR} * n + 2000000,00 \text{ EUR}$$

$$n = \frac{2000000,00 \text{ EUR}}{16,11 \text{ EUR} - 15,53 \text{ EUR}}$$

Es müssen somit mindestens **3,42 Millionen** Mikrochips verkauft werden, damit sich die Investition rentiert. Dies bedeutet, dass **4,5 Millionen** Mikrochips hergestellt werden müssen:

$$n_{\text{hers.}} = \frac{3,42 \text{ Mil.}}{0,8 * 0,95}$$