Strukturtreue Modellierung anhand von Signalflussgraphen

M. Denguir, G. Uygur, S. Sattler, Friedrich-Alexander-Universität, 91052 Erlangen, Deutschland
B. Cella, M. Schmidt, Liebherr, 88131 Lindau, Deutschland
T. Egelhofer, B. Scheffold, Rohde-und-Schwarz, 87700 Memmingen, Deutschland

Kurzfassung

Die Absicherung der funktionalen Sicherheit von Schaltungen und Systemen geschieht kostengünstig mit Hilfe der strukturtreuen Verifikation. Dabei werden die einer realen Struktur zugehörigen Funktionen strukturtreu modelliert. Für jede modellierte Funktion wird die gerichtete Gleichheit zwischen formal hergeleiteter Funktion und der durch die reale Struktur erzeugten Funktion garantiert. Zusätzlich wird sichergestellt, dass die modellierte Funktion ein zur realisierten Funktion nicht zu unterscheidendes gleichwertiges (äquivalentes) Verhalten aufweist. Der Beitrag stellt erstmalig eine strukturtreue Verifikationsmethode anhand von Signalflussgraphen vor. Diese wird mit Hilfe einer digitalen Schaltung plausibilisiert, die Ergebnisse in Signalflussgraphen dargestellt und mit Hilfe einer günstigen Datenkompaktierung abgespeichert. Später soll die Methode mit bekannten, selbst aktivierten Fehlern überprüft werden.

Abstract

The guarantee of the functional safety of circuits and systems goes on cost-effective with the help of verification with structural integrity ensured. The functions associated with a real structure are modelled in a structure-oriented manner. For every modelled function, the directed equality between formally derived function and by the real structure produced function is ensured. In addition, it is guaranteed that the modelled function has a homogeneous (equivalent) behaviour, which is structural equal to the realized function. For the first time, this article introduces a structure loyal verification method using signal flow graphs. This is made plausible with the aid of a digital circuit, the results are displayed in signal flow graphs and are stored through a favourable data compaction. Known, self-activated errors shall later verify the method.

1 Einführung

Simulationsbasierte Validierungsmethoden sind das am häufigsten verwendete Tool zur Validierung komplexer Schaltungen und Systeme. Da jedoch Design und Technik immer mehr an ihre Grenzen stoßen, ist die simulationsbasierte Validierung nicht in der Lage, alle möglichen Szenarien abzudecken, was eine nicht akzeptable Grundlage ist, insbesondere für sicherheitskritische Systeme [1]. Es ist daher bekannt, dass weit verbreitete Simulationstechniken nicht ausreichen, um die Richtigkeit komplexer Systeme sicherzustellen [2]. Die Alternative besteht darin, theoretisch fundierte formale Verifikationsinstrumente einzusetzen [3], wodurch die genaue Zuordnung von virtueller Funktionalität und realer Struktur erfolgt. Es ist somit sinnvoll, eine robuste und effektive Verifikationsmethode zu verwenden, mit deren Hilfe die Funktionalität von komplizierten und gerichteten realen Systemen getestet werden kann. Diese Methode nennen wir die „Strukturtreue Modellierung anhand von Signalflussgraphen“.

2 Strukturtreue Modellierung

Die Modellierung realer Systeme mit Hilfe von Funktionen (mit Hilfe der Beschreibung des Verhaltens), welche die Struktur bewahren, führt auf eine Darstellung als Signalflussgraph (SFG). Ein SFG ist die Präsentation einer abstrakten Algebra, mit welcher strukturtreue Abbildungen verifiziert werden können. Der SFG ist dabei die gerichtete graphische Darstellung einer Vielfachmenge (Multi-Set), welche aus sogenannten Kanten und Knoten besteht, die Morphismen (z.B. Funktionen) und Objekte (z.B. Mengen) darstellen. Die aus Knoten und Kanten bestehenden Phasenlisten eines SFG sind voneinander unabhängig und ohne Beschränkung der Allgemeinheit (oBdA) zueinander nebenläufig (simultan parallel) [4].
Dieses Datenmodell in positiver Logik (PL) mit den Eigenschaften Assoziativität (asso) und Identität (id) stellt die Vorschrift für die Kodierung eines Steuerkreises [5] dar. Dieses Datenmodell spaltet sich auf in *Operational* (OP) und *Control* (CTRL) und soll für jede Teilstruktur Spezifikation (*spec*), Test (*test*), Funktionsfähigkeit (*k*) und Nicht-Funktionsfähigkeit (/*k*) garantieren.

2.1 Schritte hin zum Datenmodell

Die Erstellung des Datenmodells gliedert sich in drei Schritte: Zuerst (1) müssen die in der realen Struktur vorkommenden realen Pins jeweils mit einem positiven oder negativen Literal bezeichnet werden, dies entspricht der Einbettung der realen Struktur in ein Kodierungsuniversum. Dann (2) wird die reale Struktur eventbasiert abstrahiert und durch einen gerichteten Signalfluss mit Hilfe von zwei Pfaden (Dual-Rail) in einem Modell nachgebildet. Anschließend (3) werden die Zustandsübergänge der Teilmodelle in aussagenlogischen Ausdrücken (AA) beschrieben, ihre Signalflussgraphen (SFG) in drei-wertiger Logik

ISBN 978-3-8007-4444-2

(in sogenannten Ternär-Vektorlisten - TVL [6]) mit (1,0,-) kodiert und als Quaternär-Vektorliste (QVL) in (1,0,-,X) formatiert.

2.2 Regeln zur Erstellung des Modells

Der erste Schritt zur Erstellung des Datenmodells, das Benennen der Pins, lässt sich mit den „gerichteten" Gesetzen der jeweiligen Physik bzw. Elektrotechnik mit Hilfe von zwei Regeln kanalisieren. Erste und wichtigste Regel lautet: Pins entsprechend der Pin-Partitionierung [4] benennen, das sind Zustände (Z), Primary Inputs (PI) und Primary Outputs (PO). Zweite Regel lautet: Pins je nach transportiertem digitalen Signalwert (1 oder 0) eventbasiert verfeinern, negatives Literal für Signalwert = 0 (Low) und positives Literal für Signalwert = 1 (High). Dabei wird das negative Literal mit dem vorangestellten Symbol „/" gekennzeichnet.

Bezüglich des zweiten Schritts zur Erstellung des Datenmodells wird die Darstellung der „substituierbaren" Complementarität mittels der Operation Switch (¬) in Dual Rail durchgeführt [7] (siehe Kapitel 3.3).

