Pfadsystemtest für unate Schaltkreise

Holger Müller

25. September 1996

Diplomarbeit nach einem Thema von Prof. Dr. Günter Hotz
in der Technischen Fakultät, Fachbereich 14 – Informatik,
der Universität des Saarlandes, Saarbrücken

Hiermit versichere ich an Eides Statt, daß ich diese Arbeit nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Saarbrücken, im September 1996

Inhaltsverzeichnis

Einleitung

Heutzutage wird bei elektronischen Schaltungen immer größerer Wert auf eine hohe Betriebsgeschwindigkeit und möglichst geringen Platzbedarf gelegt. Durch diese Effizienzsteigerung wird allerdings auch die Fehleranfälligkeit der Chips erhöht. Somit gewinnt ein Test bei der Fertigung dieser Schaltungen immer mehr an Bedeutung. Nach einer Schätzung von Milne [Mil85] entfallen heute mehr als 25 % der Produktionskosten auf den Test der produzierten Chips.

Der Test eines sequentiellen Schaltkreises, also eines Schaltkreises mit Speicherelementen, kann mit Hilfe eines *Prüfpfades* [EW78] auf den Test eines kombinatorischen Schaltkreises zurückgeführt werden. Aus diesem Grund beschränkt sich diese Arbeit auf den Test von kombinatorischen VLSI-Schaltkreisen. Bei einem Test kann weiterhin unterschieden werden, ob die gefertigte Schaltung bezüglich ihres logischen (statischen) oder temporalen (dynamischen) Verhaltens untersucht werden soll.

Liegt ein statischer Defekt innerhalb einer Schaltung vor, so verändert sich deren logische Ausgabefunktion. Will man einen Schaltkreis auf einen logischen Fehler überprüfen, reicht es schon aus, einen Eingabevektor an die Eingänge der Schaltung anzulegen, für den ein Schaltkreis, der diesen logischen Fehler enthält, eine fehlerhafte Ausgabe liefert. Es wird nun verglichen, ob die Ausgabe des Schaltkreises mit dem spezifizierten Ergebnis übereinstimmt.

Schwieriger ist es hingegen, den Schaltkreis bezüglich seines temporalen Verhaltens zu untersuchen: Bei einem dynamischen Fehler liefert der Schaltkreis zwar die richtige logische Ausgabe, es können allerdings in Abhängigkeit von der Ausgangsbelegung der Leitungen des Schaltkreises die an die Schaltung gestellten Laufzeitbedingungen nicht eingehalten werden. Somit müssen in einem ersten Schritt die Leitungen des Schaltkreises auf geeignete Ausgangswerte initialisiert werden, um dann in einem zweiten Schritt die eigentliche Testeingabe an den Schaltkreis anzulegen. Folglich reicht es für einen Laufzeittest nicht aus, wie bei einem logischen Test nur eine Testeingabe an den Schaltkreis anzulegen, sondern es muß eine Folge aus zwei Eingaben betrachtet werden.

Diese Arbeit wird sich mit dem Laufzeittest von VLSI-Schaltkreisen beschäftigen, wobei davon ausgegangen wird, daß das logische Verhalten der Schaltung korrekt ist. Hat man einen Laufzeittest für eine Schaltung durchgeführt, will man sicher sein, daß der Schaltkreis frei von Delay-Fehlern ist. Somit ist es von besonderer Wichtigkeit, den Schaltkreis *vollständig* auf Delay-Fehler zu überprüfen. Ebenfalls soll die Größe der verwendeteten Testmenge durch diese Anforderung aber nicht derart anwachsen, daß das Anwenden dieser Testmethodik in der Praxis uninteressant wird.

> Ist es also möglich, eine Testmethodik zu finden, die zum einen die Anforderung des vollständigen Tests erfüllt und zum anderen Testmengen akzeptabler Größe liefert?

Eine naive Möglichkeit, einen Schaltkreis auf Laufzeitfehler zu überprüfen, wäre es, sämtliche Eingabetupel an den Schaltkreis anzulegen. Dieser sogenannte „erschöpfende Test“ ist aber praktisch nicht mehr durchführbar, da z.B. bei einem Schaltkreis mit 100 Eingängen schon $2^{100} * 2^{100}$ verschiedene Eingabesequenzen an den Schaltkreis angelegt werden müßten. Somit muß eine Testmethodik verwendet werden, die mit wesentlich weniger Eingabekombinationen auskommt. Eine Testmethodik, die diese Anforderung erfüllt, ist der in der Praxis weit verbreitete *Pfadverzögerungstest*. Hierbei werden die Einzelpfade einer Schaltung bezüglich ihres Laufzeitverhaltens überprüft. Der Pfadverzögerungstest hat allerdings den Nachteil, daß Pfade in einem Schaltkreis existieren können, die für das korrekte Laufzeitverhalten des Schaltkreises überprüft werden müßten, aber nicht als Einzelpfade getestet werden können. Selbst nach dem Pfaddelaytest hat man daher für viele Schaltkreise keine vollständige Gewißheit, daß der Schaltkreis die an ihn gestellten Laufzeitbedingungen für alle Eingabefolgen erfüllt, d.h. der Pfadverzögerungstest ist unvollständig und genügt unserem Anspruch des vollständigen Delaytests nicht.

Ein Weg aus diesem Dilemma ist, eine Testmethodik zu verwenden, die nicht auf den Test von Einzelpfaden beschränkt ist. Eine solche Testmethodik stellt der *Pfadsystemtest* [KM94] dar, der eine Verallgemeinerung des Pfaddelaytests ist. Hierbei werden mehrere logische Einzelpfade zusammengefaßt, die dann als System getestet werden. In dieser Arbeit wird der Pfadsystemtest für eine spezielle Klasse von Schaltkreisen, den *unate Schaltkreisen*, vorgestellt. Bei diesen Schaltkreisen werden auch die zu testenden Pfade, die durch einen Einzelpfadtest nicht testbar waren, durch den Pfadsystemtest bezüglich ihres Laufzeitverhaltens überprüft. Somit ist der Pfadsystemtest für diese Klasse von Schaltkreisen eine vollständige Testmethodik.

> In dieser Arbeit wird gezeigt, wie das Testproblem für eine spezielle Klasse von Schaltkreisen, den *unate Schaltkreisen*, mit Hilfe des Pfadsystemtests gelöst werden kann.

Unate Schaltkreise haben mehrere wichtige Anwendungsgebiete in der Praxis:

- Dynamische CMOS-Logik [KLL82, GD83, HGDT84], die oft in Hochgeschwindigkeitsschaltkreisen verwendet wird [PDY92], muß als unate Schaltkreis realisiert werden.

- Selbsttestende Schaltkreise, die auf ungeordneten Codes basieren, werden oft als unate Schaltkreise realisiert [GJ88, DWB92].

Jeder beliebige kombinatorische Schaltkreis $\mathcal{K}$ kann in einen unate Schaltkreis transformiert werden, der $\mathcal{K}$ simuliert [Red73]. Hierbei wird die Größe des Schaltkreises maximal verdoppelt[1] und das Delay der Schaltung wird nicht erhöht. Für den Delaytest eines beliebigen Schaltkreises transformiert man diesen Schaltkreis in einen unate Schaltkreis, der den Ausgangsschaltkreis simuliert. Der resultierende unate Schaltkreis kann nun auch vollständig auf Verzögerungsfehler überprüft werden. Im Rahmen der vorliegenden Arbeit wurde ein Programm zur Transformation eines beliebigen Schaltkreises $\mathcal{K}$ in einen unate Schaltkreis, der $\mathcal{K}$ simuliert, entwickelt.

Wie kann nun ein vollständiger Verzögerungstest für unate Schaltkreise gefunden werden?
In [SR94] wurde gezeigt, daß man für einen vollständigen Delaytest für die Klasse der unate Schaltkreise eine Testmenge konstruieren kann, die nur von der logischen Funktion des unate Schaltkreises abhängt und die gegebene Realisierung unberücksichtigt läßt. Eine Teilaufgabe dieser Arbeit war es, diese *funktionale Delaytestmenge* für beliebige unate Schaltkreise zu berechnen. Für Schaltkreise mit nur einem Ausgang kann die Berechnung dieser Testmenge auf die Primimplikantenbestimmung zurückgeführt werden. Liegt ein Schaltkreis mit mehreren Ausgängen vor, wird die Testmenge zuerst für die zugehörigen Teilschaltkreise mit jeweils nur einem Ausgang berechnet. Um nun möglichst kompakte Testmengen für den Gesamtschaltkreis zu finden, versucht man die Testmuster der Einzelfunktionen zu überlagern. Für diese Parallelisierung der Testmengen wurden mehrere Kompaktierungsheuristiken implementiert, mit denen Verringerungen bis zu 97 % gegenüber der Summe der Einzeltestgrößen erzielt werden konnten. Für alle betrachteten ISCAS89 [BBK89] und MCNC-Benchmarkschaltkreise [Yan91] ergab sich eine durchschnittliche Verringerung von 31,5 %.

Die funktionale Testmenge hat die Vorteile, daß sie relativ einfach zu konstruieren ist und mit ihrer Hilfe jede Realisierung der Funktion als unate Schaltkreis vollständig auf Delayfehler überprüft werden kann. Die Eigenschaft der Realisierungsunabhängigkeit muß aber leider durch den Nachteil bezahlt werden, daß die Testmengengröße gegenüber dem Pfaddelaytest stark anwachsen kann. Somit erfüllt die funktionale Delaytestmenge zwar die Anforderung des vollständigen Delaytests, allerdings ist ihre Größe häufig nicht mehr akzeptabel.

Der Hauptteil dieser Arbeit beschäftigt sich mit der Frage, wie weit die Größe dieser Testmenge reduziert werden kann, wenn man sie auf eine gegebene Realisierung der Schaltung zuschneidet. Zu dieser Problemstellung liefert diese Arbeit zwei Lösungsansätze. Diese Ansätze erlauben einen trade-off zwischen Testmengengröße und Rechenzeit zur Berechnung des Delaytests. Beide Vorgehensweisen haben gemeinsam, daß die Eigenschaft des vollständigen Delaytests durch die Testmengenreduktion nicht verloren geht.

[1] Wie man in den experimentellen Resultaten von Kapitel 5 sehen wird, ist der durch die Transformation entstehende Hardwareoverhead häufig viel geringer als diese obere Schranke.

- Der erste Ansatz liefert eine möglichst kleine Teilmenge der funktionalen Testmenge, die für den vollständigen Delaytest des Schaltkreises ausreicht. Hierbei wird zunächst für jede Testsequenz der funktionalen Delaytestmenge bestimmt, welche Pfadsysteme durch diese Testsequenz überpüft werden. Aus diesen Pfadsytemmengen der einzelnen Testsequenzen wird eine Menge von Pfadsystemen gebildet, deren Test für einen vollständigen Delaytest des Schaltkreises hinreichend ist. Gesucht ist nun eine möglichst kleine Teilmenge der funktionalen Testmenge, die alle zu testenden Pfadsysteme des Schaltkreises überdeckt. Somit kann das Testmengenminimierungsproblem auf ein Überdeckungsproblem zurückgeführt werden.

- Da beim ersten Ansatz die Anzahl der zu testenden Pfadsysteme zu einer Testsequenz exponentiell zur Anzahl der Pfade des Schaltkreises sein kann, kann es vorkommen, daß ein Schaltkreis nicht mehr in zumutbarer Zeit bearbeitet werden kann. Hierfür stellt der zweite Ansatz eine grobere Vorgehensweise bereit: Es werden viele Pfadsysteme in größeren Pfadsystemen zusammengeschlossen, so daß die Anzahl der Pfadsysteme zu einer Eingabe nur noch linear zur Anzahl der Pfade des Schaltkreises ist. Durch diese größeren Pfadsysteme wird beim Lösen des Überdeckungsproblems ein Teil der Flexibilität genommen, so daß hierbei nicht so kompakte Testmengen wie beim ersten Ansatz erzeugt werden. Dafür hat die zweite Heuristik aber den Vorteil, daß die Laufzeit zur Testberechnung stark verringert werden kann. Im Vergleich zur funktionalen Testmenge kann die Testmengengröße immer noch entscheidend reduziert werden.

Praktische Resultate für ISCAS89 [BBK89] und MCNC-Benchmarks [Yan91] belegen, daß durch diese Vorgehensweise die Größe der funktionalen Testmenge stark reduziert werden kann: Bei den betrachteten Beispielschaltkreisen können Testmengenreduktionen bis zu 99,6 % erzielt werden (für alle betrachteten Schaltkreise im Durchschnitt 69,8 %). Ebenfalls sind unsere Testmengen nur in geringem Maße größer als die Testmengen eines (unvollständigen) Pfaddelaytests. Die verbesserte Testqualität wird also nur durch eine geringe Vergrößerung der Testmenge bezahlt.

Somit erfüllt die im Rahmen dieser Arbeit entwickelte Testmethodik unsere beiden Ansprüche an den Laufzeittest:

- Die von uns erzeugten Testmengen überprüfen unate Schaltkreise *vollständig* auf Delayfehler.

- Unsere Testmengen haben eine annehmbare Größe.

Ein Teil der Resultate, die bei dieser Arbeit erzielt wurden, liegt schon als Publikation in [SMR96a] und [SMR96b] vor.

Danksagungen

Mein Dank gilt zuerst Herrn Prof. Dr. Günther Hotz für die äußerst interessante Aufgabenstellung und seine Unterstützung. Herrn PD. Dr. Uwe Sparmann danke ich für die sehr gute Betreuung.
Weiterhin danke ich Dieter Luxenburger für die gelungene Zusammenarbeit, sowie allen Kommilitonen und Mitarbeitern an diesem Lehrstuhl für das hervorragende Arbeitsklima.
Insbesondere danke ich meinen Eltern, die mir ein Studium überhaupt erst ermöglichten.

Kapitel 1

Theoretische Grundlagen

In diesem Kapitel werden einige theoretische Grundlagen erläutert, die für das Verständnis der folgenden Kapitel nötig sind.
Zuerst wird das in dieser Arbeit verwendete Schaltkreismodell formal beschrieben. Anschließend wird eine kurze Einführung in den Fabrikationstest von VLSI-Schaltkreisen gegeben, wobei das zentrale Thema dieser Arbeit, der Verzögerungstest, näher erörtert wird.

1.1 Schaltkreismodell

Definition 1.1 :
$\boldsymbol{G = (V, E, \mathcal{Q}, \mathcal{Z})}$ *heißt* gerichteter Graph, *wenn gilt:*

> *V,E sind disjunkte, endliche Mengen.* V *wird als Menge der* Knoten, *E als Menge der* Kanten *von G bezeichnet. Durch die Funktionen* $\mathcal{Q}, \mathcal{Z} : E \rightarrow V$ *wird Anfangs- und Endknoten einer Kante festgelegt. Für eine Kante e von einem Knoten* v_1 *zu einem Knoten* v_2 *gilt:*
>
> $\mathcal{Q}(e) := v_1$ ($\mathcal{Z}(e) := v_2$) *heißt Quelle (Ziel) von e.*

Durch die Abb. $indeg\,(outdeg) : V \rightarrow \mathbb{N}_0$ *wird die Anzahl der in einen Knoten laufenden (aus einem Knoten laufenden) Kanten bestimmt:*

- $indeg(v) := |\{e|\, \mathcal{Z}(e) = v\}|$
- $outdeg(v) := |\{e|\, \mathcal{Q}(e) = v\}|$

In der nun folgenden Definition eines kombinatorischen Schaltkreises wird eine Beschränkung auf eine einfache Menge von Zelltypen vorgenommen:

$$\mathcal{ZTYP} := \{\,\textsc{Not}, \textsc{And}, \textsc{Or}, \textsc{Nor}, \textsc{Nand}, \textsc{Pi}, \textsc{Po}, \textsc{Fan}\}$$

d.h. primäre Inputs (Pi), primäre Outputs (Po) und Fanoutknoten (Fan) werden ebenfalls als „Zellen“ angesehen.
Spielt die Anzahl der Eingänge (Ausgänge) einer Zelle eine wesentliche Rolle, so wird diese als Index dem Zelltyp beigefügt (z.B. $\textsc{And}_2$ stellt ein And-Gatter mit 2 Inputs dar, $\textsc{Fan}_3$ ist ein Fanoutknoten mit 3 Outputs).

Definition 1.2 :
Ein 2–Tupel $\boldsymbol{\mathcal{K}} = (\boldsymbol{G}, \boldsymbol{F})$ *heißt* **kombinatorischer Schaltkreis**, *falls gilt:*

- $G = (Z, L, \mathcal{Q}, \mathcal{Z})$ *ist ein zykelfreier gerichteter Graph mit*
  - Z *ist die Menge der* Zellknoten *und korrespondiert zu den im Schaltkreis vorkommenden Gattern, Fanouts, Inputs und Outputs.*
  - L *ist die Menge der Kanten; sie entspricht den Leitungen, welche die Zellen des Schaltkreises miteinander verbinden.*
- $F = (typ, pi, po)$ *ist ein Tripel von Funktionen, mit:*
  - $typ : \{1, ..., |Z|\} \rightarrow \mathcal{ZTYP}$
    typ ordnet jeder Zelle des Schaltkreises ihren Zelltyp zu.
  - $pi : \{1, ..., |Z_{PI}|\} \rightarrow Z_{PI}$, *wobei* $Z_{PI} := \{z \in Z \mid typ(z) = \textsc{Pi}\}$
    pi numeriert die Inputs der Schaltung durch.
  - $po : \{1, ..., |Z_{PO}|\} \rightarrow Z_{PO}$, *wobei* $Z_{PO} := \{z \in Z \mid typ(z) = \textsc{Po}\}$
    po numeriert die Outputs der Schaltung durch.
- *Für die Anzahl der Eingänge und Ausgänge jeder Zelle* $z \in Z$ *gilt:*
  - $indeg(z) \geq 2$ *falls* $typ(z) \in \{\textsc{And}, \textsc{Or}, \textsc{Nor}, \textsc{Nand}\}$
  - $indeg(z) = 1$ *falls* $typ(z) \in \{\textsc{Po}, \textsc{Not}, \textsc{Fan}\}$
  - $indeg(z) = 0$ *falls* $typ(z) = \textsc{Pi}$
  - $outdeg(z) \geq 2$ *falls* $typ(z) = \textsc{Fan}$
  - $outdeg(z) = 1$ *falls* $typ(z) \in \{\textsc{Pi}, \textsc{And}, \textsc{Or}, \textsc{Nand}, \textsc{Nor}, \textsc{Not}\}$
  - $outdeg(z) = 0$ *falls* $typ(z) = \textsc{Po}$

Wird eine Eingabe an die Inputs des Schaltkreises angelegt, so können den Leitungen Werte bezüglich dieser Eingabe zugewiesen werden.

Definition 1.3 :
Sei $\mathcal{K}$ ein kombinatorischer Schaltkreis.

- *Sei e eine Eingabe an $\mathcal{K}$.*

  Die Belegung einer Leitung l in $\mathcal{K}$ bei Eingabe e bezeichnen wir mit $e(l)$.

- *Sei $< e_1, e_2 >$ eine Eingabefolge an $\mathcal{K}$.*
  Sei die Eingabe e_1 solange an den Schaltkreis angelegt, daß sich jede Leitung auf ihren Endwert bezüglich e_1 eingestellt hat. Danach wird die Eingabe e_2 an den Schaltkreis angelegt.

  Der Wert auf Leitung l heißt stabil *für Eingabefolge $< e_1, e_2 >$. $\Longleftrightarrow$*
  Der Wert auf l bleibt nach Anlegen von e_2 unverändert auf $e_1(l)$, d.h. $e_2(l) = e_1(l)$ und es treten keine Hazards auf Leitung l auf.

Wir verwenden ein einfaches Laufzeitmodell, bei dem nur den Gattern des Schaltkreises ein Delay zugeordnet wird. Dieses Delay hängt von der am Gatter anliegenden Transition und dem Input, über den diese Transition propagiert wird, ab.

Definition 1.4 :
Das Delay eines Gatters g bezüglich einer Transition $t \in \{1 \to 0, 0 \to 1\}$ an Inputleitung l ist die Zeit, die das Gatter g benötigt, um t von Input l zum Output von g zu propagieren. Dieses Delay wird mit $d_{t,l}(g)$ bezeichnet.

Definition 1.5 :
Sei $\mathcal{K}$ ein kombinatorischer Schaltkreis und e eine Eingabe an $\mathcal{K}$.

- *Die worst-case Verzögerung von $\mathcal{K}$ für Eingabe e bezeichnen wir mit $d_{\mathcal{K}}(e)$.*

- *Die worst-case Verzögerung von $\mathcal{K}$ ist:*

$$d(\mathcal{K}) := \max\{d_{\mathcal{K}}(e) | \, e \text{ Eingabe an } \mathcal{K} \, \}$$

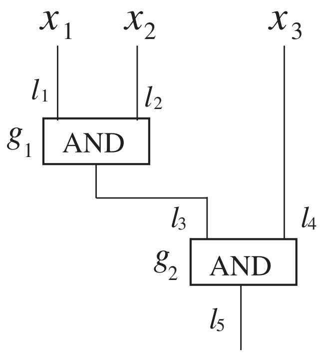

Abbildung 1.1: Beispielschaltkreis $\mathcal{K}$

Beispiel 1.1 :
Wir betrachten den Schaltkreis $\mathcal{K}$ in Abb. 1.1.

Seien die Delays der Gatter wie folgt gegeben:

$$\begin{array}{ll}
d_{0\to 1,l_1}(g_1) = 1,7 \text{ ns} & d_{0\to 1,l_2}(g_1) = 1,75 \text{ ns} \\
d_{1\to 0,l_1}(g_1) = 1,9 \text{ ns} & d_{1\to 0,l_2}(g_1) = 1,95 \text{ ns} \\
d_{0\to 1,l_3}(g_2) = 1,2 \text{ ns} & d_{0\to 1,l_4}(g_2) = 1,25 \text{ ns} \\
d_{1\to 0,l_3}(g_2) = 1,45 \text{ ns} & d_{1\to 0,l_4}(g_2) = 1,4 \text{ ns}
\end{array}$$

Die worst-case Verzögerungen der Eingaben an $\mathcal{K}$ sind:

$$\begin{array}{ll}
d_{\mathcal{K}}(000) = 1,4 \text{ ns} & d_{\mathcal{K}}(001) = 3,35 \text{ ns} \\
d_{\mathcal{K}}(010) = 1,4 \text{ ns} & d_{\mathcal{K}}(011) = 3,35 \text{ ns} \\
d_{\mathcal{K}}(100) = 1,4 \text{ ns} & d_{\mathcal{K}}(101) = 3,4 \text{ ns} \\
d_{\mathcal{K}}(110) = 1,4 \text{ ns} & d_{\mathcal{K}}(111) = 2,95 \text{ ns}
\end{array}$$

Das worst-case Delay des Schaltkreises ist:

$$d(\mathcal{K}) = d_{\mathcal{K}}(101) = 3,4 \text{ ns}$$

■

Somit ist eine formale Definition eines beliebigen kombinatorischen Schaltkreises gegeben. Solch beliebige Schaltkreise sind in der Praxis allerdings schwer bzw. nur unvollständig auf Laufzeitfehler testbar. Im folgenden wird eine Teilklasse von Schaltkreisen, die monotonen bzw. unate Schaltkreise, eingeführt. Sie können mit Hilfe der in dieser Arbeit betrachteten Testmethodik vollständig auf Laufzeitfehler überprüft werden.

1.2 Monotone und unate Schaltkreise

Definition 1.6 :
Ein kombinatorischer Schaltkreis heißt monoton*, falls in ihm keine invertierenden Gatter (also keine* NAND-*,* NOR- *oder* NOT-*Gatter) auftreten.*

Eine Verallgemeinerung der monotonen Schaltkreise sind die *quasi-monotonen* bzw. die *unate* Schaltkreise. Wie schon in der Einleitung erwähnt, sind die unate Schaltkreise bei der on-line-Fehlererkennung sowie beim Entwurf dynamischer CMOS-Logik von Bedeutung.

Definition 1.7 :
Sei $\mathcal{K}$ ein kombinatorischer Schaltkreis mit einem Ausgang.

1. *Sei p ein Pfad in $\mathcal{K}$ und i die Anzahl der invertierenden Gatter auf p. Die* Inversionsparität *$I(p)$ des Pfades p ist definiert als:*

$$I(p) := i \bmod 2$$

2. *Sei x_j eine Inputvariable von $\mathcal{K}$.*

   $\mathcal{K}$ heißt positiv (negativ) unate in x_j. $\Longleftrightarrow$
   Alle Pfade vom zu x_j korrespondierenden Input zum Output von $\mathcal{K}$ haben Inversionsparität 0 (1).

3. *$\mathcal{K}$ heißt* quasi-monoton*, falls er positiv unate in allen Eingangsvariablen ist.*

4. *$\mathcal{K}$ heißt* unate*, falls er in jeder Eingangsvariablen entweder positiv oder negativ unate ist. Ansonsten heißt er* binate.

Bemerkung 1.1 :
Da in einem monotonen Schaltkreis alle Pfade die Inversionsparität 0 haben, ist jeder monotone Schaltkreis quasi-monoton. Jeder quasi-monotone Schaltkreis ist unate.

Beispiel 1.2 :
Wir betrachten die in Abbildung 1.2 und 1.3 gegebenen Schaltkreise $\mathcal{K}_1$ und $\mathcal{K}_2$. Der Schaltkreis $\mathcal{K}_1$ ist quasi-monoton, da alle Pfade, die von den primären Inputs zum primären Output von $\mathcal{K}_1$ laufen, die Inversionsparität 0 haben. $\mathcal{K}_1$ ist also in allen Variablen positiv unate.
Der Schaltkreis $\mathcal{K}_2$ ist unate, da alle Pfade, die von einem primären Input zum primären Output von $\mathcal{K}_2$ laufen, die gleiche Inversionsparität haben. $\mathcal{K}_2$ ist positiv unate in den Varablen x_1 und x_2 und negativ unate in der Variablen x_3. ■

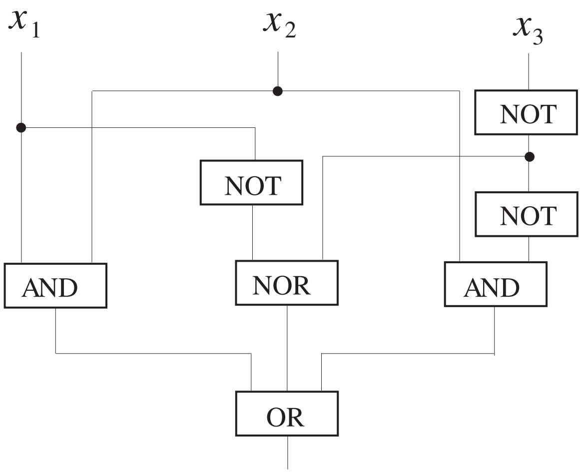

Abbildung 1.2: Beispielschaltkreis $\mathcal{K}_1$

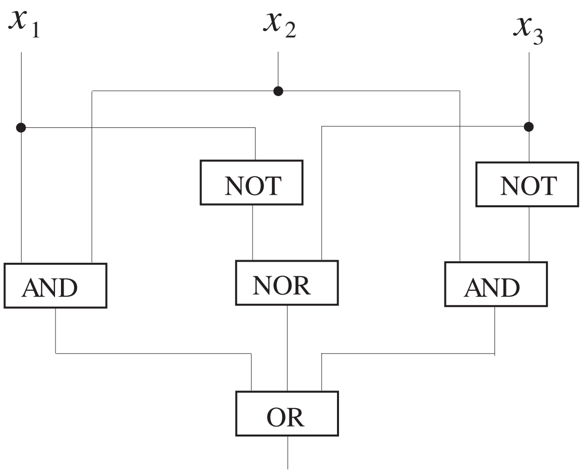

Abbildung 1.3: Beispielschaltkreis $\mathcal{K}_2$

Definition 1.8:
Sei $\mathcal{K}$ ein Schaltkreis mit m Ausgängen. $\mathcal{K}_j$, $j \in \{1, \ldots, m\}$, bezeichne den Teilschaltkreis von $\mathcal{K}$ zum j-ten Ausgang.

$\mathcal{K}$ heißt quasi-monoton (unate) $\iff$
$\mathcal{K}_j$ ist quasi-monoton (unate) $\forall\, j \in \{1, \ldots, m\}$

Einem jeden Schaltkreis kann die durch ihn berechnete *logische Funktion f* zugeordnet werden. Um den oben definierten quasi-monotonen und unate Schaltkreisen ihre logischen Funktionen zuzuordnen, genügt es, eine Teilmenge aller logischen Funktionen, die monotonen Funktionen und deren Verallgemeinerung, die unate Funktionen, zu betrachten.

Definition 1.9:
Sei $f : \boldsymbol{B}^n \to \boldsymbol{B}$ eine logische Funktion.

1. *f heißt* positiv unate *in der i-ten Eingabevariablen, falls:*

$$f(\alpha_1, \ldots, \alpha_{i-1}, 1, \alpha_{i+1}, \ldots, \alpha_n) \geq f(\alpha_1, \ldots, \alpha_{i-1}, 0, \alpha_{i+1}, \ldots, \alpha_n)$$
$$\forall\, (\alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_n) \in \boldsymbol{B}^{n-1}$$

2. *f heißt* negativ unate *in der i-ten Eingabevariablen, falls:*

$$f(\alpha_1, \ldots, \alpha_{i-1}, 1, \alpha_{i+1}, \ldots, \alpha_n) \leq f(\alpha_1, \ldots, \alpha_{i-1}, 0, \alpha_{i+1}, \ldots, \alpha_n)$$
$$\forall\, (\alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_n) \in \boldsymbol{B}^{n-1}$$

3. *f heißt* monotone Funktion, *falls sie in jeder Inputvariablen positiv unate ist.*

4. *f heißt* unate, *falls sie in jeder Inputvariablen entweder positiv oder negativ unate ist. Ansonsten heißt f* binate Funktion.

Beispiel 1.3:
Die Funktion

$$f_1 = x_1x_2 \vee x_1x_3 \vee x_2x_3$$

ist positiv unate in all ihren Variablen. Folglich ist die Funktion monoton.
Die Funktion

$$f_2 = x_1x_2 \vee x_1\overline{x_3} \vee x_2\overline{x_3}$$

ist positiv unate in den Variablen x_1 und x_2 und negativ unate in der Variablen x_3. Folglich ist die Funktion f_2 unate.
Die Funktion

$$f_3 = x_1x_2 \vee \overline{x_1}x_3$$

ist positiv unate in den Variablen x_2 und x_3 und binate in der Variablen x_1. Somit ist die Funktion f_3 binate. ■

Es ist bekannt, daß jeder monotonen Funktion $f : \boldsymbol{B}^n \rightarrow \boldsymbol{B}$ stets ein quasi-monotoner Schaltkreis zugeordnet werden kann, der gerade diese Funktion realisiert. (Nach [Hot90] enthält das Minimalpolynom zu einer monotonen Funktion keine negierten Variablen.)
Umgekehrt berechnet jeder quasi-monotone Schaltkreis stets eine monotone Funktion. (Da der Schaltkreis positiv unate in jeder Inputvariablen ist, d.h. alle Pfade eine gerade Anzahl von Inversionen haben, kann sich der Funktionswert bei Vergrößern eines Variablenwertes von 0 nach 1 nie von 1 nach 0 verkleinern.)

Für unate Schaltkreise und unate Funktionen gelten dieselben Beziehungen wie für quasi-monotone Schaltkreise und monotone Funktionen.
Sei eine unate Funktion $f : \boldsymbol{B}^n \rightarrow \boldsymbol{B}$ gegeben, die von allen Inputvariablen $x_1, \ldots, x_n$ echt abhängt. O.B.d.A. sei f in den Variablen $x_1, \ldots, x_k$ positiv und in den Variablen $x_{k+1}, \ldots, x_n$ negativ unate. f kann ein unate Schaltkreis $\mathcal{K}$ zugeordnet werden, der f berechnet und der positiv unate in den Variablen $x_1, \ldots, x_k$ und negativ unate in den Variablen $x_{k+1}, \ldots, x_n$ ist. (Alle Primimplikanten von f enthalten die Variablen $x_1, \ldots, x_k$ nur nicht-negiert und die Variablen $x_{k+1}, \ldots, x_n$ nur negiert. Beweis: Siehe Anhang A.)
Umgekehrt berechnet jeder unate Schaltkreis mit einem Ausgang, der positiv unate in den Variablen $x_1, \ldots, x_k$ und negativ unate in den Variablen $x_{k+1}, \ldots, x_n$ ist, stets eine unate Funktion, die ebenfalls positiv unate in den Variablen $x_1, \ldots, x_k$ und negativ unate in den Variablen $x_{k+1}, \ldots, x_n$ ist. (Ist der Schaltkreis positiv unate in einer Inputvariablen, kann sich der Funktionswert bei Vergrößerung des Eingabewertes dieser Variablen von 0 nach 1 und Beibehaltung der anderen Eingabewerte nie von 1 nach 0 verkleinern, da auf jedem Pfad von dieser Inputvariablen zum Ausgang der Schaltung eine gerade Anzahl

von Invertern liegt. Ist der Schaltkreis negativ unate in einer Inputvariablen, kann sich der Funktionswert bei Verkleinerung des Eingabewertes dieser Variablen von 1 nach 0 und Beibehaltung der anderen Eingabewerte nie von 1 nach 0 verkleinern, da auf jedem Pfad von dieser Inputvariablen zum Ausgang der Schaltung eine ungerade Anzahl von Invertern liegt.)

Beispiel 1.4:
Der quasi-monotone Schaltkreis $\mathcal{K}_1$ aus Abbildung 1.2 berechnet die monotone Funktion f_1 aus Beispiel 1.3.
Der unate Schaltkreis $\mathcal{K}_2$ aus Abbildung 1.3 berechnet die unate Funktion f_2 aus Beispiel 1.3. ■

1.3 Der Fertigungstest

Um das Ausliefern fehlerhafter Chips zu vermeiden oder zumindest deren Anzahl möglichst gering zu halten, ist es nötig, die gefertigten Chips einer *Testprozedur* zu unterziehen. Wie wichtig es ist, die Anzahl der fehlerhaften Chips zu minimieren, wird durch die folgende in der Praxis beobachtete Faustformel verdeutlicht [Wun91]:

> Die Kosten, die durch einen fehlerhaften Chip verursacht werden, wachsen um den Faktor 10, je später der Fehler entdeckt wird: Beim Testen des Einzelchips, auf der Platine, nach dem Einbau in das System oder erst nach der Auslieferung des Systems.

Somit ist es wichtig, einen Fehler möglichst früh zu erkennen. Beim Testen eines Schaltkreises kann unterschieden werden, ob die Schaltung im Hinblick auf ein logisches oder zeitliches Fehlverhalten hin untersucht werden soll. Ein Beispiel für logische Defekte sind die sogenannten Einfach- (Mehrfach-) *stuck-at-Fehler*. Hierbei wird angenommen, daß *genau eine* (*mindestens eine*) Leitung des Schaltkreises defekt ist und die defekten Leitungen durch den Fehler fest auf logisch 0 oder 1 liegen ($\rightarrow$stuck_at_0, stuck_at_1).
Ist sichergestellt, daß das logische Verhalten des Schaltkreises korrekt ist, muß die Schaltung weiterhin bezüglich ihres Laufzeitverhaltens überprüft werden. Gerade in der heutigen Zeit, in der immer größerer Wert auf eine Steigerung der Arbeitsgeschwindigkeit der Computer gelegt wird, ist es besonders wichtig auch temporale Defekte zu erkennen. Dies ist die Aufgabe des *Delay-* oder *Laufzeittests*, welcher im folgenden Kapitel näher beschrieben wird.

1.4 Der Laufzeittest

Für Rechenanlagen sind Geschwindigkeiten vorgesehen, mit denen diese Anlagen betrieben werden sollen. Um diese Betriebsgeschwindigkeiten zu gewährleisten, müssen die Teilschaltungen schnell genug arbeiten, d.h. jedem Schaltkreis ist eine obere Schranke τ für seine Laufzeit zugeordnet. Wird diese Laufzeitschranke überschritten, so liegt innerhalb der Schaltung ein Verzögerungsfehler vor.

