Technologieabhängigkeit von Spezifikationen digitaler Hardware

Max Fuchs

Juli 1994

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Verteilte Systeme bestehen aus mehreren, räumlich verteilten Komponenten, die nebenläufig arbeiten und miteinander kommunizieren. Anwendungen verteilter Systeme wie

- das Steuern von Produktionsprozessen oder Verkehrssicherungssystemen,
- die Modellierung von Betriebssystemen und
- das Beschreiben von Hardwareschaltungen

umfassen Teilgebiete der praktischen, der systemnahen und der technischen Informatik. Die Laufzeit verteilter Systeme ist in vielen Fällen zumindest prinzipiell unbeschränkt und die zur Beschreibung erforderlichen Modellierungen oftmals nichtdeterministisch. Ein erschöpfender Test solcher Systeme ist in der Praxis unmöglich. Darüber hinaus werden verteilte Systeme häufig für die Realisierung sicherheitskritischer Anwendungen eingesetzt, wo die Zuverlässigkeit und die Korrektheit der zu entwickelnden Systeme unabdingbar sind. Um diesen Punkten Rechnung zu tragen, bietet sich eine formale Entwicklung verteilter Systeme basierend auf mathematisch fundierten Spezifikations- und Entwurfsformalismen an. Für die Entwicklung von Systemen realistischer Größe bedarf es sowohl modularer Beschreibungs- als auch Verfeinerungstechniken ([Bro92b, San88]). Eine Entwurfsmethodik, die diese Voraussetzungen erfüllt ist Focus ([BDD+92a]) – sie stellt die Grundlage dieser Arbeit dar.

Die Entwicklung formaler Spezifikations- und Entwurfsformalismen für verteilte Systeme hat eine Reihe von tragfähigen Konzepten hervorgebracht. Dazu gehören temporallogische Ansätze [Krö87], Petrinetze [Rei85], Prozeßalgebren [Mil80, Hoa85], Statecharts [Har87], die "transition axiom method" von Lamport [Lam88] sowie Unity [CM88] um nur einige zu nennen.

Beim Entwurf verteilter Systeme spielen neben der verwendeten Entwurfsmethodik auch adäquate, dem jeweiligen Abstraktionsniveau angepaßte, formale Modellierungen eine entscheidende Rolle. Diese ermöglichen es, aufgrund geeigneter Abstraktion von Implementierungsdetails, komplexe Zusammenhänge verstehen und verifizieren zu können. Die Erstellung adäquater, formaler Beschreibungen digitaler Hardwarekomponenten zum Zwecke technologieabhängiger Untersuchungen steht im Mittelpunkt dieser Arbeit.

Die Entwicklung digitaler Bausteine stellt wegen ihrer hohen Funktionskomplexität gesteigerte Anforderungen an den Entwickler. Der Aspekt der Entwurfskorrektheit spielt neben dem Einsatz von Hardwarebausteinen in sicherheitskritischen Systemen auch in Hinblick auf die ökonomischen Randbedingungen der Höchstintegration eine entscheidende Rolle. Ein Entwurfszyklus bei hochintegrierten Bausteinen kann von der Entdeckung eines Fehlers bis zum neuen Baustein bis zu drei Monate in Anspruch nehmen, wobei ein integrierter Baustein, der sechs Monate später am Markt erscheint, bis zu einem Drittel seines Marktwerts verliert ([Eve91]).

Entscheidend bei der Modellierung digitaler Hardwareschaltungen ist es, gerade diejenigen Eigenschaften zu erfassen, die für die jeweilige Modellierungsaufgabe notwendig sind. Bei Modellen digitaler Hardware unterscheiden wir

- Simulationsmodelle und
- Modelle für den formalen Hardwareentwurf.

Simulationsmodelle sind ablauffähige Verhaltensmodelle mit deren Hilfe die Auswirkungen von Eingabesequenzen berechnet werden. Die Simulation selbst ist unvollständig, da in der Regel aus Zeitgründen nie alle möglichen Eingabesignale simuliert werden können. Für den vollständigen Test eines 32-Bit Addierers, bei einer Testgeschwindigkeit von 1 Mikrosekunde pro Addition, würde die Zeit seit der Entstehung des Universums bei weitem nicht ausreichen ([Eve91]). Formale Modelle, also solche mit wohlfundierter Semantik, sind die Grundlage für formale Verifikation, formale Transformation und automatische Synthese. Die Beschreibungen (Spezifikationen) umfassen ein breites Spektrum – von deskriptiven, eigenschaftsorientierten bis hin zu konstruktiven, also ausführbaren Beschreibungen. Eine deskriptive Beschreibung erlaubt eine abstrakte Sicht auf eine physikalische Komponente. Eine konstruktive Beschreibung vermittelt eine operationelle Sichtweise. Basierend auf der formalen Semantik lassen sich Beschreibungen auf unterschiedlichen Abstraktionsebenen in Beziehung setzen – dies werden wir im Verlauf dieser Arbeit noch ausnützen.

Wie bereits erwähnt, liegt ein wesentlicher Aspekt der Modellierung darin, lediglich die notwendigen Eigenschaften einer physikalischen Komponente unter Berücksichtigung der gestellten Modellierungsaufgabe zu erfassen – nur so erhält man verständliche und handhabbare Modelle. Will man ausschließlich das Verhalten einer Komponente erfassen, so spielen zunächst Eigenschaften wie Struktur, Platzbedarf oder Produktionskosten ([Rug91]) keine Rolle. Auch die Realisierungstechnologie wird in der Literatur für eine mehr oder weniger irrelevante Größe bei der Modellierung reinen Verhaltens von Hardwarebausteinen betrachtet. So bewertet [HNS86] den Grad der Technologieabhängigkeit bereits für Beschreibungen auf der Register-Transfer-Ebene mit Null. Die Register-Transfer-Ebene, im weiteren auch als RT-Ebene bezeichnet, ist dadurch charakterisiert, daß sie logische und arithmetische Komponenten hinsichtlich einer bitorientierten Schnittstelle beschreibt, ohne jedoch die interne Gatterstruktur oder gatterspezifische Eigenschaften einzubeziehen. Ein wenig differenzierter wird mit der Technologieabhängigkeit in [KHK93] umgegangen. Neben Verzögerungszeiten werden dort unterschiedliche Treiberstärken als technologieabhängige Größen aufgeführt. Die Einbeziehung von Treiberstärken in Beschreibungen digitaler Hardware bedeutet in der Tat die Einbeziehung gatterspezifischer Eigenschaften und somit den Wechsel der Abstraktionsebene auf die

Ebene der Gatterbeschreibungen. Was die Verzögerungszeit betrifft, so ist es sicherlich richtig, daß jede Technologie eine ihr zugrundeliegende, kleinste Schaltgeschwindigkeit besitzt. Für komplexere Schaltungen, mit Verzögerungszeiten weit über dem Vielfachen der unterschiedlichen, technologiebedingten Schaltzeiten, ist jedoch nicht mit Sicherheit auszuschließen, daß für eine geschickte Wahl der Schaltungsarchitektur sowie des topologischen Aufbaus (z.B. Leitungsführung) nicht doch jede Technologie für die konkrete Realisierung herangezogen werden kann. Diese Überlegungen führen dazu, die Verzögerungszeiten als zwingend technologieabhängige Größen auszuklammern. Im Zusammenspiel mit anderen Faktoren wie beispielsweise der maximalen Leistungsaufnahme können unterschiedliche Verzögerungszeiten aber durchaus zu unterschiedlichem Verhalten gleicher Komponenten mit unterschiedlichen Realisierungstechnologien führen – dies wollen wir im weiteren Verlauf dieser Arbeit noch deutlich herausarbeiten.

Nun stellt sich die Frage, ob Spezifikationen digitaler Hardware bereits auf der Register-Transfer-Ebene tatsächlich technologieunabhängig sind, so wie es in [HNS86] formuliert wurde. Diese Fragestellung und eine damit verbundene Klärung stehen im Zentrum dieser Arbeit. Dabei wollen wir Hardwarebeschreibungen nur dann als technologieabhängig bezeichnen, falls unterschiedliche Realisierungstechnologien zu unterschiedlichem Verhalten führen. Ausgangspunkt der Untersuchungen sind die Beschreibung eines RS-Flipflops sowie einer Busstruktur basierend auf Beschreibungskonzepten der Hardwarebeschreibungssprache VHDL. Die Wahl dieser beiden Hardwarekomponenten begründet sich durch deren zentrale Stellung im Bereich digitaler Hardware. RS-Flipflops sind die Basisbausteine aller sequentiellen Hardwarekomponenten und Busstrukturen stellen ein wesentliches Bindeglied zwischen digitalen Hardwarekomponenten dar. Abbildung 1.1 veranschaulicht die für diese Arbeit zugrundegelegten Hardwareschaltungen sowie die zugehörigen Modellierungen bzw. Modellierungsideen.

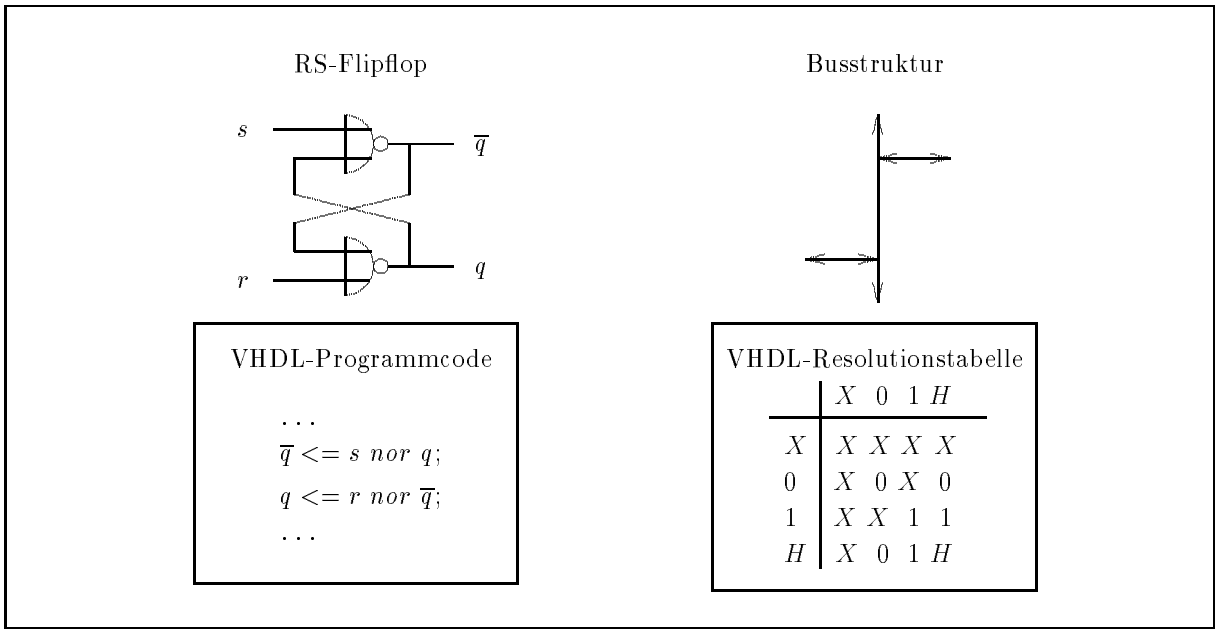


Abbildung 1.1: VHDL Modellierungskonzepte für das RS-Flipflop und die Busstruktur.

Für die Modellierung greifen wir, wie bereits erwähnt, auf Konzepte der Hardwarebeschreibungssprache VHDL zurück. VHDL ist eine von IEEE standardisierte Hardwarebeschreibungssprache ([EEE87, EEE92, LSU90]). Die Dominanz von VHDL und der dort verwendeten Konzepte spiegelt sich in der großen Akzeptanz der Sprache wieder – in [Car93] wird davon ausgegangen, daß im Jahre 1994 bereits die Hälfte aller Hardwareentwickler VHDL benutzen wird. Die in dieser Arbeit verwendeten VHDL-Konzepte umfassen u.a. das Inertialdelay-Zeitkonzept, die Verwendung von Initialwerten aber auch die Modellierung von Busstrukturen unter Verwendung geeigneter Resolutionsmechanismen. Für die Modellierung des RS-Flipflops auf der Gatterebene verwenden wir, in Anlehnung an VHDL, zwei nebenläufig auszuführende NOR-Gatter, deren Ausgänge jeweils zu den Eingängen des anderen NOR-Gatters zurückgeführt werden. Man beachte, daß keinerlei Einschränkungen bezüglich der Eingabedaten vorliegen. Die Modellierung einer Busstruktur basiert in VHDL auf einer Resolutionstabelle. Diese regelt das Busverhalten im Konfliktfall, also falls mehr als eine Komponente schreibend auf den Bus zugreift. Die Resolutionstabelle selbst berücksichtigt unterschiedliche Treiberstärken sowie den physikalischen Aufbau (Open Collector oder Tristatebus). Man beachte, daß auch die Eingabedaten im Falle der Busstruktur keinerlei Einschränkungen unterliegen – Buskonflikte sind nicht ausgeschlossen. Die hier vorgestellten Modellierungsideen von VHDL einschließlich der verwendeten Konzepte bilden die Grundlage der funktionalen Modellierung dieser Arbeit. Die im weiteren Verlauf dieser Arbeit als "technologieunabhängige Spezifikationen" bezeichneten Beschreibungen von RS-Flipflops und Busstrukturen sind technologieunabhängige im Sinne von VHDL oder technologieunabhängig in dem Sinne, daß das Abstraktionsniveau der Schaltungsbeschreibungen der RT-Ebene entspricht ([HNS86]). Da es sich diese Arbeit zur Aufgabe gemacht hat, den Sachverhalt der Technologieabhängigkeit genauer zu untersuchen, wären Formulierungen wie "sogenannte technologieunabhängige Spezifikationen", "im Sinne von VHDL technologieunabhängige Spezifikationen" oder "im Sinne von [HNS86] technologieunabhängige Spezifikationen" angebracht. Der besseren Lesbarkeit wegen wollen wir jedoch lediglich den Begriff der "technologieunabhängigen Spezifikation" verwenden, aber dabei stets dessen Verbindung zu den entsprechenden Beschreibungen in VHDL und zu [HNS86] bedenken.

1.2 Ziele, Methoden und Aufbau der Arbeit

Die Zielsetzung dieser Arbeit liegt in der Untersuchung der Technologieabhängigkeit von Beschreibungen digitaler Hardware auf der RT-Ebene. Zentrale Bedeutung kommt dabei der adäquaten Modellierung zu, wobei die wesentlichen Modellierungskonzepte der Hardwarebeschreibungssprache VHDL entliehen sind. Die hier betrachteten Realisierungstechnologien umfassen NMOS und CMOS ([Man79]) und die für die Untersuchung gewählten Hardwareschaltungen das RS-Flipflop und die Busstruktur. Im einzelnen soll die Arbeit

- adäquate Spezifikationen für die gewählten Hardwareschaltungen auf der RT-Ebene (technologieunabhängige Spezifikationen) und geeignete Beschreibungen auf der Gatterebene (technologieabhängiger Fall für NMOS und CMOS) bereitstellen,
- den Zusammenhang zwischen technologieunabhängigen und technologieabhängigen Beschreibungen unter Verwendung eines geeigneten Verfeinerungsbegriffs untersuchen

und basierend auf diesen Untersuchungen die These der Technologieunabhängigkeit von Beschreibungen auf der RT-Ebene unter Verwendung von Modellierungskonzepten von VHDL widerlegen. Bei der technologieabhängigen Modellierung der Komponenten kommt der Beschreibung fehlerhaften Verhaltens eine wesentliche Rolle zu. Die in dieser Arbeit zugrundegelegte formale Entwurfsmethodik ist Focus [BDD+92a]. Neben der zentralen Aufgabe, Technologieabhängigkeiten zu untersuchen, trägt diese Arbeit auch zur Weiterentwicklung von Focus in Hinblick

- auf die Beschreibung digitaler Hardwareschaltungen

bei. Dazu gehören neben methodischen Aspekten, wie z.B. dem Trennen von Zeit- und Verhaltensbeschreibung, auch notwendige, technische Erweiterungen zur Behandlung zeitbehafteter Hardwarekomponenten.

Im folgenden wollen wir den Aufbau der Arbeit erläutern. Kapitel 2 behandelt die Grundlagen der Entwurfsmethodik Focus. Neben Strömen, stromverarbeitenden Funktionen und Spezifikationen, basierend auf stromverarbeitenden Funktionen, behandeln wir einen Verfeinerungsbegriff, der es erlaubt, unterschiedliche Spezifikationen miteinander in Beziehung zu setzen. Um Spezifikationen lesbarer zu gestalten, führen wir tabellarische Beschreibungsformen ein.

In Kapitel 3 werden die Grundlagen zur Spezifikation digitaler Hardware gelegt – dabei steht die Hardwarebeschreibungssprache VHDL in vielen Fällen Pate. Neben einem modularen Entwurf, also hier speziell der Trennung von Zeit- und Verhaltensbeschreibungen, verwenden wir Initialwerte. Letztere dienen zum einen der Festlegung von Ausgabewerten bis der erste berechnete Wert vorliegt und zum anderen als Startwert für Rückkopplungsschleifen. Der Zeitbegriff spielt bei der Modellierung digitaler Hardwarekomponenten eine zentrale Rolle. Dies spiegelt sich sowohl in den Strömen als auch in den stromverarbeitenden Funktionen wieder. Bei gezeiteten Strömen gehen wir davon aus, daß hinter jeder Aktion im Strom ein festes Zeitintervall steht. Bei stromverarbeitenden Funktionen wird die Behandlung gezeiteter Ströme durch eine sogenannte “time progress property” festgelegt. Diese Eigenschaft sorgt dafür, daß die Zeit an den Ausgängen im gleichen Maße

voranschreitet wie an den Eingängen. Die Idee der "time progress property" wird in dieser Arbeit in Hinblick auf Verhaltens- und Zeitbeschreibungen genauer aufgeschlüsselt. Um Verzögerungszeiten von Komponenten angemessen modellieren zu können, spezifizieren wir Verzögerungskonzepte wie Transport- und Inertialdelay. Ein weiteres Kernstück dieses Kapitels behandelt die Rückkopplung. Dabei unterscheiden wir zwischen verzögerungsfreien und verzögerten Anwendungen. Aus Gründen einer einheitlichen Methodik wird für beide Anwendungsfälle ein einheitlicher Rückkopplungsoperator formuliert. Die verzögerungsfreie Variante des Rückkopplungsoperators kommt im Zusammenhang mit der Modellierung von Busstrukturen zur Anwendung. Abgeschlossen werden die Überlegungen zur Spezifikation digitaler Hardware in Focus mit einer sogenannten Zeitabstraktion. Diese gestattet es, Hardwarebeschreibungen mit unterschiedlichen Zeitbasen untereinander zu verbinden.

Das eigentliche Kernstück dieser Arbeit bilden die beiden Kapitel über die Untersuchung der Technologieabhängigkeit von Spezifikationen des RS-Flipflops und Busstrukturen auf der RT-Ebene. Kapitel 4 befaßt sich mit dem RS-Flipflop. Im Anschluß an eine Klärung von Aufbau und Wirkungsweise zerfällt das Kapitel in drei Teile. Der erste Teil ist den technologieunabhängigen Modellierungen von RS-Flipflops gewidmet, der zweite der Modellierung von NMOS- und CMOS-Varianten und der dritte den Untersuchungen der Verfeinerungsbeziehungen. Als keineswegs notwendige, aber sehr angenehme Eigenschaft der Spezifikationen für RS-Flipflops ergibt sich die Möglichkeit, das Flipflop-Verhalten selbst im Schwingungsfall adäquat beschreiben zu können. Bei der Modellierung der technologieabhängigen, also der NMOS- und CMOS-Flipflops steht nun die Frage nach einem tatsächlichen Verhaltensunterschied im Vergleich zur technologieunabhängigen Spezifikation zur Diskussion. Der entscheidende Beitrag zur Klärung dieser Fragestellung ergibt sich bei der näheren Betrachtung des physikalischen Aufbaus bzw. der damit verbundenen Leistungsaufnahme beider Realisierungstechnologien. Die im Falle von NMOS statische Leistungsaufnahme steht einer im Falle von CMOS dynamischen, insbesondere von der Schaltungsfrequenz abhängigen, Leistungsaufnahme gegenüber. In anderen Worten kann es bei CMOS-RS-Flipflops im Schwingungsfall zur thermischen Zerstörung kommen und somit zu einem in unserem Sinne tatsächlichen Verhaltensunterschied, da sich bei einer zerstörten Komponente nicht mehr alle (oder auch keine) Ausgänge über die Eingänge einstellen lassen. Basierend auf diesen Überlegungen ergeben sich nun die entsprechenden Spezifikationen für NMOS- und CMOS-RS-Flipflops, wobei die thermische Zerstörung mit in die Modellierung der CMOS-Variante einzugehen hat. Im Anschluß an die Modellierungen des RS-Flipflops gilt es nun die technologieunabhängige Spezifikation mit den technologieabhängigen Spezifikationen in Beziehung zu setzen. Die NMOS-Spezifikation erweist sich in Hinblick auf die technologieunabhängige Variante als eine Verfeinerung in unserem Sinne. Dies gilt nicht für die CMOS-Spezifikation, da nach einer eventuellen thermischen Zerstörung echte Verhaltensunterschiede zu beobachten sind.

Die Untersuchungen zur Technologieabhängigkeit von Busstrukturspezifikationen finden sich in Kapitel 5. Der prinzipielle Aufbau dieses Kapitels orientiert sich an Kapitel 4. Nach einer kurzen Einführung in Busstrukturen werden zunächst technologieunabhängige sowie NMOS- und CMOS-Spezifikationen erstellt und dann diese gemäß der Verfeinerungsrelation zueinander in Beziehung gesetzt. Die betrachteten Busstrukturen sind bidirektional, erlauben eine beliebige, aber fest zu wählende Anzahl von anzuschließenden

Komponenten und beschränken sich auf die Übertragung einer binären Information pro Zeiteinheit (Busbreite = 1). Bezüglich der anzulegenden Daten gibt es keine Einschränkungen – Buskonflikte müssen behandelt werden. Im weiteren wird davon ausgegangen, daß die Busstruktur verzögerungsfrei ist – die Verzögerungen der bidirektionalen Treiber lassen sich entsprechend dem modularen Entwurf von der Verhaltensbeschreibung trennen. Wie schon beim RS-Flipflop steht auch hier die Frage nach einem tatsächlichen Verhaltensunterschied zwischen technologieunabhängigen und technologieabhängigen Beschreibungen im Raum. Die Antwort liegt bei den Busstrukturen in der Behandlung von Buskonflikten. Setzt sich im NMOS-Fall die starke Null gegenüber der schwachen Eins noch durch, so führt das Zusammentreffen einer starken Null und einer starken Eins bei CMOS zur Schaltungszerstörung (Kurzschluß). Diese Eigenschaft gilt es nun in die Modellierung von CMOS-Busstrukturen mit aufzunehmen. Die Untersuchungsergebnisse der Verfeinerungsbeziehung stellen sich für die NMOS-Spezifikation der Busstruktur als Verhaltensverfeinerung dar – die nichtdeterministische Verhaltensbeschreibung bei Buskonflikten im technologieunabhängigen Fall wird zugunsten der starken Null im technologieabhängigen Fall entschieden. Für die CMOS-Spezifikation ergibt sich nach einem Buskonflikt, bei dem unterschiedliche Daten geschrieben werden, eine aufgrund eines Kurzschlusses zerstörte Busstruktur. Deren Verhalten steht in keiner Beziehung zur technologieunabhängigen Spezifikation.

Kapitel 6 faßt die Ergebnisse dieser Arbeit zusammen und diskutiert den Umgang mit den gewonnenen Erkenntnissen. Anhand der Busstrukturen werden die zugehörigen Spezifikationen bezüglich der Verfeinerungsrelation graphisch miteinander in Beziehung gesetzt – diese Graphik wird durch eine Spezifikation, die keinerlei Aussagen über das Verhalten nach den kritischen Anwendungen macht (also mit einer Spezifikation mit weniger Eigenschaften), erweitert. Abschließend werden die Ergebnisse dieser Arbeit mit anderen Ansätzen in Verbindung gebracht und offene Fragen diskutiert.

1.3 Vergleichbare Arbeiten

Dieser Abschnitt verschafft einen knappen Überblick über weitere Hardwarebeschreibungsformalismen sowie über Arbeiten die sich mit der Beschreibung von RS-Flipflops und Busstrukturen beschäftigen. Die hier vorgestellten Formalismen erheben keinen Anspruch auf Vollständigkeit – sie entsprechen vielmehr lediglich denjenigen Formalismen, die den hier zitierten Arbeiten über RS-Flipflops und Busstrukturen zugrunde liegen. Ein umfassender Überblick über formale Hardwarebeschreibungen und Entwicklungsmethodiken findet sich in [MT90].

STREAM

STREAM ([Del87]) ist eine formale Hardwarebeschreibungssprache basierend auf Strömen und stromverarbeitenden Funktionen, sogenannten Agenten. STREAM eignet sich für die Hardwarebeschreibung auf unterschiedlichsten Abstraktionsebenen. Diese umfassen die Systemebene, die Register-Transfer-Ebene, die Gatterebene sowie die Transistorebene. Neben einer textuellen Repräsentation von Hardwarebeschreibungen bietet STREAM auch eine graphische Darstellungsform. Die Hardwarebeschreibungssprache STREAM weist

viele Gemeinsamkeiten mit der in dieser Arbeit verwendeten Entwurfsmethodik Focus auf – beide Ansätze besitzen die gleiche semantische Grundlage. Dennoch lassen sich einige wesentliche Unterschiede in Hinblick auf die Spezifikation gezeiteter Hardware ausmachen. So unterstützt STREAM als einziges Verzögerungskonzept lediglich Transportdelay und die gezeiteten Ströme enthalten sogenannte "absence values". Letztere signalisieren nur das Voranschreiten der Zeit auf einem Verbindungskanal ohne dessen aktuellen Wert anzugeben. Bezüglich der Verzögerungskonzepte bietet Focus zur Beschreibung der Verzögerungszeiten in Komponenten zusätzlich Inertialdelay-Modellierungen. Was die "absence values" betrifft, so gehen wir in Focus davon aus, daß auf einer Leitung zu jeden Zeitpunkt ein Signalwert anliegt. Diese Annahme erlaubt es uns, einfachere Spezifikationen anzugeben, da die zusätzliche Behandlung der "absence values" entfällt. Ein weiterer Vorteil von Focus liegt in der Möglichkeit nichtdeterministischer Beschreibungen. Ein typischer Anwendungsfall dafür ist die nichtdeterministische Festlegung des Initialwerts einer Komponente, der sich unmittelbar nach dem "Einschalten" am Ausgang einstellt.

SCA

Eine algebraische Methode ([TT89, Tuc91]) zur Entwicklung verteilter, synchroner Systeme, und insbesondere von Hardware, ist SCA (Synchronous Concurrent Algorithm). SCA unterstützt Systementwicklung auf unterschiedlichen Abstraktionsebenen und gestattet es, Übergänge zwischen Spezifikationen korrekt zu beweisen. Die bei SCA zugrundeliegende Modellvorstellung basiert auf deterministischen, synchronisierten Datenflußnetzen, wobei die Kommunikation mittels unendlicher Nachrichtenströme abläuft – hier zeigt sich eine enge Verbindung zu Focus. Die Synchronisation basiert auf einem globalen Takt und den damit verbundenen Taktzyklen. Der Umgang mit Zeit besitzt in SCA einen sehr expliziten Charakter, da auf alle Nachrichten nur mit einem entsprechenden Zeitpunkt, d.h. dem Zeitpunkt des Auftretens der Nachricht, zugegriffen werden kann. Entsprechend ist ein Signal eine Funktion, die eine natürliche Zahl nimmt, und den zugehörigen Wert auf dem Signal liefert. Bezüglich der Zeitkonzepte unterstützt SCA prinzipiell Transport- und Inertialdelay – Zerodelay-Modellierungen sind ausgeschlossen. Anzumerken bleibt, daß SCA-Modellierungen, in Hinblick auf Nichtdeterminismus, durch die Einführung weiterer, interagierender Taktsignale erweitert werden können.

HOL

HOL (Higher Order Logik) ist aus Milner's LCF entstanden und wurde Anfang der 80-iger Jahre von Mike Gordon ([Gor89]) an der Universität von Cambridge entwickelt. HOL erlaubt das maschinenunterstützte, interaktive Beweisen von Theoremen basierend auf einer Logik höherer Stufe. Als Kommunikationssprache mit dem Beweissystem steht die funktionale Programmiersprache ML zur Verfügung. Theoreme in HOL entsprechen Objekten von ML, die mit ML-Funktionen (Ableitungsregeln) erzeugt werden können. Das Verhalten einer Komponente wird durch ein Prädikat über die Ein- und Ausgaben beschrieben. Die Ein- und Ausgaben ihrerseits werden durch Funktionen, die jedem Zeitpunkt einen Wert zuweisen, beschrieben – dadurch kommt die Logik höherer Stufe zum Ausdruck. Die Unterschiede zu Focus sind zum einen die relationale Verhaltensmodellierung, die eine einfache Modellierung bidirektionaler Leitungen gestattet. Zum anderen besitzt HOL eine maschinelle Beweisunterstützung. HOL eignet sich sowohl für Beweise

an Software- und Hardwaresystemen; zu letzteren gehören Teilverifikationen des VIPER-Mikroprozessors ([Coh88]), der erste Mikroprozessor, für den wesentliche Teile des Entwurfs korrekt bewiesen wurden. Eine Variante von HOL, die speziell auf den Entwurf von Hardware ausgerichtet ist, stellt LAMBDA (Logik And Mathematics Behind Automation) dar ([FFH90]).

CIRCAL
CIRCAL (CIRcuit CALculus) ist eine Prozeßalgebra zur Beschreibung und Analyse von Hardwareschaltungen ([Mil85]), wobei die Analyse dazu dient, das Verhalten zweier Beschreibungen äquivalent zu beweisen. CIRCAL unterstützt alle Ebenen des Schaltungsentwurfs einschließlich der Ebene des physikalischen Layouts. Wie in CSP ([Mil80]) oder CCS ([Hoa85]), so basiert auch in CIRCAL die Prozeßkommunikation auf "Handshaking" – sie ist also synchronisiert. Die wesentlichen Charakteristika, die CIRCAL von anderen Prozeßalgebren unterscheidet, sind Eigenschaften, die speziell auf die Beschreibung von Hardware abzielen. Dazu gehört die Möglichkeit, simultane Ereignisse beschreiben zu können, wie beispielsweise die Ereignisse $\{in0, out1\}$ und $\{in1, out0\}$ zur Verhaltensbeschreibung eines Inverters. Weitere Charakteristiken von CIRCAL sind sequentielle Komposition durch "Prefixing of Actions", parallele Komposition durch "Many to Many Synchronisation", Hiding sowie interne und externe Auswahl. Die im weiteren Verlauf dieser Arbeit zitierte Arbeit von Luca Cardelli ([Car82]) basiert ebenfalls auf einer Prozeßalgebra, welche CSP sehr nahe kommt. Die Wahl, hier dennoch CIRCAL anstelle der Prozeßalgebra von Cardelli vorzustellen, liegt am Bekanntheitsgrad von CIRCAL – das Verständnis der in [Car82] zitierten Beschreibungen ist dadurch nicht gefährdet.

VHDL
VHDL (Very High Speed Integrated Hardware Description Language) ist eine von IEEE standardisierte Hardwarebeschreibungssprache ([EEE87, EEE92]). VHDL unterstützt die Beschreibung digitaler Hardware auf der Systemebene, der Register-Transfer-Ebene und der Gatterebene. Die Komponentenbeschreibungen in VHDL bestehen jeweils aus einer Schnittstellenbeschreibung (entity) und einer Verhaltensbeschreibung (architecture) – in der Verhaltensbeschreibung ist die Zeitbeschreibung mit enthalten. Verhaltensbeschreibungen umfassen algorithmische, datenflußorientierte und strukturelle Formen. Die von VHDL unterstützten Entwurfsmethoden sind "Top-down", "Bottom-up" und jede beliebige Mischform. Ein wesentliches Charakteristika der Sprache ist die Extensionalität. Dies bedeutet, daß der Benutzer die Signalwerte um beliebige Werte anreichern kann – somit lassen sich Treiberstärken unterschiedlicher Realisierungstechnologien behandeln. Bezüglich der Zeitkonzepte unterstützt VHDL Transport- und Inertialdelay sowie Deltadelay, ein Konzept zur Beschreibung verzögerungsfreier Vorgänge. VHDL basiert auf einem ereignisgesteuerten Simulationskern. Dieser ist zwar nicht Bestandteil der Sprache, aber für das Verständnis von VHDL unerläßlich. In den offiziellen VHDL-Sprachbeschreibungen ist die Semantik der Sprache nur informell gegeben. Dies stellt den deutlichsten Unterschied zur formalen Entwurfsmethodik Focus dar. In letzter Zeit wurden allerdings eine Reihe von Anstrengungen unternommen, auch für VHDL formale Semantiken anzugeben ([SDB93, Tas92, BPS92]) – diese beschränken sich aber durchwegs auf Subsprachen von VHDL.

Beschreibungen von RS-Flipflops
Ein Kriterium bei der Beurteilung von Hardwarebeschreibungssprachen bzw. Hardwareentwurfsmethodiken ist deren Eignung in Hinblick auf eine elegante Beschreibung des RS-Flipflops. Dies spiegelt sich in einer entsprechend großen Anzahl von RS-Flipflop-Beschreibungen in der Literatur wieder. Stellvertretend greifen wir in dieser Arbeit drei Beschreibungen heraus und untersuchen deren Adäquatheit und Technologieabhängigkeit.

- Die formale Grundlage zur Beschreibung des RS-Flipflops in [TT93] bildet SCA. Das Auftreten der irregulären Eingabesequenz (vgl. Kapitel 4) wird in [TT93] zwar nicht ausgeschlossen, deren Konsequenzen jedoch inadäquat modelliert. So führt die irreguläre Eingabesequenz stets zu einer Ausgabe von $(q = 0, \overline{q} = 0)$. Bezüglich der Zeitmodellierung wird von einer Verzögerung von exakt einer Zeiteinheit ausgegangen. Für diesen speziellen Fall ist das zugrundegelegte Transportdelay-Konzept ausreichend. Unserer Kritik richtet sich hier in erster Linie gegen die Modellierung des RS-Flipflopverhaltens bei auftretender irregulärer Eingabesequenz – diese ist nicht adäquat. Die Realisierungstechnologie bleibt unberücksichtigt.

- In [Del87] bildet STREAM die formale Grundlage zur Beschreibung des RS-Flipflops. Das Auftreten der irregulären Eingabesequenz (vgl. Kapitel 4) wird verboten, da auftretende Schwingungen keine korrekte Verhaltensvorhersage erlauben. Unsere Kritik richtet sich in erster Linie gegen das verwendete Zeitmodell. [Del87] verwendet für die Zeitmodellierung Transportdelay und nicht das für Verzögerungen innerhalb von Komponenten geeignete Inertialdelay. Letzteres führt für unterschiedliche Verzögerungszeiten selbst bei der irregulären Eingabesequenz zu stabilem Ausgabeverhalten. Das Verbot der irregulären Eingabesequenz aufgrund nicht vorhersehbarer Ausgabewerte kann durch eine differenziertere Behandlung der einzelnen Verzögerungszeiten widerlegt werden. Die Realisierungstechnologie bleibt in der RS-Flipflop-Beschreibung in [Del87] unberücksichtigt.

- Die in [Car82] zugrundeliegende formale Basis zur Beschreibung von RS-Flipflops ist eine Prozeßalgebra ähnlich CSP. Die Zeitbehandlung ist kontinuierlich und basiert auf dem Transportdelay-Konzept. Dieses ist nicht geeignet zur Beschreibung von Verzögerungen in Komponenten und führt im Falle der RS-Flipflops bereits bei Betrachtungen von zu kurzen Setz- bzw. Rücksetzimpulsen (kleiner als die Verzögerungszeit beider NOR-Gatter) zu falschen Aussagen. Die irreguläre Eingabesequenz (vgl. Kapitel 4) wird bei der Beschreibung des RS-Flipflops in [Car82] nicht ausgeschlossen – ausgeschlossen wird allerdings die Behandlung der Konsequenzen. Unsere Kritik konzentriert sich bei dieser Arbeit auf die Verwendung eines inadäquaten Verzögerungskonzepts sowie auf den expliziten Ausschluß der Behandlung von irregulären Eingabesequenzen. Die Realisierungstechnologie bleibt auch hier unberücksichtigt.

Beschreibungen von Busstrukturen
Beschreibungen von Busstrukturen treten in der Literatur meist in Verbindung mit Beschreibungen von Mikroprozessoren auf. Die nachfolgenden Arbeiten erheben keinen Anspruch auf Vollständigkeit, sie stellen allerdings einen, nach unserer Meinung, repräsentativen Ausschnitt aller Busstrukturspezifikationen dar.

- Die in [Hoo94, Hoo92, Her92] vorgestellten Ansätze zur Modellierung von Busstrukturen basieren auf HOL. In [Hoo94, Hoo92] wird eine Busstruktur spezifiziert, deren Verhalten vom IEEE 896 Futurebus ([EEE88]) bestimmt wird. Die zugrundeliegende Idee basiert darauf, daß sich eine Menge von Hardwarekomponenten um das Zugriffsrecht auf den Bus bewirbt, und nur diejenige Komponente, die den Zuschlag erhält, auf den Bus schreibend zugreifen darf. Dadurch werden Buskonflikte ausgeschlossen. In [Her92] wird eine Busstruktur für einen mikroprogrammierbaren Prozessor spezifiziert. Zentraler Bestandteil der Spezifikation ist ein Prädikat, das Buskonflikte ausschließt, also nur unter der Annahme, daß keine Buskonflikte auftreten, ist diese Spezifikation korrekt. Unsere Kritik richtet sich in allen Fällen gegen den unbegründeten Ausschluß von Buskonflikten. Die hier betrachteten Busspezifikationen berücksichtigen die Realisierungstechnologie nicht.

- In [HFFM92] bildet LAMBDA die formale Grundlage. Behandelt wird der Entwurf einer Tristate-Busstruktur unter der Annahme, daß keine Buskonflikte auftreten dürfen. Dies stellt sich insbesondere in [HFFM92] als kritische Anforderung heraus, da auch eine Verbindung vom Bus zur Umgebung besteht. Unsere Kritik richtet sich erneut gegen den unbegründeten Ausschluß von Buskonflikten, was sich bei einer Berücksichtigung externer Anschlüsse an den Bus als besonders kritisch erweist. Auch hier bleibt die Realisierungstechnologie unberücksichtigt.

Kapitel 2

Grundlagen

Dieses Kapitel behandelt die mathematischen Grundlagen der Entwurfsmethodik Focus [BDD+92a]. Neben Strömen, stromverarbeitenden Funktionen und Spezifikationen behandeln wir einen Verfeinerungsbegriff, der es erlaubt, Spezifikationen miteinander in Beziehung zu setzen. Abschließend führen wir tabellarische Beschreibungsformen ein.

2.1 Focus

Die Entwurfsmethodik Focus ermöglicht eine formale, durchgängige Entwicklung verteilter, reaktiver Systeme ausgehend von einer ersten Anforderungsspezifikation über mehrere Zwischenschritte hinweg bis hin zu parallel ablaufenden Programmen. Die verwendeten Beschreibungstechniken unterstützen eine schrittweise und modulare Systementwicklung, wobei stets die methodischen Aspekte im Vordergrund stehen. Modular bedeutet in diesem Zusammenhang, daß Entwurfsentscheidungen an den Stellen korrekt bewiesen werden können, an denen sie gefällt werden, und daß Komponenten isoliert entwickelt und anschließend an jeder beliebigen Stelle der Systembeschreibung eingesetzt werden dürfen. Focus gliedert den Entwurfsprozeß in drei Phasen, wobei die Formalismen in den einzelnen Phasen so aufeinander abgestimmt sind, daß der Übergang zwischen den einzelnen Phasen problemlos bewältigt werden kann.

In der ersten Phase wird eine Anforderungsspezifikation erstellt. Diese formalisiert das vom Kunden extensional beobachtbare Systemverhalten unter Verwendung von stromverarbeitenden Funktionen oder mittels endlicher oder unendlicher Sequenzen von geeigneten Systemaktionen, sogenannten Spuren ([Web92]). Die zweite Phase umfaßt die Entwicklung einer Entwurfsspezifikation, wobei hier der interne Aufbau des zu entwerfenden Systems, basierend auf fest vorgegebenen Schnittstellen, im Mittelpunkt steht. Durch schrittweise Verfeinerung entsteht ein Netz von Systemkomponenten, die über Verbindungskanäle asynchron miteinander kommunizieren. Die Spezifikationen einzelner Systemkomponenten basieren in dieser Phase auf einer Menge von stromverarbeitenden Funktionen. Den Abschluß des Entwicklungsprozesses stellt die Implementierungsphase ([Ded92]) dar. Hier werden funktionale und prozedurale Programme entwickelt, die eine formal korrekte Implementierung der auf der Entwurfsebene entwickelten Spezifikationen darstellen. In dieser Phase steht nicht mehr die Systementwicklung im allgemeinen sondern die Entwicklung effizienter Programme im Vordergrund.

Der Entwurf verteilter Systeme umfaßt auch den Entwurf digitaler Hardware, da die Hardwarebausteine als eigenständige Systemkomponenten, die über Leitungen miteinander kommunizieren, verstanden werden können. Unter zusätzlichen Annahmen, die im weiteren Verlauf dieser Arbeit noch festzulegen sind, läßt sich daher auch die Entwurfsmethodik Focus für die Entwicklung digitaler Hardware einsetzen. Da sich im allgemeinen die Kundenwünsche im Bereich digitaler Hardware stets auf Bausteine mit fest vorgegebener Schnittstelle beziehen, bietet sich die Entwurfsspezifikation als eine adäquate Beschreibungsebene an. Demzufolge konzentrieren wir uns in dieser Arbeit ausschließlich auf die Beschreibung digitaler Hardware, zum Zwecke technologieabhängiger Untersuchungen, auf der Entwurfsebene. Im folgenden werden die mathematischen Grundlagen für Entwurfsspezifikationen erarbeitet.

2.2 Ströme

Zur Modellierung der Kommunikationsgeschichte zwischen Systemkomponenten verwenden wir Ströme. Diese repräsentieren die Geschichte von Nachrichten oder von Aktionen im System. In Hinblick auf digitale Hardware umfaßt die Menge der Aktionen die Signalpegel auf den Verbindungsleitungen zwischen Hardwarekomponenten. Diese sind die digitalen Signalpegel *low* und *high* einschließlich technologieabhängiger Varianten, wie z.B. den schwachen Highpegel H zur Beschreibung von NMOS-Schaltkreisen. Formal werden Ströme über eine Menge M wie folgt definiert:

$$M^{\omega} = M^{*} \cup M^{\infty}$$

Dabei bezeichnet M^{*} die Menge aller endlichen Ströme, M^{∞} die Menge aller unendlichen Ströme und M^{ω} die Vereinigung der endlichen und unendlichen Ströme. Der leere Strom wird mit ϵ bezeichnet.

Um nichtterminierende Berechnungen formalisieren zu können, nehmen wir ein Pseudoelement $\perp$ (Bottom), welches nicht in M vorkommt, zur Aktionsmenge hinzu. Wir setzen ferner $M^{\perp} = M \cup \{\perp\}$ und führen auf $M^{\perp}$ eine flache Ordnung mit $\perp$ als kleinstes Element ein (vgl. hierzu [Bro90]).

Für den Umgang mit Strömen sind Konstruktoren, Selektoren, Operatoren sowie eine Relation definiert. Im folgenden notieren wir Funktionsapplikationen durch $f.x$ oder $f(x)$ und mit $\mathbb{N}$ bezeichnen wir die natürlichen Zahlen einschließlich der 0. Funktionsapplikationen der Form $g.f.x$ sind rechtsassoziativ und entsprechen $g(f.x)$.

Konstruktoren:
Sei $a \in M^{\perp}$ und $s \in M^{\omega}$:

ϵ bezeichnet den leeren Strom.