Für die Durchführung des dritten Schritts zur Erstellung des Datenmodells, der Kodierung der Operationen (*spec*, *test*, *k*, */k*) in TVL, benötigen *k* und */k* Realitäten wie z.B. Bauteile und Leitungen, *spec* definierte Limits und *test* bekannte Fehlermodelle. Da sich jeder Zustand in einen Zustand Z^a (alt, present) und einen Zustand Z^n (neu, next) aufspaltet, d.h. $Z = (Z^a, Z^n)$, wird allgemein festgelegt, dass die Operationen *spec* und *test* von Z^a nach Z^n übergehen, während die Steuerungen *k* und */k* von Z^n nach Z^a übergehen. Sobald also *spec* als Operation (OP) erfüllt wird, behält ein Pin bzw. ein Zustand seinen Wert im Sinne einer Zustandsstabilisierung [8], $Z^n = Z^a$. Hingegen wird *test* als Operation (OP) erfüllt (das nennt man Fail), wechselt der Zustand in den gleichen Zustand mit negativem Literal, $Z^n = /Z^a$. Da *k* und */k* als Steuerungen (CTRL) von der jeweiligen Realität abhängen, ist es ungünstig eine allgemeine Formel für ihre Erfüllung zu entwickeln. Sie werden explizit oder impliziert modelliert (siehe Kapitel 3).

2.3 Verallgemeinertes Beispiel

Das gesuchte spezielle Datenmodell wird durch das folgende verallgemeinerte Beispiel erzeugt. Seien X und Y zwei reale Pins, die in einer Struktur vorhanden sind, dann ist ihr zugehöriger SFG (in PL) in **Bild** 1 dargestellt. Der SFG in **Bild** 1 lässt sich auf den SFG in **Bild** 2, $X = (X^a, X^n)$ und $Y = (Y^a, Y^n)$, reduzieren. In **Tabelle** 1 sind die Zustandsübergänge als Phasenlisten angegeben.

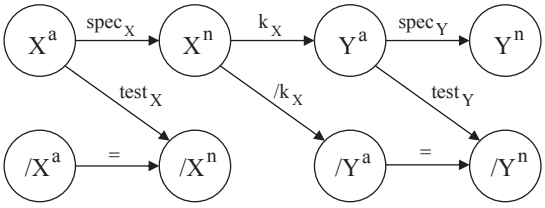

Bild 1 Verallgemeinertes Datenmodell (in PL)

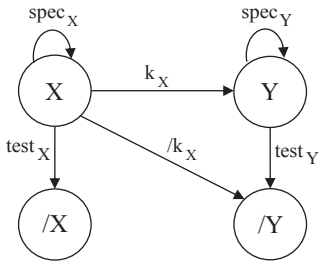

Bild 2 Verallgemeinertes reduziertes Datenmodell (in PL)

Z^a		Ereignis			Z^n		Funktion	
X	Y	L	R	F	X	Y	OP	CTRL
1	*	(1,0,-)	*	*	1	*	$spec_X$	*
*	1	*	(1,0,-)	*	1	*	*	k_X
1	*	*	*	(1,0,-)	0	*	$test_X$	*
*	0	*	(1,0,-)	*	1	*	*	$/k_X$
*	1	(1,0,-)	*	*	*	1	$spec_Y$	*
*	1	*	*	(1,0,-)	*	0	$test_Y$	*

Tabelle 1 Zustandsübergänge und Funktionen (in TVL)

Die Zustandsübergänge (Ereignisse) aus definierten Limits (L), Realität (R) und bekannten Fehlern (F) in **Tabelle** 1 sind in TVL kodiert. Das Sternsymbol „*" steht im gesamten Beitrag für undefiniert (NULL). Ist eine Listenstelle mit „*" belegt, heißt das, dass die Stelle nicht existiert.

3 Beispiel Digitale Schaltung

In diesem Kapitel soll eine reale digitale Schaltung (DUT - Device-Under-Test) mit Hilfe der „Strukturtreuen Modellierung anhand von SFG" für die Verifikation bekannter, selbstdefinierter Fehler vorbereitet werden. Wir betrachten dazu die digitale Schaltung in **Bild** 3, die bereits auf einer realen Leiterplatte vorliegt. Die Schaltung ist mit Jumpern bestückt, die uns als bekannte zu modellierende Fehler dienen. Das von uns verfolgte Ziel ist, die Verifikation anhand von absichtlich eingebauten Fehlern bzw. durch das Ein- und Ausstecken der besagten Jumper durchzuführen. Dazu wird eine automatische Testvorrichtung auf Grundlage eines µControllers entwickelt und programmiert. In diesem Beitrag beschränken wir uns auf den theoretischen Teil, den Bericht zur strukturtreuen Modellierung der Schaltung anhand von SFG.

3.1 Schematische Darstellung

Die digitale Schaltung (DUT) im **Bild** 3 stellt einen npn gesteuerten CMOS Inverter dar. Sie beinhaltet die vier Schalter S1, S2, S3 und S4, die als Verbindungsstecker (closed/open Jumper) dienen und manuell ein- bzw. ausgesteckt werden können. Um daher keine unerwünschten elektrischen Unterbrechungen oder Kurzschlüsse im Normalbetrieb zu haben, sind Schalter S1 und S4 geschlossen (closed), während Schalter S2 und S3 geöffnet (open) sind. Setzt man am Eingang (A_CTRL) eine digitale 1, wird der

ISBN 978-3-8007-4444-2

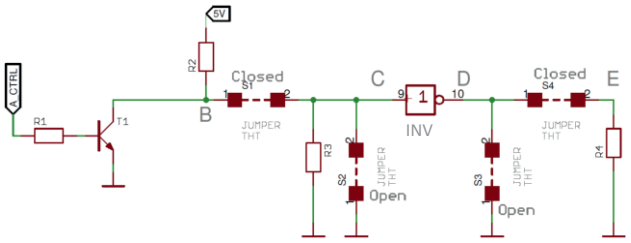

Bild 3 Schematische Darstellung des DUT

Transistor T1 leitend und der Pin B auf GND gezogen (digitale 0). Befinden sich Schalter S1 und S2 im Normalbetrieb, übernimmt Pin C diese digitale 0. Diese wird dann in Folge durch den Inverter (INV) invertiert und an Pin D als digitale 1 ausgegeben. Pin E am Ausgang übernimmt die digitale 1 von Pin D, falls die Schalter S3 und S4 auf Normalbetrieb geschaltet sind. Wird hingegen im Eingang eine digitale 0 angelegt, sperrt der Transistor T1, und Pin B übernimmt aufgrund des Spanungsteilers aus den ohmschen Widerständen R2 und R3 >> R2 die digitale 1. Diese wird durch den Inverter (INV) auf eine digitale 0 geschaltet und an den Ausgang E propagiert.

