Der testfreundliche Entwurf asynchroner Schaltungen

Dem Fachbereich Elektrotechnik und Informationstechnik
der Universität Hannover
zur Erlangung des akademischen Grades
Doktor-Ingenieur
genehmigte Dissertation
von

Dipl.-Ing. Volker Schöber

geboren am 9. Januar 1963 in Hannover

2001

Der testfreundliche Entwurf asynchroner Schaltungen

Dem Fachbereich Elektrotechnik und Informationstechnik
der Universität Hannover
zur Erlangung des akademischen Grades
Doktor-Ingenieur
genehmigte Dissertation
von

Dipl.-Ing. Volker Schöber

geboren am 9. Januar 1963 in Hannover

2001

1. Referent:	Prof. Dr.-Ing. J. Mucha
2. Referent:	Prof. Dr.-Ing. P. Pirsch
3. Referent:	Prof. Dr.-Ing. K. Jobmann

Tag der Promotion: 22. Juni 2000

Für die Unterstützung bei der Durchführung der Arbeit möchte ich mich bei Herrn Prof. Dr.-Ing J. Mucha recht herzlich bedanken. Ferner danke ich Herrn Prof. Dr.-Ing. P. Pirsch für die Übernahme des Korreferates sowie Prof. Dr.-Ing. K. Jobmann für den Vorsitz bei der Prüfung.

Kurzfassung

Die Anforderungen an den Entwurf integrierter Schaltungen zeigen durch die technologischen Innovationen in der Halbleiterproduktion, dass klassische Verfahren zunehmend an ihre Grenzen stoßen. Insbesondere die Dimensionierung der Taktversorgung integrierter Schaltungen entwickelt sich zu einem stetig wachsenden Problem. Eine globale Taktversorgung hingegen wird bei asynchronen Schaltungen nicht benötigt. Hierdurch bieten asynchrone Schaltungen eine attraktive Alternative, um das Entwickeln von *Systems on Silicon* mit digitalen und *Mixed-Signal* Modulen zu beherrschen.

Zum Entwurf asynchroner Schaltungen sind in den letzten Jahren Methoden zur Beschreibung von Prozessen vorgestellt worden, die von einer lokalen Steuerung ausgehen. Somit ist keine strenge Trennung nach Schaltnetz und Rückkopplungsnetzwerk mehr nötig, wie sie von Moore und Mealy vorgeschlagen wurde. Der Entwickler kann bei diesen neuen Schaltungskonzepten die Granularität der lokalen Steuerung festlegen. Die Größe kann dabei von kleinen Interfaceblöcken bis hin zu Mikroprozessoren oder auch Systemen variieren. Darüber hinaus bieten diese Konzepte neben der Synthese auch die Möglichkeit der Schaltungsverifikation. Asynchrone Schaltungen, welche die *Bundled Data Convention* verwenden, haben eine hohe Leistungsfähigkeit bewiesen. Auf diese Art der Schaltungen, die auch als *Self-timed Circuits* bezeichnet werden, stützt sich diese Untersuchung zum testfreundlichen Entwurf asynchroner Schaltungen bei der Verwendung von *Single-Rail Signalling*.

Die Untersuchung beginnt mit einem Rückblick auf bereits bestehende Arbeiten des Entwurfs und Tests asynchroner Schaltungen. Auf diesen Ergebnissen aufbauend, werden neue asynchrone Testverfahren vorgeschlagen. Das Problem, dass klassische asynchrone Testmethoden auf einer getakteten Testablaufsteuerung basieren, wird hierbei gelöst. Zur Synchronisation der Testdaten wird bei den vorgestellten Verfahren ein asynchroner Ansatz verfolgt, der die *Bundled Data Convention* als Synchronisation berücksichtigt. Es wird damit ermöglicht, Testverfahren für asynchrone Schaltungen zu entwerfen, die auch sicherheitskritischen Anwendungen in Modulen ohne Taktsteuerung genügen. Bei der Entwicklung der Testkonzepte für asynchrone Schaltungen wird der Einfluß der Testlogik auf die Performance des Datenpfads der Schaltung minimiert.

Es werden zu einer asynchronen Schaltungsbibliothek neue Standardzellen vorgeschlagen, um die asynchronen testfreundlichen Entwurfsverfahren zu realisieren. Die scanfähigen Register sind dabei so aufgebaut, dass keine zusätzliche Signalverzögerung im Datenpfad entsteht. Diese hier vorgestellte Idee der Einkopplung der Testdaten kann auch in der synchronen Schaltungstechnik eingesetzt werden und somit den oft genannten Nachteil des Performanceverlusts bei einem testfreundlichen Entwurf lösen. Darüber hinaus wird ein modifiziertes *Muller C-Element* vorgestellt, welches die Performanceverluste minimiert, die bei der Umschaltung zwischen Test- und Funktionsbetrieb entstehen. Schließlich wird ein Konzept zur Fehlermodellierung für klassische und nichtklassische Fehlermodelle auf Gatterebene präsentiert und auf asynchrone Schaltungselemente angewendet.

Mit Hilfe der neuen Standardzellen wird ein Konzept für passive Testhilfen beschrieben, das einem Scanpfad für synchrone Schaltungen entspricht. Es bietet eine Lösung, um extern generierte Testmuster an eingebetteten Modulen steuern und beobachten zu können. Darüber hinaus werden aktive Teststrukturen vorgestellt, um in der Schaltung Testmuster generieren und Testdaten komprimieren zu können und so die Funktionalität des synchronen BILBO Testkonzepts zu realisieren. Dieses Verfahren ermöglicht erstmalig, aufbauend auf einer asynchronen Ablaufsteuerung, Built-In Self-Test Verfahren für Schaltungen zu entwerfen, die keine Taktversorgung benötigen.

Abstract

Due to technology driven improvements of the semiconductor industry traditional design concepts will be the limiting factor of the design in deep submicron technologies. In particular, the design of clock networks will become one of the major bottlenecks. In contradiction to synchronous design, asynchronous circuits do not need a global clock distribution. They are build up with local synchronized control paths. This leads to an attractive alternative to design systems on silicon containing digital and mixed-signal modules.

The traditional circuit design theory uses large combinational network together with a feedback loop, like it is proposed from Mealy and Moore in the fifties. Methods have been presented in the last two decades for designing asynchronous circuits using locally synchronized control signal distribution. This enables design styles where the granularity of modules with its control signals can be individually specified. Asynchronous circuit designs that are using the bundled data convention have shown the competitiveness to their synchronous counterparts. This circuit design style - also named as self-timed circuits - is targeted here for designing testable and self-testable asynchronous circuits.

This thesis starts with an overview on designing and testing asynchronous circuits. Traditional asynchronous test concepts still need a clock for test purposes. Based on this, two new asynchronous test concepts are presented. They propose a solution by using the bundled data convention to synchronize the data also in test mode. An asynchronous test can now be developed for modules which do not have any global clock synchronization. This allows to design testable mixed-signal interfaces of analogue modules using handshake signals.

Asynchronous standard cells are proposed to enhance the design for testability. The proposals for the control and data path elements complete asynchronous standard cell libraries. The scan registers are designed without additional delay of the test multiplexer in the data path. The idea of coupling the data path and test path can also be adopted to synchronous designs. In principle, it solves the drawback of performance reduction in integrated circuits when scan registers are involved. In addition, the design for testability of a Muller C-Element is presented with minimal additional delay for the control path. For an asynchronous library a fault modeling concept is developed. The results show that single stuck-at faults at the primary ports of the cell are not sufficient to all possible faults that can occur at transistor level.

A test concept is presented that can be compared to a synchronous scan path. This solves the problem of controlling and observing external test pattern to asynchronous modules without using a clock distribution network. The asynchronous data and control path can now be tested in parallel. Second, a test concept is proposed for generating test pattern and compress test data on chip, like it is done with the synchronous BILBO concept. With this test concept it is now possible to detect even dynamic failures of the bundled data convention in asynchronous designs.

Schlagwörter

Testfreundlicher Entwurf, Asynchrone Schaltungen, Selbsttest, Scanpfad, Testmustergenerator, Testdatenkompression.

Keywords

Design for Testability, Self-timed Circuits, Built-In Self-Test, Scan Design, BIST, LFSR, MISR.

Inhalt

Einige Fachbegriffe

Bounded Delay	Bezeichnet die Annahme, dass die Verzögerungen der Elemente und deren Verbindungen einer Schaltung beschränkt sind.
Bundled Data Convention	Asynchrones Synchronisationsverfahren, bei dem zwei Steuerleitungen das Taktsystem einer *Pipeline* Architektur ersetzen **[80MeadC]**$_{L.2}$.
Burst Mode	Bezeichnet eine asynchrone Schaltungsmethode. Es können mehrere Ein- und Ausgangssignale sich gleichzeitig verändern **[91NowiD]**$_{L.2}$. Dies ist eine deutliche Erweiterung zum *Fundamental Mode*.
Change Diagram	Eine Beschreibungssprache von asynchronen Prozessen mit ähnlichen Eigenschaften, wie sie bei den *Signal Transition Graph* Prozessen zu finden sind **[92KishKT]**$_{L.2}$.
Communication Process	Bezeichnet eine prozedurale Sprache für asynchrone Schaltungen, die aus Basiselementen aufgebaut ist **[89Mart]**$_{L.2}$.
Deadlocks	Bezeichnet das Verhalten einer Schaltung, die einen ungewollten stabilen Zustand besitzt, der nicht mehr verlassen wird.
Decomposition by Net Contraction	Verfahren zur Partitionierung von *Signal Transition Graphs* in Untergraphen zur Abbildung der Prozesse auf primitive Elemente während der Synthese **[87Chua]**$_{L.2}$.
Delay-insensitive	Eine Schaltungseigenschaft, die erfüllt ist, wenn beliebige Verzögerungen der Elemente und Verbindungsleitungen kein Fehlverhalten der Schaltung verursachen.
Dual-Rail (Double-Rail)	Eine Datenübertragungsart, die zwei Signalleitungen zur Kodierung des Datums verwendet. Dies wird bei *Precharge* Techniken häufig eingesetzt. Der Ausgang ist dann gültig, wenn beide Signale gegensätzliche Werte annehmen.

Fail-stop	Diese Bezeichnung wird für das Verhalten einer Schaltung verwendet, welche nach der Sensibilisierung eines Fehlers im Fehlerzustand verharrt. Schaltungen dieser Art verharren, nachdem ein Haftfehler eingestellt wurde. Die Beobachtung erfolgt über das Verharren der Schaltung.
Four-Phase Signalling	Verfahren, welches zwei Module durch eine Pegelsteuerung synchronisiert (auch *Four-Cycle Signalling* oder *Equipotentional Signalling*) **[80MeadC]**$_{L.2}$. Der Zustand des Signals und nicht der Wechsel ist somit entscheidend.
Free Choice (Input Choice)	Bezeichnet das Verhalten eines Signalwechsel, der von Signalwechseln unterschiedlicher Prozesse ausgelöst werden kann **[87Chua]**$_{L.2}$. Dieses Verhalten trifft man z.B. bei einem XOR an.
Fundamental Mode	Bezeichnung für eine Schaltungsart, in der sich Signale des Eingangs nur dann ändern, wenn alle internen Zustände stabil sind.
Handshaking	Bezeichnung für die Synchronisation zwischen Sender und Empfänger mit Hilfe von *Acknowledge* und *Request* Signalleitungen.
Hazard	Dynamischer oder statischer Störimpuls auf einer Signalleitung einer Schaltung, der zu *Glitches* und *Races* führen kann **[65Mill]**$_{L.2}$.
Isochronic Fork	Verzweigungen oder Knoten, die als Äquipotentialflächen aufgefaßt werden
Liveness	Bezeichnet eine Eigenschaft von *Signal Transition Graphs* **[87Chua]**$_{L.2}$. Jeder Signalwechsel muß dabei von jedem anderen Signalwechsel ausgelöst werden können.
Marking	Ein Marking ist eine Zuweisung von Tokens an Signalwechsel im *Signal Transition Graph*. Er beschreibt die Verbindung zwischen Übergängen (*Transitions*) an den Kanten des Graphen. *Markings* bilden somit die "Orte", an denen die aktuellen Signalwechsel ablaufen. Anhand der *Markings* kann man mit Hilfe der *Transitions* die Prozesse im *Signal Transition Graph* verfolgen. Das im *Signal Transition Graph* abgebildete *Token* beschreibt dabei den initialen Zustand.
Metastabilität	Die Eigenschaft des Ausgangssignals eines Elements, bei dem sich für einen Zeitraum ein Pegel zwischen den binären Werten einstellt.
Micropipelines	Ein asynchrones Entwurfsprinzip für VLSI Schaltungen, welches mit Hilfe der *Bundled Data Convention* die Schaltungsmodule synchronisiert **[89Suth]**$_{L.2}$.
Muller C-Element	Ein Modul zur Synchronisation von Signalen. Es wird auch als *Rendezvous Element* bezeichnet.

Mutual Exclusive-Element	Ein Element zum gegenseitigen Ausschließen von zwei Signalwechseln **[80MeadC]**$_{L.2}$. Es wird auch als *Interlock Element* bezeichnet.
Noninput Choice Net	Bezeichnung für das Verhalten eines Signalwechsel, der erst ausgelöst wird, wenn eine Bedingung erfüllt ist **[87Chua]**$_{L.2}$.
Persistency	Eigenschaft eines *Signal Transition Graph*, wenn gewährleistet ist, dass ein Signalwechsel keinen anderen Wechsel invalidiert **[87Chua]**$_{L.2}$.
Petri Netz (Petri Net)	Ein Verfahren zur Modellierung von Systemen und nebenläufigen Prozessen. **[62Petr]**$_{L.2}$ bildet die Basis der modernen asynchronen Schaltungstheorie.
Place	Bezeichnet die "Orte" im *Signal Transition Graph*, an denen sich *Tokens* aufhalten können **[87Chua]**$_{L.2}$.
Pulse Mode	Eine Variante des *Fundamental Modes* für impulsförmige Signale.
Race	Bezeichnung für einen falschen stabilen Folgezustand, der durch ein *Hazard* erzeugt werden kann. Ein Automat wertet dabei einen Störimpuls als neues Eingangssignal aus und wechselt in einen stabilen aber falschen Folgezustand.
Save	Ein *Signal Transition Graph* besitzt diese Eigenschaft, wenn ein *Place* nie mehr als ein *Token* beinhalten kann **[87Chua]**$_{L.2}$. Hierdurch wird gewährleistet, dass Signalwechsel in einer nicht kontrollierbaren Reihenfolge ausgeführt werden.
Self-timed Circuit	Asynchrone Schaltung, deren Synchronisation durch lokal gesteuerte *Handshake* Signale erfolgt **[80MeadC, 80Seita]**$_{L.2}$.
Single-Rail	Datenverarbeitung bei asynchronen Schaltungen, bei der die Synchronisationssteuerung und die Datenverarbeitung auf getrennten Leitungen erfolgen **[80MeadC, 80Seita]**$_{L.2}$.
Speed-independent	Eine Schaltungseigenschaft, die erfüllt ist, wenn beliebige, aber endliche Verzögerungen in den Elementen zu keinem Fehlverhalten führen. Die Verbindungsleitungen werden dabei als verzögerungsfrei angesehen **[65Mill]**$_{L.2}$.
Signal Transition Graph	Spezialform eines *Petri Netzes*, die in **[87Chua]**$_{L.2}$ vorgestellt wurde. Sie korrespondiert zu den *Marked Graphs*.
State Graph	Zustandsgraph einer Schaltung.
State Signals	Eingangssignale (*Excitation State Signals*) und Ausgangssignale (*Response State Signals*) des Rückkopplungspfads eines Schaltwerks.
Token	Ort im *Signal Transition Graph*, der den Signalwechsel festlegt. *Tokens*, werden im *Signal Transition Graph* als ausgefüllter Kreis dargestellt und definieren die Initialzustände des Graphen. Mit Hilfe der *Tokens* können die Signalwechsel im Graphen verfolgt werden.
Transition	*Transitions* bilden die Knoten des *Signal Transition Graphs.* Wenn eine Transition „feuert“ findet der betreffende

	Signalwechsel statt. Mit ihrer Hilfe kann man die Signalwechsel im Graphen bestimmen und verfolgen.
Two-Phase Signalling	Ein Verfahren, welches zwei Module durch eine Ereignissteuerung synchronisiert (auch *Two-Cycle Signalling* oder *Event Signalling*) **[80MeadC]**$_{L.2}$. Dies bedeutet, dass zum Auslösen eines Signalwechsels nicht der Pegel, sondern dessen Wechsel entscheidend ist. Z.B.: Ein XOR verändert beim Wechsel eines seiner Eingangssignale immer sein Ausgangssignal. Es wird daher auch als OR bei *Event Signalling* bezeichnet und entsprechend eingesetzt.

Abkürzungen und Signalbezeichnungen

_n	not, Bezeichnung für ein invertiertes Signal
Acknowledge (A)	Synchronisationssignal für *Self-timed Circuits*, welches vom Empfänger generiert wird
ABC	Asynchronous BILBO Cell
AI	Acknowledge Input
AO	Acknowledge Output
ASF	Asynchronous Flip-Flop, Scanregister Zelle für das passive HIOB-Konzept mit lokalen Steuerelementen
BILBO	*Built-In Logic Block Observer* **[79KoenMZ]**$_{L.1}$
C	*Capture Signal* eines Registers
DFT	Design For Testability
DI	Data Input, Dateneingang der *Micropipeline*
DO	Data Output, Datenausgang der *Micropipeline*
e	even Mode, Bezeichnung für die geraden Registerzellen bei einer *Capture-Pass Register* Struktur
ED	Empty Detection
F	Full Set
FD	Full Detection
FSM	*Finite State Machine*
H_n	Hold, Steuersignal im passiven HIOB-Konzept für das Scanregister
HCA	HIOB Ablaufsteuerung für den Scanpfad
HCP	HIOB Ablaufsteuerung für aktive Testhilfen
HIOB	Abkürzung für Handshake Interface Observer Block; Bezeichnung für die hier entwickelten asynchronen DFT Strukturen

HIOBA	Abkürzung für Handshake Interface Observer Block für aktive Testhilfen
HIOBP	Abkürzung für Handshake Interface Observer Block für passive Testhilfen
HRA	HIOB Register für aktive Testhilfen
HRP	HIOB Register für den Scanpfad
IDDQ	Eine Testmethode, bei welcher der erhöhte Ruhestrom einer Schaltung gemessen wird
INIT	Internes Steuersignal von HIOBA
INV	Inverter
LFSR	Abkürzung für *Linear Feedback Shift Register*
MISR	Abkürzung für *Multiple Input Shift Register*
o	odd Mode, Bezeichnung für die ungeraden Registerzellen bei einer *Capture-Pass Register* Struktur
OPERATE	Operationsmodus im HIOBA-Konzept
P	*Pass Signal* eines Registers
PIPO	Parallel In Parallel Out, Abkürzung für eine der Betriebsarten der Testkonzepte
PISO	Parallel In Serial Out, Abkürzung für eine der Betriebsarten der Testkonzepte
PRELOAD	Modus zum Laden neuer Testmuster im HIOBA-Konzept
Request (R)	Synchronisationssignal für *Self-timed Circuits*, welches vom Sender generiert wird
RI	Request Input
RO	Request Output
RST, RST_n	Reset Signal eines Registers, Rücksetzen auf "0"
Re_n (Ro_n)	Reset Signal für die geraden (ungeraden) Registerstellen
S	Set Signal eines Registers, Setzen auf "1"
SAMPLE	Betriebsart zum Speichern von Testdaten im seriellen Registerpfad von HIOBA
SC	Scan Cell
SCe (SCo)	Gerades oder oberes (ungerades oder unteres) Latch eines *Capture-Pass Registers*
SCAN	Schieberegisterbetriebsart von HIOBA
SCANe	Schieberegisterbetriebsart für die geraden Registerzellen von HIOBA
SCANo	Schieberegisterbetriebsart für die ungeraden Registerzellen von HIOBA
Se_n (So_n)	Set Signal für die geraden (ungeraden) Registerstellen
SI	Internes Steuersignal von HIOBA
SIPO	Abkürzung für Serial In Parallel Out; Bezeichnung für den asynchronen Scanbetrieb, bei dem die Daten seriell angelegt und parallel abgegriffen werden
SISO	Abkürzung für Serial In Serial Out; Bezeichnung für den asynchronen Scanbetrieb
STG	Abkürzung für Signal Transition Graph

TA	Test Acknowledge, interne Steuersignale der HIOB-Konzepte
TAI	Test Acknowledge Input
TAO	Test Acknowledge Output
TCe (TCo)	Test *Capture* Signal für die geraden (ungeraden) Registerzellen
TDA	Test Data Analyzer, Abkürzung für eine der Betriebsarten des aktiven Testkonzepts
TDI	Test Data Input
TDO	Test Data Output
TFB	Test Feed Back
TI	Internes Steuersignal von HIOBA für die Initialisierung
TMC	Testable Muller C-Element, testfreundliches Muller C-Element
TMI	Internes Steuersignal von HIOBA
TMO	Internes Steuersignal von HIOBA
TMS	Test Mode Select
TO	Internes Steuersignal von HIOBA für die Initialisierung
TPG	Test Pattern Generator, auch Abkürzung für eine der Betriebsarten des aktiven Testkonzepts
TPR	Test PRELOAD, externes Steuersignal von HIOBA
TR	Test Request, internes Steuersignal der HIOB-Konzepte
TRI	Test Request Input, externes Steuersignal der HIOB-Konzepte
TRO	Test Request Output, externe Steuersignale der HIOB-Konzepte
TSe	Test Select even, internes Steuersignal von HIOBA
TSH	Test Shift, externes Steuersignal von HIOBA
TSI	Test Select Input, externes Steuersignal von HIOBP
TSM	Test Sample, externes Steuersignal von HIOBA
TSo	Test Select odd, internes Steuersignal von HIOBA
TSO	Test Select Output, externes Steuersignal von HIOBP
TSR	Test Select Request, externes Steuersignal von HIOBA
TX	Test XOR, Signal zum Umschalten in den MISR Betrieb
XL	Asynchronous MISR Latch

Abbildungsverzeichnis

Kapitel 1

Einleitung

Digitale asynchrone Schaltungen besitzen gegenüber synchronen Schaltungen, die durch einen globalen Takt gesteuert werden, eine asynchrone lokale Steuerung. Die Entwicklung asynchroner Schaltungen wurde zeitlich parallel zu der synchronen Schaltungstechnik vorangetrieben. Die Trennung von asynchronen und synchronen Entwurfskonzepten erfolgte schon in den fünfziger Jahren bei den ersten Integrationen von digitalen Schaltungen.

Synchrone Schaltungen eignen sich besser für Ad-hoc Entwicklungsmethoden, bei denen *Hazards* und *Races* bei einer Funktionsberechnung zwar nicht eliminiert, aber die negativen Auswirkungen auf den Zustand der Schaltung durch Variation der Taktperiode unterdrückt werden können. Diese Eigenschaften sind ein Grund für die hohe Akzeptanz der synchronen Schaltungsmethodik. Ein modulares Design, welches aus selbst synchronisierenden Einheiten besteht, eignet sich in hohem Maße für höchstintegrierte Schaltungen, bei denen die Modularität, die Wiederverwendbarkeit und die Synthesefähigkeit der einzelnen Schaltungskomponenten im Vordergrund stehen. Der Bedarf an einer Designmethodik dieser Art ergibt sich in der digitalen integrierten Schaltungstechnik durch die enormen, technologiebedingten Innovationsschübe aus der Halbleiterproduktion. Der testfreundliche Entwurf dieser Bausteine mit asynchroner Steuerung nimmt an Bedeutung zu, da die Steuer- und Beobachtbarkeit der Module einer Schaltung mit zunehmender Komplexität abnehmen. Auch die kurze

Zeitdauer, die für den Entwurf des Tests zur Verfügung steht, benötigt strukturierte Maßnahmen, die sicher zum Ziel führen.

Eine Datenübertragung bei asynchronen Schaltungen erfolgt mit Hilfe von Übertragungskonventionen zwischen Modulen, die jeweils als Sender und Empfänger arbeiten. Jedes Interface zwischen zwei Modulen besitzt somit eine eigene Synchronisierung. Geschwindigkeitsunterschiede werden über das Übertragungsprotokoll ausgeglichen. Intern arbeiten die Module mit einer von anderen Modulen unabhängigen Geschwindigkeit. Somit entfallen das Problem der Balancierung eines Taktnetzwerks sowie die Verwendung unterschiedlicher Taktdomänen. Auch die Störung der Schaltung durch das hochfrequente Taktsignal entfällt sowie eine Dimensionierung der Stromversorgung auf den maximalen Stromverbrauch wird nicht benötigt.

Aus den genannten Eigenschaften ergeben sich Vorteile, aber auch Nachteile gegenüber synchronen Schaltungen, die im Einzelfall für ein Design abzuwägen sind. In den letzten Jahren wurden neue Konzepte zum systematischen Entwurf asynchroner Schaltungen vorgestellt. Der testfreundliche Entwurf asynchroner Schaltungen blieb dabei weitgehend unberücksichtigt. Ein Grund hierfür sind der in geringem Maße erfolgte kommerzielle Einsatz dieser Schaltungen und der damit fehlende Bedarf an Test- und Verifikationsmethoden von integrierten asynchronen Schaltungen in der Massenproduktion.