Definition 1.10:
Sei $\mathcal{K}$ ein kombinatorischer Schaltkreis.

$$\mathcal{K} \text{ hat einen Delay-Fehler für Laufzeitschranke } \tau \Longleftrightarrow d(\mathcal{K}) > \tau$$

Aufgabe des Laufzeittestes ist es, zu überprüfen, ob ein gegebener Schaltkreis $\mathcal{K}$ bei einer gegebenen Laufzeitschranke τ rechtzeitig schaltet, d.h.:

$$\text{Überprüfe, ob } d(\mathcal{K}) \leq \tau.$$

Die **prinzipielle Vorgehensweise für den Laufzeittest** ist wie folgt:
Es wird zum Zeitpunkt t_0 ein Eingabevektor e_1 angelegt und solange gewartet, bis alle Werte im Schaltkreis ihren Endwert unter Eingabe e_1 angenommen haben. Zum Zeitpunkt t_1 wird dann die zweite Testeingabe e_2 angelegt, wobei e_1 und e_2 unterschiedliche Ausgaben erzeugen müssen.
Zum Zeitpunkt $t_2 = t_1 + \tau$ wird der Wert am Ausgang des Schaltkreises abgelesen.
Wird nun zum Zeitpunkt t_2 nicht der Ausgabewert zur Eingabe e_2 abgelesen, so liegt ein Verzögerungsfehler innerhalb des Schaltkreises $\mathcal{K}$ vor. Diese Vorgehensweise wird in der Abbildung 1.4 veranschaulicht.

Eine Möglichkeit, den Laufzeittest durchzuführen, wäre es, alle Zweiertupel von Eingaben an den Schaltkreis anzulegen. Da es aber viel zu teuer ist, den Schaltkreis für diese enorme Menge von Eingaben zu testen ($Anzahl = 2^{2n}, n = \#Inputs$), ist es nötig, eine Testmethodik zu verwenden, die mit einer Teilmenge dieser Eingabetupel auskommt.

1.4.1 Der Pfadverzögerungstest

Eine in der Praxis weit verbreitete Methode zur Überprüfung des Laufzeitverhaltens einer Schaltung ist der *Pfadverzögerungstest*. Hierbei wird überprüft, ob das Delay der Pfade einer Schaltung die Laufzeitschranke τ nicht überschreitet.
Wie schon erwähnt, sind bei unserem Laufzeitmodell (siehe Paragraph 1.1) nur den Gattern des Schaltkreises Verzögerungen zugeordnet. Das Delay eines Pfades p wird somit durch das Delay der Gatter bestimmt, die auf p liegen.

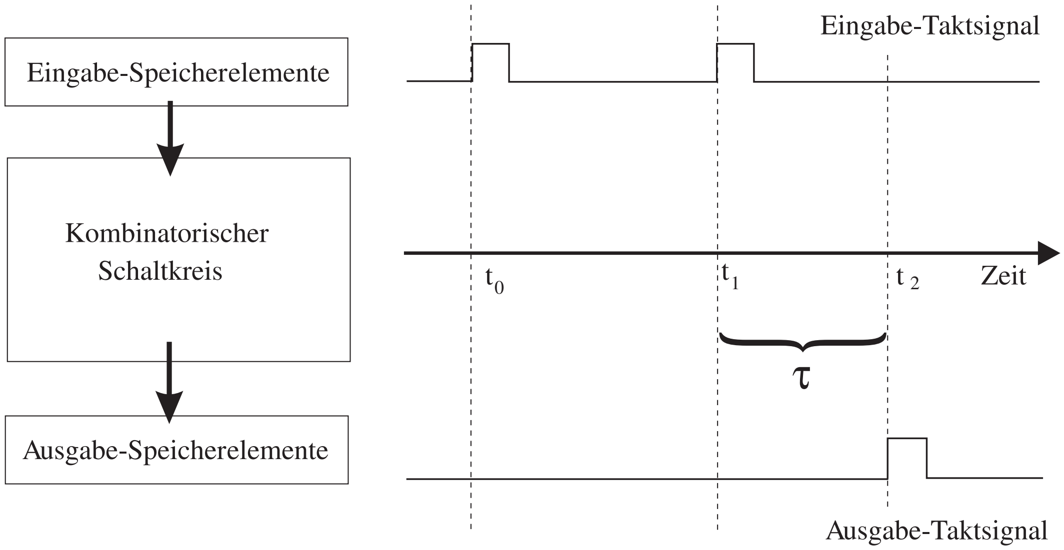

Abbildung 1.4: Vorgehensweise beim Laufzeittest

Definition 1.11 :
Ein (physikalischer) Pfad *p innerhalb eines kombinatorischen Schaltkreises $\mathcal{K}$ ist als Folge von Leitungen wie folgt definiert:*

$$p := l_1\text{-}l_2\text{-}\ldots\text{-}l_n,\ (n \geq 1)\ \mathit{mit}$$
$$\mathcal{Q}(l_{i+1}) = \mathcal{Z}(l_i),\ \forall\, i = 1, \ldots, n-1.$$

Die Länge *des Pfades p ist die Anzahl der zu p gehörenden Leitungen.*
Die Menge aller Pfade, die von einem primären Input zu einem primären Output in $\mathcal{K}$ führen, wird mit $\mathcal{P}(\mathcal{K})$ bezeichnet.

Da die auf einem Pfad liegenden Gatter für unterschiedliche Schaltrichtungen unterschiedliche Zeitverhalten aufweisen, reicht es nicht aus, nur physikalische Pfade zu betrachten. Jedem physikalischen Pfad des Schaltkreises werden daher zwei logische Pfade zugeordnet.

Definition 1.12 :
Sei $\mathcal{K}$ ein kombinatorischer Schaltkreis.

- *Ein* logischer Pfad *in $\mathcal{K}$ ist ein Tupel (p, t) mit $p \in \mathcal{P}(\mathcal{K})$ und $t \in \{0 \to 1, 1 \to 0\}$.*
- *Das Delay $d_{\mathcal{K}}(p, t)$ des logischen Pfades (p, t) ist die Laufzeit, die der Schaltkreis benötigt, um die Transition t startend beim primären Input von p über p zu propagieren.*

Bemerkung 1.2:
Die Transition am Input sowie die Forderung der Propagation dieser Transition über p legt bei den von uns betrachteten Gattertypen (AND, OR, NAND, NOR, NOT) schon die Transition an allen Gattern von p fest.

Definition 1.13:
Sei (p,t) ein logischer Pfad, l eine Leitung auf p.

> $Flanke_{(p,t)}(l)$ *ist die bei Propagation der Flanke t über den Pfad p der Leitung l zugeordnete Flanke.*

Beispiel 1.5:
Sei ein physikalischer Pfad $p = l_1$-l_2-l_3 wie in Abbildung 1.5 gegeben. Zu bestimmen ist das Delay des logischen Pfades $(p, 0 \rightarrow 1)$.

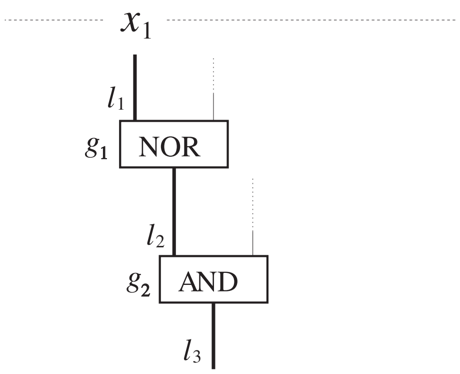

Abbildung 1.5: Beispielpfad p

An den Input x_1 des Pfades wird die Transition $0 \rightarrow 1$ angelegt. Die Forderung, daß diese Transition über den Pfad propagiert werden soll, bestimmt eindeutig die zugehörigen Transitionen auf l_2 und l_3.
Wegen der angelegten Transition gilt für l_1: $Flanke_{(p,t)}(l_1) = 0 \rightarrow 1$.
Danach wird die Transition über ein NOR-Gatter propagiert. Hierbei wird die Flanke invertiert, also: $Flanke_{(p,t)}(l_2) = 1 \rightarrow 0$.
Bei Propagation über ein AND-Gatter wird die Flanke nicht mehr verändert und somit ist $Flanke_{(p,t)}(l_3) = 1 \rightarrow 0$.
Das Delay des Pfades $(p, 0 \rightarrow 1)$ ist die Summe der Einzellaufzeiten der auf p liegenden Gatter:

$$d_{\mathcal{K}}(p, 0 \rightarrow 1) = d_{0\rightarrow 1, l_1}(g_1) + d_{1\rightarrow 0, l_2}(g_2).$$

■

Definition 1.14:
Sei $\mathcal{K}$ ein kombinatorischer Schaltkreis.

$$\mathcal{LP}(\mathcal{K}) := \{(p,t) \mid p \in \mathcal{P}(\mathcal{K}), t \in \{0 \to 1, 1 \to 0\}\}$$

bezeichnet die Menge aller logischen Pfade in $\mathcal{K}$.

Da die Laufzeit eines Schaltkreises durch seine maximale Pfadlaufzeit beschränkt ist, gilt:

Satz 1.15:
Sei $\mathcal{K}$ ein kombinatorischer Schaltkreis und $\mathcal{LP}(\mathcal{K})$ die Menge seiner logischen Pfade. Dann gilt:

$$\forall\,(p,t) \in \mathcal{LP}(\mathcal{K}) : \; d_{\mathcal{K}}(p,t) \leq \tau \implies d(\mathcal{K}) \leq \tau$$

Zur Überprüfung des Laufzeitverhaltens von Pfaden verwendet man sogenannte *robuste Tests:*

Definition 1.16:
Sei $< e_1, e_2 >$ eine Eingabefolge an den Schaltkreis $\mathcal{K}$. $< e_1, e_2 >$ heißt robuster Test *für den Pfad (p,t), wenn gilt:*

> *Funktioniert $\mathcal{K}$ korrekt für den Test $< e_1, e_2 >$ bei Laufzeitschranke τ, so kann gefolgert werden, daß $d_{\mathcal{K}}(p,t) \leq \tau$.*

Ein robuster Test für einen Pfad (p,t) muß die Transition t am Input des Pfades anlegen und garantieren, daß die Transition t über p bis zum Output der Schaltung propagiert wird. Die Transition darf am Output der Schaltung erst ablesbar sein, wenn sie über den betrachteten Pfad bis zum Output weitergeleitet wurde. Hierfür müssen die logischen Werte, die an Gattern auf dem Pfad (p,t) anliegen, bestimmte Bedingungen erfüllen. Um dies formal fassen zu können, müssen noch einige Begriffe eingeführt werden.

Definition 1.17:
Sei (p,t) ein logischer Pfad, l eine Leitung auf p und z die Zelle, deren Input l ist.

- $initial\_value_{(p,t)}(l)$ *:= Initialwert der Flanke* $Flanke_{(p,t)}(l)$
- $final\_value_{(p,t)}(l)$ *:= Endwert der Flanke* $Flanke_{(p,t)}(l)$
- *Die Leitung l heißt* Pfadinput *der Zelle z bezüglich des Pfades p. Alle anderen Inputs von z heißen* Seiteninputs *bezüglich des Pfades p.*

Sei g ein Gatter mit $typ(g) \in \{\text{And}, \text{Or}, \text{Nand}, \text{Nor}\}$ und l ein Input von g. Dann gibt es genau eine Belegung von l, die die Ausgabe von g unabhängig von der Belegung der anderen Inputs kontrolliert.

Definition 1.18:
Sei g ein Gatter mit $typ(g) \in \{\text{And}, \text{Or}, \text{Nand}, \text{Nor}\}$.

(a) Der kontrollierende Wert für g ist:

$$k(g) := \begin{cases} 0 & falls \quad g \in \{\text{And}, \text{Nand}\} \\ 1 & falls \quad g \in \{\text{Or}, \text{Nor}\} \end{cases}$$

(b) Der nicht-kontrollierende Wert für g ist:

$$nk(g) := \begin{cases} 1 & falls \quad g \in \{\text{And}, \text{Nand}\} \\ 0 & falls \quad g \in \{\text{Or}, \text{Nor}\} \end{cases}$$

Ein robuster Test zu einem Pfad (p, t) muß die Seiteninputs der auf dem Pfad liegenden Gatter auf bestimmte logische Werte festlegen. Nur so kann garantiert werden, daß der Ausgang eines Gatters auf dem Pfad erst dann schalten kann, wenn die entsprechende Transition am Pfadinput des Gatters erfolgt ist.

Lemma 1.19 ([LR87])
Sei $\mathcal{K}$ ein kombinatorischer Schaltkreis. Für eine Eingabe e an $\mathcal{K}$ bezeichne $e(l)$ den Endwert auf Leitung l bezüglich Eingabe e.
Sei (p, t) ein logischer Pfad in $\mathcal{K}$ und $< e_1, e_2 >$ eine Eingabefolge an $\mathcal{K}$ mit:

1. *$< e_1, e_2 >$ legt die Transition t an den Input von Pfad p an.*
2. *Für jedes Gatter g auf p mit $typ(g) \in \{\text{And}, \text{Nand}, \text{Or}, \text{Nor}\}$ gilt:*
   Sei l die Leitung auf p mit $\mathcal{Z}(l) = g$.

   (a) $final\_value_{(p,t)}(l) = nk(g)$

   $$\Rightarrow \forall\, l' \ mit \ (\mathcal{Z}(l') = g) \ \wedge \ (l' \neq l): \ e_2(l') = nk(g)$$

   (b) $final\_value_{(p,t)}(l) = k(g)$

   $$\begin{aligned} \Rightarrow \quad & \forall\, l' \ mit \ (\mathcal{Z}(l') = g) \ \wedge \ (l' \neq l): \\ & (e_1(l') = e_2(l') = nk(g)) \ \wedge \ (Wert \ auf \ l' \ ist \ stabil^1 \ für \ < e_1, e_2 >) \end{aligned}$$

Dann ist $< e_1, e_2 >$ ein robuster Test für den Pfad (p, t).

Beweis:
Für eine Leitung l auf p bezeichne p_l das Anfangsstück von p bis zu l.
Wir zeigen induktiv, daß die folgende Invariante bei der Ausführung des Tests $< e_1, e_2 >$ gilt:

> Sei T der früheste Zeitpunkt, zu dem nach Anlegen von e_2 der Wert $e_2(l)$ auf Leitung l beobachtet wird, dann gilt:
>
> $$d_{\mathcal{K}}(p_l, t) \leq T$$

INDUKTIONSANFANG: *p_l besteht nur aus einer Leitung.*
Beweis klar.

INDUKTIONSSCHRITT: *p_l besteht aus n $(n > 1)$ Leitungen.*
Sei g das letzte Gatter auf p_l. Der Pfadinput (Pfadoutput) von g bezüglich p_l sei l' (l).
Der Fall $typ(g) = \text{NOT}$ ist trivial.
Sei also $typ(g) \in \{\text{AND}, \text{OR}, \text{NAND}, \text{NOR}\}$. Sei T' (T) der Zeitpunkt, zu dem zum ersten Mal $e_2(l')$ auf l' ($e_2(l)$ auf l) beobachtet wird.

<u>FALL 1:</u> $e_2(l') = \text{k}(g)$ (siehe Abbildung 1.6)

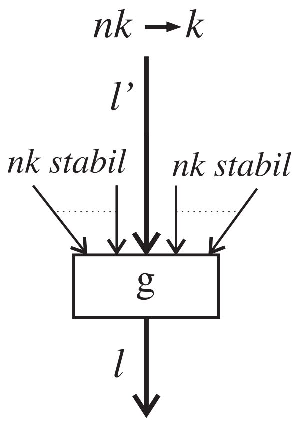

Abbildung 1.6: $e_2(l')$ kontrollierend

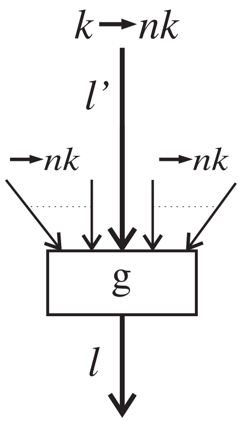

Abbildung 1.7: $e_2(l')$ nicht-kontrollierend

Da alle Seiteninputs von g stabil auf nicht-kontrollierenden Werten liegen, gilt:

$$\begin{aligned} T &= T' + d_{nk\to k, l'}(g) \\ &\overset{Ind.Vor.}{\geq} d_{\mathcal{K}}(p_{l'}, t) + d_{nk\to k, l'}(g) = d_{\mathcal{K}}(p_l, t) \end{aligned}$$

[1]siehe Def. 1.3, Seite 10

<u>FALL 2:</u> $e_2(l') = \mathrm{nk}(g)$ (siehe Abbildung 1.7)
Die Endwerte aller Seiteninputs von g sind nicht-kontrollierend bezüglich g. Der Ausgang von g kann seinen Endwert erst annehmen, wenn alle Inputs von g auf den nicht-kontrollierenden Wert gesetzt worden sind. Somit gilt:

$$\begin{aligned} T &\geq T' + d_{k\to nk,l'}(g) \\ &\overset{Ind.Vor.}{\geq} d_{\mathcal{K}}(p_{l'},t) + d_{k\to nk,l'}(g) = d_{\mathcal{K}}(p_l,t) \end{aligned}$$

■

Beispiel 1.6 :
Sei ein kombinatorischer Schaltkreis $\mathcal{K}_1$ wie in Abb. 1.8 gegeben.

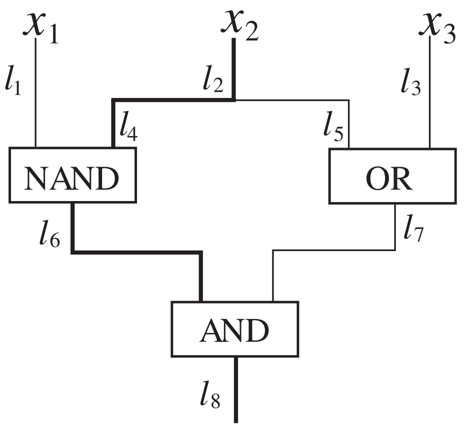

Abbildung 1.8: Beispielschaltkreis $\mathcal{K}_1$

Gesucht sei ein robuster Einzelpfadtest $< e_1, e_2 >$ zu dem logischen Pfad $(p, 0 \to 1)$, mit:

$$p = l_2\text{-}l_4\text{-}l_6\text{-}l_8$$

An den Input von p muß der Übergang $0 \to 1$ angelegt werden. Somit muß für e_1 der Input-Wert der Variablen x_2 auf logisch Null, der von e_2 auf logisch Eins gesetzt werden. l_4 ist eine Leitung des Pfades p. Es ergibt sich folgende Bedingung für den Seiteninput des mit l_4 verbundenen NAND-Gatters g:

$$\begin{aligned} &final\_value_{(p,0\to1)}(l_4) = \mathrm{nk}(g) \\ &\Rightarrow e_2(l_1) = \mathrm{nk}(g) \\ &\Rightarrow \text{Belegung von } x_1 \text{ für Eingabe } e_2 \text{ gleich } 1 \end{aligned}$$

l_6 ist ebenfalls eine Leitung von p. Diese Leitung endet in einem AND-Gatter g'. Es gilt:

$$\begin{aligned}&final\_value_{(p,0\to1)}(l_6) = \mathrm{k}(g')\\ &\Rightarrow (e_1(l_7) = e_2(l_7) = \mathrm{nk}(g')) \wedge (\text{Wert auf } l_7 \text{ ist stabil für } <e_1, e_2>)\\ &\Rightarrow \text{Belegung von } x_3 \text{ für Eingaben } e_1 \text{ und } e_2 \text{ gleich } 1\end{aligned}$$

Die möglichen Testsequenzen für $(p, 0 \to 1)$ sind:

$$<001, 111> \quad \text{und} \quad <101, 111>$$

■

1.4.2 Unvollständigkeit des Pfadverzögerungstests

Das vorige Beispiel enthält einen Schaltkreis, dessen Pfade mit dem Pfadverzögerungstest vollständig testbar sind. Wie im nachfolgenden Beispiel gezeigt wird, existieren aber auch kombinatorische Schaltkreise, deren Laufzeitverhalten mit Hilfe des Pfaddelaytests nicht vollständig überprüft werden kann. Für solche Schaltkreise ist die Methode des Pfadverzögerungstestes unvollständig.

Beispiel 1.7:
Sei ein kombinatorischer Schaltkreis $\mathcal{K}_2$ wie in Abb. 1.9 gegeben. Wir betrachten den physikalischen Pfad p_1 mit:

$$p_1 = l_3\text{-}l_6\text{-}l_8\text{-}l_{11}\text{-}l_{15}$$

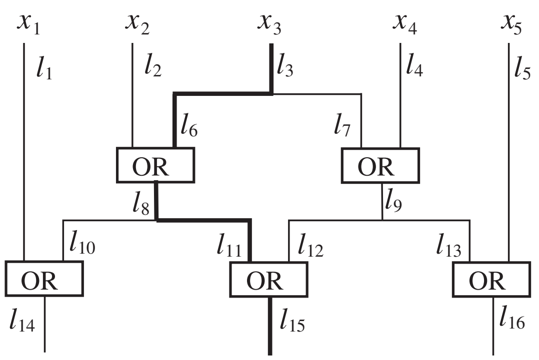

Abbildung 1.9: Beispielschaltkreis $\mathcal{K}_2$

Bei einem Test $<e_1, e_2>$ für den logischen Pfad $(p_1, 0 \to 1)$ liegt an l_{11} die $0 \to 1$ Transition an. Nach Lemma 1.19 muß an l_{12} gelten:

$$e_1(l_{12}) = e_2(l_{12}) = \mathrm{nk}(g) = 0, \; mit \; g = \mathcal{Z}(l_{11}).$$

Das ist aber nicht möglich, da $e_2(l_3) = 1$ impliziert, daß $e_2(l_{12}) = 1$ ist.
Derselbe Widerspruch ergibt sich beim Test des Pfades $p_2 = l_3\text{-}l_7\text{-}l_9\text{-}l_{12}\text{-}l_{15}$ für die Transiton $0 \to 1$. ■

Wie im vorigen Beispiel gezeigt, ist es nicht möglich, die logischen Pfade $(p_1, 0 \to 1)$ und $(p_2, 0 \to 1)$ als Einzelpfade zu testen. Wird an p_1 die Transition $0 \to 1$ angelegt, hat dies zur Folge, daß diese Transition ebenfalls über den Pfad p_2 propagiert wird. Wird also am Ausgang von l_{15} die logische 1 abgelesen, kann nicht geschlußfolgert werden, ob die Transition über den Pfad p_1 oder p_2 weitergeleitet wurde. Somit können die beiden logischen Pfade mit keiner Testmethode für Einzelpfade auf Delayfehler getestet werden. Es ist aber auch nicht nötig, diese Pfade als Einzelpfade zu testen. Für das korrekte Zeitverhalten des Schaltkreises genügt es, daß einer der beiden Pfade $(p_1, 0 \to 1)$ oder $(p_2, 0 \to 1)$ rechtzeitig schaltet. Es reicht folglich aus, das *Pfadsystem* bestehend aus beiden Pfaden zu testen. Eine mögliche Eingabefolge hierfür ist $< 00000, 00100 >$. Liest man bei Anlegen dieser Testfolge am Ausgang der Schaltung nach der Zeitspanne τ die logische Eins ab, kann man schlußfolgern:

$$d_{\mathcal{K}}(p_1, 0 \to 1) \leq \tau \quad \text{oder} \quad d_{\mathcal{K}}(p_2, 0 \to 1) \leq \tau$$

1.4.3 Der Pfadsystemtest

Zur Lösung des obigen Problems wurde von Ke und Menon [KM94] der *Pfadsystemtest* vorgeschlagen.

Definition 1.20:
Sei $\mathcal{K}$ ein kombinatorischer Schaltkreis, $P \subseteq \mathcal{LP}(\mathcal{K})$ und z eine Zelle in $\mathcal{K}$.

a) Wir sagen „z liegt in P“ g.d.w. mindestens ein Pfad von P durch z verläuft.

b) Für eine Zelle z in P ist:

$$\text{Flanken}(P, z) := \{Flanke_{(p_i,t_i)}(l) \mid (p_i, t_i) \in P \textit{ verläuft durch eine Inputleitung } l \textit{ von } z\}$$

Im folgenden werden nur spezielle Pfadmengen betrachtet, bei denen jeder Zelle auf obige Weise eine eindeutige Flanke zugeordnet ist.

Definition 1.21:
Sei $\mathcal{K}$ ein kombinatorischer Schaltkreis. Ein Pfadsystem *PS ist eine Menge von logischen Pfaden $PS \subseteq \mathcal{LP}(\mathcal{K})$, so daß:*

i) Alle Pfade von PS führen zum selben primären Ausgang von $\mathcal{K}$. Diesen primären Ausgang bezeichnen wir als den PO *von PS.*

ii) Für jede Zelle z in PS gilt:

$$|\text{Flanken}(PS, z)| = 1$$

Die eindeutig bestimmte Flanke aus Flanken(PS, z) *bezeichnen wir als* $\text{Flanke}_{PS}(z)$.

Beispiel 1.8:
Sei ein Schaltkreis $\mathcal{K}$ wie in Abbildung 1.10 gegeben. Die Pfade p_2 und p_4 sind dicker eingezeichnet:

$$p_2 = l_2\text{-}l_7\text{-}l_9\text{-}l_{12}\text{-}l_{14} \qquad p_4 = l_3\text{-}l_{10}\text{-}l_{12}\text{-}l_{14}$$

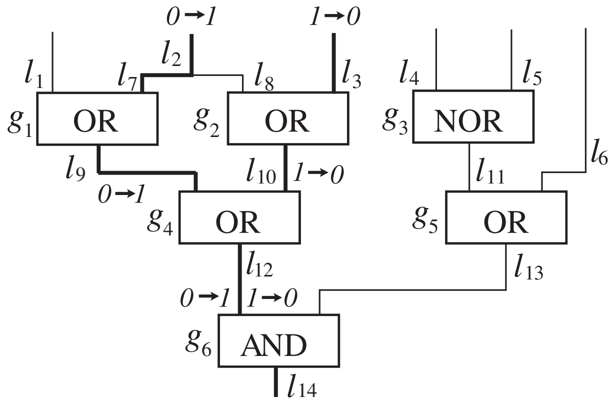

Abbildung 1.10: Pfadmenge P_1 zu $\mathcal{K}$

Die Pfadmenge $P_1 = \{(p_2, 0 \to 1), (p_4, 1 \to 0)\}$ ist kein Pfadsystem, da gilt:

$$Flanke_{(p_2,0\to1)}(l_9) = 0 \to 1$$
$$Flanke_{(p_4,1\to0)}(l_{10}) = 1 \to 0$$

Somit folgt für Flanken(P_1, g_4):

$$|\text{Flanken}(P_1, g_4)| = 2$$

Im folgenden betrachten wir die in Abbildung 1.11 durch dicke Linien angedeutete Menge von Pfaden

$$PS_1 = \{(p_5, 1 \to 0), (p_6, 1 \to 0), (p_7, 0 \to 1)\},$$

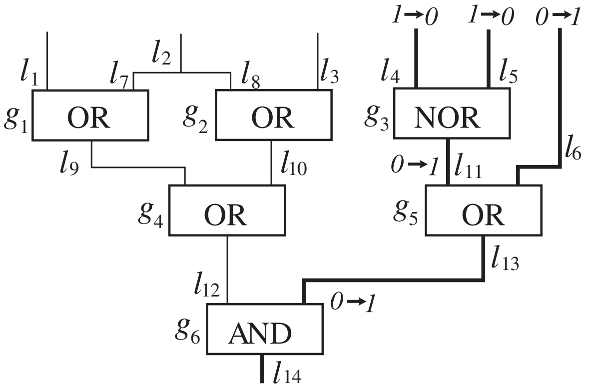

Abbildung 1.11: Pfadsystem PS_1 an $\mathcal{K}$

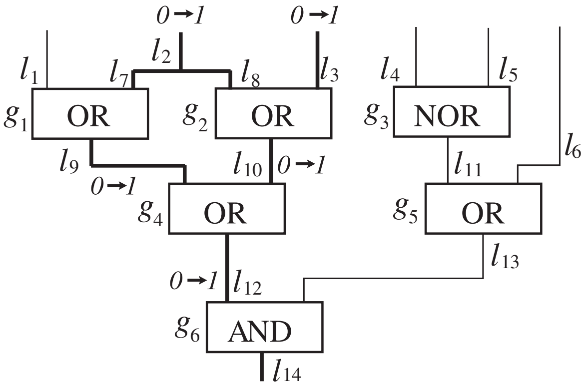

Abbildung 1.12: Pfadsystem PS_2 an $\mathcal{K}$

wobei die Pfade p_5, p_6 und p_7 wie folgt bestimmt sind:

$$p_5 = l_4\text{-}l_{11}\text{-}l_{13}\text{-}l_{14} \qquad p_6 = l_5\text{-}l_{11}\text{-}l_{13}\text{-}l_{14} \qquad p_7 = l_6\text{-}l_{13}\text{-}l_{14}$$

Diese Pfade bilden ein Pfadsystem, da gilt:

$$\begin{aligned}
&Flanke_{(p_5,1\to 0)}(l_4) = Flanke_{(p_6,1\to 0)}(l_5) = 1 \to 0\\
&Flanke_{(p_5,1\to 0)}(l_{11}) = Flanke_{(p_6,1\to 0)}(l_{11}) = Flanke_{(p_7,0\to 1)}(l_6) = 0 \to 1\\
&Flanke_{(p_5,1\to 0)}(l_{13}) = Flanke_{(p_6,1\to 0)}(l_{13}) = Flanke_{(p_7,0\to 1)}(l_{13}) = 0 \to 1
\end{aligned}$$

Somit gilt für die eindeutig bestimmten Flanken des Pfadsystems:

$$\begin{aligned}
\text{Flanke}_{PS_1}(g_3) &= 1 \to 0\\
\text{Flanke}_{PS_1}(g_5) &= 0 \to 1\\
\text{Flanke}_{PS_1}(g_6) &= 0 \to 1
\end{aligned}$$

Ein anderes Pfadsystem von $\mathcal{K}$ ist in Abbildung 1.12 dicker gezeichnet:

$$PS_2 = \{(p_2, 0 \to 1), (p_3, 0 \to 1), (p_4, 0 \to 1).\}$$

Hierbei sind die Pfade p_2, p_3 und p_4 wie folgt bestimmt:

$$p_2 = l_2\text{-}l_7\text{-}l_9\text{-}l_{12}\text{-}l_{14} \qquad p_3 = l_2\text{-}l_8\text{-}l_{10}\text{-}l_{12}\text{-}l_{14} \qquad p_4 = l_3\text{-}l_{10}\text{-}l_{12}\text{-}l_{14}$$

■

Definition 1.22:
Sei $\mathcal{K}$ ein kombinatorischer Schaltkreis. Die Tiefe *eines Pfadsystems PS zu $\mathcal{K}$ ist die Länge des längsten Pfades, der in PS enthalten ist.*

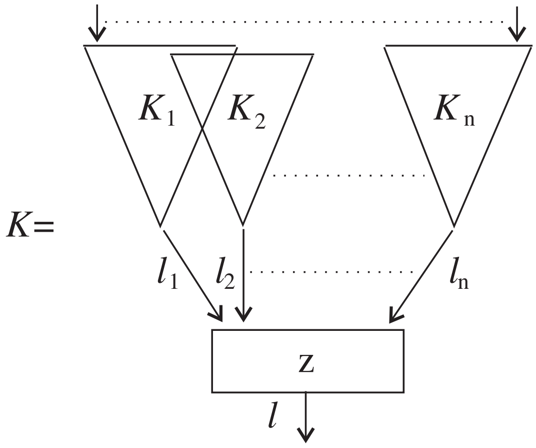

Abbildung 1.13: Situation an Zelle z

Bemerkung 1.3:
Wir betrachten einen Schaltkreis $\mathcal{K}$ mit einem primären Ausgang wie in Abbildung 1.13 gezeigt.

Sei z die Zelle, die den PO von $\mathcal{K}$ treibt. Die Inputleitungen von z seien die Leitungen $l_1, \ldots, l_n$. Den Teilschaltkreis von $\mathcal{K}$ zum i-ten Input von z bezeichnen wir mit $\mathcal{K}_i$.
Sei PS ein Pfadsystem in $\mathcal{K}$. $l_{i_1}, \ldots, l_{i_k}$ seien die Inputs von z, die in PS liegen.
Für $j \in \{1, \ldots, k\}$ sei

$$PS_{i_j} := \{(p\text{-}l_{i_j}, t) | \, (p\text{-}l_{i_j}\text{-}l, t) \in PS\}$$

die Menge aller logischen Teilpfade von PS, die in Leitung l_{i_j} enden. Dann sind $PS_{i_1}, \ldots, PS_{i_k}$ wieder Pfadsysteme zu den Schaltkreisen $\mathcal{K}_{i_1}, \ldots, \mathcal{K}_{i_k}$.

Intuitiv ist das Delay eines Pfadsystems das Delay des durch das Pfadsystem aufgespannten Teilschaltkreises (d.h. der Logik des Schaltkreises, die auf den Pfaden des Pfadsystems liegt) für die durch das Pfadsystem gegebenen Transitionen. Im folgenden wird eine formale Definition des Delays eines Pfadsystems gegeben:

Definition 1.23:
Das Delay eines Pfadsystems PS in Schaltkreis $\mathcal{K}$ bezeichnen wir mit $d_{\mathcal{K}}(PS)$. Es wird durch Induktion über die Tiefe von PS definiert:

a) Tiefe$(PS) = 1$, *d.h. PS besteht nur aus einer einzigen Leitung:*

$$d_{\mathcal{K}}(PS) := 0$$

b) Tiefe$(PS) = t$, *mit* $t \geq 2$:
Sei z die Zelle, die den PO *von PS treibt und* $l_{i_1}, \ldots, l_{i_k}$ *bzw.* $PS_{i_1}, \ldots, PS_{i_k}$ *wie in Bemerkung 1.3.*
Dann ist das Delay von PS rekursiv über die Delays der $PS_{i_j}(Tiefe(PS_{i_j}) < t$ *für* $j \in \{1, \ldots, k\})$ *definiert:*

1. *Ist z ein Fanoutknoten, so ist $k = 1$ und wir setzen*
$$d_{\mathcal{K}}(PS) := d_{\mathcal{K}}(PS_{i_1})$$
2. *Ist z kein Fanoutknoten, so gilt:*
$$d_{\mathcal{K}}(PS) := \begin{cases} \min\{d_{\mathcal{K}}(PS_{i_j}) + d_{Flanke_{PS}(z), l_{i_j}}(z) | \, j = 1, \ldots, k\} \\ \qquad \textit{falls der Endwert von } Flanke_{PS}(z) \textit{ kontrollierend ist.} \\ \\ \max\{d_{\mathcal{K}}(PS_{i_j}) + d_{Flanke_{PS}(z), l_{i_j}}(z) | \, j = 1, \ldots, k\} \\ \qquad \textit{falls der Endwert von } Flanke_{PS}(z) \textit{ nicht-kontrollierend} \\ \qquad \textit{ist.} \end{cases}$$

Im folgenden Beispiel wird zur Vereinfachung eine fallende Flanke $(1 \rightarrow 0)$ mit $\downarrow$ und eine steigende Flanke $(0 \rightarrow 1)$ mit $\uparrow$ abgekürzt. Ein Pfadsytem wird in das Klammerpaar „[,]“ eingeschlossen.