$a\&s$ hängt vorne an den Strom s die Aktion a an. Entspricht a dem Pseudoelement $\perp$, so ist der resultierende Strom ϵ.

Selektoren:
Sei $s \in M^\omega$ und $k \in \mathbb{N}$:

$ft.s$ bezeichnet das erste Element im Strom s, falls dieser nicht leer ist, $\perp$ sonst.

$rt.s$ bezeichnet den Strom s verringert um das erste Element. rt angewandt auf den leeren Strom liefert den leeren Strom.

$rt^k.s$ bezeichnet die k-fache Anwendung der Funktion rt auf den Strom s.
$rt^0.s$ entspricht dem Strom s.

Operatoren:
Sei $s, t \in M^\omega; a \in M$ und $k \in \mathbb{N}$:

$s \circ t$ bezeichnet die Konkatenation der Ströme s und t, wobei der resultierende Strom mit s beginnt und mit t fortsetzt. Ist $s \in M^\infty$, so ist der resultierende Strom identisch mit s. Der leere Strom ϵ ist bezüglich der Konkatenation das neutrale Element.

a^k bezeichnet denjenigen Strom, der durch k-faches Hintereinanderhängen der Aktion a entsteht. a^0 entspricht dem leeren Strom.

$\#s$ bezeichnet die Anzahl der Elemente (Länge) in s, falls $s \in M^*$, ∞ sonst.

$s|_k$ bezeichnet den Präfix von s der Länge k, falls $\#s > k$, s sonst.

$a ⓒ s$ bezeichnet denjenigen Teilstrom von s, der dadurch entsteht, daß ausschließlich die Elemente a aus s herausgefiltert werden (Filteroperator).

Relation:
Sei $s, t, u \in M^\omega$:

$s \sqsubseteq t$ bezeichnet eine partielle Ordnung, genannt Präfixordnung auf Strömen. Strom s ist Präfix des Stromes t, falls es einen Strom u gibt, sodaß folgende Formel gilt:

$$s \sqsubseteq t \equiv \exists u \in M^\omega : s \circ u = t$$

Ströme mit der oben angegebenen partiellen Ordnung $\sqsubseteq$ bilden einen Bereich (vollständige Halbordnung) [BDD+92a] mit ϵ als kleinstes Element und den Strömen aus M^∞ als Abschlüsse. Diese Tatsache erlaubt es uns, die Semantik von Stromrekursionen (Rückkopplungsschleifen im Netz) basierend auf präfixstetigen Funktionen unter Verwendung der Fixpunkttheorie präzise zu definieren. Die Präfixordnung erlaubt auch eine intuitive Deutung, nämlich Ströme bezüglich ihres Informationsgehalts zu vergleichen (Approximationsordnung). Gilt $s \sqsubseteq r$, dann trägt r zumindest die Information, die auch in s steckt, möglicherweise aber auch mehr.

Erweiterungen auf Tupel:
Neben Strömen wollen wir auch Tupel von Strömen betrachten. Insbesondere sollen die Tupel flach sein, d.h. beim Zusammenfügen eines n-stelligen und eines m-stelligen Tupels erzeugen wir ein $n+m$-stelliges Tupel. Im weiteren Verlauf bezeichnen wir ein n-stelliges Tupel von Strömen $M_1^\omega \times \ldots \times M_n^\omega$ mit $M_{1:n}^\omega$. Neben dem für das Zusammenfügen von Tupeln notwendigen Operator erweitern wir einen Teil der oben eingeführten Konstruktoren und Selektoren kanonisch auf Tupel von Strömen.
Sei $t_n \in A_{1:n}^\omega; t_m \in B_{1:m}^\omega$ und $a \in A_1 \times \ldots \times A_n$:

$t_n \oplus t_m$ fügt ein n-stelliges, flaches Tupel und ein m-stelliges, flaches Tupel zu einem $n+m$-stelligen, flachen Tupel zusammen. Wir überladen den Operator $\oplus$ derart, daß er zwei Einzelströme zu einem 2-stelligen Tupel bzw. auch Tupel und Einzelströme geeignet zusammenbaut.

$ft.t_n$ bezeichnet die elementweise Anwendung der ft-Funktion auf alle Ströme im Tupel t_n. Das resultierende Tupel ist undefiniert, falls zumindest ein Strom im Tupel t_n der leere Strom ist.

$rt.t_n$ bezeichnet die elementweise Anwendung der rt-Funktion auf alle Ströme im Tupel t_n.

$a \frown t_n$ bezeichnet die auf Tupel erweiterte &-Funktion, wobei das resultierende Tupel von Strömen durch elementweise Anwendung von & auf alle Elemente in a und den korrespondierenden Strömen in t_n entsteht.

Bezüglich der Präfixordnung wird die Erweiterung auf Tupel von Strömen ebenfalls elementweise durchgeführt. Den Projektionsoperator π_i wollen wir ebenfalls auf Tupel von Strömen erweitern – für ein Tupel von Strömen t_n filtert $\pi_i.t_n$ den i-ten Strom aus dem Tupel heraus. Als Kurzschreibweise wollen wie für ein n-Tupel von Strömen gleichen Typs M^ω auch $(M^\omega)^n$ notieren.

2.3 Stromverarbeitende Funktionen

Die oben vorgestellten Ströme stellen ein wichtiges Konzept zur Modellierung verteilter Systeme und somit auch zur Modellierung digitaler Hardware dar. Zusammen mit den stromverarbeitenden Funktionen bilden sie die semantische Grundlage dieser Arbeit. Stromverarbeitende Funktionen sind präfixstetige Abbildungen die Tupel von Strömen in Tupel von Strömen abbilden. Eine stromverarbeitende Funktion, die ein n-stelliges Tupel von Strömen in ein m-stelliges Tupel von Strömen abbildet, läßt sich nun wie folgt aufschreiben.

$$f : A_{1:n}^\omega \to B_{1:m}^\omega$$

Die Stetigkeit stromverarbeitender Funktionen schließt die Monotonie ein. Monotonie beschreibt die Eigenschaft, daß bereits ausgegebene Aktionen nicht mehr zurückgenommen werden können. Verlängert man die Eingabe, so kann dies allenfalls dazu führen, daß auch

die Ausgabe verlängert wird. Eine weitere Konsequenz der Monotonie gestattet den parallelen Ablauf mehrerer stromverarbeitender Funktionen in einem Netz – jede stromverarbeitende Funktion kann ihre Berechnung bereits beginnen, auch wenn nur ein Teilstück der Eingabe vorliegt. Dies spiegelt den Kausalzusammenhang zwischen Ein- und Ausgabe wieder. Formal läßt sich die Monotonie für eine stromverarbeitende Funktion f und für Tupel von Strömen $s, t \in T^{\omega}_{1:n}$ wie folgt aufschreiben:

$$s \sqsubseteq t \Rightarrow f.s \sqsubseteq f.t$$

Stetigkeit beschreibt die Eigenschaft, daß die Ausgabe einer stromverarbeitenden Funktion für einen unendlichen Eingabestrom bereits aufgrund der Ausgaben aller endlichen Approximationen der unendlichen Eingabe bestimmt werden kann. Daraus läßt sich folgern, daß für endliche Ausgaben auch nur endliche Eingaben benötigt werden – jede einzelne Ausgabeaktion ist demzufolge eine Reaktion auf höchstens endlich viele Eingaben. Formal läßt sich die Stetigkeit für eine monotone, stromverarbeitendende Funktion und eine gerichtete Menge $S \subseteq T^{\omega}_{1:n}$ wie folgt definieren:

$$lub\ \{f.s : s \in S\} = f(lub\ S)$$

Zu beachten ist, daß $\&, rt$, und ft stetig sind – die Konkatenation $\circ$ hingegen nicht einmal monoton ist.

In dieser Arbeit wollen wir neben Tupel von Strömen auch Zustände ([BDD+92a]) in der Parameterliste von stromverarbeitenden Funktionen erlauben. Diese Funktionen mit Zuständen ließen sich auch durch stromverarbeitende Funktionen im obige Sinne beschreiben, was allerdings zu Kosten der Lesbarkeit gehen würde. Die stromverarbeitenden Funktionen mit Zuständen erlauben es also, Systemzustände mit in die Eingabeparameterliste aufzunehmen, wobei wir hier ein Tupel von Zuständen $Z_1 \times \ldots \times Z_k$ mit Z bezeichnen wollen. Damit haben stromverarbeitenden Funktionen mit Zuständen folgende Form:

$$f : A^{\omega}_{1:n} \times Z \rightarrow B^{\omega}_{1:m}$$

Als weiteren Schritt in Richtung bessere Lesbarkeit wollen wir analog zu [HT90] erlauben, bestimmte Parameter, die einen besonderen Stellenwert einnehmen (z.B. keine Schnittstellenparameter der zu modellierenden Komponente darstellen), besonders zu kennzeichnen. Dies geschieht durch die Herausnahme aus der Parameterliste und einem Plazieren in einer Indexparameterliste. Formal sind beide Darstellungen äquivalent.

$$f(a, x) = f_{(a)}(x)$$

Nun wollen wir uns der Definition von Netzwerken stromverarbeitender Funktionen, die selbst als stromverarbeitende Funktionen betrachtet werden können, zuwenden – dadurch wird die Strukturierung von Spezifikationen auf der Entwurfsebene erst möglich. Neben den klassischen Kompositionsoperatoren für sequentielle und parallele Komposition sowie dem Rückkopplungsoperator betrachten wir eine äquivalente Darstellungsform basierend auf Gleichungsdefinitionen. Beginnen wollen wir mit den klassischen Kompositionsoperatoren, die in Abbildung 2.1 graphisch dargestellt sind.

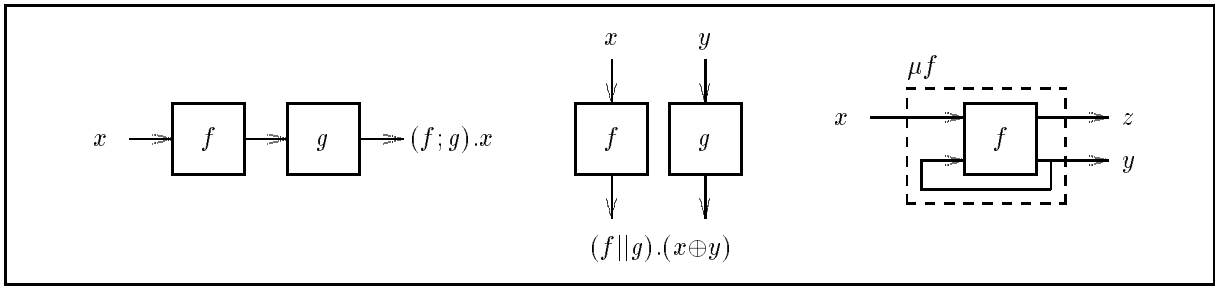


Abbildung 2.1: Kompositionsoperatoren: Sequentiell – Parallel – Rückkopplung.

Sei $f \in A^{\omega}_{1:n} \to B^{\omega}_{1:m}$ und $g \in B^{\omega}_{1:m} \to C^{\omega}_{1:k}$. Für $x \in A^{\omega}_{1:n}$ bezeichnet $(f;g).x$ die sequentielle Komposition von f und g und ist wie folgt definiert:

$$(f;g).x = g(f.x)$$

Sei $f \in A^{\omega}_{1:n} \to B^{\omega}_{1:m}$ und $g \in C^{\omega}_{1:k} \to D^{\omega}_{1:l}$. Für $x \in A^{\omega}_{1:n}$ und $y \in C^{\omega}_{1:k}$ bezeichnet $(f||g).(x \oplus y)$ die parallele Komposition von f und g und ist wie folgt definiert:

$$(f||g).(x \oplus y) = f.x \oplus g.y$$

Sei f eine $(n+1, m)$-stellige, stromverarbeitende Funktion mit $m \geq 1$, wobei $n+1$ der Eingabestelligkeit und m der Ausgabestelligkeit von f entspricht. μf ist eine von f abgeleitete, (n, m)-stellige, stromverarbeitende Funktion mit einer Rückkopplungsschleife.

Wir definieren für ein n-Tupel (von Strömen) $x \in A^{\omega}_{1:n}$
$(\mu f).x = (z \oplus y)$
falls $(z \oplus y)$ der kleinste Fixpunkt der Gleichung
$(z \oplus y) = f(x \oplus y)$
ist, wobei z ein $(m-1)$-Tupel (von Strömen) und y ein Strom ist.

Formal bezeichnet $(\mu f).x$ den kleinsten Fixpunkt relativ zu x. Das Fixpunktkonzept modelliert die Kommunikation über Rückkopplungsleitungen in geeigneter Weise. Dabei werden Ausgaben einer stromverarbeitenden Funktion zu den Eingaben dieser stromverarbeitenden Funktion zurückgesandt. Falls für Ausgaben mehr Eingabedaten notwendig wären, als vorhanden, so würde dies dazu führen, daß keine weitere Ausgabe erzeugt würde – dies wird durch die Wahl des kleinsten Fixpunkts modelliert.

Für monotone, stromverarbeitende Funktionen zusammen mit den Strömen als vollständige Halbordnung existiert bereits nach Knaster/Tarski ([BW81]) der kleinste Fixpunkt. Für stetige, stromverarbeitende Funktionen zusammen mit den Strömen als vollständige Halbordnung läßt sich der kleinste Fixpunkt entsprechend der Kleene-Kette ([BW81]) durch wiederholte Funktionsanwendung berechnen – dies entspricht genau der schrittweisen Kommunikation zwischen Komponenten.

Netzwerkstrukturen verteilter Systeme basierend auf stromverarbeitenden Funktionen lassen sich auch durch Gleichungssysteme beschreiben. Die Semantik der Gleichungssysteme von Netzwerken ist der kleinste Fixpunkt der das Gleichungssystem erfüllt – somit besitzen Netzstrukturen, die durch Gleichungssysteme oder durch die klassischen Kompositionsoperatoren beschrieben sind, die gleiche semantische Basis, sind also ineinander überführbar.

An dieser Stelle wollen wir der Vollständigkeit halber darauf hinweisen, daß zur Festlegung der Semantik von Funktionsrekursionen (rekursiver Aufruf stromverarbeitender Funktionen) ebenfalls auf die Fixpunkttheorie zurückgegriffen wird [BW81]. Die dafür notwendige Halbordnung auf Funktionen ist die "less defined"-Ordnung.

2.4 Spezifikation

Eine funktionale Spezifikation in Focus ist die Beschreibung der Eigenschaften einer Komponente unter Verwendung von stromverarbeitenden Funktionen. Technisch geschieht dies durch die Angabe eines Prädikats, das eine Menge von stromverarbeitenden Funktionen festlegt, die exakt das zu spezifizierende Verhalten wiederspiegeln. Spezifikationen in Focus umfassen sowohl eine syntaktische als auch eine semantische Schnittstelle. Die syntaktische Schnittstelle besteht aus Eingabekanälen, Zustandsparameter und den entsprechenden Typinformationen. Für diese Arbeit ergibt sich folgendes syntaktische Spezifikationsschema.

pred "Typinformationen zur Spezifikation S"

spec $S(x).f \equiv F$

Durch das Schlüsselwort **pred** werden die Typinformationen der entsprechenden Spezifikation, also auch die Typinformationen der stromverarbeitenden Funktionen f, festgelegt. **spec** signalisiert den Beginn der eigentlichen Spezifikation, wobei folgende Festlegungen gelten:

S ist der Spezifikationsname.

(x) ist eine optionale Liste von Spezifikationsparametern (realisierungsbedingte Größen wie z.B. Gatterlaufzeiten).

f ist der Bezeichner der stromverarbeitenden Funktionen.

F ist eine prädikatenlogische Formel, die die Eigenschaften der stromverarbeitenden Funktionen festlegt.

Die Liste der Spezifikationsparameter ist optional. Entfällt sie, so handelt es sich bei der Spezifikation S im mathematischen Sinne um ein Prädikat – S legt die Eigenschaften von stromverarbeitenden Funktionen gemäß F fest. Werden jedoch Spezifikationsparameter verwendet, so ist S im mathematischen Sinne ein parametrisiertes Prädikat, also eine Abbildung von den Spezifikationsparametern in ein Prädikat, welches dann letztendlich die Festlegung der Eigenschaften von stromverarbeitenden Funktionen bewerkstelligt.

Als Beispiel für eine syntaktische Schnittstelle wollen wir die eines NMOS-Inverters angeben. Dabei sind In die Menge der Eingabeaktionen, Out die Menge der Ausgabeaktionen und $del \in \mathbb{N}$ die Verzögerungszeit des Inverters.

$$\textbf{pred } INV_{nmos} : \mathbb{N} \rightarrow [(In^{\omega} \rightarrow Out^{\omega}) \rightarrow Bool]$$

$$\textbf{spec } INV_{nmos}(del).f \equiv \ F$$

Bezeichnet der Spezifikationsname S eine technologieabhängige Spezifikation, so wird die entsprechende Technologie ($nmos$ oder $cmos$) dem Spezifikationsnamen als Index hinzugefügt. Bei technologieunabhängigen Spezifikationen entfällt diese Kennzeichnung.

Die semantische Schnittstelle der Spezifikation wird durch eine Menge von stromverarbeitenden Funktionen charakterisiert. Dies erfordert die Instanziierung der Spezifikationsparameter, sofern vorhanden. Ohne diese Instanziierung wäre die semantische Schnittstelle durch eine Menge von Mengen von stromverarbeitenden Funktionen gegeben. Formal läßt sich die semantische Schnittstelle wie folgt beschreiben:

$$[\![\ S(x)\ ]\!] \equiv \{f \in A^{\omega}_{1:n} \rightarrow B^{\omega}_{1:m} | S(x).f\}$$

Ist die Menge der durch $S(x)$ beschriebenen, stromverarbeitenden Funktionen leer, so bezeichnen wir die Spezifikation S als inkonsistent – ansonsten als konsistent. Besteht die festgelegte Menge von stromverarbeitenden Funktionen genau aus einer stromverarbeitenden Funktion, so sprechen wir von einer deterministischen Spezifikation, besteht sie aus mehreren stromverarbeitenden Funktionen, so sprechen wir von einer nichtdeterministischen Spezifikation.

Der Begriff des Nichtdeterminismus fällt in Focus mit dem Begriff der Unterspezifikation zusammen ([Bro92b, Bro92a]). Nichtdeterminismus entspricht mehr der operationellen Denkweise, zwischen unterschiedlichen Alternativen während der Berechnung auszuwählen. Unterspezifikation hingegen orientiert sich an der mathematischen Denkweise, mehrere Funktionen, die ein Prädikat erfüllen, zur Auswahl zu haben. Die einheitliche Betrachtungsweise beider Varianten rechtfertigt sich dadurch, daß nach außen nicht festzustellen ist, welches Prinzip zur Anwendung kommt. So ist es möglich, die Auswahl einer Alternative entweder während des Entwurfsprozesses aufgrund von Designentscheidungen (Einschränken bzw. Eliminieren der Unterspezifikation) oder während der Ausführungszeit entsprechender lauffähiger Systeme aufgrund nichtdeterministischer Entscheidungen durchzuführen. Diese Gleichstellung von Nichtdeterminismus und Unterspezifikation erlaubt es, den Nichtdeterminismus, ohne Angabe einer zugehörigen operationellen Semantik, auf elegante Weise zu behandeln. Für entsprechende Spezifikationen in dieser Arbeit werden wir bevorzugt den Begriff des Nichtdeterminismus und nicht den der Unterspezifikation verwenden, da intuitiv die Vorstellung zu Grunde liegt, daß sich der Nichtdeterminismus dadurch auflöst, daß eine der möglichen Alternativen zur "Realisierungszeit" der Hardware ausgewählt wird.

Abschließend wollen wir uns auch auf der Ebene der Spezifikationen mit dem Aufbau von Netzwerken beschäftigen. In Anlehnung an [BDD+92a] lassen sich für diesen Zweck die Kompositionsoperatoren für stromverarbeitende Funktionen auf Spezifikationen liften. Für Spezifikationen P und Q geeigneten Typs gilt:

$$\begin{aligned}
[\![P;Q]\!] &\equiv \{f \mid f = g;h \ \wedge \ g \in [\![P]\!] \ \wedge \ h \in [\![Q]\!]\} \\
[\![P||Q]\!] &\equiv \{f \mid f = g||h \ \wedge \ g \in [\![P]\!] \ \wedge \ h \in [\![Q]\!]\} \\
[\![\mu P]\!] &\equiv \{f \mid f = \mu g \ \wedge \ g \in [\![P]\!]\}
\end{aligned}$$

Man beachte, daß die Semantik des Gesamtsystems für deterministische Spezifikationen auf [Kah74] und für nichtdeterministische Spezifikationen auf [Bro89] zurückgeht.

Ein Problem in stromverarbeitenden, funktionalen Methoden und somit auch in Focus ist die Spezifikation einer nicht strikten, fairen Mischkomponente. Seien zwei Eingabeströme beliebiger, aber endlicher Länge gegeben, so möchte man eine Komponente spezifizieren, die alle Elemente der beiden Eingabeströme fair in einen Ausgabestrom mischt. Eine solche Komponente läßt sich durch keine stetige, ja sogar monotone, stromverarbeitende Funktion realisierten ([BDD+92a]), da stromverarbeitende Funktionen nicht in der Lage sind zwischen vollständigen endlichen und partiellen Eingabeströmen zu unterscheiden. Für die Spezifikation digitaler Hardwareschaltungen in dieser Arbeit stellt sich dieses Problem nicht. Dies liegt daran, daß wir Zeit als wesentlichen Faktor zur Beschreibung digitaler Systeme mit in unsere semantischen Modelle aufnehmen werden. So werden wir ausschließlich stromverarbeitende Funktionen betrachten, die nicht mehr asynchron sondern synchron, d.h. Zeitscheibe für Zeitscheibe, ihre Eingaben abarbeiten und somit nicht in das Problem laufen, vollständige endliche und partielle Eingabeströme unterscheiden zu müssen. Detailiertere Betrachtungen zum Zeitbegriff in Strömen und stromverarbeitenden Funktionen finden sich in Kapitel 3.

An dieser Stelle wollen wir darauf hinweisen, daß das innerhalb von Spezifikationen verwendete "where"-Konstrukt die Konjunktion Boolescher Terme syntaktisch durch das Komma "," repräsentiert. Das where-Konstrukt dient der lokalen Definition von Funktionen und Variablen. Die in den Funktionen verwendeten Parameter sind implizit $\forall$-quantifiziert. Ist die Parameterliste der lokal vereinbarten Funktion jedoch im Vergleich zur aufrufenden Funktion erweitert, so wollen wir in dieser Arbeit die Quantifizierung explizit aufschreiben.

2.5 Verfeinerung

Der Begriff der Verfeinerung ([Bro92b, Bro92a]) spielt eine zentrale Rolle in Focus. Dieser Begriff setzt zwei Spezifikationen auf unterschiedlichen Abstraktionsebenen in Beziehung, nämlich eine abstrakte, möglicherweise nichtdeterministische Spezifikation mit einer konkreteren, bezüglich des Nichtdeterminismus möglicherweise eingeschränkten Spezifikation. Das Konzept der Verfeinerung erlaubt eine Zerlegung des Entwurfsprozesses in eine Vielzahl von Einzelschritten. Das Verfeinerungskonzept in Focus ist kompositional und basiert auf der logischen Implikation. Prinzipiell lassen sich folgende Anwendungsbereiche für die Verfeinerung angeben:

- Verhaltensverfeinerung
- Aktionsverfeinerung
- Schnittstellenverfeinerung
- Strukturelle Verfeinerung

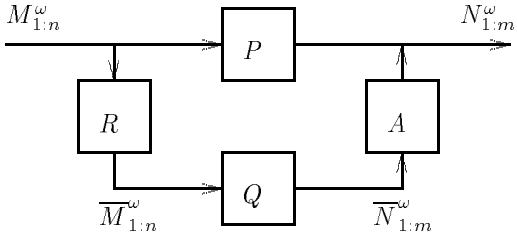
Die letzten beiden Anwendungsbereiche, die Schnittstellenverfeinerung und die strukturelle Verfeinerung, kommen in dieser Arbeit nicht unmittelbar zum Tragen, seien aber der Vollständigkeit halber erwähnt. Die Schnittstellenverfeinerung erlaubt es, die Anzahl der Ein- und Ausgabekanäle zu variieren. Die strukturelle Verfeinerung ermöglicht die Zerlegung einer Komponente in ein Netz mehrerer Komponenten.

Die Verhaltensverfeinerung setzt das Verhalten zweier Spezifikationen in Beziehung. Formal gilt für eine Spezifikation P und einer entsprechenden Verfeinerung Q folgendes:

$$Q \Rightarrow P$$

Intuitiv bedeutet dies, falls Q eine Verhaltensverfeinerung von P ist, so muß jedes Verhalten, das Q zeigt, auch in P möglich sein. Insbesondere stellt Q eine Verhaltensverfeinerung von P dar, falls die nichtdeterministische Auswahlmöglichkeit gegenüber P eingeschränkt wird.

Die Aktionsverfeinerung erlaubt Manipulationen an den Aktionsmengen. Zum einen beinhaltet dies eine Änderung der Aktionsgranularität, zum anderen umfaßt dies aber auch einfache Umbenennungen von Aktionen. Um die Aktionsverfeinerung auf die Bedürfnisse dieser Arbeit anzupassen, nehmen wir die Verhaltensverfeinerung hinzu und erhalten so eine Kombination aus beiden. Formal stellt sich die so entstandene Verfeinerungsvariante wie folgt dar:

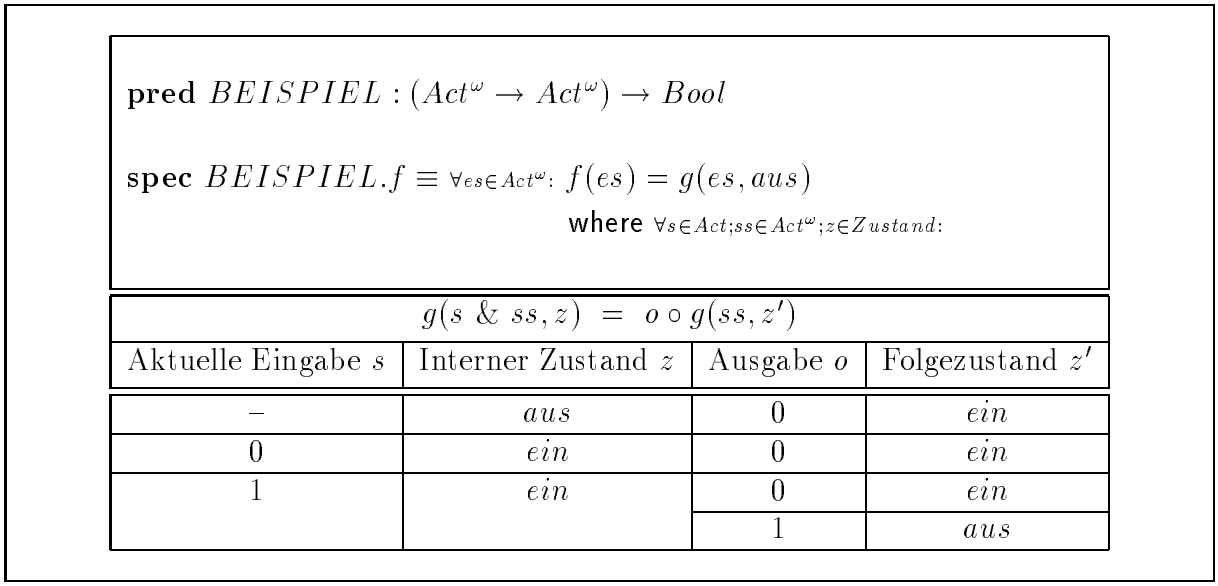


$$R; Q; A \Rightarrow P$$

Die zusätzlich eingeführten Prädikate R und A bezeichnen ein Repräsentations- und ein Abstraktionsprädikat. Das Repräsentationsprädikat R setzt ein Tupel von Eingabeströmen $M^{\omega}_{1:n}$ der abstrakteren Ebene mit einem Tupel von Eingabeströmen $\overline{M}^{\omega}_{1:n}$ der konkreteren Ebene in Beziehung. Das Abstraktionsprädikat A setzt ein Tupel von Ausgabeströmen $\overline{N}^{\omega}_{1:m}$ der konkreteren Ebene mit einem Tupel von Ausgabeströmen $N^{\omega}_{1:m}$ der abstrakteren Ebene in Beziehung. Die Formel $R; Q; A \Rightarrow P$ besagt, daß die Menge der Funktionen, die das durch sequentielle Komposition entstandene Prädikat $R; Q; A$ erfüllen, in der Mengeninklusion $\subseteq$ zu der Menge der stromverarbeitenden Funktionen stehen, die P erfüllen.

2.6 Tabellarische Spezifikation

Eine für funktionale Spezifikationen gut lesbare Notation ist die tabellarische Beschreibungsform. Die dafür notwendige semantische Grundlage, basierend auf stromverarbeitenden Funktionen, erörtern wir im folgenden. Ausgangspunkt einer tabellarischen Spezifikation sind funktionale, möglicherweise nichtdeterministische Spezifikationen, die stromverarbeitende Funktionen charakterisieren, deren Ausgabeverhalten von aktuellen Eingaben sowie internen Zuständen abhängen. Entsprechend umfaßt eine Tabelle die aktuellen Eingaben, die aktuellen internen Zustände, die daraus resultierenden Ausgaben sowie die für den nächsten rekursiven Aufruf sich neu ergebenden internen Zustände. Die so charakterisierten Tabellen lassen sich nun Zeile für Zeile in logische Ausdrücke umsetzen, indem basierend auf logischer Implikation die aktuellen Eingaben sowie die internen Zustände der Premisse und die Funktionsrekursion zuzüglich der Ausgaben und der neuen internen Zustände der Konklusion zugeordnet werden. Die Behandlung von Nichtdeterminismus erfolgt durch Disjunktion entsprechender logischer Ausdrücke. Abbildung 2.2 veranschaulicht dies anhand eines fiktiven Beispiels für die Aktionsmenge $Act = \{0, 1\}$ und die Zustandsmenge $Zustand = \{ein, aus\}$. Dieses Beispiel beschreibt eine Komponente, die im eingeschalteten Zustand auf die Eingabe 1 entweder eine 0 ausgibt und eingeschaltet bleibt, oder eine 1 ausgibt und sich selbst ausschaltet. Die Eingabe einer 0 im eingeschalteten Zustand führt zur Ausgabe einer 0 und unverändertem internen Zustand. Ist die Komponente ausgeschaltet, so wird sie durch jede Eingabe in den eingeschalteten Zustand überführt – die Komponente gibt dabei eine 0 aus.



pred $BEISPIEL : (Act^\omega \to Act^\omega) \to Bool$

spec $BEISPIEL.f \equiv \forall_{es \in Act^\omega}: f(es) = g(es, aus)$
where $\forall_{s \in Act; ss \in Act^\omega; z \in Zustand}:$

$g(s \;\&\; ss, z) \;=\; o \circ g(ss, z')$			
Aktuelle Eingabe s	Interner Zustand z	Ausgabe o	Folgezustand z'
–	aus	0	ein
0	ein	0	ein
1	ein	0	ein
		1	aus

Abbildung 2.2: Schematischer Aufbau einer tabellarischen Spezifikation.

Syntaktisch zerfällt eine tabellarische Spezifikation in drei Teile. Der erste Teil entspricht der syntaktischen Schnittstelle einer Spezifikation – diese ist unabhängig davon, ob die Semantik durch eine Tabelle oder eine logische Formel festgelegt wird. Die letzten beiden Teile dienen der Festlegung der Semantik – diese Teile sind speziell auf eine tabellarische Darstellungsform ausgerichtet. Man beachte, daß das in der Tabelle verwendete

Symbol "–" für "don't care" steht und somit jede Belegung der entsprechenden Variablen zuläßt. Um nun die semantische Schnittstelle tabellarischer Spezifikationen durch eine Menge von stromverarbeitenden Funktionen ausdrücken zu können, bedarf es genau der Umsetzung der beiden letzten Teile gemäß obiger Erläuterung. Dies führt zu folgender Spezifikation in Abbildung 2.3, wobei das Komma die Konjunktion Boolescher Terme syntaktisch repräsentiert.

$$
\begin{array}{rl}
z = aus & \Rightarrow\ g(s\ \&\ ss, z) = 0\ \&\ g(ss, ein), \\
(s = 0\ \wedge\ z = ein) & \Rightarrow\ g(s\ \&\ ss, z) = 0\ \&\ g(ss, ein), \\
(s = 1\ \wedge\ z = ein) & \Rightarrow\ (\ g(s\ \&\ ss, z) = 0\ \&\ g(ss, ein)\ \vee \\
 & \qquad\quad g(s\ \&\ ss, z) = 1\ \&\ g(ss, aus)\quad)
\end{array}
$$

Abbildung 2.3: Logische Spezifikation der beiden letzten Teile der tabellarischen Spezifikation.

Die in der tabellarischen Darstellung verwendeten Variablen o und z' treten in der logischen Spezifikation nicht auf – dies ist der Grund dafür, warum o und z' in Abbildung 2.2 nicht explizit deklariert sind, deren Typisierung läßt sich allerdings berechnen. Anzumerken bleibt, daß auch bei tabellarischen Spezifikationen nichtdeterministisches Verhalten bei jedem Auftreten entsprechender Eingabekombination oder nur beim ersten Auftreten entsprechender Eingabekombination durch geeignete Wahl der Fortsetzungsfunktion festgelegt werden kann.

Kapitel 3

Spezifikation digitaler Hardware in Focus

Die Spezifikation digitaler Systeme erfordert eine Reihe von zusätzlichen Modellannahmen, die über die Anforderungen in Focus [BDD+92a] hinausgehen. Dieses Kapitel erfaßt diese Modellannahmen und erörtert deren Formalisierung.

3.1 Modularer Entwurf

Um zum einen einfachere Spezifikationen zu erzielen und zum anderen die Wiederverwendbarkeit von Spezifikationen zu erhöhen, zerlegen wir jede Spezifikation auf der Gatterebene in eine Verhaltens- und eine Zeitbeschreibung. Diese Zerlegung geht auf [Dav88] zurück und rechtfertigt sich dadurch, daß auf der Gatterebene die Berechnung der Daten und die Verzögerung der berechneten Daten unabhängig sind. Unter Berücksichtigung bestimmter Eigenschaften von Komponenten ist sogar eine Trennung von Verhaltens- und Zeitbeschreibung auf der RT-Ebene möglich – dies wollen wir bei der Spezifikation der Busstrukturen in Kapitel 5 ausnutzen. Im Gegensatz zu [Dav88] werden in dieser Arbeit die Zeitbeschreibungen ausschließlich für die Ausgänge der Verhaltensbeschreibung erstellt. Diese Einschränkung basiert auf den Konzepten in VHDL ([LSU90]), wo Verzögerungen stets an Ausgabesignale gekoppelt sind. Somit gilt es für jeden Ausgang einer Verhaltensbeschreibung genau eine Zeitbeschreibung mit einem Ein- und Ausgabekanal zu erstellen. Abbildung 3.1 zeigt die Aufteilung in eine Verhaltens- und eine Zeitbeschreibung für eine kombinatorische und eine sequentielle Komponente mit jeweils einem Ausgang. Kombinatorische Komponenten entsprechen Schaltnetzen und sequentielle Komponenten entsprechen Schaltwerken (vergleiche hierzu [Bro93]).

3.2 Initialwerte

In Anlehnung an VHDL gehen auch wir davon aus, daß jede Komponente Initialwerte für ihre Ausgänge besitzt. In unseren Modellierungen befinden sich die Initialwerte für Ausgangsleitungen in den Zeitbeschreibungen und nicht in den Verhaltensbeschreibungen. In der Regel werden wir von einer nichtdeterministischen Auswahl der Initialwerte ausgehen,

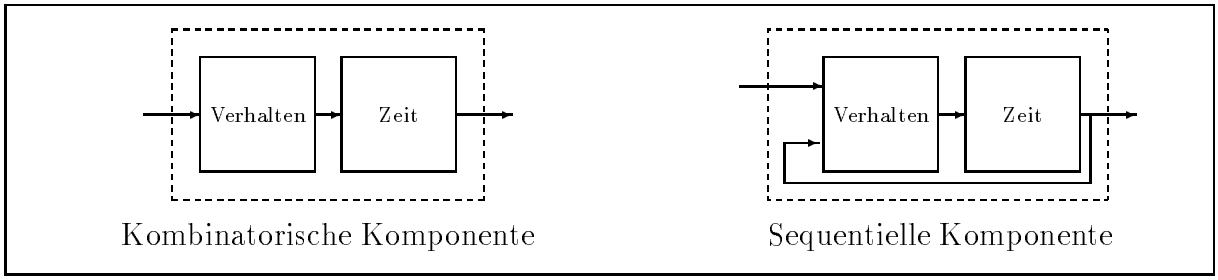


Abbildung 3.1: Modularisierung von Komponenten auf der Gatterebene.

da im allgemeinen nicht a priori feststeht, welcher Spannungspegel sich am Ausgang einer Komponente unmittelbar nach dem Einschalten einstellt. Bei verzögerten Komponenten dient der Initialwert als Ausgangswert bis der erste berechnete Wert zur Verfügung steht. Somit schließt der Initialwert die zeitliche Lücke die exakt der Verzögerungszeit der Komponente entspricht. Bei rückgekoppelten Komponenten (verzögerungsfrei bzw. verzögert) dient der Initialwert als Startwert für die Berechnung der Ausgangswerte der gesamten Rückkopplungsschleife. Bei unverzögerten Komponenten wird der Initialwert nicht benötigt.

3.3 Zeitbegriff und Ströme

Innerhalb digitaler Systeme spielt Zeit eine zentrale Rolle. Dies gilt für kombinatorische und sequentielle Komponenten gleichermaßen. Von besonderem Interesse sind dabei für uns die Gatterlaufzeiten und die Periodendauer der Taktsignale. In unserer Modellierung gehen wir von einer globalen, diskreten Zeit aus, wo in jedem Zeitintervall genau eine Aktion stattfindet ([BD92, Fuc92]). Dabei nehmen wir ferner an, daß die Zeitintervalle alle gleich groß sind und jeweils der für die Modellierung notwendigen, kleinsten Zeitspanne entsprechen. Auf der Ebene der Ströme bedeutet dies, daß jede Aktion genau für oben besagte Zeitspanne ihre Gültigkeit hat. Übertragen auf die Modellierung digitaler Hardware äußert sich also die Präsenz eines stabilen Signalpegels über eine längere Zeitspanne durch das unmittelbar aufeinanderfolgende, wiederholte Auftreten entsprechenden Signalpegels im Strom. Abbildung 3.2 veranschaulicht nun das hier eingeführte Zeitmodell anhand der Modellierung eines Impulses der Länge 5 Nanosekunden. Dabei setzen wir eine kleinste Zeitspanne von 1 ns, einen endlichen Strom s und einen unendlichen Strom t voraus.

Wesentlich ist, daß die Modellierung des Gesamtimpulses durch 5 aufeinanderfolgende Einzelimpulse erfolgt. Die Wahl der kleinsten Zeitspanne hängt von der geforderten Genauigkeit der Modellierung ab. Die Wahl muß dabei so getroffen werden, daß alle relevanten Verzögerungszeiten in der Schaltung durch ein Vielfaches der kleinsten Zeitspanne darstellbar sind.

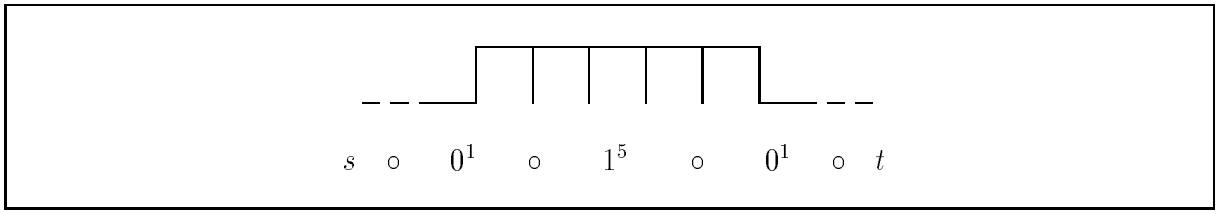


Abbildung 3.2: Zeitmodellierung mit Strömen.

3.4 Zeitbegriff und stromverarbeitende Funktionen

Die Hinzunahme des Zeitbegriffs in Strömen hat auch Auswirkungen auf die im Modell geforderten Eigenschaften für die stromverarbeitenden Funktionen. Wie bereits in [BD92] untersucht, fordert man für entsprechend gezeitete, stromverarbeitenden Funktionen folgende "time progress property":

$$\#f(i_1, \ldots, i_n) \geq min(\#i_1, \ldots, \#i_n)$$

Diese besagt, daß ein Zeitintervall am Ausgang einer gezeiteten, stromverarbeitenden Funktion zumindest durch die Eingabeströme mit Zeitintervallen gleicher Länge festgelegt ist – es können sogar weitere (zukünftige) Ausgabewerte aufgrund von Initialwerten bekannt sein. In dieser Arbeit präzisieren wir obige "time progress property" in Hinblick auf Verhaltensbeschreibungen und Zeitbeschreibungen, ohne sie jedoch verletzen zu wollen.

Für Verhaltensbeschreibungen benötigen entsprechende, stromverarbeitenden Funktionen zur Ermittlung von Ausgabewerten der Zeitspanne i alle Eingaben der Zeitspanne i, auch wenn nur ein Teil davon zur tatsächlichen Berechnung der Ausgaben benötigt wird (Zeitscheibenverarbeitung). Das bedeutet, daß die Zeit an den Ausgängen einer Verhaltensbeschreibungen in gleichem Maße voranschreitet, wie an den Eingängen – die Verarbeitung also synchron, d.h. Zeitscheibe für Zeitscheibe, und nicht asynchron abläuft. Diesen Zusammenhang verdeutlichen wir anhand einer stromverarbeitenden Funktion mit n Eingabeströmen.

$$\#f(i_1, \ldots, i_n) = min(\#i_1, \ldots, \#i_n) \qquad (1^*)$$

Für Zeitbeschreibungen – hier gehen die Initialwerte mit ein – gilt, daß das Zeitintervall am Ausgang größer oder gleich (für Zerodelay-Beschreibungen) ist, als das zur Verfügung stehende Zeitintervall am Eingang. Der Grund dafür liegt allein in der Existenz der Initialwerte. Folgende Formel verdeutlicht diesen Zusammenhang:

$$\#f(i) \geq \#i \qquad (2^*)$$

An dieser Stelle wollen wir darauf hinweisen, daß auch die Zeitbeschreibungen unter Verwendung von internen Puffern so erstellt werden können, daß sie ebenfalls $\#f(i) = \#i$ erfüllen. Dies würde jedoch zu komplizierteren Spezifikationen führen und soll im Verlauf

dieser Arbeit nicht weiter verfolgt werden. Beide obigen Forderungen, für die Verhaltensbeschreibung und die Zeitbeschreibung, schränken die semantischen Modelle für digitale Schaltungen gegenüber der Modelle bei asynchroner Kommunikation ein. Die Funktionen sind nach wie vor stetig, da sie lediglich eine Teilmenge der stetigen, stromverarbeitenden Funktionen zur Beschreibung asynchroner Systeme repräsentieren. Zu beachten ist, daß die beiden Forderungen (1^*) und (2^*) die "time progress property" in [BD92] nicht verletzen, sie allerdings für den jeweiligen Anwendungsfall konkretisieren. Wie sich noch zeigen wird, bedarf es im Falle von verzögerungsfreien Schaltungsrückkopplungen einem Pseudoinitialwert, der die Berechnung des kleinsten Fixpunkts erlaubt.

Abschließend sei bemerkt, daß bei der Modellierung von Schaltungen durch stromverarbeitende Funktionen in unserem Sinne darauf zu achten ist, daß entsprechend der Zeitscheibenverarbeitung alle Eingaben gleichmäßig abgearbeitet werden. Diese Forderung kann bereits durch syntaktische Einschränkungen erzielt werden – die semantischen Modelle bleiben davon unberührt.