3.2 Bezeichnen der Pins

Der Transistor T1 in **Bild** 3 wird beim Anlegen einer digitalen 1 am Eingang (A_CTRL) leitend. Damit ist zu verstehen, dass der durch den npn gesteuerte CMOS Inverter so beschaltet wurde, dass mit dieser Eingangsbelegung eine digitale 1 am Ausgang getrieben werden kann. Daher muss, nach Berücksichtigung der Regeln in Kapitel 2.2, der Eingangs-Pin A_CTRL der npn Basis als positives Literal A bezeichnet und als Primary Input (PI) beschrieben werden. Folglich müssen, nach Betrachtung der Realität (Kapitel 3.1), die realen Pins B und damit auch C als negatives Literal, /B und /C, und die Pins D und E als positives Literal, D und E, bezeichnet werden. Es ergeben sich somit vier Pins bzw. Zustände (/B, /C, D, E) und ein Primary Input (A). Pin E ist sowohl ein Zustand als auch Primary Output (PO). Die Schalter werden bezeichnet mit positivem Literal, wenn im Normalbetrieb closed vorliegt, sonst mit negativem Literal. Damit folgt die Auflistung S1, /S2, /S3 und S4.

3.3 SFG in Dual Rail

Nachdem wir erfolgreich den ersten Schritt der Verifikationsmethode, das Bezeichnen der Pins, durchgeführt haben, können wir nun den zweite Schritt, die Realität in Event-basiert abstrahieren, anwenden, und einen SFG in Dual-Rail mit Hilfe der Operation Switch (¬) darstellen. Durch die Operation Switch werden aus den vier ursprünglichen Zuständen (/B, /C, D, E) die vier substituierbaren, zusätzlichen komplementären Zustände (¬/B, ¬/C, ¬D, ¬E) gewonnen. Auf jeden der acht Zustände wird die Funktion *test* angewandt, was dazu führt, dass acht substituierbare komplementäre Zustände erreicht werden können. Zusammenfassend ergeben sich aus den Zuständen /B, /C, D und E die Zustände B, C, /D und /E, und aus ¬/B, ¬/C, ¬D und ¬E die Zustände /¬/B, /¬/C, /¬D und /¬E. **Bild** 4 zeigt den SFG. Der Zustand (5V, /GND) im SFG stellt die Spannungsversorgung dar. Von Zustand (5V, /GND) zum Zustand /B und dual dazu zum Zustand ¬/B erfolgt durch die Funktionsfähigkeit $k_{/GND}$ und k_{5V}.

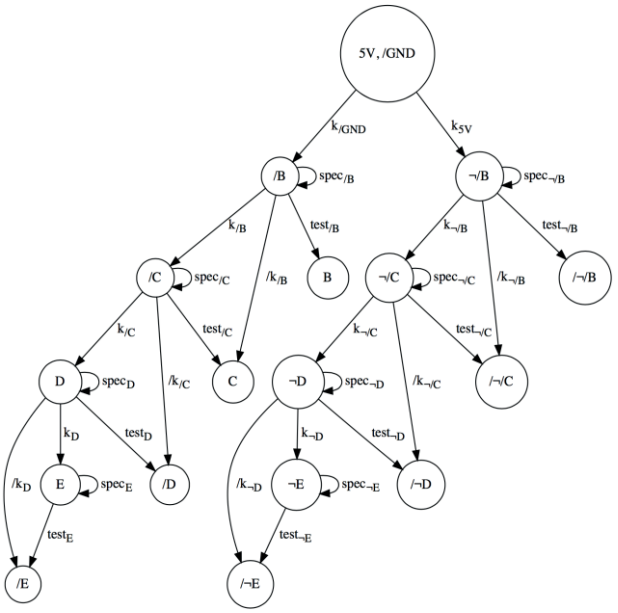

Bild 4 Datenmodell (SFG) des DUT

3.4 Erstellen der Teilmodelle (OP, CTRL)

3.4.1 Einbettung ins Kodierungsuniversum

Bevor wir im dritten Schritt zur Erstellung des Datenmodells mit der Kodierung der Zustandsübergänge *spec*, *test*, *k*, und */k* in TVL fortfahren, bilden wir die Schaltung in einer vollständigen Wertetabelle (mit allen Eingangskombinationen) ab. Es ergeben sich die folgenden sechs Wertetabellen (in TVL). Das Erstellen dieser sechs vollständigen Wertetabellen bzw. ihre Auflistung als TVL dienen der Kontrolle der Ergebnisse im dritten Schritt, welcher im Kapitel 3.5 durchgeführt wird.

/B	A	S1	/S2	/B
-	1	-	-	0
-	0	1	0	0
-	0	0	-	1
-	0	-	1	1

/B	S1	/S2	/C
0	-	-	0
1	0	-	0
1	-	0	0
1	1	1	1

/C	S1	/S2	/C
-	0	-	0
-	-	0	0
0	1	1	0
1	1	1	1

D	/S3	S4	D
1	1	-	1
1	0	-	0
0	-	-	0

D	/S3	S4	E
1	1	1	1
1	-	0	0
1	0	-	0
0	-	-	0

E	/S3	S4	E
1	1	1	1
1	-	0	0
1	0	-	0
0	-	-	0

Tabelle 2 Teilschaltungen als Wertetabellen (in TVL)

3.4.2 Kodierung der Funktionen

Folgende Kodierungstabellen **Tabelle** 3, 4, 5 und 6 zusammen mit ihren SFG dienen der genaueren Beschreibung der

gewichteten Kanten des SFG aus **Bild** 4 bzw. der spezifischen Kodierung der jeweiligen *Operational Functions spec* und *test*, und *Control Functions k* und /*k*. Dies geschieht nach Berücksichtigung der Regeln in Kapitel 2.2 analog zum Aufbau der **Tabelle** 1, $\delta(Z^aZ^n)$, in Abhängigkeit der Zustände (/B, /C, D, E), des Primary Inputs (A) und der Fehlermodelle (S1, /S2, /S3, S4).