Einen neuen Ansatz zur Schaltungsentwicklung mit asynchronen Signalen zeigte Seitz, der eine lokale Steuerung der Module vorsieht, statt einer globalen Taktsteuerung, wie sie bei synchronen Schaltungen nötig ist **[80MeadC]**$_{L.2}$. Diese Schaltungen werden als *Self-timed Circuits*[1] bezeichnet, die eine vom Datenpfad getrennte lokale Signalablaufsteuerung besitzen. Der Entwickler kann dabei die Granularität der Schaltung mit der Größe der Module individuell bestimmen. Dieses Schaltungsprinzip wurde von Sutherland für eine asynchrone Designmethodik aufgegriffen, die man auch als *Micropipeline* bezeichnet **[89Suth]**$_{L.2}$. Der Steuerpfad arbeitet dabei mit *Two-Phase Signalling.* Wenige Jahre danach wurde aufbauend auf diesem Prinzip ein asynchroner Mikroprozessor entwickelt, der bit-kompatibel zu einem synchronen Mikroprozessor ist **[93FurbDG]**$_{L.2}$. Parallel hierzu sind weitere Arbeiten vorgestellt worden, die die Möglichkeit eines

[1] Um Mißverständnissen vorzubeugen, wird in dieser Arbeit weitgehend darauf verzichtet, Fachbegriffe zu übersetzen.

praktischen Einsatzes von asynchronen Schaltungen unterstreichen[2]. Mit **[93Furb, 93Poun, 95D&T]**$_{L.2}$ wurden Vorteile moderner asynchroner Schaltungen aufgezeigt, die in Bereichen der Höchstintegration, *Low Power Design*, *Design Reuseability* und Geschwindigkeitsoptimierung liegen.

Parallel zu diesen asynchronen Schaltungsentwürfen wurden erste Testkonzepte vorgestellt, die sich stark an synchronen Testmethoden orientieren **[89LeenSa, 94KhocB, 94Ronc, 95PetlF]**$_{L.3}$. Ein testfreundlicher Entwurf mit synchronen Methoden bedeutet hier das Abbilden der asynchronen Eigenschaften auf eine synchrone Schaltung für die Dauer des Tests.

Das dynamische Verhalten asynchroner Schaltungen widerspricht nicht nur grundlegend dem von synchronen Schaltungen. Auch die Steuerung in einem für synchrone Schaltungen optimierten Prüfprozeß ist unterschiedlich. Die globale Taktsteuerung und die Parallelisierung der Messungen mit Prüfautomaten im Produktionsprozeß erzeugen ein großes Problem für den ökonomischen Test lokal getakteter, asynchroner Schaltungen. Es gibt zur Zeit keinen Vorschlag, eine Synchronisation zwischen einer asynchronen Schaltung als Testobjekt und einem synchronen Testautomaten herzustellen. Ein testfreundlicher Entwurf, der diese Randbedingungen berücksichtigt, kann das Problem entschärfen und die Massenproduktion asynchroner Schaltungen ermöglichen.

In einer Diskussion werden die vorgeschlagenen Konzepte auf ihre Eigenschaften hin bewertet. Hier gilt es zu klären, ob sich eine asynchrone Schaltung durch diese Abbildung auf eine synchrone oder kombinatorische Schaltung im Testbetrieb reduzieren läßt. Aufbauend auf den Ergebnissen dieser Diskussion sollen asynchrone Testmethoden entwickelt werden. Eine Ausrichtung erfolgt dabei auf eine häufig verwendete Schaltungsart, welche die *Bundled Data Convention* mit *Two-Phase Signalling* als Übertragungsprotokoll verwendet. Neben dem isolierten Test des Daten- und des Steuerpfads ist bei den vorgestellten Konzepten der Test des Zusammenspiels dieser Schaltungsbereiche im Fokus der Untersuchung. Bei der Entwicklung der Testkonzepte für asynchrone Schaltungen wird dabei versucht, den Einfluß der Testlogik auf die Performance und die Größe der zusätzlichen Schaltungselemente zu minimieren sowie die Konventionen der asynchronen Kommunikationsprotokolle zu berück-

[2] Siehe hierzu **[94BerkBK, 95YunD, 94SprouSM, 95Hauc]**$_{L.2}$.

sichtigen. Es soll damit ermöglicht werden, universelle Testverfahren für asynchrone Schaltungen zu entwerfen.

Die Arbeit gliedert sich in folgende Kapitel. **Kapitel 2** gibt einen Einblick in asynchrone Schaltungen und Entwurfskonzepte. **Kapitel 3** untersucht die bisherige Vorgehensweise zum Test asynchroner Schaltungen. Nach diesem Rückblick werden asynchrone Schaltungsmodule vorgestellt und ihr Fehlverhalten bei Defekten auf Transistorebene in **Kapitel 4** betrachtet. Dazu wird eine asynchrone Schaltungsbibliothek exemplarisch verwendet. Zu der Schaltungsbibliothek werden neue Standardzellen für den testfreundlichen Entwurf vorgeschlagen. Hierzu wird ein Konzept zur Modellierung klassischer und nichtklassischer Fehlermodelle auf Gatterebene präsentiert. In **Kapitel 5** wird zunächst mit Hilfe der zuvor beschriebenen Standardzellen ein Konzept für passive Testhilfen entwickelt, die mit einem synchronen Scanpfad vergleichbar sind. Daran anschließend werden aktive Teststrukturen vorgeschlagen, welche die synchrone BILBO-Technik auf asynchrone Schaltungstechniken erstmals ermöglichen. Für beide Konzepte werden asynchrone Testablaufsteuerungen verwendet. Die Ablaufsteuerungen werden dabei mit Hilfe von *Signal Transition Graphs* entwickelt **[87Chua]**$_{L.2}$. Die Arbeit endet mit einer Diskussion über die Vor- und Nachteile der vorgestellten Konzepte zum Test asynchroner Schaltungen.

Kapitel 2

Entwurfsmethoden für asynchrone Schaltungen

In diesem Kapitel folgt ein kurzer Überblick über die asynchrone Schaltungsentwicklung. Neben den unterschiedlichen Betriebsarten und Eigenschaften werden Konzepte der Automatentheorie vorgestellt, die für diese Arbeit von Bedeutung sind. Anwendungsgebiete von asynchronen Schaltungen und Modulen finden sich neben Interface-Schaltungen oder Bussteuerungen zunehmend auch in größeren Modulen, die bisher von dem synchronen Schaltungsentwurf beherrscht wurden **[95Hauc, 93FurbDG]**$_{L.2}$. Die vorgestellten Schaltungen sind noch weitestgehend in einem Prototypenstadium. Wichtig für eine Massenproduktion von asynchronen Schaltungen ist, dass sie verifizier- und testbar sind. Vorteile bei einem asynchronen Schaltungsentwurf sind:

Die Versorgungsspannung muß nicht für die Spitzenströme während der Taktflanke ausgelegt werden.

Taktnetzwerke werden nicht mehr benötigt.

Ein modularer Schaltungsentwurf ist möglich.

Für den Entwurf von Schaltungen sind Methoden entwickelt worden, um sie auf endliche Automaten abzubilden. Bei der Entwicklung dieser Entwurfsmethoden gab es zunächst keine prinzipielle Differenzierung zwischen synchronen und

asynchronen Schaltungen. Shannon entwickelte grundlegende Beschreibungsformen für digitale Schaltungen mit Hilfe von Gattern **[49Shan]**$_{L.2}$. Quine formulierte das Problem zur Vereinfachung von Logikfunktionen mit Hilfe von Algorithmen **[52Quin]**$_{L.2}$. Die Abgrenzung in zwei unterschiedliche Entwurfsansätze ist erst durch den Wunsch nach Ad-Hoc-Methoden im Schaltungsentwurf entstanden. Diese Entwurfsansätze lassen sich nur ungenügend mit einer Automatentheorie beschreiben. Bei Ad-Hoc-Entwicklungsmethoden werden fehlerhafte Übergänge, die aus *Hazards* hervorgerufen werden können, vermieden. Die Wirkungen von *Hazards* können durch ein Taktnetzwerk maskiert werden.

Den Ursprung der Theorien zum asynchronen Entwurf bilden die Automatentheorien und Gedankenexperimente aus den fünfziger und sechziger Jahren. Der Aufbau teilt sich dabei in ein Schaltnetz sowie ein Rückkopplungsnetzwerk auf. Bei synchronen Automaten besitzen die Rückkopplungspfade Speicherelemente, die mit einem synchronen Takt gesteuert werden. Zur Beschreibung und Minimierung der Logik wurden häufig Gatter oder ein Karnaughdiagramm verwendet. Huffmann entwickelte eine Vorgehensweise, mit der man sequentielle Schaltungen in Form von Matrizen beschreiben kann **[54Huff, 57Huff]**$_{L.2}$. Diese werden bei der Realisierung in ein Schaltnetz mit globalen Rückkopplungen umgewandelt. Zudem entwirft er ein Konzept zur Vermeidung von *Hazards* durch redundante Terme. Moore behandelt in seinen Gedankenexperimenten zu sequentiellen, endlichen Automaten unter anderem das Problem der Maschinenidentifikation **[56Moor]**$_{L.2}$. Aufbauend auf Huffmann und Moore, stellte Mealy eine Methode vor, um synchrone Schaltungen zu synthetisieren **[55Meal]**$_{L.2}$. Unger entwickelte ein Konzept zum asynchronen Schaltungsentwurf, welches *Hazards* und Verzögerungen der Schaltungsmodule berücksichtigt **[59Unge]**$_{L.2}$.

Der Unterschied zwischen synchronen und asynchronen Schaltungen bei diesen vorgestellten Verfahren ist der fehlende Takt in dem rückgekoppelten Pfad bei letzteren. Diese klassischen Synthesemethoden lassen sich ideal in ein PLA-basiertes Design integrieren. Dies erhöht die Attraktivität der Entwicklungsmethodik. Da PLAs aber immer seltener in Schaltungen eingesetzt werden, gibt dies einen Hinweis, dass diese Form der Schaltungsbeschreibung moderne Architekturen nur mangelhaft modellieren kann.

Asynchrone Schaltungsmodelle in *Fundamental Mode* basieren auf dem von Huffmann vorgestellten Prinzip und werden in der Regel mit Zustandsfolgeta-

bellen beschrieben. Aus diesen Tabellen werden die Zustandsgraphen[3] gewonnen, die zu einer direkten Implementierung einer Schaltungsfunktion führen, wie in **Abb. 1** zu sehen ist. Der Graph auf der linken Seite stellt dabei den Zustandsgraphen des *Muller C-Elements* dar. Die Signale zwischen den Zuständen stellen die Wechsel der Eingangssignale des Elements dar[4]. Der Start zeigt den Initialisierungszustand des Zustandsgraphen an. Hierbei können sich stabile und instabile Zustände einstellen.

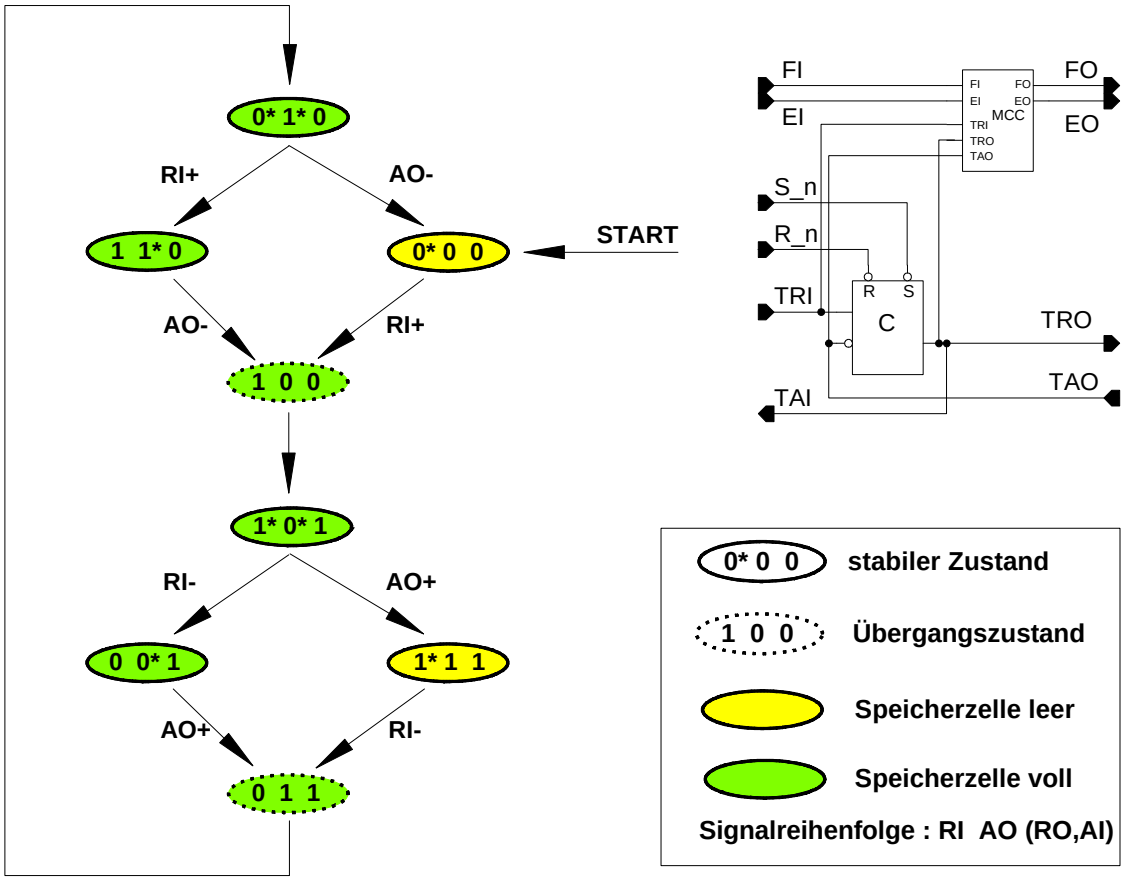

Abb. 1: Der State Graph des *Muller C-Elements* mit den zugehörigen Signalwechseln.

[3] Sie werden auch als *State Graphs* bezeichnet.

[4] Die Eingänge RI (TRI) und AO (TAO) sowie der Ausgang RO (TRO), der auch als AI (TAI) bezeichnet wird, sind hier analog zu den Steuersignalen bei asynchronen Schaltungen mit *Bundled Data* gewählt. Siehe hierzu auch die Abkürzungen und Signalbezeichnungen.

Auf der rechten Seite findet sich die Darstellung der Anwendung des *Muller C-Elements* für die Steuerung eines asynchronen Scanpfads. Schließt man den Takteingang eines Speicherelements an den Ausgang TRO an, kann das Datum abhängig von den Signalwechseln der beiden Eingänge TRI und TAO gespeichert werden[5].

Asynchrone Schaltungen in Fundamental Mode besitzen eine Ähnlichkeit zu synchronen Schaltungen. Es wird dabei davon ausgegangen, dass alle Verzögerungen der Schaltung bekannt bzw. bestimmbar sind. Dieses wird auch als *Bounded Delay Model* bezeichnet. Bei diesen Schaltungen muß bei der Bestimmung der Funktionen darauf geachtet werden, dass keine statischen oder dynamischen *Hazards* auftreten. Die *Hazard* Vermeidung kann dabei nur unter der Annahme gewährleistet werden, dass sich immer nur ein Eingangssignal ändert[6]. Dies ist eine starke Einschränkung für die Signalwechsel der Eingänge und führt bei großen Schaltungen zu extrem aufwendigen und komplexen Analysen. Eine weitere Darstellungsform für asynchrone Schaltungen mit *Bounded Delay* ist der *Burst Mode*. Hier ändert sich immer ein Bündel von Signalen am Eingang. Die Antwort wird mit einem lokalen Takt in dem Modul berechnet.

Ein Überblick über die *Switching Theory* mit asynchronen Schaltungen gab Miller **[65Mill]**$_{L.2}$. Zur Beschreibung von Schaltungseigenschaften werden dabei die Begriffe *Speed-independent* und *Delay-insensitive* verwendet. Hierbei wird auch das Majoritätselement, das *Muller C-Element*, zur Synchronisation von Signalen eingesetzt, welches von Muller **[59Mull]**$_{L.2}$ vorgestellt wurde.

Methoden mit internen Taktsteuerungen für Schaltungsmodule mit asynchronen Ein- und Ausgangssignalen wurden in **[85MolnFR, 88RoseMC]**$_{L.2}$ beschrieben. Die Schaltungsmodule werden dabei als *Q-Modules* bezeichnet und enthalten speziell entwickelte Speicher, die *Q-Flops*, die mit Hilfe eines Steuermoduls kontrolliert werden. Eine weitere Synthesemethode mit intern getakteten Modulen, die auch mehrere wechselnde Eingangssignale verarbeiten kann, wurde von **[91NowiD]**$_{L.2}$ entwickelt.

---

[5] Das MCC-Element wird in einer später beschriebenen Anwendung des *Muller C-Elements* in einem Scanpfad verwendet und ist in dieser Abbildung nicht von Bedeutung. Für den Scanpfad wird der Präfix **T** hinzugefügt.

[6] Dieses wird auch als *Single Input Change* bezeichnet.

Asynchrone Schaltungen befinden sich in vielen Realisierungen des alltäglichen Gebrauchs. Batteriebetriebene Uhren haben in der Regel einen asynchronen Aufbau, da hier das Ziel eines geringen Stromverbrauchs eine große Bedeutung hat. Auch digitale Speicher sind Module mit asynchronen Ein- und Ausgängen. Ihre Asynchronität ist ein stetes Problem in der Ansteuerung mit synchronen Bausteinen. Vorschläge zur asynchronen Realisierung der Ansteuerung sind in **[87HayeTL, 92NowiYD]**$_{L.2}$ zu finden. Moderne Speicherentwicklungen gehen jedoch dazu über, direkt um den Speicher eine synchronisierende Schale zu konstruieren und das asynchrone Verhalten zu unterdrücken.

Als breites Einsatzgebiet wurden Interface-Bausteine für Verbindungsnetzwerke präsentiert[7]. Asynchrone Kommunikationsbausteine für Infrarot-Übertragungen wurden in **[94MarsCS]**$_{L.2}$ vorgestellt. SCSI Controller mit Hilfe von asynchronen Synthesemethoden wurden in **[93YunD, 95YunD]**$_{L.2}$ entwickelt. Realisierungen von Recheneinheiten, wie eine CMOS ALU **[93Gars]**$_{L.2}$, ein Multiplizierer **[93SparNN]**$_{L.2}$ und ein Modulo N-Counter **[92EberP]**$_{L.2}$ existieren in asynchroner Schaltungstechnik. Die Verwendung von FPGA Bausteinen für asynchrone Schaltungen wurde in **[94HaucBB]**$_{L.2}$ demonstriert.

Die Fülle der vorgestellten Konzepte täuscht indes nicht darüber hinweg, dass sich diese Arbeiten im wesentlichen auf wissenschaftliche Untersuchungen konzentrieren und keine breite Anwendung darstellen. Auch die EDA Software ignoriert weitgehend die Möglichkeiten asynchroner Schaltungstechnik und deren Synthesemethoden. Mikroelektronikproduzenten verwenden nur vereinzelt asynchrone Schaltungen. Ein weiteres Hindernis bei der Verbreitung von asynchronen Schaltungen ist, dass nur wenige Schaltungsentwickler Kenntnisse im Entwurf asynchroner Schaltungen besitzen. Kommerzielle Programme, die auch in eine asynchrone Entwicklungsumgebung passen, bilden eher eine Ausnahme **[94AshkEF]**$_{L.2}$.

Erste Ansätze zur Entwicklung einer Bibliothek sind in dem Umfeld des asynchronen Mikroprozessors Amulet entstanden **[94Pave]**$_{L.2}$[8]. Ein Abbilden eines

[7] Hierzu gibt es zahlreiche Veröffentlichungen **[86DalIS, 87DalIS, 86OrtoPT, 89Subr, 92BrunMS, 92LeonB, 93LavaS, 92Yako]**$_{L.2}$.

[8] In dem Projekt Esprit 6143 - Exact wurde an der Manchester University eine asynchrone Schaltungsbibliothek entwickelt.

DCC-Decoders auf eine asynchrone Schaltungsbibliothek zeigte die Vorteile von asynchronen Bibliothekselementen **[95FarnsEL]**$_{L.2}$. Weitere Arbeiten wurden in einem industriellen Umfeld der Firma SUN von Sutherland und Sproul zum asynchronen Schaltungsentwurf vorgestellt **[94SproSM]**$_{L.2}$.

Neben der klassischen Graphentheorie, die mit globalen Signalen arbeitet, gibt es weitere Verfahren, bei denen man die Granularität der Rückkopplungen individuell bestimmen und mit Graphen formulieren kann. Petri stellte eine neue Darstellung zur Beschreibung von Schaltnetzen und Schaltwerken vor **[62Petr]**$_{L.2}$. Es können damit Prozesse über ihr Signalübergangsverhalten beschrieben werden. Snepscheut beschrieb eine Methode zur Implementierung von Programmen als integrierte Schaltungen mit dem Ziel der Parallelisierung von Prozessen **[83Snep]**$_{L.2}$. Udding entwickelte eine Methode zur formalen Beschreibung von *Delay-insensitive Circuits* **[86Uddi]**$_{L.2}$. Eine weitere Synthesestrategie für *Delay-insensitive Circuits* wurde vorgestellt und als *Communication Process* (CSP) bezeichnet **[86Mart, 87Mart]**$_{L.2}$. Eine spezielle Form der CSPs wurde als Eingabesprache in einer dafür entwickelten Syntheseumgebung eingesetzt **[88BerkRS, 88BerkS, 90Brow, 90LamL 91BerkKR]**$_{L.2}$. Decoder zur Korrektur von Signalen von portablen Compact Disk Spielern wurden in **[92KessBB, 94BerkBK]**$_{L.2}$.

Eine Methode zur Analyse und Verifikation von *Self-timed Circuits* wurde in **[92KishKTV]**$_{L.2}$ aufgezeigt, die eine große Ähnlichkeit zu *Signal Transition Graphs* aufweist. **[95Hulg]**$_{L.2}$ beschäftigte sich mit der Analyse der Performance und Verifikation des Zeitverhaltens von asynchronen Schaltungen, die mit Pertinetzen modelliert sind. Eine weitere Synthesemethode für *Self-timed Circuits* ist in **[92DaviGYa, 92DaviGYb]**$_{L.2}$ vorgestellt. Sie kam bei der Generierung von Mikrocontrollern zum Einsatz. **[93Ende, 94FarnES]**$_{L.2}$ beschäftigten sich mit der Analyse des Stromverbrauchs in asynchronen Schaltungen. Eine Untersuchung von Systemen, die synchrone und asynchrone Module enthalten, analysierte die Synchronisationsmechanismen dieser heterogenen Bausteine **[92AfgaS]**$_{L.2}$.

Signal Transition Graphs (STG) wurden von Chu entwickelt **[87Chua, 87Chub]**$_{L.2}$. Sie stellen eine Unterklasse von Petri Netzen dar. Eine Erweiterung zur

Hazardvermeidung bei STGs wurde in **[91MoonSB]**$_{L.2}$ aufgezeigt[9]. Zur Reduktion von Signalwechseln und die Transformation von STGs auf eine geringere Komplexität präsentierten **[90HungM, 90VanbCG]**$_{L.2}$ ihre Forschungsergebnisse. Auf Basis von STGs wurde eine Methode zur Synthese asynchroner Schaltungen unter der Annahme des *Bounded Wire Delays* entwickelt, die keine *Hazards* aufweisen **[91LavaKS]**$_{L.2}$.

Zur softwarebasierten Schaltungsentwicklung sind in Forschungseinrichtungen Programme entwickelt worden, die häufig über das Internet verfügbar sind. Ein Softwarepaket zur automatischen Synthese, SIS[10], wurde vorgestellt **[92SentS-L]**$_{L.2}$. Ein weiteres Syntheseprogramm mit Namen Tangram wurde bei der Firma Philips entwickelt. Es besteht aus einzelnen Komponenten, die mit Hilfe einer Beschreibungssprache zu Schaltungen zusammengesetzt werden **[91Berk-KR]**$_{L.2}$. Auch in **[94KishKT]**$_{L.2}$ wurde ein Prinzip zur Schaltungsentwicklung auf Basis der *Change Diagrams* vorgestellt.

[9] Syntheseergebnisse und Verifikationsmethoden für STGs zeigen anhand von Beispielen (*Arbiter* und *Ring Buffer*) die Leistungsfähigkeit dieses Ansatzes **[89DilINS, 89NowiD, 90ChiaK]**$_{L.2}$.

[10] SIS wird an Universitäten und Forschungseinrichtungen ständig weiterentwickelt.

2.1 Signal Transition Graphs

Dieses Kapitel beschreibt die Grundlagen der STGs. Die nachstehenden Erläuterungen basieren im wesentlichen auf den Arbeiten von Chu **[87Chua, 87Chub]**$_{L.2}$. Die Untersuchungen zum testfreundlichen asynchronen Entwurf bauen auf diesem Kapitel auf.

Ein STG ist eine formale Spezifikation der Signalwechsel eines Prozesses, der aus einer informellen Beschreibung, wie zum Beispiel einem Signalablaufdiagramm, gewonnen werden kann. Durch die Eigenschaften der STGs kann diese Automatentheorie auch als eine höhere Beschreibungsform von Prozessen und Schaltungen dienen. Die Idee der STGs ist, dass durch kausale Relationen die beschriebenen Signalwechsel in eine Schaltungsstruktur mit dem gleichen Verhalten umgewandelt werden können. Im Gegensatz zu den Zustandsgraphen werden nicht die Zustände auf die Knoten des Graphen abgebildet, sondern die Signalwechsel. Hierdurch ist bei der Umsetzung der Schaltungsfunktion die Methodik zur Kodierung der Zustände frei. Darüber hinaus können lokale Prozesse beschrieben werden, ohne das Verhalten und die Kodierung von anderen Schaltungsfunktionen zu berücksichtigen.

Durch Untersuchung der STGs auf Eigenschaften wie *Liveness*, *Save*, *Persistency*, kann bei einem Syntheseprozess garantiert werden, dass die Schaltung keine *Deadlocks* oder die Eigenschaft *Speed-independent* besitzt. Ein STG ist *Live*, wenn von jedem erreichbaren *Marking* jede *Transition* ausgeführt werden kann[11]. Er ist *Save*, wenn sich in den *Markings* niemals zwei *Tokens* befinden können. Er ist *Persistent*, wenn garantiert wird, dass ein Signalwechsel von a den Wechsel an b auslöst. Dabei muß der Übergang b abgeschlossen sein, bevor der folgende invertierte Übergang von a wieder erfolgt. **Abb. 2** verdeutlicht die Abhängigkeit zwischen STGs, *State Graphs* und einer integrierten Schaltung.