3.5 Zeitkonzepte

Bei der Erstellung von Zeitbeschreibungen werden drei unterschiedliche Zeitkonzepte betrachtet. Diese sind das Zerodelay-, das Transportdelay- und das Inertialdelay-Zeitkonzept, die alle in der Hardwarebeschreibungssprache VHDL eingesetzt werden [EEE87]. Die Zeitbeschreibungen basieren auf stromverarbeitenden Funktionen mit jeweils genau einem Ein- und Ausgabekanal. Entsprechend der Zeitscheibenverarbeitung für Zeitbeschreibungen (2^*) in Kapitel 3.4 mit Ausnahme von Zerodelay ist zu beachten, daß für jede Zeitbeschreibung die Anzahl der Elemente im Ausgabestrom größer als die Anzahl der Elemente im Eingabestrom ist. Dies ist aufgrund von Initialwerten, die das Ausgabeverhalten bereits ohne Kenntnis über die zeitsynchronen Eingaben festlegen, immer gegeben. Basierend auf diesen Überlegungen werden nun die oben erwähnten Zeitkonzepte formalisiert.

Eine Zeitkomponente genügt dem *Zerodelay*-Konzept, falls sie den Eingabestrom unverzögert an den Ausgabekanal weitergibt. Zerodelay-Beschreibungen werden meist bei der Modellierung des rein logischen Verhaltens kombinatorischer Schaltungsteile eingesetzt, da hierbei deren Zeitverhalten nicht von Interesse ist. Für einen Eingabestrom $is \in M^\omega$ ergibt sich folgende stromverarbeitende Funktion zur Beschreibung von Zerodelay.

$$\textbf{func}\ zero : M^\omega \to M^\omega$$

$$zero(is) = is$$

Eine Zeitkomponente genügt dem *Transportdelay*-Konzept, falls sie den Eingabestrom um ein festgelegtes Zeitintervall verzögert. Üblicherweise benutzt man Transportdelay zur Beschreibung von Leitungsverzögerungen. Für einen Eingabestrom $is \in M^\omega$, einen

Initialwert $init \in M$ und für eine Verzögerung $del \in \mathbb{N}$ läßt sich eine Transportdelay-Komponente unter Verwendung einer stromverarbeitenden Funktion wie folgt formalisieren.

$$\textbf{func } transport : M^\omega \times M \times \mathbb{N} \to M^\omega$$

$$transport(is, init, del) = init^{del} \circ is$$

Der resultierende Strom besteht aus der Konkatenation eines Initialstroms der Länge del ($init^{del}$) und des Eingabestroms (is). Basierend auf der kleinsten Zeitspanne (vgl. Kapitel 3.3) entspricht die Länge des Initialstroms der Verzögerungszeit. Bezüglich der Zeitscheibenverarbeitung gilt für Transportdelay, daß die Zeit am Ausgang der Zeit am Eingang stets um die Verzögerungszeit voraus ist. In anderen Worten ist die Länge des Ausgabestroms stets um del größer als die Länge des Eingabestroms – formalisiert gilt für Transportdelay $\#f.x = \#x + del$.

Eine Zeitkomponente genügt dem *Inertialdelay*-Konzept, wenn sie ausschließlich auf Signale reagiert, deren Impulsdauer mindestens der Länge der eigenen Verzögerungszeit entsprechen ([NCI75, EEE87]). Üblicherweise benutzt man Inertialdelay zur Beschreibung von Verzögerungen in Komponenten ([LSU90]). Die Berechtigung dieses Zeitmodells ergibt sich bei genauerer Betrachtung des Schaltverhaltens von Komponenten, wie es in Abbildung 3.3 dargestellt ist.

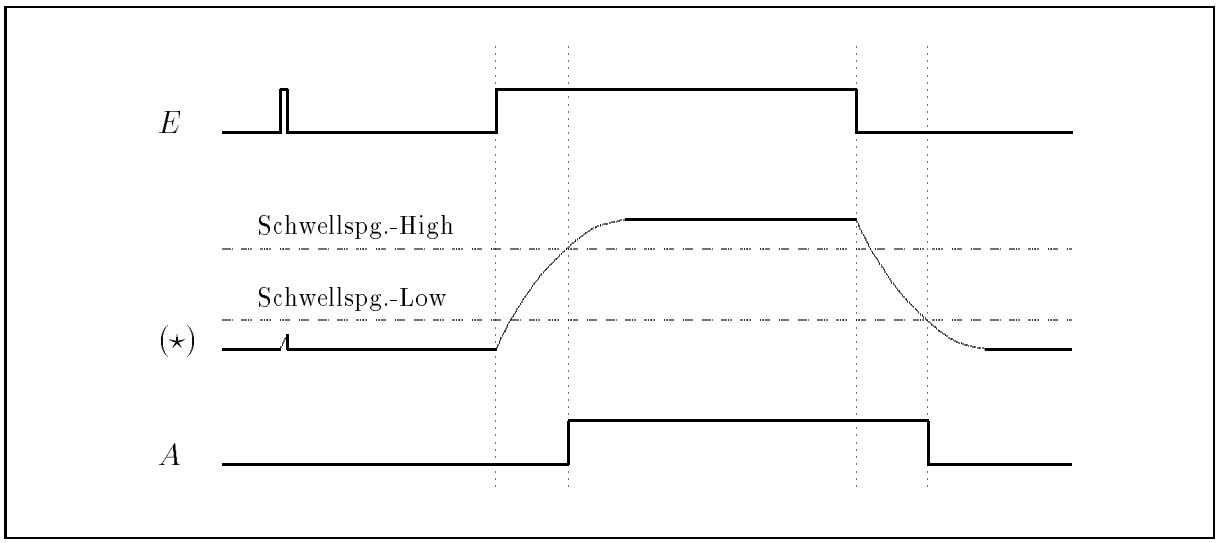


Abbildung 3.3: Schaltverhalten von Komponenten.

Dabei entspricht E dem digitalen Signalverlauf am Eingang der Inertialdelay-Komponente und $(\star)$ dem entsprechenden analogen Signalverlauf mit Angabe der Schwellschaltspannung für High- und Low-Signalpegel. Der zugehörige Signalverlauf am Ausgang der Inertialdelay-Komponente ist in A abgebildet. Man beachte, daß der erste Impuls auf-

grund seiner zu geringen Länge unterdrückt wird – der Spannungspegel erreicht nicht den notwendigen Schwellwert (Schwellspannung-High).

Das Inertialdelay-Konzept läßt sich nun für einen Eingabestrom $is \in M^\omega$, einen Initialwert $init \in M$ und eine Verzögerung $del \in \mathbb{N}$ in Anlehnung an [SB90] wie folgt formalisieren.

func $inertial : M^\omega \times M \times \mathbb{N} \to M^\omega$

$inertial(is, init, del) = init^{del} \circ h(is, init, del)$
where $h(is, old\_value, del) =$
  if $\#is \geq del$ (1)
  then if $is|_{del} = ft.is^{del}$ (2)
    then $ft.is$ & $h(rt.is, ft.is, del)$ (3)
    else old_value & $h(rt.is, old\_value, del)$ (4)
  else ϵ fi (5)

Die stromverarbeitende Funktion *inertial* erzeugt einen Initialstrom der Länge *del* (basierend auf der kleinsten Zeitspanne) gefolgt von einem modifizierten Eingabestrom. Diese Modifikation betrifft das Herausfiltern von Impulsen einer Länge kleiner als *del* und findet in der Hilfsfunktion h statt. (1) überprüft, ob der Eingabstrom zumindest die Länge *del* hat. Ist dies der Fall, so wird in (2) überprüft, ob die ersten *del* Stromelemente gleich sind. Falls ja, so wird dieser Impuls ausgegeben (vgl. (3)) – ansonsten wird lediglich der bereits am Ausgang anliegende Wert beibehalten (vgl. (4)). Ist die Länge des Eingabestroms *is* kleiner als *del*, so lassen sich im allgemeinen keine Aussagen über das Verhalten von Inertialdelay-Komponenten machen, dies äußert sich hier durch die Ausgabe von ϵ (vgl. (5)). Bezüglich der "time progress property" gilt für Inertialdelay, daß die Zeit am Ausgang der am Eingang zumindest eine aber maximal *del* Zeiteinheiten voraus ist – die tatsächliche Zahl hängt von der Länge des Eingabestroms *is* ab. Folglich ergibt sich für Inertialdelay $\#f.x \geq \#x + 1$.

An dieser Stelle wollen wir darauf hinweisen, daß das Inertialdelay-Konzept für eine Verzögerung gleich Null dem Zerodelay-Konzept entspricht – gleiches gilt auch für Transportdelay. Die vorgestellten Konzepte entsprechen der VHDL-Standardisierung von 1987 ([EEE87]). In der neuesten Restandardisierung ([EEE92]) wurde das Inertialdelay-Konzept dahingehend geändert, daß nun die Länge der zu unterdrückenden Impulse unabhängig von der Verzögerungszeit zusätzlich einzustellen ist. Diese hat allerdings auf diese Arbeit keinen Einfluß, da Technologieabhängigkeit von Spezifikationen digitaler Hardware bereits mit dem eingeschränkten Inertialdelay-Konzept untersucht werden können.

3.6 Rückkopplungsoperator

Die sequentielle und parallele Komposition läuft bei funktionalen Beschreibungen digitaler Schaltungen problemlos ab. Die Rückkopplung basierend auf dem klassischen Rück-

kopplungsoperator (μ-Operator) gestaltet sich im Falle verzögerungsfreier Rückkopplungen als nicht praktikabel. Unter Berücksichtigung der Forderung (1*) in Kapitel 3.4, der Zeitscheibenverarbeitung für Verhaltensbeschreibung, liefert nämlich der μ-Operator bei verzögerungsfreien Rückkopplungen stets den leeren Strom als trivialen Fixpunkt. Dies liegt daran, daß die Anzahl der Stromelemente auf der rückgekoppelten Leitung zu Beginn Null ist, und dann gemäß der Forderung für Verhaltensbeschreibungen (1*) in Kapitel 3.4 nicht anwachsen kann. Diese Forderung in Kapitel 3.4 dient der Einschränkung asynchron kommunizierender, stromverarbeitender Funktionen auf synchron kommunizierende, stromverarbeitende Funktionen und besagt, daß sich die Anzahl der Stromelemente am Ausgang aus dem Minimum der Anzahl der Stromelemente an den Eingängen ergibt. Da also bei Zerodelay-Modellierungen das Minimum der Anzahl der Stromelemente an den Eingängen durch die rückgekoppelte Leitung fest auf Null gelegt wird, bleibt die Anzahl der Stromelemente am Ausgang ebenso auf Null.

Der sich daraus ergebende, triviale Fixpunkt bei der Verwendung des klassischen μ-Operators kann durch die Einführung eines Initialwerts (vgl. Kapitel 3.2) umgangen werden. Dadurch wird erreicht, daß nun ein Ausgangsstrom ungleich dem trivialen Fixpunkt mit dem Initialwert als Startwert berechnet werden kann. Für die Realisierung stehen zwei unterschiedliche Konzepte zur Diskussion. Zum einen ist es denkbar, eine zusätzliche Komponente im Rückkopplungspfad der Modellierung mit aufzunehmen, die zu Beginn einen Initialwert ausgibt und sich anschließend wie die Identität verhält. Allerdings spricht gegen diese Lösung, daß für die Modellierung eine zusätzliche Komponente (Initialwertgeber) benötigt wird, die in der tatsächlichen Hardwarerealisierung nicht aufscheint. Eine weitere interessante Beobachtung mit Blick auf Zerodelay-Modellierungen und Initialwerte ist diejenige, daß die Initialwerte notwendigerweise nur zur Berechnung der Rückkopplungsschleife und nicht zur Festlegung der Ausgangswerte gebraucht werden – die Initialwerte sind also in diesem Fall eng mit der Rückkopplung verbunden.

Dies führt uns zu einer zweiten Lösung, der Erweiterung des klassischen μ-Operators. Hierbei wird ein Parameter *init* für einen Initialstrom (in dieser Arbeit stets mit einem einelementigen Initialstrom belegt) zum μ-Operator hinzugenommen und somit bedarf es keiner separaten Komponenten wie in der oben diskutierten Alternative. Bezüglich der Zeitbeschreibungen deckt der neu entstandene $\underline{\mu}$-Operator Zerodelay- und Nonzerodelay-Modellierungen ab. Abbildung 3.4 veranschaulicht den erweiterten μ-Operator.

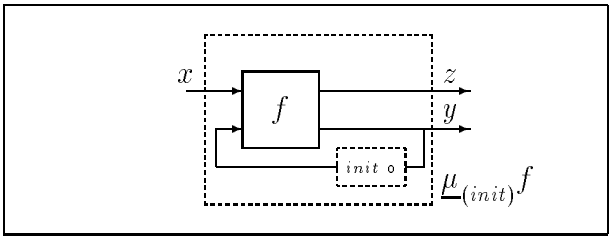


Abbildung 3.4: Schema zum erweiterten μ-Operator.

Sei f eine $(n+1,m)$-stellige, stromverarbeitende Funktion mit $m \geq 1$, wobei $n+1$ der Eingabestelligkeit und m der Ausgabestelligkeit von f entspricht. $\underline{\mu}_{(init)}f$ ist eine von f abgeleitete, (n,m)-stellige, stromverarbeitende Funktion mit einer Rückkopplungsschleife, wobei $init$ mit einem Initialstrom für die Rückkopplungsschleife zu instanziieren ist.

Wir definieren für ein n-Tupel (von Strömen) $x \in A^{\omega}_{1:n}$ und einen Initialstrom $init$
$(\underline{\mu}_{(init)}f).x = (z \oplus y)$
falls $(z \oplus y)$ die kleinste Lösung (der kleinste Fixpunkt) der Gleichung
$(z \oplus y) = f(x \oplus (init \circ y))$
ist, wobei z ein $(m-1)$-Tupel (von Strömen) und y ein Strom ist.

Formal bezeichnet $(\underline{\mu}_{(init)}f).x$ den kleinsten Fixpunkt relativ zu x und $init$. Der hier definierte $\underline{\mu}$-Operator ist so festgelegt, daß die zurückgekoppelte Leitung auf den "letzten" Eingang der Komponente f geführt wird. Im weiteren Verlauf dieser Arbeit wollen wir eine Überladung des $\underline{\mu}$-Operators dahingehend erlauben, daß die rückgekoppelte Leitung auf jeden beliebigen Eingang gelegt werden kann.

Für eine verzögerungsfreie Komponente f (Zerodelay-Loop) gestattet nun der Initialstrom $init$ des $\underline{\mu}$-Operators die Berechnung eines nichttrivialen Fixpunkts ($\neq \epsilon$). Entsprechend der intuitiven Vorstellung erscheint der Initialstrom $init$ nicht auf den Ausgabeleitungen – dies entspricht dem Verhalten einer verzögerungsfreien Komponente $\underline{\mu}_{(init)}f$.

Für verzögerte Komponenten f befinden sich gemäß Kapitel 3.2 Initialwerte an deren Ausgängen (also an den Ausgängen der zugehörigen Zeitbeschreibungen). Diese Initialwerte erlauben die Berechnung eines nichttrivialen Fixpunkts bereits mit dem klassischen μ-Operator. Die Initialwerte erscheinen hier entsprechend der intuitiven Vorstellung an den Ausgängen.

Unter Einbeziehung methodischer Aspekte ist jedoch die Verwendung eines einheitlichen Konzepts für alle auftretenden Fälle wünschenswert. Dies läßt sich durch eine geschickte Belegung des Initialstroms bei der Verwendung des $\underline{\mu}$-Operators für beide Fälle erreichen. Wie bereits erwähnt, wird für verzögerungsfreie Komponenten ein einelementiger Initialstrom an den $\underline{\mu}$-Operator übergeben. Für verzögerte Komponenten ist dies nicht notwendig, da bereits Initialwerte an den Ausgängen der zugehörigen Zeitbeschreibungen eine Berechnung eines nichttrivialen Fixpunkts erlauben. In diesem Fall wird der Initialstrom im $\underline{\mu}$-Operator anstelle eines einelementigen Stroms mit dem leeren Strom instanziiert. Insbesondere gilt $\underline{\mu}_{(\epsilon)}f \equiv \mu f$.

3.7 Zeitabstraktion

Dieser Abschnitt befaßt sich mit der Kopplung unterschiedlicher Zeitebenen innerhalb der Beschreibung digitaler Schaltungen. Dabei bezeichnen wir die Zeitebene mit der feineren Zeitmodellierung als Mikroebene und die mit der gröberen Zeitmodellierung als Makroebene. Am Beispiel eines getakteten Flipflops wäre die Zeitmodellierung auf der Makroebene durch die Taktfrequenz und die Zeitmodellierung auf der Mikroebene durch die Gatterlaufzeiten des Flipflops gegeben. Einen weiteren Anwendungsfall stellt eine rein kombinatorische Schaltung dar, wobei die Zeitskalierung auf der Makroebene so einzustel-

len ist, daß lediglich die stabilen Ausgangswerte erfaßt werden, aber die Zeitmodellierung auf der Mikroebene alle instabilen Zwischenschritte mit berücksichtigt.

Basierend auf diesen Überlegungen benötigen wir also eine Zeitabstraktion, die das Verhalten auf der Makroebene unter Einbeziehung des Verhaltens auf der Mikroebene beschreibt. Diese Idee der Zeitabstraktion wurde bereits in [Del87, Del91] aufgegriffen.

> "In general we are not interested in the intermediate values, but only in the values at the new stable state in response to new boundary conditions. Therefore we need here a time abstraction mechanism to filter out the stable state."
>
> Page 112 [Del87]

Wie bereits in Kapitel 2 erläutert, basieren Komponentenbeschreibungen in Focus auf funktionalen Spezifikationen. Die Semantik solcher Spezifikationen sind Mengen von stromverarbeitenden Funktionen. Dementsprechend sollen auch Zeitabstraktionen von Komponentenbeschreibungen wieder Spezifikationen im obigen Sinne mit Mengen von stromverarbeitenden Funktionen als Semantik sein – allerdings auf einer abstrakteren Zeitebene, der Makroebene. Technisch setzen sich Zeitabstraktionen aus Modellierungen entsprechender Komponenten auf der Mikroebene sowie aus Repräsentations- und Abstraktionsfunktionen zusammen. Die Repräsentationsfunktion bewerkstelligt die Umsetzung der Eingabedaten der Makroebene in die Eingabedaten der Mikroebene. Die Abstraktionsfunktion übernimmt die Umsetzung der Ausgabedaten der Mikroebene in die Ausgabedaten der Makroebene. In Hinblick auf die in dieser Arbeit betrachteten Komponenten, dem RS-Flipflop und den Busstrukturen, entsprechen die Repräsentationsfunktionen dem Expandieren jedes einzelnen Stromelements auf der Makroebene zu einem lokalen Eingabestrom der Mikroebene. Die Abstraktionsfunktionen analysieren die unter Verwendung der Modellierung auf der Mikroebene berechneten Ausgabeströme und geben die aus dieser Analyse resultierenden Werte an die Makroebene zurück. Abbildung 3.5 schematisiert nochmals die Wirkungsweise der Zeitabstraktion, wobei f einer Repräsentationsfunktion und g einer Abstraktionsfunktion entspricht.

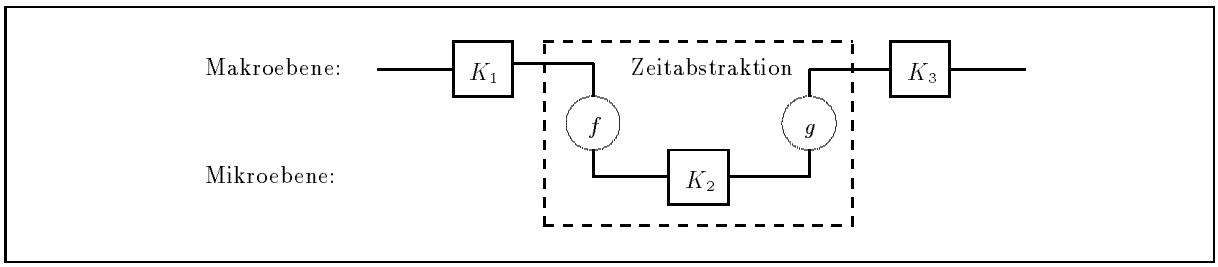


Abbildung 3.5: Schematisierung der Zeitabstraktion.

Zu beachten ist, daß die Zeitabstraktion dem jeweiligen Anwendungsfall anzupassen ist. Dies trifft insbesondere auf die Analysefunktion (Repräsentationsfunktion) zu. Zusätzlich wollen wir hier erwähnen, daß die Zeitabstraktion einen Spezialfall von Spezifikationsverfeinerung darstellt. Hierbei gilt es jedoch nicht, den Systementwurf durch Verfeinerung voranzutreiben, sondern Beschreibungen auf unterschiedlichen Zeitebenen miteinander in Beziehung zu setzen.

Bei der Beschreibung digitaler Schaltungen kann es auch von Interesse sein, mehr als zwei Zeitebenen übereinander zu legen (Systemtakt - Prozessortakt - Gatterlaufzeiten). Die hier beschriebene Zeitabstraktion läßt sich auf einfache Weise auch für diesen Anwendungsfall einsetzen. Abbildung 3.6 verdeutlicht dies anhand einer Skizze, wobei entscheidend ist, daß eine in eine Zeitabstraktion eingebettete Komponentenbeschreibung selbst wieder als Komponentenbeschreibung betrachtet werden kann.

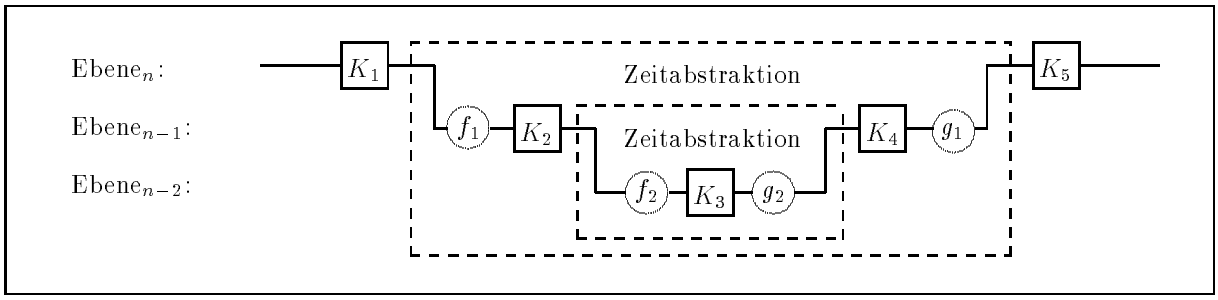


Abbildung 3.6: Mehrfachschachtelung der Zeitabstraktion.

Man beachte, daß die hier vorgestellte Zeitabstraktion die Voraussetzungen dafür schafft, das Verhalten von Systemen, deren interne Zeitbasis kleiner als die von außen angelegte Zeitbasis ist, zu berechnen. Im weiteren Verlauf dieser Arbeit wird dieses Konzept sowohl für das RS-Flipflop als auch für die Busstruktur angewandt.

Kapitel 4

RS-Flipflops

Dieses Kapitel behandelt die funktionale Modellierung des RS-Flipflops auf der RT-Ebene sowie auf der Gatterebene unter Berücksichtigung der Modellierungsaspekte von VHDL. Insbesondere betrifft dies das Einbeziehen

- aller Eingangskombinationen,
- einer adäquaten Zeitmodellierung und
- technologieabhängiger Aspekte auf der Gatterebene.

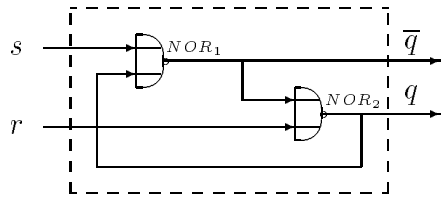
Das Hauptaugenmerk unserer Untersuchungen gilt der Fragestellung nach dem technologieabhängigen Verhalten von RS-Flipflop-Spezifikationen auf der RT-Ebene. Um die Verzögerungszeiten innerhalb RS-Flipflops adäquat erfassen zu können, werden wir uns jedoch auf die Beschreibungen der Gatterebene konzentrieren. Es wird sich zeigen, daß die technologiebedingten Verhaltensunterschiede entsprechender Varianten auf der Gatterebene durch die Verwendung aller Eingabekombinationen sowie einer adäquaten Zeitmodellierung sichtbar werden und letztendlich die Technologieunabhängigkeit der Beschreibungen auf der RT-Ebene widerlegen. Die Berücksichtigung aller Eingabekombinationen und die Verwendung von Inertialdelay, eine adäquate Zeitmodellierung für Komponenten, geht auf VHDL zurück. Die in diesem Kapitel erstellten Modellierungen ergänzen die in der Literatur bekannten Spezifikationen [Car82, Del87, Tuc91] in Hinblick auf die Behandlung aller Eingabekombinationen und einer adäquaten Zeitmodellierung. Basierend auf diesen Erweiterungen wird hier erstmals der Aspekt der Technologieabhängigkeit formal betrachtet.

Dieses Kapitel gliedert sich in technologieunabhängige und technologieabhängige Modellierungen von RS-Flipflops sowie in die Untersuchung von Verfeinerungsbeziehungen der Modellierungen untereinander. Die technologieabhängigen Untersuchungen beschränken sich auf NMOS- und CMOS-Realisierungstechniken. Beginnen wollen wir zunächst mit Betrachtungen über den Aufbau und die Wirkungsweise eines RS-Flipflops.

4.1 Aufbau und Wirkungsweise des RS-Flipflops

Sequentielle Komponenten, gekoppelt an ein Taktsignal, sind wesentliche Bestandteile in synchronen Schaltungen. Sie dienen zum einen der Kompensation unterschiedlicher Lauf-

zeiten in kombinatorischen Schaltungsteilen und zum anderen ermöglichen sie den Aufbau von synchronen Schieberegistern und Zählern. Als Basisbaustein für sequentielle Komponenten gilt das RS-Flipflop ([Per86]). Es ist nicht taktgesteuert und stellt den Grundbaustein aller sequentiellen Komponenten dar. Abbildung 4.1 zeigt ein auf NOR-Gattern basierendes RS-Flipflop und die dazugehörige Zustandsfolgetabelle (alle im folgenden getroffenen Aussagen lassen sich analog auf ein aus NAND-Gattern aufgebautes RS-Flipflop übertragen).



Operation	s_t	r_t	q_t	$\overline{q_t}$	q_{t+1}	$\overline{q_{t+1}}$
"Setzen"	1	0	1	0	1	0
"Setzen"	1	0	0	1	1	0
"Rücksetzen"	0	1	0	1	0	1
"Rücksetzen"	0	1	1	0	0	1
"Speichern"	0	0	0	1	0	1
"Speichern"	0	0	1	0	1	0
"irregulär"	1	1	1	0	0	0
"irregulär"	1	1	0	1	0	0

Abbildung 4.1: RS-Flipflop mit Zustandsfolgetabelle.

Neben den üblichen Operationen "Setzen", "Rücksetzen" und "Speichern" existiert eine sogenannte "irreguläre" Operation ([Per86]). Die Bezeichnung "irregulär" läßt sich darauf zurückführen, daß die beiden Ausgänge am RS-Flipflop nicht mehr komplementär zueinander sind. Die irreguläre Operation darf aber durchaus mit Bedacht in einer Schaltung angewandt werden, da sie stets ein deterministisches Ergebnis liefert. Die Kombination aus der irregulären Operation und einer anschließenden Operation Speichern (im weiteren Verlauf als irreguläre Eingabesequenz bezeichnet) läßt hingegen von vornherein keine Rückschlüsse auf das Ausgangsverhalten zu. Ein Verbot dieser irregulären Eingabesequenz würde wohl die Modellierung erleichtern, entspräche aber nicht den Anforderungen aus der Praxis, wo das Zusammenspiel zwischen Hardware und Software (Firmware) bereits auf der Gatterebene das Systemverhalten beeinflußt. So kann fehlerhafte Software dazu führen, daß alle denkbaren Eingangskombinationen an Hardwarekomponenten anliegen, also auch die Kombination aus der irregulären Operation und der Operation Speichern. Die Behandlung und Modellierung dieser irregulären Eingabesequenz schafft die Voraussetzungen für technologieabhängige Untersuchungen und begründet sich nicht zuletzt durch entsprechende VHDL-Modellierung

Ein weiterer wichtiger Aspekt bei der Modellierung von RS-Flipflops ist das zugrundegelegte Zeitmodell. In [Car82, Del87, Tuc91] wird als Zeitmodell für das RS-Flipflop ausschließlich Leitungsverzögerung benutzt – dies entspricht dem in Kapitel 3.5 eingeführten Transportdelay ([LSU90]). Wie bereits erwähnt ist allerdings die Inertialdelay-Modellierung im Zusammenhang mit Komponentenmodellierung das geeignete Beschreibungsmittel ([SB89, LSU90, Inc91]). Der Unterschied bei der Modellierung eines RS-Flipflops mit Transportdelay und Inertialdelay zeigt sich genau dann, wenn die oben beschriebene irreguläre Eingabesequenz auftritt, und die Verzögerungen der beiden NOR-Gatter unter-

schiedlich sind. Dabei verursacht die Transportdelay-Modellierung Schwingungen und die Inertialdelay-Modellierung einen stabilen Zustand. Ein Phänomen das, wie sich zeigen wird, abhängig von der Realisierungstechnologie unterschiedliche Auswirkungen hat.

4.2 Technologieunabhängige RS-Flipflops

In diesem Abschnitt beschreiben wir das RS-Flipflop unabhängig von der Realisierungstechnologie auf der RT-Ebene für unbekannte Verzögerungszeiten und auf der Gatterebene für bekannte Verzögerungszeiten. Der Wechsel der Abstraktionsebene von der RT-Ebene hin zur Gatterebene wird durch die Hinzunahme von Verzögerungszeiten notwendig, wobei die Spezifikation auf der Gatterebene technologieunabhängig im Sinne von VHDL ist. Abstrahiert man von den Verzögerungszeiten, so ist die technologieunabhängige Variante auf der Gatterebene verhaltensäquivalent zur RT-Beschreibung des RS-Flipflops – dies ist wesentlich, da auf diese Weise die Untersuchungen der Verhaltensverfeinerung zu den technologieabhängigen Beschreibungen auf der Gatterebene angemessen durchgeführt werden können. Inwieweit die NMOS- und CMOS-Modellierungen Verhaltensverfeinerungen darstellen, wird in Kapitel 4.4 geklärt. Zunächst erarbeiten wir die für die Modellierung notwendigen Treiberstärken und beschreiben dann das RS-Flipflop für unbekannte Verzögerungszeiten. Anschließend beschäftigen wir uns mit der Modellierung des RS-Flipflops für bekannte Verzögerungszeiten. Dabei wird zunächst der kombinatorische Anteil beschrieben, dieser dann mit dem in Kapitel 3.6 eingeführten, erweiterten Rückkopplungsoperator vervollständigt und abschließend die so entstandene Modellierung des RS-Flipflops in eine Zeitabstraktion eingebunden. Untersuchungen von RS-Flipflops mit bekannten Verzögerungsintervallen bleiben in dieser Arbeit unberücksichtigt.

4.2.1 Treiberstärken

Bei der technologieunabhängigen Modellierung von Komponenten im allgemeinen und RS-Flipflops im speziellen betrachten wir zunächst nur High- und Low-Signalpegel. Dabei wird der High-Signalpegel mit "h" und der Low-Signalpegel mit "l" modelliert. Diese Festlegung begründet sich darin, daß keinerlei Aussagen über die Realisierungstechnologie vorliegen und somit keine detailiertere Unterscheidung der Signalpegel möglich ist. Allerdings wollen wir in dieser Arbeit das Komponieren unterschiedlicher Technologien unterstützen um somit die Vorteile jeder einzelnen Technologie ausnützen zu können – dies erlaubt insbesondere die Beschreibung von Schaltkreisen, deren Einzelkomponenten auf unterschiedlichen Realisierungstechnologien basieren ([Rug91]). Das hat zur Folge, daß alle Komponenten auch technologiespezifische Eingangssignale verarbeiten können – man beachte, daß diese Maßnahmen kein Einbeziehen von technologiespezifischen Treiberstärken in die Modellierungen bedeutet, sondern vielmehr die Modellierungen gegenüber technologiespezifischer Eingangswerte unempfindlich macht. Daraus ergeben sich für die technologieunabhängigen Spezifikationen von RS-Flipflops eine Aktionsmenge (Act) aller in dieser Arbeit betrachteten Signalpegel und eine Aktionsmenge ($Act_\square$) mit ausschließlich technologieunabhängigen Signalpegeln:

- $Act = \{l, h, 0, H, 1\}$
- $Act_{\square} = \{l, h\}$

Dabei entspricht "l" dem technologieunabhängigen Low-Pegel, "h" dem technologieunabhängigen High-Pegel, "0" dem starken Low-Pegel für NMOS und CMOS, "H" dem schwachen High-Pegel für NMOS und "1" dem starken High-Pegel für CMOS. Um die nachfolgenden Spezifikationen lesbarer zu gestalten, definieren wir zwei Prädikate LO und HI zur Bestimmung von Low- und High-Signalpegeln gemäß Abbildung 4.2.

$$LO : Act \rightarrow Bool$$
$$LO.x \equiv x = 0 \vee x = l$$

$$HI : Act \rightarrow Bool$$
$$HI.x \equiv x = 1 \vee x = H \vee x = h$$

Abbildung 4.2: Klassifizierung von Treiberstärken.

Man beachte, daß die Aktionen in der Aktionsmenge Act lediglich die Eingangstransistoren der Schaltkreiskomponenten steuern und sich nicht ins Innere der Komponente fortpflanzen. So ist es beispielsweise unerheblich, ob ein Gatter von einem "starken" oder einem "schwachen" Signalpegel angesteuert wird, da der Eingangswiderstand bei den in dieser Arbeit betrachteten Schaltkreisfamilien ohnehin sehr hoch ist und somit unabhängig von den Treiberstärken sich der Ausgang entsprechenden dem logischen Verhalten einstellt.

4.2.2 Unbekannte Verzögerungszeiten

Für unbekannte Verzögerungszeiten im RS-Flipflop lassen sich Spezifikationen auf der RT-Ebene angeben. Für diese Spezifikationen bietet sich eine Unterscheidung in

- unbekannte, aber unterschiedliche Verzögerungszeiten und
- unbekannte, aber gleiche Verzögerungszeiten

an. Eine Modellierung auf einer Beschreibungsebene mit exakter Zeitbetrachtung erscheint in beiden Fällen aufgrund fehlender, quantitativer Zeitangaben müßig. Dennoch bestimmt das Inertialdelay-Konzept auch hier das Verhalten der Modelle, obwohl von einer exakten Zeitbetrachtung abstrahiert wird – dies gilt insbesondere für die bereits angesprochene Behandlung der irregulären Eingabesequenz. Für die weiteren Betrachtungen wird im folgenden angenommen, daß die unbekannten Verzögerungszeiten klein gegenüber den Signalimpulswechseln an den Eingängen sind. Diese Annahme stützt sich auf Erfahrungswerte aus der Praxis, wo Taktfrequenzen stets wesentlicher größer als Gatterlaufzeiten sind ([SB89]). Der Unterschied bei der Modellierung der beiden Fälle für unbekannte Verzögerungszeiten tritt genau dann auf, wenn die irreguläre Eingabesequenz das Verhalten des RS-Flipflops bestimmt.

Für die Spezifikationen von RS-Flipflops mit unbekannten Verzögerungszeiten führen wir eine Zustandsmenge "$State$" ein.

- $State = \{set, res, irr\}$

Die in der Menge $State$ aufgelisteten Zustände stehen für "das RS-Flipflop wurde zuletzt gesetzt (set)", "das RS-Flipflop wurde zuletzt zurückgesetzt (res)" und "das RS-Flipflop war zuletzt im irregulären Zustand (irr)".

4.2.2.1 Unterschiedliche Verzögerungszeiten

Bei unbekannten aber unterschiedlichen Verzögerungszeiten erzwingt die irreguläre Eingabesequenz entweder ein Setzen oder ein Rücksetzen des Flipflops. Dies bedeutet, daß abhängig davon, welches der beiden Gatter die kürzere Laufzeit besitzt, sich der Ausgang entweder auf $(\overline{q} = 0, q = 1)$ oder auf $(\overline{q} = 1, q = 0)$ einstellt. Ist beispielsweise das Gatter NOR_1 (siehe Abbildung 4.1) schneller, so würde dies $(\overline{q} = 1, q = 0)$ am Ausgang zur Folge haben. Modelliert wird dieses Verhalten mittels Nichtdeterminismus, wobei sich der hier vorliegende Nichtdeterminismus allerdings auf das erste Auftreten besagter Eingangskombination beschränkt, da man zwar von unterschiedlichen, aber fest fixierten Verzögerungszeiten ausgeht. Praktisch bedeutet dies, daß sich ein bestimmtes RS-Flipflop mit unbekannten aber unterschiedlichen Verzögerungszeiten beim Anlegen der irregulären Eingabesequenz stets gleich verhält. Abbildung 4.3 zeigt die entsprechende funktionale Spezifikation für das RS-Flipflop auf der RT-Ebene in Form einer Tabelle. Das RS-Flipflop ist modelliert durch eine Menge von stromverarbeitenden Funktionen,

pred $FF^{uu} : ((Act^{\omega})^2 \rightarrow (Act^{\omega}_{\Box})^2) \rightarrow Bool$

spec $FF^{uu}.f \equiv \forall_{ss,rs \in Act^{\omega}}: f(ss, rs) = g(ss, rs, set) \ \vee \ f(ss, rs) = g(ss, rs, res)$
where $\forall_{s,r \in Act; ss,rs \in Act^{\omega}; z \in State}$:

$g(s \ \& \ ss, r \ \& \ rs, z) \ = \ (\overline{q}, q) \frown g(ss, rs, z')$				
Setzen s	Rücksetzen r	Zustand z	Ausgabetupel $(\overline{q}, q)$	Folgezustand z'
$HI.s$	$LO.r$	–	(l, h)	set
$LO.s$	$HI.r$	–	(h, l)	res
$LO.s$	$LO.r$	set	(l, h)	set
$LO.s$	$LO.r$	res	(h, l)	res
$HI.s$	$HI.r$	–	(l, l)	irr
$LO.s$	$LO.r$	irr	(l, h)	set
			(h, l)	res

Abbildung 4.3: Spezifikation des RS-Flipflops für unbekannte, aber unterschiedliche Verzögerungszeiten auf der RT-Ebene.

die durch die Spezifikation FF^{uu} festgelegt sind, wobei uu für unbekannte, aber unterschiedliche Verzögerungszeiten steht. Eine Funktion erfüllt genau dann die Spezifikation FF^{uu}, wenn es neben den üblichen Anforderungen an ein RS-Flipflop zusätzlich auf die irreguläre Eingabesequenz entweder immer mit Setzen oder immer mit Rücksetzen reagiert. Der initiale Zustand des RS-Flipflops wird nichtdeterministisch auf "gesetzt" oder "zurückgesetzt" festgelegt. Der entscheidende Vorteil der Spezifikation in Abbildung 4.3 liegt in der impliziten Verwendung des Inertialdelay-Konzepts. Die hier vorgestellte Modellierung verhält sich beim Anlegen der irregulären Eingabesequenz gemäß dem in der Praxis zu beobachtbaren Verhalten – das RS-Flipflop wird entweder gesetzt oder rückgesetzt.

4.2.2.2 Gleiche Verzögerungszeiten

Bei unbekannten aber gleichen Verzögerungszeiten tritt beim Anlegen der irregulären Eingabesequenz Schwingungsverhalten am Ausgang des RS-Flipflops auf. Da die Verzögerungszeiten unbekannt sind, ist eine Aussage über die Frequenz der Schwingung unmöglich. Insbesondere ist es auch nicht möglich, die Werte an den beiden Ausgängen des RS-Flipflops zu einem vorgegebenen Zeitpunkt zu bestimmen, da sowohl $(\overline{q} = 0, q = 0)$ als auch $(\overline{q} = 1, q = 1)$ am Ausgang anliegen können. Eine adäquate Modellierung auf der RT-Ebene erhalten wir erneut durch Nichtdeterminismus. Allerdings ist hier im Vergleich zum vorherigen Abschnitt bei jedem Auftreten der irregulären Eingabekombination eine erneute, nichtdeterministische Auswahl der möglichen Ausgabe eine adäquate Modellierung. Abbildung 4.4 zeigt die entsprechende funktionale Spezifikation für das RS-Flipflop für unbekannte, aber gleiche Verzögerungszeiten in tabellarischer Form.

pred $FF^{ug} : ((Act^{\omega})^2 \times State \to (Act^{\omega}_{\square})^2) \to Bool$

spec $FF^{ug}.f \equiv \exists g{:}FF^{ug}.g \wedge \forall s,r{\in}Act; ss,rs{\in}Act^{\omega}; z{\in}State{:}$

$f(s \;\&\; ss, r \;\&\; rs, z) \;=\; (\overline{q}, q) \frown g(ss, rs, z')$

Setzen s	Rücksetzen r	Zustand z	Ausgabetupel $(\overline{q}, q)$	Folgezustand z'
$HI.s$	$LO.r$	–	(l,h)	set
$LO.s$	$HI.r$	–	(h,l)	res
$LO.s$	$LO.r$	set	(l,h)	set
$LO.s$	$LO.r$	res	(h,l)	res
$HI.s$	$HI.r$	–	(l,l)	irr
$LO.s$	$LO.r$	irr	(l,l)	irr
			(h,h)	irr

Abbildung 4.4: Spezifikation des RS-Flipflops für unbekannte, aber gleiche Verzögerungszeiten auf der RT-Ebene.

Bei dieser Spezifikation wollen wir den Zustand des Flipflops nicht nach außen verbergen – insbesondere entfällt dadurch auch die nichtdeterministische Auswahl des Initialwerts. Die Schnittstelle stimmt mit der Schnittstelle der realen Komponente nicht überein. Ein zusätzliches Einbetten in eine schnittstellenkonforme Spezifikation ist natürlich prinzipiell möglich, soll hier aber nicht weiter verfolgt werden. Die Spezifikation FF^{ug} basiert erneut auf einer Menge von stromverarbeitenden Funktionen, wobei ug für unbekannte, aber gleiche Verzögerungszeiten steht. Das Schwingungsverhalten kommt in der letzten Zeile der tabellarischen Spezifikation zum Ausdruck. Dabei ermöglicht die Anwendung einer Fortsetzungsfunktion g, die ebenfalls FF^{ug} erfüllt, auf die sich neu ergebende Eingabekonstellation genau das gewünschte nichtdeterministische Verhalten. In den deterministischen Anwendungsfällen wird, um die Spezifikation lesbarer zu gestalten, ebenfalls mit der Funktion g fortgesetzt – das Verhalten des deterministischen Teils der Spezifikation verändert sich durch diese Maßnahme nicht.

Ein weiterer interessanter Aspekt der in Abbildung 4.4 gezeigten Spezifikation ist der ihr zugrundegelegte rekursive Charakter – das Prädikat FF^{ug} stützt sich rekursiv auf die Funktion g, die ebenfalls FF^{ug} erfüllt. Der Fixpunkt dieser rekursiven Definition entspricht dem schwächsten Prädikat, das die Gleichung erfüllt. Die zugrundegelegte Ordnung auf den Prädikaten ist die Implikationsordnung (vgl. [Koz83]).

Abschließend bleibt zu bemerken, daß beide Ausgänge entgegen ihrem sonstigen Verhalten, im Schwingungsfall nicht invers zueinander sind. Die hier gezeigte Spezifikation basiert auch auf dem Inertialdelay-Konzept und deckt alle denkbaren Eingangskombinationen ab.

An dieser Stelle wollen wir deutlich darauf hinweisen, daß obige Spezifikation für unbekannte aber gleiche Verzögerungszeiten auch den Fall verzögerungsfreier RS-Flipflops adäquat modelliert. Dies liegt daran, daß im Schwingungsfall für verzögerungsfreie Flipflops der Ausgabewert nicht berechnet sondern lediglich nichtdeterministisch aus (l, l) und (h, h) ausgewählt wird. Als Konsequenz werden wir den verzögerungsfreien Fall im RS-Flipflop bei der Betrachtung bekannter Verzögerungszeiten ausschließen.