In **Tabelle** 3 behält der Zustand /B seine digitale 0, falls die Funktion $spec_{/B}$ erfüllt wird. Dies geschieht ohne Einfluss der Fehler, d.h. (S1, /S2) = [∗ ∗], der Wert des Primary Inputs A ist auf die digitale 1 gesetzt. Falls die Funktion $test_{/B}$ erfüllt ist, wechselt der Zustand /B zur digitalen 1. Dies geschieht auf Grund der Fehler (A, S1, /S2) = [0 0 -] und (A, S1, /S2) = [0 - 1]. Bei der hier genannten ersten Kombination (S1, /S2) = [0 -] wird ein bekannter Fehler als vorhanden erkannt, der Schalter S2 ist geschlossen (elektrischer Kurzschluss). Im anderen Fall, (S1, /S2) = [- 1], könnte vorkommen (S1, /S2) = [1 1], was dazu führt, dass der bekannte Fehler korrekterweise als nicht-vorhanden erkannt wird. Dadurch erhält das Datenmodell alle Antworten zu bekannten, als vorhandene Fehler erkannten Fehler und zu bekannten, als nicht-vorhandene Fehler erkannten Fehler.

Da die Spannungsversorgung als stets vorhanden angenommen wird, gelten k_{5V} und $k_{/GND}$ als erfüllt und benötigen keine Kodierung (zugehöriger SFG zu **Tabelle** 3).

Zum anderen werden $k_{/B}$ und $/k_{/B}$ in **Tabelle** 4 ohne Betrachtung der Fehlermodelle explizit modelliert. Für $k_{/B}$ mit (/C, /B) = [0 0], falls beispielsweise das Bauelement R3 vorhanden ist, und für $/k_{/B}$ mit (/C, /B) = [1 0], falls beispielsweise gilt R3 << R2.

Die Kodierung der Funktionen $k_{/C}$ und $k_{\neg C}$ wird bereits durch $spec_D$ und $spec_{\neg D}$ mit (D, /C, D) = [1 0 1] und (D, /C, D) = [0 1 0] überdeckt. Dies sind die Kanten von /C nach D und von ¬/C nach ¬D im zugehörigen SFG in **Tabelle** 5. Hingegen werden k_D und $k_{\neg D}$ in **Tabelle** 6 explizit modelliert und halten die Ausgangsstufe D. Für die Kodierung weiterer Funktionen gilt analoges Vorgehen.

Die durchgeführten Kodierungen können nun wiederum mit den vollständigen Wertetabellen (**Tabelle** 2) verglichen werden. Nicht alle Funktionen bzw. Zustandsübergänge werden im SFG im **Bild** 4 kodiert. Zu keinem Mehrwert führen ($/k_{/C}$, $/k_D$, $/k_{\neg/B}$, $/k_{\neg/C}$, $/k_{\neg D}$), nie erfüllt werden ($test_{/C}$, $test_{\neg D}$, $test_{\neg E}$). Daher werden die Zustände /¬D und /¬E in **Bild** 4 nicht erreicht. Die Zusammenführung der zugehörigen SFG in **Tabelle** 3, 4, 5 und 6 ergibt den SFG in **Bild** 5.

3.5 Aufstellen der TVL in QVL

Die in Kapitel 3.4.2 erstellten TVL sollen nun als QVL aufgestellt und zusammengefasst werden. Dies geschieht, indem das Symbol „∗“ aus den **Tabelle** 3 bis 6, das für „undefiniert“ steht, durch das Symbol „X“ ersetzt wird. **Tabelle** 7 zeigt die Zusammenfassung der TVL in QVL.

QVL ist ein Format für die Realisierung der TVL. Analog zur **Tabelle** 1 wird die Funktion $\delta(Z^aZ^n)$ aufgebaut, Input und Schalter kodieren dabei δ. In **Tabelle** 7 gilt (s, t, n) = (spec, test, ¬).

/B	A	S1	/S2	/B	Funktion
0	1	∗	∗	0	$\mathbf{spec_{/B}}$
1	0	∗	∗	1	$\mathbf{spec_{\neg/B}}$
0	0	0	-	1	$\mathbf{test_{/B}}$
0	0	-	1	1	
1	0	1	0	0	$\mathbf{test_{\neg/B}}$
1	1	-	-	0	

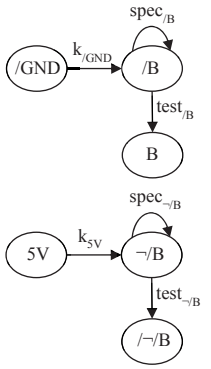

Tabelle 3 OP kodiert in /B

/C	S1	/S2	/B	Funktion
0	∗	∗	0	$\mathbf{k_{/B}}$
1	∗	∗	1	$\mathbf{k_{\neg/B}}$
1	∗	∗	0	$\mathbf{/k_{/B}}$

/C	S1	/S2	/C	Funktion
0	∗	∗	0	$\mathbf{spec_{/C}}$
1	∗	∗	1	$\mathbf{spec_{\neg/C}}$
1	0	-	0	$\mathbf{test_{\neg/C}}$

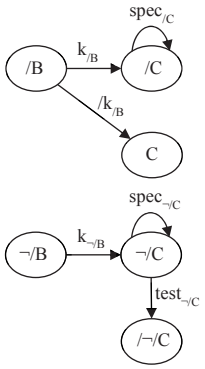

Tabelle 4 (CTRL, OP) kodiert in (/B, /C)

D	/S3	S4	/C	D	Funktion
1	∗	∗	0	1	$\mathbf{spec_D}$
0	∗	∗	1	0	$\mathbf{spec_{\neg D}}$
1	0	-	∗	0	$\mathbf{test_D}$

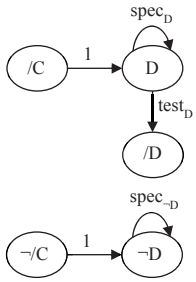

Tabelle 5 OP kodiert in D

D	/S3	S4	D	Funktion
1	∗	∗	1	$\mathbf{k_D}$
0	∗	∗	0	$\mathbf{k_{\neg D}}$

E	/S3	S4	E	Funktion
1	∗	∗	1	$\mathbf{spec_E}$
0	∗	∗	0	$\mathbf{spec_{\neg E}}$
1	-	0	0	$\mathbf{test_E}$
1	0	-	0	