Beispiel 1.9:
Wir betrachten den Schaltkreis $\mathcal{K}$ aus Abbildung 1.14. Die Pfade p_5, p_6 und p_7 (in Abbildung 1.14 dick eingezeichnet) seien wie in Beispiel 1.8 gegeben. Zusätzlich seien einige Teilpfade des Schaltkreises wie folgt bestimmt:

$$\begin{array}{lll} p_5' = l_4\text{-}l_{11}\text{-}l_{13} & p_6' = l_5\text{-}l_{11}\text{-}l_{13} & p_7' = l_6\text{-}l_{13} \\ p_5'' = l_4\text{-}l_{11} & p_6'' = l_5\text{-}l_{11} & p_7'' = l_6 \\ p_5''' = l_4 & p_6''' = l_5 & \end{array}$$

Wir betrachten das in Beispiel 1.8 bestimmte Pfadsytem (siehe Abb. 1.14)
$$PS_1 = [(p_5, \downarrow), (p_6, \downarrow), (p_7, \uparrow)].$$
Diesem Pfadsystem ist nach Defintion 1.23 das folgende Delay zugeordnet:
$$\begin{aligned} d_{\mathcal{K}}(PS_1) &= \max\{d_{\mathcal{K}}([(p_5', \downarrow), (p_6', \downarrow), (p_7', \uparrow)]) + d_{\uparrow, l_{13}}(g_6)\} \\ &= d_{\mathcal{K}}([(p_5', \downarrow), (p_6', \downarrow), (p_7', \uparrow)]) + d_{\uparrow, l_{13}}(g_6) \\ &= \min\{d_{\mathcal{K}}([(p_5'', \downarrow), (p_6'', \downarrow)]) + d_{\uparrow, l_{11}}(g_5) \, , \, d_{\mathcal{K}}([(p_7'', \uparrow)]) + d_{\uparrow, l_6}(g_5)\} \\ &\quad + d_{\uparrow, l_{13}}(g_6) \\ &= \min\{\max\{d_{\mathcal{K}}([(p_5''', \downarrow)]) + d_{\downarrow, l_4}(g_3) \, , \, d_{\mathcal{K}}([(p_6''', \downarrow)]) + d_{\downarrow, l_5}(g_3)\} \\ &\qquad + d_{\uparrow, l_{11}}(g_5) \, , 0 + d_{\uparrow, l_6}(g_5)\} \; + d_{\uparrow, l_{13}}(g_6) \\ &= \min\{\max\{d_{\downarrow, l_4}(g_3) \, , \, d_{\downarrow, l_5}(g_3)\} + d_{\uparrow, l_{11}}(g_5) \, , \, d_{\uparrow, l_6}(g_5)\} \; + d_{\uparrow, l_{13}}(g_6) \\ &= \min\{\max\{d_{\mathcal{K}}(p_5, \downarrow), d_{\mathcal{K}}(p_6, \downarrow)\} \, , \, d_{\mathcal{K}}(p_7, \uparrow)\} \end{aligned}$$

■

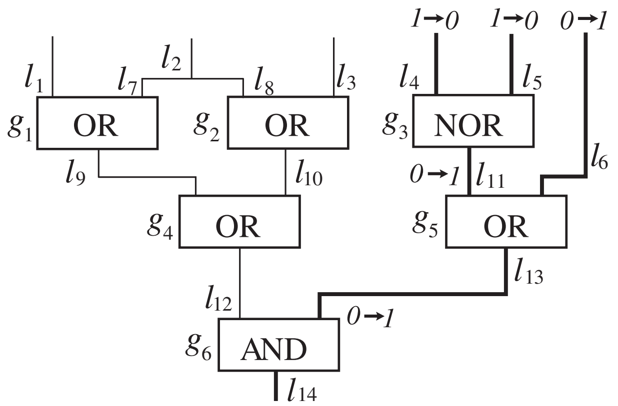

Abbildung 1.14: Beispielschaltkreis $\mathcal{K}$

Wie für einen Einzelpfad kann zu einem Pfadsystem auch wieder ein robuster Test definiert werden:

Definition 1.24:
Sei $< e_1, e_2 >$ *eine Eingabenfolge an den Schaltkreis* $\mathcal{K}$. $< e_1, e_2 >$ *heißt* robuster Test *für das Pfadsystem PS, wenn gilt:*

> *Funktioniert* $\mathcal{K}$ *korrekt für den Test* $< e_1, e_2 >$ *bei Laufzeitschranke* τ*, so kann gefolgert werden, daß* $d_{\mathcal{K}}(PS) \leq \tau$.

Bemerkung 1.4:
Setzt man die Anzahl der im System enthaltenen Pfade auf eins, sieht man, daß der robuste Einzelpfadtest einen Sonderfall des Pfadsystemtests darstellt.

Beispiel 1.10:
Wir betrachten den Schaltkreis $\mathcal{K}$ aus Abbildung 1.15. Die Pfade p_1 und p_2 sind dicker gezeichnet.

Betrachtet wird das Pfadsystem

$$PS = [(p_1, 0 \to 1)\ ,\ (p_2, 0 \to 1)].$$

Es wird die Eingabefolge

$$< e_1, e_2 >=< 00000, 00100 >$$

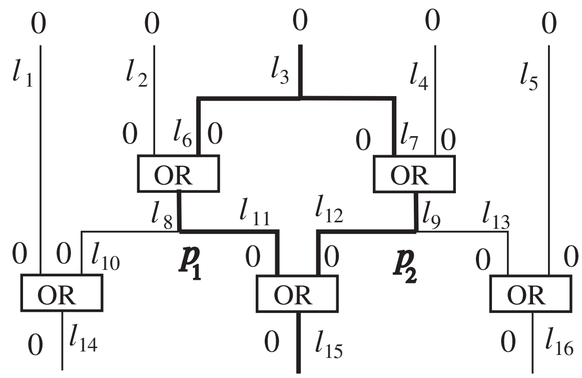

Abbildung 1.15: Belegung für e_1 an $\mathcal{K}$

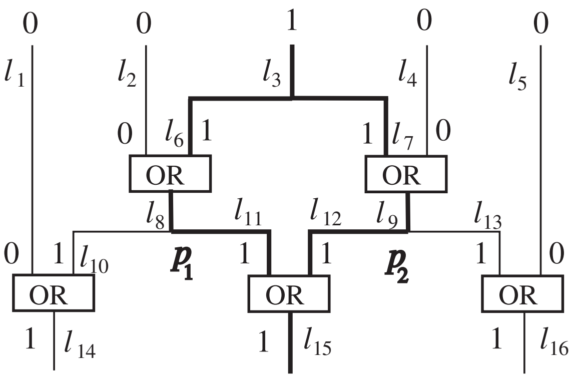

Abbildung 1.16: Belegung für e_2 an $\mathcal{K}$

an den Schaltkreis angelegt. In Abbildung 1.15 ist die Belegung aller Leitungen aus $\mathcal{K}$ bezüglich e_1, in Abbildung 1.16 die Belegung aller Leitungen bezüglich e_2 angegeben. Wie man sieht, werden durch e_1 alle Leitungen auf den Wert 0 initialisiert.

Die Leitung l_8 kann ihren logischen Endwert unter Eingabe e_2 erst annehmen, wenn die Transition von der Leitung l_6 über das Or-Gatter propagiert wurde (Leitung l_2 liegt stabil auf dem logischen Wert 0). Ebenfalls kann l_9 ihren Endwert erst annehmen, wenn die Transition von l_7 weiterpropagiert wurde. Für den Endwert von l_{15} reicht es aus, daß die Transition von l_{11} oder von l_{12} weiterpropagiert wird. Somit gilt:

$$\begin{aligned} &\mathcal{K} \text{ funktioniert korrekt für den Test } < e_1, e_2 > \text{ bei Laufzeitschranke } \tau \\ &\Longrightarrow \min\{d_{\mathcal{K}}(p_1, 0 \to 1), d_{\mathcal{K}}(p_2, 0 \to 1)\} \leq \tau \end{aligned}$$

Da $d_{\mathcal{K}}(PS) = \min\{d_{\mathcal{K}}(p_1, 0 \to 1), d_{\mathcal{K}}(p_2, 0 \to 1)\}$, ist somit $< e_1, e_2 >$ ein robuster Test für das Pfadsystem PS. ■

In den nächsten Kapiteln werden Testmengen vorgestellt, die für unate Schaltkreise auch das Laufzeitverhalten bestimmter Pfadsysteme überprüfen. Mit Hilfe dieser Testmengen können beliebige unate Schaltkreise vollständig auf Delayfehler getestet werden.

Kapitel 2

Eine funktionale Delaytestmenge

Im vorigen Kapitel wurde gezeigt, daß es Schaltkreise gibt, die mit dem Pfaddelaytest nicht vollständig auf Laufzeitfehler überprüft werden können. Als möglicher Ausweg aus diesem Problem wurde die Überprüfung des Laufzeitverhaltens von Pfadsystemen anstelle von Einzelpfaden skizziert.
Das Ziel der folgenden Kapitel ist es zu zeigen, daß mit Hilfe dieser verallgemeinerten Testmethodik die Klasse der unate Schaltkreise vollständig auf Laufzeitfehler überprüft werden kann.
Wir werden hierzu in zwei Schritten vorgehen. Zunächst wird eine funktionale Delaytestmenge angegeben, die nur von der durch den Schaltkreis berechneten logischen Funktion abhängt und die konkrete Schaltkreisrealisierung unberücksichtigt läßt [SR94, SMR96b]. Diese Testmenge ist relativ einfach zu konstruieren und hat den Vorteil, daß sie ein vollständiger Test für alle unate Realisierungen der Funktion ist. Falls also eine andere unate Realisierung für die Funktion gewählt wird, ändert sich die Testmenge nicht. Diese „Universalität“ der Testmenge wird durch eine erhöhte Testanzahl bezahlt. Daher betrachten wir im 3. Kapitel Methoden, um die funktionale Testmenge, falls ihre Größe nicht akzeptabel ist, auf eine gegebene Schaltkreisrealisierung zuzuschneiden. Die Testmenge wird hierbei verkleinert ohne die Eigenschaft des vollständigen Delaytests zu verlieren.
Im folgenden Paragraphen wird eine *funktionale Delaytestmenge* (FDT) definiert. Hierbei werden zuerst funktionale Delaytestmengen für monotone Funktionen angegeben und diese danach auf unate Funktionen verallgemeinert. Man kann sich bei der Berechnung der funktionalen Testmenge auf monotone Funktionen beschränken und diese durch eine geeignete Bijektion auf unate Funktionen übertragen.
Danach wird gezeigt, daß die Berechnung der FDT für monotone Funktionen auf die Primimplikantenberechnung reduziert werden kann. Wie man eine möglichst kompakte FDT für Schaltkreise mit mehreren Ausgängen durch Überlagerung der FDTs für die einzelnen Ausgangsfunktionen berechnen kann, wird im darauffolgenden Paragraphen erörtert. Zum Abschluß des Kapitels werden experimentelle Resultate vorgestellt, die bei der Testmengenerzeugung für über 50 Benchmarkschaltkreise erhalten wurden.

2.1 Definition der funktionalen Delaytestmenge

Wie schon erwähnt, läßt sich die funktionale Testmenge allein aus der logischen Funktion eines Schaltkreises konstruieren, wobei keine Information über die Realisierung der Schaltung verwendet wird, außer daß die Realisierung unate sein muß. Es wird zuerst eine funktionale Delaytestmenge für monotone Funktionen angegeben und danach gezeigt, wie man die Berechnung der FDT für unate Funktionen auf den monotonen Fall zurückführen kann.

2.1.1 Monotone Funktionen

Zur Beschreibung der Konstruktionsvorschrift für die funktionale Delaytestmenge für monotone Funktionen benötigen wir noch einige Definitionen:

Definition 2.1:
Seien $e = (e_1, \ldots, e_n)$, $e' = (e'_1, \ldots, e'_n) \in \boldsymbol{B}^n$ *Eingabevektoren.*
Wir definieren:

$$\begin{aligned} e > e' \quad \Longleftrightarrow \quad & \forall\, i \in \{1, \ldots, n\} : \; e_i \geq e'_i \\ & \mathit{und}\ \exists\, i \in \{1, \ldots, n\} : \; e_i > e'_i \end{aligned}$$

Bemerkung 2.1:
0^n (1^n) ist das kleinste (größte) Element bezüglich der Ordnung $>$.

Definition 2.2:
Sei $f : \boldsymbol{B}^n \to \boldsymbol{B}$ *eine monotone Funktion. Seien* $e, e' \in \boldsymbol{B}^n$ *Eingabevektoren für* f.

1. *Eine Eingabe* e *heißt* 1-Eingabe *von* f, *falls* $f(e) = 1$. *Die Menge aller 1-Eingaben von* f *bezeichnen wir mit* 1-Menge(f).

2. *Eine Eingabe* e *heißt* 0-Eingabe *von* f, *falls* $f(e) = 0$. *Die Menge aller 0-Eingaben von* f *bezeichnen wir mit* 0-Menge(f).

3. *Eine Eingabe* $e \in$ *1-Menge*(f) *heißt* minimale 1-Eingabe *von* f, *falls es keine andere Eingabe* e' *gibt, mit:*
$$(e' \in \textit{1-Menge}(f)) \wedge (e' < e)$$
Die Menge aller minimalen 1-Eingaben von f *bezeichnen wir mit* MIN-1(f) .

4. *Eine Eingabe $e \in$ 0-Menge(f), heißt* maximale 0-Eingabe *von f, falls es keine andere Eingabe e' gibt, mit:*
$$(e' \in \text{0-Menge}(f)) \wedge (e' > e)$$
Die Menge aller maximalen 0-Eingaben von f bezeichnen wir mit MAX-0(f) .

Beispiel 2.1 :
Sei $f = x_2 \vee x_1x_3$. Die Funktionstabelle von f lautet:

Eingabe	x_1	x_2	x_3	f
e_1	0	0	0	0
e_2	0	0	1	0
e_3	0	1	0	1
e_4	0	1	1	1
e_5	1	0	0	0
e_6	1	0	1	1
e_7	1	1	0	1
e_8	1	1	1	1

Zuerst wird die Menge MIN-1(f) gebildet. Dafür werden alle Elemente der 1-Menge untereinander verglichen. Man sieht: $e_3 < e_4$, $e_3 < e_7$ und $e_3 < e_8$, d.h. e_4, e_7 und e_8 sind keine Elemente von MIN-1(f) . Für e_3 und e_6 gilt:

$$\begin{aligned}&\nexists\, e \in \text{ 1-Menge}(f) \text{ mit } e < e_j \; (j = 3, 6)\\ &\Rightarrow \text{MIN-1}(f) = \{e_3, e_6\}\end{aligned}$$

Für die Elemente der Nullmenge gilt:

$$\begin{aligned}&e_2 > e_1 \wedge \nexists\, e \in \text{ 0-Menge}(f) \text{ mit } e > e_j \; (j = 2, 5)\\ &\Rightarrow \text{MAX-0}(f) = \{e_2, e_5\}\end{aligned}$$

■

Es ist schon sehr lange bekannt, daß die Menge MIN-1$(f) \cup$ MAX-0(f) eine funktionale Testmenge zur Überprüfung von stuck-at-Fehlern in quasi-monotonen Schaltkreisen darstellt.

Satz 2.3 ([Bet71, Ake73, Red73])
Sei $f : \boldsymbol{B}^n \to \boldsymbol{B}$ eine monotone Funktion.

Die Testmenge MIN-1$(f) \cup$ MAX-0(f) *überprüft jede quasi-monotone Realisierung von f vollständig auf Einzel- und Mehrfach-stuck-at Fehler.*

Beweisskizze:
Der Beweis beruht auf den folgenden Beobachtungen:

1. Wird ein quasi-monotoner Schaltkreis durch einen stuck-at-Fehler verfälscht, so berechnet der resultierende Schaltkreis wieder eine monotone Funktion.
2. Um zwei monotone Funktionen $f, f' : \boldsymbol{B}^n \to \boldsymbol{B}$ mit $f \neq f'$ voneinander zu unterscheiden, genügt es, die Ausgaben von f und f' für die Eingabevektoren aus MIN-1$(f) \cup$ MAX-0(f) zu vergleichen.

   (Annahme: $\exists e \in \boldsymbol{B}^n$ mit $f(e) \neq f'(e)$ o.B.d.A. $f(e) = 1,\ f'(e) = 0$.
   Betrachte $\tilde{e} \in$ MIN-1(f) mit $\tilde{e} \leq e$. Dann gilt: $f(\tilde{e}) = 1$ und da f' monoton ist $f'(\tilde{e}) = 0$. Also gilt auch: $f(\tilde{e}) \neq f'(\tilde{e})$.)

■

Beispiel 2.2:
Sei die Funktion f wie in Beispiel 2.1 gegeben.
Die Menge

$$\text{MIN-1}(f) \cup \text{MAX-0}(f) = \{e_3, e_6\} \cup \{e_2, e_5\} = \{e_2, e_3, e_5, e_6\}$$

stellt eine vollständige stuck-at-Testmenge für alle quasi-monotonen Realisierungen von f dar. ■

Die Mengen MIN-1(f) und MAX-0(f) sind noch mächtiger, als es durch Satz 2.3 impliziert wird. Ergänzt man jedes Element dieser Mengen durch einen geeigneten Initialvektor, so stellt die resultierende Menge von Testsequenzen einen *vollständigen Delaytest* für alle quasi-monotonen Realisierungen der Funktion f dar.

Definition 2.4:
Sei $f : \boldsymbol{B}^n \to \boldsymbol{B}$ eine monotone Funktion. Wir definieren eine funktionale Delaytestmenge zu f vermöge:

$$\text{FDT}(f) := \{< 0^n, e > | e \in \text{MIN-1}(f)\} \cup \{< 1^n, e > | e \in \text{MAX-0}(f)\}.$$

Satz 2.5 ([SR94])
Sei $f : \boldsymbol{B}^n \to \boldsymbol{B}$ eine monotone Funktion und $\mathcal{K}$ eine beliebige quasi-monotone Realisierung von f.

> *Falls $\mathcal{K}$ für alle Testsequenzen aus $\mathrm{FDT}(f)$ bei Clock-Periode τ korrekt arbeitet, so gilt:*
> $$d(\mathcal{K}) \leq \tau$$

Beispiel 2.3:
Sei die Funktion f wie in Beispiel 2.1 gegeben.
Die Menge
$$\mathrm{FDT}(f) = \{< 1^3, e_2 >, < 0^3, e_3 >, < 1^3, e_5 >, < 0^3, e_6 >\}$$
stellt eine vollständige Delaytestmenge für alle quasi-monotonen Realisierungen von f dar. ∎

2.1.2 Unate Funktionen

Wie läßt sich nun eine vollständige Delay-Testmenge für eine unate Funktion finden? Hierbei kann der Sachverhalt ausgenutzt werden, daß sich eine unate Funktion durch die im folgenden definierte Bijektion auf eine monotone Funktion zurückführen läßt. Der Einfachheit halber betrachten wir im folgenden nur unate Funktionen $f : \boldsymbol{B}^n \to \boldsymbol{B}$, die von allen Eingabevariablen echt abhängen.

Definition 2.6:
Sei $f : \boldsymbol{B}^n \to \boldsymbol{B}$ eine unate Funktion, die von allen Variablen echt abhängt und die positiv unate in den Variablen $x_1, \ldots, x_k$ und negativ unate in den Variablen $x_{k+1}, \ldots, x_n$ ist. Wir definieren eine Bijektion $\pi_f : \boldsymbol{B}^n \to \boldsymbol{B}^n$ mit:

$$\pi_f(\alpha_1, \ldots, \alpha_k, \alpha_{k+1}, \ldots, \alpha_n) := (\alpha_1, \ldots, \alpha_k, \overline{\alpha_{k+1}}, \ldots, \overline{\alpha_n}) \ \forall\, (\alpha_1, \ldots, \alpha_n) \in \boldsymbol{B}^n.$$

Lemma 2.7:
Seien f und π_f wie in Definition 2.6, dann ist $f \circ \pi_f$ monoton.

Beweis:
Wir müssen zeigen, daß $\forall\, i \in \{1, \ldots, n\}$ gilt:

$$(f \circ \pi_f)(\alpha_1, \ldots, \alpha_{i-1}, 0, \alpha_{i+1}, \ldots, \alpha_n) \leq (f \circ \pi_f)(\alpha_1, \ldots, \alpha_{i-1}, 1, \alpha_{i+1}, \ldots, \alpha_n)$$
$$\forall\, (\alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_n) \in \boldsymbol{B}^{n-1}$$

<u>Fall 1</u>: $i \leq k$

$$f \circ \pi_f(\alpha_1, \ldots, 0, \ldots, \alpha_k, \alpha_{k+1}, \ldots, \alpha_n) = f(\alpha_1, \ldots, 0, \ldots, \alpha_k, \overline{\alpha_{k+1}}, \ldots, \overline{\alpha_n})$$
$$\leq f(\alpha_1, \ldots, 1, \ldots, \alpha_k, \overline{\alpha_{k+1}}, \ldots, \overline{\alpha_n}) = f \circ \pi_f(\alpha_1, \ldots, 1, \ldots, \alpha_k, \alpha_{k+1}, \ldots, \alpha_n)$$
(da f positiv unate in x_i)

<u>Fall 2</u>: $i > k$

$$f \circ \pi_f(\alpha_1, \ldots, \alpha_k, \alpha_{k+1}, \ldots, 0, \ldots, \alpha_n) = f(\alpha_1, \ldots, \alpha_k, \overline{\alpha_{k+1}}, \ldots, 1, \ldots, \overline{\alpha_n})$$
$$\leq f(\alpha_1, \ldots, \alpha_k, \overline{\alpha_{k+1}}, \ldots, 0, \ldots, \overline{\alpha_n}) = f \circ \pi_f(\alpha_1, \ldots, \alpha_k, \alpha_{k+1}, \ldots, 1, \ldots, \alpha_n)$$
(da f negativ unate in x_i)

■

Wir definieren nun die FDT für eine unate Funktion f über die FDT der zugehörigen monotonen Funktion $f \circ \pi_f$.

Definition 2.8:
Seien f und π_f wie in Definition 2.6. Wir definieren eine funktionale Delaytestmenge zu f vermöge:

$$\mathrm{FDT}(f) := \{< \pi_f(e_1), \pi_f(e_2) > \mid\ < e_1, e_2 > \in \mathrm{FDT}(f \circ \pi_f)\}$$

Bemerkung 2.2:
Ist f eine monotone Funktion, so stimmt Definition 2.8 mit Definition 2.4 überein, da hierbei π_f die Identität von $\boldsymbol{B}^n \rightarrow \boldsymbol{B}^n$ ist.

Satz 2.9 ([SR94])
Seien f und π_f wie in Definition 2.6 und $\mathcal{K}$ eine beliebige Realisierung von f als unate Schaltkreis.

Falls $\mathcal{K}$ für alle Testsequenzen aus FDT(f) *bei Clockperiode τ korrekt arbeitet, so gilt:*

$$d(\mathcal{K}) \leq \tau$$

Da sich mit Hilfe des obigen Satzes die Berechnung der funktionalen Delaytestmenge für eine unate Funktion auf den monotonen Fall zurückführen läßt, kann man sich auf die Berechnung der FDT für monotone Funktionen beschränken.

Der Beweis der Sätze 2.5 und 2.9 beruht auf den folgenden beiden Lemmata, die für Kapitel 3 noch von Bedeutung sein werden.

Definition 2.10 :
Sei f wie in Definition 2.6. Die Menge E(f) *der essentiellen Eingabevektoren zu f ist definiert als:*

$$\mathrm{E}(f) := \pi_f(\text{MIN-1}(f \circ \pi_f) \,\cup\, \text{MAX-0}(f \circ \pi_f))$$

Bemerkung 2.3 :
Falls f monoton ist, so gilt:

$$\mathrm{E}(f) = \text{MIN-1}(f) \cup \text{MAX-0}(f)$$

Lemma 2.11 :
Sei f wie in Definition 2.6 und $\mathcal{K}$ eine beliebige Realisierung von f als unate Schaltkreis, dann gilt:

$$d_{\mathcal{K}}(e) \leq \tau \;\forall\, e \in \mathrm{E}(f) \implies d(\mathcal{K}) \leq \tau$$

Es genügt also nur das Delay von essentiellen Eingabevektoren zu überprüfen, um zu zeigen, daß der Schaltkreis bei Laufzeitschranke τ korrekt funktioniert.

Die Überprüfung des Delays einer Eingabe aus E(f) kann mit Hilfe einer einzigen Testsequenz erfolgen:

Lemma 2.12 :
Sei f wie in Definition 2.6 und $\mathcal{K}$ eine beliebige Realisierung von f als unate Schaltkreis. Nach Definition von FDT(f) *gibt es zu jedem essentiellen Eingabevektor $e \in$* E(f) *genau eine Testsequenz $< e_1, e_2 > \in$* FDT(f) *mit $e_2 = e$. Für diese Testsequenz gilt:*

Falls $\mathcal{K}$ für Testsequenz $< e_1, e >$ bei Clockperiode τ korrekt arbeitet, so ist $d_{\mathcal{K}}(e) \leq \tau$.

Der Beweis dieser beiden Lemmata, der auf [SR94] zurückgeht, ist etwas technisch und für das Verständnis der folgenden Kapitel nicht notwendig. Der interessierte Leser findet den Beweis der Lemmata 2.11 und 2.12 in [SR94, SMR96b].

Beispiel 2.4:
Sei eine unate Funktion f wie folgt gegeben:

$$f = \overline{x_2} \vee x_1\overline{x_3}$$

f ist positiv unate in der Variablen x_1 und negativ unate in den Variablen x_2 und x_3, somit ist die Bijektion $\pi_f : \boldsymbol{B}^3 \rightarrow \boldsymbol{B}^3$ wie folgt definiert:

$$\pi_f(\alpha_1, \alpha_2, \alpha_3) = (\alpha_1, \overline{\alpha_2}, \overline{\alpha_3})$$

Die f zugeordnete monotone Funktion ergibt sich zu:

$$f \circ \pi_f = x_2 \vee x_1x_3$$

Eingabe	x_1	x_2	x_3	$f \circ \pi_f$
e_1	0	0	0	0
e_2	0	0	1	0
e_3	0	1	0	1
e_4	0	1	1	1
e_5	1	0	0	0
e_6	1	0	1	1
e_7	1	1	0	1
e_8	1	1	1	1

Eingabe	x_1	x_2	x_3	f
e_1	0	0	0	1
e_2	0	0	1	1
e_3	0	1	0	0
e_4	0	1	1	0
e_5	1	0	0	1
e_6	1	0	1	1
e_7	1	1	0	1
e_8	1	1	1	0

Die FDT($f \circ \pi_f$) ist (zur Berechnung der FDT der Funktion $x_2 \vee x_1x_3$ siehe Beispiel 2.1, Seite 33):

$$\begin{aligned} \mathrm{FDT}(f \circ \pi_f) &= \{< 000, e_3 >, < 000, e_6 >, < 111, e_2 >, < 111, e_5 >\} \\ &= \{< 000, 010 >, < 000, 101 >, < 111, 001 >, < 111, 100 >\} \end{aligned}$$

Für $\mathrm{FDT}(f) = \pi_f(\mathrm{FDT}(f \circ \pi_f))$ erhält man dann:

$$\mathrm{FDT}(f) = \{< 011, 001 >, < 011, 110 >, < 100, 010 >, < 100, 111 >\}$$

Die essentiellen Eingabevektoren von f sind:

$$\begin{aligned} \mathrm{E}(f) &= \pi_f(\text{MIN-1}(f \circ \pi_f) \cup \text{MAX-0}(f \circ \pi_f)) \\ &= \pi_f(\{010, 101\} \cup \{001, 100\}) \\ &= \{001, 110, 010, 111\} \end{aligned}$$

Für jeden essentiellen Eingabevektor e existiert ein Test in FDT(f), der überprüft, ob $d_{\mathcal{K}}(e) \leq \tau$. ■

Somit ist eine Delaytestmenge definiert, die für eine unate Funktion f alle ihre Realisierungen als unate Schaltkreis vollständig auf Delayfehler überprüft. Im nächsten Paragraphen wird eine einfache Methode erörtert, diese Testmenge zu konstruieren.

2.2 Berechnung der funktionalen Delaytestmenge

In diesem Kapitel wird gezeigt, daß die Berechnung der Mengen MIN-1(f) und MAX-0(f) auf die Berechnung der Primimplikanten von f und $\overline{f}$ zurückgeführt werden kann.

2.2.1 Berechnung von MIN-1(f)

Sei f eine Funktion in n Variablen $x_1, \ldots, x_n$. $PI(f)$ bezeichne die Menge der Primimplikanten (siehe [Hot90]) von f.
Wir definieren:

$$UM(x_1, \ldots, x_n) := \{x_{i_1} \ldots x_{i_k} \,|\, i_1 \neq \ldots \neq i_k \in \{1, \ldots, n\}\} \cup \{1\}$$

als die Menge aller *unnegierten Monome* über der Variablenmenge $\{x_1, \ldots, x_n\}$.

Falls f monoton ist, so gilt [Hot90]:

$$PI(f) \subseteq UM(x_1, \ldots, x_n)$$

Jedem Monom $m \in UM(x_1, \ldots, x_n)$ kann vermöge der folgenden bijektiven Abbildung φ eindeutig ein Eingabevektor aus $\boldsymbol{B}^n$ zugeordnet werden:

$$\begin{aligned} &\varphi : UM(x_1, \ldots, x_n) \rightarrow \boldsymbol{B}^n \\ &\varphi(m) = \alpha_1 \ldots \alpha_n, \text{ mit } \quad (\alpha_i = 1) \Longleftrightarrow (x_i \text{ kommt in } m \text{ vor}) \end{aligned}$$

Durch die Abbildung φ kann aus $PI(f)$ die Menge MIN-1(f) abgeleitet werden.

Lemma 2.13:
Sei f eine monotone Funktion in den Variablen $x_1, \ldots, x_n$ und φ wie oben definiert. Dann gilt:

$$\text{MIN-1}(f) = \varphi(PI(f))$$

Beweis:
Der Beweis beruht auf der folgenden Beobachtung:

$$\begin{aligned} & m \in UM(x_1, \ldots, x_n) \text{ ist Implikant von } f \\ \Longleftrightarrow\ & \varphi(m) \in \text{ 1-Menge}(f) \end{aligned}$$

„$\Longrightarrow$" klar, da m bei Einsetzung $\varphi(m)$ erfüllt ist.

„$\Longleftarrow$" $f(\varphi(m)) = 1$ und f monoton impliziert, daß $f(\alpha) = 1 \; \forall\, \alpha \geq \varphi(m)$. Also ist m Implikant von f.

Sicherlich gilt $\forall\, m, m' \in UM(x_1, \ldots, x_n)$:

$$\begin{aligned} & m \text{ ist echtes Teilmonom (siehe [Hot90]) von } m' \\ \Longrightarrow\ & \varphi(m) < \varphi(m') \end{aligned}$$

Also entsprechen die minimalen Elemente von 1-Menge(f) bezüglich φ genau den bezüglich der Teilmonom-Beziehung kleinsten Implikanten von f aus $UM(x_1, \ldots, x_n)$, d.h.:

$$\text{MIN-1}(f) = \varphi(PI(f))$$

■

2.2.2 Berechnung von MAX-0(f)

Sei

$$NM(x_1, \ldots, x_n) := \{\overline{x_{i_1}} \ldots \overline{x_{i_k}} \,|\, i_1 \neq \ldots \neq i_k \in \{1, \ldots, n\}\} \cup \{0\}$$

die Menge aller *negierten Monome* über der Variablenmenge $\{x_1, \ldots, x_n\}$.

Falls f monoton ist, so gilt für $\overline{f}$:

$$PI(\overline{f}) \subseteq NM(x_1, \ldots, x_n)$$

Da f monoton ist, ist $\overline{f}$ negativ unate in allen Variablen. Somit kommen in den Primimplikanten zu $\overline{f}$ alle Variablen nur negiert vor (Beweis: siehe Anhang A).

Jedem Monom $m \in NM(x_1, \ldots, x_n)$ kann vermöge der folgenden bijektiven Abbildung φ^* eindeutig ein Eingabevektor aus $\boldsymbol{B}^n$ zugeordnet werden:

$$\begin{aligned} & \varphi^* : NM(x_1, \ldots, x_n) \to \boldsymbol{B}^n \\ & \varphi^*(m) = \alpha_1 \ldots \alpha_n, \text{ mit } \; (\alpha_i = 0) \Longleftrightarrow (\overline{x_i} \text{ kommt in } m \text{ vor}) \end{aligned}$$

Lemma 2.14 :
Sei f eine monotone Funktion in den Variablen $x_1, \ldots, x_n$ und φ^* wie oben definiert. Dann gilt:

$$\text{MAX-0}(f) = \varphi^*(PI(\overline{f}))$$

Beweis:
Es gilt:

$$\begin{aligned} & m \in NM(x_1, \ldots, x_n) \text{ ist Implikant von } \overline{f} \\ & \Longleftrightarrow \varphi^*(m) \in \text{0-Menge}(f) \end{aligned}$$

„$\Longrightarrow$“ klar, da m bei Einsetzung $\varphi^*(m)$ erfüllt ist, also

$$\overline{f}(\varphi^*(m)) = 1 \text{ und somit } f(\varphi^*(m)) = 0.$$

„$\Longleftarrow$“ $f(\varphi^*(m)) = 0$ und f monoton impliziert, daß $f(\alpha) = 0 \; \forall \, \alpha \leq \varphi^*(m)$. Also ist m Implikant von $\overline{f}$.

Sicherlich gilt $\forall \, m, m' \in NM(x_1, \ldots, x_n)$:

$$\begin{aligned} & m \text{ ist echtes Teilmonom von } m' \\ & \Longrightarrow \varphi^*(m) > \varphi^*(m') \end{aligned}$$

Also entsprechen die maximalen Elmenente von 0-Menge(f) bezüglich φ^* genau den bezüglich der Teilmonom-Beziehung kleinsten Implikanten von $\overline{f}$ aus $NM(x_1, \ldots, x_n)$, d.h.:

$$\text{MAX-0}(f) = \varphi^*(PI(\overline{f}))$$

■

Zur Erzeugung der Primimplikanten kann auf ein Softwarepaket zur Schaltkreissynthese zurückgegriffen werden:

> Ist eine logische Funktion f gegeben, so können mit dem Programm **ESPRESSO** (siehe [BHMSV84]) die Primimplikanten zu dieser Funktion und der dazugehörigen negierten Funktion $\overline{f}$ bestimmt werden. Hierbei wird nach dem Verfahren von Quine-McClusky (siehe [Hot90]) vorgegangen.

Bei unseren bisherigen Betrachtungen sind wir von Schaltkreisen mit einem Ausgang ausgegangen. Wie man die funktionale Delaytestmenge zu einem unate Schaltkreis mit mehreren Ausgängen berechnet, wird im nächsten Paragraphen erörtert.

2.3 Funktionaler Delaytest von multi-output-Schaltkreisen

Sei $\mathcal{K}$ ein Schaltkreis mit m Outputs. $\mathcal{K}_i$, $i \in \{1, \ldots, m\}$, bezeichne den Teilschaltkreis von $\mathcal{K}$ zum i-ten Output. Zur Überprüfung des Laufzeitverhaltens von $\mathcal{K}$ testen wir jeden Teilschaltkreis $\mathcal{K}_i$ mit Hilfe des in Kapitel 2.1 eingeführten funktionalen Delaytests. Um die Größe des Gesamttests für $\mathcal{K}$ zu minimieren, wird dabei versucht, die Tests der einzelnen $\mathcal{K}_i$ möglichst stark zu parallelisieren. Hierzu nutzen wir eine in [CL94] eingeführte Technik.

Die parallele Ausführung der Tests der einzelnen $\mathcal{K}_i$ wird dadurch erleichtert, daß in der Regel jedes $\mathcal{K}_i$ nur von einer Teilmenge aller Inputs von $\mathcal{K}$ abhängt. Das folgende Beispiel verdeutlicht diese Beobachtung.