4.2.3 Bekannte Verzögerungszeiten

In diesem Abschnitt spezifizieren wir das RS-Flipflop für den Fall bekannter Verzögerungszeiten. Das Einbeziehen von Verzögerungszeiten erfordert nun allerdings einen Wechsel der Abstraktionsebene auf die Gatterebene. Da wir die Technologieabhängigkeit am unterschiedlichen Verhalten festmachen wollen, gilt es nun als weitere Basis für unsere Untersuchungen eine modulo der Verzögerungszeit zur RT-Ebene verhaltensäquivalente, technologieunabhängige Spezifikation auf der Gatterebene zu erstellen, wobei wir geeignete Modellierungskonzepte von VHDL verwenden wollen. Diese Spezifikation wollen wir im weiteren als technologieunabhängige Spezifikation betrachten. Zur Erstellung der Spezifikation geben wir zunächst eine Modellierung des kombinatorischen Anteils, vervollständigen diese durch die Hinzunahme der Rückkopplungsschleife zu einer Beschreibung der Gesamtkomponente und binden die so entstandene Spezifikation abschließend in eine Zeitabstraktion ein. Das Einbinden der Spezifikation in eine Zeitabstraktion begründet sich durch die unterschiedlichen Zeitbasen am Eingang und im Inneren des RS-Flipflops. So könnte beispielsweise die Impulsdauer der Eingabedaten im Millisekundenbereich (gesteuert durch

Clocksignale) und die internen Gatterlaufzeiten im Nanosekundenbereich liegen. Da die Modellierung für verzögerungsfreie RS-Flipflops bereits in Abschnitt 4.2.2.2 behandelt wurde, wollen wir hier diesen Fall ausschließen.

Abbildung 4.5 zeigt das RS-Flipflop mit dem in Kapitel 3.1 motivierten modularen Aufbau. Dabei entspricht der äußere, gestrichelte Rahmen dem kombinatorischen Anteil.

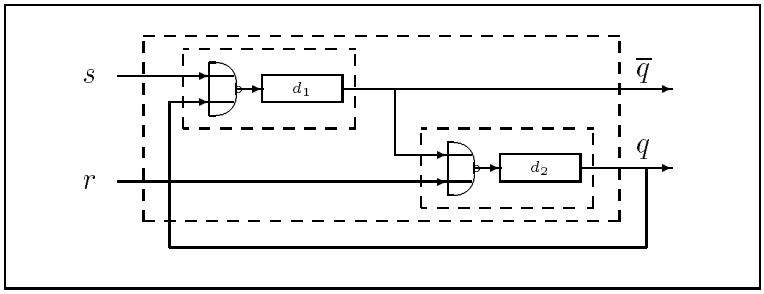


Abbildung 4.5: Modularer Aufbau des RS-Flipflops.

Die entsprechende funktionale Spezifikation gibt Abbildung 4.6 wieder. Dabei stützt

pred $NOR : ((Act^\omega)^2 \to Act^\omega_\square) \to Bool$

spec $NOR.f \equiv {}_{\forall i_1,i_2 \in Act : is_1,is_2 \in Act^\omega :}$
$f(i_1 \;\&\; is_1, i_2 \;\&\; is_2) =$ if $LO.i_1 \;\wedge\; LO.i_2$
then $h \;\&\; f(is_1, is_2)$
else $l \;\&\; f(is_1, is_2)$ fi

pred $KRS : (\mathbb{N} \times \mathbb{N}) \to [((Act^\omega)^3 \to (Act^\omega_\square)^2) \to Bool\,]$

spec $KRS(d_1, d_2).f \equiv {}_{\exists nor:NOR.nor \;\wedge\; \forall ss,rs,qs \in Act^\omega :}$
$f(ss, rs, qs) = (\overline{q}, q)$
where $\overline{q} = inertial(nor(ss, qs), init_1, d_1),$
$q = inertial(nor(\overline{q}, rs), init_2, d_2)$
where $(init_1 = h \;\wedge\; init_2 = l) \vee$
$(init_1 = l \;\wedge\; init_2 = h)$

Abbildung 4.6: Spezifikation des kombinatorischen Anteils eines RS-Flipflops.

sich die Spezifikation KRS (kombinatorischer Anteil der RS-Flipflops) auf die Spezifikation eines NOR-Gatters und auf die in Kapitel 3.5 eingeführte Funktion *inertial* zur Beschreibung der Inertialdelay-Zeitmodellierung. Der Grund dafür, daß die Rückkopplungsschleife noch unberücksichtigt bleibt, liegt in der Verwendung des bereits eingeführ-

ten Rückkopplungsoperator $\underline{\mu}$. Die Motivation für das hier gewählte Vorgehen, zunächst nur den kombinatorischen Anteil zu spezifizierten und anschließend mit Hilfe eines geeigneten Rückkopplungsoperators die Beschreibung zu vervollständigen, liegt an der schematischen Vorgehensweise zur Beschreibung rückgekoppelter Systeme. Durch die nichtdeterministische Festlegung der Initialwerte $init_1$ und $init_2$ wird der kombinatorische Anteil des RS-Flipflops in Abbildung 4.6 zu Beginn entweder gesetzt oder zurückgesetzt. Die Verzögerungszeiten d_1 und d_2 sind als Spezifikationsparameter gegeben. Der Grund dafür, warum die Initialwerte nicht auch als Spezifikationsparameter gegeben werden, liegt daran, daß die Initialwerte nicht von außen einstellbar sind – für diesen Zweck würde es einer zusätzlichen Schaltungslogik bedürfen.

4.2.3.1 Der Rückkopplungsoperator

Sequentielle Komponenten setzen sich aus kombinatorischen Komponenten zusammen, die zum einem parallel oder sequentiell miteinander verbunden sind und zum anderen mindestens einen Rückkopplungspfad, also eine Rückführung von einem Ausgang zu einem Eingang, besitzen. Diesen Rückkopplungspfad gilt es nun auch an die kombinatorische Beschreibung des RS-Flipflops anzubinden.

Unter Ausschluß des verzögerungsfreien Falls verbleibt als Zeitmodellierung das Inertialdelay-Konzept. Bei der Inertialdelay-Modellierung stecken die Initialwerte bereits in der funktionalen Spezifikation der Zeitkomponente. Dadurch erübrigt sich die Angabe eines Initialstroms im erweiterten $\underline{\mu}$-Operator. Technisch wird dies durch die Angabe des leeren Stroms anstelle des geforderten Initialstroms zum Ausdruck gebracht. Bei der Inertialdelay-Modellierung beginnen die Ausgabeströme mit den Initialwerten ($init_1$ und $init_2$) der Zeitkomponenten unmittelbar am $\overline{q}$- und q-Ausgang, sofern die Verzögerung beider NOR-Gatter größer Null sind. Dieses Verhalten entspricht auch der intuitiven Vorstellung, bei der Initialwerte die Ausgänge solange bestimmen, bis die ersten berechneten Werte vorliegen. Ist die Verzögerung einer Zeitkomponente gleich Null, so muß die Verzögerung der anderen Zeitkomponente für den Fall der Inertialdelay-Modellierung im RS-Flipflop größer Null sein. Deren Initialwerte beeinflussen dann das Verhalten des RS-Flipflops während der Initialisierungsphase und erlauben die Berechnung der Ausgabeströme. Die sich aus diesen Überlegungen ergebende Spezifikation für ein RS-Flipflop ist in Abbildung 4.7 dargestellt

$$\textbf{pred}\ RS : (\mathbb{N} \times \mathbb{N}) \to [((Act^\omega)^2 \to (Act_\square^\omega)^2) \to Bool]$$

$$\textbf{spec}\ RS(d_1, d_2).f \equiv \exists_{g:KRS(d_1,d_2).g} \wedge \forall_{ss,rs \in Act^\omega}:\ f(ss, rs) = (\underline{\mu}_{(\epsilon)} g).(ss, rs)$$

Abbildung 4.7: Spezifikation eines RS-Flipflops für bekannte Verzögerungszeiten.

Wie nun die so entstandene Spezifikation des RS-Flipflops in die Zeitabstraktion einfließt, erläutern wir im nächsten Abschnitt.

4.2.3.2 Die Zeitabstraktion

Die beiden Zeitebenen die hier zusammengebracht werden, sind die Ebene der Eingangssignale (Makroebene) und die Ebene der Gatterlaufzeiten (Mikroebene). Auf der Makroebene befindet sich unter anderem das Takttor ([Per86]) des auf der Mikroebene beschriebenen RS-Flipflops. Der Abstraktionsmechanismus erlaubt es nun eine funktionale Spezifikation zur Beschreibung des RS-Flipflopverhaltens auf der Makroebene anzugeben, wobei wir die entsprechende Spezifikation auf der Mikroebene und geeignete Repräsentations- und Abstraktionsfunktionen berücksichtigen wollen. Bevor wir die Spezifikation für das RS-Flipflops auf der Makroebene entwickeln, soll zunächst die zugehörige Abstraktionsfunktion (Analyseteil) genauer untersucht werden.

Analyse
Die Analyse dient dem Erkennen von Schwingungen und stabilem Verhalten im lokalen Ausgabestrom auf der Mikroebene. Dabei spielen

- die kleinste Zeiteinheit auf der Mikroebene ($mikro_t$),
- ein Zeitintervall auf der Makroebene wie z.B. die Taktperiode der sequentiellen Komponenten ($makro_t$),
- die Verzögerungszeiten d_1 und d_2 der beiden NOR-Gatter und
- die Tatsache, daß sich die Untersuchungen auf das RS-Flipflop beschränken,

eine entscheidende Rolle. Weiter ist wichtig, ob nun das Verhalten auf der Mikroebene innerhalb des festgelegten Zeitintervalls auf der Makroebene (also z.B. innerhalb der Taktperiode) stabil wird. Dazu betrachten wir zunächst die Stabilisierungszeiten des RS-Flipflops und untersuchen dann, unter welchen Gegebenheiten eine sinnvolle Verwendung des Analyseteils in Hinblick auf die Taktperiode $makro_t$ gegeben ist. Tabelle 4.1 präsentiert, basierend auf Abbildung 4.5, die Stabilisierungszeiten des RS-Flipflops für gegebene Zustände (gegeben durch die Belegungen von $\overline{q}$ und q) und gegebenen Eingabebelegungen (gegeben durch die Belegungen der Eingabeströme), wobei die letzten drei Zeilen für den Schwingungsfall den "worst-case" erfassen. Um zum einen den Analyseteils möglichst einfach zu gestalten, und zum anderen das Verhalten des RS-Flipflops für alle denkbaren Anwendungsfälle adäquat modellieren zu können, ist es notwendig, die Analyse zumindest basierend auf der kleinsten oberen Schranke für die Stabilisierungszeiten durchzuführen. Gemäß Tabelle 4.1 ergibt sich für die kleinste obere Schranke "$d_1 + d_2$". Natürlich stabilisiert sich das RS-Flipflop für bestimmte Zustände und Eingabebelegungen früher, aber die Forderung in diesem Kapitel, alle Eingabekombinationen zu modellieren, rechtfertigt die hier gewählte Vorgehensweise. Je nach Anwendung kann aber auch ein Zeitintervall größer $d_1 + d_2$ für die Ermittlung der stabilen Werte herangezogen werden. Dennoch schränkt die kleinste obere Schranke der Stabilisierungszeiten eine sinnvolle Anwendbarkeit des Analyseteils und somit einen sinnvollen Einsatz der Zeitabstraktion, also der Spezifikation auf der Makroebene, bezüglich des gewählten Zeitintervalls $makro_t$ ein. Unter sinnvoller Anwendbarkeit verstehen wir, daß jede denkbare Eingabekombination das von einem Schaltungsentwickler erwartete Ergebnis am Ausgang liefert. Folgende Fälle lassen sich unterscheiden:

Zustand	Eingabebelegung	Stabilisierungszeit
"Setzen" $(\overline{q}=0, q=1)$	"Speichern" $(s=0, r=0)$	0
"Setzen" $(\overline{q}=0, q=1)$	"Setzen" $(s=1, r=0)$	0
"Setzen" $(\overline{q}=0, q=1)$	"Rücksetzen" $(s=0, r=1)$	$d_1 + d_2$
"Setzen" $(\overline{q}=0, q=1)$	"irregulär" $(s=1, r=1)$	d_2
"Rücksetzen" $(\overline{q}=1, q=0)$	"Speichern" $(s=0, r=0)$	0
"Rücksetzen" $(\overline{q}=1, q=0)$	"Rücksetzen" $(s=0, r=1)$	0
"Rücksetzen" $(\overline{q}=1, q=0)$	"Setzen" $(s=1, r=0)$	$d_1 + d_2$
"Rücksetzen" $(\overline{q}=1, q=0)$	"irregulär" $(s=1, r=1)$	d_1
"irregulär" $(\overline{q}=0, q=0)$	"irregulär" $(s=1, r=1)$	0
"irregulär" $(\overline{q}=0, q=0)$	"Setzen" $(s=1, r=0)$	d_2
"irregulär" $(\overline{q}=0, q=0)$	"Rücksetzen" $(s=0, r=1)$	d_1
"irregulär" $(\overline{q}=0, q=0)$	"Speichern" $(s=0, r=0)$	$min(d_1, d_2)$ für $d_1 \neq d_2$
"irregulär" $(\overline{q}=0, q=0)$	"Speichern" $(s=0, r=0)$	Schwingung für $d_1 = d_2$
"Schwingung"	"Setzen" $(s=1, r=0)$	$d_1 + d_2$
"Schwingung "	"Rücksetzen" $(s=0, r=1)$	$d_1 + d_2$
"Schwingung"	"irregulär" $(s=1, r=1)$	$max(d_1, d_2)$

Tabelle 4.1: Stabilisierungszeiten für das RS-Flipflop.

a) $makro_t < mikro_t$: (keine sinnvolle Anwendbarkeit)
Unter dieser Voraussetzung ist ein sinnvoller Einsatz des Analyseteils bzw. der Zeitabstraktion nicht gegeben, da sich die Eingangswerte (Stromelemente der Makroebene) schneller ändern als die kleinste, auf der Mikroebene beschreibbare Verzögerungszeit. Entsprechend der Inertialdelay-Modellierung würde jede Berechnungsphase, noch bevor ein Ergebnis vorliegen würde, durch Anlegen neuer Eingabedaten unterbrochen werden und die Ausgabewerte würden unverändert bleiben.

b) $d_1 + d_2 > makro_t \geq mikro_t$: (keine sinnvolle Anwendbarkeit)
In diesem Fall ist eine Stabilisierung des RS-Flipflops auf der Mikroebene nicht zwingend gegeben, da der stabile Ausgabewert im ungünstigsten Fall erst nach $d_1 + d_2$ Zeiteinheiten anliegt. Folglich ist auch hier keine sinnvolle Anwendbarkeit gegeben.

c) $makro_t \geq d_1 + d_2$: (sinnvolle Anwendbarkeit)
Dies ist der Standardfall für die Verwendung der Zeitabstraktion, da in der Regel davon ausgegangen werden kann, daß die Taktfrequenz ($makro_t$) sehr viel größer ist, als die auf der Mikroebene beschriebenen Gatterlaufzeiten.

Neben den Stabilisierungszeiten in Tabelle 4.1 könnte man, abgesehen vom Schwingungsfall, die stabilen Ausgangswerte kanonisch bestimmen. Aber gerade der Schwingungsfall motiviert die Vorgehensweise, erst das Ausgabeverhalten zu berechnen und dann gezielt ein Stromelement aus dem Ausgabestrom herauszugreifen. Dies ist insbesondere für ein auf der Makroebene nachfolgendes, flankengesteuertes Flipflop von Bedeutung, dessen Verhalten vom selben Takt wie das hier betrachtete RS-Flipflop festgelegt wird. Dabei bestimmt der Ausgabewert auf der Mikroebene zum Zeitpunkt des Flankenwechsels auf der Makroebene das Verhalten besagten flankengesteuerten Flipflops. In anderen Worten können nun selbst im Schwingungsfall diejenigen Ausgabewerte am Flipflop ermittelt werden, die das Verhalten des nachfolgenden, flankengesteuerten Flipflops bestimmen. In diesem Fall würde man den Analyseteil der Einfachheit halber so festlegen, daß stets (bei stabilen Werten und im Schwingungsfall) der Wert unmittelbar vor dem Flankenwechsel auf der Makroebene ausgegeben wird.

Neben der Kopplung von RS-Flipflops mit flankengesteuerten Flipflops sind auch andere Applikationsmuster denkbar. Für ein auf der Makroebene nachfolgendes Master-Slave-Flipflop wäre der Ausgabewert auf der Mikroebene vor dem Sperren der Master-Komponente und dem Öffnen der Slave-Komponente für das Verhalten maßgebend. Sollte allerdings das Verhalten einer auf der Makroebene unmittelbar nachfolgenden Komponente durch ein anderes Taktsignal bestimmt werden, so müßte diese Taktperiode als weiterer Parameter in die Analysefunktion mit eingehen. Natürlich müßte man in dieser Situation die entsprechende Analysefunktion auf eine sinnvolle Anwendbarkeit bezüglich zweier Taktperioden untersuchen. Weiter ist zu beachten, daß sich eine Verzögerung in kombinatorischen Schaltungsteilen zwischen zwei sequentiellen Komponenten unmittelbar auf deren zeitliches Zusammenspiel und somit auch auf den Analyseteil auswirken würde. Natürlich spielen bei der Modellierung getakteter Systeme Setup- und Hold-Zeiten an den Dateneingängen relativ zu den Flankenwechseln am Clocksignal eine wichtige Rolle. Wir wollen aber in dieser Arbeit von diesen Zeitbetrachtungen abstrahieren, da sie keine neuen Aspekte bezüglich der technologieabhängigen Betrachtungen des RS-Flipflops beitragen. Vielmehr würde die Hinzunahme von Setup- und Hold-Zeiten die hier entwickelten Modellierungen unnötig aufblähen und verkomplizieren.

Abbildung 4.8 präsentiert eine Funktion *analyse*, die nun speziell auf ein unmittelbar auf der Makroebene nachfolgendes, flankengesteuertes Flipflop ausgelegt ist. Dabei wird angenommen, daß sowohl ein gemeinsames Taktsignal sowie eine verzögerungsfreie Verbindung zwischen den Komponenten existiert. Zur Ermittlung der Ausgabewerte auf der Mikroebene werden stets die Stromelemente ausgegeben, die unmittelbar vor dem Taktwechsel ($makro_t$) auf der Makroebene anliegen. Diese Wahl ist dadurch motiviert,

func $analyse_{(\mathbb{N}\times\mathbb{N})} : (Act^{\omega}_{\Box})^2 \rightarrow (Act^{\omega}_{\Box})^2$

$$analyse_{(makro_t, mikro_t)}(\overline{q}s, qs) = (ft.rt^{(X-1)}.\overline{q}s, ft.rt^{(X-1)}.qs) \frown analyse_{(makro_t, mikro_t)}(rt^X.\overline{q}s, rt^X.qs)$$
where $X = makro_t \div mikro_t$

Legende:

$\overline{q}s, qs$:	Lokale Ausgabeströme auf der Mikroebene
$makro_t$:	Für die Stabilisierung dominante Zeitspanne auf der Makroebene
$mikro_t$:	Kleinste Zeiteinheit auf der Mikroebene
$. \div .$:	Operator für ganzzahlige Division

Abbildung 4.8: Die Funktion *analyse*.

sowohl im stabilen Fall (maximale Stabilisierungszeit $= d_1 + d_2$) als auch im Schwingungsfall (abgestimmt auf ein nachfolgendes, flankengesteuertes Flipflop) mit einem einfachen Analyseteil adäquate Werte zu liefern. Die Funktion *analyse* setzt dann entsprechend der Zeitspanne $makro_t \div mikro_t$ mit den nächsten Eingabewerten fort, wobei der Operator $. \div .$ der ganzzahligen Division entspricht. Der entscheidende Vorteil bei der hier vorgestellten Vorgehensweise liegt im Gegensatz zur Spezifikation des RS-Flipflops in Kapitel 4.2.2 (unbekannte aber gleiche Verzögerungen) darin, daß für einen speziellen Anwendungsfall der Analyseteil so modifiziert werden kann, daß selbst bei Schwingungen eine adäquate Schaltungsmodellierung möglich ist.

Zeitabstraktion

Im folgenden wird nun die Spezifikation $\overline{RS}$ für das RS-Flipflop auf der Makroebene angegeben. Diese Spezifikation setzt sich aus

- der Spezifikation des RS-Flipflops auf der Mikroebene (RS),
- der Abstraktionsfunktion *analyse* und
- einer Repräsentationsfunktion *repeat*

zusammen. Die Funktion *repeat* expandiert die Elemente eines Eingabestroms der Makroebene zu lokalen Eingabeströmen der Mikroebene der Länge $makro_t \div mikro_t$. Unter ausschließlicher Berücksichtigung von Fall *c*) der Verwendbarkeitsanalyse ($makro_t \geq d_1 + d_2$) ergibt sich für das RS-Flipflop die in Abbildung 4.9 gezeigte Spezifikation $\overline{RS}$.

$$\textbf{pred}\ \overline{RS} : (\mathbb{N}^4) \to [((Act^\omega)^2 \to (Act^\omega_\Box)^2) \to Bool]$$

$$\begin{aligned}\textbf{spec}\ \overline{RS}(d_1, d_2, makro_t, mikro_t).f \equiv\ & \exists g{:}RS(d_1,d_2).g \wedge \forall ss,rs \in Act^\omega{:} \\ & f(ss, rs) = analyse_{(makro_t, mikro_t)}.g(repeat.ss, repeat.rs) \\ & \textsf{where}\ repeat(y) = (ft.y)^{(makro_t \div mikro_t)} \circ repeat(rt.y)\end{aligned}$$

Abbildung 4.9: Die Spezifikation $\overline{RS}$ für ein RS-Flipflop.

Das Expandieren eines jeden Stromelements der Makroebene zu einem Strom der Länge $makro_t \div mikro_t$ geschieht, wie bereits erwähnt, mit Hilfe der Funktion *repeat*. Dabei baut *repeat* sukzessive einen Eingabestrom für die Mikroebene auf, der aus lauter endlichen Teilströmen der Länge $makro_t \div mikro_t$ besteht. Die weiteren Berechnungen stützen sich auf die Spezifikation RS des RS-Flipflops auf der Mikroebene und auf die Funktion *analyse*. An dieser Stelle wollen wir nochmals betonen, daß die hier entwickelte Spezifikation $\overline{RS}$ selbst im Schwingungsfall einen für die Schaltungsumgebung adäquaten Wert abliefert.

Diese Spezifikation ist modulo der Verzögerungszeit verhaltensäquivalent zu den technologieunabhängigen Spezifikationen auf der RT-Ebene. Abstrahiert man von der Verzögerungszeit, so stellen sich die gleichen Werte an den Ausgängen ein. Abgesehen von der irregulären Eingabesequenz ist dies offensichtlich, denn eine Realisierung mit zwei NOR-Gattern erzeugt exakt das in FF^{uu} und FF^{ug} beschriebene Ausgabeverhalten. Bezüglich der irregulären Eingabesequenz müßte die Funktion *analyse* nach erfolgter Zeitabstraktion nichtdeterministisch die Werte an den Ausgängen wählen, da der Zeitbezug sowohl quantitativ als auch qualitativ fehlen würde – dies entspräche dem Verhalten der Beschreibungen auf der RT-Ebene. Aufbauend auf dieser modulo der Verzögerungszeiten bezüglich der RT-Spezifikation verhaltensäquivalenten, technologieunabhängigen Spezifikation wollen wir im nächsten Abschnitt untersuchen, ob und in welcher Weise sich nun unterschiedliche Realisierungstechnologien auf das Verhalten des hier spezifizierten RS-Flipflops auswirken.

4.3 Technologieabhängige RS-Flipflops

Nun behandeln wir die Modellierung technologieabhängiger RS-Flipflops. Berücksichtigt werden unterschiedliche Treiberstärken, Zeitrestriktionen der Mikro- und der Makroebene sowie das Verhalten bei Schwingungen. Bereits die Behandlung unterschiedlicher Treiberstärken an den Komponentenausgängen erfordert den Wechsel des Abstraktionsniveaus von der RT-Ebene hin zur Gatterebene – dieser Wechsel wird aufgrund detailierter Untersuchungen im Schwingungsfall weiter untermauert. Es wird sich zeigen, daß gerade im Schwingungsfall, basierend auf unterschiedlichen Leistungsaufnahmen, ein signifikanter Verhaltensunterschied zwischen NMOS- und CMOS-Modellen auftritt. Charakterisierungsmerkmale wie Platzbedarf oder gar Produktionskosten ([Rug91]) bleiben unberücksichtigt.

Entsprechend der in dieser Arbeit untersuchten Realisierungstechnologien gliedert sich dieser Abschnitt in die Spezifikation von NMOS- und CMOS-RS-Flipflops. Nach Möglichkeit sollen bisher erstellte Spezifikationen (vgl. technologieunabhängige RS-Flipflop-Spezifikationen) Wiederverwendung finden. Zunächst wollen wir uns der Spezifikation der NMOS-Flipflops widmen. Es wird sich zeigen, daß die technologieunabhängigen Beschreibungen abgesehen von der Verwendung geeigneter Treiberstärken auch hier adäquat sind.

4.3.1 NMOS-RS-Flipflops

Zur Spezifikation von NMOS-RS-Flipflops geben wir, basierend auf NMOS-Treiberstärken, geeignete Modellierungen für bekannte Verzögerungszeiten an. Dabei beschreiben wir in Anlehnung an die technologieunabhängigen Spezifikationen zunächst den kombinatorischen Anteil und dann, mittels des $\underline{\mu}$-Operators, die Gesamtkomponente. Abschließend wird die so entstandene Beschreibung in eine geeignete Zeitabstraktion eingebunden, um die unterschiedlichen Zeitbasen am Eingang und im Inneren des RS-Flipflops bewältigen zu können.

4.3.1.1 Treiberstärken

In Anlehnung an die technologieunabhängigen Spezifikationen, die eingangsseitig alle in dieser Arbeit beschriebenen Treiberstärken verarbeiten, definieren wir auch hier zwei Aktionsmengen.

- $Act = \{l, h, 0, H, 1\}$
- $Act_{nmos} = \{0, H\}$

Die Aktionsmenge Act erfaßt erneut alle in dieser Arbeit berücksichtigten Treiberstärken. Act_{nmos} beinhaltet Aktionen zur Modellierung von NMOS-Treiberstärken, nämlich den starken Lowpegel "0" und den schwachen Highpegel "H". Im weiteren benutzen wir auch in diesem Abschnitt die in Abbildung 4.2 eingeführten Prädikate HI und LO zur Treiberstärkenbestimmung.

4.3.1.2 Bekannte Verzögerungszeiten

In diesem Abschnitt spezifizieren wir NMOS-RS-Flipflops für bekannte Verzögerungszeiten. In Analogie zur technologieunabhängigen Variante in 4.2.3 erstellen wir zunächst eine Spezifikation für den entsprechenden kombinatorischen Anteil, erweitern diesen mit Hilfe des $\underline{\mu}$-Operators zur Gesamtbeschreibung und binden die so entstandene Spezifikation in eine geeignete Zeitabstraktion ein. Wie bereits angedeutet, ergibt sich, daß die technologieunabhängigen Spezifikationen bis auf die Treiberstärken zur Beschreibung von NMOS-Bausteinen übernommen werden können. Gleiches gilt bis auf die Zeitabstraktion auch für CMOS-Modellierungen fehlerfreier RS-Flipflops. Ohne erneut jeden Schritt in der Spezifikationsentwicklung für NMOS-Flipflops mit bekannten Verzögerungszeiten zu motivieren, präsentieren wir im folgenden die zugehörigen Modellierungen.

Abbildung 4.10 beinhaltet die technologieabhängige Spezifikation des kombinatorischen Anteils eines RS-Flipflops. Dabei basiert die Modellierung wie gehabt auf der Spezifikation eines NOR-Gatters sowie der Zeitverzögerungsfunktion $inertial$. Die Spezifikation des NOR-Gatters und die Spezifikation des kombinatorischen Anteils eines RS-Flipflops sind für NMOS ausgelegt.

$$\textbf{pred}\ NOR_{nmos} : ((Act^{\omega})^2 \to Act^{\omega}_{nmos}) \to Bool$$

$$\begin{aligned}\textbf{spec}\ NOR_{nmos}.f \equiv\ & \forall i_1,i_2 \in Act : is_1,is_2 \in Act^{\omega} : \\ & f(i_1\ \&\ is_1, i_2\ \&\ is_2) = \textsf{if}\ LO.i_1\ \wedge\ LO.i_2 \\ & \qquad \textsf{then}\ H\ \&\ f(is_1,is_2) \\ & \qquad \textsf{else}\ 0\ \&\ f(is_1,is_2)\ \textsf{fi}\end{aligned}$$

$$\textbf{pred}\ KRS_{nmos} : (\mathbb{N}\times\mathbb{N}) \to [((Act^{\omega})^3 \to (Act^{\omega}_{nmos})^2) \to Bool\,]$$

$$\begin{aligned}\textbf{spec}\ KRS_{nmos}(d_1,d_2).f \equiv\ & \exists nor : NOR_{nmos}.nor\ \wedge\ \forall ss,rs,qs \in Act^{\omega} : \\ & f(ss,rs,qs) = (\overline{q},q) \\ & \textsf{where}\ \overline{q} = inertial(nor(ss,qs), init_1, d_1), \\ & \qquad q = inertial(nor(\overline{q},rs), init_2, d_2) \\ & \qquad \textsf{where}\ (init_1 = H\ \wedge\ init_2 = 0)\ \vee \\ & \qquad\qquad (init_1 = 0\ \wedge\ init_2 = H)\end{aligned}$$

Abbildung 4.10: Spezifikation des kombinatorischen Anteils eines NMOS-RS-Flipflops.

Die Gesamtbeschreibung eines technologieabhängigen RS-Flipflops erfolgt unter der Verwendung des $\underline{\mu}$-Operators gemäß Abbildung 4.11. Abgesehen von der Adäquatheit der Spezifikation für NMOS–Realisierungen stellt sich die Modellierung analog der technologieunabhängigen Variante in Abbildung 4.7 dar. Hier soll nochmals betont werden, daß diese Beschreibung nicht für verzögerungsfreie RS-Flipflops geeignet ist.

$$\textbf{pred}\ RS_{nmos} : (\mathbb{N}\times\mathbb{N}) \to [((Act^{\omega})^2 \to (Act^{\omega}_{nmos})^2) \to Bool\,]$$

$$\textbf{spec}\ RS_{nmos}(d_1,d_2).f \equiv \exists g : KRS_{nmos}(d_1,d_2).g\ \wedge\ \forall ss,rs \in Act^{\omega} :\ \ f(ss,rs) = (\underline{\mu}_{(\epsilon)}g).(ss,rs)$$

Abbildung 4.11: Spezifikation eines NMOS-RS-Flipflops für bekannte Verzögerungszeiten.

Die Spezifikation für NMOS-RS-Flipflops auf der Makroebene entspricht bis auf die Treiberstärken der technologieunabhängigen Variante. Auch hier ist die Analysefunktion auf ein auf der Makroebene nachfolgendes, flankengesteuertes Flipflop ausgelegt. Entsprechend der technologieunabhängigen Spezifikation der Analysefunktion in Abbildung 4.8

wird sowohl im Schwingungsfall als auch bei stabilem Ausgangsverhalten der Ausgabewert unmittelbar vor dem Flankenwechsel auf der Makroebene als resultierender Wert ausgewählt. Eine sinnvolle Verwendung der Analysefunktion sowie der Spezifikation für das NMOS-RS-Flipflop auf der Makroebene ist nur für $makro_t \geq d_1 + d_2$ gegeben. Die Spezifikation $\overline{RS}_{nmos}$ und die zugehörige Analysefunktion $analyse\_nmos$ sind in den Abbildungen 4.13 und 4.12 dargestellt.

func $analyse\_nmos_{(\mathbb{N}\times\mathbb{N})} : (Act^{\omega}_{nmos})^2 \rightarrow (Act^{\omega}_{nmos})^2$

$$analyse\_nmos_{(makro_t, mikro_t)}(\overline{q}s, qs) = (ft.rt^{(X-1)}.\overline{q}s, ft.rt^{(X-1)}.qs) \frown analyse\_nmos_{(makro_t, mikro_t)}(rt^X.\overline{q}s, rt^X.qs)$$
where $X = makro_t \div mikro_t$

Legende:

$\overline{q}s, qs$:	Lokale Ausgabeströme auf der Mikroebene
$makro_t$:	Für die Stabilisierung dominante Zeitspanne auf der Makroebene
$mikro_t$:	Kleinste Zeiteinheit auf der Mikroebene
$. \div .$:	Operator für ganzzahlige Division

Abbildung 4.12: Die Funktion $analyse\_nmos$ für NMOS-RS-Flipflops.

pred $\overline{RS}_{nmos} : (\mathbb{N}^4) \rightarrow [((Act^{\omega})^2 \rightarrow (Act^{\omega}_{nmos})^2) \rightarrow Bool\,]$

spec $\overline{RS}_{nmos}(d_1, d_2, makro_t, mikro_t).f \equiv \exists g{:}RS_{nmos}(d_1,d_2).g \wedge \forall ss,rs \in Act^{\omega}{:}$
$f(ss, rs) = analyse\_nmos_{(makro_t, mikro_t)}.g(repeat.ss, repeat.rs)$
where $repeat(y) = (ft.y)^{(makro_t \div mikro_t)} \circ repeat(rt.y)$

Abbildung 4.13: Die Spezifikation $\overline{RS}_{nmos}$ für ein NMOS-RS-Flipflop.

Innerhalb der Spezifikation $\overline{RS}_{nmos}$ werden die Eingabeströme entsprechend der Zeitspanne auf der Makroebene mittels der Funktion $repeat$ expandiert, anschließend in die Funktion g, die $RS_{nmos}(d_1, d_2)$ erfüllt, eingespeist und deren Ergebnisse analysiert. Die hier vorgestellte Spezifikation $\overline{RS}_{nmos}$ einschließlich der Analysefunktion ist nicht für die CMOS-Realisierungstechnologie geeignet. Wie sich zeigen wird, kann es im Schwingungsfall bei CMOS-Komponenten im Gegensatz zu NMOS-Komponenten zur thermischen Zerstörung kommen. Dies gilt es im nächsten Abschnitt zu untersuchen und zu formalisieren.

4.3.2 CMOS-RS-Flipflops

In diesem Abschnitt modellieren wir CMOS-RS-Flipflops. Nach der Festlegung der Treiberstärken diskutieren wir den Einfluß von Gatterlaufzeiten (bekannte Verzögerungszeiten) auf das Verhalten von CMOS-Komponenten im allgemeinen und CMOS-RS-Flipflops im speziellen. Dabei wird sich zeigen, daß im Schwingungsfall thermische Schaltungszerstörung aufgrund zu hoher Leistungsaufnahme auftreten kann. Die Formalisierung dieses Phänomens und dessen Auswirkungen auf die einzelnen Spezifikationsteile schließen diesen Abschnitt.

4.3.2.1 Treiberstärken

Analog zur Festlegung der Treiberstärken für NMOS-Flipflops definieren wir die Aktionsmengen Act und Act_{cmos}. Um allerdings auch thermische Schaltungszerstörung ausdrücken zu können, führen wir zusätzlich eine um die Aktion "†" (Schaltungszerstörung) erweiterte Aktionsmenge ein.

- $Act = \{l, h, 0, H, 1\}$
- $Act_{cmos} = \{0, 1\}$
- $Act_{cmos_\dagger} = \{0, 1, \dagger\}$

Dabei beinhaltet die Aktionsmenge Act wie üblich alle in dieser Arbeit berücksichtigten Treiberstärken. Act_{cmos} berücksichtigt die CMOS-spezifischen Treiberstärken, nämlich den starken Low-Pegel "0" und den starken High-Pegel "1". $Act_{cmos_\dagger}$ entspricht der um die Aktion "†" erweiterten Aktionsmenge Act_{cmos}. Hier wollen wir bereits darauf hinweisen, daß die Aktion † bezüglich der Zeitabstraktion, also der Spezifikation auf der Makroebene, lokal ist und somit unmittelbar angrenzende Schaltungsbeschreibungen unbeeinflußt läßt. Weiter verwenden wir auch hier die Prädikate HI und LO zur Treiberstärkenbestimmung.

4.3.2.2 Bekannte Verzögerungszeiten

Dieser Abschnitt beschäftigt sich mit der Spezifikation von CMOS-RS-Flipflops für bekannte Verzögerungszeiten auf der Gatterebene, wobei der verzögerungsfreie Fall ausgeschlossen wird. Zunächst wollen wir jedoch signifikante Verhaltensunterschiede zwischen NMOS- und CMOS-Realisierungen erarbeiten.

Neben den unterschiedlichen Treiberstärken (schwacher Highpegel bei NMOS und starker Highpegel bei CMOS) kann es im Schwingungsfall bei CMOS-Realisierungen zur thermischen Schaltungszerstörung kommen, nicht aber bei NMOS-Realisierungen. Dies läßt sich anhand des Aufbaus beider Technologievarianten ([Man79, SB89]) gemäß Abbildung 4.14 begründen. Bei CMOS bilden stets zwei komplementäre MOS-Transistoren (N-Kanal und P-Kanal MOS) eine CMOS-Komponente, die als Elementarbaustein zur Realisierung von CMOS-Schaltkreisen dient. Leistung wird bei CMOS-Komponenten praktisch nur während eines Schaltvorgangs (dynamische Leistungsaufnahme) verbraucht, da zum einen beide MOS-Transistoren für eine kurze Zeit leitend sind und zum anderen die Ausgangskapazität umgeladen wird (dominanter Anteil). Die statische Leistungsaufnahme bei

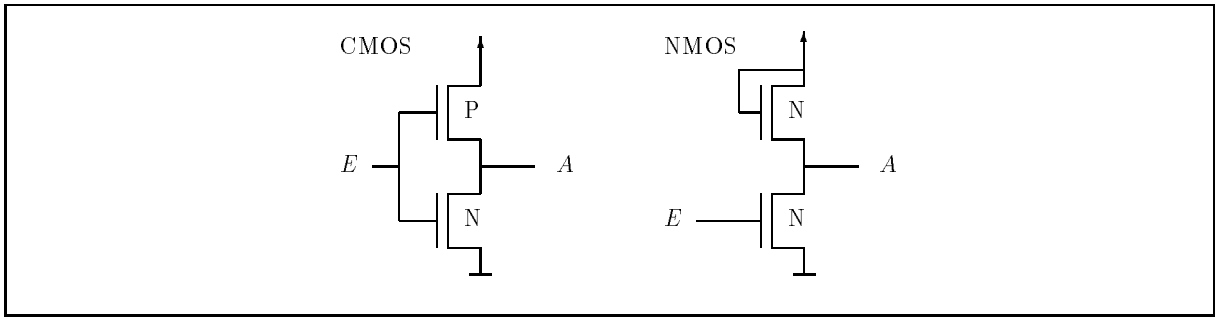


Abbildung 4.14: Elementarbausteine für CMOS- und NMOS-Schaltungen.

CMOS-Komponenten beruht nur auf parasitären Leckströmen und ist vernachlässigbar klein ([WE85]). NMOS-Komponenten sind nur mit N-Kanal MOS-Transistoren aufgebaut, wobei stets ein MOS-Transistor aus einem Paar als Lastwiderstand realisiert ist. Dies verursacht im Vergleich zu CMOS eine permanent hohe, statische Leistungsaufnahme – aber keine dynamische Leistungsaufnahme.

Die für unsere Untersuchungen interessante Fragestellung zielt nun auf das Verhalten unter Hinzunahme von Zeit ab, d.h. kann man unter Verwendung von bekannten Verzögerungszeiten der einzelnen Gatter zusätzliche Aussagen über das Verhalten von CMOS-Komponenten machen. In der Tat läßt sich feststellen, daß im Gegensatz zu NMOS-Komponenten die Leistungsaufnahme bei CMOS-Komponenten abhängig von der angelegten Schaltfrequenz ist. Gemäß [WE85, SB89] beeinflussen die Lastkapazität C_L, die Versorgungsspannung V_{DD} und die Schaltfrequenz f_S die dynamische Leistungsaufnahme einer CMOS-Komponente.

$$P_{dyn} = C_L * V_{DD}^2 * f_S$$

Dies bedeutet für ein CMOS-RS-Flipflop mit Gatterlaufzeiten im Nanosekundenbereich und einer daraus resultierenden Schwingfrequenz weit über der Megahertzgrenze, daß die dynamische Leistungsaufnahme die maximal verträgliche Leistungsaufnahme übersteigen kann. Dies führt dann zur thermischen Zerstörung der CMOS-Komponente. Diese Eigenschaft gilt es nun als eine Erweiterung in Hinblick auf Technologieabhängigkeit zu modellieren.

Zunächst stellt sich die Frage nach einer adäquaten Modellierung von Schaltungszerstörung. In der hier gewählten Variante wird die ursprüngliche Modellierung durch eine Modellierung fehlerhafter Komponenten ersetzt. Dies gestattet insbesondere die Auswirkungen thermisch zerstörter Komponenten auf die Funktionsweise des Gesamtsystems oder auf sicherheitsrelevante Teilsysteme zu studieren. Denkbar wäre auch eine Variante, basierend auf der pragmatischen Vorstellung, Hardware mit thermisch zerstörten Komponenten möglichst schnell zu erkennen. Diese Variante würde eine rasche Behebung der Ursachen erlauben und würde zu einer robusten Hardwaremodellierung führen. Letztere Variante wollen wir in dieser Arbeit nicht weiter verfolgen.

Bei der Modellierung fehlerhafter Komponenten stützen wir uns auf bereits bekannte Fehlerannahmen, die im Bereich des rechnergestützten Entwurfs von VLSI-Schaltungen eingesetzt werden ([Wad78, GCV80, Wil86]). Für die Untersuchung von thermisch zerstör-

ten CMOS-Schaltungen sind sowohl die klassischen stuck-at-0 (SA0) und stuck-at-1 (SA1) Fehlerannahmen als auch die nichtklassische stuck-at-open Fehlerannahme ([Wad78]) und deren Auswirkung auf die Ausgänge von CMOS-Komponenten von Bedeutung. Abbildung 4.15 motiviert das Auftreten dieser Fehler anhand eines NOR-Gatters (Grundbaustein eines RS-Flipflops) und verdeutlicht deren Wirkungen.

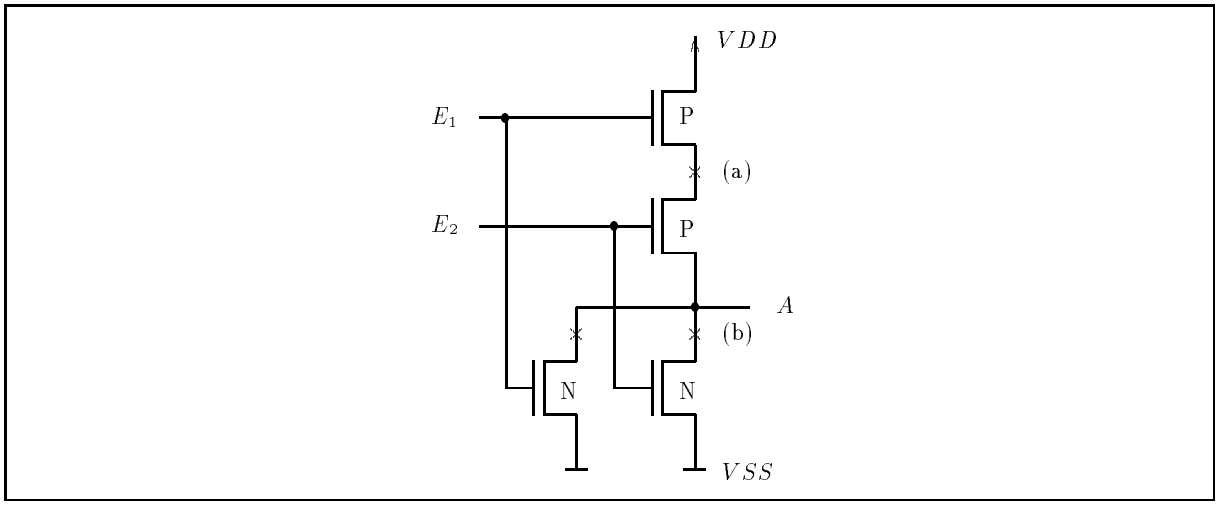


Abbildung 4.15: Fehlerannahmen für ein CMOS-NOR-Gatter.