[11] Die Markings, auch *Tokens* genannt, eines STGs sind eine Sammlung von *Places* und beschreiben mit Hilfe der Übergänge (*Transitions*) zwischen ihnen das Verhalten des Graphens. *Tokens* werden mit ausgefüllten Kreisen dargestellt. Mit *Transitions* werden die Verknüpfungen der *Markings* bezeichnet.

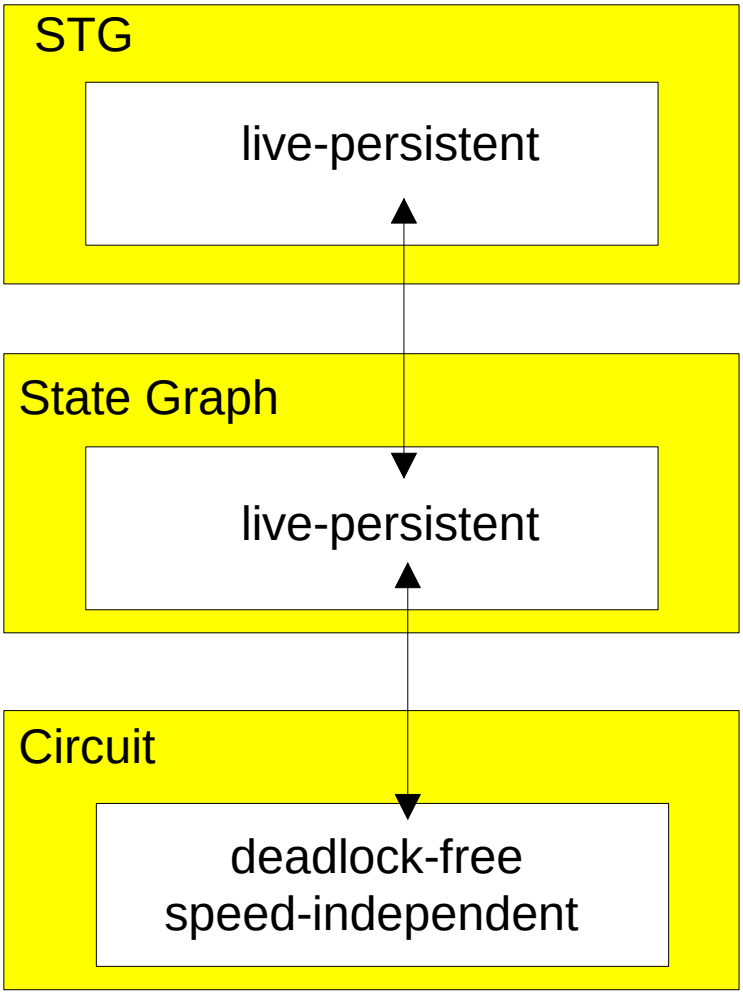

Abb. 2: STGs können Eigenschaften besitzen, die nach einer Synthese robuste Schaltungen ermöglichen.

Mit STGs ist die Entwicklung und Spezifikation einer Schaltungsfunktion möglich, deren Umsetzung mit Syntheseprogrammen erfolgen kann. Der STG wird in ein *State Graph* durch den sogenannten *Process Decomposition* by *Net Contraction* überführt. Hierdurch können die Zustände und Schaltungselemente der Schaltung ermittelt werden. Der erste Schritt ist dabei die Zerteilung der STGs in Untergraphen, die einzelne Funktionen der Schaltung darstellen. Danach kann eine Zustandskodierung für die State Graphs erfolgen. Im letzten Schritt wird der State Graph durch Schaltungselemente ersetzt. **Abb. 3** zeigt die Symbole, die zur graphischen Entwicklung von STGs nach Chu verwendet werden, die jeweils eine Entsprechung zur Syntax der STGs besitzen. Pfeile stellen die Verbindung zwischen *Transition*s dar (**3a**). Ausgefüllte Punkte, sogenannte *Tokens*, dienen zur Kennzeichnung von initialen Zuständen. Sie werden im Verlauf des Prozesses an nachfolgende *Transitions* weitergeleitet. Signalnamen an den Pfeilen stellen eine Bedingung dar, die zur Ausführung der *Transition* erfüllt sein muß (**3a**). Treffen zwei Signalwechsel aufeinander, kennzeichnet man dies zusätzlich mit einem Signalnamen (**3b**). Dies wird auch UND-Verknüpfung (AND) genannt. Es müssen dabei alle Eingänge, hier b- oder a+, mit einem *Token* besetzt sein, bevor der Signalwechsel ausgeführt und das *Token* an den nächsten Signalwechsel weitergeleitet wird. Bei einer ODER -Verknüpfung (OR) (**3c**) genügt ein *Token*, um den Signalwechsel auszulösen. Eingangssignale können mit unter-

strichenen Signalnamen gekennzeichnet werden. Gestrichelte Linien stellen externe Signalwechsel dar.

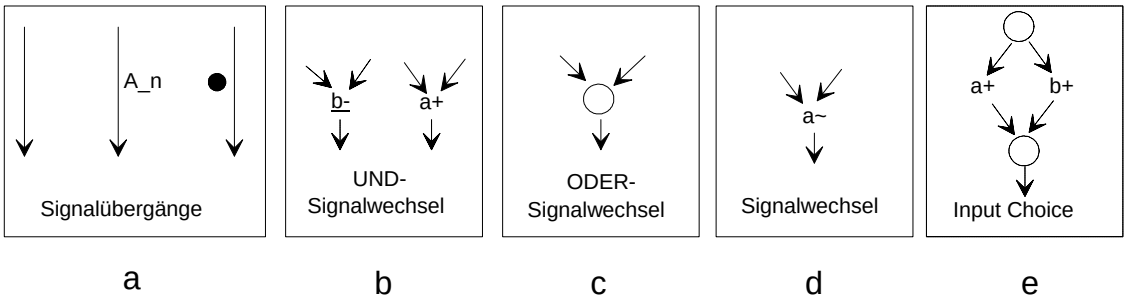

Abb. 3: Symbole zur graphischen Darstellung von STGs.

Von Moon wurde eine Erweiterung der STGs vorgeschlagen, bei der auch Signalwechsel, unabhängig von der Art des Wechsels, verwendet werden können. Dies wird mit dem Zeichen "~" gekennzeichnet (**3d**). Es reduziert somit die Anzahl der darzustellenden Übergänge in STGs **[91MoonSB]**$_{L.2}$. *Free (Input) Choice Nets* und *Noninput Choice Nets* ergeben sich aus mehreren Signalwechseln, die einem Signalwechsel folgen können (**3e**). Bei einem *Noninput Choice* Übergang muß die Bedingung an dem Pfeil wahr sein, um diesen Übergang auszulösen. Eine Bedingung ist hier nicht dargestellt. *Free Choice Nets* ermöglichen einen *Transition* an die folgende Stufe, wenn einer der *Transitions*, hier a+ oder b+, erfolgt ist. Eine einfache Schaltungsrealisierung ist eine XOR-Funktion, die - unabhängig von dem Zustand des anderen Eingangs - den Ausgangspegel bei einem Signalwechsel ändert.

Als Beispiel für die Darstellungsform der STGs wird ein Element zur Realisierung der *Two-Phase* und *Four-Phase Bundled Data Convention* genommen. Im Anfangszustand sind beide Signale, a und b, auf "0". Der Anfangszustand wird dabei mit *Tokens* markiert. Nach dem Wechsel beider Signale auf "1" (a+ und b+), erfolgt auch ein positiver Wechsel an Signal c. Es geht erst dann wieder auf "0" zurück, wenn an beiden "Eingangssignalen" ein negativer Signalwechsel erfolgt ist. Somit wird die Synchronisation der Signale a und b erreicht. Eingangssignale werden auch als externe Signale bezeichnet und bei STGs gestrichelt dargestellt. Die **Abb. 4** stellt den STG des *Muller C-Elements* dar und beschreibt die *Bundled Data Convention* für *Self-timed Circuits*. Links befindet sich die Darstellung nach Chu, in der Mitte die Darstellung nach Moon.

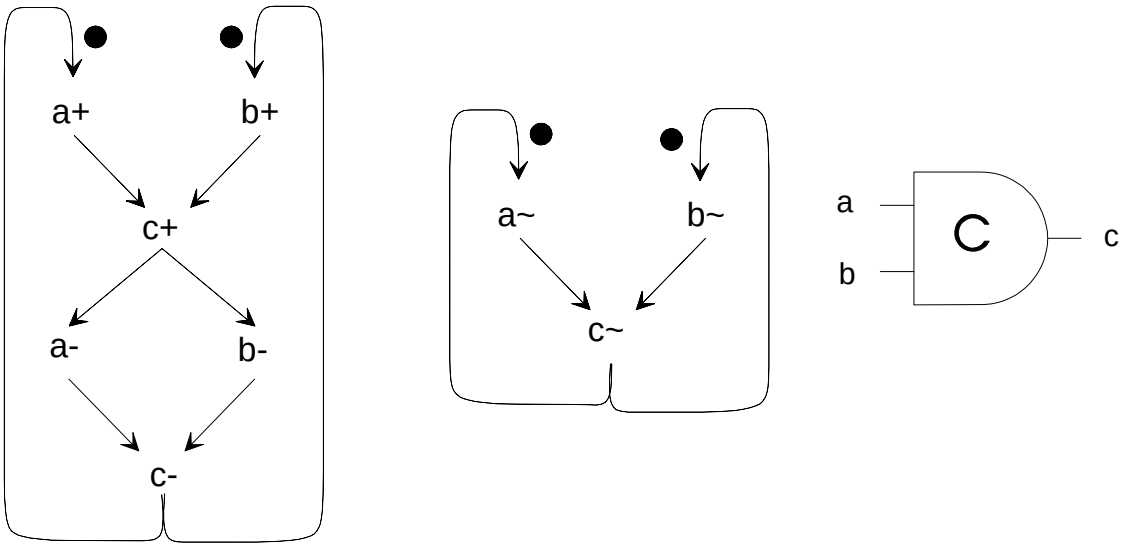

Abb. 4: Der STG des *Muller C-Elements*.

Das *Muller C-Element* (siehe auch **Abb. 21**) dient somit zur AND-Verknüpfung zweier Ereignisse. Das Element ändert immer dann seinen Ausgangswert, wenn beide Eingänge einen Signalwechsel vollzogen haben **[59Mull]**$_{L.2}$. Die Ausgänge der *Muller C-Elemente* schalten die Speicherelemente. *Delay-insensitive Circuits* mit Gattern, die nur einen Ausgang besitzen, können nur aus *Muller C-Elementen*, Invertern und Signalleitungen aufgebaut werden **[95Hauc]**$_{L.2}$. Das Verhalten des *Muller C-Elements* ist der Grund für dessen Bedeutung in asynchrone Schaltungen, die mit Handshake Signalen arbeiten. **Tabelle 2** stellt die Zustände des *Muller C-Elements* dar.

Tabelle 2: Funktionstabelle des *Muller C-Elements*

A	B	C	Erklärung
0	0	0	Der Wechsel auf "0" ist erfolgt.
0	1	Halten	Der Ausgangszustand wird gehalten.
1	0	Halten	Der Ausgangszustand wird gehalten.
1	1	1	Der Wechsel auf "1" ist erfolgt.

Neben der AND-Verknüpfung wird eine OR-Verknüpfung für Signalwechsel benötigt. Hierfür eignet sich das XOR Element, dessen STG in **Abb. 5** zu sehen ist. Wechselt einer der Eingänge a oder b seinen Wert, so wechselt auch der Ausgang c seinen Wert. Der nicht ausgefüllte Kreis stellt die OR-Verknüpfung für Signalwechsel in STGs dar. In der Mitte erkennt man die Darstellung nach Moon,

die eine geringere Anzahl von Signalwechseln benötigt. Bei STGs können sich hinter dem Kreis mehrere *Transitions* mit Signalbedingungen befinden, die zur Ausführung des Signalwechsel zutreffen müssen. Diese Möglichkeit, eine Bedingung an einen Signalwechsel zu knüpfen, nennt Chu *Noninput Choice*. Besitzt ein Signalwechsel mehrere Ausgänge ohne eine Bedingung, wird dies als *Free Choice* oder *Input Choice* bezeichnet. Bei einem XOR ist ein positiver oder negativer Signalwechsel von c abhängig vom aktuellen Pegel des Ausgangs.

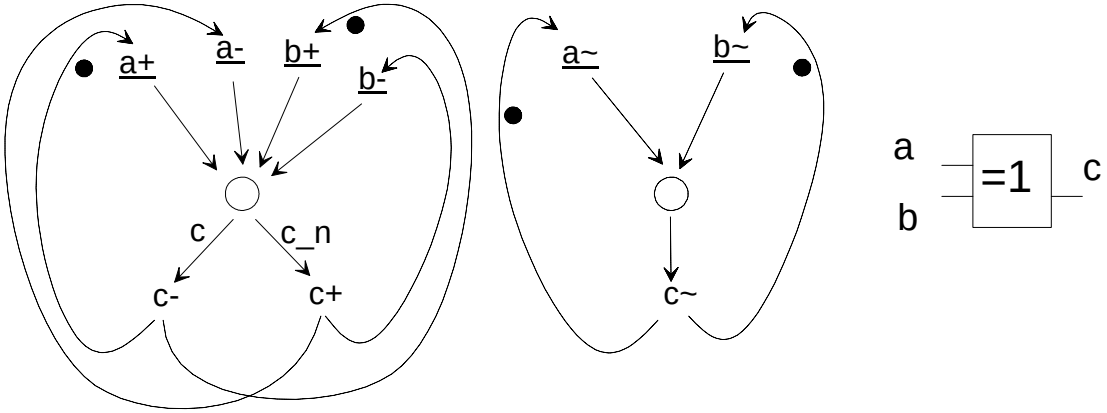

Abb. 5: Das XOR dient zur OR-Verknüpfung zweier Ereignisse.

Sollen Signalwechsel einer Leitung auf zwei Ausgänge abwechselnd verteilt werden, ist ein weiteres Element nötig. **Abb. 6** zeigt den STG des *Toggle* Elements. Man erkennt auch hier an den Signalwechseln Bedingungen, die erfüllt sein müssen, um ein *Token* an dieser Stelle weiterzugeben. Das *Toggle* Element schaltet somit die positiven Flanken auf den Ausgang T und die negativen Flanken auf den Ausgang F. In der Mitte befindet sich wieder die Darstellung nach Moon. Man kann dabei nicht erkennen, welche Initialzustände die Ausgänge des *Toggle* Elements besitzen. Dies muß zusätzlich definiert werden.

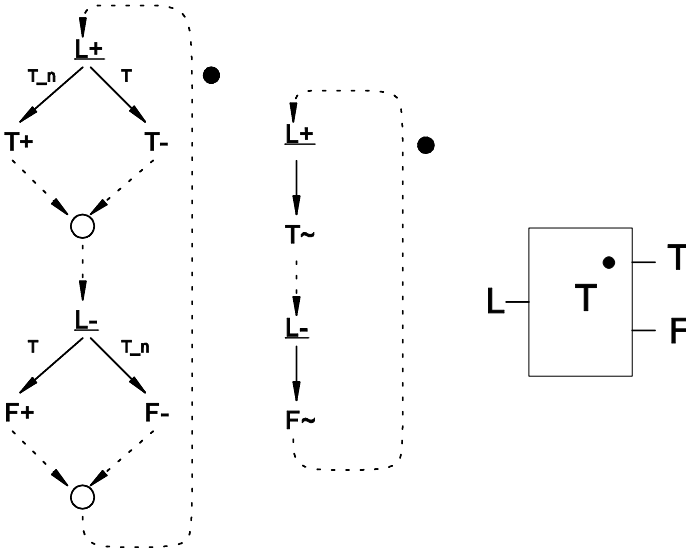

Abb. 6: Das *Toggle* Element verteilt Ereignisse abwechselnd auf Ausgänge.

Zur Entwicklung von STGs können Signalablaufdiagramme verwendet werden. Als Beispiel wird hier die Umwandlung eines *Two-Phase Signals* auf ein *Four-Phase Signal* beschrieben, welches in **Abb. 7** zu sehen ist. Erfolgt der Signalwechsel an dem 2p Signal (**1**), wechselt darauf das Signal 4p (**2**). Das interne Signal T schaltet (**3**), um das Zurücksetzen von 4p zu ermöglichen (**4**). Mit dem Schalten von F wird signalisiert, dass der nächste Wechsel an 2p erfolgen kann. Bei einer Schaltungsrealisierung ist F der Ausgang, der mit dem Eingang 2p verbunden ist. Die Verbindung zwischen beiden Signalen ist hier nicht dargestellt. Danach erfolgen die Signalwechsel für den negativen Signalwechsel von 2p (**5-8**).

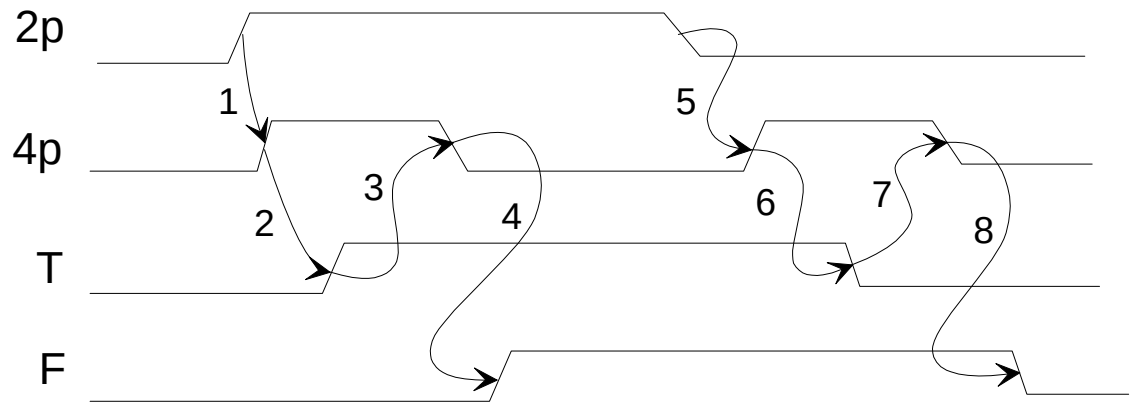

Abb. 7: Das Signalablaufdiagramm beschreibt die Umwandlung von *Two-Phase* auf *Four-Phase* Signale.

Überträgt man dieses Signalablaufdiagramm in einen STG, ergibt sich eine Darstellung, wie in **Abb. 8**. Bei positivem und auch bei negativem Signalwechsel von 2p erfolgt das Signal 4p. Dies wird mit dem Kreis symbolisiert. Auch hier ist auf der rechten Seite die Darstellung nach Moon wiedergegeben.

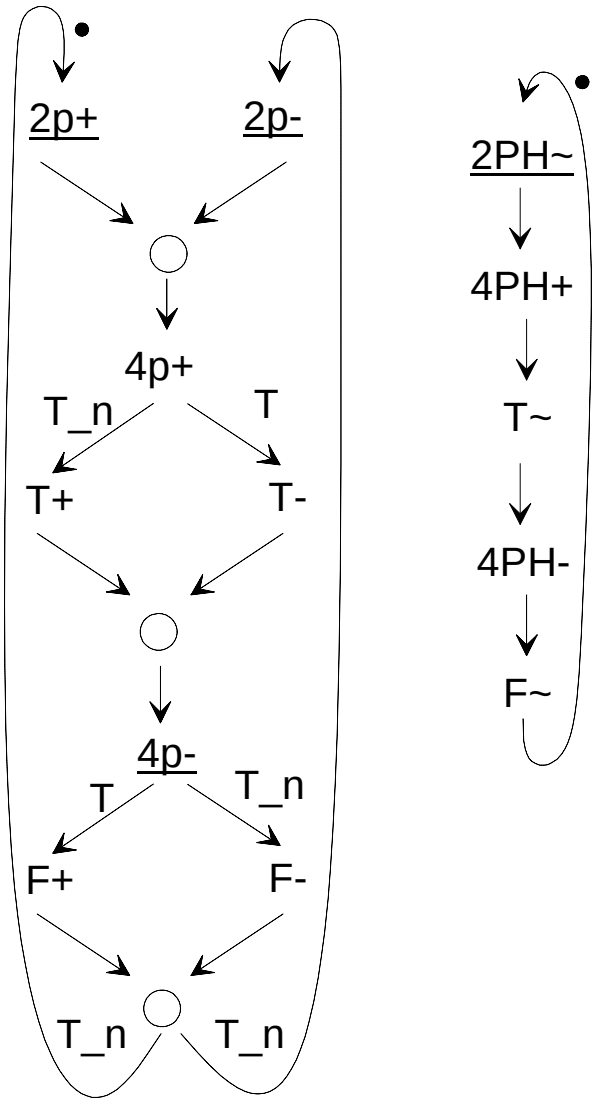

Abb. 8: Der STG zur Umwandlung von *Two-Phase* auf *Four-Phase* Signale.

Aus dieser Darstellung läßt sich nun eine Schaltungsrealisierung entwickeln. Durch Reduktion von Signalen ist es möglich, den STG in zwei STGs aufzuteilen, die jeweils XOR und ein Toggle Element darstellen. **Abb. 9** zeigt eine mögliche Realisierung der *Two-Phase* auf *Four-Phase* Umwandlung mit diesen Elemen-

ten. Die gestrichelte Linie von F zu 2p deutet an, dass sich hier noch weitere Logik im Steuerpfad befinden kann. Hiermit kann ein pegelgesteuertes Latch an ein ereignisgesteuertes *Two-Phase* Signal angeschlossen werden. Das Steuersignal des Speicherelements wird dabei an 4p angeschlossen. Die Signalleitung 4p kann durch eine komplexere Ablaufsteuerung ersetzt werden, die mit Signalpegeln und nicht mit Signalereignissen arbeitet, wie es in der Schaltungstechnik üblich ist. Obwohl das *Four-Phase* Signal doppelt so viele Wechsel benötigt, ist es sehr attraktiv, da in den meisten Fällen die Zeit zur Berechnung eines Signalwertes bei *Two-Phase* Signalen größer ist als die Übertragungsdauer der zusätzlichen Wechsel. Dies ergibt sich aus den *Noninput Choice* Bedingungen. Hinzu kommt eine kompaktere Realisierung der Speicherelemente bei *Four-Phase Signalen*. Schaltet man ein *Muller C-Element* vor den externen Ausgang 2p, kann die *Two-Phase Bundled Data Convention* mit den Signalen RI und AI sowie RO und AO in Verbindung mit einem pegelgesteuerten Latch arbeiten **[95FurbD]**$_{L.2}$. Die *Capture-Pass* Struktur des Registers ist dann nicht mehr notwendig[12].

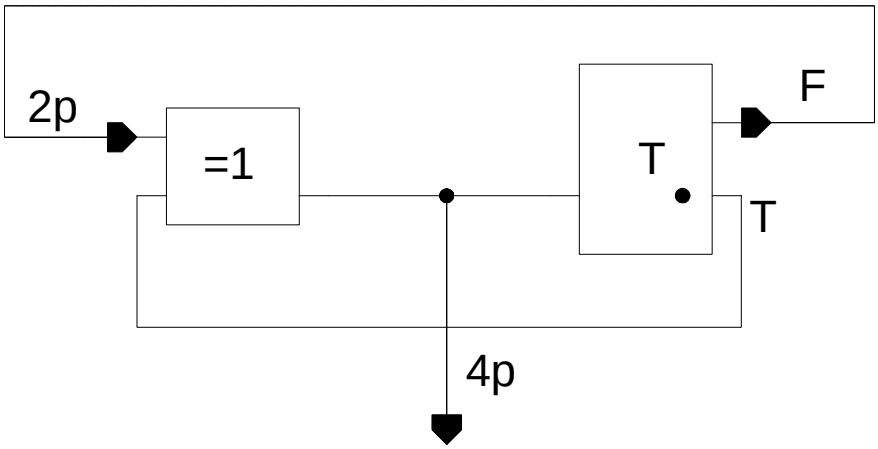

Abb. 9: Eine Schaltung zur Umwandlung von *Two-Phase* auf *Four-Phase* Signale.

[12] Bei dieser Realisierung wäre der Füllgrad einer *Micropipeline* maximal die Hälfte der Registerstufen, da immer jede zweite Registerstufe einen Operanden speichert, während die anderen Registerstufen geöffnet sind. In **[95FurbD, 95DayW]**$_{L.2}$ werden Methoden beschrieben, die den Füllgrad und die Verarbeitungsgeschwindigkeit erhöhen. Im nächsten Kapitel werden die *Micropipelines*, insbesondere das Füllen und Leeren der Registerstufen, näher beschrieben.

2.2 Schaltungskonzepte für *Self-timed Circuits*

Seitz prägte den Begriff der *Self-timed Circuits* und beschrieb ein *Handshake* Protokoll, die *Bundled Data Convention* **[80MeadC]**$_{L.2}$. Hierbei verwendete er ereignisgesteuerte (*Two-Phase Signalling*) oder zustandsgesteuerte (*Four-Phase Signalling*) Synchronisationsprotokolle mit Hilfe zweier Steuersignale. Als Synchronisationselement verwendete er das *Muller C-Element*. Dadurch kann bei diesem Baustein auf eine globale Taktversorgung verzichtet werden.

Es gibt zwei unterschiedliche Methoden zur Übertragung von Daten und Steuersignalen. Das *Dual-Rail* Verfahren arbeitet mit zwei Leitungen für jedes Datum. Durch die Redundanz ist es möglich, mit vier Signalzuständen die Synchronisation vorzunehmen. Sind die Signale der Leitungen invertiert, wird ein gültiges Datum übertragen. Sind beide Signalzustände auf einem Potential, ist die Berechnung des neuen Datums noch nicht abgeschlossen. Da es zwei Möglichkeiten mit gleichen Potentialen gibt, kann einer dieser Zustände zur Fehlererkennung oder zu Testzwecken genutzt werden. Schaltungen mit *Dual-Rail* Signalen werden seltener zur Kommunikation zwischen Modulen verwendet, da sie einen sehr hohen Verdrahtungsaufwand benötigen. Sie finden häufig Einsatz innerhalb von Modulen[13]. Eine Methode zur Synthese von *Speed-independent Circuits* mit *Dual-Rail Code* wird in **[88BertC]**$_{L.2}$ vorgestellt. Ein weiteres Beispiel für die Effektivität einer *Dual-Rail* Implementierung ist ein integrierter Speicher. Viele Speichertypen besitzen zwei interne Signalleitungen zum Lesen und Schreiben der Daten. Sie können somit mit kleinsten Spannungsdifferenzen das gespeicherte Signal verstärken und auslesen.