Beispiel 2.5 :
Gegeben sei der unate Schaltkreis $\mathcal{K}$ aus Abbildung 2.1 mit den Teilschaltkreisen $\mathcal{K}_1$ und $\mathcal{K}_2$. Diese berechnen die Funktionen f_1 und f_2:

$$\begin{aligned} f_1 &= \overline{x_2} \vee x_1\overline{x_3} \\ f_2 &= x_4 \vee x_1 x_5 \end{aligned}$$

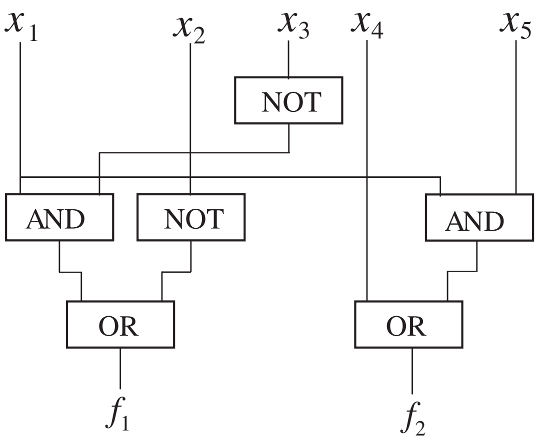

Abbildung 2.1: Beispielschaltkreis $\mathcal{K}$

Die funktionalen Delaytestmengen zu diesen Funktionen sind: [1]

$$\mathrm{FDT}(f_1) = \{< e_1, e_2 >, < e_3, e_4 >, < e_5, e_6 >, < e_7, e_8 >\} \text{ mit:}$$

[1] Für die Berechnung der funktionalen Delaytestmenge zu f_1 siehe Beispiel 2.4, Seite 39. f_2 entspricht der Funktion f aus Beispiel 2.3, Seite 35, durch Variablenumbenennung.

$$
\begin{aligned}
& < e_1, e_2 > = < 100\text{--}, 010\text{--} >, \ < e_3, e_4 > = < 011\text{--}, 001\text{--} >, \\
& < e_5, e_6 > = < 100\text{--}, 111\text{--} >, < e_7, e_8 > = < 011\text{--}, 110\text{--} > \\
\mathrm{FDT}(f_2) \ = \ & \{< e_9, e_{10} >, < e_{11}, e_{12} >, < e_{13}, e_{14} >, < e_{15}, e_{16} >\} \text{ mit:} \\
& < e_9, e_{10} > = < 1\text{--}11, 0\text{--}01 >, < e_{11}, e_{12} > = < 0\text{--}00, 0\text{--}10 >, \\
& < e_{13}, e_{14} > = < 1\text{--}11, 1\text{--}00 >, < e_{15}, e_{16} > = < 0\text{--}00, 1\text{--}01 >
\end{aligned}
$$

Hierbei sind don't-care-positions, d.h. Eingänge die auf beliebige Werte aus $\{0, 1\}$ gesetzt werden dürfen, durch „-“ gekennzeichnet.
Versucht man die Tests der Einzelfunktionen parallel auszuführen, so muß man die Eingabetupel der funktionalen Delaytestmengen bezüglich ihrer Kompatibilität überprüfen.

Beispielsweise sind die beiden Eingabefolgen $< e_1, e_2 >$ und $< e_{13}, e_{14} >$ nicht kompatibel, da die Eingaben e_2 und e_{14} in der Variablen x_1 nicht kompatibel sind.

Vergleicht man die Eingabefolgen $< e_1, e_2 >$ und $< e_9, e_{10} >$, erhält man, daß e_1 zu e_9 und e_2 zu e_{10} kompatibel ist. Diese beiden Testsequenzen können parallel durch Anlegen der Testsequenz $< e_{1+9}, e_{2+10} > = < 10011, 01001 >$ ausgeführt werden.
Ebenso erhält man, daß die Eingabefolgen $< e_3, e_4 >$ und $< e_{11}, e_{12} >$, die Eingabefolgen $< e_5, e_6 >$ und $< e_{13}, e_{14} >$ und die Eingabefolgen $< e_7, e_8 >$ und $< e_{15}, e_{16} >$ kompatibel sind. Somit kann $\mathcal{K}$ mit nur 4 Testsequenzen

$$\{< 10011, 01001 >, < 01100, 00110 >, < 10011, 11100 >, < 01100, 11001 >\}$$

vollständig auf Delayfehler überprüft werden. ∎

Bemerkung 2.4:
Durch diese Vorgehensweise kann die Größe der funktionalen Delaytestmenge zu $\mathcal{K}$ im günstigsten Fall bis auf $\max\{|\mathrm{FDT}(f_i)| \,|\, i = 1, \ldots, m\}$ gedrückt werden.

Das Problem der optimalen Kombination solcher Testmengen mit don't-care-positions ist NP-hart (siehe [CL94]). Deshalb verwenden wir zur Testmengenkombination eine heuristische Vorgehensweise:

Sei T_i die Menge der Testsequenzen zu f_i.
$T := T_1$;
for $(i = 2, \ldots, m)$ do
$T :=$ Kombination von T mit T_i;
od;

Beobachtung:
Die Reihenfolge, in der die T_i kombiniert werden, spielt eine große Rolle für das Endresultat. Es ist von Vorteil die T_i mit den wenigsten don't-cares zuerst zu betrachten, da sie am schwersten zu kombinieren sind.

Beispiel 2.6:
Seien folgende Eingabefolgen $< e_1, e_2 >, \ldots, < e_9, e_{10} >$ aus Testmengen $T_1, \ldots, T_5$ gegeben:

$$\begin{aligned} < e_1, e_2 > &= < \text{0---}, \text{1---} > \\ < e_3, e_4 > &= < \text{-110}, \text{-100} > \\ < e_5, e_6 > &= < \text{-0-0}, \text{-1-0} > \\ < e_7, e_8 > &= < \text{100-}, \text{110-} > \\ < e_9, e_{10} > &= < \text{10-0}, \text{11-0} > \end{aligned}$$

Für die Anzahl der don't-care-positions der einzelnen Testmengen gilt:

$$\text{don't-cares}(T_1) = 3,\ \text{don't-cares}(T_3) = 2,\ \text{don't-cares}(T_2, T_4, T_5) = 1$$

Bearbeitet man die Testmengen mit wenigen don't-cares zuerst, so werden zuerst T_2, T_4 und T_5, danach T_3 und abschließend T_1 betrachtet. In unserem Beispiel sind $< e_7, e_8 >$ und $< e_9, e_{10} >$ kompatibel. $< e_3, e_4 >$ ist zu keinem der beiden kompatibel. Die Menge T_{2+4+5} enthält somit die Elemente:

$$T_{2+4+5} = \{\ldots, < 1000, 1100 >, < \text{-110}, \text{-100} >, \ldots\}$$

Nun wird die Eingabefolge $< e_5, e_6 >$ betrachtet, die auf das Verknüpfungsergebnis von $< e_7, e_8 >$ und $< e_9, e_{10} >$ paßt. Eingabe $< e_1, e_2 >$ ist zur Eingabe $< e_3, e_4 >$ kompatibel. Somit ergibt sich für die Menge $T_{2+4+5+3+1}$:

$$T_{2+4+5+3+1} = \{\ldots, < 1000, 1100 >, < 0110, 1100 >, \ldots\}$$

Hätte man die Testmengen in der umgekehrten Reihenfolge betrachtet, so hätte sich folgende Menge $T_{1+3+2+4+5}$ ergeben:

$$T_{1+3+2+4+5} = \{\ldots, < \text{00-0}, \text{11-0} >, < 1000, 1100 >, < \text{-110}, \text{-100} >, \ldots\}$$

Sowohl die Eingabefolge $< e_1, e_2 >$ als auch die Eingabefolge $< e_5, e_6 >$ wäre mit einer Eingabefolge mit weniger don't-care-positions verknüpfbar gewesen. Da diese beiden Folgen jedoch untereinander kombiniert werden, wird ihnen die größere Flexibilität genommen. Das Verknüpfungsergebnis ist mit keiner weiteren Eingabefolge mehr kombinierbar. ■

Um die Elemente aus T_i mit T zu verknüpfen, gibt es mehrere mögliche heuristische Vorgehensweisen:

- Die Elemente aus T_i werden vorsortiert:
  Nach der neuen Ordnung werden Elemente, die mit wenigen Eingaben aus T verknüpfbar sind, zuerst bearbeitet. Besser kompatible Elemente werden erst gegen Ende betrachtet.

- Ist ein Element aus T_i mit mehreren Elementen aus T verknüpfbar, so wird das Element aus T ausgewählt, bei dessen Verknüpfungsergebnis mit dem Element aus T_i möglichst wenig don't-care-positions überdeckt werden.
- Unter allen Elementen aus T, die mit dem Element aus T_i verknüpfbar sind, wird das Element ausgewählt, das die wenigsten don't-care-positions enthält.

Die Vorsortierung der Elemente aus T_i kann mit einer der anderen beiden Vorgehensweisen kombiniert werden. In der Praxis konnten durch diese Vorsortierung keine besseren Ergebnisse erzielt werden, so daß sich eine zusätzliche Vorsortierung als unnötig erweist.

2.4 Praktische Resultate

Die Berechnung der funktionalen Delaytestmenge wurde für zwei verschiedene Gruppen von Benchmarkschaltkreisen durchgeführt und ausgewertet. In Tabelle 2.1 sind die kombinatorischen Teile der ISCAS89-Benchmarks (siehe [BBK89]) bis zum Schaltkreis $cs1494$ aufgeführt; Tabelle 2.2 enthält die Ergebnisse für synthetisierte multi-level-Implementationen[2] der zweistufigen MCNC-Benchmarks (siehe [Yan91]).

Da unsere Testmengen nur für unate Funktionen gültig sind, wurden hier unate Versionen dieser Schaltkreise verwendet. Die gewählte Vorgehensweise zur Transformation einer beliebigen Schaltung in einen unate Schaltkreis ist in Kapitel 5 beschrieben. Die erste Spalte der Tabellen enthält jeweils den Namen der betrachteten Schaltung, danach wird die Anzahl der zugehörigen logischen Pfade angegeben.
Man kann zeigen, daß es für den Laufzeittest vieler Schaltkreise hinreichend ist, nur eine Teilmenge aller logischen Pfade zu überprüfen [LSBSV93, CC93, SLCR95]. Spalte 3 gibt die Anzahl der logischen Pfade einer möglichst kleinen, für den Laufzeittest hinreichenden Teilmenge von Pfaden an. Sie wurde mit der Heuristik aus [SLCR95] berechnet.
In Spalte 4 ist zum Vergleich die Anzahl der durch den Testmustergenerator aus [RLP87, Red92] testbaren Pfade aufgeführt.[3] Dieses Programm beschränkt sich auf den *robusten Einzelpfadtest* (siehe Def. 1.19). Es fällt auf, daß bei vielen Schaltkreisen eine Differenz zwischen den Spalten 3 und 4 besteht, d.h. zwischen zu testenden und den wirklich getesteten Pfaden. Somit wird das Laufzeitverhalten dieser Schaltkreise durch den Einzelpfadtest **nicht vollständig überprüft** (siehe hierzu auch Beispiel 1.7 aus Kapitel 1).
In Spalte 5 ist die Größe der Testmengen angegeben, die mit diesem komprimierenden Testmustergenerator berechnet wurden. Bei der Vorgehensweise des Testmustergenerators wird versucht, eine möglichst kleine Testmenge zu finden [Red92].

[2]Jeder Schaltkreis wurde durch Anwendung von *skript.rugged* innerhalb des Synthesetools SIS (siehe [SSM+92]) erzeugt.

[3]Hierbei wurde ein Backtracklimit von 100 verwandt. Taucht in Spalte 4 ein „?“ auf, so konnte der Testmustergenerator den entsprechenden Schaltkreis auf Grund der zu hohen Pfadanzahl nicht mehr bearbeiten.

Name	# log. Pfade	# hinreichend	# testbar	# Tests(Path Delay)	#FDT
cs27	56	50	50	22	30
cs298	462	420	418	89	252
cs344	710	682	676	169	5596
cs349	730	686	680	176	5596
cs382	800	797	797	154	1180
cs400	896	837	823	168	1174
cs444	1070	976	959	222	1174
cs510	738	738	738	245	1256
cs526	820	778	776	167	620
cs641	3488	2711	2703	436	10178
cs713	43624	6070	5654	808	10926
cs820	984	984	984	287	1790
cs832	1012	1012	1012	312	1770
cs953	2312	2312	2312	389	4130
cs1196	6196	5316	5079	1385	36648
cs1238	7118	5028	4514	1168	31878
cs1423	89452	63543	62431	12154	?
cs1488	1924	1924	1924	509	3144
cs1494	1952	1952	1952	497	3062

Tabelle 2.1: FDT für ISCAS89-Benchmarks

Spalte 6 stellt die Anzahl der Testvektoren unserer funktionalen Delaytestmenge für die angegebenen Schaltkreise dar. Hierbei werden auch die Initialvektoren zu den Testeingaben hinzugerechnet. Zur Kompaktierung wurde die in Paragraph 2.3 beschriebene Heuristik verwandt. Ist bei der FDT-Größe ein „?“ angegeben, so bedeutet dies, daß die Primimplikantenerzeugung unter SIS auf Grund von Speichermangel nicht durchführbar war. Hierbei macht sich der Effekt bemerkbar, daß die Anzahl der Primimplikanten zu einer Funktion exponentiell sein kann.
Vergleicht man die Größe der Testmengen des Testmustergenerators mit der Größe unserer funktionalen Delay-Testmenge, bemerkt man, daß unsere Testmenge fast immer größer ist (die einzigen Ausnahmen sind die Schaltkreise bw, $e64$ und $misex1$). Dies ist nicht verwunderlich, da unsere Testmenge im Gegensatz zu der vom Testmustergenerator berechneten Testmenge ein vollständiger Delaytest für den jeweils betrachteten Schaltkreis ist (beispielsweise bei Schaltkreis $sao2$, bei dem nur 80,2 % der zu testenden Pfade durch den robusten Einzelpfadtest testbar sind, ist unsere Testmenge um 31,2 % größer). Doch auch bei Schaltkreisen, die durch den robusten Einzelpfadtest vollständig getestet werden, ist unsere Testmenge wesentlich größer (wie z.B. bei Schaltkreis $cs953$). Dies ist der Preis, der für die Universalität der FDT bezahlt werden muß. Im nächsten Kapitel wird deshalb unsere Testmenge auf eine gegebene Realisierung der Schaltung zugeschnitten, wodurch

Name	# log. Pfade	# hinreichend	# testbar	#Tests(Path Delay)	#FDT
Z5xp1	20102	3230	2127	1026	1084
Z9sym	852	852	852	669	2132
alu4	26694	12988	10287	5615	94120
apex1	13760	13385	13298	2437	40970
apex2	2860	2512	2432	800	9600
apex3	35270	12056	11248	3101	16548
apex5	23848	11123	10516	1291	9990
b12	402	326	315	131	186
bw	24380	5610	3083	984	702
c5xp1	4832	1850	1429	839	918
c9sym	1222	1134	1117	847	4788
clip	2694	1798	1563	1022	3348
con1	44	44	44	22	52
cps	12608	12267	12227	1865	16414
duke2	5972	4734	4551	901	9446
e64	4290	4290	4290	135	132
ex4	2060	2052	2052	380	1468
ex5	8318	6829	6598	970	3294
inc	2884	1534	1257	689	786
misex1	1286	759	654	252	186
misex2	494	443	431	108	238
misex3	40578	21039	17433	6459	31944
misex3c	1856452	53984	?	?	72542
pdc	3544	3173	3121	673	8144
rd53	310	288	281	243	490
rd73	4116	1971	1690	1225	5148
rd84	3994	2933	2679	1659	11266
sao2	5076	2683	2151	1122	1472
spla	6274	5180	5086	1133	6656
squar5	930	384	143	175	216
t481	128	128	128	94	13938
table5	12694	11234	10813	2700	81146
vg2	914	893	893	358	2426

Tabelle 2.2: FDT für MCNC-Benchmarks

die Größe der Testmenge reduziert werden kann.

Um einen Eindruck zu geben, wie weit die Testanzahl durch Kompaktierung mit der in Kapitel 2.3 vorgestellten Heuristik verringert werden kann, enthalten die Tabellen 2.3 und

Name	# no_compact	# compact	% Verringerung	Zeit (sec)
cs27	56	30	46,4	1
cs298	584	252	56,9	1
cs344	6090	5596	8,1	20
cs349	6090	5596	8,1	20
cs382	1856	1180	36,4	2
cs400	1868	1174	37,2	2
cs444	1868	1174	37,2	2
cs510	1618	1256	22,4	4
cs526	1152	620	46,2	1
cs641	12816	10178	20,6	54
cs713	12784	10926	14,5	3:42
cs820	2616	1790	31,6	4
cs832	2604	1770	32,0	4
cs953	6770	4130	39,0	22
cs1196	47362	36648	22,6	19:13
cs1238	39176	31878	18,6	14:51
cs1488	4636	3144	32,2	17
cs1494	4540	3062	32,6	16

Tabelle 2.3: Kompaktierungsheuristik für ISCAS89-Benchmarks

2.4 hierzu experimentelle Resultate.

In der ersten Spalte wird wiederum der Name der betrachteten Schaltung angegeben. Spalte 2 (# no_compact) enthält die Summe der Größen der FDTs der single-output-Funktionen. In Spalte 3 (# compact) sind die FDT-Größen nach Kompaktieren durch die Heuristik angeführt. Um die Güte dieser Kompaktierung darzustellen, zeigt Spalte 4 den Prozentsatz auf, um den die Anzahl der Testtupel verringert werden konnte. Dieser Prozentsatz berechnet sich wie folgt:

$$\frac{\#\text{no\_compact} - \#\text{compact}}{\#\text{no\_compact}} * 100\%$$

Die Tabelle wird durch eine Laufzeitmessung (in Sekunden) abgeschlossen, die auf einer SUN SparcStation 10 durchgeführt wurde.

Betrachtet man die Tabellen näher, so fällt auf, daß die Ersparnis bei der Testmenge großen Schwankungen unterliegt. Als Beispiel für eine „gute“ Kompaktierung sei hier der Schaltkreis e64 angeführt, bei dem eine Ersparnis von 97% erzielt wird. Bei diesem Beispiel konnte eine maximale Kompaktierung erreicht werden:

$$\#\text{FDT}(f_{1,\ldots,n}) = \max\{\#\text{FDT}(f_i) | i = 1, \ldots, n\}$$

Name	# no_compact	# compact	% Verringerung	Zeit (sec)
Z5xp1	1384	1084	21,7	1
alu4	101526	94120	7,3	51:24
apex1	57688	40970	29,0	2:03:29
apex2	10208	9600	6,0	1:04
apex3	20978	16548	21,1	21:46
apex5	19496	9990	48,8	2:27
b12	300	186	38,0	1
bw	1660	702	57,7	1
c5xp1	1204	918	23,8	1
clip	3560	3348	6,0	4
con1	72	52	27,8	1
cps	24046	16414	31,7	8:07
duke2	15770	9446	40,1	2:38
e64	4420	132	97,0	3
ex4	5110	1468	71,3	32
ex5	7432	3294	55,7	8
inc	1044	786	24,7	1
misex1	438	186	57,5	1
misex2	692	238	65,6	1
misex3	39706	31944	19,5	7:25
misex3c	76286	72542	4,9	32:40
pdc	11766	8144	30,8	1:10
rd53	506	490	3,1	1
rd73	5624	5148	8,5	8
rd84	11630	11266	3,1	35
sao2	1626	1472	10,0	1
spla	11128	6656	40,2	45
squar5	288	216	25,0	1
table5	112352	81146	27,8	1:44:11
vg2	4638	2426	47,7	20

Tabelle 2.4: Kompaktierungsheuristik für MCNC-Benchmarks

Beispiele für eine „schlechte“ Kompaktierungsrate sind z.B. cs344 oder rd53. Dies kann unterschiedliche Gründe haben. Zum einen ist es möglich, daß die einzelnen Testmengen der single-output-Funktionen zwar ungefähr gleiche Größe haben, aber schlecht kompatibel zueinander sind (z.B. rd53). Zum anderen kann es auch sein, daß eine Menge im Vergleich zu den anderen sehr groß ist und somit eine Kompaktierung mit den anderen kleineren Mengen kaum ins Gewicht fällt (z.B. cs344).

Kapitel 3

Realisierungsabhängige Testmengenreduktion

Im vorigen Kapitel wurde die funktionale Delaytestmenge (FDT) eingeführt. Sie stellt einen vollständigen Verzögerungstest für *alle* Realisierungen einer logischen Funktion f als unate Schaltkreis dar. Es wurde gezeigt, daß die Konstruktion der funktionalen Delaytestmenge auf die Primimplikantenerzeugung für die negierte und nicht-negierte Funktion zurückgeführt werden kann. Wie man in den praktischen Resultaten des letzten Kapitels gesehen hat, muß die Realisierungsunabhängigkeit der FDT allerdings durch den Nachteil bezahlt werden, daß sich die funktionale Testmenge gegenüber dem robusten Einzelpfadtest stark vergrößern kann. Es stellt sich die Frage, ob sich die Größe der Testmenge bei gegebenem Design der Schaltung reduzieren läßt, ohne die wesentliche Eigenschaft des vollständigen Verzögerungstests zu verlieren.

Zur Lösung dieses Problems gehen wir wie folgt vor:
Gegeben sei die funktionale Testmenge zu einer Funktion f und eine Realisierung von f als unate Schaltkreis $\mathcal{K}$. Zuerst wird hergeleitet, welche Pfadsysteme aus $\mathcal{K}$ bezüglich einer Laufzeitschranke τ zu testen sind, um sicherzustellen, daß das worst-case-Delay einer Eingabe e kleiner oder gleich dieser Laufzeitschranke ist. Es ist bekannt, daß es für einen vollständigen Delaytest des Schaltkreises ausreicht, nur das Delay der Eingaben aus $\mathrm{E}(f)$ zu überprüfen (vgl. Lemma 2.11). Folglich reicht auch ein Test aller Pfadsysteme aus den Pfadsystemmengen dieser Eingaben für den vollständigen Delaytest des Schaltkreises aus. Will man die FDT reduzieren, muß man eine Teilmenge dieser Testmenge finden, die gerade alle notwendigen Pfadsysteme des Schaltkreises testet. Hierzu muß für jede Testsequenz der funktionalen Delaytestmenge bestimmt werden, welche Pfadsysteme sie überprüft. Da die gesuchte Teilmenge der FDT alle wesentlichen Pfadsysteme des Schaltkreises überdecken (testen) soll, wird unser Testmengenminimierungsproblem somit auf ein Überdeckungsproblem zurückgeführt.

3.1 Minimums-Pfadsysteme

In Paragraph 1.4.2 wurde gezeigt, daß es Schaltkreise gibt, die mit dem Pfaddelaytest nicht vollständig auf Laufzeitfehler überprüft werden können. Ein Weg aus diesem Dilemma ist, statt Einzelpfaden auch den Test von *Pfadsystemen* zuzulassen. Wir werden hier eine spezielle Form eines Pfadsystems verwenden (siehe auch Definition 1.21), bei der sich das Delay des Pfadsytems als das Minimum der Delays der im System enthaltenen Pfade ergibt.

Notation:
Sei L eine Menge von logischen Pfaden des Schaltkreises $\mathcal{K}$, dann ist:

$$d_{min}(L) := min\{d_{\mathcal{K}}(p,t) \,|\, (p,t) \in L\}$$

Definition 3.1 :
Sei $\mathcal{K}$ ein kombinatorischer Schaltkreis und PS ein Pfadsystem aus $\mathcal{K}$.

(a) PS heißt Minimums-Pfadsystem, *falls*

$$d_{\mathcal{K}}(PS) = d_{min}(PS)$$

(b) Die Menge aller Minimums-Pfadsysteme von $\mathcal{K}$ bezeichnen wir mit $\mathcal{MPS}(\mathcal{K})$.

Im Rest des Kapitels werden Pfadsysteme in das Klammerpaar „[,]“ eingeschlossen. Das folgende Beispiel veranschaulicht die Begriffe aus Defintion 3.1:

Beispiel 3.1 :
Wir betrachten den Schaltkreis $\mathcal{K}$ aus Abbildung 3.1. Die Pfade in $\mathcal{K}$ seien wie folgt bestimmt:

$$p_1 = l_1\text{-}l_4\text{-}l_5 \qquad p_2 = l_2\text{-}l_4\text{-}l_5 \qquad p_3 = l_3\text{-}l_5$$

Das Pfadsystem

$$PS_1 = [(p_1, 0 \to 1), (p_2, 0 \to 1), (p_3, 0 \to 1)]$$

ist kein Minimums-Pfadsystem, da gilt:

$$d_{\mathcal{K}}(PS_1) = \max\{\min\{d_{\mathcal{K}}(p_1, 0 \to 1), d_{\mathcal{K}}(p_2, 0 \to 1)\}, d_{\mathcal{K}}(p_3, 0 \to 1)\}$$

Das Pfadsystem

$$PS_2 = [(p_1, 0 \to 1), (p_2, 0 \to 1)]$$

ist ein Minimums-Pfadsystem, da gilt:

$$d_{\mathcal{K}}(PS_2) = \min\{d_{\mathcal{K}}(p_1, 0 \to 1), d_{\mathcal{K}}(p_2, 0 \to 1)\} = d_{min}(PS_2)$$

■

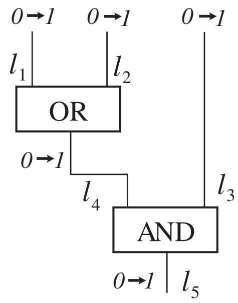

Abbildung 3.1: Beispielschaltkreis $\mathcal{K}$

Bemerkung 3.1:
Sei PS ein Minimums-Pfadsystem und z eine Zelle in PS. $l_1, \ldots, l_k$ seien die Inputs von z, die in PS liegen. Sind die Endwerte der Transitionen auf diesen Leitungen nicht-kontrollierend bezüglich z, so laufen alle durch z führenden Pfade aus PS durch dieselbe Inputleitung von z (d.h. die Anzahl der Inputs von z, die in PS liegen, ist 1). Andernfalls ist in der Delayformel von PS eine „Maximumsbildung" enthalten, wodurch PS kein Minimums-Pfadsystem wäre. In Beispiel 3.1 sind die Endwerte der Transitionen des Pfadsytems PS_1 auf den (unterschiedlichen) Leitungen l_3 und l_4 nicht-kontrollierend bezüglich des AND-Gatters. Folglich ist PS_1 kein Minimums-Pfadsystem.

3.2 Bestimmung der zu testenden Pfadsysteme

Gegeben sei ein unate Schaltkreis $\mathcal{K}$. Zunächst wird hergeleitet, welche Pfadsysteme für eine Eingabe e an $\mathcal{K}$ zu testen sind, um zu zeigen, daß $d_{\mathcal{K}}(e) \leq \tau$. Mit Hilfe dieser Pfadsystemmengen der Eingaben an $\mathcal{K}$ wird anschließend die Menge der Pfadsysteme des Schaltkreises $\mathcal{K}$ bestimmt, deren Test für den vollständigen Delaytest von $\mathcal{K}$ ausreicht.

3.2.1 Zu testende Pfadsysteme für eine Eingabe e

Im folgenden wir bestimmt, welche Pfadsysteme getestet werden müssen, um für eine Eingabe e an $\mathcal{K}$ zu garantieren, daß $d_{\mathcal{K}}(e) \leq \tau$.

Satz 3.2:
Sei $\mathcal{K}$ ein unate Schaltkreis mit genau einem primären Ausgang und e ein Eingabevektor an $\mathcal{K}$. Dann existiert eine Menge von Minimums-Pfadsystemen $\mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ mit:

$$d_{\mathcal{K}}(e) = \max\{d_{\mathcal{K}}(PS) \mid PS \in \mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e)\}$$

Beispiel 3.2 :
Sei ein Schaltkreis $\mathcal{K}$ wie in Abbildung 3.2 gegeben. Die dicker gezeichneten Pfade seien wie folgt bestimmt:

$$p_1 = l_3\text{-}l_6\text{-}l_8\text{-}l_{10}\text{-}l_{14}\text{-}l_{17} \qquad p_2 = l_3\text{-}l_6\text{-}l_8\text{-}l_{11}\text{-}l_{15}\text{-}l_{17}$$
$$p_3 = l_3\text{-}l_7\text{-}l_9\text{-}l_{12}\text{-}l_{15}\text{-}l_{17} \qquad p_4 = l_3\text{-}l_7\text{-}l_9\text{-}l_{13}\text{-}l_{16}\text{-}l_{17}$$

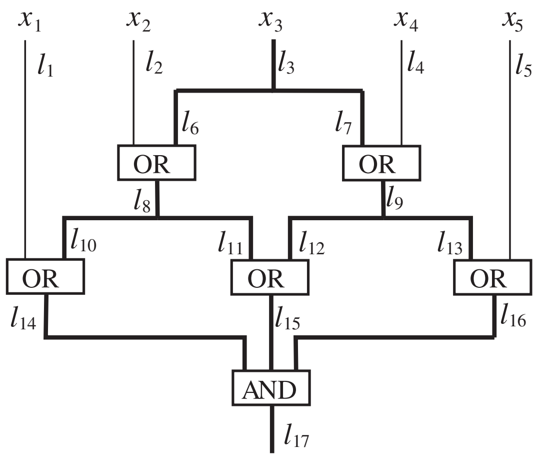

Abbildung 3.2: Beispielschaltkreis $\mathcal{K}$

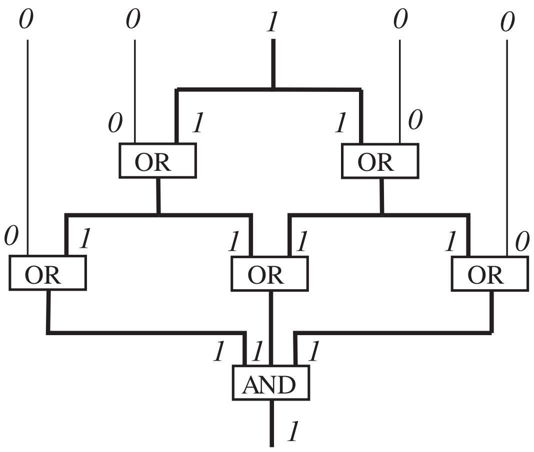

Abbildung 3.3: Belegung für e an $\mathcal{K}$

Es wird die Eingabe $e = 00100$ an $\mathcal{K}$ angelegt. Die Belegungen aller Leitungen von $\mathcal{K}$ bezüglich e sind in Abbildung 3.3 angegeben.

Der Ausgabewert des Schaltkreises ist 1 bezüglich der Eingabe e. Damit dieser Ausgabewert berechnet werden kann, müssen die Leitungen l_{14}, l_{15} und l_{16} den Wert 1 angenommen haben. Leitung l_{14} (l_{16}) kann bezüglich der Eingabe e nur über den Pfad p_1 (p_4) auf 1 gesetzt werden. Leitung l_{15} kann entweder über den Pfad p_2 oder über den Pfad p_3 auf 1 gesetzt werden. Also erhalten wir:

$$d_{\mathcal{K}}(e) = \max\{d_{\mathcal{K}}(p_1, 0 \to 1), \min\{d_{\mathcal{K}}(p_2, 0 \to 1), d_{\mathcal{K}}(p_3, 0 \to 1)\}, d_{\mathcal{K}}(p_4, 0 \to 1)\}$$

und somit:

$$\mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e) = \{[(p_1, 0 \to 1)]\ ,\ [(p_2, 0 \to 1), (p_3, 0 \to 1)]\ ,\ [(p_4, 0 \to 1)]\}$$

■

Für den Beweis von Satz 3.2 ist es notwendig, noch das folgende Lemma einzuführen, mit dessen Hilfe man eine min{...} Formel in eine max{...} Formel umwandeln kann. Hierdurch kann im Beweis unsere Delay-Formel in die gewünschte „Maximum-Form" gebracht werden.

Lemma 3.3 :
Seien $M_1, \ldots, M_n$ Teilmengen einer vollständig geordneten Menge M. Dann gilt:

$$\min\{\max M_1, \ldots, \max M_n\} \;=\; \max\{\min\{m_1, \ldots, m_n\} \,|\, m_i \in M_i, \forall\, i \in \{1, \ldots, n\}\}$$

Beweis:
(a) $\min\{\max M_1, \ldots, \max M_n\} \;\leq\; \max\{\min\{m_1, \ldots, m_n\} \,|\, m_i \in M_i, \forall\, i \in \{1, \ldots, n\}\}$:
Diese Ungleichung folgt direkt aus der Tatsache, daß die Menge $\{\max M_1, \ldots, \max M_n\}$ eine der Mengen $\{m_1, \ldots, m_n\}$, $m_i \in M_i \;\forall\, i = 1, \ldots, n$ ist, die auf der rechten Seite der Ungleichung betrachtet werden.

(b) $\min\{\max M_1, \ldots, \max M_n\} \;\geq\; \max\{\min\{m_1, \ldots, m_n\} \,|\, m_i \in M_i, \forall\, i \in \{1, \ldots, n\}\}$:
Sei $j \in \{1, \ldots, n\}$ so daß $\max M_j = \min\{\max M_1, \ldots, \max M_n\}$.
Trivialerweise gilt:

$$\max M_j \geq \min\{m_1, \ldots, m_n\},\ \forall \{m_1, \ldots, m_n\},\ m_i \in M_i$$

■

In dem nun folgenden Beweis des Satzes 3.2 wird eine steigende Flanke ($0 \rightarrow 1$) mit $\uparrow$, eine fallende Flanke ($1 \rightarrow 0$) mit $\downarrow$ abgekürzt.

Beweis [zu Satz 3.2]:
Der Beweis erfolgt durch vollständige Induktion über die Anzahl der Gatter des Schaltkreises $\mathcal{K}$.

INDUKTIONSANFANG: $\mathcal{K}$ enthält keine Gatter, d.h. $\mathcal{K}$ enthält genau eine Leitung l.
$\mathcal{MIN}$-$\mathcal{PS}_{\mathcal{K}}(e)$ enthält genau ein Pfadsystem mit dem logischen Pfad $(l, \overline{e} \rightarrow e)$, d.h.

$$\mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e) = \{[(l, \overline{e} \rightarrow e)]\}$$

INDUKTIONSSCHRITT: $\mathcal{K}$ enthält $\alpha > 0$ Gatter.
Sei l die Leitung, die mit dem primären Ausgang von $\mathcal{K}$ verbunden ist. Sei g das Gatter, dessen Ausgang mit der Leitung l verbunden ist.
Sei o.B.d.A. g ein AND-Gatter mit den zwei Input-Leitungen l_1 und l_2 (für die anderen Fälle analog). Die Teilschaltkreise, die die Leitungen l_1 und l_2 als Ausgänge haben, werden mit $\mathcal{K}_1$ und $\mathcal{K}_2$ bezeichnet.
$d_{\mathcal{K}_1}(e)$ und $d_{\mathcal{K}_2}(e)$ bezeichnen das (worst-case) Delay der Schaltkreise $\mathcal{K}_1$ und $\mathcal{K}_2$ bezüglich der Eingabe e. Die beschriebene Situation ist in Abb. 3.4 dargestellt.