Die klassischen stuck-at Fehlerannahmen sind Fehler deren Auswirkungen ständig meßbar sind. Ein stuck-at-0 am Ausgang bedeutet, daß dieser ständig auf dem Null-Pegel (VSS) liegt. Ein stuck-at-1 am Ausgang bedeutet, daß dieser ständig auf dem Eins-Pegel (VDD) liegt. Bei beiden klassischen Fehlern handelt es sich um Unterbrechungen gekoppelt mit Überbrückungen von Leitungen oder Transistorstrecken. Der wesentliche Aspekt ist jedoch, daß bei diesen Fehlern der Ausgang stets auf einem festen Potential liegt. Weitaus interessanter als die stuck-at-0 und stuck-at-1 Fehlerannahmen sind stuck-at-open Fehlerannahmen. Sie basieren auf Unterbrechungen bzw. auf immer sperrenden Transistoren und versetzen den Ausgang in einen hochohmigen Zustand. Wir unterscheiden zwei Arten von stuck-at-open Fehlerannahmen. Der stuck-at-open-VDD (SOPVDD) modelliert eine Unterbrechung auf dem Verbindungsstück zu VDD und der stuck-at-open-VSS (SOPVSS) modelliert die Unterbrechung zu VSS. Man beachte, daß bei einem SOPVDD nur eine Transistorstrecke unterbrochen sein muß (siehe Abbildung 4.15 (a)), wohingegen bei einem (SOPVSS) beide Transistorstrecken (siehe Abbildung 4.15 (b)) unterbrochen sein müssen. Die Forderung nach der Verbindungsunterbrechung beider Transistorstrecken im Falle SOPVSS begründet sich darin, daß erst nach Zerstörung der zweiten Transistorstrecke die Ursache der thermischen Überhitzung (Schwingung) abgestellt ist. Außerdem wird hier das gleichzeitige Auftreten von SOPVSS und SOPVDD ausgeschlossen, da bereits beim ersten auftretenden Fehler die Ursache der thermischen Zerstörung beseitigt ist und somit das Auftreten eines weiteren Fehlers gegenstandslos wird. In [Wad78, Wil86] basiert die Modellierung für eine stuck-at-open Fehlerannahme bei einer CMOS-Komponenten auf der kapazitiven Wirkung des Leitungsstücks, das den

Ausgang mit dem Gate der nachfolgenden Komponente verbindet. Dies führt insbesondere dazu, daß die zuletzt auf dem Leitungsstück gespeicherte Ladung das Verhalten des nachfolgenden Gates unter folgenden Annahmen bestimmt:

- Der noch funktionstüchtige Teil ((a) oder (b) in Abbildung 4.15) der CMOS-Komponente ist aufgrund der Eingabebelegung gesperrt.
- Der zerstörte, aber aufgrund der Eingabebelegung freigegebene Teil ist nicht mehr in der Lage, die Leitungskapazität umzuladen.

So bleibt beispielsweise bei einer Fehlerannahme SOPVDD und einer Eingabebelegung $(E_1 = 0, E_2 = 0)$ die Ladung der Ausgangsleitung erhalten. Aufgrund der Leckströme in CMOS-Schaltungen ist die in einer Leitungskapazität gespeicherte Ladung natürlich nur von zeitlich begrenzter Dauer. Da allerdings diese Zeitspanne stark vom topologischen Aufbau (Länge sowie Breite der Leitungen und Leitungsführung) der Schaltung abhängt, wollen wir die Leckströme für unsere Betrachtungen idealisieren (Leckströme = 0).

Unter Berücksichtigung der stuck-at-0, stuck-at-1 und stuck-at-open Fehlerannahmen ergibt sich die in Abbildung 4.16 präsentierte Spezifikation für ein fehlerhaftes NOR-Gatter – dem Grundbaustein des RS-Flipflops. In den ersten 6 Zeilen der Modellierung wird der Fehlertyp nichtdeterministisch ausgewählt. Die restlichen Zeilen charakterisieren dann das Verhalten eines NOR-Gatters unter der Annahme SA0, SA1, SOPVDD und SOPVSS. Bei den stuck-at-open Fehlerannahmen sind jeweils zwei unterschiedliche Startsituationen modelliert. Die eine Startsituation beschreibt das Eintreten der thermischen Zerstörung unmittelbar nachdem die Ausgangsleitung auf Eins lag und die andere das Eintreten der thermischen Zerstörung unmittelbar nachdem die Ausgangsleitung auf Null lag.

$$\textbf{pred}\ NOR^F_{cmos} : ((Act^\omega)^2 \to Act^\omega_{cmos}) \to Bool$$

$$\begin{aligned}
\textbf{spec}\ NOR^F_{cmos}.f \equiv\ & \forall_{ss,rs \in Act^\omega}: \\
& (\ f(ss,rs) = g_{SA0}(ss,rs) \quad \vee \\
& \ \ f(ss,rs) = g_{SA1}(ss,rs) \quad \vee \\
& \ \ f(ss,rs) = g_{SOPVDD}(ss,rs,1) \quad \vee \\
& \ \ f(ss,rs) = g_{SOPVDD}(ss,rs,0) \quad \vee \\
& \ \ f(ss,rs) = g_{SOPVSS}(ss,rs,1) \quad \vee \\
& \ \ f(ss,rs) = g_{SOPVSS}(ss,rs,0) \quad)
\end{aligned}$$

where $\forall_{ss,rs \in Act^\omega; z \in Act_{cmos}}$:

$$g_{SA0}(ss,rs) = 0^{min(\#ss,\#rs)},$$

$$g_{SA1}(ss,rs) = 1^{min(\#ss,\#rs)},$$

if $LO.ft.ss \ \wedge \ LO.ft.rs$
then $g_{SOPVDD}(ss,rs,z) = z \ \& \ g_{SOPVDD}(rt.ss, rt.rs, z)$
else $g_{SOPVDD}(ss,rs,z) = 0 \ \& \ g_{SOPVDD}(rt.ss, rt.rs, 0)$ fi,

if $LO.ft.ss \ \wedge \ LO.ft.rs$
then $g_{SOPVSS}(ss,rs,z) = 1 \ \& \ g_{SOPVSS}(rt.ss, rt.rs, 1)$
else $g_{SOPVSS}(ss,rs,z) = z \ \& \ g_{SOPVSS}(rt.ss, rt.rs, z)$ fi

Abbildung 4.16: Spezifikation eines fehlerhaften CMOS-NOR-Gatters.

Nun gilt es die Spezifikation eines CMOS-RS-Flipflops anzugeben, welche selbst den Fall der Schaltungszerstörung adäquat erfaßt. Grundlage dafür sind zum einen die in Abbildung 4.16 präsentierte Spezifikation eines fehlerhaften CMOS-NOR-Gatters und zum anderen die notwendigen Spezifikationsteile eines fehlerfreien CMOS-RS-Flipflops auf der Mikroebene. Letztere entsprechen modulo der Treiberstärken exakt den in Abbildung 4.10 und 4.11 aufgeführten NMOS-Spezifikationen. Der Vollständigkeit halber sind die entsprechenden Spezifikationen für ein fehlerfreies CMOS-RS-Flipflop auf der Mikroebene in Abbildung 4.17 und 4.18 aufgeführt.

$$\textbf{pred}\ NOR_{cmos} : ((Act^{\omega})^2 \to Act^{\omega}_{cmos}) \to Bool$$

$$\begin{aligned}\textbf{spec}\ NOR_{cmos}.f \equiv\ & \forall i_1,i_2 \in Act : is_1,is_2 \in Act^{\omega}: \\ & f(i_1\ \&\ is_1, i_2\ \&\ is_2) = \mathsf{if}\ LO.i_1\ \wedge\ LO.i_2 \\ & \qquad \mathsf{then}\ 1\ \&\ f(is_1,is_2) \\ & \qquad \mathsf{else}\ 0\ \&\ f(is_1,is_2)\ \mathsf{fi}\end{aligned}$$

$$\textbf{pred}\ KRS_{cmos} : (\mathbb{N}\times\mathbb{N}) \to [((Act^{\omega})^3 \to (Act^{\omega}_{cmos})^2) \to Bool\,]$$

$$\begin{aligned}\textbf{spec}\ KRS_{cmos}(d_1,d_2).f \equiv\ & \exists nor : NOR_{cmos}.nor\ \wedge\ \forall ss,rs,qs \in Act^{\omega}: \\ & f(ss,rs,qs) = (\overline{q},q) \\ & \mathsf{where}\ \overline{q} = inertial(nor(ss,qs),init_1,d_1), \\ & \qquad q = inertial(nor(\overline{q},rs),init_2,d_2) \\ & \qquad \mathsf{where}\ (init_1 = 1\ \wedge\ init_2 = 0)\ \vee \\ & \qquad\qquad (init_1 = 0\ \wedge\ init_2 = 1)\end{aligned}$$

Abbildung 4.17: Spezifikation des kombinatorischen Anteils eines CMOS-RS-Flipflops.

$$\textbf{pred}\ RS_{cmos} : (\mathbb{N}\times\mathbb{N}) \to [((Act^{\omega})^2 \to (Act^{\omega}_{cmos})^2) \to Bool\,]$$

$$\textbf{spec}\ RS_{cmos}(d_1,d_2).f \equiv \exists g : KRS_{cmos}(d_1,d_2).g\ \wedge\ \forall ss,rs \in Act^{\omega}:\ f(ss,rs) = (\underline{\mu}_{(\epsilon)}g).(ss,rs)$$

Abbildung 4.18: Spezifikation eines CMOS-RS-Flipflops für bekannte Verzögerungszeiten.

Nun wollen wir die Möglichkeit einer thermischen Zerstörung mit in die Spezifikation eines CMOS-RS-Flipflops aufzunehmen. Dabei wollen wir ausnutzen, daß die Ursache der thermischen Zerstörung (Schwingung) bereits durch die Zerstörung des ersten NOR-Gatters unterbunden wird. Abbildung 4.19 zeigt die entsprechende Modellierung KRS^{F}_{cmos}, die gegenüber der Modellierung KRS_{cmos} um die Behandlung thermischer Zerstörung erweitert ist.

$$\textbf{pred}\ KRS^F_{cmos} : (\mathbb{N}\times\mathbb{N}) \to [((Act^\omega)^3 \to (Act^\omega_{cmos})^2) \to Bool]$$

$$\begin{aligned}\textbf{spec}\ KRS^F_{cmos}(d_1,d_2).f \equiv\ & \exists nor{:}NOR_{cmos}.nor \wedge \exists n\hat{o}r{:}NOR^F_{cmos}.n\hat{o}r \wedge \forall ss,rs,qs{\in}Act^\omega{:}\\ & f(ss,rs,qs) = (\overline{q},q)\\ & \textsf{where}\ \ (\ \overline{q} = inertial(nor(ss,qs),init_1,d_1) \wedge\\ & \qquad\ q = inertial(n\hat{o}r(\overline{q},rs),init_2,d_2)\quad\ ) \vee\\ & \qquad (\ \overline{q} = inertial(n\hat{o}r(ss,qs),init_1,d_1) \wedge\\ & \qquad\ q = inertial(nor(\overline{q},rs),init_2,d_2)\quad\ )\\ & \qquad \textsf{where}\ (init_1 = 1\ \wedge\ init_2 = 0) \vee\\ & \qquad\qquad (init_1 = 0\ \wedge\ init_2 = 1)\end{aligned}$$

Abbildung 4.19: Spezifikation des kombinatorischen Anteils eines fehlerhaften CMOS-RS-Flipflops.

Die Änderungen beschränken sich auf das Ersetzen eines der beiden fehlerfreien NOR-Gatter durch ein fehlerhaftes NOR-Gatter und auf die Einführung einer nichtdeterministischen Auswahl, die bestimmt, welches der beiden NOR-Gatter fehlerhaft ist, d.h. thermisch zerstört wurde. Wie bereits erwähnt, liegt der Grund warum nur ein NOR-Gatter ersetzt wird darin, daß die Ursache der thermischen Zerstörung (Schwingung) bereits durch die Zerstörung des ersten NOR-Gatters unterbunden wird. Aufbauend auf der Spezifikation KRS^F_{cmos}, welche lediglich den kombinatorischen Anteil des RS-Flipflops erfaßt, läßt sich unter Verwendung des $\underline{\mu}$-Operators die Spezifikation RS^F_{cmos} gemäß Abbildung 4.20 angeben.

$$\textbf{pred}\ RS^F_{cmos} : (\mathbb{N}\times\mathbb{N}) \to [((Act^\omega)^2 \to (Act^\omega_{cmos})^2) \to Bool]$$

$$\textbf{spec}\ RS^F_{cmos}(d_1,d_2).f \equiv \exists g{:}KRS^F_{cmos}(d_1,d_2).g \wedge \forall ss,rs{\in}Act^\omega{:}\ \ f(ss,rs) = (\underline{\mu}_{(\epsilon)}g).(ss,rs)$$

Abbildung 4.20: Spezifikation eines fehlerhaften CMOS-RS-Flipflops.

Ein zentraler Punkt bei der Behandlung thermisch zerstörter CMOS-Komponenten ist das Erkennen einer solchen Zerstörung. Diese Aufgabe fällt der Analysefunktion in der Zeitabstraktion zu. Wie bereits gezeigt, hängt die dynamische Leistungsaufnahme von der Schaltfrequenz, der Versorgungsspannung und der umzuladenden Lastkapazität ab. Unter der Annahme, diese Werte in Form von Konstanten zur Verfügung zu haben (also ohne explizite Aufnahme in die Liste der Spezifikationsparameter), skizzieren wir die notwendigen Änderungen innerhalb der Analysefunktion anhand von Abbildung 4.21. Der wesentliche Unterschied zur NMOS- bzw. technologieunabhängigen Variante liegt in der Erkennung und Signalisierung thermischer Zerstörung – dazu ist eine Erweiterung der Indexparameter um die Gatterlaufzeiten notwendig. Im Schwingungsfall (irreguläre Eingabesequenz und beide Gatterlaufzeiten gleich) wird basierend auf der kapazitiven Last, der Versorgungs-

func $analyse\_cmos_{(\mathbb{N}^4)} : (Act^{\omega}_{cmos})^2 \times (Act^{\omega})^2 \times Act^2_{cmos} \rightarrow (Act^{\omega}_{cmos_\dagger})^2$

$analyse\_cmos_{(d_1,d_2,makro_t,mikro_t)}(\overline{q}s, qs, ss, rs, \overline{q}, q) =$
  if $(ft.ss = 0 \ \wedge \ ft.rs = 0) \ \wedge \ (\overline{q} = 0 \ \wedge \ q = 0) \wedge$
    $(d_1 = d_2) \ \wedge \ (C_L * V_{DD}^2 * \frac{1}{d_1+d_2} > P_{tot})$
  then $(\dagger^1, \dagger^1)$
  else $(ft.rt^X.\overline{q}s, ft.rt^X.qs) \frown$
  $analyse\_cmos_{(d_1,d_2,makro_t,mikro_t)}(rt^Y.\overline{q}s, rt^Y.qs, rt.ss, rt.rs, ft.rt^X.\overline{q}s, ft.rt^X.qs)$ fi
  where $X = Y - 1$,
    $Y = makro_t \div mikro_t$

Legende:

$\overline{q}s, qs$:	Lokale Ausgabeströme auf der Mikroebene
ss, rs:	Aktuelle Eingabeströme auf der Makroebene
$\overline{q}, q$:	Zuletzt berechnete Ausgabewerte für die Makroebene
d_1, d_2:	Verzögerungszeiten der beiden NOR-Gatter
$makro_t$:	Für die Stabilisierung dominante Zeitspanne auf der Makroebene
$mikro_t$:	Kleinste Zeiteinheit auf der Mikroebene
$. \div .$:	Operator für ganzzahlige Division
C_L:	Kapazitive Ausgangslast
V_{DD}:	Versorgungsspannung
P_{tot}:	Maximale Verlustleistung

Abbildung 4.21: Die Funktion *analyse_cmos* für CMOS-RS-Flipflops.

spannung, der maximal zulässigen Leistungsaufnahme und den Gatterlaufzeiten (d_1 und d_2) festgestellt, ob es aufgrund zu hoher Leistungsaufnahme zur thermischen Zerstörung des CMOS-RS-Flipflops kommt. Ist dies der Fall, so wird eine Zerstörungsmeldung $(\dagger^1, \dagger^1)$ in Form eines Tupels von zwei einelementigen Strömen ausgegeben. Hier ist eine Fortsetzung der Analysefunktion nicht notwendig, da in dieser Situation die Berechnung mit einem partiell zerstörten RS-Flipflop neu aufgesetzt wird (vgl. Spezifikation $\overline{RS}_{cmos}$ in Abbildung 4.22). Kommt es nicht zur Schaltungszerstörung, so wird unter der Annahme eines auf der Makroebene nachfolgenden, flankengesteuerten Flipflops der Wert unmittelbar vor dem Flankenwechsel ausgegeben. Die Werte C_L, V_{DD} und P_{tot} (maximal verträgliche Leistungsaufnahme) sind für eine konkrete Schaltungsmodellierung entsprechend der jeweiligen Applikation und der Datenblätter des Halbleiterherstellers festzulegen.

Als letzte Modifikation bezüglich der CMOS-Modellierung eines RS-Flipflops verbleibt die Einbindung in eine Zeitabstraktion, also die Erstellung einer Spezifikation auf der Makroebene (vgl. $\overline{RS}_{cmos}$ in Abbildung 4.22). Der Wechsel zwischen den Modellierungen einer fehlerfreien Schaltung und einer fehlerhaften Schaltung wird genau dann durchgeführt, wenn die Funktion *analyse_cmos* eine thermische Schaltungszerstörung feststellt. Tritt keine thermische Zerstörung auf, so verhält sich die Spezifikation $\overline{RS}_{cmos}$ wie die NMOS- bzw. die technologieunabhängige Variante, da stets auf dem Modell der fehler-

$$\textbf{pred}\ \overline{RS}_{cmos} : (\mathbb{N}^4) \to [((Act^\omega)^2 \to (Act^\omega_{cmos})^2) \to Bool\,]$$

$$\begin{aligned}
&\textbf{spec}\ \overline{RS}_{cmos}(d_1,d_2,makro_t,mikro_t).f \equiv \exists g{:}RS_{cmos}(d_1,d_2).g \wedge \exists \hat{g}{:}RS^F_{cmos}(d_1,d_2).\hat{g} \wedge \forall ss,rs{\in}Act^\omega{:}\\
&\quad f(ss,rs) = ffd(out,0)\\
&\quad \textsf{where}\ out = analyse_cmos_{(d_1,d_2,makro_t,mikro_t)}(g(repeat.ss,repeat.rs),ss,rs,1,1),\\
&\qquad ffd(o,k) = \textsf{if}\ ft.o = (\dagger,\dagger)\\
&\qquad\qquad \textsf{then}\ analyse_{(makro_t,mikro_t)}.\hat{g}(repeat.rt^k.ss, repeat.rt^k.rs)\\
&\qquad\qquad \textsf{else}\ ft.o \frown ffd(rt.o,k+1)\ \textsf{fi}\\
&\qquad \textsf{where}\ repeat(y) = (ft.y)^{(makro_t \div mikro_t)} \circ repeat(rt.y)
\end{aligned}$$

Abbildung 4.22: Die Spezifikation $\overline{RS}_{cmos}$ für ein CMOS-RS-Flipflop.

freien Schaltung gearbeitet wird. Das Erkennen von Schaltungszerstörung in der Spezifikation $\overline{RS}_{cmos}$ geschieht technisch unter der Verwendung der Hilfsfunktion ffd (find first destruction). Diese Funktion durchsucht sukzessive den von der Funktion* *analyse_cmos* erzeugten Ausgabestrom nach einer Zerstörungsmeldung $(\dagger,\dagger)$. Tritt keine Schaltungszerstörung auf, so erzeugt die Funktion ffd unter Verwendung des auf Tupel definierten Konkatenationsoperators $\frown$ ein Paar von Ausgabeströmen (vgl. $(\overline{q},q)$ in Abbildung 4.5). Kommt es zur Zerstörung so wird mit der Modellierung des thermisch zerstörten RS-Flipflops und den Eingaben, die zur Zerstörung geführt haben $(rt^k.ss, rt^k.rs)$, fortgesetzt. Der Parameter k bestimmt dabei die aktuelle Position im Eingabestrom. Die Modellierung im thermisch zerstörten Fall unterscheidet sich von der Modellierung im fehlerfreien Fall zum einem darin, daß sie mit der Modellierung fehlerhafter RS-Flipflops arbeitet (RS^F_{cmos}). Zum anderen kann es bei dem in unserem Sinne partiell zerstörten RS-Flipflop nicht mehr zu Schwingungen kommen – folglich wird für die Modellierung im thermisch zerstörten Fall auf eine einfachere Analysefunktion $(analyse)$ zurückgegriffen. Diese Analysefunktion entspricht, abgesehen von den Treiberstärken, der Variante, wie sie auch für die technologieunabhängige Modellierungen von RS-Flipflops verwendet wird, da nach bereits erfolgter Zerstörung eine Untersuchung in Hinblick auf eine erneute Zerstörung gegenstandslos wird. Die für die Funktion *analyse_cmos* zusätzlich eingeführte Aktion $\dagger$ bleibt lokal bezüglich der Spezifikation $\overline{RS}_{cmos}$, da die Ausgabeströme ausschließlich vom Typ Act^ω_{cmos} sind.

Zusammenfassend bleibt festzustellen, daß zwischen der NMOS- und der CMOS-Realisierungstechnologie ein signifikanter Verhaltensunterschied unter Berücksichtigung bekannter Verzögerungszeiten besteht. Dieser Verhaltensunterschied (thermische Zerstörung) äußert sich ausschließlich im Schwingungsfall und nur dann, wenn die maximale Leistungsaufnahme überschritten wird. In anderen Worten ergeben sich neben den Unterschiedlichen Treiberstärken echte Verhaltensunterschiede abhängig von der Realisierungstechnologie. Aufbauend auf dieser Erkenntnis untersuchen wir im nächsten Abschnitt die Beziehungen (Verfeinerungsbeziehung) der technologieunabhängigen RS-Flipflop-Spezifikationen zu den NMOS- und CMOS-Ausprägungen.

*Da die Funktion *analyse_cmos* auf einem flachen 6-Tupel arbeitet, müßte streng formal die Parameterliste durch $\oplus$ aus einem 2-Tupel und einem 4-Tupel aufgebaut werden.

4.4 Verfeinerung

In diesem Abschnitt untersuchen wir die Relation zwischen technologieunabhängigen und technologieabhängigen RS-Flipflop-Spezifikationen. Ausgangspunkt sind die technologieunabhängigen Spezifikationen für unbekannte Verzögerungszeiten auf der RT-Ebene und die modulo der Verzögerungszeiten verhaltensäquivalenten, technologieunabhängigen Spezifikationen auf der Gatterebene. Zunächst wollen wir die technologieabhängigen Spezifikationen mit der technologieunabhängigen Spezifikation auf der Gatterebene in Beziehung setzen und dann Rückschlüsse auf die technologieunabhängigen RT-Beschreibungen ziehen. Grundlage der Untersuchungen ist der in [Bro92a] entwickelte Verfeinerungsbegriff. Wie sich zeigen wird, gilt für RS-Flipflops nicht mehr uneingeschränkt, daß deren technologieabhängige Spezifikationen stets Verfeinerungen der technologieunabhängigen Variante darstellen. Abbildung 4.23 veranschaulicht die Verfeinerungsbeziehung zwischen technologieunabhängigen und technologieabhängigen RS-Flipflop-Spezifikationen auf der Gatterebene.

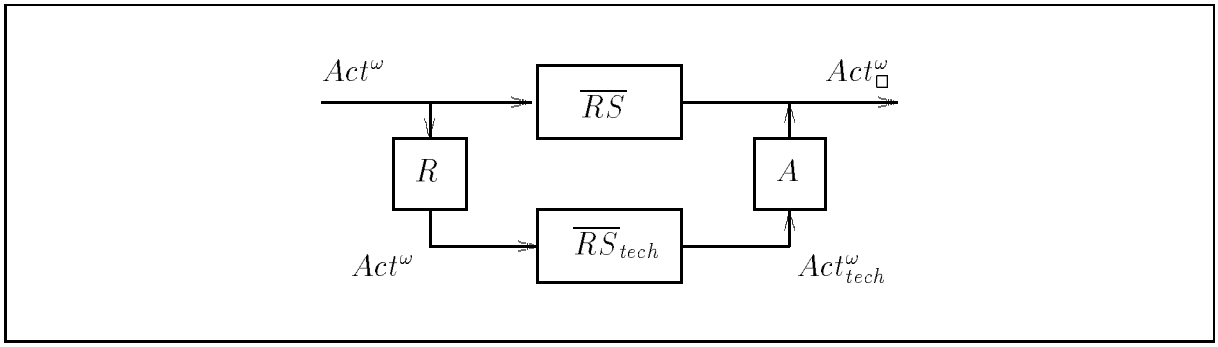


Abbildung 4.23: Verfeinerung von RS-Flipflop-Spezifikationen auf der Gatterebene.

Dabei bezeichnet $\overline{RS}$ die technologieunabhängige Spezifikation eines RS-Flipflops und $\overline{RS}_{tech}$ die entsprechende technologieunabhängige Variante, also $\overline{RS}_{nmos}$ oder $\overline{RS}_{cmos}$ – die Aktionsmenge Act_{tech} ist analog umzusetzen. Zur Vereinfachung werden die Eingabeleitungen so wie die Ausgabeleitungen durch eine Datenleitung entsprechenden Typs dargestellt. Die Repräsentations- und Abstraktionsprädikate sind mit R bzw. mit A bezeichnet. Aufbauend auf diesen Verfeinerungsschemata diskutieren wir auf informeller Basis zunächst die Beziehung zwischen der technologieunabhängigen Spezifikation eines RS-Flipflops $\overline{RS}$ und der NMOS-RS-Flipflop-Spezifikation $\overline{RS}_{nmos}$. Anschließend untersuchen wir den Zusammenhang der technologieunabhängigen Spezifikation eines RS-Flipflops $\overline{RS}$ zu der CMOS-RS-Flipflop-Spezifikation $\overline{RS}_{cmos}$. Der Zusammenhang zur Beschreibung auf der RT-Ebene wird in beiden Fällen mit berücksichtigt.

4.4.1 NMOS-RS-Flipflops

Die in dieser Arbeit erstellte Spezifikation $\overline{RS}_{nmos}$ eines NMOS-RS-Flipflops auf der Makroebene ist eine Verfeinerung der entsprechenden technologieunabhängigen Variante. Ab-

bildung 4.24 veranschaulicht die für diese Verfeinerung notwendigen Repräsentations- und Abstraktionsfunktionen, die durch die Prädikate R_{nmos} und A_{nmos} beschrieben sind.

$$\textbf{pred } R_{nmos} : (Act^{\omega} \rightarrow Act^{\omega}) \rightarrow Bool$$

$$\textbf{spec } R_{nmos}.f \equiv \forall_{s \in Act^{\omega}}:\ f.s = s$$

$$\textbf{pred } A_{nmos} : (Act^{\omega}_{nmos} \rightarrow Act^{\omega}_{\Box}) \rightarrow Bool$$

$$\textbf{spec } A_{nmos}.f \equiv \forall_{s \in Act^{\omega}_{nmos}}: \quad f(0\ \&\ s) = l\ \&\ f.s\ \wedge \quad f(H\ \&\ s) = h\ \&\ f.s$$

Abbildung 4.24: Repräsentations- und Abstraktionsfunktion für NMOS-RS-Flipflops.

Die Repräsentationsfunktion entspricht dabei der Identitätsfunktion. Die Abstraktionsfunktion hingegen bildet die starke Null auf den Low-Pegel "l" und die schwache Eins auf den High-Pegel "h" ab. Verfeinerung in unserem Sinne bedeutet nun, daß jedes Verhalten, das $\overline{RS}_{nmos}$ zeigt, auch in $\overline{RS}$ möglich ist. Das ist für die NMOS-RS-Flipflop-Spezifikation uneingeschränkt der Fall. Hier ließe sich sogar ein noch stärkerer Verfeinerungsbegriff, wie er in [HT90] eingeführt wird, verwenden. Dieser Verfeinerungsbegriff basiert auf der Äquivalenz von Spezifikationen – semantisch legt sowohl die abstraktere als auch die konkretere Spezifikation bis auf Isomorphie die gleiche Funktionsmenge fest. Die Verfeinerungsbeziehung modulo der Verzögerungszeiten zur technologieunabhängigen Variante auf der RT-Ebene ist hier offensichtlich.

4.4.2 CMOS-RS-Flipflops

Die Spezifikation eines CMOS-RS-Flipflops $\overline{RS}_{cmos}$ ist keine Verfeinerung der entsprechenden technologieunabhängigen Variante. Der Grund dafür liegt in dem sich ändernden Verhalten nach einer thermischen Zerstörung. So ist es dann nicht mehr möglich jedes Verhalten, das $\overline{RS}_{cmos}$ zeigt, auch in $\overline{RS}$ zu beobachten. Beispielsweise könnte ein CMOS-RS-Flipflop zu einem Zeitpunkt so zerstört werden, daß der Ausgang $\overline{q}$ stets auf Eins (SA1-Fehlerannahme) liegt. Unter dieser Annahme führt ein Setzen des RS-Flipflops zu einem späteren Zeitpunkt nicht dazu, daß der $\overline{q}$-Ausgang auf Null geht – der Verhaltensunterschied einer entsprechenden CMOS-Modellierung zur technologieunabhängigen Modellierung sowohl auf der Gatterebene als auch auf der RT-Ebene ist offensichtlich.

Schließt man allerdings in den Eingabesequenzen der CMOS-RS-Flipflop-Spezifikation die irreguläre Eingabesequenz aus, so erfahren der Verfeinerungsbegriff, ja sogar auch der stärkere Verfeinerungsbegriff nach [HT90], ihre Gültigkeit. Die für diesen eingeschränkten Anwendungsfall notwendigen Abstraktions- und Repräsentationsfunktionen sind in Abbildung 4.25 aufgeführt.

$$\textbf{pred}\ R_{cmos} : (Act^{\omega} \to Act^{\omega}) \to Bool$$

$$\textbf{spec}\ R_{cmos}.f \equiv \forall_{s \in Act^{\omega}}:\ f.s = s$$

$$\textbf{pred}\ A_{cmos} : (Act^{\omega}_{cmos} \to Act^{\omega}_{\Box}) \to Bool$$

$$\textbf{spec}\ A_{cmos}.f \equiv \forall_{s \in Act^{\omega}_{cmos}}: \quad f(0\ \&\ s) = l\ \&\ f.s\ \wedge \quad f(1\ \&\ s) = h\ \&\ f.s$$

Abbildung 4.25: Repräsentations- und Abstraktionsfunktion für CMOS-RS-Flipflops unter Ausschluß der irregulären Eingabesequenz.

Abschließend läßt sich zusammenfassen, daß die Spezifikation eines NMOS-RS-Flipflops uneingeschränkt der Verfeinerungsrelation nach [Bro92a] genügt. Bei der Spezifikation des CMOS-RS-Flipflops gilt dies nur für Eingaben ohne der irregulären Eingabesequenz – im allgemeinen besteht jedoch keine Verfeinerungsrelation zur technologieunabhängigen Variante. Die RT-Beschreibungen bzw. die zu den RT-Beschreibungen modulo der Verzögerungszeiten verhaltensäquivalente Spezifikation sind also in der Tat technologieabhängig, da die CMOS-Variante des RS-Flipflops in keiner Verfeinerungsbeziehung zu den technologieunabhängigen Spezifikationen steht.

Dieser Zusammenhang wird in Kapitel 6.1 nochmals graphisch veranschaulicht. Zwar bezieht sich die dort gezeigte Graphik in Abbildung 6.1 auf den konkreten Fall der Busstrukturen – diese Graphik läßt sich jedoch auch prinzipiell auf das RS-Flipflop übertragen.

Kapitel 5

Busstrukturen

Dieses Kapitel behandelt die funktionale Modellierung von Busstrukturen auf der RT-Ebene sowie auf der Gatterebene unter Verwendung der Modellierungskonzepte von VHDL. Berücksichtigt werden dabei

- die Behandlung von Konfliktsituationen,
- eine adäquate Zeitmodellierung und
- technologieabhängige Aspekte auf der Gatterebene.

Die Fragestellung nach technologieabhängigem Verhalten auf der RT-Ebene spielt die zentrale Rolle in dieser Arbeit. Insbesondere wird sich zeigen, daß gerade Buskonflikte unterschiedliches Verhalten, abhängig von der Realisierungstechnologie, verursachen – dies widerlegt die These der Technologieunabhängigkeit der Beschreibungen auf der RT-Ebene. Die betrachteten Realisierungstechnologien sind die CMOS- und die NMOS-Technologie. Der Aspekt der Technologieabhängigkeit soll anhand einer Busstruktur zur Übertragung einer Nachricht (z.B. ein Bit) untersucht werden, wobei eine beliebige Anzahl von Schaltkreiskomponenten auf die eine Busleitung lesend und/oder schreibend zugreifen dürfen. Die Anzahl dieser Schaltkreiskomponenten wollen wir über das gesamte Kapitel 5 mit n bezeichnen – n ist also eine Konstante, die je nach Bedarf entsprechend zu deklarieren ist und sich in einem Großteil der Spezifikationen zur Festlegung der Funktionalität sowie zur Indizierung von Strömen wiederfindet. Eine Erweiterung des Konzepts einer Busleitung auf Busstrukturen größerer Breite wird in Kapitel 6 erörtert. Ein wichtiger Modellierungsaspekt ist eine adäquate Zeitmodellierung. Neben der Verwendung von Inertialdelay-Zeitmodellen für die Bustreiber soll auch eine Zerodelay-Beschreibung der gesamten Busstruktur ermöglicht werden. Die hier gewählte funktionale Modellierung von Busstrukturen ergänzt die in der Literatur bekannten Ansätze [Hoo94, Hoo92, HFFM92, Her92] zur Spezifikation von Busstrukturen um die Modellierung von Buskonflikten und um die formale Behandlung von Technologieabhängigkeiten.

Dieses Kapitel unterteilt sich im wesentlichen in technologieunabhängige Modellierungen auf der RT-Ebene und in technologieabhängige Modellierungen auf der Gatterebene sowie in die Untersuchung der Zusammenhänge der Spezifikationen untereinander (Verfeinerung). Der Wechsel der Beschreibungsebene von der RT-Ebene zur Gatterebene begründet sich in der Hinzunahme technologiespezifischer Ausgangstreiberstärken sowie in

der Berücksichtigung von Gattereigenschaften. Die technologieabhängigen Spezifikationen berücksichtigen sowohl die NMOS als auch die CMOS Technologie. Beginnen wollen wir dieses Kapitel mit Betrachtungen über Aufbau und Klassifizierung von Busstrukturen.

5.1 Klassifizierung und Aufbau von Busstrukturen

Busstrukturen sind im allgemeinen dadurch gekennzeichnet, daß sie den Transport und Austausch von Daten zwischen mehreren Schaltkreiskomponenten über ein gemeinsames Medium ermöglichen. Die Richtung des Datentransports wird über Steuerleitungen von den Schaltkreiskomponenten aus geregelt. Dies erlaubt im allgemeinen, daß Schaltkreiskomponenten lesend und/oder schreibend auf die Busstruktur zugreifen.

Technisch zerfällt eine Busstruktur in Bustreiber und einem Übertragungsmedium. Abbildung 5.1 veranschaulicht eine solche Busstruktur. Das Übertragungsmedium, im

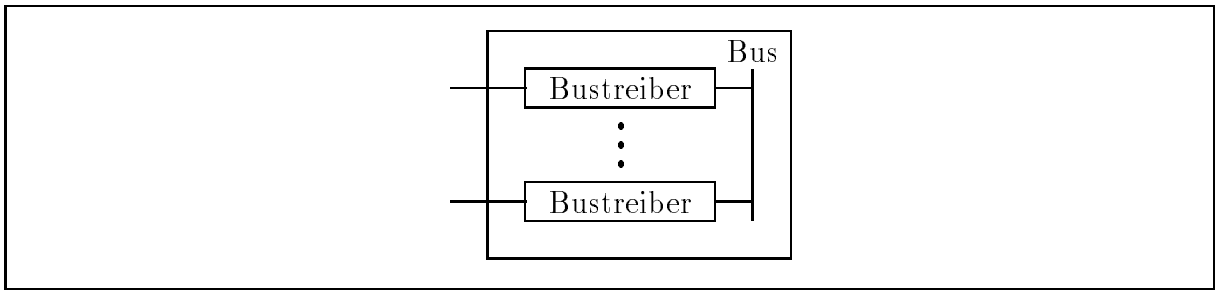


Abbildung 5.1: Schematischer Aufbau einer Busstruktur.

weiteren als Bus bezeichnet, entspricht im allgemeinen einem Bündel paralleler Leitungsstücke, deren Anzahl der jeweiligen Verarbeitungsbreite der angeschlossenen Schaltkreiskomponenten angepaßt ist. In der Literatur ([Sei90]) werden die Busstrukturen entsprechend ihrer Verwendung auch als Adreßbus, Steuerbus sowie Datenbus bezeichnet.

Ist eine Busstruktur nur für die Übertragung von Daten in einer Richtung geeignet, so sind deren Bustreiber unidirektional. In diesem Fall gilt für die an der Busstruktur angeschlossenen Schaltkreiskomponenten, daß sie stets nur lesend oder stets nur schreibend zugreifen können. Bidirektionale Bustreiber erlauben eine Datenübertragung in jeder Richtung. Hier ändert sich das Zugriffsverhalten der Schaltkreiskomponenten auf die Busstruktur dynamisch.

In dieser Arbeit betrachten wir ausschließlich Busstrukturen, die aus einer frei wählbaren Anzahl von bidirektionalen Bustreibern, also einer frei wählbaren Anzahl von angeschlossenen Schaltkreiskomponenten, sowie einem Bus bestehen, der genau eine Nachricht pro Zeiteinheit übertragen kann. Bidirektionale Bustreiber sowie die variable Anzahl von anschließbaren Schaltkreiskomponenten rechtfertigen sich durch die universelle Verwendbarkeit der entstehenden Spezifikationen.

Für die bidirektionalen Bustreiber nehmen wir an, daß deren Ausgangsstufen Tristatefähig sind. Dies bedeutet, daß neben den Werten “Low” und “High” ein dritter, “hochohmiger” Wert (mit “Z” bezeichnet) am Ausgang eines jeden Treibers eingestellt werden

kann. Dieser Wert ermöglicht es einem Bustreiber, der ausgangsseitig mit dem Bus verbunden ist, sich vom Bus abzukoppeln (d.h. den Ausgang hochohmig zu schalten), falls er nicht schreibend auf diesen zugreifen will. Es wird sich zeigen, daß bei der Modellierung von Tristate-fähigen Bustreibern unter Berücksichtigung von Konfliktsituationen interessante Unterschiede bei CMOS und NMOS auftreten, die auf Kurzschlüsse auf dem Bus zurückzuführen sind.

5.2 Technologieunabhängige Busstrukturen

In diesem Abschnitt modellieren wir Busstrukturen auf der RT-Ebene unabhängig von der Realisierungstechnologie. Dabei diskutieren wir zunächst die für die Modellierung geeigneten Treiberstärken sowie deren Einfluß auf das Verhalten bei Buskonflikten. Anschließend spezifizieren wir das Verhalten der bidirektionalen Bustreiber und des Busses um dann, aufbauend auf deren Einzelbeschreibungen, eine Spezifikation der gesamten Busstruktur zu erstellen. Abschließend gilt es noch die zeitlichen Anforderungen an die Busstruktur sicherzustellen um eine problemlose Einbettung in jeden beliebigen, zeitlichen Rahmen zu gewährleisten.

5.2.1 Treiberstärken

Analog zur Festlegung der Treiberstärken für die technologieunabhängigen Spezifikationen von Flipflops modellieren wir auch hier High-Signalpegel mit "h" und Low-Signalpegel mit "l". Ergänzend wird der hochohmige Zustand, wie er am Ausgang von Bustreibern auftreten kann, als zusätzlicher Pegel betrachtet und mit "Z" modelliert. Dabei ergeben sich für die Spezifikation der Busstruktur folgende drei Aktionsmengen:

- $Act = \{l, h, 0, H, 1\}$
- $Act_{\square} = \{l, h\}$
- $Act_Z = \{l, h, Z\}$

Die erste Aktionsmenge Act umfaßt alle für diese Arbeit relevanten Signale, die von NMOS-, CMOS- und technologieunabhängigen Komponentenbeschreibungen erzeugt werden können. Die Motivation ist erneut die Idee, Modellierungen basierend auf unterschiedlichen Realisierungstechnologien einschließlich der technologieunabhängigen Modellierungen beliebig komponieren zu können. Die Aktionsmenge $Act_{\square}$ beschränkt sich auf die Ausgangswerte technologieunabhängiger Komponentenbeschreibungen. Act_Z erweitert $Act_{\square}$ um den hochohmigen Wert "Z". Wie der weitere Verlauf dieses Kapitels noch zeigen wird, benutzen wir die Aktionsmenge Act_Z ausschließlich innerhalb der Busstrukturbeschreibungen, d.h. die Verwendung von "Z" ist lokal und beeinflußt die angrenzenden Beschreibungen nicht. Um die nachfolgenden Spezifikationen lesbarer zu gestalten, definieren wir in Analogie zu Kapitel 4 zwei Prädikate LO und HI gemäß Abbildung 5.2. LO legt diejenigen Aktionen fest, die zur Modellierung von Low-Signalpegeln benötigt werden – den technologieunabhängigen Low-Pegel "l" und den starken Low-Pegel "0", wie er sowohl für NMOS- als auch für CMOS-Modellierungen benötigt wird. HI legt

$$LO : Act \rightarrow Bool$$
$$LO.x \equiv x = 0 \vee x = l$$

$$HI : Act \rightarrow Bool$$
$$HI.x \equiv x = 1 \vee x = H \vee x = h$$

Abbildung 5.2: Klassifizierung von Treiberstärken.

diejenigen Aktionen fest, die zur Modellierung von High-Signalpegeln benötigt werden – den technologieunabhängigen High-Pegel "h", den schwachen High-Pegel "H" zur Spezifikation von NMOS-Komponenten und den starken High-Pegel "1" zur Spezifikation von CMOS-Komponenten.

5.2.2 Behandlung von Buskonflikten

Buskonflikte treten dadurch auf, daß mehrere Schaltkreiskomponenten gleichzeitig unterschiedliche Daten auf den Bus schreiben. In unserer Modellierung auf der RT-Ebene, also für den technologieunabhängigen Fall, entsprechen den Daten auf dem Bus "l", "h" und "Z", wobei "Z" als Repräsentant hochohmiger Ausgänge neutral bezüglich auftretender Konflikte ist. In anderen Worten ergeben sich Konflikte nur, falls gleichzeitig High- und Low-Signale geschrieben werden. Zur Behandlung dieser Situation greifen wir die Ideen zur Konfliktbehandlung auf, wie sie in der Hardwarebeschreibungssprache VHDL benutzt werden. Dabei sollen in diesem Abschnitt nur die technologieunabhängigen Aspekte der Konfliktlösung betrachtet werden. VHDL modelliert einen auftretenden Konflikt durch ein zusätzliches Symbol, das für einen unbekannten Wert steht (vgl. hierzu [Gro92]). Dieser Wert ist unbekannt in dem Sinne, daß er sowohl für High- als auch für Low-Signalpegel stehen kann. In unseren Spezifikationen nehmen wir genau diese Idee auf und modellieren einen Konflikt auf einem Bus, der durch gleichzeitiges Schreiben von Low- und High-Signalen entsteht, durch die nichtdeterministische Auswahl eines Wertes aus $\{l, h\}$. Diese Modellierung ist angemessen, da unterschiedliche Treiberstärken noch unberücksichtigt bleiben. Tabelle 5.3 veranschaulicht die Konfliktbehandlung (resolution) zweier Werte, wobei $\star$ einem neuen Wert entspricht, der stellvertretend für "entweder l oder h" steht – die Einführung des Wertes $\star$ verleiht der zweistelligen Resolutionsfunktion die notwendige Assoziativität. Um die Konfliktbehandlung auch für n schreibende Schaltkreiskomponenten nutzbar zu machen, geben wir eine Spezifikation RES an. Diese Spezifikation legt die Eigenschaften von Funktionen f genau so fest, daß diese zur Konfliktbehandlung n schreibender Schaltkreiskomponenten geeignet sind. RES stützt sich auf $n-1$ Funktionen res. Aus Gründen der besseren Lesbarkeit wollen wir auf eine explizite Spezifikation dieser Funktionen verzichten. Abbildung 5.4 präsentiert die Spezifikation RES, wobei n der Anzahl der anzuschließenden Schaltkreiskomponenten entspricht.