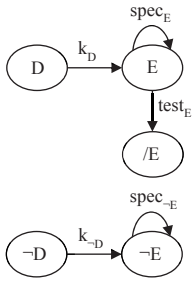

Tabelle 6 (CTRL, OP) kodiert in (D, E)

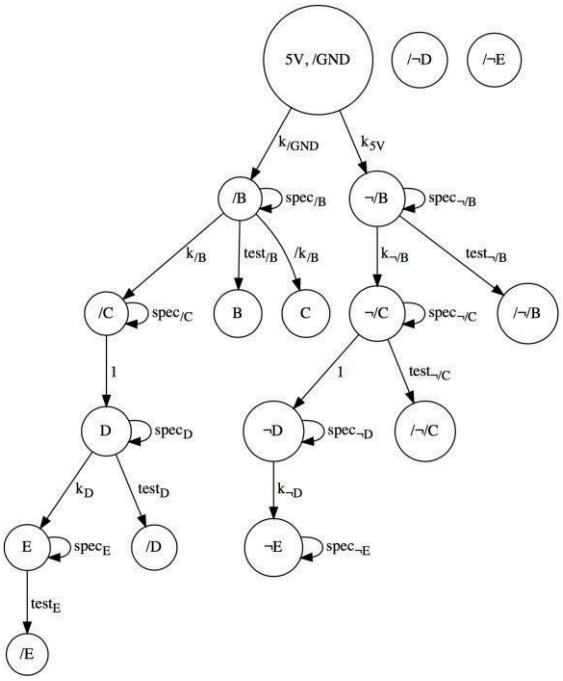

Bild 5 Datenmodell (SFG) des DUT

3.6 Anwendungsfälle

In diesem Kapitel werden nun für die vorgegebene, digitale Schaltung die zu erwartenden Ergebnisse der Verifikationsmethode vorgestellt. Über ein selbstverfasstes Programm in VBA (Visual Basic Application) können Ist-Werte in Tabellen mit den Soll-Werten aus **Tabelle** 7 verglichen werden. Das Programm visualisiert die entsprechenden SFG, die aus dem resultierenden Datenmodell **Bild** 5 mit Ausnahme der beiden nicht erreichten Zustände /¬D und /¬E stammen. Alle Phasenlisten (Kanten mit ihren Knoten) in diesem SFG sind oBdA als nebenläufig zu sehen. Dabei werden die Operationen *spec*, *test*, *k* und */k* - wenn sie erfüllt sind - farblich als grüne, rote, schwarze und blaue Pfeile dargestellt. Wenn sie nicht erfüllt werden, dienen gestrichelte Pfeile der Visualisierung.

Im ersten Fall (**Bild** 6) sollen alle Schalter S im Normalbetrieb laufen (die bekannten Fehler sind nicht vorhanden), erhalten also eine digitale 1 in der Ist-Tabelle. Input A hat eine digitale 1. Da die *spec* bzgl. /B, /C, D und E erfüllt ist, behält jeder Zustand (/B, /C, D, E) seinen Wert ($Z^n = Z^a$) und dazugehörige Kanten werden als grüner Pfeile dargestellt. Genauso gelten k bzgl. /B und D als erfüllt. Die restlichen Funktionen gelten aufgrund der unterschiedlichen Ist- und Soll-Werte als nicht erfüllt und sind dementsprechend gekennzeichnet.

Im zweiten Fall (**Bild** 7) wird Input A auf die digitale 0 gesetzt, was dazu führte, dass /B die digitale 1 übernimmt. Folglich ist *spec* bzgl. /B nicht mehr erfüllt, dafür *test*. Da jedoch alle Schalter im Normalbetrieb sind, werden bekannte Fehler als nicht-vorhandene Fehler erkannt.

In dritten Fall (**Bild** 8) wird S1 auf open gesetzt, er erhält also die digitale 0. Der zugehörige SFG ist dem SFG aus dem zweiten Fall (**Bild** 7) identisch, wohingegen nun ein bekannter Fehler als vorhandener Fehler erkannt wird.

Die Ist-Tabelle zum vierten Fall (**Bild** 9) ist analog zum ersten Fall ausgefüllt, so dass jetzt *spec* in der dualen Rail bzgl. ¬/B, ¬/C, ¬D und ¬E erfüllt ist. Analog dazu können weitere Beispiele generiert werden.

Es können nun Suchfunktionen programmiert werden, welche die QVL (**Tabelle** 7) durchgehen und bestimmte Kriterien wie z.B. Testabdeckung, Fehlerabdeckung und Prüfschärfe bestimmen. Die durch eine komplexe Realität erzeugten QVL können jedoch in ihren Dimensionen große Ausmaße annehmen. Um deswegen den geringeren Speicherbedarf und die kürzere Rechenzeit zu sichern, sollte die Datenbank (QVL) (ohne Informationsverlust) kompaktiert werden. Das letzte Kapitel illustriert das Vorgehen bei einer solchen verlustlosen Datenkompression.

Z^a				PI	Schalter				Z^n				Funktion																
/B	/C	D	E	A	S1	/S2	/S3	S4	/B	/C	D	E	$s_{/B}$	$s_{n/B}$	$t_{/B}$	$t_{n/B}$	$k_{/B}$	$k_{n/B}$	$/k_{n/B}$	$s_{/C}$	$s_{n/C}$	$t_{n/C}$	s_D	s_{nD}	t_D	k_D	k_{nD}	s_E	s_{nE}	t_E
0	x	x	x	1	x	x	x	x	0	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1	x	x	x	0	x	x	x	x	1	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0	x	x	x	0	0	-	x	x	1	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0	x	x	x	0	-	1	x	x	1	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1	x	x	x	0	1	0	x	x	0	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1	x	x	x	1	-	-	x	x	0	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x
x	0	x	x	x	x	x	x	x	0	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	x	x
x	1	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	x
x	1	x	x	x	x	x	x	x	0	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x
x	0	x	x	x	x	x	x	x	x	0	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x
x	1	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x
x	1	x	x	x	0	-	x	x	x	0	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x
x	1	x	x	x	-	0	x	x	x	0	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x
x	x	1	x	x	x	x	x	x	x	0	1	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x
x	x	0	x	x	x	x	x	x	x	1	0	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x
x	x	1	x	x	x	x	0	-	x	x	0	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x
x	x	1	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x
x	x	0	x	x	x	x	x	x	x	x	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x
x	x	x	1	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x
x	x	x	0	x	x	x	x	x	x	x	x	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x
x	x	x	1	x	x	x	-	0	x	x	x	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1
x	x	x	1	x	x	x	0	-	x	x	x	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1