[13] Ein Beispiel ist die *Differential Cascade Voltage Switch Logic* (DCVSL) Technik **[95Hauc]**$_{L.2}$. Sie besitzt neben einem NMOS-Modul zur Berechnung der Schaltungsfunktion Precharge Transistoren, um die Stromversorgung zu unterbrechen und das *Completion Detection Signal* zurückzusetzen. Weitere Ergebnisse mit der *Dual-Rail* und *DCVSL* Technik sind in **[88LauRM, 91DeanWD, 92McAu, 93SaloK, 94Heer, 95Heer, 96MaezK]**$_{L.2}$ zu finden.

Eine weitere und häufig angewandte Methode trennt die Steuer- und Dateninformation der Schaltungsmodule. Sie wird als *Single-Rail* Verfahren bezeichnet. Die Steuersignale sind mit *Request* für das Signal zum Anlegen von neuen Daten aus der Quelle und *Acknowledge* für das Signal zur Rückantwort aus der Datensenke bezeichnet. Ein paralleler Datenbus überträgt die Operanden. Die Granularität der Schaltung ist dabei frei wählbar. Eine Trennung von Steuer- und Datenpfad ermöglicht es der synchronen Schaltungstechnik verwandte Entwicklungsmethoden zu verwenden.

2.2.1 Schaltungen mit Single-Rail Signalen

Schaltungen mit *Single-Rail* Signalen besitzen einen ähnlichen Aufbau wie getaktete Datenpfade. Der Unterschied liegt in der lokalen Synchronisation zwischen zwei Modulen anstelle eines globalen Takts. **Abb. 10** stellt das Prinzip der *Bundled Data Convention* nach Seitz dar **[80MeadC]**$_{L.2}$. Mit zwei Steuerleitungen wird die Datenübertragung zwischen dem Sender und dem Empfänger synchronisiert. Zur Verarbeitung von Daten wird zusätzlich ein Logikblock zwischen Sender und Empfänger in den Datenpfad integriert. Dabei entstehen Verzögerungen im Datenpfad, die auch bei der Generierung des *Request* Signals berücksichtigt werden müssen.

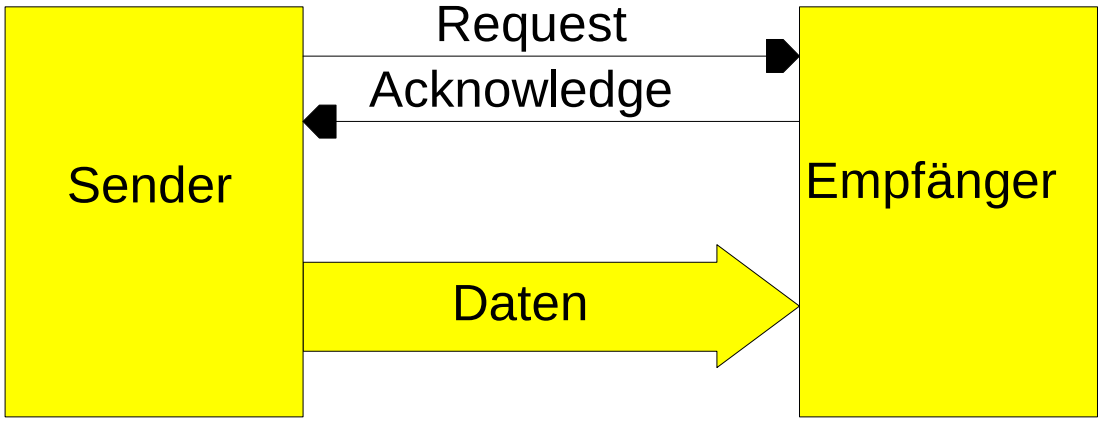

Abb. 10: Die Struktur eines Datenpfads mit einer *Bundled Data Convention* nach Seitz.

Zur Synchronisation werden zwei Protokolle von Seitz vorgeschlagen, die im folgenden erklärt werden. **Abb. 11** zeigt das Signalablaufdiagramm der *Two-Phase Bundled Data Convention*. Es ermöglicht eine ereignisgesteuerte Synchronisation zwischen zwei Schaltungsmodulen. Das bedeutet, dass der Signalwechsel und nicht der Signalpegel entscheidend für ein Synchronisationssignal ist. Da die Signalverarbeitung in integrierten Schaltungen aber in der Regel mit Pegeln arbeitet, ist die Initialisierung der Module entscheidend.

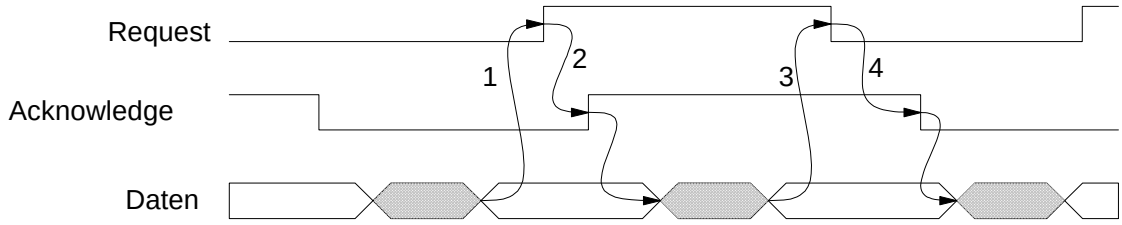

Abb. 11: Die *Two-Phase Bundled Data Convention*.

Aufbauend auf dieser Ereignissteuerung beschrieb Sutherland die *Micropipelines* als eine Art elastische Pipelinestruktur **[89Suth]**$_{L.2}$. Um eindeutige Signalpegel zu verwenden, wird weiterhin eine Konvention vorgeschlagen, die ein Rücksetzen der Pegel vorsieht. Hierdurch werden für die Datenübertragung vier Signalwechsel benötigt. **Abb. 12** zeigt das Signalablaufdiagramm der *Four-Phase Bundled Data Convention*. Es ermöglicht eine pegelgesteuerte Synchronisation zwischen zwei Schaltungsmodulen und vereinfacht die Signaldekodierung in Schaltungsmodulen auf Kosten von zwei zusätzlichen Signalwechseln. Die Steuersignale erreichen ihren Initialwert, bevor das nächste Datum verarbeitet wird. Dieses Protokoll kommt der Schaltungstechnologie entgegen.

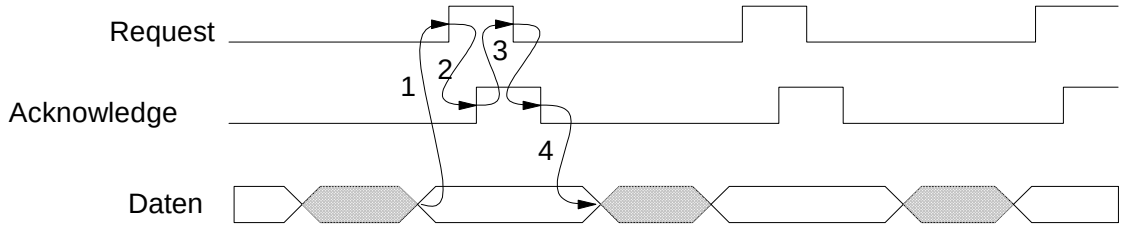

Abb. 12: Die *Four-Phase Bundled Data Convention*.

2.2.2 *Micropipelines*

Micropipelines basieren auf dem Prinzip der *Self-timed Circuits*. Dieses Unterkapitel behandelt die *Micropipelines*, die in **[89Suth]**$_{L.2}$ vorgestellt wurden. Die Organisation der zu verarbeitenden Logik im Datenpfad ist dabei sehr ähnlich zu einem synchronen Schaltungsentwurf. Insbesondere das Füllen und Leeren von *Micropipelines* wird hier näher untersucht, da dies zum Verständnis der in **Kapitel 4** vorgestellten testfreundlichen Entwurfskonzepte für asynchrone Schaltungen entscheidend beiträgt. **Abb. 13** zeigt das Schaltungsprinzip mit dem Steuerpfad als lokale Steuerung und dem Datenpfad zur Propagierung der Operatoren nach Sutherland.

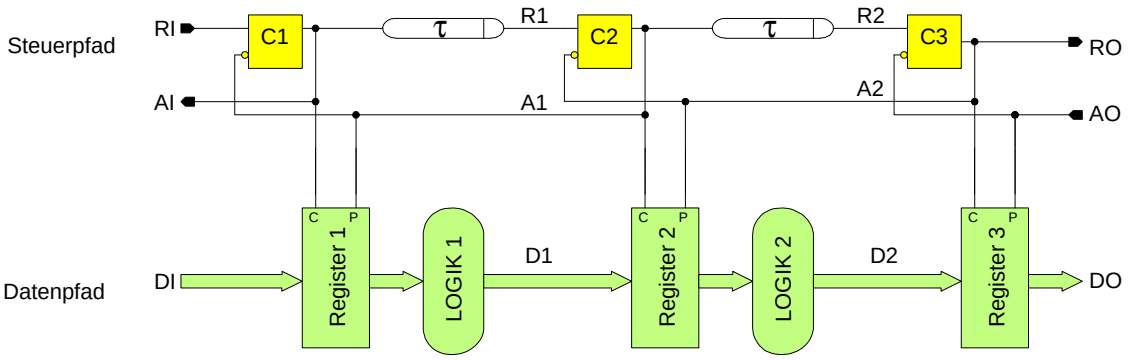

Abb. 13: Eine *Micropipeline* nach Sutherland.

Erfolgt an RI ein Signalwechsel, liegt ein gültiges Datum am Eingang DI an. Das *Muller C-Element* schaltet seinen Ausgang und speichert mit dem *Capture* Signal das Datum im Register. Mit dem C Signal wird auch ein *Acknowledge* an den Sender zurückgesendet, damit dieser ein neues Datum berechnen kann. Gleichzeitig liegt das Datum am Ausgang von Register 1 und damit an der Logik 1 an. Die Logik berechnet den neuen Operanden für die zweite Registerstufe. Ist das Ausgangssignal von C1 (R1) durch das Verzögerungselement gelaufen, kann nunmehr C2 schalten und seinerseits das Datum speichern. Es sendet dabei wieder das Signal A1 zurück, mit dem Register 1 seinen Ausgang auf die andere Registerzelle schaltet. Sutherland beschreibt eine Registerstruktur mit *Capture* und *Pass* Signalen, wie sie in **Abb. 14** dargestellt ist. Die beiden Register-Latches sind nicht hintereinander, sondern parallel angeordnet. Hieraus ergibt sich eine schnelle Datenverarbeitung der Signale. Durch das Öffnen der Schalter C und P kann dieses Register auch transparent geschaltet werden. Dies ist interessant, wenn man die Performance überprüfen oder den Datenpfad auf Verzögerungsfehler testen will. Ein paralleles Register einer *Micropipeline* besteht aus einer Vielzahl von Speicherzellen aus **Abb. 14**. Der Datenpfad wird aus Pipelinestufen von Registerbänken zusammengeschaltet. Für jede Registerstruktur wird dabei ein *Muller C-Element* benötigt. Es folgt nun eine grundlegende Betrachtung zur Initialisierung der Register einer Micropipeline.

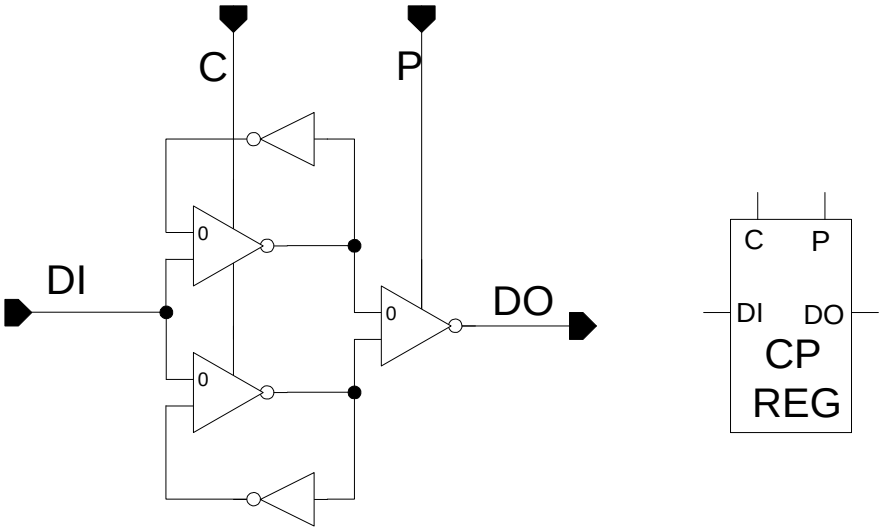

Abb. 14: Die Realisierung eines nichttransparenten Speichers nach Sutherland.

Ähnliche Methoden zur Entwicklung von Pipelinestrukturen mit Verzögerungselementen wurden in **[89WongMF]**$_{L.2}$ und **[88KomoTT]**$_{L.2}$ vorgestellt. Ein Syntheseprogramm verdeutlicht die Leistungsfähigkeit des Konzepts der *Micropipeline* **[89BrunS]**$_{L.2}$. Zur Optimierung der Pipelinestrukturen wurde der Steuerpfad auf seine Geschwindigkeit hin untersucht **[95DayW, 95FurbD]**$_{L.2}$. Hierbei wurde, ausgehend von einer Beschreibung mit STGs, das *Muller C-Element* durch geschwindigkeitsoptimierte Zellen ersetzt. Eine Pipeline für High Performance Anwendungen wurde in **[96YunBA]**$_{L.2}$ präsentiert. Weitere Implementierungen in Form von Mikroprozessoren unterstreichen die Leistungsfähigkeit der Entwurfsmethodik der *Micropipelines* **[93FurbDG, 94FurbDG]**$_{L.2}$[14]. Der Unterschied zu synchronen Schaltungen liegt hauptsächlich in der Synchronisation der Registerelemente. Hierdurch können viele Module aus einem synchronen Design übernommen sowie synchrones Design Know-How verwendet werden. Der asynchrone Amulet Prozessor ist bitkompatibel zu einem synchronen Prozessor der Firma ARM **[93Furb]**$_{L.2}$. Teile des Designs wurden dabei von dem synchronen Entwurf übernommen und in die *Micropipeline* integriert. Die Implementierung des asynchronen Mikroprozessors, Amulet, wurde in **[94Pave]**$_{L.2}$ beschrieben.

[14] Weitere Mikroprozessorrealisierungen sind in **[89MartBL, 90JacoB, 91MengBM, 94NanyUK, 94TierMB]**$_{L.2}$ zu finden.

- **Die Initialisierung der *Micropipeline* ist entscheidend für ihre Funktionstüchtigkeit. Diese Betrachtung erlangt für einen passiven testfreundlichen Entwurf mit Hilfe von asynchronen Signalen eine zentrale Bedeutung, da mit jedem parallelen Einlesen eines Datums in den seriellen Scanpfad auch dort die *Muller C-Elemente* initialisiert werden müssen.**

Das Konzept des asynchronen Scanpfads ist in **Kapitel 5.1** näher beschrieben. Vergleichbare Operationen, wie die im folgenden beschriebenen Lese- und Schreiboperationen einer *Mircropipeline*, müssen auch bei den testfreundlichen Entwurfshilfen zum Steuern und Beobachten eines Datums erfolgen. Das Prinzip der Initialisierung von Registerbänken durch die *Muller C-Elemente* bei der elastischen Pipeline von Sutherland ist in **Abb. 15** dargestellt. Links sind in jeder Zeile die Zustände einer vierstufigen *Micropipeline* skizziert. Die Registerstufen werden dabei geleert und wieder gefüllt. Rechts sind die Zwischenzustände der *Muller C-Elemente* beim Weiterleiten eines Signals durch die Pipelinestufen dargestellt. Im Zustand 1 ist die *Micropipeline* mit Daten gefüllt. Im Zustand 5 sind die Register transparent geschaltet. Aus dem aktuellen und vorangegangenen Zustand des *Muller C-Elements* läßt sich berechnen, ob das Register speichert oder transparent ist. Beim Initialisieren einer *Micropipeline* werden die *Muller C-Elemente* so gesetzt, dass sich die Register öffnen und alle Stufen bereit zum Aufnehmen von Daten sind. Dies ist in Zeile (**5**) zu sehen. Es gibt dabei zwei Möglichkeiten, die Speicherzelle zu initialisieren. Es werden alle *Muller C-Elemente* auf "0" oder auf "1" gesetzt. Wird ein Datum an den Eingang der Pipeline angelegt, erfolgt eine Weiterleitung durch die einzelnen Stufen. Dies wird auch als *Forward Propagation* bezeichnet, da der Zustand "1" des *Muller C-Elements* bis zur letzten Stufe vorwärts propagiert wird (**5a - 5c**). Ist eine *Micropipeline* gefüllt, sind alle aufeinander folgenden *Muller C-Elemente* abwechselnd auf "1" und "0" gesetzt (**1**). Wird ein Datum ausgelesen, übertragen die Registerstufen ihren Inhalt in die nächste freie Stufe. Damit rückt eine leere Registerstufe bis zum Anfang der *Micropipeline* vor. Dieses wird mit den Zwischenzuständen (**1a - 1c**) dargestellt. Diese Steuerung nennt man auch Backward Propagation, da die Signalwechsel von der letzten zur ersten Stufe durch den Steuerpfad laufen. Analog zum Leeren der *Micropipeline* gibt es auch zwei Möglichkeiten zum

Vollsetzen (**1**). Die *Muller C-Elemente* werden alternierend mit "0" und "1" initialisiert, wobei man einmal mit einer "1" oder mit einer "0" beginnt.

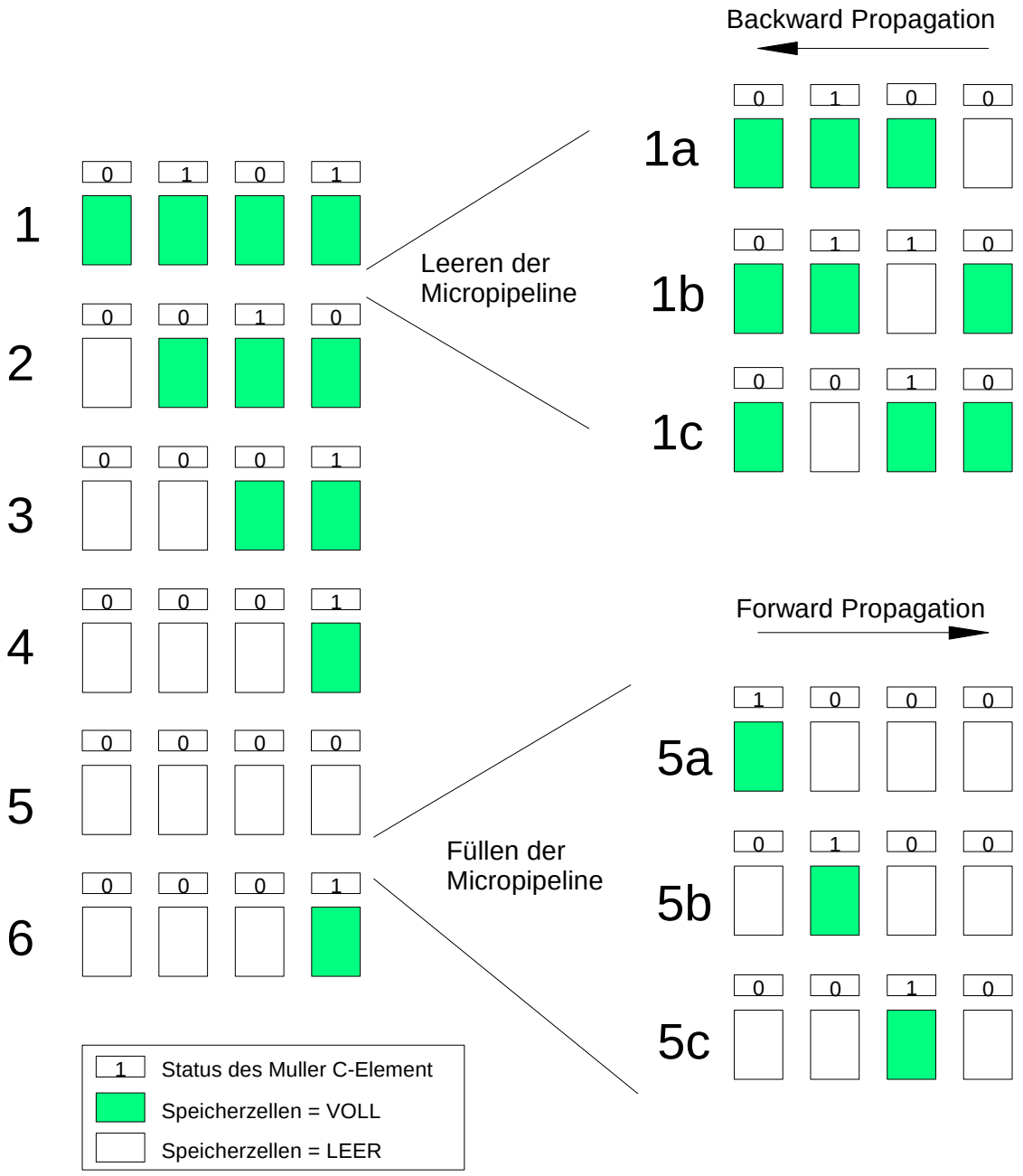

Abb. 15: Speicherbelegungen und Steuersignale beim Leeren und Füllen einer *Micropipeline*. Die Nummerierung stellt hierbei die zeitliche Abfolge dar.

- **Beim Initialisieren von *Micropipelines* muß darauf geachtet werden, dass zwischen den Übergängen von Pipelineketten keine Fehler auftreten. Dieses kann zu *Deadlocks* führen.**

Im Testbetrieb ist eine alternierende Initialisierung der *Muller C-Elemente* einer *Micropipeline* nützlich. Dies kann das Laden einer Registerbank[15] über einen seriellen Pfad ermöglichen. Sind zwei serielle *Micropipeline* Ketten über eine Umschaltung auch parallel miteinander verbunden, müssen die Zustände der *Muller C-Elemente* der parallelen Ketten richtig eingestellt werden[16]. Hierdurch ergibt sich eine voneinander abhängige Initialisierungsvorschrift der *Muller C-Elemente* der seriellen Registerelemente und der *Muller C-Elemente* der *Micropipeline* Ketten. Dies kann zu *Deadlocks* führen, wenn mehr als zwei Pipelinestufen zusammengesetzt werden.

Im folgenden wird näher auf das Verhalten bei einer Initialisierung von Steuer- und Datenpfad eingegangen, da es entscheidend zum Verständnis des testfreundlichen Entwurfs beiträgt. Dabei wird untersucht, wie eine Umsetzung von einer parallelen auf eine serielle Datenstruktur bei einer *Micropipeline* erfolgt. Die parallele Struktur wird dabei nicht dargestellt. Die Kommunikation erfolgt über die Zustände der *Muller C-Elemente*[17]. **Abb. 16** zeigt die Initialisierung der Steuerelemente, wenn eine *Micropipeline* geleert wird sowie den Speicherzustand der jeweiligen Registerzellen. Es wird dabei jeweils ein Flip-Flop gesteuert. Es entspricht der Anordnung eines Scanpfads, in dem seriell ein neues Datum eingelesen werden soll, um es an die zu testende Logik anzulegen. Die Pfeile deuten die Abhängigkeiten zwischen der Steuerung (Muller C-Elemente) und dem Datenpfad (Flip-Flop) an.

---

[15] Die serielle Anordnung der Registerzellen sind jeweils die Scanelemente eines Scanpfads.

[16] Dies ist der Fall, wenn die Logik zwischen den Registerketten mit Hilfe der Scanpfade getestet werden soll. Bei einem Scanpfad kann man alle Registerelemente seriell auslesen (Testbetrieb), aber auch bei Bussen parallel an die nachfolgende Stufen übergeben (Normal- oder Funktionsbetrieb).

[17] Siehe hierzu **Kapitel 5.1**.