Da die Gatteranzahlen dieser Schaltkreise kleiner als α sind, kann induktiv angenommen werden, daß Satz 3.2 für diese Teilschaltungen erfüllt ist, d.h. es existieren Pfadsystemmengen $\mathcal{MIN}$-$\mathcal{PS}_{\mathcal{K}_1}(e)$ und $\mathcal{MIN}$-$\mathcal{PS}_{\mathcal{K}_2}(e)$ mit:

$$\begin{aligned} d_{\mathcal{K}_1}(e) &= \max\{d_{\mathcal{K}_1}(PS) \,|\, PS \in \mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}_1}(e)\} \;\; \textit{und} \\ d_{\mathcal{K}_2}(e) &= \max\{d_{\mathcal{K}_2}(PS) \,|\, PS \in \mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}_2}(e)\}. \end{aligned} \tag{3.1}$$

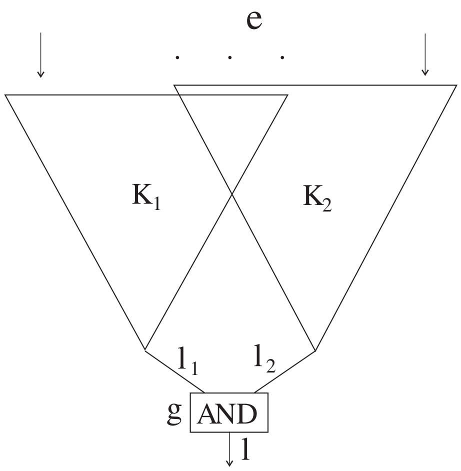

Abbildung 3.4: Struktur des Schaltkreises $\mathcal{K}$

Sei L eine Menge von logischen Pfaden, die an einem Input des Gatters g enden.

$$L \circ g := \{(p\text{-}l, t) |\, (p, t) \in L\}$$

bezeichnet die Pfade, die durch Ausdehnen der Pfade aus L bis zum Output l von g entstehen.
Weiterhin seien definiert:

$$\begin{aligned} \mathcal{MIN}\text{-}\mathcal{PS}_1(e) &:= \{PS \circ g |\, PS \in \mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}_1}(e)\} \\ \mathcal{MIN}\text{-}\mathcal{PS}_2(e) &:= \{PS \circ g |\, PS \in \mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}_2}(e)\}. \end{aligned} \tag{3.2}$$

In Abhängigkeit von den Inputwerten des Gatters g bezüglich der Eingabe e sind 3 Fälle zu unterscheiden:

<u>FALL 1:</u> Ein Inputwert an g ist kontrollierend (logisch Null), der andere nicht-kontrollierend (logisch Eins).
o.B.d.A. Sei der Wert an l_1 kontrollierend bezüglich e.
Für das Delay der Eingabe gilt somit[1]:

$$d_{\mathcal{K}}(e) = d_{\mathcal{K}_1}(e) + d_{\downarrow,l_1}(g) \tag{3.3}$$

Aus den Formeln (3.1), (3.2) und (3.3) folgt:

$$\begin{aligned} d_{\mathcal{K}}(e) &= \max\{d_{\mathcal{K}_1}(PS) |\, PS \in \mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}_1}(e)\} + d_{\downarrow,l_1}(g) \\ &= \max\{d_{\mathcal{K}_1}(PS) + d_{\downarrow,l_1}(g) |\, PS \in \mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}_1}(e)\} \\ &= \max\{d_{\mathcal{K}}(PS') |\, PS' \in \mathcal{MIN}\text{-}\mathcal{PS}_1(e)\} \end{aligned}$$

[1] Diese Delay-Formel gilt für unate Schaltkreise (siehe [SR94]), in allgemeinen Schaltkreisen muß diese Beziehung nicht gelten. Für Fall 2 und 3 das Beweises gilt dies analog.

$\Longrightarrow \mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e) := \mathcal{MIN}\text{-}\mathcal{PS}_1(e)$

<u>FALL 2:</u> Beide Inputwerte an g sind nicht-kontrollierend.
Es gilt:

$$d_{\mathcal{K}}(e) = \max\{d_{\mathcal{K}_1}(e) + d_{\uparrow,l_1}(g)\ ,\ d_{\mathcal{K}_2}(e) + d_{\uparrow,l_2}(g)\} \tag{3.4}$$

Aus den Formeln (3.1), (3.2) und (3.4) folgt:

$$\begin{aligned} d_{\mathcal{K}}(e) &= \max\{\max\{d_{\mathcal{K}_1}(PS) \mid PS \in \mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}_1}(e)\} + d_{\uparrow,l_1}(g)\ , \\ &\qquad \max\{d_{\mathcal{K}_2}(PS) \mid PS \in \mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}_2}(e)\} + d_{\uparrow,l_2}(g)\ \} \\ &= \max\{\max\{d_{\mathcal{K}}(PS') \mid PS' \in \mathcal{MIN}\text{-}\mathcal{PS}_1(e)\}\ , \\ &\qquad \max\{d_{\mathcal{K}}(PS') \mid PS' \in \mathcal{MIN}\text{-}\mathcal{PS}_2(e)\}\} \\ &= \max\{d_{\mathcal{K}}(PS') \mid PS' \in\ \mathcal{MIN}\text{-}\mathcal{PS}_1(e) \cup \mathcal{MIN}\text{-}\mathcal{PS}_2(e)\} \end{aligned}$$

$\Longrightarrow \mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e) := \mathcal{MIN}\text{-}\mathcal{PS}_1(e) \cup \mathcal{MIN}\text{-}\mathcal{PS}_2(e)$

<u>FALL 3:</u> Beide Inputwerte an g sind kontrollierend.
Es gilt:

$$d_{\mathcal{K}}(e) = \min\{d_{\mathcal{K}_1}(e) + d_{\downarrow,l_1}(g)\ ,\ d_{\mathcal{K}_2}(e) + d_{\downarrow,l_2}(g)\} \tag{3.5}$$

Aus den Formeln (3.1), (3.2) und (3.5) folgt:

$$\begin{aligned} d_{\mathcal{K}}(e) &= \min\{\max\{d_{\mathcal{K}_1}(PS_1) \mid PS_1 \in \mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}_1}(e)\} + d_{\downarrow,l_1}(g)\ , \\ &\qquad \max\{d_{\mathcal{K}_2}(PS_2) \mid PS_2 \in \mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}_2}(e)\} + d_{\downarrow,l_2}(g)\ \} \\ &= \min\{\max\{d_{\mathcal{K}}(PS_1') \mid PS_1' \in \mathcal{MIN}\text{-}\mathcal{PS}_1(e)\}\ , \\ &\qquad \max\{d_{\mathcal{K}}(PS_2') \mid PS_2' \in \mathcal{MIN}\text{-}\mathcal{PS}_2(e)\}\} \end{aligned}$$

Nach Lemma 3.3 gilt für beliebige Teilmengen M_1 und M_2 einer vollständig geordneten Menge M:

$$\min\{\max M_1\ ,\ \max M_2\} = \max\{\min\{m_1, m_2\} \mid m_i \in M_i, i \in \{1,2\}\}$$

Hiermit folgt für die Delay-Formel von $d_{\mathcal{K}}(e)$:

$$d_{\mathcal{K}}(e) = \max\{\min\{d_{\mathcal{K}}(PS_1'), d_{\mathcal{K}}(PS_2')\} \mid PS_1' \in \mathcal{MIN}\text{-}\mathcal{PS}_1(e),\ PS_2' \in \mathcal{MIN}\text{-}\mathcal{PS}_2(e)\}$$

Nach Voraussetzung sind alle Pfadsysteme aus $\mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}_1}(e)$ und $\mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}_2}(e)$ Minimums-Pfadsysteme. Da die Endwerte der Transitionen auf den Inputleitungen von g für alle Pfade aus diesen Pfadsystemmengen kontrollierend bezüglich des Gatters g sind (siehe Voraussetzung von Fall 3), ist auch die Vereinigung von einem Minimums-Pfadsystem aus $\mathcal{MIN}\text{-}\mathcal{PS}_1(e)$ mit einem Minimums-Pfadsystem aus $\mathcal{MIN}\text{-}\mathcal{PS}_2(e)$ wieder ein Minimums-Pfadsystem.

$\Longrightarrow \mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e) := \{PS_1' \cup PS_2' \mid PS_1' \in \mathcal{MIN}\text{-}\mathcal{PS}_1(e),\ PS_2' \in \mathcal{MIN}\text{-}\mathcal{PS}_2(e)\}$

Insgesamt folgt:

$$d_{\mathcal{K}}(e) = \max\{d_{\mathcal{K}}(PS') \mid PS' \in \mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e)\}$$

■

Mit Hilfe von Satz 3.2 können wir nun angeben, welche Pfadsysteme zu testen sind, um für eine Eingabe e zu zeigen, daß $d_{\mathcal{K}}(e) \leq \tau$.

Korollar 3.4:
Sei $\mathcal{K}$ ein unate Schaltkreis mit einem primären Ausgang, e eine Eingabe an $\mathcal{K}$ und $\mathcal{MIN}$-$\mathcal{PS}_{\mathcal{K}}(e)$ wie in Satz 3.2 gegeben, dann gilt:

$$d_{\mathcal{K}}(PS) \leq \tau \; \forall \, PS \in \mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e) \implies d_{\mathcal{K}}(e) \leq \tau$$

3.2.2 Zu testende Pfadsysteme für den Gesamtschaltkreis

Aus Lemma 2.11 wissen wir, daß es für einen vollständigen Laufzeittest ausreicht, nur das Delay der essentiellen Eingaben aus $\mathrm{E}(f)$ zu überprüfen. Nach Korollar 3.4 kann das Delay dieser Eingaben überprüft werden, indem man das Delay der Pfadsysteme aus den zugehörigen Pfadsystemmengen testet.
Hat man also das Delay der Pfadsystemmengen aller Eingaben aus $\mathrm{E}(f)$ überprüft, kann man schlußfolgern, daß der Schaltkreis rechtzeitig schaltet.

Somit kann die Menge der zu testenden Pfadsysteme des Schaltkreises bestimmt werden.

Satz 3.5:
Sei $\mathcal{K}$ ein unate Schaltkreis mit einem primären Ausgang und f die von $\mathcal{K}$ berechnete boolsche Funktion. Wir definieren

$$\mathcal{MIN}\text{-}\mathcal{PS}(\mathcal{K}) := \bigcup_{e \in \mathrm{E}(f)} \mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e).$$

Dann gilt:

$$d_{\mathcal{K}}(PS) \leq \tau \; \forall \, PS \in \mathcal{MIN}\text{-}\mathcal{PS}(\mathcal{K}) \implies d(\mathcal{K}) \leq \tau$$

Beweis:
Folgt direkt aus Lemma 2.11 und Korollar 3.4. ■

Es wurde eine Pfadsystemmenge gefunden, deren Überprüfung einen vollständigen Laufzeittest für den betrachteten Schaltkreis garantiert. Die Größe dieser Menge ist allerdings nicht optimal, da Pfadsysteme enthalten sind, die durch andere (kleinere) Pfadsysteme mitgetestet werden und somit „unnötig“ sind. Durch einen Vergleich der einzelnen Systeme kann die Menge $\mathcal{MIN}$-$\mathcal{PS}(\mathcal{K})$ zu einer Menge $\mathcal{MIN}$-$\mathcal{PS}_{min}(\mathcal{K})$ zusammengestrichen werden, die nur noch aus *minimalen Pfadsystemen* besteht.

Satz 3.6:
Sei $\mathcal{K}$ ein Schaltkreis mit genau einem primären Ausgang und $\mathcal{MIN}\text{-}\mathcal{PS}(\mathcal{K})$ wie in Satz 3.5. Wir bezeichnen mit

$$\begin{aligned}\mathcal{MIN}\text{-}\mathcal{PS}_{min}(\mathcal{K}) \;:=\; & \{PS \in \mathcal{MIN}\text{-}\mathcal{PS}(\mathcal{K}) \mid \nexists\, PS' \in \mathcal{MIN}\text{-}\mathcal{PS}(\mathcal{K}) : \\ & (PS' \subseteq PS) \wedge (PS' \neq PS)\}\end{aligned}$$

die Menge der minimalen Minimums-Pfadsysteme *aus $\mathcal{MIN}\text{-}\mathcal{PS}(\mathcal{K})$.*
Dann gilt:

$$d_{\mathcal{K}}(PS) \leq \tau \;\forall\, PS \in \mathcal{MIN}\text{-}\mathcal{PS}_{min}(\mathcal{K}) \implies d(\mathcal{K}) \leq \tau$$

Beweis:
Seien PS und PS' Minimums-Pfadsysteme aus der Menge $\mathcal{MIN}\text{-}\mathcal{PS}(\mathcal{K})$, mit: $(PS \subseteq PS') \wedge (PS \neq PS')$.
Für das Delay eines Minimums-Pfadsystems aus $\mathcal{MIN}\text{-}\mathcal{PS}(\mathcal{K})$ gilt:

$$d_{\mathcal{K}}(PS) = \min\{d_{\mathcal{K}}(p,t) \mid (p,t) \in PS\}$$

Da PS' mehr Pfade als PS enthält und einer dieser zusätzlichen Pfade das Delay von PS' reduzieren kann, gilt für die Delays der Systeme:

$$d_{\mathcal{K}}(PS') \leq d_{\mathcal{K}}(PS)$$

Hiermit wird der Test von PS' durch den Test von PS unnötig, so daß folgt:

$$\begin{aligned}d_{\mathcal{K}}(PS) \leq \tau \;\forall\, PS \in \mathcal{MIN}\text{-}\mathcal{PS}_{min}(\mathcal{K}) \;&\implies\; d_{\mathcal{K}}(PS) \leq \tau \;\forall\, PS \in \mathcal{MIN}\text{-}\mathcal{PS}(\mathcal{K}) \\ &\overset{\text{Satz3.5}}{\implies}\; d(\mathcal{K}) \leq \tau\end{aligned}$$

■

Beispiel 3.3:
Wir betrachten den Schaltkreis $\mathcal{K}$ aus Abb. 3.5. Die Pfade des Schaltkreises seien wie folgt bestimmt:

$$\begin{array}{ll} p_1 = l_1\text{-}l_{14}\text{-}l_{17} & p_2 = l_2\text{-}l_8\text{-}l_{10}\text{-}l_{14}\text{-}l_{17} \\ p_3 = l_2\text{-}l_8\text{-}l_{11}\text{-}l_{15}\text{-}l_{17} & p_4 = l_3\text{-}l_6\text{-}l_8\text{-}l_{10}\text{-}l_{14}\text{-}l_{17} \\ p_5 = l_3\text{-}l_6\text{-}l_8\text{-}l_{11}\text{-}l_{15}\text{-}l_{17} & p_6 = l_3\text{-}l_7\text{-}l_9\text{-}l_{12}\text{-}l_{15}\text{-}l_{17} \\ p_7 = l_3\text{-}l_7\text{-}l_9\text{-}l_{13}\text{-}l_{16}\text{-}l_{17} & p_8 = l_4\text{-}l_9\text{-}l_{12}\text{-}l_{15}\text{-}l_{17} \\ p_9 = l_4\text{-}l_9\text{-}l_{13}\text{-}l_{16}\text{-}l_{17} & p_{10} = l_5\text{-}l_{16}\text{-}l_{17} \end{array}$$

Die Funktion des Schaltkreises ist:

$$f = x_3 \vee x_2 x_4 \vee x_1 x_4 \vee x_2 x_5$$

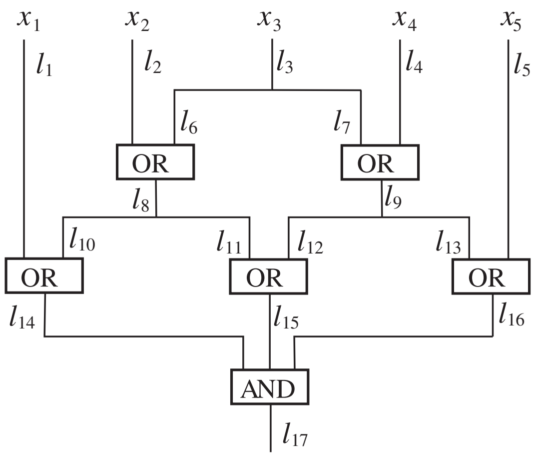

Abbildung 3.5: Beispielschaltkreis $\mathcal{K}$

Die Menge der essentiellen Eingabevektoren ergibt sich zu:

$$\begin{aligned} \mathrm{E}(f) &= \mathrm{MIN\text{-}1}(f) \cup \mathrm{MAX\text{-}0}(f) \\ &= \{00100, 01010, 10010, 01001\} \cup \{11000, 10001, 00011\} \\ &= \{00100, 01010, 10010, 01001, 11000, 10001, 00011\} \end{aligned}$$

Die Elemente dieser Menge seien in der obigen Reihenfolge mit $e_1, \ldots, e_7$ numeriert. Die Pfadsystemmengen zu diesen Eingaben e_1 (siehe auch Beispiel 3.2) bis e_7 sind:

$$\begin{aligned} \mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e_1) &= \{[(p_4,\uparrow)]\ ,\ [(p_5,\uparrow),(p_6,\uparrow)]\ ,\ [(p_7,\uparrow)]\} \\ \mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e_2) &= \{[(p_2,\uparrow)]\ ,\ [(p_3,\uparrow),(p_8,\uparrow)]\ ,\ [(p_9,\uparrow)]\} \\ \mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e_3) &= \{[(p_1,\uparrow)]\ ,\ [(p_8,\uparrow)]\ ,\ [(p_9,\uparrow)]\} \\ \mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e_4) &= \{[(p_2,\uparrow)]\ ,\ [(p_3,\uparrow)]\ ,\ [(p_{10},\uparrow)]\} \\ \mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e_5) &= \{[(p_7,\downarrow)]\ ,\ [(p_9,\downarrow)]\ ,\ [(p_{10},\downarrow)]\} \\ \mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e_6) &= \{[(p_3,\downarrow)]\ ,\ [(p_5,\downarrow)]\ ,\ [(p_6,\downarrow)]\ ,\ [(p_8,\downarrow)]\} \\ \mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e_7) &= \{[(p_1,\downarrow)]\ ,\ [(p_2,\downarrow)]\ ,\ [(p_4,\downarrow)]\} \end{aligned}$$

Die Menge der zu testenden Pfadsysteme $\mathcal{MIN}\text{-}\mathcal{PS}(\mathcal{K})$ ist somit:

$$\begin{aligned} \mathcal{MIN}\text{-}\mathcal{PS}(\mathcal{K}) = \{ & [(p_1,\uparrow)]\ ,\ [(p_2,\uparrow)]\ ,\ [(p_3,\uparrow)]\ ,\ [(p_4,\uparrow)]\ ,\ [(p_7,\uparrow)]\ ,\ [(p_8,\uparrow)]\ , \\ & [(p_9,\uparrow)]\ ,\ [(p_{10},\uparrow)]\ ,\ [(p_5,\uparrow),(p_6,\uparrow)]\ ,\ [(p_3,\uparrow),(p_8,\uparrow)]\ , \\ & [(p_1,\downarrow)]\ ,\ [(p_2,\downarrow)]\ ,\ [(p_3,\downarrow)]\ ,\ [(p_4,\downarrow)]\ ,\ [(p_5,\downarrow)]\ ,\ [(p_6,\downarrow)]\ , \\ & [(p_7,\downarrow)]\ ,\ [(p_8,\downarrow)]\ ,\ [(p_9,\downarrow)]\ ,\ [(p_{10},\downarrow)]\} \end{aligned}$$

Diese Menge ist nicht minimal, da gilt:

$$[(p_3,\uparrow)] \subset [(p_3,\uparrow),(p_8,\uparrow)]$$

Somit kann dieses Pfadsystem entfernt werden und es ergibt sich die Menge $\mathcal{MIN}\text{-}\mathcal{PS}_{min}(\mathcal{K})$:

$$\mathcal{MIN}\text{-}\mathcal{PS}_{min}(\mathcal{K}) = \mathcal{MIN}\text{-}\mathcal{PS}(\mathcal{K}) \setminus \{[(p_3, \uparrow), (p_8, \uparrow)]\}$$

■

3.3 Test der Pfadsysteme

In Kapitel 2, Lemma 2.12, haben wir gesehen, wie überprüft werden kann, ob das worst-case-Delay einer Eingabe e aus der Menge der essentiellen Eingabevektoren kleiner oder gleich einer Laufzeitschranke τ ist:

> Sei $\mathcal{K}$ ein unate Schaltkreis mit einem primären Ausgang und f die von $\mathcal{K}$ berechnete Funktion. $\forall\, e \in \mathrm{E}(f)$ existiert eine Eingabenfolge $< init(e), e >$ aus der funktionalen Delaytestmenge zu f, so daß gilt:
>
> > $\mathcal{K}$ funktioniert korrekt für die Eingabefolge $< init(e), e >$ bei Laufzeitschranke $\tau \implies d_{\mathcal{K}}(e) \leq \tau$.

Es kann geschlußfolgert werden, daß die Pfadsysteme aus $\mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ ebenfalls durch die Testfolge $< init(e), e >$ getestet werden können.

Korollar 3.7:
Sei $\mathcal{K}$ ein unate Schaltkreis mit einem primären Ausgang, f die von $\mathcal{K}$ berechnete Funktion und e eine Eingabe an $\mathcal{K}$ mit $e \in \mathrm{E}(f)$. Es gilt:

> *$\mathcal{K}$ funktioniert korrekt für die Eingabesequenz $< init(e), e >$ bei Laufzeitschranke τ*
> *$\implies d_{\mathcal{K}}(PS) \leq \tau \;\forall\, PS \in \mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$.*

Beweis:
Folgt direkt aus Lemma 2.12 und Satz 3.2. ■

Hiermit sind die Voraussetzungen gegeben, den zentralen Satz dieses Paragraphen zu formulieren. Er ermöglicht es, die funktionale Delay-Testmenge zu einer reduzierten Delay-Testmenge zusammenzustreichen.

Satz 3.8:
Sei $\mathcal{K}$ ein unate Schaltkreis mit einem primären Ausgang und f die von $\mathcal{K}$ berechnete Funktion. Sei $E \subseteq \mathrm{E}(f)$ eine Menge von Eingaben an $\mathcal{K}$ mit:

$$\bigcup_{e \in E} \mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e) \supseteq \mathcal{MIN}\text{-}\mathcal{PS}_{min}(\mathcal{K})$$

Die Eingabesequenzen aus $\{< init(e), e > \mid e \in E\}$ sind ein vollständiger Delaytest für den Schaltkreis $\mathcal{K}$.

Beweis:
Der Schaltkreis $\mathcal{K}$ funktioniert korrekt für alle Testsequenzen aus $\{< init(e), e > \mid e \in E\}$.

$$\stackrel{Korollar 3.7}{\Longrightarrow} \forall e \in E \text{ gilt:} \quad d_{\mathcal{K}}(PS) \leq \tau \; \forall \, PS \in \mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e).$$

Es gilt nach Voraussetzung:

$$\begin{aligned} & \mathcal{MIN}\text{-}\mathcal{PS}_{min}(\mathcal{K}) \subseteq \bigcup_{e \in E} \mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e) \\ & \Longrightarrow \quad d_{\mathcal{K}}(PS) \leq \tau \; \forall \, PS \in \mathcal{MIN}\text{-}\mathcal{PS}_{min}(\mathcal{K}) \\ & \stackrel{Satz 3.6}{\Longrightarrow} d(\mathcal{K}) \leq \tau \end{aligned}$$

■

Beispiel 3.4:
Seien die funktionale Delaytestmenge für einen Schaltkreis $\mathcal{K}$ und die Pfadsystemmenge $\mathcal{MIN}\text{-}\mathcal{PS}_{min}(\mathcal{K})$ wie in Beispiel 3.3 gegeben. Die Pfadsystemmenge der Eingabe e_2 ist:

$$\mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e_2) = \{[(p_2, \uparrow)] \;,\; [(p_3, \uparrow), (p_8, \uparrow)] \;,\; [(p_9, \uparrow)]\}$$

Wie schon gesehen, ist das Pfadsystem $[(p_3, \uparrow), (p_8, \uparrow)]$ nicht in $\mathcal{MIN}\text{-}\mathcal{PS}_{min}(\mathcal{K})$ enthalten und muß somit nicht getestet werden. Das Pfadsystem $[(p_2, \uparrow)]$ ist ebenfalls in der Pfadsystemmenge von e_4 enthalten, das System $[(p_9, \uparrow)]$ in der Menge von e_3. Alle Pfadsysteme der Pfadsystemmenge von e_2 sind somit unnötig oder werden ebenfalls von anderen Eingaben der funktionalen Delaytestmenge getestet. Die Eingabe e_2 ist für den vollständigen Delay-Test unnötig, d.h. die Testsequenz $< init(e_2), e_2 >$ kann für $\mathcal{K}$ aus der funktionalen Delaytestmenge gestrichen werden. ■

Eine minimale Teilmenge der funktionalen Delaytestmenge zu finden, die die Voraussetzungen aus Satz 3.8 erfüllt, führt unsere Testmengenminimierung auf ein Überdeckungsproblem zurück. Dies soll durch Abb. 3.6 veranschaulicht werden.

Die Zeilen der Matrix entsprechen den Eingaben $e_1, \ldots, e_k$ der Menge $\mathrm{E}(f)$, die Spalten entsprechen den Pfadsystemen $PS_1, \ldots, PS_n$ der Pfadsystemmenge $\mathcal{MIN}\text{-}\mathcal{PS}_{min}(\mathcal{K})$.

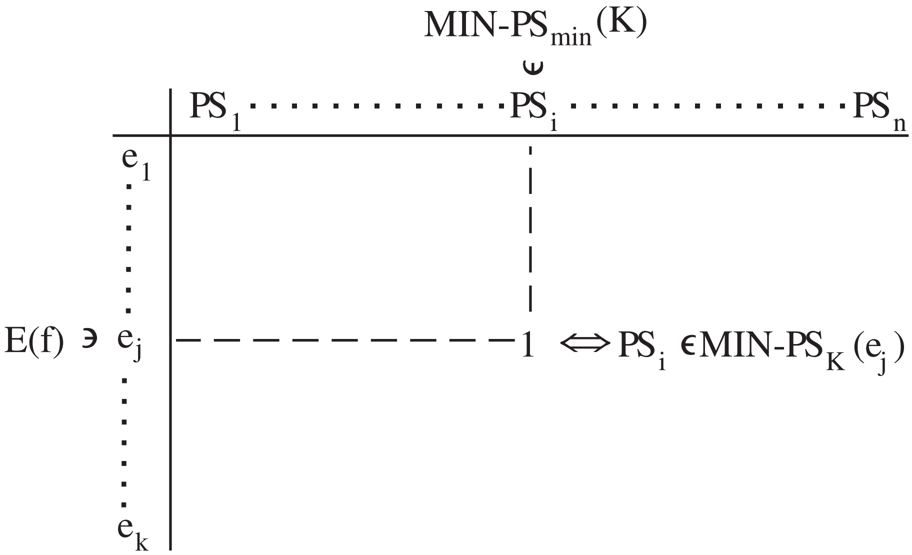

Abbildung 3.6: Das Überdeckungsproblem

Am Schnittpunkt der Zeile von e_j und der Spalte von PS_i steht genau dann eine 1, wenn $PS_i \in \mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e_j)$.
Gesucht ist eine minimale Anzahl von Zeilen, durch deren Einsen sämtliche Spalten der Matrix überdeckt werden. Das Finden der optimalen Lösung eines Überdeckungsproblemes stellt ein NP-vollständiges Problem dar, so daß dieses Optimum durch eine heuristische Lösung angenähert wird. Auf die Implementierung dieser Heuristik wird im nächsten Paragraphen eingegangen.

3.4 Berechnung einer reduzierten Delaytestmenge

In den vorhergehenden Paragraphen wurde gezeigt, daß die funktionale Delaytestmenge für einen gegebenen Schaltkreis zu einer reduzierten Delaytestmenge zusammengestrichen werden kann, ohne daß dabei die Eigenschaft des vollständigen Delaytests verloren geht. In diesem Paragraphen wird auf die Berechnung dieser Testmenge eingegangen. Wir gehen hierbei zuerst von single-output-Schaltkreisen aus und zeigen danach wie man eine reduzierte Testmenge für multi-output-Schaltkreise findet.

3.4.1 Vorberechnung der einelementigen Pfadsysteme

Um die Menge der zu testenden Pfadsysteme des Schaltkreises $\mathcal{K}$ zu berechnen, müssen zuerst zu den Eingaben aus E(f) die zugehörigen Pfadsystemmengen bestimmt werden. Man kann hierbei in einem ersten Schritt zu einer Eingabe e alle Pfadsysteme aus

$\mathcal{MIN}$-$\mathcal{PS}_{\mathcal{K}}(e)$ aufbauen und anschließend die minimalen Elemente auswählen.
Allerdings kann die Anzahl der Pfadsysteme aus der Menge $\mathcal{MIN}$-$\mathcal{PS}_{\mathcal{K}}(e)$ exponentiell in der Anzahl der Pfade des Schaltkreises sein, wodurch sich das Problem ergibt, daß diese Mengen bei einigen Schaltkreisen nicht mehr in zumutbarer Zeit berechnet werden können.
Um die Anzahl der aufgebauten Pfadsysteme gering zu halten, müssen nicht-minimale Pfadsysteme möglichst früh erkannt und eliminiert werden. Hierbei hilft die folgende Beobachtung weiter:

> Die meisten nicht-minimalen Pfadsysteme enthalten einen logischen Pfad, der von einer anderen Eingabe an $\mathcal{K}$ als Einzelpfad getestet wird.

Aus diesem Grund bestimmt man in einer Vorberechnung alle „*Einzelpfadsysteme*" der Menge $\mathcal{MIN}$-$\mathcal{PS}_{min}(\mathcal{K})$.

Definition 3.9:
Ein Einzelpfadsystem *ist ein Pfadsystem, das nur aus einem logischen Pfad besteht.*

Wird nun die Menge $\mathcal{MIN}$-$\mathcal{PS}_{\mathcal{K}}(e)$ zu einer Eingabe e aufgebaut, können nicht-minimale Pfadsysteme, die eines dieser Einzelpfadsysteme enthalten, sofort weggestrichen werden.

3.4.2 Berechnen der Pfadsystemmengen

Mit dem folgenden Algorithmus kann die Pfadsystemmenge zu einer Eingabe e aufgebaut werden. Der Algorithmus orientiert sich an der im Beweis zu Satz 3.2 verwendeten Vorgehensweise. Hierbei wird die Vorberechnung der Einzelpfadsysteme ausgenutzt.

Der Prozedur *berechne* werden als Parameter eine Zelle z und ein Pfad p übergeben. Sei $\tilde{z}$ der primäre Ausgang des Schaltkreises. Dann erfolgt der erste Aufruf durch $berechne(\tilde{z}, \{\})$.

Algorithmus 3.1 (Berechnung von $\mathcal{MIN}$-$\mathcal{PS}_{\mathcal{K}}(e)$ zu einer Eingabe e):

procedure berechne(z,p)
    $\mathcal{MIN}$-$\mathcal{PS}_{\mathcal{K}}(e) := \{\}$
    $PS := \{\}$
    $Einzelsyst$:= Menge der Einzelpfadsysteme
    if ($typ(z) =$ PO)
    then
        \\\\ Nimm die Leitung l, die mit PO verbunden ist, in p_{neu} auf:

```
        p_neu := l mit z = Z(l)
        z_neu := Q(l)
        MIN-PS_K(e) := berechne(z_neu, p_neu)
    fi
    if (typ(z) ∈ {AND, OR, NAND, NOR})
    then
        if (∀ l_i mit Z(l_i) = z : e(l_i) = non_controlling(z))
        then
            forall(l_i mit Z(l_i) = z)
            do
                p_neu := l_i-p
                z_neu := Q(l_i)
                MIN-PS_K(e) := MIN-PS_K(e) ∪ berechne(z_neu, p_neu)
                \\ Die aus „berechne“ zurückgelieferten Pfadsysteme werden
                \\ in MIN-PS_K(e) vereinigt.
                \\ Siehe Fall 2 des Beweises von Satz 3.2.
            od
        else
            L := {l_1, ..., l_k}, mit:
                (e(l_i) = controlling(z)) ∧ (z = Z(l_i)) , i = 1, ..., k
            if (#L = 1)
            then
                p_neu := l_1-p
                z_neu := Q(l_1)
                MIN-PS_K(e) := berechne(z_neu, p_neu)
                \\ Siehe Fall 1 des Beweises von Satz 3.2.
            else
                forall(l_i ∈ L)
                do
                    Menge_i := {}
                    p_neu := l_i-p
                    z_neu := Q(l_i)
                    Menge_i := berechne(z_neu, p_neu)
                    forall((p, t) mit [(p, t)] ∈ Menge_i)
                    do
                        if((p, t) ∈ Einzelsyst)
                        then
                            \\ Der Pfad wird auch als Einzelpfadsystem getestet.
                            Menge_i := Menge_i \ {[(p, t)]}
                        fi
                    od
                    if (Menge_i = {}) \\(*)
                    then
```

```
                    return({})
                    \\ Eine Menge enthält keine Elemente mehr ⟹
                    \\ An dieser Stelle werden keine Pfadsysteme aufgebaut.
                fi
            od
            MIN-PS_K(e) := Kombinationen(Menge_1, ..., Menge_k)
            \\ Es werden die Kombinationen aus den Elementen der Mengen
            \\ gebildet und diese neue Menge an MIN-PS_K(e) zurückgeliefert.
            \\ D.h. MIN-PS_K(e) := {PS_1 ∪ ... ∪ PS_k | PS_i ∈ Menge_i,
            \\ i = 1, ..., k}.
            \\ Siehe Fall 3 des Beweises von Satz 3.2.
        fi
    fi
fi
if (typ(z) ∈ {FAN, NOT})
then
    p_neu := l-p mit z = Z(l)
    z_neu := Q(l)
    MIN-PS_K(e) := berechne(z_neu, p_neu)
fi
if (tpy(z) = PI)
then
    Sei l Leitung mit z = Q(l)
    PS := [(p, ē(l) → e(l))]
    MIN-PS_K(e) := {PS}
fi
return(MIN-PS_K(e))
```

Können aus einer der zu verknüpfenden Mengen alle Elemente gestrichen werden (im Algorithmus mit $(*)$ markiert), so werden an dieser Stelle des Algorithmus keine Pfadsysteme aufgebaut. Es kann die leere Menge zurückgeliefert werden, da keines der aufzubauenden Pfadsysteme minimal gewesen wäre (in jedem dieser Systeme wäre ein Pfad enthalten gewesen, der auch als Einzelpfadsystem getestet wird).

Durch diese Vorgehensweise können Testmengenreduktionen auch für Schaltkreise durchgeführt werden, die ohne die Vorausberechnung nicht bearbeitbar gewesen wären. Dies wird in den praktischen Resultaten des Kapitels 3.5 belegt.

Um die Menge $\mathcal{MIN}\text{-}\mathcal{PS}_{min}(\mathcal{K})$ zu erhalten, müssen aus den noch verbleibenden Elementen die minimalen ausgewählt werden.

3.4.3 Testmengenminimierung

Wie schon erwähnt, läßt sich unser Testmengenminimierungsproblem auf ein Überdeckungsproblem zurückführen. Da dies ein NP-hartes Problem ist, muß es durch einen heuristischen Ansatz gelöst werden. Hierzu kann man auf Lösungsvorschläge aus der Literatur zurückgreifen (siehe z.B. [Hot74, Mic94], Überdeckungsproblem für Primimplikanten).

3.4.4 Multi-Output-Funktionen

Will man die funktionale Delaytestmenge eines multi-output-Schaltkreises reduzieren, ist es von Vorteil von den funktionalen Testmengen der single-output-Schaltkreise auszugehen. Diese Vorgehensweise hat gegenüber der Bearbeitung der schon überlagerten Testmenge des multi-output-Schaltkreises (siehe Kapitel 2.3) den folgenden Vorteil:

> Wir betrachten einen Schaltkreis $\mathcal{K}$, der aus den beiden single-output-Schaltkreisen $\mathcal{K}_1$ und $\mathcal{K}_2$ besteht. Sei e eine Eingabe aus der funktionalen Delaytestmenge zu $\mathcal{K}$, die man durch Überlagerung der Eingaben e_1 und e_2 erhalten hat. Die Pfadsystemmenge der Eingabe e_1 könne vollständig durch andere Eingaben an $\mathcal{K}_1$ überdeckt werden. Bei der Pfadsystemmenge zu e_2 ist eine vollständige Überdeckung durch andere Eingaben an $\mathcal{K}_2$ nicht möglich. Die Eingabe e_1 ist für den Test von $\mathcal{K}_1$ (und somit auch für den Test von $\mathcal{K}$) unnötig und kann gestrichen werden. Da die Pfadsystemmenge zur Eingabe e nicht vollständig überdeckt werden kann, muß die Eingabe auf jeden Fall in unserer reduzierten Testmenge enthalten sein.
> Folglich geht beim Lösen unseres Überdeckungsproblems ein Teil der Flexibilität verloren.