"technologieunabhängig"

res	l	h	Z	$\star$
l	l	$\star$	l	$\star$
h	$\star$	h	h	$\star$
Z	l	h	Z	$\star$
$\star$	$\star$	$\star$	$\star$	$\star$

Abbildung 5.3: Technologieunabhängige Konfliktbehandlung.

pred $RES : (Act_Z^n \to Act_Z) \to Bool$

spec $RES.f \equiv \forall_{i_1,\ldots,i_n \in Act_Z}$: $f(i_1,\ldots,i_n) =$ if $X = \star$ then l else X fi $\vee$
$f(i_1,\ldots,i_n) =$ if $X = \star$ then h else X fi
where $X = res(i_1, res(i_2,\ldots,res(i_{n-1},i_n)\ldots)$

Abbildung 5.4: Technologieunabhängige Spezifikation einer n-stelligen Resolutionsfunktion zur Konfliktbehandlung auf Bussen.

Hier soll nochmals betont werden, daß die nichtdeterministische Auswahl der sich einstellenden Werte bei Konflikten auf dem Bus ausschließlich dadurch begründet ist, daß keinerlei Aussagen über die Realisierungstechnologie gemacht wird. Im Abschnitt technologieabhängiger Busstrukturen wird sich zeigen, daß die sich ergebenden Werte auf dem Bus in Konfliktsituationen im Falle von NMOS-Realisierungen deterministisch ermittelbar sind und im Falle von CMOS-Realisierungen zu Kurzschlüssen führen.

5.2.3 Bidirektionale Bustreiber

Bidirektionale Bustreiber ermöglichen den Schaltkreiskomponenten das Lesen und/oder Schreiben auf einen gemeinsamen Bus. Der schematische Aufbau ([Man79]) eines bidirektionalen Bustreibers ergibt sich entsprechend der linken Schaltkreisskizze in Abbildung 5.5.

Neben einer gerichteten Eingangsleitung (i) zum Schreiben von Daten auf den Bus, einer gerichteten Ausgangsleitung (o_1) zum Lesen von Daten vom Bus und einem bidirektionalen Anschluß an den Bus vervollständigen zwei Selektionsleitungen die Schnittstelle. Das Verhalten eines bidirektionalen Bustreibers läßt sich in Abhängigkeit der Selektionsleitungen wie folgt zusammenfassen.

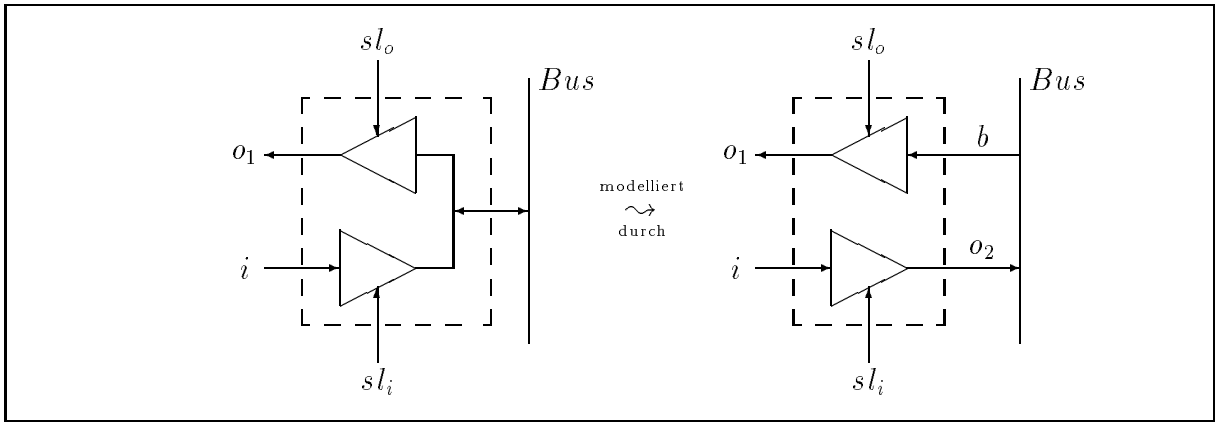


Abbildung 5.5: Schematischer Aufbau eines bidirektionalen Bustreibers.

- $(sl_i, sl_o) = (Low, Low)$ – weder Lesen noch Schreiben.
- $(sl_i, sl_o) = (Low, High)$ – Lesen vom Bus.
- $(sl_i, sl_o) = (High, Low)$ – Schreiben auf den Bus.
- $(sl_i, sl_o) = (High, High)$ – gleichzeitiges Lesen und Schreiben.

Die Modellierung eines bidirektionalen Bustreibers erfolgt nun gemäß der rechten Schaltkreisskizze in Abbildung 5.5. Neben dem Modularitätskonzept, das die Trennung von Verhaltens- und Zeitbeschreibung beinhaltet, modellieren wir die bidirektionale Leitung als zwei unabhängige, gerichtete Leitungen. Zusätzlich modellieren wir den hochohmigen Zustand (verursacht durch $sl_i = l$) des Treibers in Richtung Bus durch das Schreiben des bereits eingeführten Wertes "Z". Dies rechtfertigt sich durch die neutrale Wirkung von "Z" bei Buskonflikten. Außerdem modellieren wir den hochohmigen Zustand (verursacht durch $sl_o = l$) des Treibers in Richtung Schaltkreiskomponente durch die Angabe des zuletzt ausgegebenen Wertes. Diese Modellierung basiert auf der Annahme, daß die Leitung am Ausgang o_1 den zuletzt ausgegebenen Wert aufgrund ihrer Leitungskapazität hält. Der initiale Wert des Treibers in Richtung Schaltkreiskomponente wird nichtdeterministisch bestimmt. Eine entsprechende Modellierung auf der RT-Ebene basierend auf BT ist in Abbildung 5.6 in tabellarischer Form gegeben.

Diese Spezifikation schreibt das Modellverhalten in Abhängigkeit aller Belegungen der Selektionsleitungen fest. Die Funktion g, die letztendlich das Verhalten des Bustreibers beschreibt, stützt sich auf einen zusätzlichen Eingabeparameter zur Speicherung eines internen Zustands (Zustandsvariable a) – dieser beinhaltet den auszugebenden Wert für den Ausgang o_1, falls $sl_o = l$. Die technologieunabhängige lokale Variable $i_\square$ dient der Signalumwandlung von Act nach $Act_\square$ für das "Durchreichen" von Daten von den Schaltkreiskomponenten zum Bus. Warum wir den Typ Bt, unter Verwendung der Schlüsselworts **type**, hier explizit angeben, wird im weiteren Verlauf noch deutlich.

type $Bt = (Act^{\omega})^3 \times Act_{\square}^{\omega} \rightarrow (Act_{\square}^{\omega} \times Act_{Z}^{\omega})$

pred $BT : Bt \rightarrow Bool$

spec $BT.f \equiv \forall sl_i, sl_o, is \in Act^{\omega}; bs \in Act_{\square}^{\omega}:$

$$f(sl_i, sl_o, is, bs) = g(sl_i, sl_o, is, bs, l) \vee$$
$$f(sl_i, sl_o, is, bs) = g(sl_i, sl_o, is, bs, h)$$

where $\forall s_i, s_o, i \in Act; sl_i, sl_o, is \in Act^{\omega}; bs \in Act_{\square}^{\omega}; a, b \in Act_{\square}:$

$$g(s_i \;\&\; sl_i, s_o \;\&\; sl_o, i \;\&\; is, b \;\&\; bs, a) = (o_1, o_2) \frown g(sl_i, sl_o, is, bs, a')$$

s_i	s_o	i	b	a	o_1	o_2	a'
$LO.s_i$	$LO.s_o$	–	–	–	a	Z	a
$LO.s_i$	$HI.s_o$	–	–	–	b	Z	b
$HI.s_i$	$LO.s_o$	–	–	–	a	$i_{\square}$	a
$HI.s_i$	$HI.s_o$	–	–	–	b	$i_{\square}$	b

where $i_{\square}$ = if $HI.i$ then h else l fi

Abbildung 5.6: Technologieunabhängige Spezifikation eines Bustreibers auf der RT-Ebene.

5.2.4 Verbindungsmedium "Bus"

Bei der Modellierung des Busses gilt es eine Leitung mit einer frei wählbaren Anzahl von Anschlüssen an Schaltkreiskomponenten zu beschreiben. Die Anzahl der Anschlüsse an Schaltkreiskomponenten wird durch die jeweilige Anwendung bestimmt. Bei der Modellierung des Zeitverhaltens wird davon ausgegangen, daß der Bus selbst keine Verzögerungszeit besitzt. Sollte keiner der Bustreiber, die am Bus angeschlossen sind, zu einem Zeitpunkt auf den Bus schreiben, so entspricht der neue Spannungspegel auf dem Bus dem alten Spannungspegel gemäß der bereits vorhandenen kapazitiven Ladung auf dem Bus. Der initiale Wert auf dem Bus wird durch die nichtdeterministische Auswahl zwischen "l" und "h" modelliert. Sollte mindestens ein Bustreiber auf den Bus schreibend zugreifen, so ermittelt sich der resultierende Wert auf dem Bus unter Verwendung der in Abschnitt 5.2.2 eingeführten Resolutionsfunktion. Abbildung 5.7 zeigt die entsprechende technologieunabhängige Busspezifikation auf der RT-Ebene. Diese Spezifikation basiert auf der Spezifikation $HBUS$. Diese Technik (Verwendung einer eingebetteten Spezifikation) ist aufgrund der stets erneut nichtdeterministischen Auswahl des Buswertes im Konfliktfall erforderlich. Der zuletzt ermittelte Wert auf dem Bus wird in der Zustandsvariablen a gespeichert. Das Variieren der Fortsetzungsfunktion auf g erlaubt eine erneute nichtdeter-

$$\textbf{pred}\ BUS : ((Act_Z^\omega)^n \to Act_\square^\omega) \to Bool$$

$$\textbf{spec}\ BUS.f \equiv \exists g{:}HBUS.g \wedge \forall s_1,\ldots,s_n \in Act_Z^\omega{:}\ f(s_1,\ldots,s_n) = g(s_1,\ldots,s_n,l) \ \vee\ f(s_1,\ldots,s_n) = g(s_1,\ldots,s_n,h)$$

$$\textbf{pred}\ HBUS : ((Act_Z^\omega)^n \times Act \to Act_\square^\omega) \to Bool$$

$$\textbf{spec}\ HBUS.f \equiv \exists res{:}RES.res \wedge \exists g{:}HBUS.g \wedge \forall i_1,\ldots,i_n \in Act_Z; s_1,\ldots,s_n \in Act_Z^\omega; a \in Act_\square{:}\ f(i_1\ \&\ s_1,\ldots,i_n\ \&\ s_n,a) = \textsf{if}\ X = Z\ \textsf{then}\ a\ \&\ g(s_1,\ldots,s_n,a)\ \textsf{else}\ X\ \&\ g(s_1,\ldots,s_n,X)\ \textsf{fi}\ \textsf{where}\ X = res(i_1,\ldots,i_n)$$

Abbildung 5.7: Technologieunabhängige Spezifikation eines Busses auf der RT-Ebene.

ministische Auswahl der Resolutionsfunktionen res (vgl. Abbildung 5.4) beim nächsten Auftreten eines Konflikts.

5.2.5 Busstruktur

Eine Busstruktur, wie sie in Abbildung 5.8 veranschaulicht ist, besteht aus n Bustreibern sowie einem gemeinsamen Übertragungsmedium (BUS). Unter der Annahme, daß

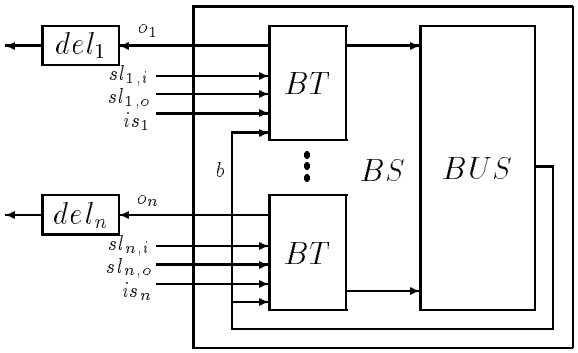


Abbildung 5.8: Schematischer Aufbau einer Busstruktur aus den Einzelbeschreibungen BT und BUS.

der Bus selbst keine Verzögerungszeit besitzt, lassen sich die Zeitbeschreibungen (del_i) für die Bustreiber (BT) über den Bus (BUS) hinweg an die Ausgänge $o_1,\ldots,o_n$ ziehen. Dabei wollen wir annehmen, daß alle verwendeten Bustreiber die gleichen Verzögerungs-

zeiten aufweisen. Diese Forderung ist unter Vernachlässigung von Fertigungstoleranzen gerechtfertigt, da Busstrukturen in der Regel homogen bezüglich der verwendeten Bauteile aufgebaut sind. Unter dieser Voraussetzung läßt sich das Modularitätskonzept, das für Beschreibungen auf der Gatterebene Gültigkeit besitzt, auch auf diesen speziellen Anwendungsfall, der Beschreibung einer Busstruktur, auf die RT-Ebene übertragen. Die so entstehende Konfiguration entspricht wieder dem Modularitätskonzept, wobei nun die Busstruktur (BS) selbst als Verhaltensbeschreibung und die Verzögerungen (del_i) an den Ausgängen $o_1, \ldots, o_n$ als zugehörige Zeitbeschreibungen verstanden werden können. Aufgrund dieser Überlegungen ist sowohl eine Zerodelay-Modellierung als auch jede beliebige Inertialdelay-Modellierung denkbar. Eine genauere Betrachtung des zeitlichen Verhaltens einschließlich der Zerodelay-Modellierung wird in Abschnitt 5.2.6 durchgeführt.

Aus Abbildung 5.8 ergibt sich nun schematisch eine technologieunabhängige Beschreibung einer Busstruktur auf der RT-Ebene. Diese ist in Abbildung 5.9 dargestellt.

$$\textbf{pred}\ BS : ((Bt)^n \times (Act^\omega)^{3*n} \rightarrow (Act^\omega_\Box)^n) \rightarrow Bool$$

$$\begin{aligned}\textbf{spec}\ BS.f \equiv\ & \exists bus{:}BUS.bus\ \wedge\ \forall bt_1,\ldots bt_n \in Bt; sl_{1,i}, sl_{1,o}, sl_{2,i}, \ldots, sl_{n,o}, is_1, \ldots, is_n \in Act^\omega: \\ & f(bt_1, \ldots, bt_n, sl_{1,i}, sl_{1,o}, sl_{2,i}, \ldots, sl_{n,o}, is_1, \ldots, is_n) = \\ & (\ \pi_1.bt_1(sl_{1,i}, sl_{1,o}, is_1, b),\ \ldots,\ \pi_1.bt_n(sl_{n,i}, sl_{n,o}, is_n, b)\ ) \\ & \textsf{where}\ b = bus(\pi_2.bt_1(sl_{1,i}, sl_{1,o}, is_1, init^1 \circ b), \ldots, \\ & \qquad \pi_2.bt_n(sl_{n,i}, sl_{n,o}, is_n, init^1 \circ b)) \\ & \qquad \textsf{where}\ (init = l\ \vee\ init = h)\end{aligned}$$

Abbildung 5.9: Technologieunabhängige Spezifikation einer Busstruktur auf der RT-Ebene.

Ein wesentliches Merkmal dieser Spezifikation ist die Festlegung des Agentennetzwerkes durch Gleichungsdefinitionen, wobei das Netzwerk durch eine Menge von (rekursiven) Stromgleichungen beschrieben wird. Die Ausgabe, die von einer stromverarbeitenden Funktion f erzeugt wird, ist ein Tupel bestehend aus n Strömen die die Werte an den Ausgängen $o_1, \ldots, o_n$ der Busstruktur beinhalten. Der zur Berechnung dieser Ausgabewerte notwendige Strom b wird durch eine rekursive Stromgleichung im lokalen Definitionsbereich (where-Konstrukt) der Spezifikation festgelegt. Der Initialstrom $init^1$ (also ein Strom der Länge 1 bestehend aus dem Wert $init$) innerhalb der rekursiven Stromgleichung ermöglicht die Berechnung eines nichttrivialen Fixpunkts ($\neq \epsilon$). Die Wahl, den Initialstrom $init^1$ so wie in Abbildung 5.9 dargestellt einzusetzen, führt dazu, daß dieser nicht auf der Leitung b erscheint (vgl. hierzu die Behandlung des Initialstroms im μ-Operator). Wie sich im Abschnitt der Zeitabstraktion zeigen wird, hat der Initialstrom $init^1$ aufgrund der dort erläuterten Zweizyklenberechnung keinen Einfluß auf die Ausgabeströme der Busstruktur – er ist also beliebig wählbar, erlaubt jedoch die Berechnung eines nichttrivialen Fixpunkts.

An dieser Stelle soll deutlich darauf hingewiesen werden, daß die Spezifikation in Abbildung 5.9 bewußt in dieser Form gewählt ist. Dies betrifft vorallem die Funktionen bt_i, die

als Parameter übergeben werden, und begründet sich in der Wiederverwendbarkeit obiger Spezifikation auch für technologieabhängige Beschreibungen. Dort wird sich zeigen, daß sich bei der CMOS-Technologie das Verhalten der Bustreiber aufgrund von Kurzschlüssen ändern kann, was sich durch die Übergabe modifizierter Bustreiberfunktionen in der entsprechenden Modellierung elegant bewerkstelligen läßt.

5.2.6 Zeitabstraktion

Aufgrund der zeitsynchronen Verarbeitung aller Komponenten gilt auch für den Bustreiber, daß dieser ein auf den Bus zu schreibendes Datum gleichzeitig mit einem vom Bus zu lesenden Datum von den Eingaben wegzunehmen hat. Um nun den zu schreibenden Wert auch beim gleichzeitigen Lesen berücksichtigen zu können, ist eine Verarbeitung in zwei Zyklen notwendig. Im ersten Zyklus schreiben die Schaltkreiskomponenten Daten auf den Bus (über die Bustreiber). Im zweiten Zyklus kann dann der neu berechnete Wert gelesen werden. Da bei der hier gezeigten Spezifikation für eine Busstruktur die Zeitbeschreibung von der Verhaltensbeschreibung getrennt ist, ist es notwendig, daß die Modellierungen sowohl für Zerodelay als auch für Inertialdelay adäquat ist. Um ein einheitliches Konzept sowohl für Zerodelay- als auch für Inertialdelay-Modellierungen zu erstellen, greifen wir auf die Zeitabstraktion zurück. Die Zeitabstraktion entspricht einer Spezifikation der Busstruktur auf einer Zeitebene ohne Zweizyklenverarbeitung – auf dieser Ebene werden beim gleichzeitigen Lesen und Schreiben die geschriebenen Werte unmittelbar beim Lesen berücksichtigt. Innerhalb dieser Spezifikation $\overline{BS}$ wird jedes Eingabeelement in den Eingabeströmen verdoppelt (Repräsentationsfunktion), diese Eingabeströme dann auf Funktionen der Spezifikation der Busstruktur BS angewandt, und aus den resultierenden Strömen jeweils nur jedes zweite Element ausgegeben (Abstraktionsfunktion). Durch diese Maßnahme ist eine Einbindung der Spezifikation BS, wie sie in Abbildung 5.9 formalisiert wurde, in jeden zeitlichen Rahmen, also auch in Zerodelay-Modellierungen gewährleistet. Die Spezifikation $\overline{BS}$ einer Busstruktur, die von der Zweizyklenabarbeitung abstrahiert, ist in Abbildung 5.10 aufgeschrieben. Das Abstraktionsniveau der Verhaltensbeschreibung entspricht der RT-Ebene und die Spezifikation ist technologieunabhängig.

pred $\overline{BS} : ((Act^\omega)^{3*n} \to (Act^\omega_\square)^n) \to Bool$

spec $\overline{BS}.f \equiv \exists bs,bt_1,\ldots,bt_n : BS.bs \wedge BT.bt_1 \wedge \ldots \wedge BT.bt_n \wedge$
$\forall sl_{1,i},sl_{1,o},sl_{2,i},\ldots,sl_{n,o},is_1,\ldots,is_n \in Act^\omega :$
$f(sl_{1,i}, sl_{1,o}, sl_{2,i}, \ldots, sl_{n,o}, is_1, \ldots, is_n) =$
$\delta.bs(bt_1, \ldots bt_n, \sigma.sl_{1,i}, \sigma.sl_{1,o}, \sigma.sl_{2,i}, \ldots, \sigma.sl_{n,o}, \sigma.is_1, \ldots, \sigma.is_n)$
where $\sigma(a \;\&\; s) = a \;\&\; a \;\&\; \sigma(s)$,
$\delta(a \frown b \frown s) = b \frown \delta(s)$

Abbildung 5.10: Die technologieunabhängige Spezifikation $\overline{BS}$ auf der RT-Ebene.

Die stromverarbeitenden Funktionen f der Spezifikation $\overline{BS}$ verarbeiten die Selektionsströme ($sl_{i,j}$) und die Dateneingabeströme (is_i). Die Initialwerte zur Festlegung der Werte an den Ausgängen $o_1, \ldots, o_n$ und des Wertes auf dem Bus werden lokal in den Spezifikationen der Bustreiber sowie des Busses nichtdeterministisch festgelegt. Das Verdoppeln jedes Eingabeelements übernimmt die Repräsentationsfunktion σ. Das Herausfiltern jedes zweiten Eingabeelements leistet die Abstraktionsfunktion δ.

Abschließend soll betont werden, daß die Schnittstelle der Spezifikation $\overline{BS}$ mit der Schnittstelle der Busstruktur in Abbildung 5.8 übereinstimmt und daß die Festlegung der Eigenschaften der Funktionen bt_i lokal in $\overline{BS}$ erfolgen. Warum die Festlegung der Funktionen bt_i und deren Eigenschaften gerade so erfolgt, wird bei der Spezifikation der CMOS-Busstruktur deutlich.

Die hier entstandene Spezifikation $\overline{BS}$ für Busstrukturen ist technologieunabhängig im Sinne von [HNS86] – sie beschreibt das Busstrukturverhalten auf der RT-Ebene und läßt Treiberstärken (ausgangsseitig) sowie Gattereigenschaften unberücksichtigt. Ob nun diese Spezifikation Verfeinerungen in NMOS- und CMOS-Modellierungen zuläßt, wird im weiteren Verlauf dieses Kapitels untersucht.

5.3 Technologieabhängige Busstrukturen

In diesem Abschnitt modellieren wir Busstrukturen auf der Gatterebene abhängig von der Realisierungstechnologie. Dabei sollen die Spezifikationen unterschiedliche Treiberstärken an den Ausgängen sowie unterschiedliches Verhalten in Konfliktfällen erfassen – letzteres bedarf insbesondere der Einbeziehung von Gattereigenschaften. Allerdings bedingt bereits das Einbeziehen der ausgangsseitigen Treiberstärken einen Wechsel der Abstraktionsebene auf das Niveau von Gatterbeschreibungen. Auf weitere Charakterisierungsmerkmale von Schaltkreistechnologien [Rug91] wie Platzbedarf, Leistungsaufnahme oder gar Produktionskosten soll nicht eingegangen werden.

Die betrachteten Realisierungstechnologien umfassen NMOS und CMOS. Entsprechend gliedert sich dieser Abschnitt in die Spezifikation von NMOS- bzw. CMOS-Busstrukturen. Dabei soll besonders darauf geachtet werden, daß bereits erstellte Spezifikationen aus Kapitel 5.2, soweit möglich, wiederverwendet werden. Beginnen wollen wir mit der Modellierung von NMOS-Busstrukturen. Dort wird sich zeigen, daß Buskonflikte unter Verwendung der Resolutionsidee von VHDL deterministisch gelöst werden können und stets zu definierten Werten auf dem Bus führen. Bei der anschließenden Modellierung von CMOS-Busstrukturen kann es im Gegensatz zu NMOS-Realisierungen zur Zerstörung der Busstruktur kommen. An dieser Stelle wollen wir analog zur Vorgehensweise in Kapitel 4 eine Modellierung einer fehlerhaften Busstruktur angeben.

5.3.1 NMOS-Busstrukturen

In diesem Teilabschnitt geben wir zunächst die für NMOS-Schaltkreise relevanten Treiberstärken an. Anschließend wird analog zu Kapitel 5.2 die zugehörige Busstruktur basierend auf einer Resolutionsfunktion, den Bustreibern sowie dem Bus spezifiziert, wobei wir gatterspezifische Eigenschaften berücksichtigen wollen. Abschließend wird für die Spezifikation der NMOS-Busstruktur noch eine geeignete Zeitabstraktion erstellt, um auch hier das Einbinden in jeden zeitlichen Rahmen und insbesondere in Zerodelay-Beschreibungen gewährleisten zu können.

5.3.1.1 Treiberstärken

Entsprechend der Modellierung der NMOS-Treiberstärken für Flipflops modellieren wir auch hier den schwachen High-Signalpegel mit "H" und den starken Low-Signalpegel mit "0". Ergänzend wird der hochohmige Zustand, wie er an den Ausgängen der Bustreiber auftreten kann, mit "Z" modelliert. Da Spezifikationen für NMOS-Bausteine auch mit CMOS- bzw. technologieunabhängigen Modellierungen eingangsseitig verbunden werden können, benötigen wir auch eine Aktionsmenge die diese Signalpegel beinhaltet. Daraus ergeben sich für die Spezifikation von NMOS-Busstrukturen auf der Gatterebene folgende Aktionsmengen:

- $Act = \{l, h, 0, H, 1\}$
- $Act_{nmos} = \{0, H\}$
- $Act_{nmos_Z} = \{0, H, Z\}$

Wie bereits für technologieunabhängige Spezifikationen von Busstrukturen gilt auch hier, daß "Z" lokal bezüglich der Busstruktur ist und somit angrenzende Modellierungen nicht beeinflußt. An dieser Stelle soll darauf hingewiesen werden, daß die in Abschnitt 5.2.1 eingeführten Prädikate LO und HI auch für die technologieabhängigen Spezifikationen von Busstrukturen verwendet werden.

5.3.1.2 Behandlung von Buskonflikten

Analog zur Behandlung der Buskonflikte für technologieunabhängige Spezifikationen von Busstrukturen greifen wir auch bei NMOS-Busstrukturen auf VHDL-Konzepte ([Gro92]) zurück. Abbildung 5.11 veranschaulicht den Resolutionsmechanismus im Konfliktfall für zwei Schaltkreiskomponenten.

"technologieabhängig $_{nmos}$"			
res_{nmos}	0	H	Z
0	0	0	0
H	0	H	H
Z	0	H	Z

Abbildung 5.11: Konfliktbehandlungen innerhalb NMOS-Busstrukturen.

Entscheidend dabei ist, daß sich beim gleichzeitigen Schreiben eines schwachen High-Signalpegels und eines starken Low-Signalpegels letzterer aufgrund des physikalischen Aufbaus von NMOS-Komponenten durchsetzt. So läßt sich in jeder Konfliktsituation ein eindeutig definierter Wert angeben. Die Dominanz der Low-Signalpegel in NMOS-Schaltungen drückt sich in der "0"-Spalte bzw. in der "0"-Reihe obiger Tabelle aus. Um die oben beschriebene Konfliktbehandlung auch für n schreibende Schaltkreiskomponenten verfügbar zu machen, geben wir analog zu Abschnitt 5.2.2 eine Spezifikation RES_{nmos} gemäß Abbildung 5.12 an.

pred $RES_{nmos} : (Act^n_{nmos_Z} \rightarrow Act_{nmos_Z}) \rightarrow Bool$

spec $RES_{nmos}.f \equiv \forall_{i_1,\ldots,i_n \in Act_{nmos_Z}}:$
$$f(i_1,\ldots,i_n) = res_{nmos}(i_1, res_{nmos}(i_2,\ldots,res_{nmos}(i_{n-1},i_n)\ldots)$$

Abbildung 5.12: Spezifikation einer NMOS-Resolutionsfunktion.

Im Gegensatz zur technologieunabhängigen Spezifikation der Resolutionsfunktion für Busstrukturen ergibt sich bei NMOS-Busstrukturen der resultierende Wert auf dem Bus deterministisch.

5.3.1.3 Bidirektionale Bustreiber

Die Modellierung eines NMOS-Bustreibers erfolgt modulo der Ausgangstreiberstärken analog zur technologieunabhängigen Modellierung von Bustreiber. Dies betrifft insbesondere das Modularitätskonzept, das Modellieren der bidirektionalen Leitung durch zwei unabhängige, gerichtete Leitungen, das Schreiben von "Z" im Falle hochohmiger Ausgänge und das Festhalten des Werts am Ausgang o_1 aufgrund von Leitungskapazitäten. Die entsprechende Spezifikation in tabellarischer Form findet sich in Abbildung 5.13.

type $Bt_{nmos} = (Act^{\omega})^3 \times Act^{\omega}_{nmos} \rightarrow (Act^{\omega}_{nmos} \times Act^{\omega}_{nmos_Z})$

pred $BT_{nmos} : Bt_{nmos} \rightarrow Bool$

spec $BT_{nmos}.f \equiv \forall sl_i, sl_o, is \in Act^{\omega}; bs \in Act^{\omega}_{nmos}:$
$f(sl_i, sl_o, is, bs) = g(sl_i, sl_o, is, bs, 0) \vee$
$f(sl_i, sl_o, is, bs) = g(sl_i, sl_o, is, bs, H)$
where $\forall s_i, s_o, i \in Act; sl_i, sl_o, is \in Act^{\omega}; bs \in Act^{\omega}_{nmos}; a, b \in Act_{nmos}:$

$g(s_i \;\&\; sl_i, s_o \;\&\; sl_o, i \;\&\; is, b \;\&\; bs, a) = (o_1, o_2) \frown g(sl_i, sl_o, is, bs, a')$

s_i	s_o	i	b	a	o_1	o_2	a'
$LO.s_i$	$LO.s_o$	–	–	–	a	Z	a
$LO.s_i$	$HI.s_o$	–	–	–	b	Z	b
$HI.s_i$	$LO.s_o$	–	–	–	a	i_{nmos}	a
$HI.s_i$	$HI.s_o$	–	–	–	b	i_{nmos}	b

where i_{nmos} = if $HI.i$ then H else 0 fi

Abbildung 5.13: Spezifikation eines NMOS-Bustreibers.

Um das eingangsseitige Anschließen von NMOS-, CMOS- und technologieunabhängigen Komponentenmodellierungen an die Spezifikationen von Bustreiber zu ermöglichen, verwenden wir die bereits eingeführten Prädikate LO und HI für das Erkennen von Low- und High-Signalpegel. Die Ausgangswerte für die Modellierung der Bustreiber beschränken sich hier entsprechend der NMOS-Technologie auf Act_{nmos} bzw. Act_{nmos_Z}. Dabei ist darauf zu achten, daß die "durchgereichten" Eingabedaten auf die Werte "H" und "0" umgesetzt werden (vgl. hierzu die technologieabhängige Variable i_{nmos} im lokalen Definitionsbereich der Spezifikation).

5.3.1.4 Verbindungsmedium "Bus"

Die Spezifikation eines NMOS-Busses entspricht im wesentlichen der technologieunabhängigen Busspezifikation aus Abschnitt 5.2.4 – allerdings ist die NMOS-Busmodellierung auf NMOS-Treiberstärken und auf die NMOS-Resolutionsfunktion abgestimmt. Die entsprechende Spezifikation ist in Abbildung 5.14 formalisiert.

$$\textbf{pred}\ BUS_{nmos} : ((Act^{\omega}_{nmos_Z})^n \rightarrow Act^{\omega}_{nmos}) \rightarrow Bool$$

$$\begin{aligned}\textbf{spec}\ BUS_{nmos}.f \equiv\ & \exists res{:}RES_{nmos}.res\ \wedge\ \forall s_1,\ldots,s_n \in Act^{\omega}_{nmos_Z}{:}\\ & f(s_1,\ldots,s_n) = g(s_1,\ldots,s_n,0)\ \vee\ f(s_1,\ldots,s_n) = g(s_1,\ldots,s_n,H)\\ & \textsf{where}\ \forall i_1,\ldots,i_n \in Act_{nmos_Z}; s_1,\ldots,s_n \in Act^{\omega}_{nmos_Z}; a \in Act_{nmos}{:}\\ & \qquad g(i_1\ \&\ s_1,\ldots,i_n\ \&\ s_n,a) =\\ & \qquad \textsf{if}\ res(i_1,\ldots,i_n) = Z\\ & \qquad \textsf{then}\ a\ \&\ g(s_1,\ldots,s_n,a)\\ & \qquad \textsf{else}\ res(i_1,\ldots,i_n)\ \&\ g(s_1,\ldots,s_n,res(i_1,\ldots,i_n))\ \textsf{fi}\end{aligned}$$

Abbildung 5.14: Spezifikation eines NMOS-Busses.

Analog zur technologieunabhängigen Spezifikation der Buskomponente wird auch bei der NMOS-Busspezifikation der "alte" Wert auf dem Bus ausgegeben, falls kein Bustreiber schreibend zugreift, also alle hochohmig sind. Falls mindestens ein Bustreiber auf den Bus schreibt, ergibt sich der neue Wert mittels der NMOS-Resolutionsfunktion. Der Initialwert auf dem Bus wird in dieser Modellierung nichtdeterministisch festgelegt und ist lokal bezüglich der Schnittstelle.

5.3.1.5 Busstruktur

Die Modellierung von NMOS-Busstrukturen ergibt sich erneut völlig schematisch durch geeignetes Zusammensetzen der Einzelspezifikationen gemäß Abbildung 5.15.

$$\textbf{pred}\ BS_{nmos} : ((Bt_{nmos})^n \times (Act^{\omega})^{3*n} \rightarrow (Act^{\omega}_{nmos})^n) \rightarrow Bool$$

$$\begin{aligned}\textbf{spec}\ BS_{nmos}.f \equiv\ & \exists bus{:}BUS_{nmos}.bus\ \wedge\ \forall bt_1,\ldots bt_n \in Bt_{nmos}; sl_{1,i}, sl_{1,o}, sl_{2,i},\ldots,sl_{n,o}, is_1,\ldots,is_n \in Act^{\omega}{:}\\ & f(bt_1,\ldots bt_n, sl_{1,i}, sl_{1,o}, sl_{2,i},\ldots,sl_{n,o}, is_1,\ldots,is_n) =\\ & \quad (\ \pi_1.bt_1(sl_{1,i}, sl_{1,o}, is_1, b),\ \ldots,\ \pi_1.bt_n(sl_{n,i}, sl_{n,o}, is_n, b)\ )\\ & \quad \textsf{where}\ b = bus(\pi_2.bt_1(sl_{1,i}, sl_{1,o}, is_1, init^1 \circ b),\ldots,\\ & \qquad\qquad \pi_2.bt_n(sl_{n,i}, sl_{n,o}, is_n, init^1 \circ b))\\ & \qquad \textsf{where}\ (init = 0\ \vee\ init = H)\end{aligned}$$

Abbildung 5.15: Spezifikation einer NMOS-Busstruktur.

Analog zur technologieunabhängigen Version erfolgt das Anbinden von Verzögerungskomponenten, die Beschreibung des Netzwerks als Gleichungssystem und die Handhabung der Bustreiber in bekannter Weise. Was die Treiberstärken betrifft, so können eingangsseitig NMOS-, CMOS- und technologieunabhängige Treiberstärken behandelt werden. Ausgangsseitig sind jedoch die Treiberstärken gemäß der zu modellierenden Realisierungstechnologie NMOS auf starke Low-Pegel und schwache High-Pegel beschränkt. Diese Spezifikation ist somit technologieabhängig und der Gatterebene zuzuordnen.

5.3.1.6 Zeitabstraktion

Die Einführung einer Zeitabstraktion für NMOS-Busstrukturen, also einer Spezifikation $\overline{BS}_{nmos}$ der Busstruktur, läßt sich analog zur technologieunabhängigen Variante auf die Zweizyklenverarbeitung zurückführen und begründet sich in der Notwendigkeit, alle Zeitkonzepte einschließlich Zerodelay modellieren zu können. Dabei wird unter Verwendung der Repräsentationsfunktion σ jedes Eingabeelement der Eingabeströme verdoppelt, diese Eingabeströme auf Funktionen gemäß der Spezifikation BS_{nmos} angewandt und aus den resultierenden Strömen mittels der Abstraktionsfunktion δ jeweils nur jedes zweite Stromelement ausgegeben. Abbildung 5.16 präsentiert die entsprechende Spezifikation $\overline{BS}_{nmos}$.

pred $\overline{BS}_{nmos} : ((Act^{\omega})^{3*n} \rightarrow (Act^{\omega}_{nmos})^{n}) \rightarrow Bool$

spec $\overline{BS}_{nmos}.f \equiv \exists bs,bt_1,\ldots,bt_n : BS_{nmos}.bs \wedge BT_{nmos}.bt_1 \wedge \ldots \wedge BT_{nmos}.bt_n \wedge$
$\forall sl_{1,i},sl_{1,o},sl_{2,i},\ldots,sl_{n,o},is_1,\ldots,is_n \in Act^{\omega}:$
$f(sl_{1,i}, sl_{1,o}, sl_{2,i}, \ldots, sl_{n,o}, is_1, \ldots, is_n) =$
$\delta.bs(bt_1, \ldots bt_n, \sigma.sl_{1,i}, \sigma.sl_{1,o}, \sigma.sl_{2,i}, \ldots, \sigma.sl_{n,o}, \sigma.is_1, \ldots, \sigma.is_n)$
where $\sigma(a \;\&\; s) = a \;\&\; a \;\&\; \sigma(s)$,
$\delta(a \frown b \frown s) = b \frown \delta(s)$

Abbildung 5.16: Die NMOS-Spezifikation $\overline{BS}_{nmos}$.

Wie bereits in der technologieunabhängigen Version sind auch hier die Initialwerte nicht sichtbar – sie stecken lokal in den Beschreibungen der Bustreiber und des Busses. Die Berücksichtigung der NMOS-Technologie drückt sich durch die Verwendung von NMOS-Bustreibern sowie einer NMOS-Busstruktur aus.

5.3.2 CMOS-Busstrukturen

Dieser Teilabschnitt befaßt sich mit der Spezifikation von CMOS-Busstrukturen. Die hier interessante Fragestellung zielt nun darauf ab, ob die Einbeziehung der Realisierungstechnologie neben unterschiedlicher Treiberstärken auch unterschiedliches Busstrukturverhalten verursachen kann. In der Tat wird sich zeigen, daß sich im Konfliktfall das Verhalten von NMOS- und CMOS-Busstrukturen unterscheidet.

Wie bereits anhand des Aufbaus von NMOS-Komponenten im Kapitel über technologieabhängige Flipflop-Spezifikationen gezeigt wurde, bestehen NMOS-Elementarbausteine aus zwei N-Kanal MOS-Transistoren, wobei der an VDD angeschlossene Transistor als Lastwiderstand realisiert ist. Dieser Lastwiderstand verursacht zum einen den schwachen High-Signalpegel, sorgt aber auch dafür, daß bei NMOS-Busstrukturen im Konfliktfall kein Kurzschluß auftritt. Wie bereits angedeutet entsteht ein solcher Kurzschluß im Konfliktfall innerhalb einer CMOS-Busstruktur. Um dies zu verdeutlichen betrachten wir zunächst einen kleinen, aber wesentlichen Ausschnitt einer Busstruktur gemäß Abbildung 5.17. Dabei handelt es sich um zwei Bustreiber BT und BT', die beide unterschiedliche

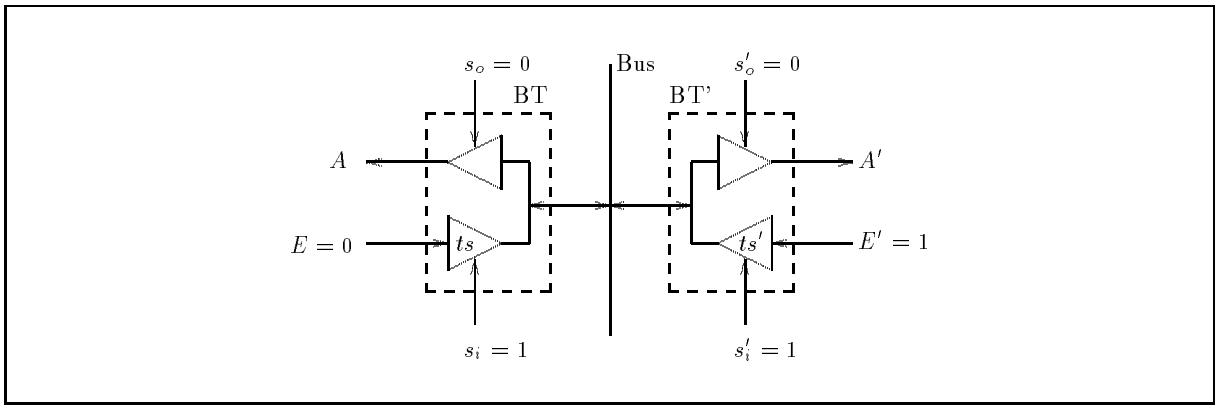


Abbildung 5.17: Ausschnitt einer CMOS-Busstruktur.

Werte auf einen gemeinsamen Bus schreiben. Bustreiber BT schreibt eine starke Null (via ts) und Bustreiber BT' eine starke Eins (via ts') – keiner der beiden Bustreiber liest vom Bus. Der entstehende Kurzschluß wird nun sichtbar, falls man die beiden Treiber ts und ts' auf Transistorebene [WE85] betrachtet. Abbildung 5.18 skizziert die entsprechende CMOS-Transistorkonfiguration. Die für diese Untersuchung irrelevanten Inverter (zur Erzeugung der Signalpegel $\overline{E}$, $\overline{E'}$, $\overline{s_i}$ und $\overline{s'_i}$) sind nicht erfaßt. Aufgrund der Werte an den Selektionsleitungen sind jeweils die beiden mittleren Transistoren von ts und ts' geöffnet. Das Schreiben einer Null am Eingang E (also $\overline{E} = 1$) öffnet den verbleibenden N-Transistor in ts und legt somit eine starke Null auf den Bus. Das Schreiben einer Eins am Eingang E' (also $\overline{E'} = 0$) öffnet den verbleibenden P-Transistor in ts' und legt somit eine starke Eins auf den Bus. Der so entstehende Kurzschluß (in Abbildung 5.18 durch die gepunktete Linie dargestellt) zerstört dann einen der beiden Treiber und führt zu einem veränderten Verhalten der Busstruktur. Die Modellierung einer CMOS-Busstruktur unter Berücksichtigung auftretender Kurzschlüsse und deren Konsequenzen soll nun unter Verwendung der bisher erstellten Spezifikationen bzw. neu zu erstellender Spezifikationen durchgeführt werden.