Tabelle 7 Modell der Schaltung (in QVL)

ISBN 978-3-8007-4444-2

Z^a				PI	Schalter				Z^n			
/B	/C	D	E	A	S1	/S2	/S3	S4	/B	/C	D	E
0	0	1	1	1	1	1	1	1	0	0	1	1

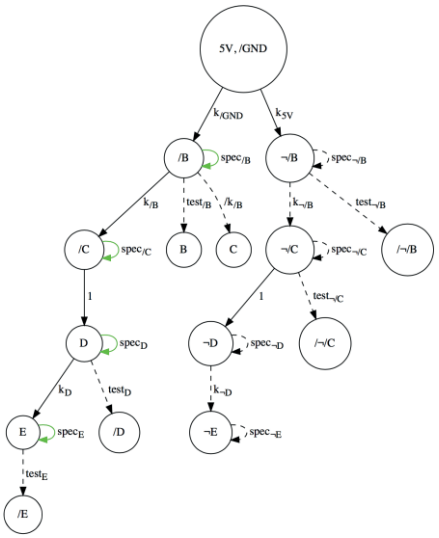

Bild 6 Fall 1: Ist-Tabelle und zugehöriger SFG

Z^a				PI	Schalter				Z^n			
/B	/C	D	E	A	S1	/S2	/S3	S4	/B	/C	D	E
0	0	1	1	0	0	1	1	1	1	0	1	1

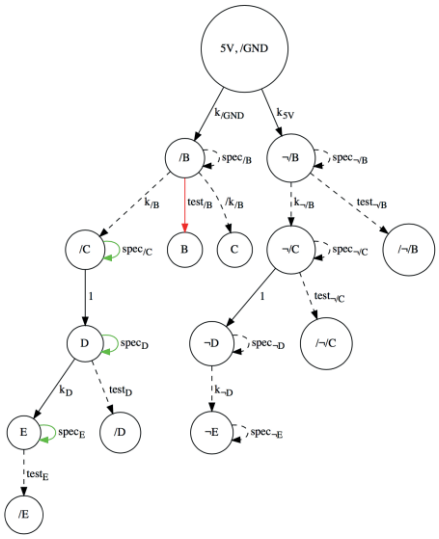

Bild 8 Fall 3: Ist-Tabelle und zugehöriger SFG

Z^a				PI	Schalter				Z^n			
/B	/C	D	E	A	S1	/S2	/S3	S4	/B	/C	D	E
0	0	1	1	0	1	1	1	1	1	0	1	1

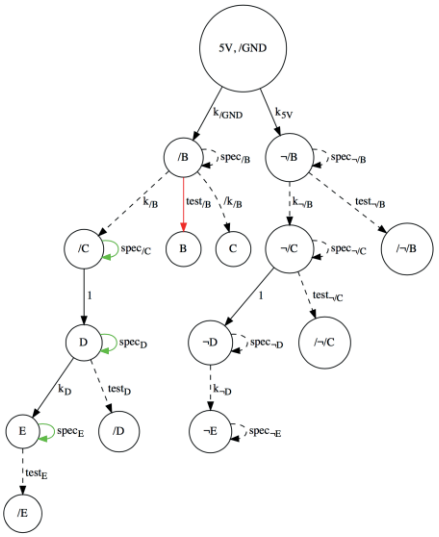

Bild 7 Fall 2: Ist-Tabelle und zugehöriger SFG

Z^a				PI	Schalter				Z^n			
/B	/C	D	E	A	S1	/S2	/S3	S4	/B	/C	D	E
1	1	0	0	0	1	1	1	1	1	1	0	0

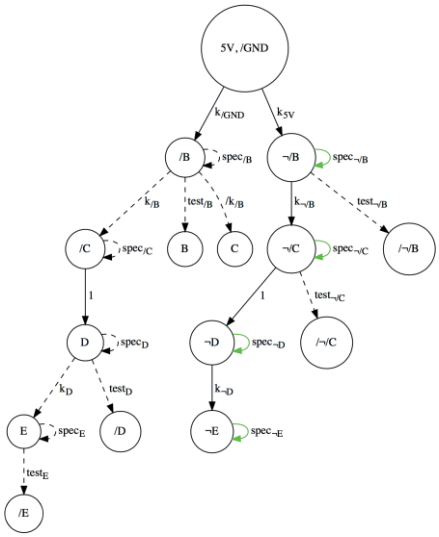

Bild 9 Fall 4: Ist-Tabelle und zugehöriger SFG

4 Datenkompaktierung

¦	0	1	*	-
0	0	id	$*_0$	$-_0$
1	id	1	$*_1$	$-_1$
*	0_*	1_*	*	$-_*$
-	0_-	1_-	$*_-$	-

Tabelle 8 Verknüpfungstabelle

Die Verknüpfungstabelle (**Tabelle** 8) gibt die stellenweise Verknüpfung von QVL wieder. Das Symbole ¦ steht nach eigener Definition für die stellenweise, reguläre *Compaction* zweier Quaternär-Vektoren QV1 und QV2. In **Tabelle** 8 beziehen sich die Zeilen auf QV1, die Spalten auf QV2. Die Diagonale ist selbsterklärend, z.B. ¦(0,0) = 0. Die restlichen Stellen sind mit Symbolen gefüllt, die als existente Endergebnisse der Kompaktierung zu betrachten sind. Der regulären Compaction von (0,1) mit (1,0) wird das Symbol „id“ zugewiesen. Das Symbol „*0“ bedeutet, dass QV2 als „*“ Untermenge von QV1 als „0“ ist, der