Hat man reduzierte Testmengen zu den single-output-Funktionen eines multi-output-Schaltkreises gebildet, so können diese zu einer einzigen Testmenge für den Gesamtschaltkreis überlagert werden. Hierbei kann wie in Kapitel 2.3 (siehe Seite 42) vorgegangen werden.

Im folgenden wird auf einige experimentelle Resultate eingegangen, die bei der Berechnung unserer reduzierten Delaytestmenge erhalten wurden.

3.5 Praktische Resultate

Für die Testmengenreduktion wurden die Benchmarkschaltkreise aus Kapitel 2.4 verwendet. In Tabelle 3.1 sind die Ergebnisse für die kombinatorischen Teile der ISCAS89-Bechmarks aufgeführt; in Tabelle 3.2 werden die mit SIS synthetisierten MCNC-Benchmarks betrachtet.

Spalte 1 enthält jeweils den Namen der betrachteten Schaltung. In Spalte 2 ist die Anzahl der Testmuster der in Paragraph 2.4 ermittelten FDT angeführt. Spalte 3 gibt die Anzahl

Name	# FDT	# RDT	# $\mathcal{MIN}$-$\mathcal{PS}_{min}(\mathcal{K})$	# (≥ 2)	Zeit	% Verring.
cs27	30	24	50	0	1	20,0
cs298	252	136	419	1	2	46,0
cs344	5596	224	682	6	23	96,0
cs349	5596	222	686	6	23	96,0
cs382	1180	302	797	0	5	74,4
cs400	1174	306	870	47	6	73,9
cs444	1174	382	1009	50	6	67,5
cs510	1256	322	738	0	6	74,4
cs526	620	318	777	1	3	48,7
cs641	10178	636	2729	26	1:39	93,8
cs820	1790	324	984	0	9	82,0
cs832	1770	336	1012	0	9	81,0
cs953	4130	504	2312	0	30	87,8
cs1196	36648	1480	5308	229	6:40	96,0
cs1238	31878	1384	4947	433	5:18	95,7
cs1488	3144	756	1924	0	17	76,0
cs1494	3062	754	1952	0	17	75,4
ϕ						75,6

Tabelle 3.1: RDT für ISCAS89-Benchmarks

der Testpattern der Testmenge an, die wir durch Reduktion der FDT erhalten haben. Zur Vereinfachung wird diese reduzierte Testmenge im folgenden mit RDT abgekürzt.

Um die FDT besser mit der RDT vergleichen zu können, wird in Spalte 7 der Prozentsatz angegeben, um den die funktionale Delaytestmenge verringert werden konnte. Es fällt auf, daß bei vielen Schaltkreisen die Größe der FDT entscheidend eingeschränkt werden konnte. (Die beste Testmengenreduktion ergab sich bei dem Schaltkreis $t481$ mit einer Verringerung um $99,6\%$.)
Es existieren allerdings auch Beispiele, bei denen die Größenreduktion nur gering ist. Bei dem Schaltkreis $e64$ sind die FDT und die RDT sogar identisch. In diesen Fällen war die FDT für die hier betrachtete Schaltkreisrealisierung schon sehr gut und kann dementsprechend nur noch geringfügig eingeschränkt werden.
Durchschnittlich konnte eine Testmengenreduktion von $69,7\%$ erzielt werden ($66,4\%$ bei den MCNC-Benchmarks, $75,6\%$ bei den ISCAS89-Benchmarks), so daß sich in den meisten Fällen eine Anwendung des Algorithmus auszahlt.
Spalte 4 der beiden Tabellen enthält die Anzahl der zu testenden minimalen Pfadsysteme der entsprechenden Schaltkreise. Die Anzahl der Testsequenzen (Anzahl der Testmuster $\div 2$) ist meist viel geringer als die Anzahl dieser Pfadsysteme, so daß durch eine einzige Testsequenz mehrere Pfadsysteme parallel überprüft werden. Als bestes Beispiel sei hier

Name	# FDT	# RDT	# $\mathcal{MIN}$-$\mathcal{PS}_{min}(\mathcal{K})$	# (≥ 2)	Zeit	% Verring.
Z5xp1	1084	702	91316	89189	6:05:00	35,2
Z9sym	2132	256	852	0	27	88,0
apex1	40970	1982	13503	205	15:58	95,2
apex2	9600	522	2760	328	1:28	94,6
apex3	16548	2662	12895	1647	7:10	83,9
apex5	9990	1790	12900	2384	2:08	82,1
b12	186	118	332	17	1	36,6
c5xp1	918	576	5734	4305	1:55	37,3
c9sym	4788	362	1132	15	1:26	92,4
clip	3348	630	1710	147	22	81,2
con1	52	26	44	0	1	50,0
cps	16414	2264	12272	45	2:45	86,2
duke2	9446	812	4896	345	2:12	91,4
e64	132	132	4290	0	25	0,0
ex4	1468	228	2052	0	26	84,5
ex5	3294	1352	6698	100	45	59,0
inc	786	448	1818	561	6	43,0
misex1	186	156	811	157	2	16,1
misex2	238	142	434	3	2	40,3
pdc	8144	836	3146	25	1:02	89,7
rd53	490	164	289	8	1	66,5
rd73	5148	912	7269	5579	4:46	82,3
rd84	11266	1046	3171	492	1:21	90,7
sao2	1472	592	4780	2629	3:34	59,8
spla	6656	1306	5182	96	1:01	80,4
squar5	216	144	347	204	1	33,3
t481	13938	50	128	0	1:00	99,6
table5	81146	2666	14020	3207	37:59	96,7
vg2	2426	242	893	0	19	90,0
ϕ						66,4

Tabelle 3.2: RDT für MCNC-Benchmarks

der Schaltkreis $Z5xp1$ angeführt, bei dem durch eine Testsequenz im Durchschnitt 260 Pfadsysteme überprüft werden.
Spalte 5 enthält die Anzahl der zu testenden Pfadsysteme, die mindestens zwei Pfade enthalten („≥ 2“). Diese „echten “ Pfadsysteme stellen eine Teilmenge von $\mathcal{MIN}$-$\mathcal{PS}_{min}(\mathcal{K})$ dar. Steht in dieser Spalte eine Null, so bedeutet dies, daß der transformierte unate Schaltkreis durch einen robusten Pfaddelaytest vollständig testbar ist. Ein Beispiel, bei dem die

Anzahl der „echten" Pfadsysteme die Anzahl der robust testbaren Einzelpfade („Einzelpfadsysteme") bei weitem übersteigt, stellt der Schaltkreis $Z5xp1$ dar. Da in diesen Pfadsystemen nur Pfade enthalten sind, die nicht durch einen robusten Einzelpfadtest überprüft werden können, verdeutlicht diese hohe Anzahl nochmals, daß der Pfadsystemtest wesentlich mächtiger als der Pfaddelaytest ist.
Spalte 6 gibt die Laufzeit des Algorithmus an. Für die meisten betrachteten Schaltkreise liegt die Laufzeit im Sekunden- bzw. im Minutenbereich.
Eine Ausnahme bildet wiederum der Schaltkreis $Z5xp1$, für den eine Laufzeit im Stundenbereich benötigt wird. Dies liegt an der extrem hohen Anzahl von Pfadsystemen, die für die Eingaben aufgebaut werden müssen (also: $\mathcal{MIN}$-$\mathcal{PS}_{\mathcal{K}}(e)$). Aus diesen Pfadsystemmengen erweisen sich sogar mehr als 90000 Pfadsysteme als notwendig (also: $\mathcal{MIN}$-$\mathcal{PS}_{min}(\mathcal{K})$).

Durch Tabelle 3.3 wird dargestellt, wie wichtig es ist, eine Vorausberechnung der Einzelpfadsysteme durchzuführen. Spalte 1 gibt wieder den Namen der betrachteten Schaltung an, wobei hier nur einige charakteristische Schaltkreise ausgewählt worden sind. Spalte 2 enthält die Laufzeiten der Heuristik ohne Vorausberechnung. Zum Vergleich sind diesen in Spalte 3 die Ergebnisse der Heuristik mit Vorausberechnung gegenübergestellt.

Man sieht, daß durch die Vorausberechnung die Laufzeit drastisch reduziert werden kann: Zum Beispiel kann bei Schaltkreis $c5xp1$ die Laufzeit durch die Vorausberechnung vom Stundenbereich in den Minutenbereich gesenkt werden. Der entscheidende Vorteil tritt bei den Schaltkreisen auf, bei denen in der ersten Spalte ein Fragezeichen steht. Diese Schaltkreise waren ohne eine Vorausberechnung aus Laufzeit- oder Speicherplatzgründen nicht bearbeitbar. Durch die Vorausberechnung der Einzelpfadsysteme kann nun auch bei diesen Schaltkreisen die funktionale Delaytestmenge reduziert werden. (Zum Beispiel kann für den vorher nicht bearbeitbaren Schaltkreis $rd84$ die Testmengenreduktion in weniger als 2 Minuten vorgenommen werden.)
Bei Schaltkreisen, die nur wenig Rechenzeit benötigen, kann sich die Laufzeit durch die Vorausberechnung erhöhen, was aber bei diesen kleinen Rechenzeiten nur geringfügig ins Gewicht fällt. (Zum Beispiel erhöht sich bei Schaltkreis cps die Rechenzeit von 1 Minute 37 Sekunden auf 2 Minuten 45 Sekunden.)

Im letzten Kapitel wurde die Größe der funktionalen Testmenge mit den Größen der Testmengen eines kompaktierenden Testmustergenerators (siehe [Red92]) verglichen. Dieser Testmustergenerator führt einen robusten Einzelpfadtest durch. Aus diesem Grund sind dessen Testmengen für viele der betrachteten Schaltkreise nicht vollständig (siehe hierzu auch Tabelle 2.1 auf Seite 46 und Tabelle 2.2 auf Seite 47). Bei fast allen betrachteten Schaltkreisen war die funktionale Testmenge größer als die Testmenge des Testmustergenerators. Es stellt sich die Frage, wie gut nun die Größen der reduzierten Delaytestmengen gegenüber den vom Testmustergenerator erzeugten Testmengen sind. Die Ergebnisse sind in den Tabellen 3.4 und 3.5 dargestellt.

Spalte 1 enthält den Namen der betrachteten Schaltung. Zum besseren Vergleich wird in Spalte 2 nochmals die Größe der funktionalen Delay-Testmenge angegeben. In Spalte 3 ist die Größe unserer reduzierten Testmengen dargestellt. Spalte 4 beinhaltet die kompaktierte Testmengengröße des Testmustergenerators.

Name	Zeit ohne Vorausberechnung	Zeit mit Vorausberechnung
Z5xp1	?	6:05:00
Z9sym	24	27
apex1	12:46	15:58
apex2	59	1:28
apex3	6:01	7:10
c5xp1	3:39:23	1:55
c9sym	1:33	1:26
clip	30	22
cps	1:37	2:45
duke2	1:26	2:12
ex4	16	26
ex5	28	45
inc	1:23	6
misex1	5	2
misex2	1	2
pdc	38	1:02
rd53	1	1
rd73	?	4:46
rd84	?	1:21
sao2	?	3:34
spla	35	1:01
t481	35	1:00
table5	40:56	37:59

Tabelle 3.3: Zeitvergleich für Vorausberechnung

Beim Vergleich der Zahlen fällt auf, daß durch die Reduktion die großen Unterschiede zwischen unserer Testmengengröße und der des kompaktierenden Testmustergenerators weggefallen sind. Bei den ISCAS89-Benchmarks sind unsere Testmengen zwar stets größer, sie nehmen aber nie mehr als die doppelte Größe des robusten Einzelpfadtests an.
Beim Vergleich der Testmengengrößen für die MCNC-Benchmarks bemerkt man, daß unsere vollständige Testmenge für viele der betrachteten Schaltkreise besser ist. Das eindrucksvollste Beispiel stellt der Schaltkreis *c9sym* dar, für dessen Test der Testmustergenerator aus [Red92] mehr als die doppelte Menge von Testeingaben benötigt.
Es ist verwunderlich, daß die *vollständigen* Testmengen der reduzierten Testmenge oft kleiner sind als die *unvollständigen* Testmengen des Testmustergenerators. Dies liegt an den unterschiedlichen Vorgehensweisen bei der Testmengenkonstruktion (siehe hierzu [Red92]).

Name	# FDT	# RDT	# Tests(Path Delay)
cs27	30	24	22
cs298	252	136	89
cs344	5596	224	169
cs349	5596	222	176
cs382	1180	302	154
cs400	1174	306	168
cs444	1174	382	222
cs510	1256	322	245
cs526	620	318	167
cs641	10178	636	436
cs820	1790	324	287
cs832	1770	336	312
cs953	4130	504	389
cs1196	36648	1480	1385
cs1238	31878	1384	1168
cs1488	3144	756	509
cs1494	3062	754	497

Tabelle 3.4: Vergleich 1 für ISCAS89-Benchmarks

Somit stellt sich folgende interessante Frage: Um wieviel größer sind unsere Testmengen im Vergleich zu den Testmengen eines Einzelpfadtests, bei dem mit der gleichen Methodik für die Testmengenminimierung vorgegangen wird? D.h., um wieviel wächst die Testmenge an, wenn man statt einem (unvollständigen) Pfaddelaytest einen (vollständigen) Pfadsystemtest durchführt?
Hierzu wurde ein Programm entwickelt, das eine möglichst kleine Testmenge für den Einzelpfadtest erzeugt und die gleiche Vorgehensweise bei der Testmengenminimierung verwendet. Die Ergebnisse sind in den Tabellen 3.6 und 3.7 festgehalten.

Spalte 1 enthält wieder den Namen des betrachteten Schaltkreises, Spalte 2 gibt die Anzahl der zu testenden Pfade des Gesamtschaltkreises an (siehe hierzu auch Kapitel 2.4). In Spalte 3 ist die Anzahl der robust getesteten Einzelpfade angegeben[2]. Diese Pfade werden *sowohl* durch den Pfaddelaytest *als auch* durch die reduzierte Testmenge überprüft. Spalte 4 enthält die Anzahl der zu testenden Pfadsysteme, die mehr als zwei logische Pfade enthalten (d.h. die Menge $\mathcal{MIN}$-$\mathcal{PS}_{min}(\mathcal{K})$ ohne die Einzelpfadsysteme). Diese Pfadsysteme werden *nur* durch unseren Pfadsystemtest überprüft.
In Spalte 5 ist die Testmenengröße der RDT angegeben, zum Vergleich dazu enthält

[2]Einige Zahlen unterscheiden sich geringfügig von der Anzahl der robust getesteten Einzelpfade in Tabelle 2.2. Für diese Schaltkreise hat der Testmustergenerator einige Pfade wegen des gesetzten Backtracklimits entfallen lassen.

Name	# FDT	# RDT	#Tests(Path Delay)
Z5xp1	1084	702	1026
Z9sym	2132	256	669
apex1	40970	1982	2437
apex2	9600	522	800
apex3	16548	2662	3101
apex5	9990	1790	1291
b12	186	118	131
c5xp1	918	576	839
c9sym	4788	362	847
clip	3348	630	1022
con1	52	26	22
cps	16414	2264	1865
duke2	9446	812	901
e64	132	132	135
ex4	1468	228	380
ex5	3294	1352	970
inc	786	448	689
misex1	186	156	252
misex2	238	142	108
pdc	8144	836	673
rd53	490	164	243
rd73	5148	912	1225
rd84	11266	1046	1659
sao2	1472	592	1122
spla	6656	1306	1133
squar5	216	144	175
t481	13938	50	94
table5	81146	2666	2700
vg2	2426	242	358

Tabelle 3.5: Vergleich 1 für MCNC-Benchmarks

Spalte 6 die Testmengengrößen für den Pfaddelaytest. Ist ein Schaltkreis mit dem Pfaddelaytest vollständig testbar (in Spalte 4 steht eine Null), sind die Testmengengrößen identisch, was daran liegt, daß beide Programme die gleiche Vorgehensweise bei der Testmengenminimierung verwenden. Bei den anderen Schaltkreisen sieht man, daß sich die Testgröße durch den zusätzlichen Test der Pfadsysteme nur geringfügig erhöht. Das beste Beispiel stellt der Schaltkreis $Z5xp1$ dar:

Der Schaltkreis enthält 1101 Pfade, die für einen vollständigen Delaytest des

Name	# Zu testende Pfade	# rob. getestet	# (≥ 2)	# RDT	# PD-Test
cs27	50	50	0	24	24
cs298	420	418	1	136	134
cs344	682	676	6	224	220
cs349	686	680	6	222	218
cs382	797	797	0	302	302
cs400	837	823	47	306	306
cs444	976	959	50	382	376
cs510	738	738	0	322	322
cs526	778	776	1	318	318
cs641	2711	2703	26	636	630
cs820	984	984	0	324	324
cs832	1012	1012	0	336	336
cs953	2312	2312	0	504	504
cs1196	5316	5079	229	1480	1474
cs1238	5028	4514	433	1384	1358
cs1488	1924	1924	0	756	756
cs1494	1952	1952	0	754	754

Tabelle 3.6: Vergleich 2 für ISCAS89-Benchmarks

Schaltkreises zu testen sind, durch einen Pfaddelaytest aber nicht getestet werden. Für den zusätzlichen Test dieser Pfade erhöht sich die Anzahl der benötigten Testmuster nur um 74.

Vergleicht man die Schaltkreise in den Tabellen 3.1 - 3.7 mit den Schaltkreisen aus Paragraph 2.4, fällt auf, daß hier einige Schaltkreise fehlen. Für diese Beispielschaltkreise ist die Anzahl der aufgebauten Pfadsysteme so hoch, daß sie mit unserem Algorithmus nicht mehr bearbeitet werden können. Entweder läuft der Speicher bei der Verwaltung der Systeme über oder der Algorithmus kann die RDT nicht mehr in akzeptabler Zeit bilden. Will man auch für diese Schaltkreise die Testmenge einschränken, muß eine Methodik bereitgestellt werden, bei der weniger Pfadsysteme aufgebaut werden. Ein solcher Algorithmus wird im nächsten Kapitel vorgestellt. Im Vergleich zu den in diesem Kapitel gebildeten Testmengen wird bei dieser „groberen“ Vorgehensweise die Testmengenreduktion geringer ausfallen. Trotzdem wird sich die Testmengengröße gegenüber der funktionalen Delaytestmenge stark vermindern lassen.

Name	# Zu testende Pfade	# rob. getestet	# (≥ 2)	# RDT	# PD-Test
Z5xp1	3230	2129	89189	702	628
Z9sym	852	852	0	256	256
apex1	13385	13298	205	1982	1972
apex2	2512	2432	328	522	502
apex3	12056	11248	1647	2662	2634
apex5	11123	10516	1291	1790	1780
b12	326	315	17	118	114
c5xp1	1850	1429	4305	576	516
c9sym	1134	1117	15	362	362
clip	1798	1563	147	630	618
con1	44	44	0	26	26
cps	12267	12227	45	2264	2264
duke2	4734	4551	345	812	804
e64	4290	4290	0	132	132
ex4	2052	2052	0	228	228
ex5	6829	6598	100	1352	1350
inc	1534	1257	561	448	432
misex1	759	654	157	156	150
misex2	443	431	3	142	142
pdc	3173	3121	25	836	832
rd53	288	281	8	164	164
rd73	1971	1690	5579	912	750
rd84	2933	2679	492	1046	998
sao2	2683	2151	2629	592	528
spla	5180	5086	96	1306	1276
squar5	384	143	204	144	142
t481	128	128	0	50	50
table5	11234	10813	3207	2666	2240
vg2	893	893	0	242	242

Tabelle 3.7: Vergleich 2 für MCNC-Benchmarks

Kapitel 4

Beschleunigung der Testmengenreduktion

Im vorigen Kapitel wurde eine Methodik vorgestellt, mit der es möglich ist, die Größe der funktionalen Delaytestmenge einzuschränken. Die entstehende Testmenge überprüft eine gegebene Schaltkreisrealisierung vollständig auf Delay-Fehler. Sie hat allerdings den Nachteil, daß für ihre Konstruktion die Menge $\mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ zu allen Eingaben der funktionalen Testmenge aufgebaut werden muß, welche im schlechtesten Fall exponentiell in der Anzahl der Pfade sein kann. Aus diesem Grund ist solch eine reduzierte Testmenge für Schaltkreise mit vielen Pfadsystemen nicht mehr berechenbar oder es wird eine sehr hohe Rechenzeit zur Berechnung der reduzierten Testmenge benötigt. Um auch für diese Schaltkreise die Größe der FDT einzuschränken, muß eine Methodik entwickelt werden, die die Anzahl der aufgebauten Pfadsysteme gering hält. Da die hier verwendete Vorgehensweise „grober" ist als die Methodik des vorigen Kapitels (es werden nicht die Pfadsysteme aus $\mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$, sondern größere, mehrere Pfadsysteme aus $\mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ umfassende Systeme betrachtet), kann die Testmenge nur in geringerem Maß komprimiert werden. Im Vergleich wird die Güte der Resultate der hier betrachteten Vorgehensweise also schlechter sein als die der Resultate des vorigen Ansatzes. Der Vorteil ist, daß sich die Rechenzeit für die Testmengenreduktion verbessern wird und alle von uns betrachteten Benchmarkschaltkreise bearbeitet werden können. Auch bei diesem Ansatz ist die reduzierte Testmenge wieder ein *vollständiger Delaytest* für den gegebenen Schaltkreis.

Die Vorgehensweise zur Reduzierung der Größe der FDT ist ähnlich wie bei der Reduktion des vorigen Kapitels, so daß auch der Aufbau dieses Kapitel stark an den des letzten Kapitels angelehnt ist.
Analog zum letzten Kapitel wird für eine Eingabe e eine Menge von Pfadsystemen bestimmt, deren Test bezüglich einer Laufzeitschranke τ sicherstellt, daß das worst-case-Delay von e kleiner oder gleich dieser Laufzeitschranke ist. Der Unterschied zwischen der hier gebildeten Menge $\mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ und der Menge $\mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ des letzten Kapitels ist, daß die Anzahl der Pfadsysteme in der Menge $\mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ exponentiell in der Anzahl

der Pfade sein kann, wohingegen die Anzahl der Pfadsysteme in der Menge $\mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ nur linear in der Anzahl der Pfade ist.

Die verbleibende Vorgehensweise ist analog zu der des letzten Kapitels. Mit Hilfe dieser Pfadsystemmengen der einzelnen Eingaben kann eine Menge von umfassenden Pfadsystemen $\mathcal{U}\text{-}\mathcal{PS}(\mathcal{K})$ des Schaltkreises gebildet werden, deren Test für einen vollständigen Delaytest des Schaltkreises ausreicht. Die Größe der FDT kann nun reduziert werden, indem eine Teilmenge der funktionalen Testmenge bestimmt wird, die alle Elemente aus $\mathcal{U}\text{-}\mathcal{PS}(\mathcal{K})$ überdeckt. Abgeschlossen wird dieses Kapitel wiederum durch praktische Resultate.

4.1 Bestimmung der zu testenden umfassenden Pfadsysteme

4.1.1 Das maximale Pfadsystem zu einer Eingabe

Der Nachteil der Technik von Kapitel 3 zur Reduzierung der FDT ist, daß hierbei im schlechtesten Fall exponentiell viele Pfadsysteme pro Eingabe aufgebaut werden. Ziel des hier verwendeten Ansatzes ist es, diese Anzahl der aufgebauten Pfadsysteme möglichst gering zu halten.
Aus diesem Grund wird in diesem Kapitel eine andere Art von Pfadsystemen vorgestellt: Die *maximalen Pfadsyteme* und anschließend die *umfassenden Pfadsysteme*, deren Konstruktion auf die maximalen Pfadsysteme zurückgeht.

Durch ein maximales Pfadsystem können alle Minimums-Pfadsysteme aus der Pfadsystemmenge $\mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ zu einer Eingabe e zu *einem einzigen* Pfadsystem zusammengefaßt werden.

Definition 4.1:
Sei $\mathcal{K}$ ein kombinatorischer Schaltkreis mit genau einem primären Ausgang und e eine Eingabe an $\mathcal{K}$. Das maximale Pfadsystem *zur Eingabe e ($MAX\text{-}PS_{\mathcal{K}}(e)$) wird induktiv über die Anzahl der Zellen von $\mathcal{K}$ definiert, wobei primäre Inputs und Outputs des Schaltkreises nicht zu den Zellen gerechnet werden.*

a) $\mathcal{K}$ enthält keine Zellen, d.h. $\mathcal{K}$ besteht nur aus der Leitung l.

$$MAX\text{-}PS_{\mathcal{K}}(e) := \{(l, \overline{e} \rightarrow e)\}$$

b) $\mathcal{K}$ enthält $\alpha > 0$ Zellen.
Die vorliegende Situation ist in Abbildung 4.1 dargestellt. Sei z die Zelle, die den primären Ausgang von $\mathcal{K}$ speist.

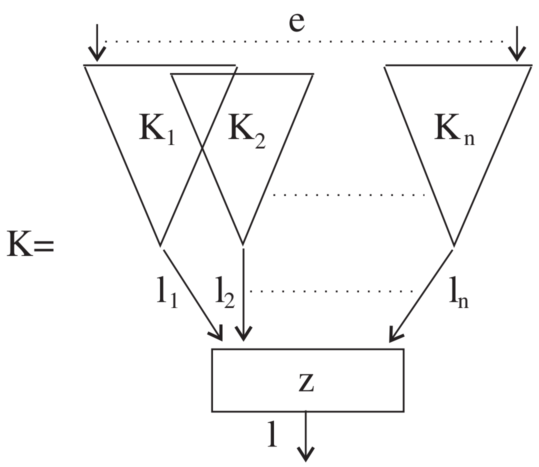

Abbildung 4.1: Situation an Zelle z

<u>FALL 1:</u> *z ist ein Fanoutknoten, d.h. n=1.*
In diesem Fall setzen wir:

$$MAX\text{-}PS_{\mathcal{K}}(e) := MAX\text{-}PS_{\mathcal{K}_1}(e) \circ z = \{(p\text{-}l, t) | (p, t) \in MAX\text{-}PS_{\mathcal{K}_1}(e)\}$$

<u>FALL 2:</u> *z ist kein Fanoutknoten.*
Sei L die Menge der Inputleitungen von z, die bei Eingabe e auf den kontrollierenden Wert $k(z)$ gesetzt wird.

<u>UNTERFALL 1:</u> $L \neq \emptyset$
Wir setzen:

$$MAX\text{-}PS_{\mathcal{K}}(e) := \bigcup_{l_i \in L} (MAX\text{-}PS_{\mathcal{K}_i}(e) \circ z)$$

<u>UNTERFALL 2:</u> $L = \emptyset$
Wir setzen:

$$MAX\text{-}PS_{\mathcal{K}}(e) := \bigcup_{i=1,\ldots,n} (MAX\text{-}PS_{\mathcal{K}_i}(e) \circ z)$$

Liegen an den Inputleitungen einer Zelle nur nicht-kontrollierende Werte an, werden alle Inputleitungen in das maximale Pfadsystem aufgenommen. Ansonsten werden alle Inputleitungen aufgenommen, an denen der kontrollierende Wert anliegt.

Beispiel 4.1 :
Wir betrachten den Schaltkreis in Abbildung 4.2. Die Pfade des Schaltkreises seien wie

folgt bestimmt:

$$\begin{array}{ll} p_1 = l_1\text{-}l_{15} & p_2 = l_2\text{-}l_{10}\text{-}l_{13}\text{-}l_{15} \\ p_3 = l_3\text{-}l_4\text{-}l_{10}\text{-}l_{13}\text{-}l_{15} & p_4 = l_3\text{-}l_5\text{-}l_{11}\text{-}l_{13}\text{-}l_{15} \\ p_5 = l_6\text{-}l_{11}\text{-}l_{13}\text{-}l_{15} & p_6 = l_7\text{-}l_{12}\text{-}l_{14}\text{-}l_{15} \\ p_7 = l_8\text{-}l_{12}\text{-}l_{14}\text{-}l_{15} & p_8 = l_9\text{-}l_{14}\text{-}l_{15} \end{array}$$

Die Belegungen der Leitungen aus $\mathcal{K}$ seien wie in Abbildung 4.3 gegeben.

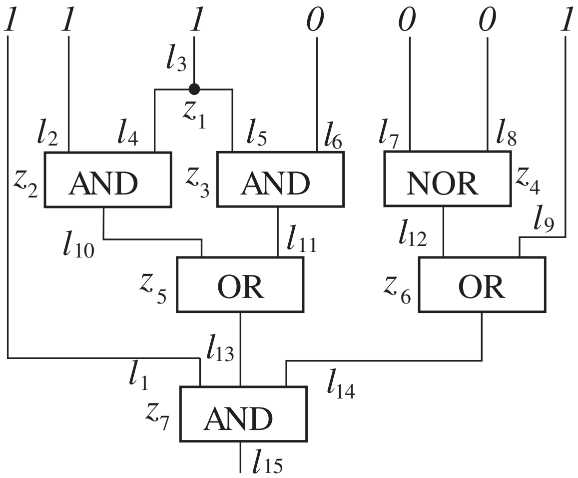

Abbildung 4.2: Beispielschaltkreis $\mathcal{K}$

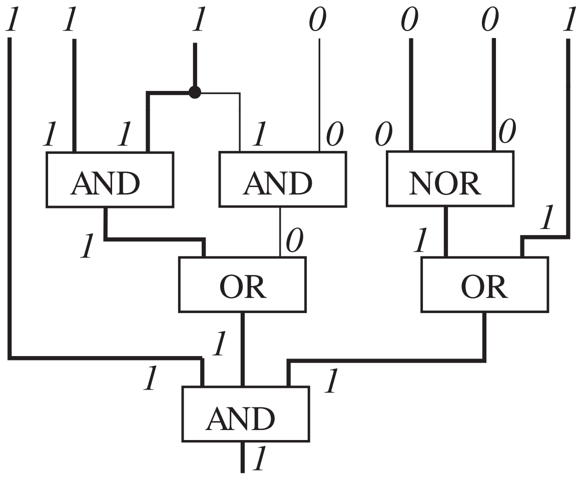

Abbildung 4.3: Belegungen der Leitungen

Sei MAX-PS$_{\mathcal{K}_i}(e)$ das maximale Pfadsystem für den Teilschaltkreis $\mathcal{K}_i$ mit der Ausgangsleitung l_i $(i = 1, \ldots, 15)$ bei Eingabe e. Die Belegungen der Inputs der Zelle z_7 für Eingabe e sind alle nicht-kontrollierend, so daß gilt:

$$\text{MAX-PS}_{\mathcal{K}}(e) = (\text{MAX-PS}_{\mathcal{K}_1}(e) \circ z_7) \cup (\text{MAX-PS}_{\mathcal{K}_{13}}(e) \circ z_7) \cup (\text{MAX-PS}_{\mathcal{K}_{14}}(e) \circ z_7)$$

$\mathcal{K}_1$ besteht nur aus der Leitung l_1, somit ist MAX-PS$_{\mathcal{K}_1}(e)$:

$$\text{MAX-PS}_{\mathcal{K}_1}(e) = \{(l_1, \uparrow)\}$$

Bei Eingabe e liegt an Zelle z_5 genau ein kontrollierender Wert an, an den Inputs der Zelle z_2 sind alle Belegungen nicht-kontrollierend. Zelle z_1 ist ein Fanoutknoten. Es folgt für MAX-PS$_{\mathcal{K}_{13}}(e)$:

$$\begin{aligned} \text{MAX-PS}_{\mathcal{K}_{13}}(e) &= \text{MAX-PS}_{\mathcal{K}_{10}}(e) \circ z_5 \\ &= ((\{(l_2, \uparrow)\} \circ z_2) \cup (\{(l_3\text{-}l_4, \uparrow)\} \circ z_2)) \circ z_5 \\ &= \{(l_2\text{-}l_{10}, \uparrow), (l_3\text{-}l_4\text{-}l_{10}, \uparrow)\} \circ z_5 \\ &= \{(l_2\text{-}l_{10}\text{-}l_{13}, \uparrow), (l_3\text{-}l_4\text{-}l_{10}\text{-}l_{13}, \uparrow)\} \end{aligned}$$

Die Belegungen der Inputs der Zelle z_6 für die Eingabe e sind beide kontrollierend bezüglich der Zelle z_6, an Zelle z_4 liegen für e nur nicht-kontrollierende Werte an. Somit gilt:

$$\text{MAX-PS}_{\mathcal{K}_{14}}(e) = (\text{MAX-PS}_{\mathcal{K}_{12}}(e) \circ z_6) \cup (\text{MAX-PS}_{\mathcal{K}_9}(e) \circ z_6)$$

$$\begin{aligned} &= (\{(l_7\text{-}l_{12}, \downarrow), (l_8\text{-}l_{12}, \downarrow)\} \circ z_6) \cup (\{(l_9, \uparrow)\} \circ z_6) \\ &= \{(l_7\text{-}l_{12}\text{-}l_{14}, \downarrow), (l_8\text{-}l_{12}\text{-}l_{14}, \downarrow), (l_9\text{-}l_{14}, \uparrow)\} \end{aligned}$$

Somit folgt für das maximale Pfadsystem des Schaltkreises zur Eingabe e:

$$\text{MAX-PS}_{\mathcal{K}}(e) = \{(p_1, \uparrow), (p_2, \uparrow), (p_3, \uparrow), (p_6, \downarrow), (p_7, \downarrow), (p_8, \uparrow)\}$$

Dieses maximale Pfadsystem ist in Abbildung 4.3 dicker eingezeichnet. ∎

Satz 4.2:
Sei $\mathcal{K}$ ein unate Schaltkreis mit genau einem primären Ausgang und e eine Eingabe an $\mathcal{K}$. Für das maximale Pfadsystem zur Eingabe e gilt:

$$d_{\mathcal{K}}(e) = d_{\mathcal{K}}(\textit{MAX-PS}_{\mathcal{K}}(e))$$

($d_{\mathcal{K}}(\textit{MAX-PS}_{\mathcal{K}}(e))$ bezeichnet das Delay des Pfadsystems $\textit{MAX-PS}_{\mathcal{K}}(e)$,siehe Def. 1.23.)

Beweis:
Der Beweis erfolgt durch vollständige Induktion über die Anzahl der Zellen des Schaltkreises $\mathcal{K}$, wobei primäre Inputs und Outputs des Schaltkreises nicht zu den Zellen gerechnet werden.

INDUKTIONSANFANG: $\mathcal{K}$ enthält keine Zellen, d.h. $\mathcal{K}$ enthält genau eine Leitung l.
MAX-PS$_{\mathcal{K}}(e)$ ist das Pfadsystem mit dem logischen Pfad $(l, \overline{e} \to e)$, d.h.

$$\text{MAX-PS}_{\mathcal{K}}(e) = \{(l, \overline{e} \to e)\}$$

INDUKTIONSSCHRITT: $\mathcal{K}$ enthält $\alpha > 0$ Zellen.
Sei l die Leitung, die mit dem primären Ausgang von $\mathcal{K}$ verbunden ist. Sei z die Zelle, deren Ausgang mit der Leitung l verbunden ist.
Sei o.B.d.A. z ein AND-Gatter mit den Input-Leitungen $l_1, \ldots, l_n$ (für die anderen Fälle analog). Der Teilschaltkreis, dessen Ausgangsleitung die Leitung l_i ist, wird mit $\mathcal{K}_i$ bezeichnet $(i = 1, \ldots, n)$.
$d_{\mathcal{K}_i}(e)$ bezeichnet das (worst-case) Delay der Eingabe e bezüglich des Schaltkreises $\mathcal{K}_i$ $(i = 1, \ldots, n)$. Die beschriebene Situation ist in Abb. 4.4 dargestellt.