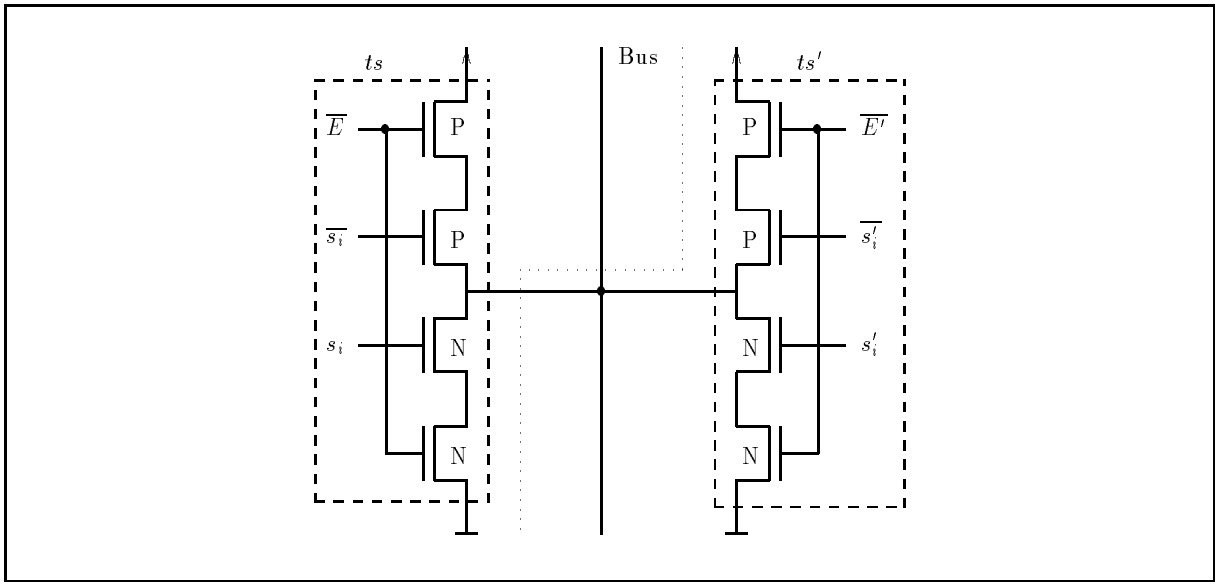


Abbildung 5.18: Ausschnitt einer CMOS-Busstruktur auf Transistorebene.

An dieser Stelle wollen wir das prinzipielle Vorgehen in diesem Abschnitt kurz erörtern um, basierend auf dem so gewonnenen Überblick, spätere Entscheidungen besser begründen zu können. Abbildung 5.19 zeigt die in die Zeitabstraktion eingebundene Spezifikation der CMOS-Busstruktur in vereinfachter Form.

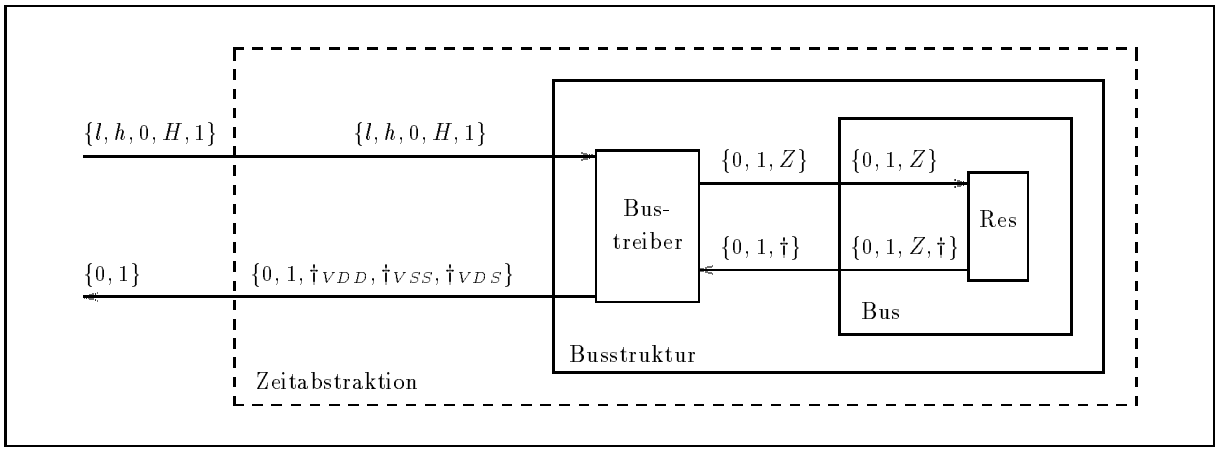


Abbildung 5.19: Vereinfachende Darstellung der Modellierung einer CMOS-Busstruktur.

Die Vereinfachung betrifft das Zusammenfassen der Bustreiber, sowie aller Eingangs- (Selektions- und Busleitungen) und Ausgangsleitungen. Die Steuerung der Bustreiber, sowie die Umsetzung beliebiger Eingangspegel am Bustreiber auf CMOS-Pegel geschieht analog zur Modellierung der NMOS-Busstruktur. Die möglichen Werte auf dem Bus umfassen die starke Null 0, die starke Eins 1 sowie den Kurzschluß, der durch "†" modelliert wird. Falls kein Kurzschluß auftritt wird der Wert auf dem Bus analog zur Modellierung der NMOS-Busstruktur behandelt und je nach Applikation nach außen weitergegeben. Tritt allerdings ein Kurzschluß auf, so kommt es zur (teilweisen) Zerstörung eines der Bustreiber, die aktiv (also durch Schreiben einer 1 oder einer 0) am Kurzschluß beteiligt

waren. Modelliert wird dies dadurch, daß der Bus im Konfliktfall an die Bustreiber den Wert † schickt, die Bustreiber selbst feststellen, ob sie aktiv auf den Bus geschrieben haben und, falls dies der Fall war, eine Nachricht über ihren möglichen Zerstörungsgrad ($\dagger_{VDD}, \dagger_{VSS}$ und $\dagger_{VDS}$) an die Zeitabstraktion schicken. Der Zerstörungsgrad hängt davon ab, welchen Wert ein Bustreiber aktiv zum Buskonflikt beigetragen hat und ob er bereits partiell zerstört war. Eine umfassende Diskussion dazu führen wir im Teilabschnitt 5.3.2.3, der Modellierung der Bustreiber. Die Zeitabstraktion wählt nun im Falle eines Buskonflikts einen der beteiligten Bustreiber, die dies selbst durch das Senden von $\dagger_{VDD}, \dagger_{VSS}$ oder $\dagger_{VDS}$ signalisieren, nichtdeterministisch aus und ersetzt diesen durch eine Modellierung eines fehlerhaften Bustreibers. Anschließend startet die Zeitabstraktion die Berechnung, die zum Buskonflikt geführt hat, erneut – allerdings nun mit einer modifizierten Busstruktur. Wesentlich dabei ist, daß die Modifikation der Busstruktur in der Zeitabstraktion stattfindet und Meldungen über mögliche Zerstörungsgrade die Zeitabstraktion nicht verlassen. Natürlich können auch mehr als zwei Bustreiber in einen Konflikt verwickelt sein und somit eine Zerstörung mehrerer Bustreiber in einem Zyklus bewirken – wie dies technisch abläuft, werden wir im weiteren Verlauf klären.

Der verbleibende Teil dieses Abschnitts behandelt zunächst die Definition der relevanten Treiberstärken zur Modellierung von CMOS-Schaltkreisen. Hier gilt es auch die Aktionen für das Auftreten von Kurzschlüssen mit in die Modellierung aufzunehmen. Anschließend befassen wir uns mit der Konfliktbehandlung innerhalb CMOS-Busstrukturen, wobei auch hier die Ideen von VHDL berücksichtigt werden. Im nächsten Teilabschnitt gilt es dann die Bustreiber zu modellieren – dies betrifft sowohl fehlerfreie als auch fehlerhafte (aufgrund eines Kurzschlusses) Bustreiber unter Einbeziehung von Treiberstärken und deren Auswirkungen im Konfliktfall. Abschließend spezifizieren wir den Bus und die Busstruktur und definieren eine Zeitabstraktion, die es gestattet, eine Beschreibung eines fehlerfreien Bustreibers durch eine Beschreibung eines fehlerhaften Bustreibers innerhalb einer Busstruktur zu ersetzen.

5.3.2.1 Treiberstärken

In Analogie zur Festlegung der Treiberstärken für CMOS-Flipflops modellieren wir auch hier den starken High-Signalpegel mit "1" und den starken Low-Signalpegel mit "0". Außerdem wird wie bisher in diesem Kapitel der hochohmige Zustand mit "Z" modelliert. Um den Kurzschluß auf dem Bus bzw. die unterschiedlichen Zerstörungsgrade in den Bustreibern ausdrücken zu können, führen wir weitere Aktionen ein ($\dagger, \dagger_{VDD}, \dagger_{VSS}$ und $\dagger_{VDS}$). Da Modellierungen von CMOS-Bausteinen von NMOS-, CMOS- und technologieunabhängigen Komponentenmodellierungen eingangsseitig angesteuert werden können, benötigen wir auch hier eine Aktionsmenge die diese Signalpegel beinhaltet. Basierend auf diesen Anforderungen definieren wir in Anlehnung an die Aktionsmengen in Abbildung 5.19 folgende Mengen:

- $Act = \{l, h, 0, H, 1\}$
- $Act_{cmos} = \{0, 1\}$
- $Act_{cmos_Z} = \{0, 1, Z\}$

- $Act_{cmos_{\dagger}} = \{0, 1, \dagger\}$
- $Act_{cmos_{Z_{\dagger}}} = \{0, 1, Z, \dagger\}$
- $Act_{cmos_{\ddagger}} = \{0, 1, \dagger_{VDD}, \dagger_{VSS}, \dagger_{VDS}\}$

An dieser Stelle wollen wir darauf hinweisen, daß "Z" und "†" bezüglich der Busstruktur und daß "$\dagger_{VDD}$", "$\dagger_{VSS}$" und "$\dagger_{VDS}$" bezüglich der Zeitabstraktion lokal sind – sie beeinflussen somit die angrenzenden Schaltungsbeschreibungen nicht.

5.3.2.2 Behandlung von Buskonflikten

Wie bereits für NMOS- und technologieunabhängige Modellierungen von Busstrukturen greifen wir auch für die Behandlung von Buskonflikten innerhalb von Modellierungen für CMOS-Busstrukturen auf VHDL-Konzepte ([Gro92]) zurück. Allerdings bedarf es hierbei im Konfliktfall einer gewissen Abweichung von den Ideen in VHDL. [Gro92] gibt für den Konfliktfall einen Wert an, der sowohl für eine starke Eins als auch für eine starke Null stehen kann – wir hingegen berücksichtigen die physikalischen Eigenschaften von CMOS und modellieren Schaltungszerstörung. Dies spiegelt sich in der Tabelle zur Konfliktbehandlung in Abbildung 5.20 wieder.

"technologieabhängig $_{cmos}$"				
res_{cmos}	0	1	Z	†
0	0	†	0	†
1	†	1	1	†
Z	0	1	Z	†
†	†	†	†	†

Abbildung 5.20: Konfliktbehandlungen innerhalb CMOS-Busstrukturen.

Beim gleichzeitigen Schreiben von "0" und "1" wird dies in Form von "†" explizit ausgedrückt. Tritt Schaltungszerstörung in Form von † auf, so setzt sich dieser Wert durch (dominante †-Zeile und Spalte in Abbildung 5.20). Die Einbindung der in Abbildung 5.20 definierten, zweistelligen Resolutionsfunktion res_{cmos} in eine n-stellige Resolutionsfunktion geschieht in Anlehnung an die NMOS- und die technologieunabhängige Version und ist in Abbildung 5.21 formalisiert.

Zu beachten ist, daß die Aktionsmenge der Ausgabedaten ($Act_{cmos_{Z_{\dagger}}}$) im Vergleich zur NMOS-Variante um das Symbol der Schaltungszerstörung angereichert ist.

pred $RES_{cmos} : (Act^n_{cmos_Z} \rightarrow Act_{cmos_{Z_t}}) \rightarrow Bool$

spec $RES_{cmos}.f \equiv \forall_{i_1,\ldots,i_n \in Act_{cmos_Z}}:$
$f(i_1,\ldots,i_n) = res_{cmos}(i_1, res_{cmos}(i_2,\ldots,res_{cmos}(i_{n-1},i_n)\ldots)$

Abbildung 5.21: Spezifikation einer CMOS-Resolutionsfunktion.

5.3.2.3 Bidirektionale Bustreiber

Die Spezifikation der bidirektionalen CMOS-Bustreiber basiert analog der Spezifikation von NMOS-Bustreibern auf dem Modularitätskonzept, der Beschreibung bidirektionaler Leitungen durch gerichtete Leitungen sowie der Modellierung des hochohmigen Zustands durch "Z". Da sich allerdings bei CMOS-Busstrukturen ein Kurzschluß auf dem Bus unmittelbar auf die aktiv beteiligten Bustreiber auswirkt, bedarf es im Vergleich zu NMOS-Bustreibern einer detailierteren Beschreibung. Diese Beschreibung orientiert sich an den Einzelkomponenten *ts* (Treiber zum Schreiben von Daten auf den Bus) und *tl* (Treiber zum Lesen von Daten vom Bus). Der Treiber *tl* ist nur eingangsseitig, d.h. nur über die Gate-Anschlüsse, mit dem Bus verbunden und somit vor einer Zerstörung durch auftretende Buskonflikte geschützt. Dies gilt nicht für einen aktiv am Buskonflikt beteiligten Treiber *ts*. Wie bereits anhand Abbildung 5.18 diskutiert, kann es aufgrund eines Kurzschlusses zur Zerstörung kommen. Da mindestens zwei Bustreiber an einem Buskonflikt aktiv beteiligt sind, steht zunächst nicht fest welcher der Bustreiber letztendlich zerstört wird. Die Idee ist nun, daß jeder der am Buskonflikt aktiv beteiligten Bustreiber ein Meldung über eine mögliche Zerstörung an die "darüberliegenden" Zeitabstraktion schickt und diese nichtdeterministisch auswählt, welcher Bustreiber dann tatsächlich zerstört wird. Bevor wir die Meldungen über mögliche Zerstörungen, die an die Zeitabstraktion zu schicken sind, formulieren, wollen wir zunächst die auf einen Kurzschluß zurückzuführenden Zerstörungen analysieren.

Analog zur Modellierung fehlerhafter RS-Flipflops in Kapitel 4.3.2 benutzen wir auch zur Modellierung zerstörter Bustreiber (durch einen Kurzschluß) bekannte Fehlerannahmen ([Wad78]). Die Fehlerannahmen umfassen die stuck-at-1 (SA1), die stuck-at-0 (SA0) und die stuck-at-open (SOP) Fehlerannahmen (vgl. Kapitel 4.3.2). Eine SA1 bzw. SA0 Fehlerannahme auf einer Leitung modelliert einen festen Eins- bzw. Null-Pegel. Die SOP Fehlerannahme modelliert eine Leitungsunterbrechung. Um das Auftreten der Fehlerannahmen in der Modellierung zu motivieren, betrachten wir nochmals die Transistorstruktur eines CMOS-Bustreibers wie sie in Abbildung 5.22 skizziert ist.

Aufgrund der Selektionsleitung s_i (bzw. $\overline{s_i}$) ist sichergestellt, daß am Ausgang A entweder eine starke Null ($\overline{E} = 1$) bzw. eine starke Eins ($\overline{E} = 0$) anliegt oder der Ausgang hochohmig ist. Die möglichen Zerstörungen bei Kurzschluß beschränken sich nun auf eine SA1 oder eine SOP Fehlerannahme, falls der Bustreiber aktiv mit dem Schreiben einer Eins am Buskonflikt beteiligt ist und dabei aufgrund des auftretenden Kurzschlusses entweder eine permanente Verbindung zu VDD entsteht (SA1) oder die Verbindung durchgetrennt wird (SOP) (vgl. (a) in Abbildung 5.22). Beim aktiven Schreiben einer Null

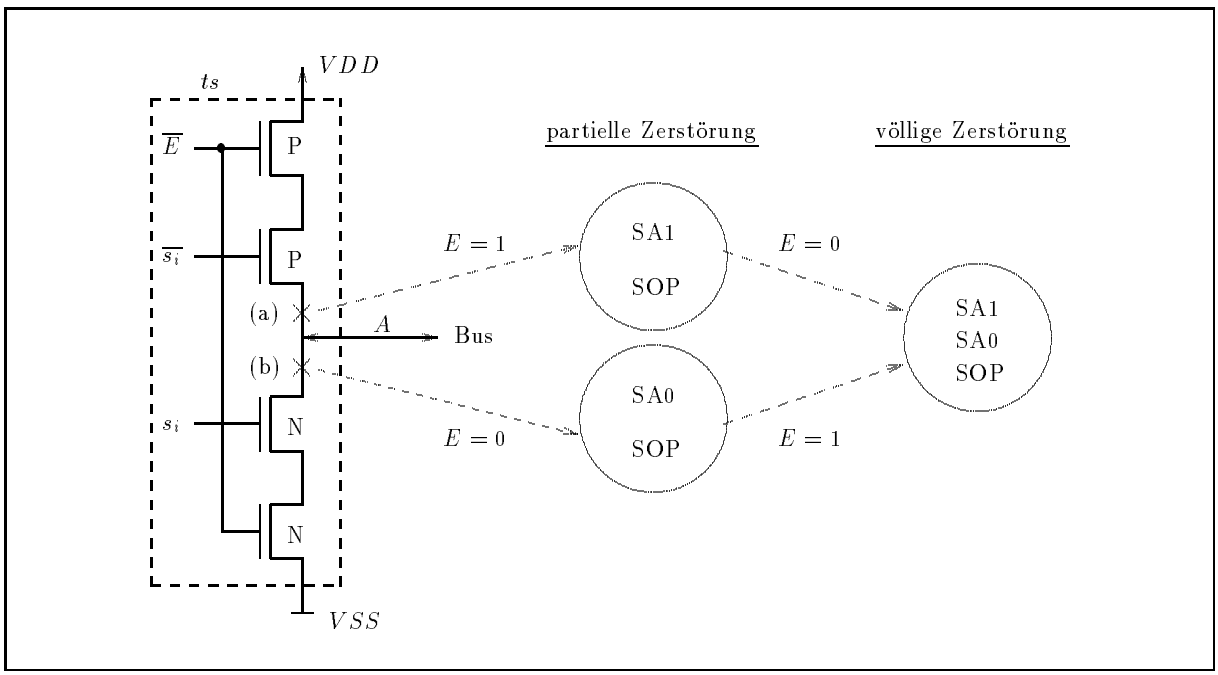


Abbildung 5.22: Fehlerannahmen für einen CMOS-Bustreiber.

im Konfliktfall modellieren wir eine auf einen Kurzschluß zurückzuführende Zerstörung durch eine SA0 oder eine SOP Fehlerannahme, da entweder eine permanente Verbindung zu VSS entsteht (SA0) oder die Verbindung durchgetrennt wird (SOP) (vgl. (b) in Abbildung 5.22). Eine partielle Zerstörung tritt auf, falls sich entweder eine Zerstörung gemäß (a) oder eine Zerstörung gemäß (b) einstellt. Die Komponente ist vollständige zerstört, falls sowohl eine Zerstörung gemäß (a) als auch eine Zerstörung gemäß (b) vorliegt. Die sich dann ergebenden Fehlerannahmen umfassen SA1, SA0 und SOP. Dabei ist darauf zu achten, daß selbst völlig zerstörte Bustreiber noch aktiv auf den Bus schreiben können (SA1, SA0) – diese Eigenschaft bleibt solange erhalten bis der Bustreiber aufgrund des selben oder weiterer Kurzschlüsse hochohmig vom Bus abgekoppelt wird (SOP).

Die oben diskutierten Zerstörungen im Kurzschlußfall lassen sich in drei Klassen einteilen und durch folgende Meldungen an die Zeitabstraktion weiterleiten:

- partielle Zerstörung (a) – Meldung: $\dagger_{VDD}$
- partielle Zerstörung (b) – Meldung: $\dagger_{VSS}$
- völlige Zerstörung (a) und (b) – Meldung: $\dagger_{VDS}$

Die Auswahl einer konkreten Fehlerannahme, also z.B. SA0 oder SOP bei partieller Zerstörung (a), findet nichtdeterministisch in der Modellierung des entsprechend zerstörten Bustreibers statt.

Neben der Zerstörung von Bustreibern, die auf das Schreiben unterschiedlicher Werte mehrerer Bustreiber zurückgehen, kann auch ein lokaler Kurzschluß in einem bereits partiell zerstörten Bustreiber auftreten. Beispielsweise würde das Schreiben einer Eins am Bustreiber, der bereits den partiellen Zerstörungsgrad SA0 aufweist, zu einem Kurzschluß

innerhalb dieses Bustreiber (also nicht über den Bus) führen. Auch diese Variante der Zerstörung gilt es zu erfassen.

Die Spezifikationen der CMOS-Bustreiber umfassen nun vier unterschiedliche Modellierungen, eine Spezifikation für einen fehlerfreien Bustreiber und drei Spezifikationen für Bustreiber mit Zerstörungen gemäß (a), (b) und (a) und (b). Die Spezifikation BT_{cmos} für den fehlerfreien Bustreiber findet sich in Abbildung 5.23.

$$\textbf{type } Bt_{cmos} = (Act^{\omega})^3 \times Act^{\omega}_{cmos_\dagger} \rightarrow (Act^{\omega}_{cmos_\ddagger} \times Act^{\omega}_{cmos_Z})$$

$$\textbf{pred } BT_{cmos} : Bt_{cmos} \rightarrow Bool$$

$$\begin{aligned}
\textbf{spec } BT_{cmos}.f \equiv\ & \forall sl_i, sl_o, is \in Act^{\omega}; bs \in Act^{\omega}_{cmos_\dagger}: \\
& f(sl_i, sl_o, is, bs) = g(sl_i, sl_o, is, bs, 0) \ \vee \ f(sl_i, sl_o, is, bs) = g(sl_i, sl_o, is, bs, 1) \\
& \textsf{where } \forall s_i, s_o, i \in Act; sl_i, sl_o, is \in Act^{\omega}; b \in Act_{cmos_\dagger}; bs \in Act^{\omega}_{cmos_\dagger}; a \in Act_{cmos_\ddagger}: \\
& \qquad g(s_i \& sl_i, s_o \& sl_o, i \& is, b \& bs, a) = (tl(s_o, b, a, ts(s_i, i)), ts(s_i, i)) \\
& \qquad\qquad \frown g(sl_i, sl_o, is, bs, tl(s_o, b, a, ts(s_i, i))) \\
& \qquad \textsf{where } ts(s_i, i) = \textsf{if } HI.s_i \textsf{ then } i_{cmos} \textsf{ else } Z \textsf{ fi}, \\
& \qquad\qquad tl(s_o, b, a, w) = \textsf{if } b = \dagger \wedge w = 1 \textsf{ then } \dagger_{VDD} \\
& \qquad\qquad\qquad \textsf{elif } b = \dagger \wedge w = 0 \textsf{ then } \dagger_{VSS} \\
& \qquad\qquad\qquad \textsf{elif } b = \dagger \wedge w = Z \textsf{ then } a \\
& \qquad\qquad\qquad \textsf{elif } HI.s_o \textsf{ then } b \textsf{ else } a \ \textsf{ fi} \\
& \qquad\qquad \textsf{where } i_{cmos} = \ \textsf{if } HI.i \textsf{ then } 1 \textsf{ else } 0 \textsf{ fi}
\end{aligned}$$

Abbildung 5.23: Spezifikation eines CMOS-Bustreibers.

Die Ausgabetupel setzen sich aus den Ausgabewerten der beiden Treiber tl und ts zusammen. Der Treiber ts schreibt abhängig von der zugehörigen Selektionsleitung den Eingabewert (umgesetzt mittels i_{cmos} auf CMOS-Werte) oder "Z" auf die Busleitung. Der Treiber tl liest den resultierenden Wert auf dem Bus (b) und überprüft im Falle eines Kurzschlusses (signalisiert durch $b = \dagger$) ob der Treiber selbst aktiv am Buskonflikt beteiligt ist ($w = 1, w = 0$). Ist dies der Fall, so wird eine entsprechende Meldung ($\dagger_{VDD}, \dagger_{VSS}$) ausgegeben – ansonsten wird der zuletzt ausgegebene Wert, der aufgrund der anstehenden Konfliktbehandlung ohnehin unberücksichtigt bleibt, erneut ausgegeben. Wenn auf dem Bus kein Kurzschluß auftritt, so gibt der Treiber tl entsprechend seiner Selektionsleitung den aktuellen Buswert oder den zuletzt ausgegebenen Wert aus. Letzteres modelliert die Kapazität der Ausgabeleitung in Richtung angeschlossener Schaltungskomponenten. An dieser Stelle wollen wir deutlich darauf hinweisen, daß die hier vorgestellte Modellierung nur unter der Annahme einer Zweizyklenverarbeitung adäquat ist, wie sie bereits für NMOS- und technologieunabhängige Busstrukturspezifikationen eingeführt wurde und für CMOS-Busstrukturspezifikationen in Teilabschnitt 5.3.2.6 eingeführt wird. Diese Zweizyklenverarbeitung erlaubt es im ersten Schritt Werte auf den Bus zu schreiben um im zweiten Schritt auf einen möglichen Kurzschluß mit unveränderten Eingabewerten angemessen zu reagieren.

Die nachfolgenden drei Modellierungen fehlerhafter Bustreiber orientieren sich an der fehlerfreien Version, wobei geringe Modifikationen entsprechend der zu modellierenden Fehler einzubringen sind. Abbildung 5.24 zeigt die Spezifikation eines nach (a) zerstörten Bustreibers. Die Funktionen g_{SA1} und g_{SOP} modellieren die Fehlerannahmen stuck-at-1

pred $BT_{\dagger_{VDD}} : Bt_{cmos} \rightarrow Bool$

spec $BT_{\dagger_{VDD}}.f \equiv \forall sl_i, sl_o, is \in Act^{\omega}; bs \in Act^{\omega}_{cmos_{\dagger}}:$

$f(sl_i, sl_o, is, bs) = g(sl_i, sl_o, is, bs, 0) \ \vee \ f(sl_i, sl_o, is, bs) = g(sl_i, sl_o, is, bs, 1)$

where $\forall s_i, s_o, i \in Act; sl_i, sl_o, is \in Act^{\omega}; b \in Act_{cmos_{\dagger}}; bs \in Act^{\omega}_{cmos_{\dagger}}; a \in Act_{cmos_{\ddagger}}:$

$g(s_i \& sl_i, s_o \& sl_o, i \& is, b \& bs, a) =$ if $HI.s_i \ \wedge \ LO.i$ then$(\dagger_{VDS}, Z)$ else
$(tl(s_o, b, a, g_{SA1}(s_i, i)), g_{SA1}(s_i, i))$ fi
$\frown g(sl_i, sl_o, is, bs, tl(s_o, b, a, g_{SA1}(s_i, i))) \ \vee$

$g((s_i \& sl_i, s_o \& sl_o, i \& is, b \& bs, a) = (tl(s_o, b, a, g_{SOP}(s_i, i)), g_{SOP}(s_i, i))$
$\frown g(sl_i, sl_o, is, bs, tl(s_o, b, a, g_{SOP}(s_i, i)))$

where $g_{SA1}(s_i, i) = 1,$
$g_{SOP}(s_i, i) = Z,$
$tl(s_o, b, a, w) =$ if $b = \dagger \wedge w = 1$ then $\dagger_{VDD}$
elif $b = \dagger \wedge w = 0$ then $\dagger_{VDS}$
elif $b = \dagger \wedge w = Z$ then a
elif $HI.s_o$ then b else a fi

Abbildung 5.24: Spezifikation eines nach (a) zerstörten CMOS-Bustreibers.

und stuck-at-open. Die Auswahl, welche Fehlerannahme (SA1,SOP) das Modellverhalten bestimmt, wird nichtdeterministisch durchgeführt. Dieser Nichtdeterminismus beschränkt sich auf das erste Auftreten. Wird die Wahl einmal getroffen, so verhält sich die Modellierung der fehlerhaften Komponente stets gleich. Um einen Kurzschluß innerhalb des Bustreibers festzustellen, wird im Falle der Fehlerannahme SA1 stets überprüft, ob über den noch intakten Transistor eine Null geschrieben wird. Ist dies der Fall, so wird auf diesen so entstandenen lokalen Kurzschluß durch die Ausgabe von "$\dagger_{VDS}$" an die Zeitabstraktion und von "Z" an den Bus reagiert. Letzteres stellt sicher, das zunächst der lokale Kurzschluß "aufgelöst" wird, und erst dann der sich ergebende Wert auf den Bus geschrieben wird. Eine weitere Abweichung von der Spezifikation fehlerfreier Bustreiber betrifft die Zerstörungsmeldung im Konfliktfall, falls der Bustreiber selbst aktiv eine Null auf den Bus schreibt. In dieser Situation beinhaltet die Zerstörungsmeldung eine völlige Zerstörung der Bustreibers ($\dagger_{VDS}$).

$$\textbf{pred}\ BT_{\dagger_{VSS}} : Bt_{cmos} \to Bool$$

$$\begin{aligned}
\textbf{spec}\ BT_{\dagger_{VSS}}.f \equiv\ & \forall sl_i, sl_o, is \in Act^{\omega}; bs \in Act^{\omega}_{cmos_{\dagger}}: \\
& f(sl_i, sl_o, is, bs) = g(sl_i, sl_o, is, bs, 0) \ \vee\ f(sl_i, sl_o, is, bs) = g(sl_i, sl_o, is, bs, 1) \\
& \textsf{where}\ \forall s_i, s_o, i \in Act; sl_i, sl_o, is \in Act^{\omega}; b \in Act_{cmos_{\dagger}}; bs \in Act^{\omega}_{cmos_{\dagger}}; a \in Act_{cmos_{\ddagger}}: \\
& \qquad g(s_i \& sl_i, s_o \& sl_o, i \& is, b \& bs, a) = \textsf{if}\ HI.s_i \ \wedge\ HI.i\ \textsf{then}\ (\dagger_{VDS}, Z)\ \textsf{else} \\
& \qquad\qquad (tl(s_o, b, a, g_{SA0}(s_i, i)), g_{SA0}(s_i, i))\ \textsf{fi} \\
& \qquad\qquad \frown g(sl_i, sl_o, is, bs, tl(s_o, b, a, g_{SA0}(s_i, i)))\ \vee \\
& \qquad g(s_i \& sl_i, s_o \& sl_o, i \& is, b \& bs, a) = (tl(s_o, b, a, g_{SOP}(s_i, i)), g_{SOP}(s_i, i)) \\
& \qquad\qquad \frown g(sl_i, sl_o, is, bs, tl(s_o, b, a, g_{SOP}(s_i, i))) \\
& \qquad \textsf{where}\ \ g_{SA0}(s_i, i) = 0, \\
& \qquad\qquad g_{SOP}(s_i, i) = Z, \\
& \qquad\qquad tl(s_o, b, a, w) = \textsf{if}\ b = \dagger \wedge w = 1\ \textsf{then}\ \dagger_{VDS} \\
& \qquad\qquad\qquad \textsf{elif}\ b = \dagger \wedge w = 0\ \textsf{then}\ \dagger_{VSS} \\
& \qquad\qquad\qquad \textsf{elif}\ b = \dagger \wedge w = Z\ \textsf{then}\ a \\
& \qquad\qquad\qquad \textsf{elif}\ HI.s_o\ \textsf{then}\ b\ \textsf{else}\ a\ \textsf{fi}
\end{aligned}$$

Abbildung 5.25: Spezifikation eines nach (b) zerstörten CMOS-Bustreibers.

Abbildung 5.25 präsentiert die Spezifikation eines nach (b) zerstörten Bustreibes. Die Fehlerannahmen werden durch g_{SA0} und g_{SOP} modelliert. Auch hier ist die Auswahl der Fehlerannahmen nichtdeterministisch im Sinne der Spezifikation von $BT_{\dagger_{VDD}}$. Lokale Zerstörung tritt auf, falls bei einer partiellen Zerstörung SA0 eine Eins geschrieben wird. Die Zerstörungsmeldung beinhaltet im Konfliktfall "völlige Zerstörung", falls der Bustreiber aktiv eine Eins auf den Bus schreibt.

Die Spezifikation eines völlig zerstörten Bustreibers ist in Abbildung 5.26 wiedergegeben. Die Fehlerannahmen umfassen hier SA0, SA1 und SOP, wobei die Auswahl erneut nichtdeterministisch im obigen Sinne erfolgt. Lokale Zerstörung kann hier nicht mehr auftreten. Die Zerstörungsmeldungen im Konfliktfall beinhalten stets $\dagger_{VDS}$.

Zusammenfassend bleibt festzuhalten, daß bei einem Buskonflikt der Bus an alle Bustreiber die Aktion † schickt und jeder einzelne Bustreiber feststellt, ob er aktiv am Buskonflikt beteiligt war. Falls dies der Fall ist, schickt der Bustreiber eine Meldung über eine mögliche Zerstörung an die Zeitabstraktion ($\dagger_{VDD}, \dagger_{VSS}, \dagger_{VDS}$). Die Zeitabstraktion ersetzt dann nichtdeterministisch einen der am Buskonflikt beteiligten Bustreiber durch einen fehlerhaften Bustreiber, also durch einen Bustreiber der entsprechend der Meldung ($\dagger_{VDD}, \dagger_{VSS}, \dagger_{VDS}$) fehlerhaftes Verhalten aufweist. Die Auswahl, welche Fehlerannahme (SA1,SA0,SOP) bezogen auf die jeweilige Zerstörungsmeldung dann zum Tragen kommt, fällt nichtdeterministisch in der Spezifikation des zerstörten Bustreibers.

$$\textbf{pred}\ BT_{\dagger_{VDS}} : Bt_{cmos} \to Bool$$

$$\begin{aligned}
\textbf{spec}\ BT_{\dagger_{VDS}}.f \equiv\ & \forall sl_i, sl_o, is \in Act^{\omega}; bs \in Act^{\omega}_{cmos_\dagger}: \\
& f(sl_i, sl_o, is, bs) = g(sl_i, sl_o, is, bs, 0)\ \vee\ f(sl_i, sl_o, is, bs) = g(sl_i, sl_o, is, bs, 1) \\
& \textsf{where}\ \forall s_i, s_o, i \in Act; sl_i, sl_o, is \in Act^{\omega}; b \in Act_{cmos_\dagger}; bs \in Act^{\omega}_{cmos_\dagger}; a \in Act_{cmos_\ddagger}: \\
& \quad g(s_i \& sl_i, s_o \& sl_o, i \& is, b \& bs, a) = (tl(s_o, b, a, g_{SA0}(s_i, i)), g_{SA0}(s_i, i)) \\
& \qquad \frown g(sl_i, sl_o, is, bs, tl(s_o, b, a, g_{SA0}(s_i, i)))\ \vee \\
& \quad g(s_i \& sl_i, s_o \& sl_o, i \& is, b \& bs, a) = (tl(s_o, b, a, g_{SA1}(s_i, i)), g_{SA1}(s_i, i)) \\
& \qquad \frown g(sl_i, sl_o, is, bs, tl(s_o, b, a, g_{SA1}(s_i, i)))\ \vee \\
& \quad g(s_i \& sl_i, s_o \& sl_o, i \& is, b \& bs, a) = (tl(s_o, b, a, g_{SOP}(s_i, i)), g_{SOP}(s_i, i)) \\
& \qquad \frown g(sl_i, sl_o, , is, bs, tl(s_o, b, a, g_{SOP}(s_i, i))) \\
& \quad \textsf{where}\ g_{SA1}(s_i, i) = 1, \\
& \qquad g_{SA0}(s_i, i) = 0, \\
& \qquad g_{SOP}(s_i, i) = Z, \\
& \qquad tl(s_o, b, a, w) = \textsf{if}\ b = \dagger \wedge w = 1\ \textsf{then}\ \dagger_{VDS} \\
& \qquad\qquad \textsf{elif}\ b = \dagger \wedge w = 0\ \textsf{then}\ \dagger_{VDS} \\
& \qquad\qquad \textsf{elif}\ b = \dagger \wedge w = Z\ \textsf{then}\ a \\
& \qquad\qquad \textsf{elif}\ HI.s_o\ \textsf{then}\ b\ \textsf{else}\ a\ \textsf{fi}
\end{aligned}$$

Abbildung 5.26: Spezifikation eines nach (a) und (b) zerstörten CMOS-Bustreibers.

5.3.2.4 Verbindungsmedium "Bus"

In Anlehnung an die Spezifikation von NMOS-Bussen beschreiben wir in Abbildung 5.27 CMOS-Busse.

$$\textbf{pred}\ BUS_{cmos} : ((Act^{\omega}_{cmos_Z})^n \to Act^{\omega}_{cmos_\dagger}) \to Bool$$

$$\begin{aligned}
\textbf{spec}\ BUS_{cmos}.f \equiv\ & \exists res{:}RES_{cmos}.res\ \wedge\ \forall s_1, \ldots, s_n \in Act^{\omega}_{cmos_Z}: \\
& f(s_1, \ldots, s_n) = g(s_1, \ldots, s_n, 0)\ \vee\ f(s_1, \ldots, s_n) = g(s_1, \ldots, s_n, 1) \\
& \textsf{where}\ \forall i_1, \ldots, i_n \in Act_{cmos_Z}; s_1, \ldots, s_n \in Act^{\omega}_{cmos_Z}; a \in Act_{cmos_\dagger}: \\
& \quad g(i_1\ \&\ s_1, \ldots, i_n\ \&\ s_n, a) = \\
& \quad \textsf{if}\ res(i_1, \ldots, i_n) = Z \\
& \quad \textsf{then}\ a\ \&\ g(s_1, \ldots, s_n, a) \\
& \quad \textsf{else}\ res(i_1, \ldots, i_n)\ \&\ g(s_1, \ldots, s_n, res(i_1, \ldots, i_n))\ \textsf{fi}
\end{aligned}$$

Abbildung 5.27: Spezifikation eines CMOS-Busses.

Im Gegensatz zur NMOS-Variante ist der Typ der Ausgabeelemente um die Aktion † erweitert ($Act_{cmos_\dagger}$). Diese Spezifikation modelliert einen Buskonflikt durch das Senden von † und kann zusammen mit den Spezifikationen der Bustreiber zu einer Spezifikation einer CMOS-Busstruktur erweitert werden.

5.3.2.5 Busstruktur

Auch bei der Modellierung von CMOS-Busstrukturen greifen wir auf die Vorgehensweise zur Erstellung der NMOS-Variante in 5.3.1.5 zurück. Durch schematisches Zusammensetzen der Bestandteile einer CMOS-Busstruktur erhalten wir auch hier eine Gesamtbeschreibung wie sie sich in Abbildung 5.28 darstellt.

$$\textbf{type } Bs_{cmos} = (Bt_{cmos})^n \times (Act^\omega)^{3*n} \rightarrow (Act^\omega_{cmos_\ddagger})^n$$

$$\textbf{pred } BS_{cmos} : Bs_{cmos} \rightarrow Bool$$

$$\begin{aligned}\textbf{spec } BS_{cmos}.f \equiv\ & \exists bus{:}BUS_{cmos}.bus \ \wedge\ \forall bt_1,\ldots bt_n \in Bt_{cmos}; sl_{1,i}, sl_{1,o}, sl_{2,i},\ldots, sl_{n,o}, is_1,\ldots, is_n \in Act^\omega{:} \\ & f(bt_1,\ldots bt_n, sl_{1,i}, sl_{1,o}, sl_{2,i},\ldots, sl_{n,o}, is_1,\ldots, is_n) = \\ & (\ \pi_1.bt_1(sl_{1,i}, sl_{1,o}, is_1, b),\ \ldots,\ \pi_1.bt_n(sl_{n,i}, sl_{n,o}, is_n, b)\ ) \\ & \textsf{where } b = bus(\pi_2.bt_1(sl_{1,i}, sl_{1,o}, is_1, init^1 \circ b),\ldots, \\ & \qquad \pi_2.bt_n(sl_{n,i}, sl_{n,o}, is_n, init^1 \circ b)) \\ & \qquad \textsf{where } (init = 0\ \vee\ init = 1)\end{aligned}$$

Abbildung 5.28: Spezifikation einer CMOS-Busstruktur.

Im Vergleich zur NMOS- bzw. zur technologieunabhängigen Modellierung erfordert die Behandlung von Buskonflikten und deren Folgen Ausgabeelemente vom Typ $Act_{cmos_\ddagger}$. Letzteres gestattet es, neben den üblichen CMOS-Signalpegeln ("1","0") auch Zerstörungsmeldungen nach außen weiterzuleiten.

5.3.2.6 Zeitabstraktion

Die Notwendigkeit für das Einbinden der Busstrukturbeschreibung BS_{cmos} in eine Zeitabstraktion wurde bereits hinreichend diskutiert. Neben der Zweizyklenverarbeitung ergibt sich für diese Spezifikation im Zusammenhang mit CMOS-Busstrukturen, aufgrund der Zerstörung von Bustreibern im Konfliktfall, eine zusätzliche Anforderung. Diese Anforderung betrifft das Austauschen von Beschreibungen fehlerfreier Bustreiber durch Beschreibungen fehlerhafter Bustreiber. Die Schnittstelle der Spezifikation $\overline{BS}_{cmos}$ entspricht der Schnittstelle der physikalischen Komponente in Abbildung 5.8. Abbildung 5.29 veranschaulicht die Spezifikation $\overline{BS}_{cmos}$ für CMOS-Busstrukturen, die sich auf eine Spezifikation FFC abstützt.

$$\textbf{pred}\ \overline{BS}_{cmos} : ((Act^{\omega})^{3*n} \to (Act^{\omega}_{cmos})^{n}) \to Bool$$

$$\begin{aligned}
\textbf{spec}\ \overline{BS}_{cmos}.f \equiv\ & \exists bs{:}BS_{cmos}.bs \wedge \exists g{:}FFC.g \wedge \\
& \exists bt_1,\ldots,bt_n{:}BT_{cmos}.bt_1 \wedge \ldots \wedge BT_{cmos}.bt_n \wedge \\
& \forall sl_{1,i}, sl_{1,o}, sl_{2,i}, \ldots, sl_{n,o}, is_1, \ldots, is_n \in Act^{\omega}: \\
& f(sl_{1,i}, sl_{1,o}, sl_{2,i}, \ldots, sl_{n,o}, is_1, \ldots, is_n) = \\
& \quad g(bs, bt_1, \ldots bt_n, sl_{1,i}, sl_{1,o}, sl_{2,i}, \ldots, sl_{n,o}, is_1, \ldots, is_n)
\end{aligned}$$

$$\textbf{pred}\ FFC : (Bs_{cmos} \times (Bt_{cmos})^{n} \times (Act^{\omega})^{3*n} \to (Act^{\omega}_{cmos})^{n}) \to Bool$$

$$\begin{aligned}
\textbf{spec}\ FFC.f \equiv\ & \exists g{:}FFC.g \wedge \\
& \exists bt_{\dagger_{VDD}}, bt_{\dagger_{VSS}}, bt_{\dagger_{VDS}} : BT_{\dagger_{VDD}}.bt_{\dagger_{VDD}} \wedge BT_{\dagger_{VSS}}.bt_{\dagger_{VSS}} \wedge BT_{\dagger_{VDS}}.bt_{\dagger_{VDS}} \wedge \\
& \forall bs \in Bs_{cmos}; bt_1, \ldots, bt_n \in Bt_{cmos}; sl_{1,i}, sl_{1,o}, sl_{2,i}, \ldots, sl_{n,o}, is_1, \ldots, is_n \in Act^{\omega}: \\
& f(bs, bt_1, \ldots bt_n, sl_{1,i}, sl_{1,o}, sl_{2,i}, \ldots, sl_{n,o}, is_1, \ldots, is_n) = \\
& \quad h(\delta.bs(bt_1, \ldots bt_n, \sigma.sl_{1,i}, \sigma.sl_{1,o}, \sigma.sl_{2,i}, \ldots, \sigma.sl_{n,o}, \sigma.is_1, \ldots, \sigma.is_n), 0) \\
& \textsf{where}\ \ \sigma(a\ \&\ s) = a\ \&\ a\ \&\ \sigma(s), \\
& \qquad \delta(a \frown b \frown s) = b \frown \delta(s), \\
& \qquad \textsf{if}\ \forall l \in \mathbb{N}{:}\ (1 \le l \le n \Rightarrow \pi_l.ft.o \notin \{\dagger_{VDD}, \dagger_{VSS}, \dagger_{VDS}\}) \\
& \qquad \textsf{then}\ h(o,k) = ft.o \frown h\ (rt.o, k+1) \\
& \qquad \textsf{else}\ \exists j \in \mathbb{N}; \tilde{bt_1}, \ldots, \tilde{bt_n} \in Bt_{cmos}: \\
& \qquad\quad 1 \le j \le n \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \wedge \\
& \qquad\quad (\ (\pi_j.ft.o = \dagger_{VDD}\ \wedge\ \tilde{bt_j} = bt_{\dagger_{VDD}})\ \vee \\
& \qquad\quad\ \ (\pi_j.ft.o = \dagger_{VSS}\ \wedge\ \tilde{bt_j} = bt_{\dagger_{VSS}})\ \vee \\
& \qquad\quad\ \ (\pi_j.ft.o = \dagger_{VDS}\ \wedge\ \tilde{bt_j} = bt_{\dagger_{VDS}})\qquad)\ )\ \wedge \\
& \qquad\quad \forall l \in \mathbb{N}{:}\ ((1 \le l \le n\ \wedge\ l \ne j) \Rightarrow \tilde{bt_l} = bt_l)\qquad \wedge \\
& \qquad\quad h(o,k) = g(bs, \tilde{bt_1}, \ldots \tilde{bt_n}, rt^k.sl_{1,i}, rt^k.sl_{1,o}, \\
& \qquad\qquad\qquad rt^k.sl_{2,i}, \ldots, rt^k.sl_{n,o}, rt^k.in_1, \ldots, rt^k.in_n)\ \textsf{fi}
\end{aligned}$$

Abbildung 5.29: Die CMOS-Spezifikation $\overline{BS}_{cmos}$.