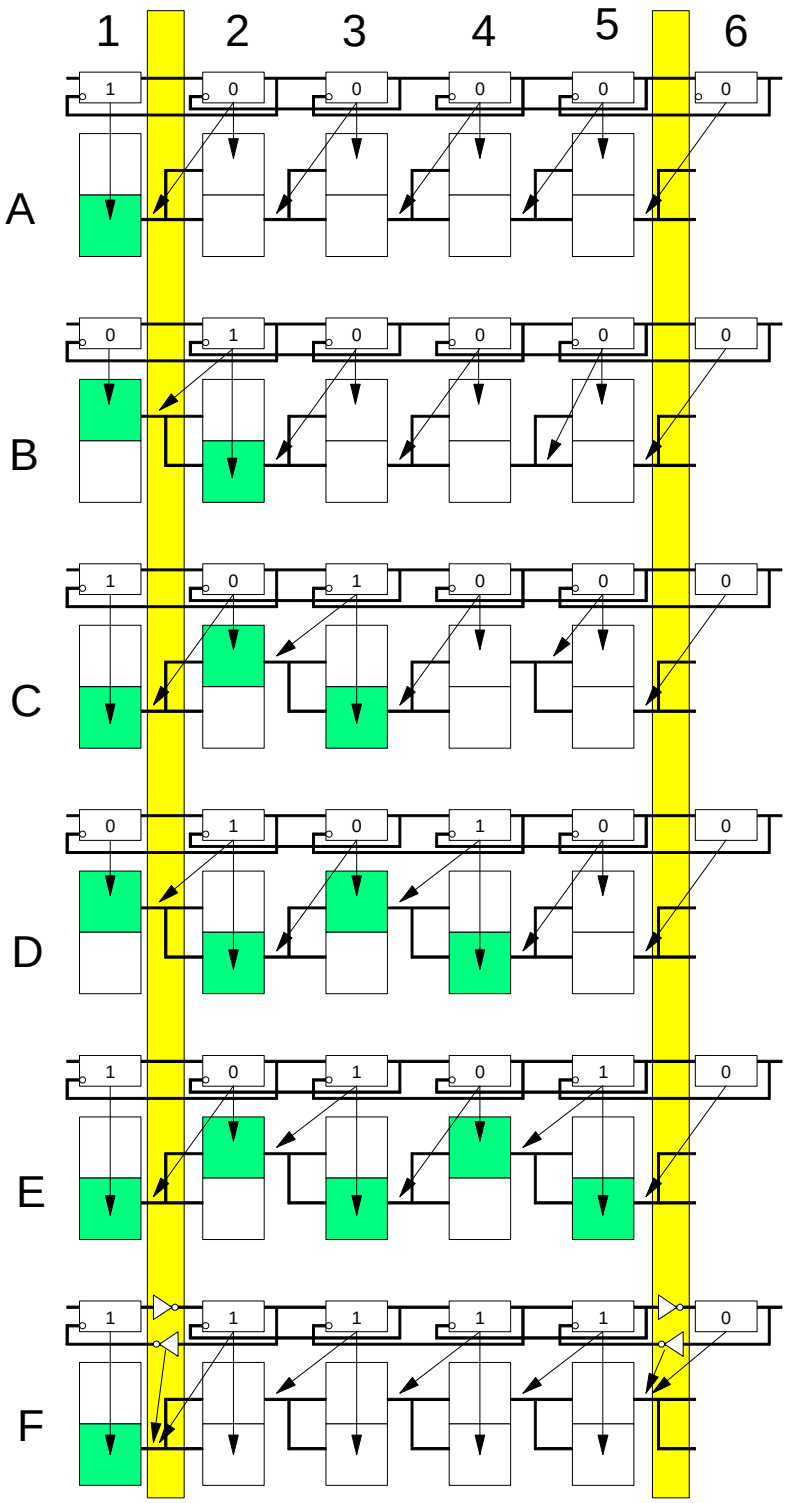

Abb. 16: Alle *Muller C-Elemente* der *Micropipeline* werden vor einem Lesevorgang auf ein Potential gesetzt.

Registerzelle 2A ist transparent geschaltet und wird das nächste gültige Datum aus der unteren Registerzelle von 1A übernehmen, in welcher ein gültiges Datum gespeichert ist. Je nach dem Pegel des *Muller C-Elements* wird ein Datum aus der unteren oder oberen Speicherzelle der vorhergehenden Stufe in eine der Zellen der aktuellen Stufe übertragen. Ist ein Datum gespeichert, wird dies mit einer grauen Fläche markiert. Die Balken zwischen der 1. und 2. Spalte sowie der 5. und 6. Spalte weisen auf den Beginn und das Ende des Scanpfads hin. Die 1. Stufe ist die letzte Stufe des Testsenders, und die 6. Stufe gehört zum Testempfänger. Bei einer Initialisierung sind somit nur die Stufen 2 bis 5 betroffen. Das Register wird schrittweise mit Daten gefüllt. In Zeile E ist dieser Vorgang abgeschlossen. Man erkennt, dass die Daten dort abwechselnd in der oberen und unteren Speicherzelle gepuffert werden. Aufeinander folgende Daten werden somit immer in unterschiedlichen Latches des *Capture-Pass* Registers gespeichert. Nach dem Initialisieren in Zeile A wird das Register bis zur Zeile E gefüllt und ist bereit zum Auslesen. Die Inverter in Zeile F innerhalb des vertikalen Balkens deuten den Abschluss eines Scanpfades an, bei dem es notwendig sein kann, Pegelumwandlungen durchzuführen. Diese ergeben sich aus den notwendigen Initialisierungswerten des Steuerpfades.

In einem Scanpfad würden die Daten parallel aus den Registerzellen ausgelesen. Dieses entspräche dem Übergang von Zeile E nach F. Hat der Scanpfad eine gerade Anzahl von Registerstufen, würde bei dem nächsten Testmuster die gleiche Zelle mit Daten gefüllt sein. Der parallele Datenpfad erwartet aber einen Operanden in der oberen Zelle. Der nächste Operand muß demnach so eingelesen werden, dass das Datum auf den gegenüberliedenden Latches zu finden ist. Um dies zu gewährleisten, erfolgt eine Initialisierung der *Muller C-Elemente* mit dem Pegel "1". Um die Signale von dem Sender und Empfänger richtig zu übertragen, werden Inverter eingesetzt. Besitzt der Scanpfad eine ungerade Anzahl von Registerstufen, müssen diese Pegelumwandlungen nicht erfolgen.

Ein weiteres Problem ergibt sich durch die Position der Registerinhalte, die abwechselnd in der oberen und unteren Zelle des Speicherelements gespeichert sind. Würde man die Daten parallel abgreifen, führt dies zu einem Fehler. Dieses Problem kann man dadurch lösen, dass die Schalter der Registerzellen so angeschlossen werden, dass sie alle Daten immer in der oberen oder der unteren Zelle speichern. Hierzu werden Speicherzellen verwendet, wie sie in **Abb. 41** zu sehen sind. Performanceverluste entstehen dabei nicht. Nach dem Einlese-

vorgang eines Scanregisters wird nun der Auslesevorgang behandelt. **Abb. 17** zeigt ein mit gültigen Daten gefülltes Register und den anschließenden Auslesevorgang. In einem Scanpfad würde dies der Speicherung einer Testantwort und dem anschließenden seriellen Auslesevorgang entsprechen. Das gültige Datum ist in abwechselnder Reihenfolge im Register gespeichert. Das Setzen der *Muller C-Elemente* erfolgt über einen weiteren Initialisierungsvorgang. Danach werden die Daten des Registers über die 6. Stufe ausgelesen. Dieses erfolgt analog zu dem Einlesevorgang. Ist der Zustand der Zeile E erreicht, kann ein neues Datum eingelesen werden. In Zeile F wird dieses neue Muster gespeichert. Vergleicht man Zeile A mit F, so ist auch hier zu erkennen, dass unterschiedliche Speicherplätze des *Capture-Pass* Registers besetzt sind.

Es müssen die Steuersignale nach der zweiten Initialisierung an den Ein- und Ausgängen invertiert werden, wenn die *Micropipeline* eine gerade Anzahl von Registerstufen besitzt. Zum Initialisieren der *Muller C-Elemente* können Reset- und Set-Eingänge verwendet werden. Hierbei gibt es vier Möglichkeiten: Das Leeren der Register durch Setzen der *Muller C-Elemente* auf "1" oder "0", das abwechselnde Setzen der *Muller C-Elemente* mit "1" und "0", beginnend mit einer "1" oder einer "0". Es müssen alternierend die Reset und Set Eingänge des *Muller C-Elements* gesetzt werden. Dadurch ergeben sich folgende hilfreiche Bezeichnungen für die Position der Speicherelemente: gerade Registerstufe (*even*) und ungerade Registerstufe (*odd*) der *Micropipeline*.

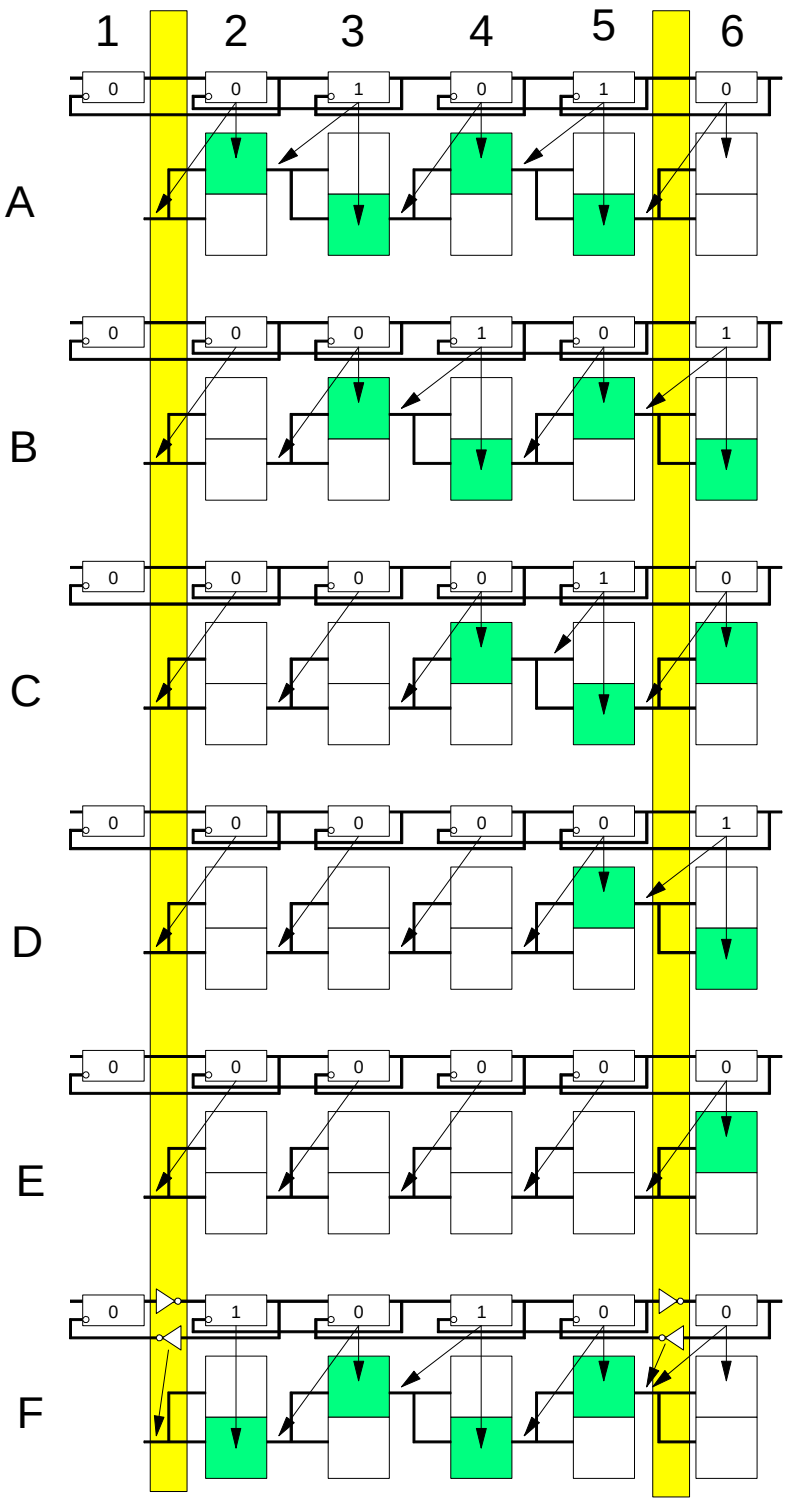

Abb. 17: Zum Auslesen werden alle benachbarten *Muller C-Elemente* mit unterschiedlichen Zuständen initialisiert.

Kapitel 3

Zum Test asynchroner Schaltungen

Die Entwicklung von effektiven Testverfahren und testfreundlichen Entwurfsmethoden ist Gegenstand intensiver Forschung der letzten 35 Jahre. Für den testfreundlichen Entwurf wurden passive Methoden entwickelt, wie Scanpfad **[73WillA]**$_{L.1}$ und LSSD **[77WillE, 77EichW]**$_{L.1}$, Testbusstandards auf Schaltungs- und Systemebene, wie die Boundary Scan Architektur oder MTM-Testbuscontroller. Darüber hinaus wurden aktive Testmethoden, wie die BILBO Technik **[79KonnMZ]**$_{L.1}$, entwickelt.

Der Test asynchroner Schaltungen ist ein vergleichsweise neues Gebiet des digitalen Testens. Für den breiten Einsatz von asynchronen Schaltungen ist neben leistungsfähigen Entwicklungsmethoden der ökonomische Test in einer Produktionsumgebung von entscheidender Bedeutung. Ein Test der lokalen asynchronen Steuerung benötigt neue Testmethodiken in einer synchronen Testerumgebung. Tester und deren Steuerung basieren auf einer globale Taktsteuerung. Dieses wichtige und häufig übersehene Instrumentarium des synchronen testfreundlichen Entwurfs, der globale Takt, fehlt bei asynchronen Schaltungen. Soll dieser für asynchrone Schaltungen immer wieder genannte Vorteil nicht durch synchrone Testhilfen eingebüßt werden, müssen testfreundliche Entwurfsmethoden entwickelt werden, die keinen Takt benötigen. Das Testen asynchroner Schaltungen, die mit einer hohen Performance arbeiten,

wird durch die Signale mit Bundled Data problematisch. Bei höchsten Qualitätsanforderungen an die Schaltung reicht das Haftfehlermodell nicht mehr aus, um eine akzeptable Testqualität zu erreichen. Auch der Vorteil von *Self-timed Circuits*, dass alle Module mit einer individuellen Geschwindigkeit arbeiten, stellt den Produktionstest vor neue Probleme. Die Eigenschaft, dass gültige Daten am Ein- und Ausgang durch *Handshake-Signale* übergeben werden, ist beim Testen in einer Produktionsumgebung nicht vorgesehen. Im Gegenteil, für die Testautomaten sind zur Steigerung der Testgeschwindigkeit synchrone Pipeliningstrukturen zur Adaptierung der Testmuster über die Pinelektronik an den Schaltkreis entwickelt. Testmuster werden dort über Zwischenspeicher auf dem Load-Board gepuffert, bevor sie an die Schaltung angelegt oder abgegriffen werden. Testhilfen innerhalb der asynchronen Schaltung, die solche Methoden unterstützen, können entscheidende Vorteile besitzen.

Erste Mikroprozessoren besassen in einigen Fällen asynchrones Verhalten. In **[62SeshF]**$_{L.3}$ wurde daher schon früh ein Konzept für einen asynchronen Produktionstest vorgestellt. In **[84BellV]**$_{L.3}$ wurde propagiert, dass asynchrones Signalverhalten bei einem Funktionstest berücksichtigt werden muß. Ein Test für einen asynchronen Mikroprozessor wurde in **[89MartBl]**$_{L.2}$ entwickelt. Da dieser Prozessor fast vollständig *Delay-insensitive* ist, besitzt er bei einer Fehlermodellierung mit Haftfehlern ein Verhalten, dass die Schaltung nach einer Fehlersensibilisierung stoppt. Auch für den Test eines DCC Moduls zur Fehlerkorrektur in einem DAT Recorder wurde die Modellierung von Haftfehlern an den Ausgängen der Elemente propagiert **[94Ronc]**$_{L.3}$. Der Nachweis, dass alle Fehler sich steuern lassen, wurde bei diesen Arbeiten nicht geführt.

Das Gebiet des Testens asynchroner Schaltungen teilt sich neben dem Test mit Hilfe von Testautomaten in die Bereiche Testvorbereitung und testfreundlicher Entwurf auf. Diese Bereiche werden im folgenden untersucht. Zum Abschluss erfolgt eine Diskussion zum Test asynchroner Schaltungen in einer Testerumgebung.

3.1 Die Testvorbereitung

Die Testvorbereitung dient der Generierung der Testmuster und der Bestimmung ihrer Qualität. Dazu wird eine Fehlermodellierung physikalischer Defekte und deren Simulation auf Gatterebene verwendet. Ziel der Fehlermodellierung ist die Adaption dieser physikalischen Defekte auf eine Abstraktionsebene, welche eine geringere Komplexität bei der Berechnung ihrer Auswirkungen besitzt. Somit muß immer zwischen Genauigkeit und Rechenaufwand bei der Fehlermodellierung abgewogen werden.

Ein Fehlermodell, in dem angenommen wird, dass sich das Fehlverhalten in primitiven Elementen als Haftfehler an deren Ein- und Ausgängen bemerkbar macht, hat sich als effektive Darstellung von möglichen Fehlerursachen erwiesen. Es wurde in **[59Eldr]**$_{L.1}$ vorgestellt. Die zunehmende Schaltungskomplexität und steigende Performance erzeugen die Notwendigkeit, das Zeitverhalten der Schaltung zu berücksichtigen **[80GaliCV, 87Maly]**$_{L.1}$. Dies hat zur Folge, dass zusätzliche nichtklassische Fehlermodelle angenommen werden müssen, um Prüfmuster zu ermitteln, die die Auswirkungen weiterer physikalischer Defekte abbilden[18]. **[83BarzR]**$_{L.1}$ stellte das Verzögerungsfehlermodell vor, bei dem eine zusätzliche Signalverzögerung angenommen wurde. Um die Fülle der möglichen Verzögerungen einzugrenzen, stellte **[86WaicL]**$_{L.1}$ ein Fehlermodell vor, bei dem Defekte angenommen werden, die eine zusätzliche Verzögerung über das Ende der Taktperiode haben. Bei asynchronen Schaltungen mit der *Bundled Data Convention* können solche Fehler auftreten. *Speed-independent Circuits* reagieren auf zusätzliche Verzögerungen lediglich mit einem später wechselnden Ausgangssignal; die Schaltungsgeschwindigkeit wird reduziert.

In **[86Koep, 87Maly]**$_{L.1}$ wird das Layout zur Reduzierung der Auswirkungen von Defekten berücksichtigt. In **[93AltM]**$_{L.1}$ wird die Modellierung nichtklassischer Fehler auf eine synchrone Schaltungsbibliothek vorgestellt, um die Qualität entwickelter Testmuster für sicherheitskritische Anwendungen durch aussage-

[18] Vergleiche hierzu auch die Untersuchungen von **[77HsieRV, 77StorB, 85Smit, 87MourM]**$_{L.1}$.

kräftigere Fehlermodelle zu verbessern. **[71PutzP]**$_{L.3}$ entwickelte ein Konzept, in dem der D-Algorithmus auf asynchrone Schaltungen angewendet wird. Rückkopplungen werden dabei aufgebrochen und in eine iterative, zyklische Schaltung umgewandelt. Bei einer großen Anzahl von Rückkopplungen führt dies zu einem starken Anwachsen der Schaltungsgröße und somit zu Problemen bei der Simulation und der Testmusterberechnung. Hinzu kommt, dass bei diesem Konzept von der klassischen Modellierung mit großen Schaltnetzen und Rückkopplungsnetzwerken ausgegangen wird.

Ein weiterer Vorschlag wandelt die Schaltungen in boolsche Gleichungen um **[74Chap]**$_{L.3}$. Die Entwicklung der Teststrategie erfolgte auf Basis des Huffmann-Modells oder einer iterativen Schaltung. In **[76BreuF]**$_{L.3}$ wurde, ausgehend von der Umwandlung der Schaltung in ein iteratives Model, ein Test für asynchrone Schaltungen vorgeschlagen, bei dem zusätzlich die Erkennung von *Hazards* berücksichtigt wurde. Auch in **[88ChenAK]**$_{L.3}$ wurde die automatische Testmustergenerierung synchroner und asynchroner Schaltungen untersucht. Eine Testmustergenerierung auf Switch-Level wurde in **[95EinsS]**$_{L.3}$ vorgestellt. Ein weiterer Ansatz zur automatischen Testmustergenerierung für Einzelhaftfehler an den Eingängen der Zellen (stuck-at-input) wurde in **[97RoigCP]**$_{L.3}$ präsentiert. Hierbei wurde die asynchrone Schaltung als synchroner endlicher Automat beschrieben. Die auf Basis einer synchronen Schaltungsbeschreibung generierten Testmuster können über einen synchronen Testautomaten an die asynchrone Schaltung angelegt werden. Die Schaltung befindet sich dabei aber in einer synchronen Betriebsart. Das asynchrone Verhalten wurde hierbei nicht überprüft.

Bei den oben beschriebenen Untersuchungen wurde davon ausgegangen, dass eine Einzelhaftfehlerannahme an den Ein- und Ausgängen der Elemente ausreicht. Es konnte aber gezeigt werden, dass zum Testen der Funktion des *Muller C-Elements* nicht allein Hafterfehlermodelle zu Grunde gelegt werden dürfen **[95BrzoR]**$_{L.3}$. Ausgehend von unterschiedlichen Modellierungen konnte das Fehlverhalten innerhalb der *Muller C-Elemente* nur in 57% - 85% der Fälle auf Haftfehler an den Ein- und Ausgängen abgebildet werden. Auch **[95SchoH]**$_{L.3}$ zeigte, dass sich Fehler auf Transistorebene eines pseudo-statischen *Muller C-Elements* auf Haftfehler an den Ein- und Ausgängen einer Realisierung mit Gattern abbilden lassen. Hierbei wurden Haftfehler an den Ein- und Ausgängen der Gatter angenommen, die zur Verhaltensbeschreibung des *Muller C-Elements*

eingesetzt wurden. Bezogen auf das Verhalten des *Muller C-Elements* ergab sich dabei in vielen Fällen eine Funktionskonvertierung. Fehler, die parametrische Auswirkungen besaßen, konnten mit einer erhöhten Stromaufnahme sensibilisiert werden.

Ein IDDQ Test wurde für einen Fehlerkorrektor eines Digital Compact Cassette (DCC) Spielers vorgeschlagen, der als Prototyp bei Philips vorliegt **[94Ronc]**$_{L.3}$. Um die Schaltung für den erhöhten Stromaufnahmetest zu steuern, wurde der Steuerpfad so verändert, dass ein synchrones Halten aller Signale möglich ist. Diese Funktion wird nur im Testbetrieb verwendet. Dies wurde durch eine globale Taktsteuerung und die Modifikation aller Steuerelemente erreicht. Durch die Einrichtung eines weiteren Zustands im Testbetrieb in einem ähnlichen asynchronen Design wurde die Beobachtbarkeit für die Strommessung erhöht **[96RonkB]**$_{L.3}$.

3.2 Der testfreundliche Entwurf asynchroner Schaltungen

Durch die Erhöhung der Steuer- und Beobachtbarkeit von Modulen in einer Schaltung durch sogenannte virtuelle Testpunkte wird die Testbarkeit erhöht. Es wird versucht, das sequentielle Verhalten der Schaltung auf eine kombinatorische Komplexität zu reduzieren. Dieses erreicht man mit Hilfe von Scanpfadtechniken. Die Register sind mit Hilfe eines zweiten Eingangs durch einen Prüfbus direkt steuerbar und über den Ausgang beobachtbar. Neben der einphasigen synchronen Scan-Technik wird auch die zweiphasige pegelgesteuerte LSSD Technik eingesetzt.

Ein erster Ansatz zum testfreundlichen Entwurf asynchroner Schaltungen wurde in **[84HearSD]**$_{L.3}$ beschrieben. Als Basis dienten synchrone Testmethoden für asynchrone Schaltungen. **[84Suss]**$_{L.3}$ stellte ein Konzept mit der LSSD-Technik vor. Auch in **[89LeenS, 93Leen]**$_{L.3}$ wurde die Einbettung von asynchronen Modulen in eine synchrone Schale während des Tests propagiert. Hierbei ging man auch von dem klassischen Fall aus, dass Rückkopplungen nur global auftreten werden. Dies steht im Widerspruch zu aktuellen asynchronen Synthesemethoden. Ein weiteres scanbasiertes, synchrones Testkonzept für synchrone und asynchrone Schaltungen, mit dem auch ein Verzögerungstest berücksichtigt wird, beschrieb **[89SakaHO]**$_{L.3}$. Die Register werden dabei durch Scanzellen ausgetauscht und in ein hierarchisches Konzept integriert. In **[91GuilSY]**$_{L.3}$ wurde ein Vorschlag zur Fehlermodellierung und Testmustergenerierung in Verbindung mit einem synchronen Scanpfad gemacht. Das Signal zur Beendigung der Funktionsberechnung, welches anzeigt, wann der Operand am Datenausgang gültig ist, diente dabei als *Capture Signal* für den Scanpfad. Hierdurch wurde ein asynchrones Schaltverhalten beim Test berücksichtigt. Ein weiteres synchrones Scandesign ist in **[93WeySF]**$_{L.3}$ zu finden. Die Realisierung eines partiellen Scanpfads für ein Fehlerkorrekturmodul in einem DCC Spieler, welches mit der Tangram-Syntheseumgebung erzeugt wurde, präsentierte **[94Ronc]**$_{L.3}$.

Auch diese Schaltung basiert auf dem Prinzip der *Self-timed Circuits*. Heuristiken zum testfreundlichen Schaltungsentwurf mit *Handshake* Signalen zur Integration eines synchronen Scanpfads und eines LSSD Designs stellte **[96Ronc]**$_{L.3}$ vor.

[91MartH]$_{L.3}$ entwickelte eine Art *Toggle* Test für asynchrone Schaltungen. Hierbei wurde davon ausgegangen, dass bestimmte Klassen von Schaltungen bei Haftfehlern entweder zu früh schalten oder alle Signalwechsel stoppen. Das heißt, dass keine weiteren Signalwechsel im Steuerpfad nach der Sensibilisierung erfolgen. Diese Art von Fehlern sollen nach Martin immer beobachtbar sein. Dies trifft bei der Betrachtung von Haftfehlern auch zu. Bei der Fehlermodellierung in einzelnen Modulen der Ereignissteuerung hat sich gezeigt, dass ein Fehler auf Transistorebene eine Funktionskonvertierung oder einen Parameterfehler auf Gatterebene verursachen kann. Dies läßt sich nicht zweifelsfrei darauf zurückführen, dass die Schaltung zu früh schaltet oder stoppt. Auch die Annahme, dass ein *Fail-stop* Verhalten sich am Ausgang der Schaltung bemerkbar macht, hängt von den entwickelten Testmustern ab, die das Fehlverhalten auch an die Ausgänge der Schaltung weiterleiten müssen.