Da die Zellenanzahlen dieser Schaltkreise kleiner als α sind, kann induktiv angenommen werden, daß Satz 4.2 für diese Teilschaltungen erfüllt ist, d.h. für die maximalen Pfadsysteme MAX-PS$_{\mathcal{K}_i}(e)$, $i = 1, \ldots n$, gilt:

$$d_{\mathcal{K}_i}(e) = d_{\mathcal{K}_i}(\text{MAX-PS}_{\mathcal{K}_i}(e)) \tag{4.1}$$

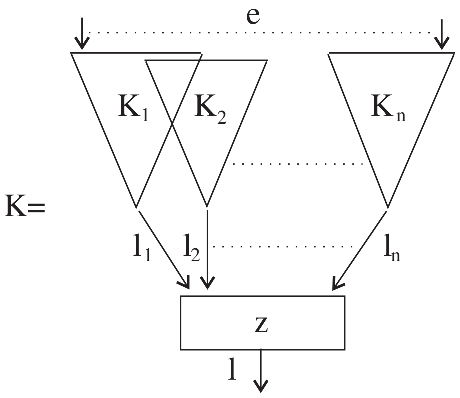

Abbildung 4.4: Struktur des Schaltkreises $\mathcal{K}$

Sei L die Menge der Inputleitungen von z, die bei Eingabe e auf den kontrollierenden Wert k(z) gesetzt wird.

<u>FALL 1:</u> $L \neq \emptyset$
Es gilt:

$$d_{\mathcal{K}}(e) = \min\{d_{\mathcal{K}_i}(e) + d_{\downarrow,l_i}(z) | \forall\, i \text{ mit } l_i \in L\} \tag{4.2}$$

Aus den Formeln (4.1) und (4.2) folgt:

$$d_{\mathcal{K}}(e) = \min\{d_{\mathcal{K}_i}(\text{MAX-PS}_{\mathcal{K}_i}(e)) + d_{\downarrow,l_i}(z) | \forall\, i \text{ mit } l_i \in L\}$$

Da an allen Leitungen aus L bei Eingabe e der kontrollierende Wert von z anliegt, werden nach Definition 4.1 die maximalen Pfadsysteme aller mit diesen Inputs verbundenen Teilschaltkreise um die Outputleitung l der Zelle z verlängert und in einem einzigen maximalen Pfadsystem des Gesamtschaltkreises vereinigt. Nach der Delayformel eines Pfadsystems (siehe Def. 1.23) gilt:

$$d_{\mathcal{K}}(\text{MAX-PS}_{\mathcal{K}}(e)) = \min\{d_{\mathcal{K}_i}(\text{MAX-PS}_{\mathcal{K}_i}(e)) + d_{\downarrow,l_i}(z) | \forall\, i \text{ mit } l_i \in L\}$$

Somit folgt:

$$d_{\mathcal{K}}(e) = d_{\mathcal{K}}(\text{MAX-PS}_{\mathcal{K}}(e))$$

<u>FALL 2:</u> $L = \emptyset$
Es gilt:

$$d_{\mathcal{K}}(e) = \max\{d_{\mathcal{K}_i}(e) + d_{\uparrow,l_i}(z) | \, i = 1, \ldots, n\} \tag{4.3}$$

Aus den Formeln (4.1) und (4.3) folgt:

$$d_{\mathcal{K}}(e) = \max\{d_{\mathcal{K}_i}(\text{MAX-PS}_{\mathcal{K}_i}(e)) + d_{\uparrow,l_i}(z) | \, i = 1, \ldots, n\}$$

Da an allen Inputleitungen von z der nicht-kontrollierende Wert anliegt, werden nach Definition 4.1 die maximalen Pfadsysteme aller mit diesen Inputs verbundenen Teilschaltkreise um die Outputleitung l der Zelle z verlängert und in einem einzigen maximalen

Pfadsystem des Gesamtschaltkreises vereinigt. Nach der Delayformel eines Pfadsystems (siehe Def. 1.23) gilt:

$$d_{\mathcal{K}}(\text{MAX-PS}_{\mathcal{K}}(e)) = \max\{d_{\mathcal{K}_i}(\text{MAX-PS}_{\mathcal{K}_i}(e)) + d_{\uparrow,l_i}(z) | \, i = 1, \ldots, n\}$$

Somit folgt:

$$d_{\mathcal{K}}(e) = d_{\mathcal{K}}(\text{MAX-PS}_{\mathcal{K}}(e))$$

■

Somit würde es für den Test einer Eingabe e ausreichen, das zugehörige maximale Pfadsystem zu dieser Eingabe zu überprüfen. Da aber jeder Eingabe jeweils nur noch *ein* Pfadsystem zugeordnet wäre, wäre man beim später zu lösenden Überdeckungsproblem sehr unflexibel. Aus diesem Grund werden im folgenden die *umfassenden Pfadsysteme* eingeführt, die flexibler als die maximalen Pfadsysteme sind.

4.1.2 Zu testende umfassende Pfadsysteme für eine Eingabe

Im vorigen Paragraphen wurden die maximalen Pfadsysteme eingeführt, die es ermöglichen alle Minimums-Pfadsysteme aus der Menge $\mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ zu einer Eingabe e zu einem einzigen Pfadsystem zusammenzufassen. Da die maximalen Pfadsysteme aber sehr unflexibel zu handhaben sind, wird hier eine andere Art von Pfadsystemen vorgestellt, die umfassenden Pfadsysteme, deren Konstruktion auf den maximalen Pfadsytemen aufbaut.

Durch solch ein umfassendes Pfadsystem können mehrere Minimums-Pfadsysteme zu einem einzigen System zusammengefaßt werden. Um die hier betrachteten Pfadsysteme besser von den Minimums-Pfadsystemen des vorigen Kapitels unterscheiden zu können, werden diese umfassenden Pfadsysteme im folgenden mit UPS bezeichnet und in das Klammerpaar „$<,>$“ eingeschlossen.

Es soll eine Menge von umfassenden Pfadsystemen einer Eingabe e zugeordnet werden, so daß der Test dieser Menge bezüglich einer Laufzeitschranke τ garantiert, daß auch das worst-case-Delay der Eingabe kleiner oder gleich der Laufzeitschranke ist. Diese Menge von umfassenden Pfadsystemen wird im folgenden Algorithmus bestimmt.

Definition 4.3:
Sei $\mathcal{K}$ ein Schaltkreis mit genau einem primären Ausgang und e eine Eingabe an $\mathcal{K}$.

> *Die in Algorithmus 4.1 bestimmte Menge von umfassenden Pfadsystemen zur Eingabe e wird mit $\mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ bezeichnet.*

Der folgenden Prozedur werden als Parameter eine Zelle z und eine Ausgangsleitung l von z übergeben. Betrachtet wird wiederum ein Schaltkreis $\mathcal{K}$ mit nur einem Ausgang. Sei $\tilde{z}$ der primäre Ausgang des Schaltkreises. Dann erfolgt der erste Aufruf der Prozedur mit: $berechne\_UPS(\tilde{z}, \{\})$.

Algorithmus 4.1 (Berechnung von $\mathcal{U}$-$\mathcal{PS}_{\mathcal{K}}(e)$ zu einer Eingabe e):

```
procedure berechne_UPS(z,l)
    U-PS_K(e) := {}
    U-PS := {}
    if (typ(z) = PO)
    then
        l_1 := Leitung mit z = Z(l_1)
        z_1 := Q(l_1)
        U-PS_K(e) := berechne_UPS(z_1, l_1)
    fi
    if (typ(z) ∈ {AND, OR, NAND, NOR})
    then
        if (∀l_i mit Z(l_i) = z : e(l_i) = non_controlling(z))
        then
            forall(l_i mit Z(l_i) = z)
            do
                z_i := Q(l_i)
                U-PS := U-PS ∪ berechne_UPS(z_i, l_i)  \\ (FALL 2)
                \\ Die aus „berechne_UPS“ zurückgelieferten umfassenden
                \\ Pfadsystemmengen werden in U-PS vereinigt.
            od
            U-PS_K(e) := {{(p-l, t)|(p, t) ∈ UPS}|UPS ∈ U-PS}
            \\ Die Pfade der umfassenden Pfadsysteme aus U-PS werden um die
            \\ Ausgangsleitung l von z verlängert.
        else
            L := {l_1, ..., l_k}, mit:
                (e(l_i) = controlling(z)) ∧ (z = Z(l_i)) , i = 1, ..., k
            if (#L = 1)
            then
                z_1 := Q(l_1)
                U-PS := berechne_UPS(z_1, l_1)  \\ (FALL 1)
                U-PS_K(e) := {{(p-l, t)|(p, t) ∈ UPS}|UPS ∈ U-PS}
            else
                UPS := max-ps(z)  \\ (FALL 3)
                \\ Die Prozedur „max-ps(z)“ bildet das maximale Pfadsystem
                \\ MAX-PS_K(e) für den Schaltkreis, dessen einziger Ausgang von
                \\ der Zelle z getrieben wird.
                U-PS_K(e) := {UPS}
                \\ Das Pfadsystem UPS wird in U-PS_K(e) aufgenommen.
            fi
        fi
    fi
```

if ($typ(z) \in \{\textsc{Fan}, \textsc{Not}\}$)
then
    $l_1 :=$ Leitung mit $z = \mathcal{Z}(l_1)$
    $z_1 := \mathcal{Q}(l_1)$
    $\mathcal{U}\text{-}\mathcal{PS} :=$ berechne_UPS(z_1, l_1)
    $\mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}}(e) := \{\{(p\text{-}l, t) | (p, t) \in UPS\} | UPS \in \mathcal{U}\text{-}\mathcal{PS}\}$
fi
if ($typ(z) = \textsc{Pi}$)
then
    $UPS :=< (l, \overline{e(l)} \rightarrow e(l)) >$
    $\mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}}(e) := \{UPS\}$
fi
return($\mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$)

Bemerkung 4.1:
Die Vorgehensweise in Algorithmus 4.1 ist für die Fälle 1 und 2 dieselbe wie bei der Berechnung der Menge $\mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$. Der Unterschied liegt in der Bearbeitung von Fall 3 (mehrere Inputs des betrachteten Gatters g haben kontrollierende Werte bezüglich der Eingabe e):
Im Algorithmus 4.1 wird das maximale Pfadsystem des Teilschaltkreises aufgebaut, dessen Ausgangsleitung gleich dem Ausgang von g ist. Dieses maximale Pfadsystem ist das *einzige* Element, das für diesen Teilschaltkreis in $\mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ aufgenommen wird. Beim Aufbau der Menge $\mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ hätte an dieser Stelle der Fall auftreten können, daß exponentiell viele Minimums-Pfadsysteme in die Menge aufgenommen werden müssen (siehe Beweis des Satzes 3.2, Fall 3).

Bemerkung 4.2:
Bei der Berechnung von Einzelpfadsystemen (siehe Definition 3.9, Seite 63) in Algorithmus 4.1 treten nur Fall 1 und 2 auf. Die Vorgehensweise bei der Konstruktion der Menge $\mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ ist für diese Pfade dieselbe wie in Algorithmus 4.1. Somit sind in der Menge $\mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ zu einer Eingabe e dieselben Einzelpfadsysteme enthalten, die auch in der Menge $\mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ zu dieser Eingabe enthalten sind.

Zur besseren Unterscheidung wird im folgenden ein Minimums-Pfadsystem in das Klammerpaar „[,]“, ein umfassendes Pfadsystem in das Klammerpaar „<, >“ eingeschlossen. Eine steigende Flanke ($0 \rightarrow 1$) wird mit $\uparrow$, eine fallende Flanke ($1 \rightarrow 0$) mit $\downarrow$ abgekürzt.

Beispiel 4.2:
Wir betrachten den Schaltkreis $\mathcal{K}$ aus Abbildung 4.5. Die Pfade des Schaltkreises seien wie folgt bestimmt:

$$
\begin{array}{ll}
p_1 = l_1\text{-}l_{15} & p_2 = l_2\text{-}l_{10}\text{-}l_{13}\text{-}l_{15} \\
p_3 = l_3\text{-}l_4\text{-}l_{10}\text{-}l_{13}\text{-}l_{15} & p_4 = l_3\text{-}l_5\text{-}l_{11}\text{-}l_{13}\text{-}l_{15} \\
p_5 = l_6\text{-}l_{11}\text{-}l_{13}\text{-}l_{15} & p_6 = l_7\text{-}l_{12}\text{-}l_{14}\text{-}l_{15} \\
p_7 = l_8\text{-}l_{12}\text{-}l_{14}\text{-}l_{15} & p_8 = l_9\text{-}l_{14}\text{-}l_{15}
\end{array}
$$

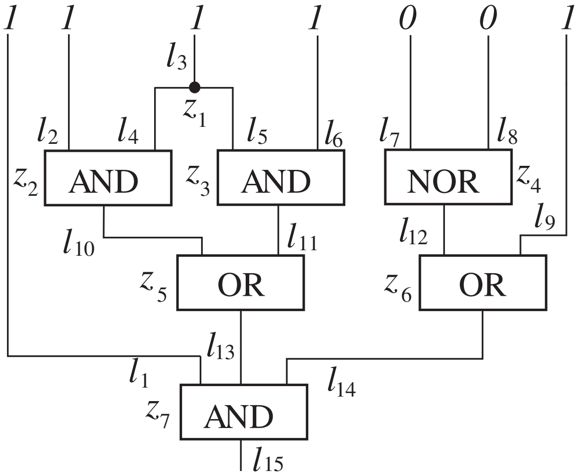

Abbildung 4.5: Beispielschaltkreis $\mathcal{K}$

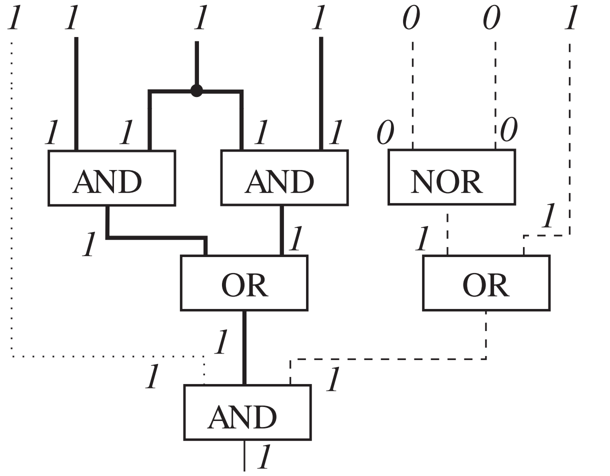

Abbildung 4.6: Belegung für e an $\mathcal{K}$

Es wird die Eingabe $e = (1111001)$ an den Schaltkreis angelegt. Die Belegung der Leitungen aus $\mathcal{K}$ für Eingabe e ist in Abbildung 4.6 gegeben. $\mathcal{K}_i,\ i = 1, \ldots, 15$, bezeichne den Teilschaltkreis von $\mathcal{K}$, der in Leitung l_i endet. $\mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}_i}$ bezeichne die Menge der umfassenden Pfadsysteme zur Eingabe e bezüglich des Teilschaltkreises $\mathcal{K}_i$.

An allen Inputleitungen der Zelle z_7 liegen bezüglich der Eingabe e nur nicht-kontrollierende Werte an. Es folgt aus Algorithmus 4.1, Fall 2:

$$\mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}}(e) = (\mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}_1} \cup \mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}_{13}} \cup \mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}_{14}}) \circ z_7$$

Da der Teilschaltkreis $\mathcal{K}_1$ nur aus einer einzigen Leitung besteht, die mit einem primären Input verbunden ist (in Abbildung 4.6 punktiert eingezeichnet), gilt:

$$\mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}_1} = \{< (l_1, \uparrow) >\}$$

An den Inputs der Zelle z_5 liegen bezüglich der Eingabe e zwei kontrollierende Werte an. Im Algorithmus, Fall 3, wird das maximale Pfadsystem des Teilschaltkreises mit Ausgangsleitung l_{13} zurückgeliefert (in Abbildung 4.6 dicker eingezeichnet).

$$
\begin{aligned}
\mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}_{13}} &= \{\text{MAX-PS}_{\mathcal{K}_{13}}\} \\
&= \{< (l_2\text{-}l_{10}\text{-}l_{13}, \uparrow), (l_3\text{-}l_4\text{-}l_{10}\text{-}l_{13}, \uparrow), (l_3\text{-}l_5\text{-}l_{11}\text{-}l_{13}, \uparrow), (l_6\text{-}l_{11}\text{-}l_{13}, \uparrow) >\}
\end{aligned}
$$

Auch an Zelle z_6 liegen wieder zwei kontrollierende Werte an, so daß hier das maximale Pfadsystem für den Teilschaltkreis mit Ausgangsleitung l_{14} zurückgeliefert wird (in Abbildung 4.6 gestrichelt eingezeichnet).

$$\begin{aligned} \mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}_{14}} &= \{\text{MAX-PS}_{\mathcal{K}_{14}}\} \\ &= \{< (l_7\text{-}l_{12}\text{-}l_{14}, \downarrow), (l_8\text{-}l_{12}\text{-}l_{14}, \downarrow), (l_9\text{-}l_{14}, \uparrow) >\} \end{aligned}$$

Insgesamt folgt für $\mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$:

$$\begin{aligned} \mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}}(e) &= \{< (p_1, \uparrow) > \;,\; < (p_2, \uparrow), (p_3, \uparrow), (p_4, \uparrow), (p_5, \uparrow) > \;, \\ &\quad < (p_6, \downarrow), (p_7, \downarrow), (p_8, \uparrow) >\} \end{aligned}$$

Für die Eingabe e ist die Menge der Minimums-Pfadsysteme $\mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ wie folgt gegeben:

$$\begin{aligned} \mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e) &= \{[(p_1, \uparrow)] \;,\; [(p_2, \uparrow), (p_4, \uparrow)] \;,\; [(p_2, \uparrow), (p_5, \uparrow)] \;,\; [(p_3, \uparrow), (p_4, \uparrow)] \;, \\ &\quad [(p_3, \uparrow), (p_5, \uparrow)] \;,\; [(p_6, \downarrow), (p_8, \uparrow)] \;,\; [(p_7, \downarrow), (p_8, \uparrow)]\} \end{aligned}$$

Vergleicht man die Mengen $\mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ und $\mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$, fällt auf, daß die Einzelpfadsysteme aus $\mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ ebenfalls auch in $\mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ enthalten sind. Der Unterschied zwischen den Mengen ergibt sich bei den Pfadsystemen, die mehr als einen logischen Pfad enthalten: Für mehrere Pfadsysteme aus $\mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ wird nur ein umfassendes Pfadsystem in $\mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ aufgebaut. Liegen an einem Gatter bezüglich der Eingabe e mehrere kontrollierende Werte an, so wird bei der Berechnung von $\mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ nur *ein* maximales Pfadsystem zurückgeliefert. Bei der Konstruktion von $\mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ müssen hingegen die Kombinationen aus den Pfadsystemen der zurückgelieferten Mengen gebildet werden (Fall 3 im Beweis zu Satz 3.2). Anstelle der sieben Elemente der Menge $\mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ enthält die Menge $\mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ nur noch drei Elemente. ∎

Satz 4.4:
Sei $\mathcal{K}$ ein unate Schaltkreis mit genau einem primären Ausgang und e eine Eingabe an $\mathcal{K}$. Die Menge der umfassenden Pfadsysteme $\mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ zur Eingabe e an $\mathcal{K}$ sei durch Algorithmus 4.1 berechnet. Dann gilt für das worst-case-Delay der Eingabe e:

$$d_{\mathcal{K}}(e) = \max\{d_{\mathcal{K}}(UPS) \,|\, UPS \in \mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}}(e)\}$$

Beweis:
Der Beweis erfolgt durch vollständige Induktion über die Anzahl der Gatter des Schaltkreises $\mathcal{K}$.

INDUKTIONSANFANG: $\mathcal{K}$ enthält keine Gatter, d.h. $\mathcal{K}$ enthält genau eine Leitung l.
$\mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ enthält genau ein Pfadsystem mit dem logischen Pfad $(l, \overline{e} \rightarrow e)$, d.h.

$$\mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}}(e) = \{< (l, \overline{e} \rightarrow e) >\}$$

INDUKTIONSSCHRITT: $\mathcal{K}$ enthält $\alpha > 0$ Gatter.
Sei l die Leitung, die mit dem primären Ausgang von $\mathcal{K}$ verbunden ist. Sei g das Gatter, dessen Ausgang mit der Leitung l verbunden ist.
Sei o.B.d.A. g ein AND-Gatter mit den zwei Input-Leitungen l_1 und l_2 (für die anderen Fälle analog). Die Teilschaltkreise, die mit den Leitungen l_1 und l_2 verbunden sind, werden mit $\mathcal{K}_1$ und $\mathcal{K}_2$ bezeichnet.
$d_{\mathcal{K}_1}(e)$ und $d_{\mathcal{K}_2}(e)$ bezeichnen das (worst-case) Delay der Eingabe e bezüglich der Schaltkreise $\mathcal{K}_1$ und $\mathcal{K}_2$. Die beschriebene Situation ist in Abb. 4.7 dargestellt.

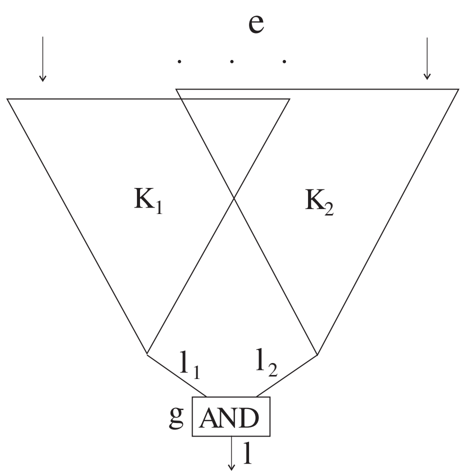

Abbildung 4.7: Struktur des Schaltkreises $\mathcal{K}$

Da die Gatteranzahlen dieser Schaltkreise kleiner als α sind, kann induktiv angenommen werden, daß Satz 4.4 für diese Teilschaltungen erfüllt ist, d.h. für die Pfadsystemmengen $\mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}_1}(e)$ und $\mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}_2}(e)$ gilt:

$$\begin{aligned} d_{\mathcal{K}_1}(e) &= \max\{d_{\mathcal{K}_1}(UPS) | UPS \in \mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}_1}(e)\} \ \ \textit{und} \\ d_{\mathcal{K}_2}(e) &= \max\{d_{\mathcal{K}_2}(UPS) | UPS \in \mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}_2}(e)\}. \end{aligned} \tag{4.4}$$

Es seien:

$$\begin{aligned} \mathcal{U}\text{-}\mathcal{PS}_1(e) &:= \{UPS \circ g | \, UPS \in \mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}_1}(e)\} \\ \mathcal{U}\text{-}\mathcal{PS}_2(e) &:= \{UPS \circ g | \, UPS \in \mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}_2}(e)\}. \end{aligned} \tag{4.5}$$

In Abhängigkeit von den Inputwerten des Gatters g bezüglich der Eingabe e sind 3 Fälle zu unterscheiden:

<u>Fall 1:</u> Ein Inputwert an g ist kontrollierend (logisch Null), der andere nicht-kontrollierend (logisch Eins).
O.B.d.A. sei der Wert an l_1 kontrollierend bezüglich e.
Für das Delay der Eingabe gilt somit:

$$d_{\mathcal{K}}(e) = d_{\mathcal{K}_1}(e) + d_{\downarrow,l_1}(g) \tag{4.6}$$

Aus den Formeln (4.4), (4.5) und (4.6) folgt:

$$\begin{aligned} d_{\mathcal{K}}(e) &= \max\{d_{\mathcal{K}_1}(UPS) \mid UPS \in \mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}_1}(e)\} + d_{\downarrow,l_1}(g) \\ &= \max\{d_{\mathcal{K}_1}(UPS) + d_{\downarrow,l_1}(g) \mid UPS \in \mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}_1}(e)\} \\ &= \max\{d_{\mathcal{K}}(UPS') \mid UPS' \in \mathcal{U}\text{-}\mathcal{PS}_1(e)\} \end{aligned}$$

In Algorithmus 4.1 wird an der mit (FALL 1) markierten Stelle die Menge der umfassenden Pfadsysteme des Teilschaltkreises, dessen Ausgang die kontrollierende Inputleitung des Gatters ist, zurückgeliefert und um die Ausgangsleitung des Gatters verlängert. Die hierbei erzeugte Menge $\mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ entspricht der Menge $\mathcal{U}\text{-}\mathcal{PS}_1(e)$, so daß unsere Delayformel für $\mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ erfüllt ist.

<u>Fall 2:</u> Beide Inputwerte an g sind nicht-kontrollierend.
Es gilt:

$$d_{\mathcal{K}}(e) = \max\{d_{\mathcal{K}_1}(e) + d_{\uparrow,l_1}(g)\ ,\ d_{\mathcal{K}_2}(e) + d_{\uparrow,l_2}(g)\} \tag{4.7}$$

Aus den Formeln (4.4), (4.5) und (4.7) folgt:

$$\begin{aligned} d_{\mathcal{K}}(e) &= \max\{\max\{d_{\mathcal{K}_1}(UPS) \mid UPS \in \mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}_1}(e)\} + d_{\uparrow,l_1}(g)\ , \\ &\qquad \max\{d_{\mathcal{K}_2}(UPS) \mid UPS \in \mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}_2}(e)\} + d_{\uparrow,l_2}(g)\ \} \\ &= \max\{\max\{d_{\mathcal{K}}(UPS') \mid UPS' \in \mathcal{U}\text{-}\mathcal{PS}_1(e)\}\ , \\ &\qquad \max\{d_{\mathcal{K}}(UPS') \mid UPS' \in \mathcal{U}\text{-}\mathcal{PS}_2(e)\}\} \\ &= \max\{d_{\mathcal{K}}(UPS') \mid UPS' \in \mathcal{U}\text{-}\mathcal{PS}_1(e) \cup \mathcal{U}\text{-}\mathcal{PS}_2(e)\} \end{aligned}$$

In Algorithmus 4.1 wird an der mit (FALL 2) markierten Stelle die Menge der umfassenden Pfadsysteme für alle Teilschaltkreise, deren Outputs den Inputleitungen des Gatters g entsprechen, zurückgeliefert, in einer einzigen Menge vereinigt und um die Ausgangsleitung des Gatters verlängert. Die hierbei erzeugte Menge $\mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ entspricht der Vereinigung der Mengen $\mathcal{U}\text{-}\mathcal{PS}_1(e)$ und $\mathcal{U}\text{-}\mathcal{PS}_2(e)$, d.h. $\mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}}(e) = \mathcal{U}\text{-}\mathcal{PS}_1(e) \cup \mathcal{U}\text{-}\mathcal{PS}_2(e)$. Somit ist die Delayformel für $\mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ erfüllt.

<u>Fall 3:</u> Beide Inputwerte an g sind kontrollierend.
Sei MAX-PS$_{\mathcal{K}}(e)$ das maximale Pfadsystem von $\mathcal{K}$ für Eingabe e. Es gilt nach Satz 4.2:

$$d_{\mathcal{K}}(e) = d_{\mathcal{K}}(\text{MAX-PS}_{\mathcal{K}}(e))$$

In Algorithmus 4.1 wird an der mit (FALL 3) markierten Stelle das maximale Pfadsystem des Schaltkreises $\mathcal{K}$ zurückgeliefert. Dieses maximale Pfadsystem, ist das *einzige* Pfadsystem, das in der Menge $\mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ enthalten ist. Somit folgt:

$$\begin{aligned} d_{\mathcal{K}}(e) &= d_{\mathcal{K}}(\text{MAX-PS}_{\mathcal{K}}(e)) \\ &= \max\{d_{\mathcal{K}}(UPS') \mid UPS' \in \mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}}(e)\} \end{aligned} \qquad \blacksquare$$

Mit Hilfe von Satz 4.4 kann nun angegeben werden, welche umfassenden Pfadsysteme zu testen sind, um für eine Eingabe e zu zeigen, daß das worst-case-Delay von e kleiner oder gleich einer Laufzeitschranke τ ist.

Korollar 4.5:
Sei $\mathcal{K}$ ein unate Schaltkreis mit genau einem primären Ausgang, e Eingabe an $\mathcal{K}$ und $\mathcal{U}$-$\mathcal{PS}_{\mathcal{K}}(e)$ wie in Algorithmus 4.1 bestimmt, dann gilt:

$$d_{\mathcal{K}}(UPS) \leq \tau \; \forall\, UPS \in \mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}}(e) \implies d_{\mathcal{K}}(e) \leq \tau$$

4.1.3 Zu testende umfassende Pfadsysteme für den Gesamtschaltkreis

Wie schon erwähnt, ist die folgende Vorgehensweise analog zu der des letzten Kapitels, der Vollständigkeit halber werden die Sätze nochmals wiederholt.

Aus Kapitel 2, Lemma 2.11, wissen wir, daß es ausreicht, nur die essentiellen Eingaben aus $\mathrm{E}(f)$ zu testen, um einen vollständigen Laufzeittest für den Schaltkreis zu erhalten. Nach Korollar 4.5 kann das Delay dieser Eingaben überprüft werden, indem man das Delay der umfassenden Pfadsysteme aus den zugehörigen umfassenden Pfadsystemmengen überprüft.
Hat man also das Delay der umfassenden Pfadsystemmengen aller Eingaben aus $\mathrm{E}(f)$ getestet, kann man schlußfolgern, daß der Schaltkreis rechtzeitig schaltet.

Satz 4.6:
Sei $\mathcal{K}$ ein unate Schaltkreis mit genau einem primären Ausgang. Wir definieren:

$$\mathcal{U}\text{-}\mathcal{PS}(\mathcal{K}) := \bigcup_{e \in \mathrm{E}(f)} \mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$$

Dann gilt:

$$d_{\mathcal{K}}(UPS) \leq \tau \; \forall\, UPS \in \mathcal{U}\text{-}\mathcal{PS}(\mathcal{K}) \implies d(\mathcal{K}) \leq \tau$$

Beweis:
Analog zum vorigen Kapitel, siehe Satz 3.5. ∎

Eine Menge von umfassenden Pfadsystemen, die sich äquivalent zu $\mathcal{MIN}$-$\mathcal{PS}_{min}(\mathcal{K})$ verhält, läßt sich nicht wie in Satz 3.6 herleiten. Ist ein kleines umfassendes Pfadsystem

Teilmenge eines größeren, so kann nicht gefolgert werden, daß das größere umfassende Pfadsystem für den Test des Schaltkreises unnötig ist. Dies liegt daran, daß das Delay eines umfassenden Pfadsystems im allgemeinen nicht dem Minimum der Delays der im System enthaltenen Pfade entspricht. Folglich ist auch ein Schluß wie bei den Minimums-Pfadsystemen nicht möglich.

Beispiel 4.3:
Man betrachte das umfassende Pfadsystem aus Beispiel 4.2 mit:

$$UPS =< (p_6, \downarrow), (p_7, \downarrow), (p_8, \uparrow) >$$

Hat man das umfassende Pfadsystem $< (p_6, \downarrow) >$ bezüglich einer Laufzeitschranke τ überprüft, kann man nicht schlußfolgern, daß auch das Delay des Pfadsystems UPS kleiner oder gleich τ ist, da gilt:

$$d_{\mathcal{K}}(UPS) = \min\{\max\{d_{\mathcal{K}}(p_6, \downarrow), d_{\mathcal{K}}(p_7, \downarrow)\}, d_{\mathcal{K}}(p_8, \uparrow)\}$$

Es kann also nicht geschlußfolgert werden, daß das Pfadsystem UPS für den Tests des Schaltkreises unnötig ist. ∎

Bemerkung 4.3:
Man sieht leicht ein, daß die Menge der *nicht-überdeckbaren umfassenden Pfadsysteme* für einen vollständigen Delaytest des Schaltkreises ausreichen. D.h.:

Sei

$$\mathcal{U}\text{-}\mathcal{PS}_{n\ddot{u}}(\mathcal{K}) := \{UPS \in \mathcal{U}\text{-}\mathcal{PS}(\mathcal{K}) |\ \nexists\ Einzelpfadsysteme\ UPS_1, \ldots, UPS_n \in \mathcal{U}\text{-}\mathcal{PS}(\mathcal{K}) : \\ (UPS_i \neq UPS, \forall i = 1, \ldots, n) \ \wedge\ (\textstyle\bigcup_{i=1}^{n} UPS_i = UPS)\}$$

Dann gilt:

$$d_{\mathcal{K}}(UPS) \leq \tau\ \forall\, UPS \in \mathcal{U}\text{-}\mathcal{PS}_{n\ddot{u}}(\mathcal{K}) \implies d(\mathcal{K}) \leq \tau.$$

(Beweisskizze:
Sei PS ein Pfadsystem. Es gilt:

$$d_{\mathcal{K}}(PS) \leq \max\{d_{\mathcal{K}}(p,t) | (p,t) \in PS\}$$

Hieraus kann geschlußfolgert werden:

$$d_{\mathcal{K}}(p,t) \leq \tau\ \forall (p,t) \in PS \implies d_{\mathcal{K}}(PS) \leq \tau$$

Durch den Test der Einzelpfadsysteme wird also das Überprüfen des Pfadsystems PS für den Test des Schaltkreises unnötig.)

4.2 Test der umfassenden Pfadsysteme

Nachdem die Menge der zu testenden umfassenden Pfadsysteme $\mathcal{U}\text{-}\mathcal{PS}_{n\ddot{u}}(\mathcal{K})$ bestimmt ist, kann die eigentliche Testmengenreduktion vorgenommen werden. Gesucht wird eine minimale Teilmenge der FDT, die alle Pfadsysteme aus $\mathcal{U}\text{-}\mathcal{PS}_{n\ddot{u}}(\mathcal{K})$ überdeckt. Das Testmengenminimierungsproblem stellt sich damit wieder als Überdeckungsproblem dar (siehe auch Paragraph 3.3).
Das folgende Korollar gibt an, welche Pfadsysteme durch eine Testsequenz der funktionalen Delaytestmenge überprüft werden.

Korollar 4.7:
Sei $\mathcal{K}$ ein unate Schaltkreis, f die von $\mathcal{K}$ berechnete Funktion und e eine Eingabe an $\mathcal{K}$ mit: $e \in \mathrm{E}(f)$. Funktioniert $\mathcal{K}$ korrekt für die Eingabesequenz $< init(e), e >$ innerhalb der Laufzeitschranke τ, dann gilt:

$$d_{\mathcal{K}}(UPS) \leq \tau \;\forall\, UPS \in \mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$$

Beweis:
Analog zu vorigem Kapitel, siehe Korollar 3.7. ■

Es kann nun der zentrale Satz dieses Paragraphen angegeben werden. Er ermöglicht es, die funktionale Testmenge auf den gegebenen Schaltkreis $\mathcal{K}$ zuzuschneiden.

Satz 4.8:
Sei $\mathcal{K}$ ein unate Schaltkreis, f die von $\mathcal{K}$ berechnete Funktion und E eine Menge von Eingaben an $\mathcal{K}$ mit:

- $E \subseteq \mathrm{E}(f)$
- $\bigcup\limits_{e \in E} \mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}}(e) \supseteq \mathcal{U}\text{-}\mathcal{PS}_{n\ddot{u}}(\mathcal{K})$

Die Eingabesequenzen $\{< init(e), e > \mid e \in E\}$ sind ein vollständiger Delaytest für den Schaltkreis $\mathcal{K}$.

Beweis:
Der Schaltkreis $\mathcal{K}$ funktioniert korrekt für alle Testsequenzen aus $\{< init(e), e > \mid e \in E\}$

$$\stackrel{Korollar 4.7}{\Longrightarrow} \forall e \in E \text{ gilt: } d_{\mathcal{K}}(UPS) \leq \tau \;\forall\, UPS \in \mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$$

Es gilt nach Voraussetzung:

$$
\begin{aligned}
& \mathcal{U}\text{-}\mathcal{PS}_{n\ddot{u}}(\mathcal{K}) \subseteq \bigcup_{e\in E} \mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}}(e) \\
\Longrightarrow \quad & d_{\mathcal{K}}(UPS) \leq \tau \ \forall\, UPS \in \mathcal{U}\text{-}\mathcal{PS}_{n\ddot{u}}(\mathcal{K}) \\
\stackrel{Bem.4.3}{\Longrightarrow} \quad & d(\mathcal{K}) \leq \tau
\end{aligned}
$$

■

Gesucht ist eine minimale Teilmenge der FDT, deren umfassende Pfadsystemmengen alle umfassenden Pfadsysteme aus $\mathcal{U}\text{-}\mathcal{PS}_{n\ddot{u}}(\mathcal{K})$ enthalten. Das Testmengenreduktionsproblem kann wieder auf das Überdeckungsproblem zurückgeführt werden (zur Veranschaulichung siehe Paragraph 3.3).