Die Spezifikation $\overline{BS}_{cmos}$ erfüllt zwei Aufgaben. Zum einen gewährleistet sie eine schnittstellenkonforme Beschreibung und zum anderen übergibt sie der Funktion g ausschließlich fehlerfreie Bustreiber – dies modelliert die Startsituation in der noch keine Kollisionen aufgetreten sind. Die Spezifikation FFC (Find First Collison) gewährleistet das Austauschen der Bustreiber. Dabei durchsucht eine Hilfsfunktion h sukzessive den erzeugten Ausgabestrom o auf Kollisionen. Tritt im aktuell untersuchten Ausgabetupel $ft.o$ keine Kollision auf (vgl. then-Zweig in Abbildung 5.29), so wird dieses Ausgabetupel ausgegeben und der Rest des Stroms o weiter auf Kollisionen untersucht. Im Konfliktfall (vgl. else-Zweig in Abbildung 5.29) kommt es nun zu einem rekursiven Aufruf einer Funktion g mit veränderten Parametern, wobei g ebenfalls FFC erfüllt. Dabei wird einer der am Buskonflikt aktiv beteiligten Bustreiber nichtdeterministisch ($\exists j$) ausgewählt und

durch einen fehlerhaften Bustreiber entsprechend der Zerstörungsmeldung ersetzt. Außerdem werden die beim Aufruf der Funktion f eingespeisten Ströme derart verkürzt (rt^k), daß nun ein erneutes Aufsetzen an der Kollisionsstelle möglich wird. Der rekursive Aufruf der Funktion g erlaubt in einem weiteren Konfliktfall den zu ersetzenden Bustreiber erneut nichtdeterministisch auszuwählen.

Hier wird nun deutlich, warum wir die Bustreiber als Parameter in die Busstruktur (bs) auf der Mikroebene einbringen – dies erlaubt im Konfliktfall bei CMOS-Busstrukturen ein elegantes Austauschen der Bustreiber. Die Bustreiber genügen je nach Zerstörungsmeldung den Spezifikationen ($BT_{cmos}, BT_{\dagger_{VDD}}, BT_{\dagger_{VSS}}$ und $BT_{\dagger_{VDS}}$). Für den ersten Aufruf der Busstruktur bs wollen wir annehmen, daß alle Bustreiber fehlerfrei sind, also alle der Spezifikation BT_{cmos} genügen (vgl. hierzu die Spezifikation $\overline{BS}_{cmos}$).

Sind mehr als zwei Bustreiber aktiv am Buskonflikt beteiligt, so kann es zur Zerstörung mehrerer Bustreiber in einem "Zyklus" kommen. Angenommen Bustreiber bt_1 und bt_2 schreiben eine Eins und Bustreiber bt_3 schreibt eine Null. Die Zeitabstraktion (d.h. FFC) könnte nun bt_1 als den zerstörten Bustreiber wählen und durch einen Bustreiber mit der Fehlerannahme SOP ersetzen. Dies würde aber den Buskonflikt noch nicht auflösen, da bt_2 und bt_3 nach wie vor unterschiedliche Werte schreiben – eine erneute Auswahl eines zu ersetzenden Bustreibers, also eine Mehrfachzerstörung innerhalb eines Zyklus, wäre erforderlich. Natürlich könnte man sich auch den Fall vorstellen, daß in dem hier gewählten Beispiel bt_1 durch einen Bustreiber mit der Fehlerannahme SA1 ersetzt würde, was ein erneutes Einbeziehen des Bustreibers in den Buskonflikt zur Folge hätte. Da wir allerdings einen fairen nichtdeterministischen Auswahlmechanismus voraussetzen, wird sich jeder Buskonflikt nach endlich vielen "Zyklen" auflösen.

Jeder Buskonflikt bewirkt einen rekursiven Aufruf der Funktion g, die ebenfalls FFC erfüllt, und somit auch einen erneuten Aufruf der Busstruktur bs. Das hat natürlich auch zur Folge, daß die nicht ersetzten Bustreiber und der Bus mit ihren lokalen Initialwerten erneut gestartet werden. Beim Bus spielt diese erneute "Initialisierung" keine Rolle, da ohnehin der Wert auf dem Bus neu berechnet wird. Bei den Initialwerten der Bustreiber ist dies auf den ersten Blick kritischer. Dies liegt daran, daß die Initialwerte im allgemeinen nicht mit den zuletzt geschriebenen Werten übereinstimmen. Zwei Argumente sprechen allerdings für die hier gewählte Variante. Zum einen koppeln sich die Schaltkreiskomponenten die aktuell nicht vom Bus lesen ohnehin von dieser Leitung ab, d.h. das Verhalten der Schaltkreiskomponenten ist unabhängig von den Werten auf dem Bus. Zum anderen ist die Ladung auf einer Leitung (Leitungskapazität) ohnehin nicht beliebig lange zu konservieren – parasitäre Widerstände führen dazu, daß die Ladung abfließt und sich der Signalpegel ändert. Unter diesen Gesichtspunkten ist die hier entwickelte Modellierung adäquat, vorallem wenn man das Hauptaugenmerk der Spezifikationen, die Technologieabhängigkeit, in den Mittelpunkt rückt.

Wie nun die technologieunabhängige Spezifikation $\overline{BS}$ der Busstruktur auf der RT-Ebene mit den NMOS- und CMOS-Varianten auf der Gatterebene in Beziehung zu setzen sind, untersuchen wir im nächsten Abschnitt.

5.4 Verfeinerung

In diesem Abschnitt untersuchen wir die Relation zwischen der technologieunabhängigen Spezifikation von Busstrukturen auf der RT-Ebene und den technologieabhängigen Spezifikationen von Busstrukturen auf der Gatterebene. Als Grundlage dient der in [Bro92a] entwickelte Verfeinerungsbegriff. Wie sich zeigen wird, gilt für Busstrukturen nicht mehr uneingeschränkt, daß deren technologieabhängige Spezifikationen stets Verfeinerungen der technologieunabhängigen Variante darstellen. Abbildung 5.30 veranschaulicht die Verfeinerungsbeziehung zwischen der technologieunabhängigen und den technologieabhängigen Spezifikationen von Busstrukturen.

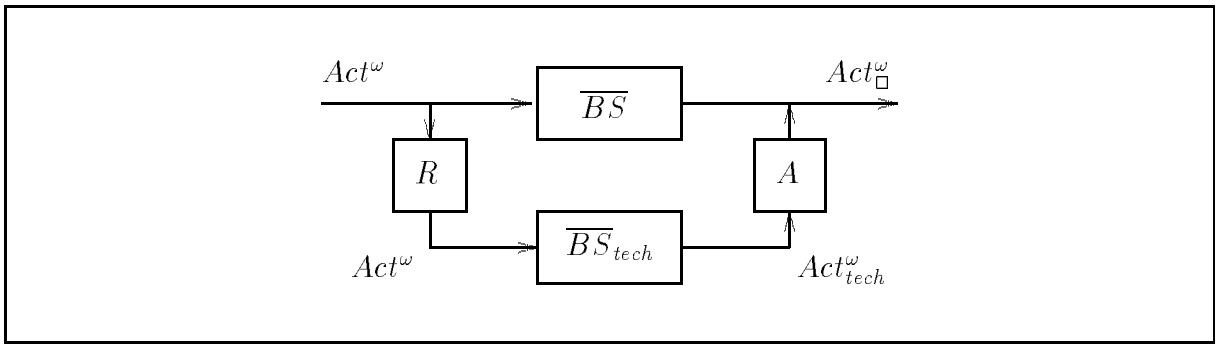


Abbildung 5.30: Verfeinerung von Busstrukturspezifikationen.

Dabei bezeichnet $\overline{BS}$ die Spezifikation einer Busstruktur auf der RT-Ebene und $\overline{BS}_{tech}$ die entsprechende technologieunabhängige Variante, also je nach Bedarf $\overline{BS}_{nmos}$ oder $\overline{BS}_{cmos}$ – die Aktionsmenge Act_{tech} ist dabei analog zu besetzen. Zur weiteren Vereinfachung werden die Eingabeleitungen so wie die Ausgabeleitungen durch eine Datenleitung entsprechenden Typs dargestellt. Die Repräsentations- und Abstraktionsprädikate sind mit R bzw. mit A bezeichnet. Aufbauend auf diesen Vereinfachungen diskutieren wir auf informeller Basis zunächst die Beziehung zwischen der technologieunabhängigen Spezifikation von Busstrukturen $\overline{BS}$ und der NMOS-Busstrukturspezifikation $\overline{BS}_{nmos}$. Anschließend untersuchen wir den Zusammenhang zwischen der technologieunabhängigen Spezifikation von Busstrukturen $\overline{BS}$ und der CMOS-Busstrukturspezifikation $\overline{BS}_{cmos}$.

5.4.1 NMOS-Busstrukturen

Die in dieser Arbeit entwickelte Spezifikation einer NMOS-Busstruktur $\overline{BS}_{nmos}$ ist eine Verfeinerung der zugehörigen technologieunabhängigen Variante. Abbildung 5.31 veranschaulicht die für diese Verfeinerung notwendigen Repräsentations- und Abstraktionsfunktionen, die durch die Prädikate R_{nmos} und A_{nmos} beschrieben sind.

$$\textbf{pred}\ R_{nmos} : (Act^{\omega} \to Act^{\omega}) \to Bool$$

$$\textbf{spec}\ R_{nmos}.f \equiv \forall_{s \in Act^{\omega}}:\ f.s = s$$

$$\textbf{pred}\ A_{nmos} : (Act^{\omega}_{nmos} \to Act^{\omega}_{\Box}) \to Bool$$

$$\textbf{spec}\ A_{nmos}.f \equiv \forall_{s \in Act^{\omega}_{nmos}}: \quad f(0\ \&\ s) = l\ \&\ f.s\ \wedge \quad f(H\ \&\ s) = h\ \&\ f.s$$

Abbildung 5.31: Repräsentations- und Abstraktionsfunktion für NMOS-Busstrukturen.

Die Repräsentationsfunktion entspricht dabei der Identitätsfunktion. Die Abstraktionsfunktion hingegen bildet die starke Null auf den Low-Pegel "l" und die schwache Eins auf den High-Pegel "h" ab. Verfeinerung in unserem Sinne bedeutet nun, daß jedes Verhalten, das $\overline{BS}_{nmos}$ zeigt, auch in $\overline{BS}$ möglich ist. Im konfliktfreien Fall ist dies offensichtlich, da abgesehen von der Signalkonvertierung (vgl. Abstraktionsfunktion) das Verhalten identisch ist. Treten allerdings Konflikte auf, so kann die NMOS-Busstruktur $\overline{BS}_{nmos}$ als eine aufgrund neuer Entwurfsentscheidungen (im Konfliktfall setzt sich der starke Null-Pegel immer durch) entstandene Verfeinerung verstanden werden. Würde man die Konflikte innerhalb von Busstrukturen ausschließen, so ließe sich ein noch stärkerer Verfeinerungsbegriff, wie er in [HT90] eingeführt wird, verwenden.

5.4.2 CMOS-Busstrukturen

Die Spezifikation einer CMOS-Busstruktur $\overline{BS}_{cmos}$ ist keine Verfeinerung der entsprechenden technologieunabhängigen Variante auf der RT-Ebene. Der Grund dafür liegt in dem sich ändernden Verhalten nach einem Buskonflikt. So ist es dann nicht mehr möglich jedes Verhalten, das $\overline{BS}_{cmos}$ zeigt, auch in $\overline{BS}$ zu beobachten. Beispielsweise könnte ein Bustreiber zum Zeitpunkt t so zerstört werden, daß er stets eine starke Eins auf den Bus schreibt. Will man dann zum Zeitpunkt $t+1$ genau über diesen Bustreiber eine Null auf den Bus schreiben, so würde dieser Bustreiber anstelle der Null tatsächliche eine Eins schreiben – also ein von der technologieunabhängigen Variante völlig unterschiedliches Verhalten zeigen.

Beschränkt man allerdings die Eingabesequenzen der CMOS-Busstrukturbeschreibungen auf Sequenzen ohne konfliktverursachenden Zugriffe, so erfahren der Verfeinerungsbegriff, ja sogar auch der stärkere Verfeinerungsbegriff nach [HT90] (vgl. Kapitel 4.4), ihre Gültigkeit. Die für diesen eingeschränkten Anwendungsfall notwendigen Abstraktions- und Repräsentationsfunktionen sind in Abbildung 5.32 aufgeführt.

$$\textbf{pred}\ R_{cmos} : (Act^{\omega} \to Act^{\omega}) \to Bool$$

$$\textbf{spec}\ R_{cmos}.f \equiv \forall s \in Act^{\omega}:\ f.s = s$$

$$\textbf{pred}\ A_{cmos} : (Act^{\omega}_{cmos} \to Act^{\omega}_{\Box}) \to Bool$$

$$\textbf{spec}\ A_{cmos}.f \equiv \forall s \in Act^{\omega}_{cmos}:\ f(0\ \&\ s) = l\ \&\ f.s \wedge f(1\ \&\ s) = h\ \&\ f.s$$

Abbildung 5.32: Repräsentations- und Abstraktionsfunktion für CMOS-Busstrukturen in konfliktfreien Anwendungen.

Abschließend läßt sich zusammenfassen, daß die Spezifikation einer NMOS-Busstruktur uneingeschränkt der Verfeinerungsbeziehung nach [Bro92a] genügt. Bei der Spezifikation einer CMOS-Busstruktur gilt dies nur für konfliktfreie Anwendungen. Im allgemeinen besteht jedoch keine Verfeinerungsbeziehung zur technologieunabhängigen Variante auf der RT-Ebene. Eine technologieunabhängige Beschreibung, wie in Abbildung 5.10 gegeben, ist also in der Tat technologieabhängig, da sich insbesondere keine CMOS-Realisierung aus ihr entwickeln läßt. Dieser Zusammenhang ist im nächsten Kapitel in Abbildung 6.1 nochmals veranschaulicht.

Kapitel 6

Schlußbemerkung

Die vorliegende Arbeit widerlegt die These der Technologieunabhängigkeit von Beschreibungen digitaler Hardware auf der RT-Ebene. Diese Feststellung basiert auf der Untersuchung von Verfeinerungsbeziehungen zwischen adäquaten, technologieunabhängigen RT-Beschreibungen und adäquaten, technologieabhängigen Gatterbeschreibungen von RS-Flipflops und Busstrukturen. Dieses Kapitel soll nun die Ergebnisse dieser Arbeit nochmals kurz zusammenfassen, mit anderen Arbeiten in Beziehung setzen und offene Fragen diskutieren.

6.1 Zusammenfassung und Einordnung der Ergebnisse

Die Zielsetzung dieser Arbeit war

- die Untersuchung der Technologieabhängigkeit von Beschreibungen digitaler Hardware auf der RT-Ebene unter
- Verwendung adäquater Modellierungen (VHDL-Konzepte) sowohl auf der RT-Ebene als auch auf der Gatterebene.

Als Grundlage für die Beantwortung der Frage nach der Technologieabhängigkeit diente ein existierender Verfeinerungsbegriff, der zwei Spezifikationen genau dann in Verbindung bringt, falls die eine als Implementierung oder Realisierung der anderen betrachtet werden kann. Die für die Untersuchung zugrundegelegten Hardwarekomponenten waren das RS-Flipflop als zentraler Baustein sequentieller Hardwarekomponenten und die Busstruktur als wichtiges Verbindungsmedium in Hardwareschaltungen. Die betrachteten Realisierungstechnologien beschränkten sich auf NMOS und CMOS und die zugrundegelegte formale Spezifikationsmethodik war Focus, die im Zuge dieser Arbeit bezüglich der Spezifikation digitaler Hardware erweitert wurde.

Die These der Technologieunabhängigkeit digitaler Hardwarespezifikationen auf der RT-Ebene konnte durch Untersuchungen am RS-Flipflop und an Busstrukturen widerlegt werden. Eine Schlüsselstellung kam in beiden Fällen der Modellierung von Schaltungszerstörung zu. Die Ursache der Schaltungszerstörung lag beim CMOS-RS-Flipflop in der zu hohen Leistungsaufnahme im Schwingungsfall und bei CMOS-Busstrukturen an den Buskonflikten und den damit verbundenen Kurzschlüssen. In beiden Fällen weicht das

Ausgangsverhalten nach der Schaltungszerstörung vom Ausgangsverhalten intakter Komponenten ab. In anderen Worten stellen RS-Flipflop-Beschreibungen und Busstrukturbeschreibungen, beide in CMOS, keine Realisierung der technologieunabhängigen Varianten auf der RT-Ebene dar. Für NMOS-Realisierungen beider Schaltungen gilt die Verfeinerungsbeziehung. Zusammenfassend bleibt festzustellen, daß die Beschreibungen digitaler Hardwarekomponenten auf der RT-Ebene im allgemeinen nicht technologieunabhängig sind.

Welche Konsequenzen lassen sich nun für Spezifikationen von RS-Flipflops und Busstrukturen auf der RT-Ebene in Hinblick auf deren Technologieabhängigkeit ziehen. Im folgenden erörtern wir zwei alternative Beschreibungsparadigmen und wägen deren Vor- und Nachteile ab.

- Die erste Alternative beruht auf dem Ausschluß der irregulären Anwendungen, der irregulären Eingabesequenz bei RS-Flipflops und der Buskonflikte bei Busstrukturen, die letztendlich zur Technologieabhängigkeit führen. Das verwendete Spezifikationsparadigma basiert auf dem "Rely/Guarantee"-Konzept [SDW93, FP94]. Eine Komponentenspezifikation gewährleistet ihr Verhalten nur dann, falls die Eingaben gewissen Anforderungen entsprechen. In unserem Falle arbeiten die Hardwarekomponenten nur dann korrekt, falls die irregulären Applikationen ausgeschlossen sind. Da die Praxis zur Zeit noch weit von vollständig verifizierten, komplexen Hardwaresystemen entfernt ist, erfordert das Einhalten der "Rely"-Bedingung zusätzliche Hardware.
  - \+ Technologieunabhängigkeit auf der RT-Ebene
  - – zusätzliche Hardware
- Die zweite Alternative geht von einer völlig liberalen Verhaltensbeschreibung der Komponenten nach dem ersten Auftreten der irregulären Anwendungen aus. In anderen Worten bleibt das Verhalten von RS-Flipflops nach der irregulären Eingabesequenz und das Verhalten von Busstrukturen nach Konflikten unterspezifiziert – jedes Verhalten ist zulässig. Die liberale Verhaltensweise läßt sich bei differenzierter Betrachtung der irregulären Anwendung natürlich auch weiter einschränken. So könnte man bei Busstrukturen zwischen echten Konflikten, bei denen unterschiedliche Werte geschrieben werden, und solchen Konflikten, bei denen der gleiche Wert von mehreren geschrieben wird, unterscheiden. Nur im Anschluß an echte Konflikte wäre dann jedes Verhalten zulässig. Natürlich sind solche liberalen Spezifikationen technologieunabhängig, da eine Verfeinerungsbeziehung in unserem Sinne zu den technologieabhängigen Beschreibungen besteht. Allerdings läßt die sehr liberale Verhaltensspezifikation nach den irregulären Anwendungen einige Wünsche offen, da keine Aussagen über das weitere Komponentenverhalten, unter Berücksichtigung zerstörter Teile, gemacht werden können.
  - \+ Technologieunabhängigkeit auf der RT-Ebene
  - – zu liberale Verhaltensbeschreibung nach Schaltungszerstörung

Am Beispiel der Busstruktur soll die Beziehung der in dieser Arbeit erstellten Modellierungen zur liberalen Verhaltensbeschreibung anhand von Abbildung 6.1 skizziert werden.

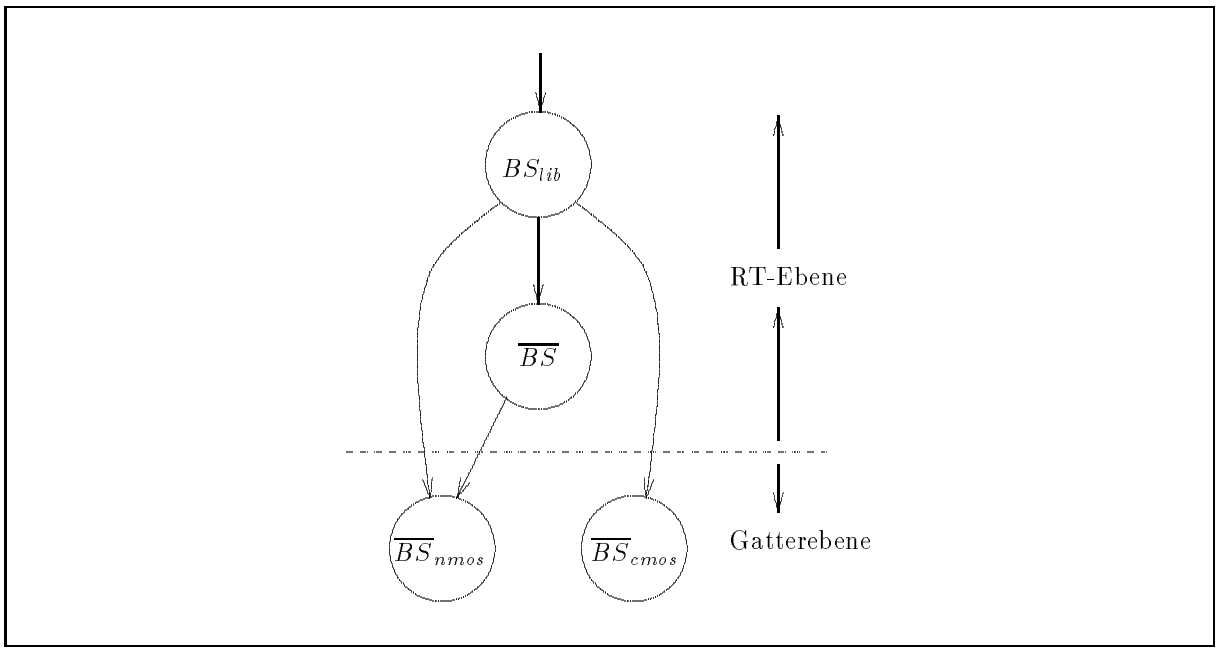


Abbildung 6.1: Verfeinerungsbeziehung zwischen Busstrukturspezifikationen.

Die liberale Verhaltensbeschreibung BS_{lib} stellt eine abstrakte Spezifikation sowohl für die technologieabhängigen Spezifikationen auf der Gatterebene als auch für die im Sinne von [HNS86] technologieunabhängige Spezifikation auf der RT-Ebene dar. Die Spezifikation $\overline{BS}$ kann lediglich in eine NMOS-Spezifikation verfeinert werden, nicht aber in eine CMOS-Spezifikation – die Spezifikation $\overline{BS}$ ist also technologieabhängig. Die hier aufgeführten Spezifikationen können nach oben durch noch liberalere (also solche mit weniger Eigenschaften), technologieunabhängige Spezifikationen erweitert werden. Nach unten läßt sich Abbildung 6.1 durch Hinzunahme von Spezifikationen mit weiteren Realisierungsdetails (also weiteren Eigenschaften) ausbauen. Hier wird deutlich, daß der Aspekt der Technologieunabhängigkeit lediglich eine Frage der erfaßten Eigenschaften in Spezifikationen ist – eine durchaus bekannte Tatsache, die allerdings nicht uneingeschränkt für Spezifikationen der RT-Ebene gilt.

Ein Aspekt, der hinsichtlich der technologieabhängigen Untersuchungen von Spezifikationen auf der RT-Ebene nicht unmittelbar im Zentrum der Untersuchungen steht, ist die Frage nach der Komponierbarkeit der erstellten Spezifikationen. Dabei stellt sich zunächst die Frage nach der Typverträglichkeit der Ströme zu komponierender Beschreibungen. In Anlehnung an [Rug91] wurden die Spezifikationen so erstellt, daß sie eingangsseitig alle technologiespezifischen Treiberstärken verarbeiten können. Weiter gilt sowohl für die RS-Flipflop-Spezifikationen als auch für die Busstrukturspezifikationen, daß Aktionen wie "Z" oder "†" lokal sind und somit die Ausgabeströme lediglich technologiespezifische Aktionen beinhalten. Neben der Typverträglichkeit der zu komponierenden Spezifikationen bedarf es im Falle der Komposition von RS-Flipflop-Spezifikationen im Schwingungsfall besonderer Sorgfalt. Im Zentrum der Überlegungen steht hier die Idee mögliche Schwingungen lokal bezüglich einer Taktperiode zu halten. Diese Idee ("race free environment")

spiegelt sich auch in den Entwurfsregeln für getaktete Schaltungen unter Berücksichtigung von Testaspekten in [Wil86] wieder. Die getakteten Komponenten umfassen pegel- und flankengesteuerte Flipflops sowie das Master-Slave Flipflop, wobei jedes der Flipflops das RS-Flipflop als Grundbaustein beinhaltet ([Per86]). Für unsere Betrachtungen ergeben sich in Anlehnung an [Wil86] folgende zwei Anforderungen für die zu modellierenden Hardwareschaltungen:

1) Zwei aufeinanderfolgende, pegelgesteuerte Flipflops müssen verschiedene Clocksignale besitzen, die ein gleichzeitiges Aktivieren beider Flipflops nicht ermöglichen. Dies bedeutet insbesondere, daß sich die aktiven Pegel der Clocksignale nicht überlappen dürfen.

2) Rückkopplungsschleifen in denen ausschließlich pegelgesteuerte Flipflops enthalten sind, müssen mindestens zwei dieser Flipflops enthalten. Für Rückkopplungsschleifen gilt somit insbesondere stets 1).

Diese Einschränkungen gelten ausschließlich für pegelgesteuerte Flipflops, da sich nur bei diesem Typ eine Schwingung über mehrere, hintereinandergeschaltete, aktivierte, pegelgesteuerte Flipflops fortpflanzen kann. Dies wird nun allerdings aufgrund von Forderung 1) unterbunden. Flankengesteuerte Flipflops sind unter Berücksichtigung von setup- und hold-Zeiten nur zu einem Zeitpunkt sensibel gegenüber Eingabedaten und genügen somit auf natürliche Weise der Forderung, Schwingung lokal zu halten. Das Master-Slave Flipflop besteht im wesentlichen aus zwei aufeinanderfolgenden, pegelgesteuerten Flipflops die durch unterschiedliche Pegel des selben Clocksignals aktiviert werden und somit Forderung 1) stets erfüllen. Zusammenfassend bleibt festzustellen, daß unter Berücksichtigung obiger Forderungen die in dieser Arbeit erstellten Spezifikationen beliebig komponiert werden können.

An dieser Stelle wollen wir das Ergebnis dieser Arbeit mit anderen Arbeiten in Verbindung bringen. Bezüglich der RS-Flipflop-Spezifikation bleibt in [TT93, Car82] die Technologieabhängigkeit der erstellten Spezifikationen unbeachtet. In beiden Ansätzen wird die irreguläre Eingabesequenz nicht ausgeschlossen, deren Konsequenzen für unterschiedliche Realisierungstechnologien bleiben jedoch unberücksichtigt. In [Del87] wird die irreguläre Eingabesequenz ausgeschlossen. Dies kommt der ersten Alternative zum Umgang mit technologieabhängigen Spezifikationen in diesem Abschnitt sehr nahe. Hier sehen wir diese Arbeit als Argumentationshilfe. Wird der Ausschluß der irregulären Eingabesequenz in [Del87] noch durch nicht vorhersagbare Ausgabewerte begründet, so würde eine Argumentation in unserem Sinne auf der Technologieunabhängigkeit der erstellten Spezifikationen basieren. Ganz im Sinne des Ausschlusses irregulärer Anwendungen sind die Spezifikationen von Busstrukturen in [Hoo94, Hoo92, Her92, HFFM92]. Auch hier leistet unsere Arbeit die notwendige Rechtfertigung, ausschließlich technologieunabhängige Beschreibungen auf abstrakter Spezifikationsebene erstellen zu wollen.

6.2 Offene Fragen

In dieser Arbeit beschränken sich die Untersuchungen zur Technologieabhängigkeit von Spezifikationen auf der RT-Ebene ausschließlich auf RS-Flipflops und Busstrukturen. Eine naheliegende Fragestellung zielt auf eine Klassifizierung der Beschreibungen aller digitaler Hardwarekomponenten auf der RT-Ebene in technologieunabhängige und technologieabhängige Beschreibungen ab. Ein weiterer offener Punkt in dieser Arbeit ist der Umgang mit dem Wissen der Technologieabhängigkeit von Spezifikationen auf der RT-Ebene. Zwar wurden in Abschnitt 6.1 bereits zwei Alternativen aufgezeigt, diese aber noch nicht detailliert untersucht und verglichen. So bleibt die Frage nach einem geeigneten Spezifikationsparadigma unter Bewahrung der Technologieunabhängigkeit auf der RT-Ebene offen. Wie bereits erwähnt stellt diese Arbeit eine Erweiterung von Focus hinsichtlich der Beschreibung digitaler Hardware dar. Die Eignung von Focus für den Entwurf verteilter Softwaresysteme wurde bereits unter Beweis gestellt [BDD+92b]. Inwieweit Focus für den gemeinsamen Entwurf von Hardware und Software, also für Hardware/Software Codesign, geeignet ist, bedarf noch genauerer Untersuchungen. Ein interessantes Anwendungsgebiet ist dabei sicherlich der Entwurf mikroprogrammgesteuerter Prozessoren.

Zum Abschluß wollen wir zu einigen, wesentlichen Entscheidungen in dieser Arbeit, sofern ohnehin nicht schon geschehen, Stellung nehmen. Die technologieunabhängigen Beschreibungen des RS-Flipflops auf der RT-Ebene berücksichtigen die Verzögerungszeit nur der Quantität nach. Eine adäquate, qualitative Erfassung von Zeit erfordert einen Wechsel der Abstraktionsebene auf die Gatterebene. Um dennoch den Charakter einer technologieunabhängigen Beschreibung im Sinne der Beschreibung auf der RT-Ebene zu bewahren, erstellen wir, unter Verwendung der Modellierungskonzepte von VHDL, eine modulo der Verzögerungszeiten verhaltensäquivalente, technologieunabhängige Beschreibung auf der Gatterebene. Diese dient als Grundlage weiterer Untersuchungen bezüglich der Technologieabhängigkeit. Die diskutierten Busstrukturen beschränkten sich ausschließlich auf eine gemeinsame Datenleitung, also auf eine Verarbeitungsbreite von Eins. Der Grund dafür lag darin, das Hauptaugenmerk auf die Entwicklung technologieunabhängiger und technologieabhängiger Spezifikationen von Busstrukturen zu legen und nicht auf die Spezifikation einer Busstruktur mit mehr als einer Datenleitung. An dieser Stelle wollen wir jedoch eine mögliche Lösung skizzieren. Um eine Busstruktur mit mehr als einer Datenleitung zu spezifizieren, genügt es den Parallelkompositionsoperator in geeigneter Weise zu verwenden. Dabei läßt sich die gewünschte Verarbeitungsbreite durch eine entsprechende Anzahl parallel geschalteter Beschreibungen von Busstrukturen modellieren. Was die Beweisverpflichtungen und die damit verbundenen formalen Beweise in dieser Arbeit betrifft, so werden beide Aspekte nur informell abgehandelt. Natürlich sehen wir, nicht zuletzt wegen der Verwendung einer formalen Entwurfsmethodik, dies als einen Schwachpunkt dieser Arbeit. Allerdings berufen wir uns auf offensichtlich erkennbare Diskrepanzen, z.B. bezüglich dem Verhalten von technologieunabhängigen Spezifikationen von Busstrukturen und den zugehörigen CMOS-Varianten, und verzichten auf formale Beweise. Eine weitere grundlegende Entscheidung bei der Modellierung von Hardwarekomponenten betrifft die Annahme diskreter Schaltungsaufbauten. Diese Annahme erlaubt es beispielsweise, die Auswirkungen thermischer Zerstörung lokal zu beschreiben, ohne auf unmittelbar topologisch benachbarte Schaltungsteile eingehen zu müssen. Die

Behandlung integrierter Bausteine ist so nicht möglich, da dort thermische Zerstörungen nicht mehr lokal sind, sondern Auswirkungen auf alle Schaltungsteile auf dem selben Chip besitzen. Für die prinzipielle Untersuchung der Technologieabhängigkeit von Hardwarebeschreibungen auf der RT-Ebene reichen die Beschreibungen diskreter Schaltungen jedoch aus.

Literaturverzeichnis

[BD92] M. Broy and C. Dendorfer. Modelling of Operating System Structures by Timed Stream Processing Functions. *Journal of Functional Programming*, 2(1):1–21, 1992. Also appeared as SFB-Bericht 342/22/90 A.

[BDD+92a] M. Broy, F. Dederichs, C. Dendorfer, M. Fuchs, T. F. Gritzner, and R. Weber. The Design of Distributed Systems — An Introduction to FOCUS. SFB-Bericht 342/2/92 A, Technische Universität München, January 1992.

[BDD+92b] M. Broy, F. Dederichs, C. Dendorfer, M. Fuchs, T. F. Gritzner, and R. Weber. Summary of Case Studies in FOCUS — A Design Method for Distributed Systems. SFB-Bericht 342/3/92 A, Technische Universität München, January 1992.

[BPS92] D. Borrione, L. Pierre, and A. Salem. Formal Verification of VHDL Descriptions in the Prevail Environment. *IEEE Design & Test of Computers*, 1992.

[Bro89] M. Broy. Towards a Design Methodology for Distributed Systems. In M. Broy, editor, *Constructive Methods in Computing Science*, volume 55 of *NATO ASI Series F: Computer and System Sciences*, pages 311–364. Springer, 1989.

[Bro90] M. Broy. Functional Specification of Time Sensitive Communicating Systems. Technical report, International Summer School, Marktoberdorf, Germany, July 1990.

[Bro92a] M. Broy. Compositional Refinement of Interactive Systems. Technical Report 89, Digital Systems Research Center, Palo Alto, California 94301, July 1992.

[Bro92b] M. Broy. (Inter)-Action Refinement: The Easy Way – Compositional Refinement of Interactive Systems. Technical Report 10, International Summer School, Marktoberdorf, Germany, July 1992.

[Bro93] M. Broy. *Informatik – Eine grundlegende Einführung Teil II.* Springer, 1993.

[BW81] F.L. Bauer and H. Wössner. *Algorithmische Sprache und Programmentwicklung.* Springer, 1981.

[Car82] L. Cardelli. *An Algebraic Approach to Hardware Description and Verification.* PhD thesis, University of Edinburgh, Department of Computer Science, April 1982.

[Car93] M. Carroll. VHDL — Panacea or Hype? *IEEE SPECTRUM*, June 1993.

[CM88] K. Chandy and J. Misra. *Parallel Program Design.* Addison-Wesley, 1988.

[Coh88] A. Cohn. A Proof of Correctness of the Viper Microprocessor: The First Level. In G. Birtwistle and P. Subrahmanyam, editors, *VLSI Specification, Verification and Synthesis.* Kluwer Academic Publishers, 1988.

[Dav88] B. Davie. *A Formal, Hierarchical Design and Validation Methodology for VLSI.* PhD thesis, University of Edinburgh, Department of Computer Science, October 1988.

[Ded92] F. Dederichs. Transformation verteilter Systeme: Von applikativen zu prozeduralen Darstellungen. SFB-Bericht 342/17/92 A, Technische Universität München, August 1992. Doktorarbeit.

[Del87] C. Delgado Kloos. *Semantics of Digital Circuits.* Springer, 1987. LNCS 285.

[Del91] C. Delgado Kloos. Fixpoints and Flipflops. In M. Broy, editor, *Informatik und Mathematik*, pages 73–85. Springer, 1991.

[EEE87] The Institute of Electrical and Electronics Engineers. *IEEE Standard VHDL Language Reference Manual.* IEEE, 1987.

[EEE88] The Institute of Electrical and Electronics Engineers. *IEEE Standard Backplane Bus Specification for Multiprocessor Architectures: Futurebus.* IEEE, 1988.

[EEE92] The Institute of Electrical and Electronics Engineers. *IEEE Standard VHDL Language Reference Manual.* IEEE, 1992.

[Eve91] H. Eveking. *Verifikation digitaler Systeme.* Teubner, 1991.

[FFH90] M. Fourman, M. Francis, and R. Harris. Formal System Design – Interactive Synthesis based on Computer-Assisted Formal Reasoning. *Formal VLSI Specification and Synthesis – VLSI Design Methods I*, 1990.

[FP94] M. Fuchs and J. Philipps. Formal Development of a Production Cell in Focus – A Case Study. In C. Lewerenz and T. Lindner, editors, *Case Study Production Cell*, chapter XII, pages 191 – 200. Forschungszentrum Informatik – Universität Karlsruhe, 1994. to appear also in LNCS 891.

[Fuc92] M. Fuchs. Functional Modeling of Clocked Hardware Circuits. Technical Report TUM-I9211, Technische Universität München, April 1992.

[GCV80] J. Galiay, Y. Crouzet, and M. Vergniault. Physical versus Logical Fault Models MOS LSI Circuits: Impact on their Testability. *IEEE Transactions on Computers*, C-29(6), June 1980.

[Gor89] M. Gordon. *The HOL-System: Tutorial, Description, Reference Manual.* Cambridge Research Center, 1989.

[Gro92] IEEE Model Standard Group. Std_logic_1164 Muti-Value Logic System. Public Domain Software, February 1992.

[Har87] D. Harel. Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming*, 8, 1987.

[Her92] J. Herbert. Incremental Design and Formal Verification of Microcoded Microprocessors. In V. Stavridou, T. Melham, and R. Boute, editors, *Theorem Provers in Circuit Design*, IFIP. Elsevier Science Publishers B.V. (North-Holland), 1992.

[HFFM92] R. Hughes, M. Francis, S. Finn, and G. Musgrave. Formal Tools for Tri-State Design in Busses. In L. Claesen, editor, *HOL Workshop*, June 1992.

[HNS86] E. Hörbst, N. Nett, and H. Schwärtzel. *VENUS Entwurf von VLSI-Schaltungen.* Springer, 1986.

[Hoa85] C. Hoare. *Communicating Sequential Processes.* Prentice Hall, 1985.

[Hoo92] J. Hooman. A Compositional Method for the Top-Down Design of Real-Time Systems. In *Proceedings 4th Euromicro Workshop on Real-Time Systems*, pages 86–91. IEEE, 1992.

[Hoo94] J. Hooman. Compositional Verification of a Distributed Real-Time Arbitration Protocol. *Real-Time Systems*, 6:173–205, 1994.

[HT90] N. Harman and J. Tucker. The Formal Specification of a Digital Correlator. In K. McEvoy and J. Tucker, editors, *Theoretical Foundations of VLSI Design*, chapter 5, pages 161 – 262. Cambridge University Press, 1990.

[Inc91] Synopsys Inc. Synopsys VHDL Standard Logic Library. Public Domain, 1991.

[Kah74] G. Kahn. The Semantics of a Simple Language for Parallel Programming. In J. Rosenfeld, editor, *Information Processing 74*, pages 471–475. North-Holland, 1974.

[KHK93] P. Kanthamanon, G. Hellestrand, and M. Kam. VHDL vs. Functional Hardware Description: A Comparison and Critique. Technical Report 9306, University of New South Wales, April 1993.

[Koz83] D. Kozen. Results on the Propositional MY-Calculus. *TCS*, 27, 1983.

[Krö87] F. Kröger. *Temporal Logic of Programs.* Springer, 1987.

[Lam88] L. Lamport. A simple Approach to Specifying Concurrent Systems. *Communications of the ACM*, 32(1), 1988.

[LSU90] R. Lipsett, C. Schaefer, and C. Ussery. *VHDL: Hardware Description and Design.* Kluwer Academic Publishers, 1990.

[Man79] M. Mano. *Digital and Computer Design.* Prentice-Hall, Inc., Englewood Cliffs, N.J. 07632, 1979.

[Mil80] R. Milner. A Calculus of Communicating Systems. In *LNCS 92.* Springer, 1980.

[Mil85] G. Milne. Circal and the Representation of Communication, Concurrency and Time. *ACM Trans. on Programming Languages and Systems*, 7(2), 1985.

[MT90] K. McEvoy and J. Tucker. Theoretical Foundations of Hardware Design. In K. McEvoy and J. Tucker, editors, *Theoretical Foundations of VLSI Design*, volume 10 of *Cambridge Tracts in Theoretical Computer Science*, chapter 1, pages 1 – 62. Cambridge University Press, 1990.

[NCI75] H. Nagle, B. Carroll, and J. Irwin. *An Introduction to Computer Logic.* Prentice-Hall, 1975.

[Per86] P. Pernards. *Digitaltechnik.* Hüthig Verlag, 1986.

[Rei85] W. Reisig. *Petri Nets – An Introduction.* Number 4 in EATCS Monograph. Springer, 1985.

[Rug91] I. Ruge. Grundlagen der integrierten Schaltungen. Vorlesungsskript an der TU-München, 1991.

[San88] D. Sannella. A Survey of Formal Software Development Methods. Technical Report 8856, University of Edinburgh, Department of Computer Science, July 1988.

[SB89] D. Schilling and C. Belove. *Electronic Circuits.* McGraw-Hill. Inc., 1989.

[SB90] C. Seger and R. Bryant. Modeling of Circuit Delays in Symbolic Simulation. In L. Claesen, editor, *Formal VLSI Correctness Verification.* Elsevier Science Publishers B. V. (North-Holland), 1990.

[SDB93] L. Sanches, C. Delgado Kloos, and P. Breuer. Stream Semantics for VHDL: An Example. *Workshop on Design Methodologies for Microelectronics and Signalprocessing*, October 1993.

[SDW93] K. Stølen, F. Dederichs, and R. Weber. Assumption/Commitment Rules for Networks of Asynchronously Communicating Agents. SFB-Bericht 342/2/93 A, Technische Universität München, February 1993.

[Sei90] M. Seifert. *Digitale Schaltungen.* Verlag Technik GmbH Berlin, 1990.

[Tas92] J.Van Tassel. A Formalisation of the VHDL Simulation Cycle. Technical Report 249, University of Cambridge, 1992.

[TT89] B. Thompson and J. Tucker. Synchronous Concurrent Algorithms. Technical report, Centre for Theoretical Computer Science, University of Leeds, 1989.

[TT93] B. Thompson and J. Tucker. Equational Specification of Synchronous Concurrent Algorithms and Architektures. In *Working Material for the Lectures of J. Tucker.* NATO Science Committee, 1993.

[Tuc91] J. Tucker. Equational Specification of Synchronous Concurrent Algorithms and Architectures. Technical report, International Summer School, Marktoberdorf, Germany, July 1991.

[Wad78] R. Wadsack. Fault Modeling and Logic Simulation of CMOS and NMOS Integrated Circuits. *The Bell System Technical Journal*, 57(5), 1978.

[WE85] N. Weste and K. Eshraghian. *Principles of CMOS VLSI Design - A Systems Perspective.* Addison-Wesley Publishing Company, October 1985.

[Web92] R. Weber. Eine Methodik für die formale Anforderungsspezifikation verteilter Systeme. SFB-Bericht 342/14/92 A, Technische Universität München, März 1992. Doktorarbeit.

[Wil86] T. Williams. *VLSI Testing.* North-Holland, 1986.