ISBN 978-3-8007-4444-2

Stern aus QV2 wird konkretisiert in der „0“ von QV1. Das Symbol „$-_0$“ bedeutet dann, dass QV2 als „-“ Obermenge von QV1 als „0“ ist, das „-“ aus QV2 wird konkretisiert in der „0“ von QV1. Restliche Symbole gelten entsprechend. Um eine vollständig parallele Verarbeitung während der Kompaktierung zu erreichen, wird die stellenweise Quaternär-Listenkodierung aus **Tabelle** 8 zusätzlich in ein 5 stelliges Kodierungsuniversum (die ersten 5 Spalten der **Tabelle** 9) eingebettet. Betragsstriche akkumulieren dabei jeweils das Vorkommen des darin angegebenen Symbols. Beispielweise bedeutet | id | = 0, dass „id“ bei der regulären *Compaction* der beiden Quaternär-Vektoren QV1 und QV2 nicht vorkommt, während | id | = 1 bedeutet, dass „id“ genau einmal vorkommt. $|(0,1,-)_*| > 0$ bedeutet, dass „0_*“, „1_*“ und/oder „$-_*$“ mindestens einmal vorkommen. Die ersten fünf Spalten geben alle existenten möglichen Kombinationen an, bei denen eine Kompaktierung existiert. Die restlichen fünf Spalten aus **Tabelle** 9 geben dann die Symbole an, mit denen Symbole aus **Tabelle** 8 jeweils ersetzt werden müssten. Die konkreteren TVL können mit der vorgestellten Methode ebenfalls kompaktiert werden, dies sind die Zeilen 1 bis 3 und 9 bis 12 für $|(0,1,-)_*| = |*_{(0,1,-)}| = 0$. Für alle anderen nicht aufgeführten Kombinationen (Zeilen) in den ersten fünf Spalten in **Tabelle** 9 existiert keine Kompaktierung. In den ersten acht Zeilen von **Tabelle** 9 liegt eine Untermengenbeziehung vor, in Zeile neun ein Consensus und in den letzten drei Zeilen eine Überlappung. Wenn die Kodierung des Zwischenergebnisses einer der Kombinationen aus den ersten neun Zeilen aus **Tabelle** 9 entspricht, dann sind die beiden Quaternärvektoren Untermengen des Endergebnisses und werden deswegen durch das Endergebnis ersetzt. Sollte die Kodierung einer der letzten drei Zeilen entsprechen, dann ersetzt das Endergebnis QV1, wenn QV1 eine Untermenge des Endergebnisses ist. Dasselbe gilt für QV2 als Untermenge des Endergebnisses. Sollten beide Quaternärvektoren nicht Untermenge des Endergebnisses sein, dann fungiert das Endergebnis als weiterer Quaternär-Vektor, um einer fortschreitenden Kompaktierung zu dienen. Folglich wird die QVL wie beim Quine-McCluscy-Verfahren [9] erst größer, letztendlich bei voranschreitender Kompaktierung kleiner
Mit Hilfe der Rechenbeispiele aus **Tabelle** 10 kann leicht nachvollzogen werden: Im Fall 1 folgt als Zwischenergebnis der regulären *Compaction* [0 1 0] ¦ [- 1 0] = [$-_0$ 1 0]. Die Kodierung dieses Zwischenergebnisses, [0 0 0 0 >0], entspricht der zweiten Zeile in **Tabelle** 9, was dazu führt, dass „$-_0$“ noch durch „-“ ersetzt wird. Die Fälle 2 bis 4 sind analog zu verstehen und beziehen sich auf Zeile 2, 5 und 7 aus **Tabelle** 9. Das Zwischenergebnis im Fall 5 entspricht einer Kodierung von [1 0 >0 0 0], was in keiner der Zeilen aus **Tabelle** 9 vorkommt. Damit wird das Zwischenergebnis [id 0 $*_1$] nicht weiterverwendet. Im Fall 6 werden drei Ternär-Vektoren betrachtet und zum besseren Verständnis farblich markiert. Zunächst ergibt sich für das Endergebnis der Kompaktierung von TV1 (blau) und TV2 (rot) [- 1], was aus Betrachtung von Zeile 10 in **Tabelle** 9 folgt. Dieses Ergebnis überdeckt TV1, was dazu führt, dass TV1 mit [- 1] ersetzt wird und anschließend mit TV3 (grün) verodert wird. Nach Zeile 8 aus **Tabelle** 9 führt dies dazu, dass beide Ternärvektoren (TV) mit [- -] ersetzt werden. Dieses Ergebnis wird mit dem noch verbliebenen TV2 verodert und aus Zeile 3 (**Tabelle** 9) folgt die Ersetzung beider TV mit dem Endergebnis [- -].

	| id |	**$\|(0,1,-)_*\|$**	**$\|*_{(0,1,-)}\|$**	**$\|(0,1)_-\|$**	**$\|-_{(0,1)}\|$**	**id**	**$(0,1,-)_*$**	**$*_{(0,1,-)}$**	**$(0,1)_-$**	**$-_{(0,1)}$**
1	0	0	0	0	0	*	*	*	*	*
2	0	0	0	0	> 0	*	*	*	*	-
3	0	0	0	> 0	0	*	*	*	-	*
4	0	0	> 0	0	0	*	*	(0,1,-)	*	*
5	0	0	> 0	> 0	0	*	*	(0,1,-)	-	*
6	0	> 0	0	0	0	*	(0,1,-)	*	*	*
7	0	> 0	0	0	> 0	*	(0,1,-)	*	*	-
8	0	> 0	> 0	0	0	*	(0,1,-)	(0,1,-)	*	*
9	1	0	0	0	0	-	*	*	*	*
10	1	0	0	0	>0	-	*	*	*	(0,1)
11	1	0	0	>0	0	-	*	*	(0,1)	*
12	1	0	0	>0	>0	-	*	*	(0,1)	(0,1)

Tabelle 9 Alle existenten Kombinationen und deren Ersetzungswerte

Fall 1	Fall 2	Fall 3	Fall 4	Fall 5	Fall 6
0 1 0	0 1 - 1	* 1 0	0 * 1 *	1 0 1	1 1 = 1 1 + - 1 + - -
¦ - 1 0	¦ * 1 0 *	¦ 1 - 0	¦ * - * 1	¦ 0 0 *	0 - = 0 - + - 0 + 0 -
$-_0$ 1 0	$*_0$ 1 0 $*_1$	1_* $-_1$ 0	$*_0$ $-_*$ $*_1$ 1_*	id 0 $*_1$	- 0 id -1 - id 0_- -
⇒ **- 1 0**	⇒ **0 1 - 1**	⇒ **1 - 0**	⇒ **0 - 1 1**	⇒ * * *	- - = - 1 = - - = - -