4.3 Praktische Resultate

In diesem Paragraphen werden die Ergebnisse, die mit der Methodik dieses Kapitels erhalten wurden, mit den Ergebnissen des letzten Kapitels verglichen. Zur Vereinfachung werden im folgenden die Testmengen, bei denen Minimums-Pfadsyteme aufgebaut werden, mit MIN-RDT, die Testmengen, bei denen umfassende Pfadsysteme aufgebaut werden, mit U-RDT bezeichnet.

In den Tabellen 4.1 und 4.2 wird die U-RDT mit der MIN-RDT aus Kapitel 3 verglichen. In Tabelle 4.1 sind wieder Ergebnisse für ISCAS89-Benchmarks, in Tabelle 4.2 für MCNC-Benchmarks aufgeführt.

Spalte 1 der Tabellen enthält den Namen der jeweils betrachteten Schaltung. Spalte 2 gibt zum besseren Vergleich nochmals die Größe der FDT an. Spalte 3 enthält die Größe der MIN-RDT, Spalte 6 die Größe der U-RDT . Es fällt auf, daß die Größe der MIN-RDT im allgemeinen besser ist. Dies liegt daran, daß die Vorgehensweise beim Testen der umfassenden Pfadsysteme grober ist:

> Nehmen wir an, daß in $\mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ zu einer Eingabe n Minimums-Pfadsysteme vorhanden sind. In $\mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ zu dieser Eingabe seien diese n Pfadsysteme zu einem einzigen umfassenden Pfadsystem zusammengeschlossen. Die Pfadsysteme aus $\mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ seien aus der Menge $\mathcal{MIN}\text{-}\mathcal{PS}_{min}(\mathcal{K})$, das Pfadsystem aus $\mathcal{U}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ sei aus der Menge $\mathcal{U}\text{-}\mathcal{PS}_{n\ddot{u}}(\mathcal{K})$.
> Um die n Systeme aus $\mathcal{MIN}\text{-}\mathcal{PS}_{\mathcal{K}}(e)$ mit den Pfadystemmengen anderer Eingaben zu überdecken, können n unterschiedliche Eingaben herangezogen werden.
> Damit beim Lösen des Überdeckungsproblems für die umfassenden Pfadsysteme die Eingabe e gestrichen werden kann, muß genau das gleiche umfassende Pfadsystem in der umfassenden Pfadsystemmenge einer anderen Eingabe vorhanden sein.

Name	# FDT	# MIN-RDT	# ≥ 2	Zeit	# U-RDT	# ≥ 2	Zeit
cs27	30	24	0	1	24	0	1
cs298	252	136	1	2	136	2	2
cs344	5596	224	6	23	228	5	24
cs349	5596	222	6	23	226	5	23
cs382	1180	302	0	5	302	0	5
cs400	1174	306	47	6	306	8	5
cs444	1174	382	50	6	382	11	6
cs510	1256	322	0	6	322	0	6
cs526	620	318	1	3	318	2	3
cs641	10178	636	26	1:39	636	8	1:39
cs713	10926	?	?	?	2994	2353	4:29
cs820	1790	324	0	9	324	0	9
cs832	1770	336	0	9	336	0	9
cs953	4130	504	0	30	504	0	30
cs1196	36648	1480	229	6:40	1494	42	7:01
cs1238	31878	1384	433	5:18	1518	136	5:55
cs1488	3144	756	0	17	756	0	17
cs1494	3062	754	0	17	754	0	17

Tabelle 4.1: U-RDT für ISCAS89-Benchmarks

Bei allen Schaltkreisen, die mit beiden Heuristiken bearbeitbar sind, ergibt sich eine durchschnittliche Vergrößerung der MIN-RDT zur U-RDT von 9,1 %. Als extremstes Beispiel sei hier der Schaltkreis $table5$ genannt, bei dem die U-RDT fast viermal so groß wie die MIN-RDT ist (MIN-RDT: 2666, U-RDT: 10590). Trotz dieser größeren Testmengen hat die U-RDT einen entscheidenden Vorteil: Es können viel größere Schaltkreise bearbeitet werden als bei der MIN-RDT. Tritt in der Spalte 3 ein Fragezeichen auf, bedeutet dies, daß bei diesen Schaltkreisen bei der Berechnung der MIN-RDT entweder der Speicher übergelaufen ist, oder daß die Reduktion nicht mehr in akzeptabler Zeit durchführbar war. Die U-RDT ist für diese Schaltkreise noch berechenbar, da hier die Anzahl der berechneten Pfadsysteme wesentlich kleiner ist. Die Anzahl der nötigen Minimums-Pfadsysteme ($\mathcal{MIN}$-$\mathcal{PS}_{min}(\mathcal{K})$) ist in Spalte 4 angegeben, die Anzahl der nötigen umfassenden Pfadsysteme ($\mathcal{U}$-$\mathcal{PS}_{nü}(\mathcal{K})$) in Spalte 7. Zum besseren Vergleich sind hier die Einzelpfadsysteme weggelassen, da sie für beide Mengen gleich sind. Wie stark sich der Unterschied zwischen Minimums-Pfadsystemen und umfassenden Pfadsystemen auswirken kann, ist z.B. am Schaltkreis $Z5xp1$ ersichtlich: Hier enthält die Menge $\mathcal{MIN}$-$\mathcal{PS}_{min}(\mathcal{K})$ 89189 „echte" Minimums-Pfadsysteme, die Menge $\mathcal{U}$-$\mathcal{PS}_{nü}(\mathcal{K})$ hingegen nur 386 „echte" umfassende Pfadsysteme. Allerdings treten auch Beispiele auf, bei denen die Anzahl der Elemente aus $\mathcal{U}$-$\mathcal{PS}_{nü}(\mathcal{K})$ die Anzahl der Elemente aus $\mathcal{MIN}$-$\mathcal{PS}_{min}(\mathcal{K})$ leicht übersteigt (z.B. Schaltkreis $ex5$). Dies liegt daran, daß bei der Bildung der Menge $\mathcal{U}$-$\mathcal{PS}_{nü}(\mathcal{K})$ „unnötige"

Name	# FDT	# MIN-RDT	# ≥ 2	Zeit	# U-RDT	# ≥ 2	Zeit
Z5xp1	1084	702	89189	6:05:00	760	386	13
Z9sym	2132	256	0	27	256	0	27
alu4	94120	?	?	?	9888	9901	55:08
apex1	40970	1982	205	15:58	2210	89	16:53
apex2	9600	522	328	1:28	570	52	1:30
apex3	16548	2662	1647	7:10	2690	892	7:22
apex5	9990	1790	2384	2:08	1818	189	2:10
b12	186	118	17	1	120	13	1
bw	702	?	?	?	636	752	10
c5xp1	918	576	4305	1:55	592	156	7
c9sym	4788	362	15	1:26	362	14	1:26
clip	3348	630	147	22	766	111	22
con1	52	26	0	1	26	0	1
cps	16414	2264	45	2:45	2274	55	2:46
duke2	9446	812	345	2:12	852	177	2:16
e64	132	132	0	25	132	0	25
ex4	1468	228	0	26	228	0	26
ex5	3294	1352	100	45	1406	222	44
inc	786	448	561	6	466	122	5
misex1	186	156	157	2	158	155	2
misex2	238	142	3	2	142	7	2
misex3	31944	?	?	?	6964	5947	16:30
misex3c	72542	?	?	?	26234	49605	15:00:44
pdc	8144	836	25	1:02	840	31	1:03
rd53	490	164	8	1	172	8	1
rd73	5148	912	5579	4:46	976	210	27
rd84	11266	1046	492	1:21	1074	119	1:22
sao2	1472	592	2629	3:34	690	360	14
spla	6656	1306	96	1:01	1362	145	1:03
squar5	216	144	204	1	148	16	1
t481	13938	50	0	1:00	50	0	32
table5	81146	2666	3207	37:59	10590	4556	43:15
vg2	2426	242	0	19	242	0	20

Tabelle 4.2: U-RDT für MCNC-Benchmarks

umfassenden Pfadsysteme vollständig durch Einzelpfadsysteme überdeckt werden müssen und sich somit mehr umfassende Pfadsysteme als „notwendig“ erweisen.
Die Spalten 5 und 8 bieten schließlich noch einen Vergleich der Laufzeiten. Ein klarer

Laufzeitvorteil wird z.B. bei Schaltkreis $Z5xp1$ erzielt (MIN-RDT: >6 h, U-RDT: 13 sec). Daß auch die U-RDT eine starke Verringerung der FDT darstellt, macht ein Vergleich der Spalten 2 und 6 klar. Interessant sind hierbei vor allem Schaltkreise, bei denen die MIN-RDT nicht berechnet werden konnte. Für Schaltkreis $alu4$ konnte die FDT zum Beispiel auf fast ein Zehntel der ursprünglichen Größe reduziert werden.

Kapitel 5

Transformation von allgemeinen Schaltkreisen in unate Schaltkreise

Die von uns betrachteten Testmengen haben die Eigenschaft, daß sie nur für unate Schaltkreise gültig sind. Für allgemeine Schaltkreise kann folgender Trick angewandt werden: Liegt ein allgemeiner binate Schaltkreis vor, so kann dieser Schaltkreis durch einen unate Schaltkreis simuliert werden. Der resultierende unate Schaltkreis kann dann mit Hilfe unserer Testmengen vollständig auf Delayfehler überprüft werden.
Eine Möglichkeit einen beliebigen Schaltkreis in einen unate Schaltkreis zu transformieren, stellt das in diesem Kapitel beschriebene *Paritätspushen* dar. Dieser Algorithmus verallgemeinert ein Verfahren aus [Red73]. Bei der Schaltkreistransformation durch Paritätspushen wird die Anzahl der Gatter und die Anzahl der Inputs maximal verdoppelt. Die Tiefe des Schaltkreises wird durch die Transformation nicht verändert. Da auch das Fanout eines Fanoutknotes durch den Algorithmus nicht zunehmen kann, wird das Delay des Schaltkreises nicht erhöht.

5.1 Die Schaltkreistransformation

Im folgenden werden wir nur eine Teilmenge der Zellen des Schaltkreises betrachten, nämlich alle Zellen mit Ausnahme der Fanoutknoten.

Definition 5.1:
Sei $\mathcal{K}$ ein kombinatorischer Schaltkreis.

(a) *Eine Zelle z aus $\mathcal{K}$ hat die Inversionsparität 0 (Inversionsparität 1) $\Longleftrightarrow$*
Jeder Pfad von z zu einem primären Output von $\mathcal{K}$ hat die Inversionsparität[1] 0 (1).

[1] Für die Inversionsparität eines Pfades siehe Kapitel 1, Definition 1.7.

(b) Ein Zelle z aus $\mathcal{K}$ hat die Inversionsparität $1/0 \iff$
Es existiert ein Pfad von z zu einem primären Ausgang von $\mathcal{K}$, der die Inversionsparität 0 hat und es existiert ein Pfad von z zu einem primären Ausgang von $\mathcal{K}$, der die Inversionsparität 1 hat.

Beispiel 5.1 :
Sei ein Schaltkreis $\mathcal{K}$ wie in Abbildung 5.1 gegeben.

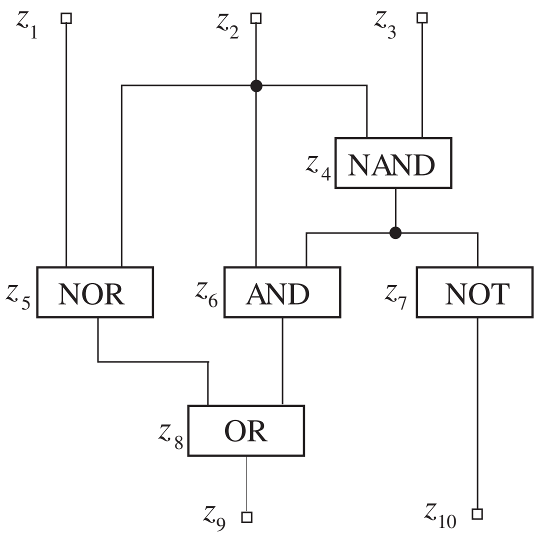

Abbildung 5.1: Beispielschaltkreis $\mathcal{K}$

Die Inversionsparitäten für die Zellen aus $\mathcal{K}$ sind wie folgt bestimmt:

$$\begin{array}{lcl@{\qquad}lcl}
\text{Inversionsparität}(z_1) & = & 1 & \text{Inversionsparität}(z_2) & = & 1/0 \\
\text{Inversionsparität}(z_3) & = & 1/0 & \text{Inversionsparität}(z_4) & = & 1/0 \\
\text{Inversionsparität}(z_5) & = & 0 & \text{Inversionsparität}(z_6) & = & 0 \\
\text{Inversionsparität}(z_7) & = & 0 & \text{Inversionsparität}(z_8) & = & 0 \\
\text{Inversionsparität}(z_9) & = & 0 & \text{Inversionsparität}(z_{10}) & = & 0
\end{array}$$

■

Enthält ein Schaltkreis keine Zellen mit 1/0 Inversionsparität, so ist der Schaltkreis unate. Ziel ist es also einen Schaltkreis zu konstruieren, dessen Zellen nur noch 1 oder 0 als Inversionsparität haben.

<u>Gegeben</u>: Ein kombinatorischer Schaltkreis $\mathcal{K}$ mit Inputs $x_1, \ldots, x_n$. Seien $x_1, \ldots, x_k$ die Inputs mit 1/0 Inversionsparität.

<u>Ziel</u>: Konstruiere einen unate Schaltkreis $\mathcal{K}'$ mit:

- $\mathcal{K}'$ hat Inputs $x_1, x_1', \ldots, x_k, x_k', x_{k+1}, \ldots, x_n$
- $\mathcal{K}'$ simuliert $\mathcal{K}$ vermöge

$$\mathcal{K}(e_1, \ldots, e_n) = \mathcal{K}'(e_1, e_1, \ldots, e_k, e_k, e_{k+1}, \ldots, e_n)\ \forall (e_1, \ldots, e_n) \in \boldsymbol{B}^n$$

Eine Möglichkeit, zu einem beliebigen Schaltkreis einen unate Schaltkreis zu konstruieren, der den ursprünglichen Schaltkreis simuliert, ist das *Paritätspushen.* In unserem Algorithmus werden alle Zellen mit 1/0 Inversionsparität verdoppelt. Eine Kopie treibt alle Pfade mit Inversionsparität 0, die andere treibt alle Pfade mit Inversionsparität 1. Durch diese Verdopplung enthält der resultierende Schaltkreis keine Zellen mit 1/0 Inversionsparität und ist somit unate. Wir bearbeiten die Zellen des Schaltkreises ausgehend von den primären Ausgängen in umgekehrter topologischer Sortierung.

<u>Algorithmus</u>: Paritätspushen
In einem ersten Schritt werden für alle Zellen des Schaltkreises die Inversionsparitäten berechnet.
Danach wird im zweiten Schritt des Algorithmus der Schaltkreis $\mathcal{K}$ schrittweise in einen unate Schaltkreis $\mathcal{K}'$ transformiert. Hierbei geht man induktiv startend bei den primären Outputs des Schaltkreises vor. Wird eine Zelle vom Algorithmus bearbeitet, so wurden alle Nachfolger schon bearbeitet.

<u>INDUKTIONSANFANG:</u> Sei z ein primärer Ausgang des Schaltkreises.
$\Longrightarrow$ Keine Veränderung des Schaltkreises.

<u>INDUKTIONSSCHRITT</u>: Sei z eine beliebige Zelle des Schaltkreises.
Induktiv sei der von den bereits bearbeiteten Zellen aufgespannte Schaltkreis schon unate, d.h. alle Zellen dieses Schaltkreises haben entweder Inversionsparität 0 oder 1.

In Abhängigkeit von der nächsten zu bearbeitenden Zelle z sind zwei Fälle zu unterscheiden:

<u>FALL 1</u>: z hat keine 1/0 Inversionsparität.
$\Longrightarrow$ Keine Veränderung des Schaltkreises.

<u>FALL 2</u>: z hat 1/0 Inversionsparität.
Seien $z_1, \ldots, z_m$ die Zellen, mit denen der Ausgang von z verbunden ist. Da der Schaltkreis unterhalb von z schon unate ist, hat jedes der z_i entweder Inversionsparität 0 oder 1. Sei o.B.d.A. z eine Zelle mit 2 Inputs. Die mit den Inputs von z verbundenen Zellen werden mit $\tilde{z}_1$ und $\tilde{z}_2$ bezeichnet. Die eben beschriebene Situation ist in Abbildung 5.2 dargestellt.

Sei

$$\begin{aligned} Z_0 &:= \{z_i \mid z_i \text{ hat Inversionsparität } 0\} \text{ und} \\ Z_1 &:= \{z_i \mid z_i \text{ hat Inversionsparität } 1\}. \end{aligned}$$

Verdopple z zu z' und z''. Verbinde die Inputs von z' und z'' analog zu den Inputs von z. Verbinde den Ausgang von z' mit allen Elementen aus Z_0. Verbinde den Ausgang von z''

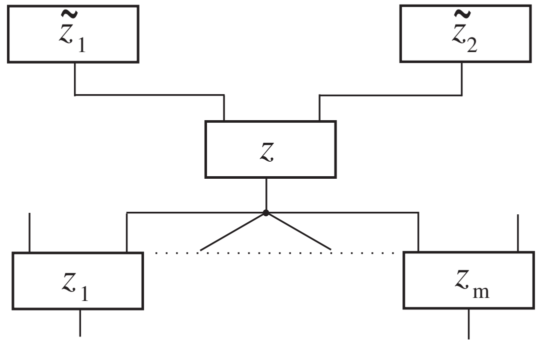

Abbildung 5.2: Situation vor Verdopplung von z

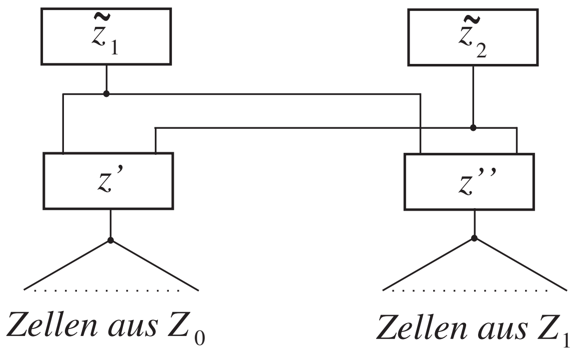

Abbildung 5.3: Verdopplung von z

mit allen Elementen aus Z_1. Die eben beschriebene Vorgehensweise ist in Abbildung 5.3 dargestellt.

$\Longrightarrow$ Es gilt nach Konstruktion:

z' hat Inversionsparität 0 und z'' hat Inversionsparität 1.

Man überlegt sich leicht, daß der durch den Algorithmus berechnete Schaltkreis $\mathcal{K}'$ den ursprünglichen Schaltkreis $\mathcal{K}$ vermöge

$$\mathcal{K}(e_1, \ldots, e_n) = \mathcal{K}'(e_1, e_1, \ldots, e_k, e_k, e_{k+1}, \ldots, e_n) \; \forall (e_1, \ldots, e_n) \in \boldsymbol{B}^n$$

simuliert.

Im folgenden Beispiel soll der eben beschriebene Algorithmus verdeutlicht werden:

Beispiel 5.2:
Sei ein binate Schaltkreis wie in Abbildung 5.4 gegeben. In einem ersten Schritt sind die Inversionsparitäten der Zellen des Schaltkreises zu bestimmen. Die Zellen z_2, z_3 und z_4 haben eine 1/0 Inversionsparität. Im zweiten Schritt des Algorithmus müssen diese Zellen somit verdoppelt werden. Die Zellen des Schaltkreises erhalten eine topologische Numerierung. Unser Algorithmus startet bei der Zelle mit der größten Nummer, also am primären Ausgang (z_9). Die Zellen $z_8, \ldots, z_5$ bleiben unverändert, da sie keine 1/0 Inversionsparität haben. Die erste zu verdoppelnde Zelle ist die Zelle z_4.

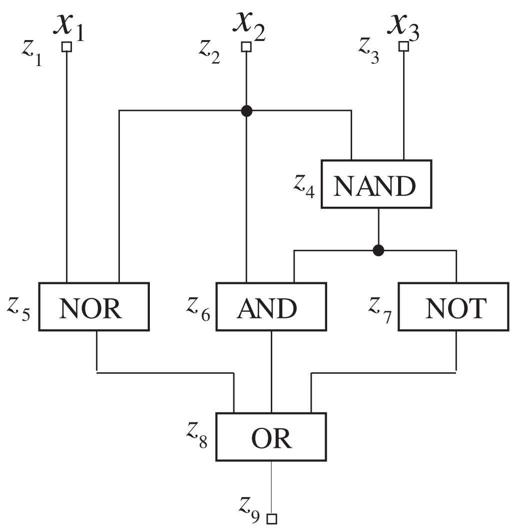

Abbildung 5.4: Binate Schaltkreis $\mathcal{K}$

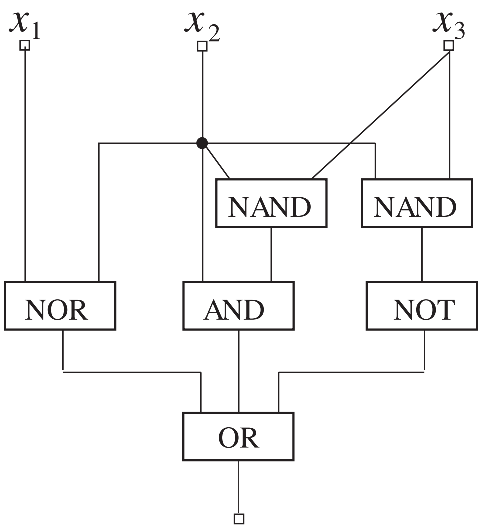

Abbildung 5.5: Verdopplung des NAND

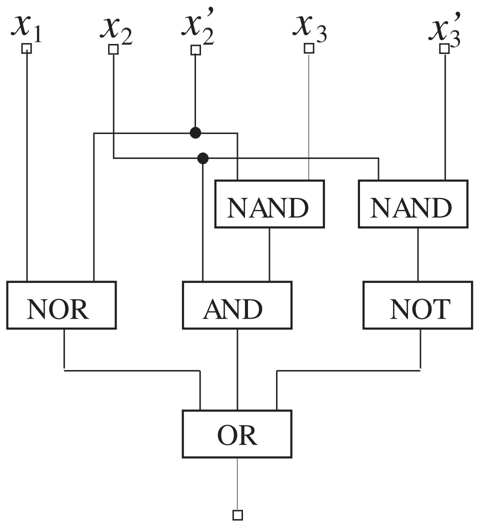

Abbildung 5.6: Unate Schaltkreis $\mathcal{K}'$

Diese Verdopplung von z_4 (NAND-Gatter) wird in Abbildung 5.5 veranschaulicht. Der Ausgang der einen Kopie wird mit allen Zellen verbunden, die mit dem ursprünglichen NAND-Gatter verbunden waren und die Inversionsparität 0 haben. Die zweite Kopie des NAND-Gatters wird entsprechend mit den Zellen aus der Menge Z_1 verbunden. Die Einänge der beiden Kopien werden mit den Gattern verbunden, mit denen das ursprüngliche NAND-Gatter verbunden war.
Entsprechend werden auch die Zellen z_3 und z_2 (primäre Inputs des Schaltkreises) verdoppelt.

In Abbildung 5.6 ist der unate Schaltkreis $\mathcal{K}'$ dargestellt, der den ursprünglichen binate Schaltkreis $\mathcal{K}$ simuliert. ■

5.2 Praktische Resultate

Der in Kapitel 5.1 vorgestellte Algorithmus zum Paritätspushen wurde implementiert und auf die allgemeinen ISCAS89- und MCNC-Benchmarkschaltkreise angewandt (siehe auch Kapitel 2.4). In den Tabellen 5.1 und 5.2 wird der binate Schaltkreis mit dem unate Schaltkreis verglichen, der den binate Schaltkreis simuliert.

Spalte 1 enthält wiederum den Namen der Schaltung. In Spalte 2 sind die Kosten des originalen Schaltkreises aufgeführt. In Spalte 3 sind die Kosten für den durch Paritätspushen erzeugten unate Schaltkreis dargestellt. Hierbei wird folgendes Kostenmaß angewandt:

$$\begin{aligned} Kosten(\textsc{Not}) &= \frac{1}{2} \\ Kosten(\textsc{Nand}, \textsc{Nor}) &= \frac{1}{2} * \#Inputs \\ Kosten(\textsc{And}, \textsc{Or}) &= \frac{1}{2} + \frac{1}{2} * \#Inputs \end{aligned}$$

Da nicht-invertierende Zellen in der Praxis meist aus invertierenden Grundbausteinen aufgebaut werden (z.B. AND= NAND+ NOT), wird diesen Zelltypen ein höheres Kostenmaß zugeordnet. In Spalte 4 wird die prozentuale Kostenerhöhung von Spalte 2 nach Spalte 3 angegeben.

Betrachtet man diese Prozentzahlen, bemerkt man, daß die Schaltkreisgröße beim Paritätspushen keinesfalls ständig verdoppelt wird. Vielmehr bleiben die Kosten für die betrachteten Benchmarkschaltkreise meist in erträglichem Rahmen (Kostensteigerung $< 30\%$). Man kann daraus schließen, daß große Teile der Schaltkreise in unate Form vorliegen. Man muß allerdings anmerken, daß die von uns betrachteten Schaltkreise meist Kontrollogik darstellen. Bei arithmetischen Schaltkreisen wäre die Kostensteigerung höher (siehe z.B. $alu4$).
Es sei hier nochmals erwähnt, daß die Tiefe des Schaltkreises durch Paritätspushen nicht verändert wird. Das Delay der Schaltung bleibt gleich oder wird geringer (falls Fanoutknoten aufgespalten werden).

Name	Kosten(alt)	Kosten(neu)	Steigerung(%)
cs27	10,5	18,5	76,2
cs298	145,5	163	12,0
cs344	161	200,5	24,5
cs349	163,5	204	24,8
cs382	170,5	204,5	20,0
cs400	178	219	23,0
cs444	189,5	230,5	21,6
cs510	243,5	258,5	6,2
cs526	264,5	289,5	9,5
cs641	321	444,5	38,5
cs713	351	545	55,3
cs820	446,5	453	1,5
cs832	455,5	462	1,4
cs953	414	434	4,8
cs1196	614	820,5	33,6
cs1238	643,5	858	33,3
cs1488	968,5	1014	4,7
cs1494	975,5	1021	4,7

Tabelle 5.1: Paritätspushen bei ISCAS89-Benchmarks

Name	Kosten(alt)	Kosten(neu)	Steigerung(%)
Z5xp1	107	194,5	81,8
Z9sym	215,5	222	3,0
alu4	181	331,5	83,2
apex1	1104	1179	6,8
apex2	225,5	255,5	13,3
apex3	1357	1619	19,3
apex5	663,5	825	24,3
b12	74	95,5	29,1
bw	127	209,5	65,0
c5xp1	98,5	169	71,6
c9sym	226,5	251,5	11,0
clip	129	192	48,8
con1	17,5	19,5	11,4
cps	970,5	1087	12,0
duke2	345	415	20,3
e64	174,5	175	0,3
ex4	663,5	692,5	4,4
ex5	413,5	472	14,2
inc	91	154,5	69,8
misex1	46	82,5	79,4
misex2	83,5	109,5	31,1
misex3	559	840	50,3
misex3c	417,5	739,5	77,1
pdc	335	419	25,1
rd53	34	59	73,5
rd73	62	107,5	73,4
rd84	130	200,5	54,2
sao2	128,5	218,5	70,0
spla	535	607	13,5
squar5	49	85	73,5
t481	30,5	57,5	88,5
table5	735	845	15,0
vg2	84	87,5	4,2

Tabelle 5.2: Paritätspushen bei MCNC-Benchmarks

Anhang A

Primimplikanten einer unate Funktion

Lemma A.1:
Sei f eine unate Funktion, die positiv unate in den Variablen $x_1, \ldots, x_k$ und negativ unate in den Variablen $x_{k+1}, \ldots, x_n$ ist. Sei $PI(f)$ die Menge der Primimplikanten von f.

> *Jeder Primimplikant aus $PI(f)$ enthält die Variablen $x_1, \ldots, x_k$ nur nichtnegiert und die Variablen $x_{k+1}, \ldots, x_n$ nur negiert.*

Beweis:
Sei x_i eine Eingabevariable zu f.

1. x_i ist positiv unate.
   Annahme: $p \in PI(f)$ enthält $\overline{x_i}$, d.h. $p = p' \wedge \overline{x_i}$.
   Sei $\alpha = \alpha_1 \ldots \alpha_{i-1} 0 \alpha_{i+1} \ldots \alpha_n$ eine Belegung, die p erfüllt.
   Dann gilt für $\alpha' = \alpha_1 \ldots \alpha_{i-1} 1 \alpha_{i+1} \ldots \alpha_n$:
   $$f(\alpha') \geq f(\alpha)$$
   Also:
   $$f(\alpha) = 1 \implies f(\alpha') = 1$$
   Somit ist p' ebenfalls Implikant von f, was ein Widerspruch zu $p \in PI(f)$ ist.

2. x_i ist negativ unate.
   Annahme: $p \in PI(f)$ enthält x_i, d.h. $p = p' \wedge x_i$.
   Sei $\alpha = \alpha_1 \ldots \alpha_{i-1} 1 \alpha_{i+1} \ldots \alpha_n$ eine Belegung, die p erfüllt.
   Dann gilt für $\alpha' = \alpha_1 \ldots \alpha_{i-1} 0 \alpha_{i+1} \ldots \alpha_n$:
   $$f(\alpha') \geq f(\alpha)$$

Also:

$$f(\alpha) = 1 \implies f(\alpha') = 1$$

Somit ist p' ebenfalls Implikant von f, was ein Widerspruch zu $p \in PI(f)$ ist.

■

Literaturverzeichnis

[Ake73] S.B. Akers. Universal test sets for logic networks. *IEEE Transactions on Computers*, C-22:835–839, June 1973.

[BBK89] F. Brglez, D. Bryan, and K. Kozminski. Combinational profiles for sequential benchmark circuits. In *International Symposium on Circuits and Systems*, pages 1929–1934, 1989.

[Bet71] R. Betancourt. Derivation of minimum test sets for unate logic circuits. *IEEE Transactions on Computers*, C-20:1264–1269, November 1971.

[BHMSV84] R.K. Brayton, G.D. Hachtel, C.T. McMullen, and A.L. Sangiovanni-Vincentelli. *Logic Minimization Algorithms for VLSI Synthesis.* The Kluwer International Series in Engineering and Computer Science. Kluwer Academic Publishers, 1984.

[CC93] K.-T. Cheng and H.-C. Chen. Delay testing for non-robust untestable circuits. In *International Test Conference*, pages 954–961, 1993.

[CL94] B. Chen and C.L. Lee. A complement-based fast algorithm to generate universal test sets for multi-output functions. *IEEE Transactions on Computer-Aided Design*, CAD-13:370–377, 1994.

[DWB92] K. De, C. Wu, and P. Banerjee. Reliability driven logic synthesis of multilevel circuits. In *Proceedings of the 1992 International Symposium on Circuits and Systems*, pages 1105–1108, 1992.

[EW78] E.B. Eichelberger and T.W. Williams. A logic design structure for LSI testability. *Journal of Design Automation and Fault-Tolerant Computation*, 2:165–178, 1978.

[GD83] N. Goncalves and H. DeMan. NORA: A racefree dynamic CMOS technique for pipelined logic structures. *IEEE Journal of Solid State Circuits*, 1983.

[GJ88] G. Gupta and N.K. Jha. A universal test set for CMOS circuits. *IEEE Transactions on CAD*, 7:590–597, 1988.

[HGDT84] L.G. Heller, W.R. Griffin, J.W. Davis, and N.G. Thoma. Cascode voltage switch logic: A differential CMOS logic family. In *IEEE International Solid State Circuits Conference*, 1984.

[Hot74] G. Hotz. *Schaltkreistheorie*. de Gruyter Lehrbuch. Walter de Gruyter Verlag, 1974.

[Hot90] G. Hotz. *Einführung in die Informatik*. Leitfäden und Monographien der Informatik. B.G. Teubner Verlag, Stuttgart, 1990. 545 Seiten.

[KLL82] R.H. Krambeck, C.M. Lee, and H.-F.S. Law. High-speed compact circuits with CMOS. *IEEE Journal of Solid State Circuits*, 1982.

[KM94] W. Ke and P.R. Menon. Synthesis of delay-verifiable two-level circuits. In *European Design and Test Conference*, pages 297–301, 1994.

[LR87] C.J. Lin and S.M. Reddy. On delay fault testing in logic circuits. *IEEE Transactions on CAD*, pages 694–703, September 1987.

[LSBSV93] W.K. Lam, A. Saldanha, R.K. Brayton, and A.L. Sangiovanni-Vincentelli. Delay fault coverage and performance tradeoffs. In *30th Design Automation Conference*, pages 446–451, 1993.

[Mic94] G. De Micheli. *Synthesis and Optimization of digital circuits*. Mc Graw-Hill, 1994.

[Mil85] B. Milne. Testability, 1985 technology forecast. *Electronic Design*, pages 143–166, 1985.

[PDY92] V. Peng, D.R. Donchin, and Y.-T. Yen. Design methodology and CAD tools for the NVAX microprocessor. In *Proceedings of the IEEE Int. Conf. on Computer Design*, 1992.

[Red73] S.M. Reddy. Complete test sets for logic functions. *IEEE Transactions on Computers*, C-22:1016–1020, November 1973.

[Red92] L.N. Reddy. *Compact Test Sets for Digital Logic Circuits*. PhD thesis, University of Iowa, Dept. of Elec. and Comp. Engr, August 1992.

[RLP87] S.M. Reddy, C.J. Lin, and S. Patil. An automatic test pattern generator for the detection of path delay faults. In *International Conference on CAD*, pages 284–287, 1987.

[SLCR95] U. Sparmann, D. Luxenburger, K.T. Cheng, and S.M. Reddy. Fast identification of robust dependent path delay faults. In *Proceedings of the 32nd Design Automation Conference*, pages 119–125, 1995.

[SMR96a] U. Sparmann, H. Müller, and S.M. Reddy. Minimal delay test sets for unate gate networks. In *Proceedings of the 5th Asian Test Symposium*, 1996.

[SMR96b] U. Sparmann, H. Müller, and S.M. Reddy. Universal delay test sets for logic networks. *to appear in: IEEE Transactions on VLSI Systems*, 1996. (Preliminary version available as technical report 5-30-94, Department of ECE, University of Iowa, Iowa City, IA 52242, USA).

[SR94] U. Sparmann and S.M. Reddy. Universal delay test sets for logic networks. Technical Report 5-30-94, Department of ECE, University of Iowa, Iowa City, Iowa 52242, USA, 1994. (To appear in IEEE Transactions on VLSI Systems).

[SSM+92] E. Sentovich, K. Singh, C. Moon, H. Savoj, R. Brayton, and A. Sangiovanni-Vincentelli. Sequential circuit design using synthesis and optimization. In *International Conference on Computer Design*, pages 328–333, 1992.

[Wun91] H.-J. Wunderlich. *Hochintegrierte Schaltungen: Prüfgerechter Entwurf und Test*. Springer-Verlag, 1991.

[Yan91] S. Yang. Logic synthesis and optimization benchmarks user guide (version 3.0). Technical report, Microelectronics Center of North Carolina, P.O. Box 12889, Research Triangle Park, NC 27709, January 1991.