Ein Scandesign für DCVSL-Technik, welches nur wenige zusätzliche Transistoren benötigt, wurde in einem Addierer plaziert **[93SaloK]**$_{L.3}$. Hierbei halten die Speicherelemente die Information dynamisch. Ein Ladungs- und damit Informationsverlust kann aber nicht ausgeschlossen werden. Des weiteren wurden die komplementären Signalleitungen, die zur Berechnung der *Dual-Rail* Signale nötig sind, beim Test nicht berücksichtigt und sind daher nur bedingt testbar[19]. Für die in der Veröffentlichung von **[88RoseMC]**$_{L.2}$ präsentierten *Q-Modules* wurde ein Testkonzept für das asynchrone Designkonzept vorgeschlagen, welches auf der synchronen LSSD Technik beruhte. Da die interne Steuerung mit Hilfe eines Takts erfolgt, kann das Modul intern getestet werden. Das Problem zum Test der asynchronen Interfaces bleibt aber bestehen. *Self-timed Circuits* mit *Dual-Rail Encoding* wurden als *Self-checking* für Einzelhaftfehler bezeichnet **[95DaviGY-]**$_{L.3}$. Der vierte Zustand der *Dual-Rail* Signale, der für den Operationsbetrieb nicht benötigt wird, kann in den Modulen für die Fehlererkennung verwendet werden. Sensibilisierte Fehler außerhalb der Module stoppen die Schaltung (Fail-stop).

---

[19] Es wurden unterschiedliche testfreundliche Entwurfskonzepte für Makromodule vorgeschlagen. Ein testfreundliches asynchrones systolisches Array wurde in **[86RanaLC]**$_{L.3}$ präsentiert. In **[89Cars, 90CarsB]**$_{L.3}$ wurde ein testfreundlicher asynchroner Zähler vorgestellt, in dem die *Toggle* Elemente scanfähig realisiert sind. Ein testfreundlich entworfener Blocksorter und Addierer wurde von **[96Petl]**$_{L.3}$ vorgestellt.

Ein testfreundlicher Entwurf für *Micropipelines*, in den ein testfreundlicher Entwurf der *Muller C-Elemente* integriert ist, wurde erstmalig in **[94KhocB]$_{L.3}$** vorgestellt. Die *Muller C-Elemente* werden dabei so verändert, dass sie sich über eine Taktsteuerung synchronisieren lassen. Die *Micropipeline* ist somit im Daten- und Steuerpfad in ein synchrones Design umgewandelt. In **[95PageKB]$_{L.3}$** wurden Modifikationen zum testfreundlichen Entwurf der einzelnen Steuerlogikmodule beschrieben. In **[95PetIF]$_{L.3}$** wurde ein Scanpfad für *Self-timed Circuits* entwickelt, der die Register einer *Micropipeline* mit Multiplexern erweitert. Zur Steuerung wird ein eigener Teststeuerpfad verwendet, der parallel zum Steuerpfad des Datenpfads liegt. Auch hier schaltet eine Multiplexersteuerung zwischen Operations- und Teststeuersignalen um. Während des Tests mußten die Steuersignale für die Daten auf einem bestimmten Potential gehalten werden. Darüber hinaus wurden aktive Testhilfen vorgestellt, die nach einem vergleichbaren Verfahren ablaufen **[96PetI]$_{L.3}$**. Es wurden Verfahren zum testfreundlichen Entwurf von *Muller C-Elementen* auf Transistorebene präsentiert, die ein besseres Fehlverhalten gegenüber *Shorts* und *Opens* besitzen sollen. Dabei wurde ein Scanpfad für *Muller C-Elemente* vorgeschlagen, um den Steuerpfad mit bestimmten Initialisierungsmustern zu laden. Durch die Multiplexer in den zeitkritischen Bereichen des Steuer- und Datenpfads erhöhen sich neben der Fläche aber auch die Signallaufzeiten erheblich[20].

In **[90AshaGD]$_{L.3}$** wurde eine Synthesemethode beschrieben, die einen testfreundlichen Entwurf sowie eine Redundanzerkennung unterstützt. Auf Basis eines *Implicit Signal Transistion Graphs* wurde es ermöglicht, redundante Zustände zu erkennen und die FSM zu optimieren. Für eine bestimmte Klasse von synthesefähigen asynchronen Schaltungen wurden in **[91BeerM]$_{L.3}$**. Aussagen über die Testbarkeit gewonnen und anhand von Interface-Schaltungen erläutert[21]. Zur Fehlermodellierung der Schaltungen, die mit Hilfe von STGs modelliert sind, wird dabei das Eingangshaftfehlermodell und das Pfadverzögerungsfehlermodell berücksichtigt. Die asynchrone Schaltung wird in kombinatorische

[20] Ein weiteres testfreundliches *Micropipeline* Design wurde in **[95GlorO]$_{L.3}$** diskutiert. Es sah eine Umschaltung in eine synchrone Testbetriebsart vor.

[21] Eine weitere Klasse von synthesefähigen asynchronen Schaltungen, die keine *Hazards* aufweisen, sind in **[91KeutLS, 92Lava, 94LavaKL, 95KeutLS]$_{L.3}$** zu finden.

Schaltungsmodule mit speichernden Elementen an den Ein- und Ausgängen umgewandelt.

3.3 Diskussion zum Stand der Technik

Im folgenden werden verschiedene Punkte diskutiert, um Anforderungen für einen testfreundlichen Entwurf asynchroner Schaltungen zu beschreiben, die in **Kapitel 4** und **Kapitel 5** bearbeitet werden.

In **[57Huff]**$_{L.2}$ wurde das Auftreten von *Hazards* als Fehlverhalten einer realen Schaltung bei der Abbildung ihrer gewünschten Funktion durch einen Designer bezeichnet. Hierbei wird zwischen statischen *Hazards*, bei dem das Signal kurzzeitig seinen gewünschten Wert wechselt, und dynamischen *Hazards* unterschieden. Im zweiten Fall erfährt das Ausgangssignal zusätzlich zu den erwarteten Signalwechseln noch einen weiteren, kurzen Impuls. Auch in **[59Unge]**$_{L.2}$ wurde das Problem von *Hazards* in asynchronen Schaltungen beschrieben, die durch Verzögerungen in realen Schaltungen entstehen. Hierbei wurde auch eine asynchrone Schaltung mit *Muller C-Elementen* und XOR Gattern untersucht. **[64YoelR, 65Eich]**$_{L.2}$ formulierten eine 3-wertige Logik, um Programme zur Berechnung und Simulation von *Hazards* zu erstellen. Ein dritter Wert wird dabei als unbekannter Zustand neben "0" und "1" eingeführt.

Als *Races* werden instabile Übergangsphasen eines oder mehrerer Signale bezeichnet, die in einem "Wettbewerb" mit einem Zustandswechsel stehen. Bei *Critical Races* kann es dabei zu einem falschen Folgezustand kommen. Eine Untersuchung hat gezeigt, dass man die Anzahl der Rückkopplungen zur Vermeidung von *Hazards* und Races minimieren kann **[89BrzoS]**$_{L.2}$. *Races* und *Hazards* können mit erhöhtem Schaltungsaufwand reduziert werden. Dies erfolgt in der Regel durch Redundanzen, die zu Problemen bei der automatischen Testmustergenerierung führen.

- **Die Behandlung von *Hazards* besitzt bei einem asynchronen testfreundlichen Schaltungsentwurf im Gegensatz zu synchronen Schaltungsentwürfen eine große Bedeutung,**

da der Takt als Instrument zu ihrer Vermeidung der Auswirkung fehlt.

Eine weitere asynchrone Signalübertragungsart bei *Bounded Delay* ist der *Burst Mode*, bei der sich immer ein Bündel von Signalen am Eingang ändern darf, und die Antwort mit einem lokalen Takt in dem Modul berechnet wird. Diese Schaltungsformen bergen ein großes Testproblem, da die Funktionstüchtigkeit durch Verzögerungsfehler invalidiert werden kann.

Bei unterschiedlichen Ansätzen zum Test asynchroner Schaltungen wird von ideal arbeitenden, primitiven Elementen ausgegangen, um Schaltungen mit den Eigenschaften *Delay-insensitive* oder *Speed-independent* zu erhalten. In **[92BrzoE]**$_{L.2}$ zeigte sich, dass *Muller C-Elemente* nicht die Eigenschaft *Delay-insensitivity* aufweisen. Der Nachteil, dass bei Verzweigungen unter ungünstigen Bedingungen *Hazards* entstehen können, wurde in **[92Berk]**$_{L.2}$ beschrieben. Es wurde dabei angenommen, dass diese sogenannten *Isochronic Forks* Äquipotentialflächen bilden. Auch dies ist eine stark idealisierte Darstellung physikalischer Realisierungen von Leitungssystemen, die im Zuge neuer Technologieschritte auf ihre Gültigkeit überprüft werden muß. Metastabile Zustände können durch die Verwendung realer asynchroner Schaltungsmodule auch im fehlerfreien Zustand nicht ausgeschlossen werden **[95Unge]**$_{L.2}$[22]. Digitale Fehlersimulatoren berücksichtigen zur Zeit den metastabilen Zustand nicht.

- **Es gilt zu untersuchen, ob Fehler auf Transistorebene auf das Ausgangssignal asynchroner Module ähnliche Auswirkungen haben, wie die metastabilen Zustände von Elementen in fehlerfreiem Zustand. Hierdurch kann sich ein dynamisches Testproblem ergeben, da durch Fehlermechanismen keine eindeutigen Binärsignale an den Ausgängen erzeugt werden. Verzögerungen oder falsche Folgezustände können das Resultat dieser Fehler sein.**

[22] Zur Charakterisierung von metastabilen Zuständen wurde versucht, das Zeitfenster, in dem die Übergänge stattfinden, analytisch und meßtechnisch zu bestimmen **[95RogiV, 96Fole]**$_{L.2}$.

Um die Ruhestrommessung in asynchronen Schaltungen zu ermöglichen, wurde ein Vorschlag beschrieben **[96RoncB]**$_{L.3}$. Er setzt dabei die lokale Steuerung während der IDDQ-Messung durch eine globale Steuerung außer Kraft.

- **Die Ruhestrommessung in asynchronen Schaltungen ist nur unter stark eingeschränkten Bedingungen möglich, da durch die lokale Steuerung ein Verharren aller Module der Schaltung in einem Zustand nicht vorgesehen ist.**

Bei einem Defekt innerhalb der Elemente geht man davon aus, dass ihr Fehlverhalten sich auf eine Haftfehlermodellierung an den Ein- und Ausgängen abbilden läßt. Der Beweggrund hierfür ist der Wunsch, die Komplexität auf ein Minimum zu reduzieren. Darüber hinaus ist dieses Fehlermodell in der synchronen Schaltungestechnik akzeptiert. In der CMOS Technik enthalten Gatter zwei Module, die gegensätzlich entweder den Ausgang treiben oder in den hochohmigen Zustand gehen. Eine Funktion realisiert hierbei über einen Pull-up Zweig das Aufladen des Ausgangs des Gatters. Der andere Funktionsblock am Pull-down Zweig ermöglicht das Entladen des Ausgangs. Die Funktionsblöcke sind so aufgebaut, dass das Treiben gegeneinander ausgeschlossen sein soll. Fehler in einem dieser Module kann zu einem Verhalten führen, dass nicht mit dem Haftfehlermodell abgebildet werden kann.

- **Asynchrone Schaltungen werden mit Methoden gefertigt, die aus der synchronen Schaltungstechnik bekannt sind. Somit sind auch die Defektmöglichkeiten bei beiden Schaltungstechniken vergleichbar.**

- **Die Einzelhaftfehlerannahme an den Ein- und Ausgängen der Standardzellen reicht nicht aus, um Fehlerursachen innerhalb der Elemente geeignet zu modellieren. Zusätzliche Auswirkungen, Verzögerungsfehler, metastabile Zu-**

stände und eine Funktionskonvertierung des Moduls, gilt es zu untersuchen[23].

Eine asynchrone Schaltung enthält Elemente zur lokalen Zustandssicherung, die speicherndes Verhalten besitzen. Mit ihnen kann der Steuerpfad der Schaltungen mit den Eigenschaften *Speed-independent* oder auch *Delay-insensitive* entwickelt werden **[94Pave]**$_{L.2}$. Das bedeutet, dass dieser Bereich der Schaltung eine vergleichbar hohe Fehlerabdeckung für Haftfehler besitzen wird, da sie die Fortschaltung der Signale verhindern und dies zu einem Anhalten der Schaltung führt. Das wäre vergleichbar mit einem Haftfehler in einem Teil des Taktnetzwerks einer synchronen Schaltung. Die Registerstufen würden nicht mehr mit neuen Daten geladen.

- **Asynchrone Schaltungen, die als *Speed-independent* oder *Delay-insensitive* entwickelt wurden, besitzen eine hohe Haftfehlerabdeckung. Nach der Sensibilisierung des Fehlers hält die Schaltung in diesem Zustand an. Der Haftfehler im Steuerpfad verhindert einen weiteren Signalwechsel, da er den STG verändert. Damit kann der Fehler entdeckt werden.**

Es hat sich gezeigt, dass die Verwendung von Haftfehlern einen Teil der möglichen Fehler von Standardzellen nicht entdecken kann **[95BrzoR, 95SchoH]**$_{L.3}$. Ob dies auch für andere Elemente der asynchronen Schaltungstechnik zutrifft, ist zu überprüfen. Die ausschließliche Verwendung von Haftfehlern an den Ein- und Ausgängen der Module konnte nicht alle möglichen Defekte in einer synchronen Standardzellenbibliothek modellieren **[93AltM]**$_{L.1}$.

- **Nichtklassische Fehlermodelle sind notwendig, um Defekte von Transistorebene auf Gatterebene zu transformieren. Die Fehlermodellierung muß um neue Modelle er-**

[23] Siehe hierzu auch **Kapitel 4**.

weitert werden. Eine anschließende Testmustergenerierung und Fehlersimulation ist in diesem Falle zur Quantifizierung der Testqualität notwendig. Zusätzlich werden testfreundliche Entwurfsmaßnahmen benötigt, die diese Muster an die Schaltungsteile anlegen.

Der Datenpfad einer *Micropipeline* kann mit konventionellen Methoden entwickelt werden. Zur Testmusterberechnung können daher Verfahren angewendet werden, die für kombinatorische und sequentielle Schaltungen entwickelt wurden. Eine prinzipiell andere Vorgehensweise ist bei Single-Rail Signalen nicht nötig. Der Einsatz von Modulen aus einem synchronen Design in dem asynchronen Amulet Mikroprozessor - er ist bit-kompatibel zum synchronen ARM Prozessor - bestätigt diese Aussage **[94FurbDG]**$_{L.2}$. Zu diesem Thema existieren zahlreiche Untersuchungen, die den Test mit taktbasierten Testhilfen lösen möchten. Ein Takt wird bei diesen Konzepten nur für die testfreundlichen Teile benötigt. Darüber hinaus wird die lokale Steuerung der Module durch den Test des Datenpfads, der mit synchronen testfreundlichen Entwurfsmethoden entwickelt wurde, nicht getestet.

- **Synchrone Testhilfen decken nicht den Test aller Bereiche einer asynchronen Schaltung ab. Insbesondere der Steuerpfad mit seiner *Bundled Data Convention* bleibt durch einen taktgesteuerten Test mangelhaft geprüft. Es ist ein testfreundlicher Entwurf notwendig, um Testmuster an alle Bereiche der Schaltung anlegen und beobachten zu können.**

Sich lokal auswirkende Prozeßfehler können durch die topologische Trennung von Steuer- und Datenpfad unterschiedliche Verzögerungen hervorrufen. Hieraus ergibt sich das Problem der Überprüfung der *Bundled Data Convention*. Sie verlangt, dass die Datensignale stets schneller propagiert werden als die Steuersignale. Verzögerungsfehler im Datenpfad wirken sich bei unveränderten Steuer-Signalwechseln sehr problematisch aus. Das Speichern eines falschen Datums

kann daher nicht ausgeschlossen werden. **Abb. 18** zeigt, dass die Verzögerung des Steuerpfads stets größer sein muß als die des Datenpfads. Die Verzögerung im Steuerpfad zwischen zwei Registerstufen muß daher so dimensioniert sein, dass die Signale im Steuerpfad stets langsamer als die Signale im Datenpfad sind. Ein asynchroner Scanpfad für *Micropipelines* wurde in **[96SchoK]**$_{L.3}$ dargelegt. Er ermöglicht einen gleichzeitgen Test des Datenpfads, des Steuerpfads und der *Bundled Data Convention*.

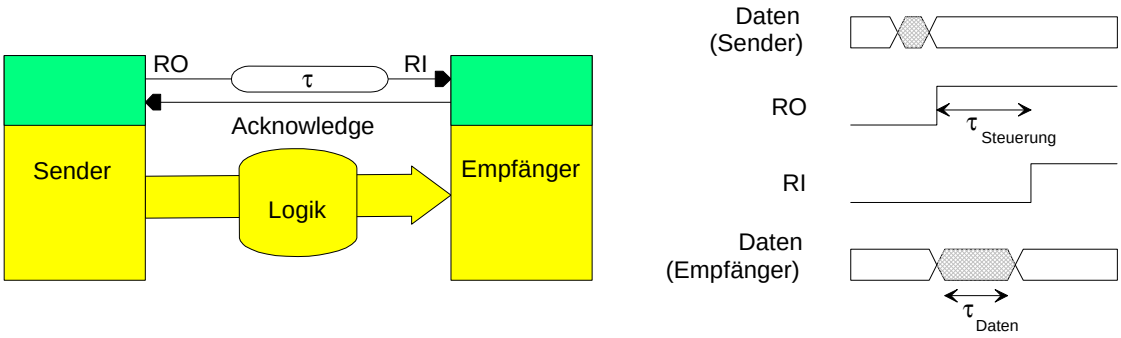

Abb. 18: Das Problem des Tests der *Bundled Data Convention*.

- **Bei *Self-timed Circuits* müssen neben der Datenpfadlogik und der Logik des Steuerpfads auch die *Bundled Data Convention* getestet werden. Ein Test des Steuer- und Datenpfads auf das funktionale und dynamische Fehlverhalten hin ist daher notwendig.**

Zur Einhaltung der *Bundled Data Convention* wird häufig ein Verzögerungselement in die *Request* Leitung eingebaut, das eine größere Verzögerung als der längste funktionale Pfad des aktuellen Datenpfadelements besitzt. Der Schaltungsentwickler muß dabei den längsten funktionalen Pfad bestimmen und von den strukturell möglichen Pfaden unterscheiden können[24].

Bei Schaltungen mit einer geringen Streuung in der Verzögerung der unterschiedlichen funktionalen Pfade führt dies zu guten Ergebnissen. Bei Schaltungen, die deutlich unterschiedliche Signalverzögerungen im Datenpfad besitzen, führt dies in den meisten Fällen der Datenübertragung zu einer schlechten Geschwindigkeitsausnutzung. Zur Erzeugung der Verzögerung gibt es neben

[24] Hierzu gibt es in der synchronen Schaltungstechnik Vorschläge **[95Mahl]**$_{L.1}$.

einem einfachen Verzögerungselement zwei weitere Möglichkeiten. Die erste Möglichkeit kann mit Hilfe einer Erweiterung der Funktion der Datenlogik erfolgen, die abhängig von der Bearbeitung der funktionalen Pfade ein zusätzliches Steuersignal setzt, welches mit dem *Request* und *Acknowledge* Signal des Steuerpfads verknüpft wird. Wenn der funktional längste Pfad von der Abarbeitung aller Elemente des Schaltungsblocks abhängig ist, kann ein solches Signal einfach erzeugt werden. Solche funktionalen Pfade sind bei ALUs mit *Ripple Carry Look Ahead* Struktur zu finden. Diese Methode wurde bei der Recheneinheit eines asynchronen Prozessors eingesetzt und hat Geschwindigkeitsvorteile zu dem vergleichbaren synchronen Prozessor erzielt **[93Gars]**$_{L.2}$.

- **Im Gegensatz zu einer synchronen Schaltung kann bei einer *Micropipeline* das Verhältnis Datenverzögerung zur Verzögerung der Synchronisationssteuerung nicht individuell nach der Produktion eingestellt werden. Die Berücksichtigung von Verzögerungsfehlermodellen ist für asynchrone Schaltungen mit der *Bundled Data Convention* daher von großer Bedeutung.**

Eine weitere Möglichkeit zur Erzeugung bietet sich bei Verwendung einer CMOS Technologie, die im Vergleich zur Stromaufnahme im Schaltpunkt eine geringe Ruhestromaufnahme der Gatter besitzt. Zur Zeit werden solche integrierten Stromaufnahmesensoren entwickelt, die den Ruhestrom einer Schaltung erkennen sollen[25]. Da zu vermuten ist, dass asynchrone Schaltungen in Bereichen mit geringer Stromaufnahme ein großes Einsatzgebiet haben werden, sind an diese Sensoren höchste Anforderungen auf die Stromempfindlichkeit und Robustheit gegen Schwankungen der Versorgungsspannung bei einer hohen Meßgeschwindigkeit gestellt. Als Beispiel kann hier ein asynchron arbeitender Dekoderblock zur Fehlerkorrektur angeführt werden, der seine Versorgungsspannung im Betrieb zwischen 2 und 5 Volt umschaltet und dadurch die Energieaufnahme stark reduziert **[94BerkBK]**$_{L.2}$. In den meisten Schaltungen mit der *Bundled Data Convention* werden Verzögerungselemente ohne direkte Abhängigkeiten zur

[25] Ruhestromsensoren für das *Completion Detection* Signal wurden in **[94DeanDH, 95GrasJ, 96GrasMK]**$_{L.2}$ vorgestellt.

Datenlogik verwendet. Diese Verzögerungselemente sind der kritische Bereich beim Test der Schaltung. Der Schaltungsentwickler wird versuchen, die zusätzliche zeitliche Sicherheit bei der Verzögerung des Steuersignals gegenüber dem Datenpfad zu minimieren. Hierdurch nimmt die Sensibilität der Schaltung auf zusätzliche Verzögerungen im Datenpfad zu. Da Steuer- und Datenpfad auch nicht dicht beieinander auf der Schaltung platziert sind, können lokale Technologieschwankungen eine stärkere Rolle spielen.

- **Zum Test der dynamischen Eigenschaften einer *Bundled Data Convention* muß ein Test mit dynamischen Eigenschaften in der Schaltung ablaufen. Interne aktive Testhilfen, die die dynamischen Testeigenschaften unterstützen, sind daher eine attraktive Alternative zu externen Testhilfen.**

- **Die Testmustergenerierung und die Testdatenkompression können mit der Datenverarbeitung der Schaltung erfolgen. Das asynchrone Verhalten muß dabei nicht mehr von externen Testhilfen gesteuert werden.**

Analoge Signale besitzen keinen Takt. Für die Umsetzung der analogen in digitale Signale eignen sich daher in großem Maße asynchrone Schaltungen, da eine externe Zeitquantisierung nicht mehr nötig ist. Bei der Entwicklung von asynchronen Schaltungen mit Hilfe der STG Technik wurde als Beispiel ein Analog-Digital Umsetzer verwendet, um die Leistungsfähigkeit der Methode zu verdeutlichen **[87Chua]**$_{L.2}$. Neben der Entwicklung von *Mixed-Signal* Bausteinen mit asynchronen Schaltungen besteht die Möglichkeit, auch testfreundliche Entwurfsmaßnahmen für analoge Schaltungen mit asynchronem Verhalten zu entwickeln.

- **Für den testfreundlichen Entwurf von Mixed-Signal Schaltungen könnte sich ein asynchroner Entwurfsstil eignen,**

der das Signal zur Beendigung der Datenkonversion von einem analogen in ein digitales Signal lokal erzeugt.

Es ergeben sich folgende Ziele für den asynchronen testfreundlichen Entwurf, die in den folgenden Kapiteln berücksichtigt werden müssen:

- **Eine Modellierung physikalischen Defekte soll auf Gatterebene abgebildet werden.**

- **Entwurf von neuen Standardzellen für den testfreundlichen Schaltungsentwurf.**

- **Der Test der Datenpfadelemente, des Steuerpfads und die Überprüfung der *Bundled Data Convention* sollten gleichzeitig erfolgen.**

- **Der Einfluß auf die Geschwindigkeit und die Größe der Schaltung sollte minimal sein.**

Kapitel 4

Asynchrone Standardzellen und deren Fehlermodellierung

Im folgenden Kapitel werden, neben einer kurzen Beschreibung der asynchronen Basiszellen, neue Zellen für den testfreundlichen Entwurf asynchroner Schaltungen vorgeschlagen. Die neuen Zellen können im Datenpfad oder im Steuerpfad eingesetzt werden. Darüber hinaus wird ein Verfahren zur Fehlermodellierung für asynchrone Standardzellen vorgestellt. Die Fehlermodellierung wird für primitive Elemente einer Bibliothek beschrieben. Anschließend erfolgt für komplexere Module, die sich aus primitiven Elementen zusammensetzen, eine Untersuchung. Fehlersimulatoren können mit der Beschreibung der Funktion und der Fehlermodellierung von primitiven Elementen die Funktion der Makromodule berechnen. Somit ist es möglich, innerhalb der Module Haftfehler zu modellieren und deren Auswirkungen an den Grenzen der Schaltung zu untersuchen. Hieraus ergibt sich ein Konzept zu einer hierarchischen Fehlermodellierung für asynchrone Schaltungen.

Es werden unterschiedliche Gatterrealisierungen für Bausteine zum testfreundlichen Entwurf asynchroner Schaltungen präsentiert. Basierend auf einer Transistorrealisierung der Zellen wird ein Konzept zur Fehlermodellierung asynchroner Schaltungsbibliotheken entwickelt, welches sich auch an der Fehlermodellierungsmethodik für synchrone und kombinatorische Bibliothekselemente von

[93AltM]$_{L.1}$ orientiert. In den primitiven Zellen werden Intragate-Fehler verwendet. Dabei werden einzelne Kurzschlüsse und offene Verbindungen in der Transistorrealisierung angenommen und ihre Auswirkungen auf eine Gatterrealisierung betrachtet. Es folgt die Abbildung der Fehlereigenschaften der Zellen auf Transistorebene auf ein Fehlermodell, welches für die Gatterebene anwendbar ist. Hierdurch ergibt sich eine Abstufung der möglichen Fehlerüberdeckung, die abhängig von der Leistungsfähigkeit des Simulators ist. Als einfachste Modellierung von Fehlern innerhalb der Standardzellen werden Einzelhaftfehler an den Ein- und Ausgängen der Schaltungen angenommen. Einige Defekte auf Transistorebene wirken sich als Konvertierung der Funktion der Standardzelle aus.