Tabelle 10 Rechenbeispiele zum Kompaktieren

ISBN 978-3-8007-4444-2

Z^a				PI	Schalter				Z^n				Funktion (Kodierungsuniversum)															
/B	/C	D	E	A	S1	/S2	/S3	S4	/B	/C	D	E	$s_{/B}$ (/B, A, /B)	$s_{n/B}$ (/B, A, /B)	$t_{/B}$ (/B, A, S1, /S2, /B)	$t_{n/B}$ (/B, A, S1, /S2, /B)	$k_{/B}$ (/C, /B)	$k_{n/B}$ (/C, /D)	/$k_{n/B}$ (/C, /B)	$s_{/C}$ (/C, /C)	$s_{n/C}$ (/C, /C)	$t_{n/C}$ (/C, S1, /S2, /C)	s_D (D, /C, D)	s_{nD} (D, /C, D)	t_D (D, /S3, S4, D)	k_D (D, D)	k_{nD} (D, D)	s_E (E, E)	s_{nE} (E, E)	t_E (E, /S3, S4, E)
1	X	X	X	1	-	-	X	X	0	X	X	X	X	X	X	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X
1	0	1	X	0	1	0	X	X	0	0	1	X	X	X	X	1	1	X	X	1	X	X	1	X	X	X	X	X	X	X
1	X	0	0	0	X	X	X	X	1	1	0	0	X	1	X	X	X	X	X	X	X	X	X	1	X	X	X	1	X
0	1	1	1	0	-	1	0	-	1	1	0	1	X	X	1	X	X	X	X	X	1	X	X	X	1	X	X	1	X	X
0	1	1	1	0	0	-	-	0	1	0	1	0	X	X	1	X	X	1	X	X	X	1	X	X	X	1	X	X	X	1
0	1	0	1	1	-	0	0	-	0	0	0	0	1	X	X	X	X	X	1	X	X	1	X	X	X	X	1	X	X	1

Tabelle 11 Kompakte QVL aus **Tabelle** 7

Das hier erklärte Vorgehen der Kompaktierung wird auf die QVL des erreichten Modells in **Tabelle** 7 iterative mit Hilfe eines selbstverfassten Programcodes angewandt, was zu **Tabelle** 11 führt. Hier wird beispielsweise die Funktion $t_{n/B}$ erfüllt bzw. erhält den Wert 1, wenn für dessen Kodierungsuniversum (/B, A, S1, /S2, /B) = [1 1 - - 0] gilt. Diese Bedingung entspricht derselben aus **Tabelle** 7. Analog können alle weiteren Funktionen überprüft werden und damit die Datenkompaktierung bestätigt werden. Es wird ein sehr hoher Grad der Kompaktierung erwartet, in unserem Beispiel $2+\log_2$ (AnzahlZeilen).

5 Zusammenfassung und Ausblick

Die in diesem Beitrag behandelte strukturtreue Modellierung anhand von Signalflussgraphen ist eine strukturtreue Verifikationsmethode, die dazu dient, Systeme bzw. Schaltungen auf bekannte Fehler zu testen. Unter strukturtreuer Modellierung verstehen wir die Übereinstimmung der formal hergeleiteten Funktion mit der real erzeugten Funktion. Die Verifikation lässt sich mit Hilfe von bestimmten hergeleiteten Regeln in drei Schritten durchführen: von der Einbettung des realen Modells in ein Kodierungsuniversum, über die Angabe eines Datenmodells als SFG in Dual Rail, bis hin zur Erstellung der Teilmodelle (OP, CTRL) in AA, was im Aufstellen einer QVL resultiert. Im Vergleich zu bekannten Methoden wie Simulation und Validierung kümmert sich die vorgeschlagene Methodik um die Richtung der Struktur und kann daher die Struktur des zu prüfenden Gerätes innerhalb seiner gerichteten Struktur bewahren. In unserer Terminologie wird dies als Verifikation bezeichnet. Damit besteht die Möglichkeit, algebraische Methoden anzuwenden, die unter Idempotenz abgeschlossen sind. Ein Ergebnis daraus ist schneller und kompakter Code. Inwieweit solche Ansätze aber übernommen werden können, wird sich noch zeigen. Der Aufwand für die Modellierung der zugrunde liegenden Struktur liegt dabei in der Abstraktion des Signalflussgraphen (SFG). Die hier vorgestellte (verlustlose) Kompaktierung sichert einen geringen Bedarf an Speicherplatz und damit geringe Rechenzeiten. Es können QVL als auch TVL kompaktiert werden. Diese zeilenweise kompaktierte Datenbank ist für ihren funktionalen Anteil spaltenweise mehrdimensional, z.B. bei Abhängigkeiten bzw. Korrelationen zwischen den einzelnen Funktionen (*spec*, *test*, *k*, */k*) und ihren möglichen Fehlermodellen (S). Also ist es sinnvoll solche Zusammenhänge auch unter dem Aspekt von Klassifikationsverfahren zu untersuchen, welche in Dimensionen von Datenmengen klassifizieren können. Dies ermöglichen die Hauptkomponenten- und die Lineare Diskriminanzanalyse [10].

6 Literatur

[1] Xu, Y.: *Algorithms for Automatic Generation of Relative Timing Constraints*. Ph.D. dissertation, Salt Lake City, UT, USA. (2011).

[2] Kurshan, R. P.; McMillan, K. L.: *Analysis of digital circuits through symbolic reduction*. IEEE Trans. *on CAD of Integrated Circuits and Systems*, pp. 1356–1371. (1991).

[3] Yoneda , T.; Kitai ,T.; Myers, C. J.: *Automatic Derivation of Timing Constraints by Failure Analysis*. in CAV, vol. 2404. Springer, pp. 195–208. (2002)

[4] Uygur, G.; Sattler, S.: *Structure Preserving Modeling for Safety Critical Systems*. IEEE 20th International Mixed-Sig-nals Testing Workshop (IMSTW). (2015)

[5] Siegfried, W.: Operationszustand versus Steuerzustand - eine äußerst zweckmäßig Unterscheidung. Technische Universität Kaiserslautern. (2000)

[6] Posthoff, Ch.; Steinbach, B.: *Logic Functions and Equations. Springer*. (2004)

[7] Gössel, M.: Automatentheorie für Ingenieure. Akademie Verlag. Berlin. (1991)

[8] Zander, H.J.: Logischer Entwurf binärer Systeme. VEB Verlag Technik. Berlin. (1989).

[9] E. McCluskey, *Logic Design Principles* (Prentice-Hal, Englewood Cliffs, New Jersey, 1986)

[10] Xie, Y.; Zhang, T.: *A fault diagnosis approach using SVM with data dimension reduction by PCA and LDA method. Chinese Automation Congress* (CAC). pp. 869 - 874. (2015)

ISBN 978-3-8007-4444-2