Es wurde in **[95BrzoR]**$_{L.3}$ gezeigt, dass unabhängig von der Implementierung eines *Muller C-Elements* immer die gleiche Anzahl an Testmustern ausreicht, um Haftfehler innerhalb der Modulbeschreibung zu sensibilisieren. Dies galt auch für die Beschreibung des *Muller C-Elements* mit boolschen Differenzen. Als weitere Möglichkeit zur Fehlermodellierung asynchroner Elemente wurde angenommen, dass eine Verhaltensbeschreibung der Transistorrealisierung auf Gatterebene möglich ist, die in einer Fehlersimulationsumgebung auf Haftfehler getestet wird. Bei einigen Fehlern, die insbesondere Module mit *Pass* Transistoren betreffen, führt keines der beiden genannten Modellierungsverfahren zum Erfolg, da sich diese Fehler dynamisch oder in einer erhöhten Stromaufnahme widerspiegeln, was auch als *IDDQ Fault* bezeichnet wird. Dynamische Fehler benötigen häufig ein Initialisierungsmuster, bevor sie mit einem Testmuster sensibilisiert werden **[93AltM]**$_{L.1}$. Sie werden daher auch als *Conditional Faults* bezeichnet.

- **Defekte, die einen erhöhten Ruhestrom verursachen, lassen sich mit Stromsensoren beobachten, wenn dieser Defekt in asynchronen Schaltungen mit einem Testmuster eingestellt ist.**

- **Einige Defekte verursachen keine funktionalen, sondern dynamische Fehler. Sie sind eventuell durch Testmusterpaare erkennbar, die einen falschen Folgezustand am Ausgang sichtbar machen.**

In primitiven Zellen werden Kurzschlüsse (*Shorts*) und offene Leitungen (*Stuck-Open*) als mögliche Auswirkungen von Defekten angenommen. **Abb. 19** zeigt die Fehler auf der Transistorebene, die als Grundlage zur Ermittlung des Fehlverhaltens auf Gatterebene dienen. Weiterhin werden die Kurzschlüsse zwischen beliebigen Knoten innerhalb der Gatter betrachtet. Die Wahrscheinlichkeit, dass zwischen zwei getrennt plazierten Modulen Fehler auftreten, die nicht durch die Haftfehlermodellierung an den Ein- und Ausgängen der Zellen abgedeckt werden, ist als vernachlässigbar anzusehen. Nach der Untersuchung von primitiven Elementen wird ein hierarchisches Konzept zur Fehlermodellierung in Makroelementen beschrieben. Weiterhin werden neue Bibliothekselemente für die testfreundlichen Entwurfsmaßnahmen in **Kapitel 4.2** vorgestellt. Die Bibliothekselemente für den testfreundlichen Entwurf werden auf ihre Testbarkeit untersucht[26].

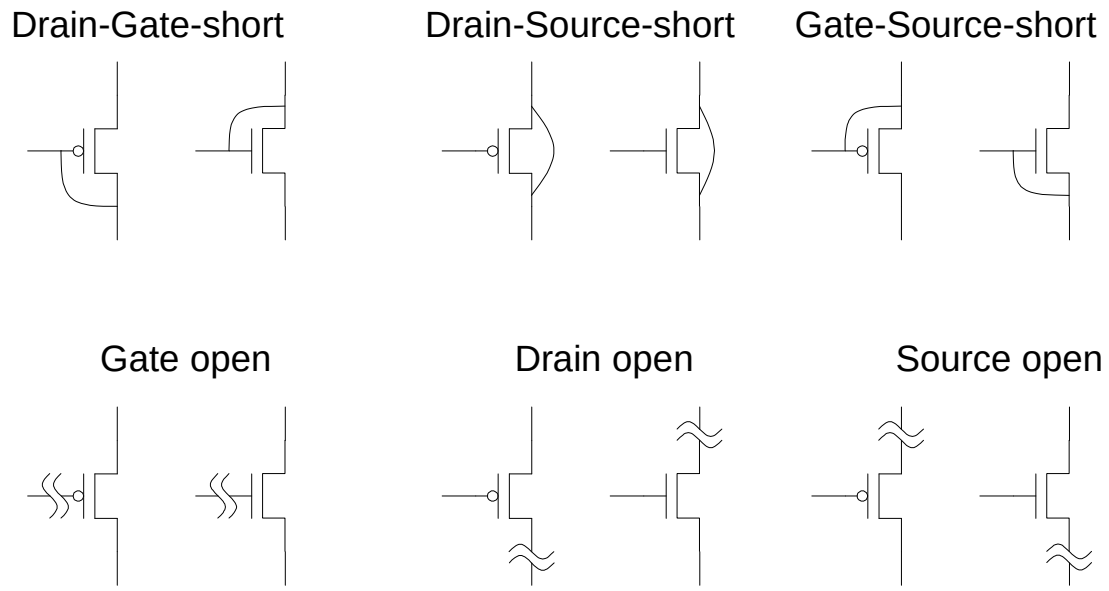

Abb. 19: Angenommene Fehler auf Transistorebene.

Ein *Stuck-Open* am Gate des Transistors wird als ein Pegel mit einem festen Potential (GND oder VDD) während des Tests angesehen. Kleinste Ladungsmengen, die sich auf dem *Gate* gesammelt haben, reichen, um den Transistor geöffnet oder geschlossen zu halten. Bei einem *Stuck-Open* an *Drain* und

[26] Es wird dabei vorausgesetzt, dass die Fehlermodellierung für primitive Elemente einer synchronen Bibliothek bereits auf Basis von **[93AltM]**$_{L.1}$ erfolgt ist.

Source wird angenommen, dass der Transistor keine Treiberfähigkeit mehr besitzt.

4.1 Standardzellen für den asynchronen Schaltungsentwurf

Asynchrone Schaltungen besitzen eine grundlegend andere Signalverarbeitung zur Synchronisation der speichernden Elemente als synchrone Schaltungen[27]. Um eine hohe Signalverarbeitungsgeschwindigkeit zu erhalten, ist es sinnvoll, Standardzellen zu entwickeln. Beispiele für solche Standardzellen wurden an der Universität in Manchester entwickelt und anhand von Schaltungen verifiziert **[94Pave]**$_{L.2}$[28]. Es fehlen jedoch Elemente zum asynchronen testfreundlichen Entwurf. Eine weitere Möglichkeit für eine Schaltungsbibliothek für *Micropipelines* ist die Abbildung der Gatter auf eine synchrone Bibliothek **[95FarnEL]**$_{L.2}$. Hierbei wird aber ein vielfaches der Fläche bei reduzierter Performance benötigt. Auch eine Simulation hat gezeigt, dass bei gleicher Architektur eine Schaltung auf Basis einer synchronen Bibliothek langsamer als eine Schaltung mit einer asynchronen Bibliothek **[96Ohla]**$_{L.3}$ ist.

Der Unterschied zu einer synchronen Standardzellenbibliothek ist im wesentlichen in zusätzlichen Elementen zur Synchronisierung des Steuerpfads zu sehen. Eine weitere Differenz entsteht durch die Anwendung asynchroner Schaltungen in Bereichen, die mit einer geringen Stromaufnahme arbeiten sollen. Hierfür werden die Zellen auf eine reduzierte Leistungsaufnahme hin optimiert.

- **Für synchrone Schaltungen, die auf einen geringen Leistungsverbrauch hin optimiert werden, sind asynchrone Schaltungselemente zur Stromabschaltung sehr hilfreich.**

Muller stellte ein Element zur Synchronisation zweier Signale vor **[59Mull]**$_{L.2}$. Es ermöglicht eine AND-Verknüpfung zwischen zwei Signalwechseln. Es wird auch

---

[27] Vergleiche hierzu **Kapitel 2**.

[28] In dieser Bibliothek wurden Vorschläge für synchrone testfreundliche Entwurfsmaßnahmen bei asynchronen Elementen vorgestellt, die im Department of Computer Science an der University of Manchester entwickelt wurden **[96Petl]**$_{L.2}$.

als Majoritätselement bezeichnet **[65Mill]**$_{L.2}$. *Multi-input Muller C-Elemente* wurden auf Basis einer synchronen Standardzellbibliothek, die eine geringe Pfadverzögerung besitzen **[93WuuV]**$_{L.2}$, vorgestellt. Verschiedene Untersuchungen befassen sich mit der Implementierung von Elementen zur Ablaufsteuerung sowie mit Registerzellen, die auf Geschwindigkeit **[94Roin]**$_{L.2}$ und geringe Stromaufnahme **[94HossWA]**$_{L.2}$ hin optimiert sind. Weiterhin werden einige Basismodule für die Steuerung und Datenspeicherung aufgezeigt und untersucht[29]. Eine Fehlermodellierung kann bei diesen Zellen, in denen sich häufig *Pass* Transistoren befinden, ebenso wie bei einer synchronen Schaltungsbibliothek erfolgen.

- **Durch eine *Pass* Transistorstruktur können nicht alle Fehler der Transistorebene auf Haftfehler der Gatterebene abgebildet werden. P und N Kanal Transistoren sind parallel angeordnet und bilden somit rekonvergierende Pfade.**

Wenn einer der beiden *Pass* Transistoren einen Fehler in der Art aufweist, dass er immer sperrt, wird nur noch ein Signalpegel mit dem richtigen Potential durchgeschaltet. Der andere Pegel wird mit einem um die Schwellenspannung reduzierten Wert weitergeleitet. Dies erzeugt ein verändertes Übergangsverhalten, wenn die nachfolgende Stufe den verringerten Signalpegel immer noch dem richtigen digitalen Wert zuordnet. Befindet sich die Ausgangsspannung im Schaltbereich des nachfolgenden Transistors, können kleinste Spannungsschwankungen einen falschen Logikwert verursachen. Ist einer der *Pass* Transistoren immer durchgeschaltet, kann der Ausgang nicht mehr hochohmig geschaltet werden. Es kommt dann zu Kurzschlüssen bei Bussen mit anderen treibenden Transistoren, wodurch eine erhöhte Ruhestromaufnahme und ein Verzögerungsfehler verursacht werden.

[29] Die primitiven Zellen basieren in den meisten Fällen auf der von **[89Suth]**$_{L.2}$ beschriebenen Funktion. Bei den Realisierungen auf Transistorebene wurden Strukturen verwendet, wie sie häufig in der Literatur zu finden sind.

4.1.1 Elemente zur Ereignissteuerung

Asynchrone Schaltungen besitzen eine Vielzahl von Modulen, die zur Synchronisation zwischen Schaltungsteilen dienen. Eine Auswahl der Module wird im folgenden behandelt.

Abb. 20 zeigt die kompakte Realisierung eines XORs. Es wird für die OR-Verknüpfung von Signalwechseln verwendet. Der Ausgang **C** des XORs wechselt sein Signal immer, wenn entweder **A** oder **B** das Signal wechselt. Das XOR wird häufig eingesetzt, um ein *Two-Phase* Signal zu einem *Four-Phase* Signal umzuwandeln[30].

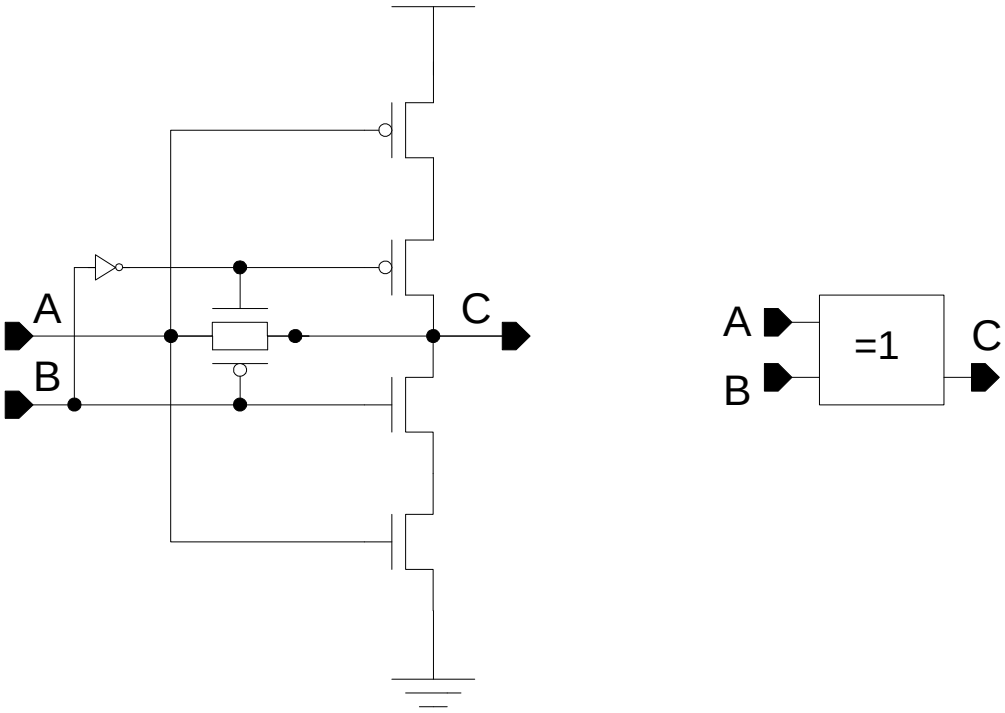

Abb. 20: Eine kompakte Realisierung des XORs mit einem *Pass* Transistor.

Der *Pass* Transistor bereitet Schwierigkeiten bei der Fehlermodellierung auf Gatterebene, da das Haftfehlermodell nicht ausreicht. Schaltet einer der *Pass* Transistoren des XORs immer, kann der Eingang **A** direkt über den *Pass* Transistor und über den Tristate-Inverter auf den Ausgang geschaltet werden. Dies führt zu einem erhöhten Ruhestrom vom Eingang **A** über den Tristate-Inverter zu

[30] Siehe hierzu auch **Abb. 9**.

VDD oder GND. Hier läßt sich das Ausgangsverhalten nicht mehr mit binären Werten beschreiben. Die genauen Treiberleistungen der Transistoren müssen bei der Berechnung des Ausgangssignals berücksichtigt werden.

Eine Untersuchung zeigte **[96Rick]**$_{L.3}$, dass sich nur 50% der *Shorts* auf Transistorebene des XORs auf Gatterebene als Haftfehler an den Ein- und Ausgängen abbilden lassen[31]. Weitere 38% können durch Haftfehler innerhalb eines äquivalenten Schaltungsmoduls erfasst werden. Es setzt sich aus AND und OR Gattern zusammen, die aus den Mintermen des XORs gewonnen wurden. 11% der *Shorts* auf Transistorebene verursachten eine erhöhte Ruhestromaufnahme. Bemerkenswert bei offenen Anschlüssen der Transistoren (*Stuck-Open*) ist, dass 66% der Fehler auf Gatterebene sich nur mit einem Initialisierungsmuster einstellen lassen. Nur 8% der Fehler auf Transistorebene sind als Haftfehler an den Ein- und Ausgängen des XORs auf Gatterebene modellierbar.

Abb. 21 zeigt eine kompakte Form des *Muller C-Elements* für eine pseudostatische Realisierung **[89Suth]**$_{L.2}$. Sie wird häufig gewählt, da sie mit wenigen Transistoren auskommt, jedoch die Ladung durch das speichernde Verhalten des rückgekoppelten Inverters im Gegensatz zu einer Kapazität nicht verliert. Bei Schaltungen für eine geringe Stromaufnahme wird diese Realisierung vermieden, da beim Umladen des Speichers für kurze Zeit ein Querstrom fließt. Der Rückkopplungsinverter muß dabei so dimensioniert sein, dass er eine geringere Treiberleistung als die Eingangsstufe besitzt. Dieses vergrößert die Fläche der treibenden Transistoren am Eingang. Um dies zu umgehen, wird der rückkoppelnde Inverter mit einem hohen Widerstand versehen oder im Umschaltvorgang hochohmig geschaltet.

[31] Hierbei wurden die dargestellten Transistormodelle analytisch untersucht. Die Ergebnisse wurden mit SPICE Simulationen überprüft, wenn kein eindeutiges Signalverhalten an den Ausgängen ermittelt werden konnte.

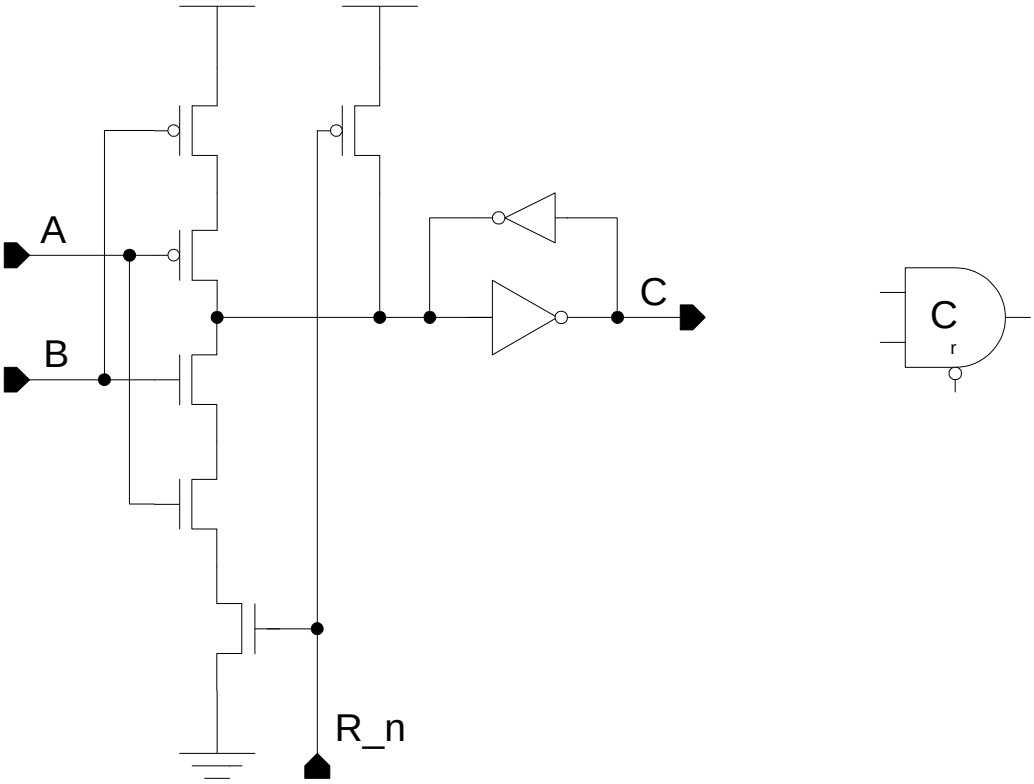

Abb. 21: Das *Muller C-Element* auf Transitorebene in pseudostatischer Realisierung.

Bei der Fehlermodellierung zeigte sich in **[96Rick]**$_{L.3}$ ein ähnliches Bild für das *Muller C-Element*, wie dies auch schon beim XOR zu sehen war. 50% der Transistorfehler lassen sich als Haftfehler auf Gatterebene an den Ein- und Ausgängen modellieren. Man kann für das Transistormodell auf Gatterebene boolsche Differenzen verwenden, die man aus der Minterm-Darstellung der Funktionstabelle gewonnen hat. Bei einer Fehlermodellierung können mit den Ein- und Ausgängen dieser AND und OR Gatter weitere 27% der Fehler modelliert werden. Weitere 13% der Fehler benötigen ein Initialisierungsmuster, um den Fehler mit dem folgenden Testmuster am Ausgang des Gatters zu beobachten. In 88% der *Shorts* konnte ein erhöhter Ruhestrom verzeichnet werden. Diese hohe Abdeckung durch das IDDQ Fehlermodell ist aus der Rückkopplung des Inverters zu sehen, der bei bestimmten Signalbelegungen im Fehlerfall den erhöhten Ruhestrom auslöst.

Mutual Exclusion Elemente (MUTEX) dienen zum gegenseitigen Ausschließen zweier Signalwechsel **[85Mart]**$_{L.2}$. Ein Signalwechsel von **R2** wird solange nicht an **A2** weitergeleitet, bis der Signalwechsel **R1** über **A1** abgearbeitet ist. Dadurch können Signalwechsel solange verzögert werden, bis eine geordnete Weiterver-

arbeitung erfolgt. **Abb. 22** zeigt eine Realisierung des *Mutual Exclusion* Elements. Es ist eine Kombination aus einem Speicher auf der linken Seite und zwei Invertern, deren VDD-Pfade mit den Ausgängen des Speicherelements verbunden sind. Der Eingang der Inverter wird an den jeweils invertierten Ausgang des Speichers angeschlossen.

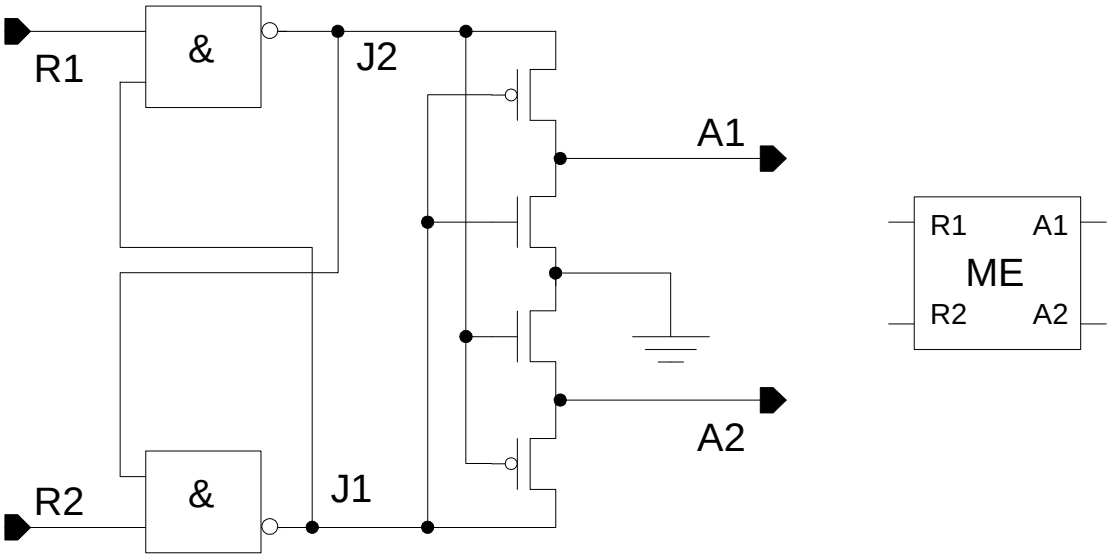

Abb. 22: Eine Realisierung des *Mutual Exclusion* Elements.

Das *Mutual Exclusion* Element ist eine Kombination von Gatter- und Transistorebene. Auf der linken Seite befindet sich ein Speicherelement. Die Ausgänge des Speicherelements werden als VDD Anschlüsse für zwei Inverter verwendet. Untersuchungen haben ergeben, dass bei gleichzeitigem Wechsel der Eingangssignale für einen kurzen Zeitraum ein metastabiler Zustand mit der Folge einer erhöhten Stromaufnahme auftreten kann **[94BrzoS]**$_{L.2}$. Nach kurzer Zeit wechselt dieser instabile in einen stabilen Zustand, bei dem die Ausgangssignale gegensätzlich sind. Der metastabile Zustand kann in Logiksimulatoren nicht berücksichtigt werden, da er für kurze Zeit eine Halbierung der Ausgangsspannung verursacht. Zur präzisen Modellierung müßte ein neuer Signalwert, VDD/2, für den Simulator definiert werden. Daraus ergibt sich ein zusätzlicher Pegel, dessen Signalverhalten für alle Gatter bestimmt werden muß. Neben den Problemen bei der Logiksimulation eines metastabilen Zustands kann sich auch im Fehlerfall ein solcher Zustand einstellen, wenn zwei Fehler im Falle eines Defekts ein Widerstandsnetzwerk zwischen VDD und GND verursachen.

Für die Fehlermodellierung in einem auf Logik basierenden Simulator müßte ein zusätzlicher metastabiler Signalwert berücksichtigt werden. Eine Testmusterberechnung für erhöhte Stromaufnahmefehler kann diese Defekte durch den Querstrom im Ruhezustand detektieren. Zur Fehlermodellierung werden die Gatter und Transistoren getrennt behandelt. *Shorts*, die an den Transistoren angenommen sind, halten bei 17% der Fehler die Ausgänge auf festem Potential. 33% sind Haftfehler an den internen Knoten **J1** und **J2**. 50% der *Shorts* und 100% der *Stuck-Opens* lassen sich nach einem Initialisierungsmuster mit dem folgenden Muster an den Ausgängen beobachten[32].

Mit Hilfe eines Latches ist es möglich, Signalwechsel im Steuerpfad zu speichern, sofern eine synchrone Schaltungsbibliothek verwendet wird. Ein Latch mit hochohmig schaltbaren Invertern ist in **Abb. 31** zu sehen. Der vorhergehende Signalwechsel wird so lange gespeichert, bis dieser von der nachfolgenden Stufe verarbeitet ist. Danach öffnet sich der Speicher und der nächste Signalwechsel kann abgearbeitet werden. Bei der *Bundled Data Convention* kann ein solches Element zum Halten von Signalwechseln eingesetzt werden, um kein Fehlverhalten in der Steuerlogik zu erhalten. Eine weitere Realisierungsform, die in der asynchronen Schaltungstechnik häufig angewendet wird, ist in **Abb. 23** dargestellt. Durch die individuelle Steuerung der Pfade nach VDD mit dem Transistor am Eingang **A** bzw. nach GND mit dem Transistor am Eingang **D**, ist es möglich, zwischen dem positiven und dem negativen Signalwechsel zu unterscheiden[33].

Die Fehlermodellierung ist hier analog zum pseudostatischen *Muller C-Element* zu sehen, da dieses Element die gleiche Anordnung der Transistoren besitzt. Dieses erkennt man beim Kurzschließen der Steuereingänge A und D. Es ist somit eine modifizierte Form des *Muller C-Elements* und wird häufig aus Performancegründen eingesetzt. Bei mehreren Steuergrößen zum Steuern von Signalwechseln können auch komplexe Funktionen integriert werden. Solche Module werden verwendet, um komplexe Schaltvorgänge, die mit Hilfe der Signaltheorie optimiert wurden, auf eine Schaltungsrealisierung zu übertragen.

---

[32] Für die Gatter wird die Fehlermodellierung für AND und OR angenommen.

[33] Der Pull-up und der Pull-down Zweig können dabei individuell mit mehreren Signalen verknüpft werden, wie es im Fall des *Muller C-Elements* getan wurde.

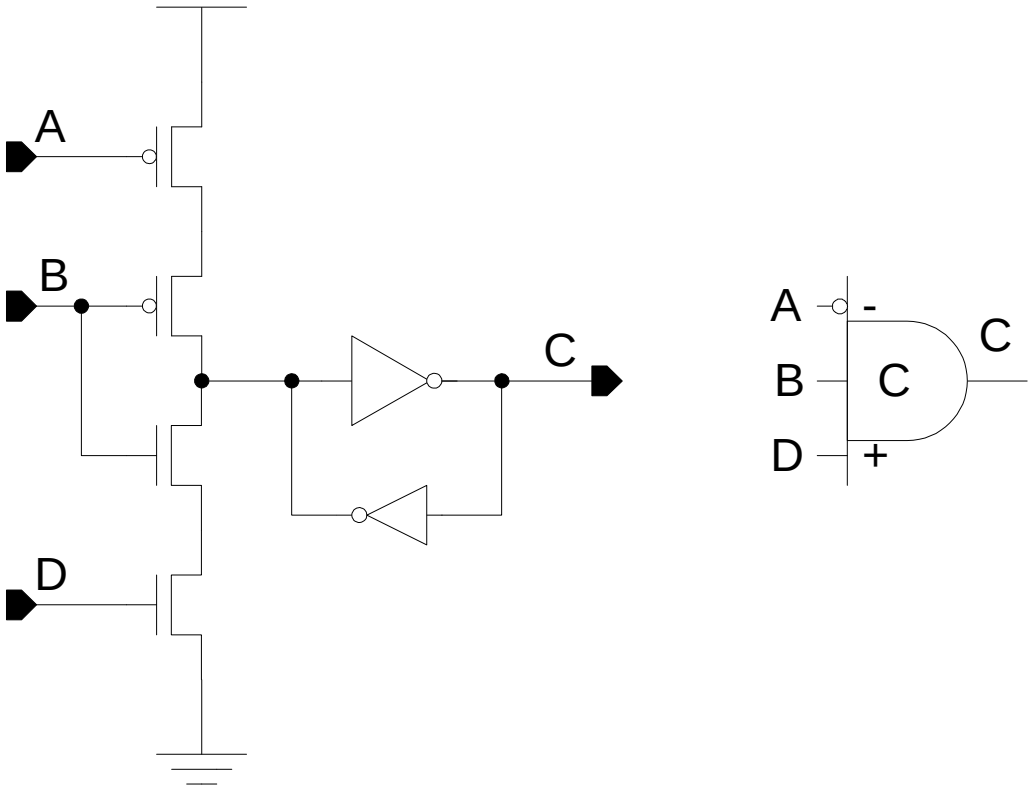

Abb. 23: Eine Zelle zum Speichern von Signalwechseln.

4.1.2 Makroelemente

Im folgenden werden Makroelemente für die asynchrone Schaltungstechnik beschrieben, die auf den vorgestellten primitiven Elementen aufsetzen. Auf sie ist das hierarchische Fehlermodellierungskonzept anzuwenden. Somit ist eine einfache, aber effektive Fehlermodellierung in einer Simulationsumgebung möglich. **Abb. 24** zeigt ein *Toggle* Modul, das aus einer zweistufigen Logik aufgebaut ist **[89Suth]**$_{L.2}$. Es steuert abwechselnd die Signalwechsel auf die Ausgänge **DOT** und **BLANK**. Ein häufiges Einsatzgebiet ist die Umwandlung von *Four-Phase* auf *Two-Phase* Signale in Steuerpfaden.

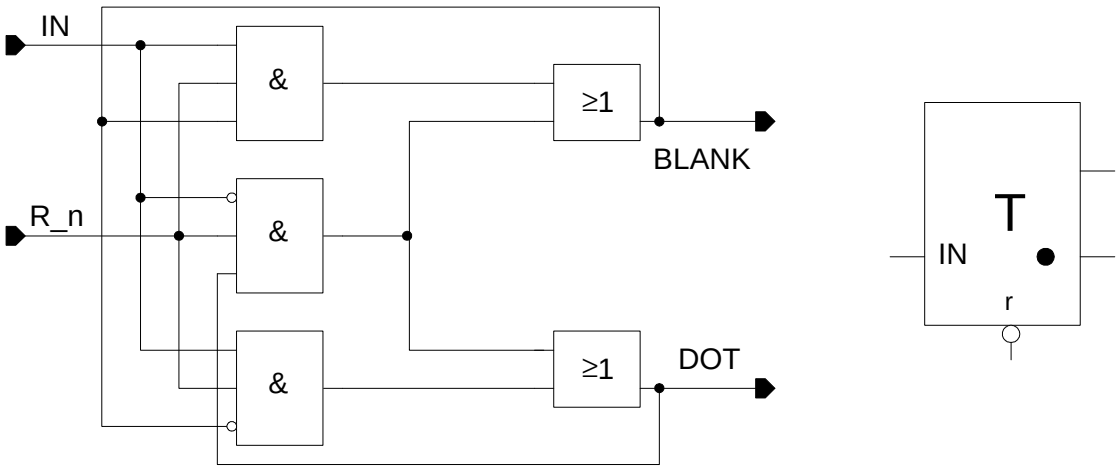

Abb. 24: Das *Toggle* Modul in Gatterrealisierung.

Eine Fehlermodellierung mit Haftfehlern an den Ein- und Ausgängen der primitiven Elemente des Moduls ist hierbei ausreichend. Dies entspricht auch der Verfahrenweise bei synchronen Schaltungen. Wird das *Toggle* als Zelle mit Transistoren aufgebaut, muß überprüft werden, ob diese Fehlermodellierung alle Fehler auf der Transistorebene abdeckt.

Abb. 25 zeigt, dass auch für das *Decision Wait* Modul eine Realisierung mit Standardelementen möglich ist. Dieses Element wird unter anderem benutzt, um ein *CALL* Element zu entwickeln. Liegt ein Signalwechsel an **A1** oder **A2** an, wird er an eines der *Muller C-Elemente* weitergeleitet. Mit einem Wechsel von **FIRE** wird ein Signalwechsel an **Z1** oder **Z2** erzeugt. Durch die Rückkopplung der Ausgänge auf den zweiten Eingang des XORs wird ein weiterer Signalwechsel an den Eingängen erst nach einem erneuten Wechsel von **FIRE** erfolgen.

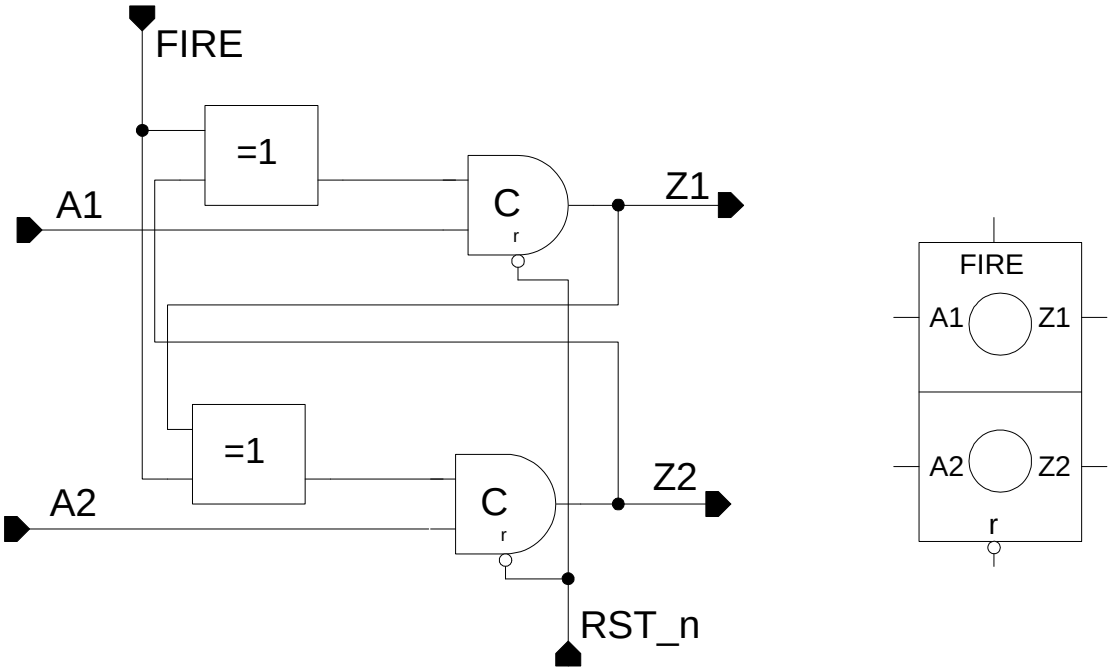

Abb. 25: Das *Decision Wait* Modul auf Gatterebene.

Beim *Decision Wait* Signal kann wieder das hierarchische Fehlermodellierungskonzept angewendet werden, da für die Basiselemente bereits eine Modellierung existiert und eine Haftfehlermodellierung an den Ein- und Ausgängen durch einen Fehlersimulator möglich ist. *Arbiter* kontrollieren den Mechanismus zur Ansteuerung von geteilten Ressourcen. Eine Ampelanlage steuert beispielsweise den Zugang zu einer Kreuzung von mehreren Straßen aus. In digitalen Systemen werden insbesondere Speicher und Datenbusse von unterschiedlichen Quellen und Senken unabhängig voneinander angesprochen. Um Kollisionen zu vermeiden, werden *Arbiter* eingesetzt. In **[80Seitb]**$_{L.2}$ wurde das Konzept von *Arbitern* mit Hilfe eines *Mutual Exclusion* Elements präsentiert[34]. Exemplarisch soll hier ein *Arbiter* untersucht werden, der mit einem *Mutual Exclusion* Element, zwei Speicherelementen und zwei EXOR Gattern aufgebaut ist. Siehe **Abb. 26**.

[34] Für *Arbiter* gibt es eine Vielzahl an Realisierungsformen **[85DillC, 86Blac, 92LeonB, 94KishKT]**$_{L.2}$. Diese unterschiedlichen Realisierungsformen sind in der vielfältigen Anwendung von *Arbitern* für Bussteuerungen begründet.

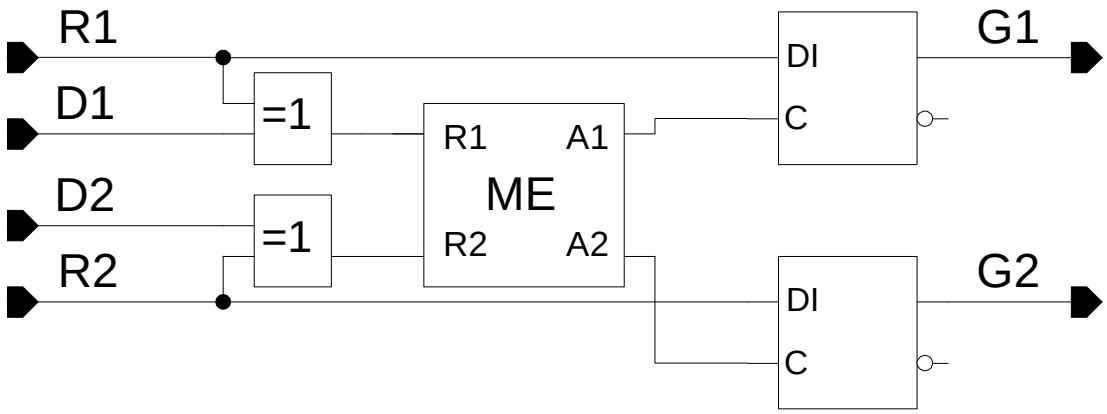

Abb. 26: Der *Arbiter* wird aus mehreren Komplexgattern zusammengesetzt.

Hier ist auch wieder eine hierarchische Fehlermodellierung möglich. Werden *Arbiter* aus Transistoren aufgebaut, um spezielle, asynchrone Eigenschaften aufweisen zu können, muß für diesen Fall eine Fehlermodellierung auf Transistorebene vorgenommen werden. Auch ein *CALL* Element kann mit Hilfe einer hierarchischen Fehlermodellierung abgebildet werden, da es häufig aus einem *Decision Wait* und einem XOR aufgebaut ist. **Abb. 27** zeigt diese Realisierung. Es wird zur Steuerung von Empfängern eingesetzt, die von zwei Sendern Signale erhalten sollen. Das *Mutual Exclusion* Element ermöglicht die sequentielle Verarbeitung von Signalwechseln an **R1** und **R2.**

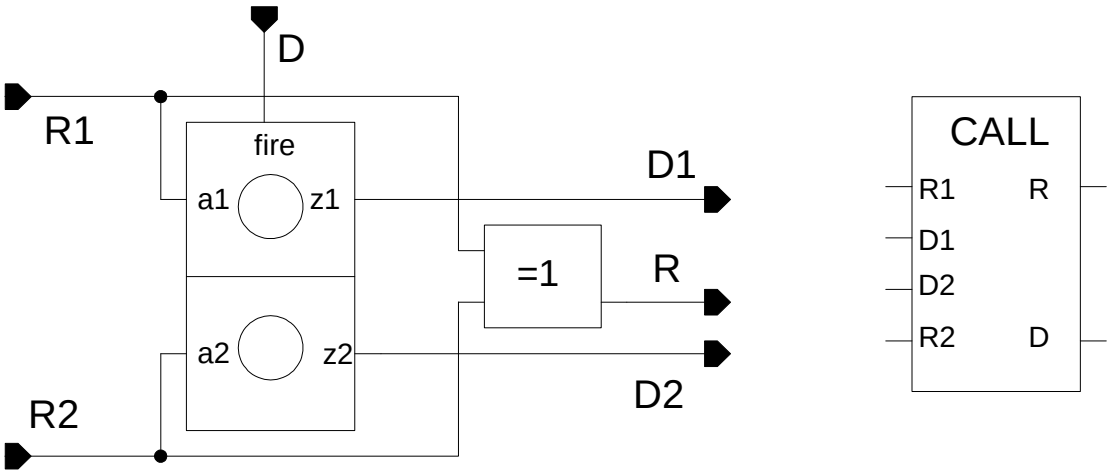

Abb. 27: Das *CALL* Element.

Für das *Select* Modul in **Abb. 28** kann - wie schon zuvor bei den anderen Modulen - eine hierarchische Fehlermodellierung vorgenommen werden. Das *Select* Modul wird als Demultiplexer für Steuersignale in asynchronen Schaltungen

verwendet. Abhängig davon, wie der Steuereingang **SEL** gesetzt ist, werden die Signalwechsel auf einen der beiden Steuereingänge weitergeleitet. Das *Select* Modul besitzt eine ähnliche Funktion, wie das *Toggle* Element. Durch den zusätzlichen Steuereingang wird über die Fortschaltung des Signals entschieden.

Werden statt der beschriebenen hierarchischen Realisierung komplexe, dicht gepackte Zellen entwickelt, erhöht sich die Wahrscheinlichkeit, dass Fehler zwischen den Signalleitungen und den Transistoren benachbarter Zellen auftreten. Dann sind Kurzschlüsse zwischen den Zellen nicht auszuschließen. Bei Untersuchungen am *Muller C-Element* konnte dies bestätigt werden**[95BrzoR, 95HeiS]**$_{L.3}$.

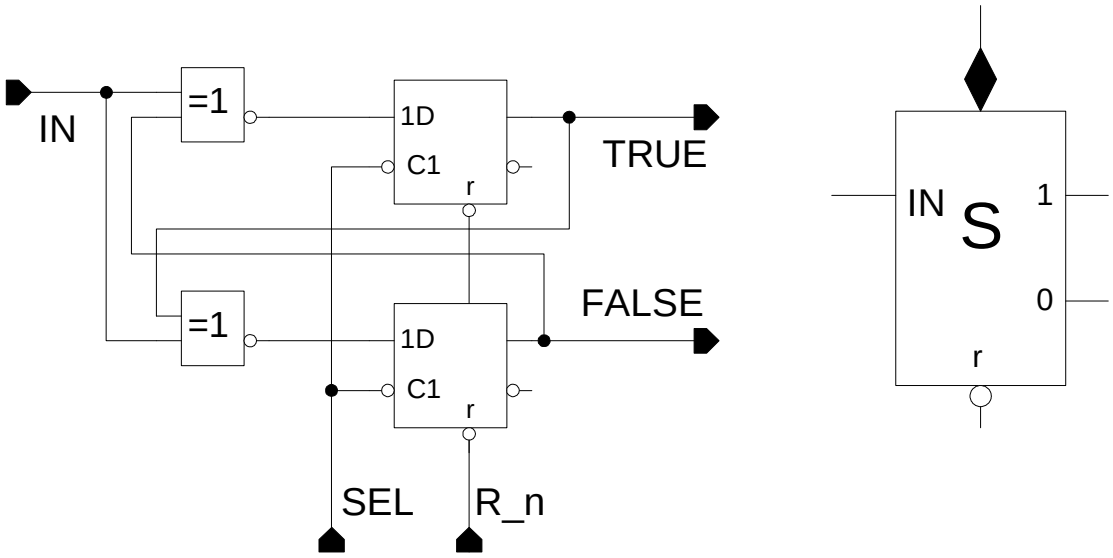

Abb. 28: Das *Select* Modul.

4.1.3 Speicherzellen

Neben der Signalablaufsteuerung gibt es verschiedene Realisierungen zur Datenspeicherung. Speicherelemente werden in Pipelinestufen bei vielen Entwürfen von asynchronen Schaltungen auf ihre Geschwindigkeit und Fläche untersucht[35]. Im folgenden sollen prinzipielle Strukturen erläutert werden, die für den testfreundlichen Entwurf wichtig sind. Es werden dabei zwei Arten von Schaltern untersucht, die häufig Verwendung finden.

Abb. 29 zeigt eine mögliche Realisierungsform eines Schalters mit *Pass* Transistoren. Das Steuersignal C schaltet mit den Steuereingängen der Transistoren jeweils einen der *Pass* Transistoren durch, wobei *Source* mit dem Eingang und *Drain* mit dem Ausgang verbunden ist. Ein Inverter verstärkt das Signal zum Ausgang.

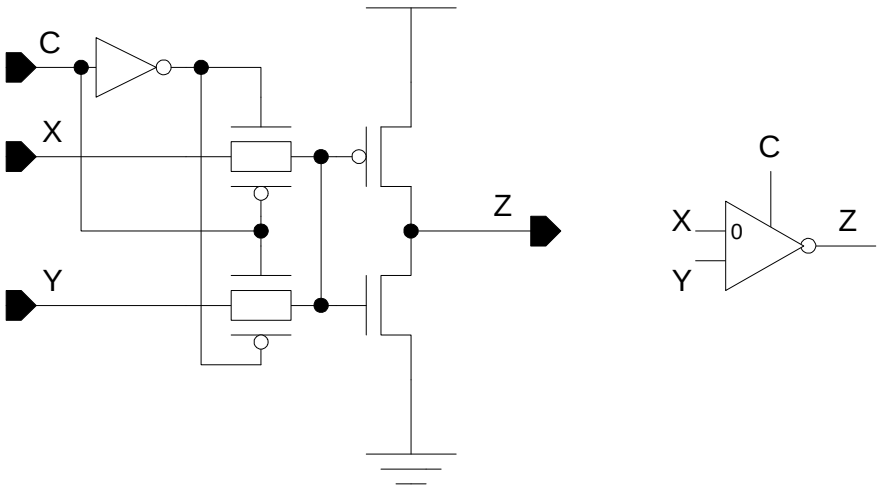

Abb. 29: Ein invertierter Schalter mit *Pass* Transistoren (MUX pass).

Diese kompakte Realisierung ist für die Fehlermodellierung mit Haftfehlern problematisch. Ursache hierfür ist die parallele Struktur der P-Kanal und N-Kanal Transistoren, die als rekonvergierende Pfade in der Schaltung angesehen werden können. Ist einer der beiden *Pass* Transistoren defekt, schaltet der andere Transistor nur einen Signalpegel ohne zusätzliche Verzögerung und ohne

[35] Siehe hierzu **[95DayW, 95FurbD]**$_{L.2}$.

Spannungsabfall durch. Der andere Pegel erhält ein schlechteres Signalübertragungsverhalten, welches nur schwer auf ein Fehlermodell auf Gatterebene abgebildet werden kann. Ein Defekt kann sich auf die Treiberleistung des Schalters und damit auf das dynamische Verhalten der Schaltung auswirken. Hierbei ist zu bemerken, dass die nachfolgende Stufe von der Vorgängerstufe des *Pass* Transistors getrieben wird und keine Verbindung nach VDD und GND besitzt. Es ist also nötig, zwischen den *Pass* Transistoren Treiber einzusetzen, die das Treiben des Ausgangs über VDD und GND ermöglichen. Geschieht dies nicht, verschlechtert sich die Signalform mit jeder *Pass* Transistorstufe. Die Verschlechterung der Signalform kann metastabile Zustände und Querströme bei CMOS Technik erzeugen.

Modelliert man für alle Transistorkurzschlüsse und unterbrochenen Leitungen das Fehlverhalten auf Gatterebene, ergibt sich folgendes Bild: 15% lassen sich als Haftfehler an den Ein- und Ausgängen und 5% als kurzgeschlossene Eingänge modellieren. 10% ergeben eine Funktionskonvertierung. In 30% der Transistorfehler wird eine Bedingung zur Sensibilisierung des Fehlers benötigt. Bei diesen Fehlern, die durch kapazitive Lasten ein sequentielles Verhalten erzeugen, werden Pfade zwischen den Ein- und Ausgängen unterbrochen. Je nach Komplexität werden zur Beobachtung ein Testmuster oder eine Folge von Testmustern benötigt. 30% der Fehler können mit einer IDDQ Messung sicher erkannt werden. In 10% der Fälle äußert sich ein Transistorfehler nur mit einem dynamischen Fehlverhalten. Dieses entsteht durch einen Defekt an einem der *Pass* Transistoren.

- **Produktionsfehler, die nur einen Pfad einer *Pass* Transistorstruktur unterbrechen, wirken sich im günstigsten Fall auf eine verschlechterte Treiberfähigkeit der nachfolgenden Stufe aus. Da viele Produktionstests dynamisches Verhalten bei der Verwendung von Testmusterpaaren nur bedingt berücksichtigen, kann eine Fehlererkennung nicht garantiert werden.**

Abb. 30 zeigt eine weitere Realisierungsform mit zwei hochohmig schaltbaren Invertern. Diese, um zwei Transistoren größere Schaltung ermöglicht eine

bessere Modellierung von Fehlern auf Transistorebene. Hier entstehen keine rekonvergierenden Pfade durch die Schaltungsstruktur. Vielmehr werden die Eingangssignale nach VDD oder GND durchgeschaltet. Die nachfolgende Stufe ist somit über die *Drain* und *Source*-Anschlüsse des Transistors mit dem Hoch- oder Niedrigspannungspegel der Schaltung verbunden. Das Latch mit Tristate Elementen besitzt daher eine einfachere Fehlermodellierung auf Gatterebene. Bei einer Fehlermodellierung ergibt sich hier eine andere Verteilung zu der Realisierungsform von **Abb. 29**. Es können nur 7% als Eingangshaftfehler modelliert werden. Eine Funktionsänderung tritt in 25% der Fälle auf. Bei 43% der Transistorfehler läßt sich ein Fehlverhalten mit einem bestimmten Testmuster oder Testmusterpaar am Ausgang beobachten. Diese Fehler lassen sich mit Testmustern in einem digitalen Fehlersimulator bearbeiten. Kein logisches Fehlverhalten, sondern ein erhöhter Ruhestrom ist bei 25% der Fehler zu beobachten.

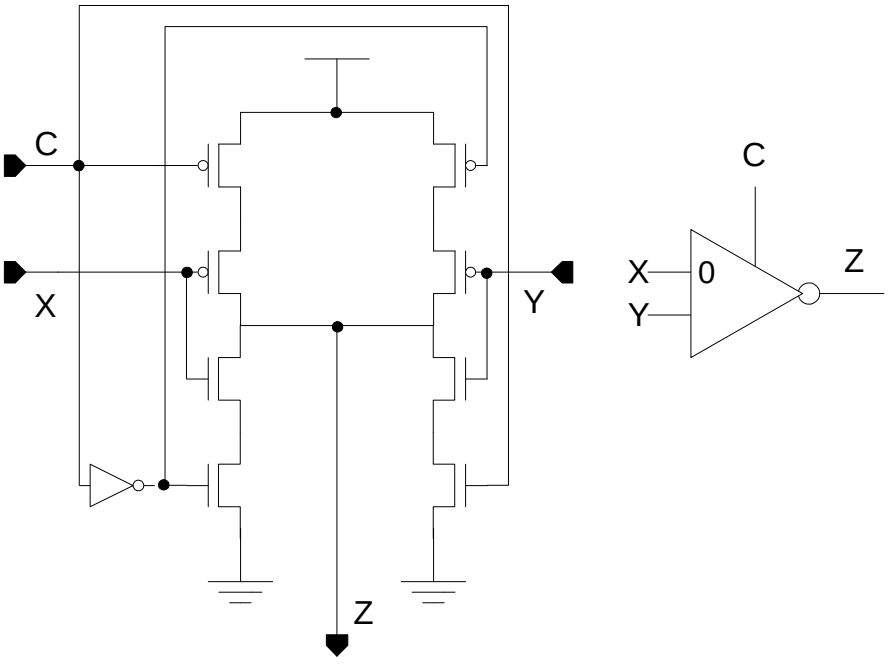

Abb. 30: Ein invertierter Schalter mit hochohmig schaltbaren Invertern (MUX tri).

- **Fehlersimulatoren können bei der Sensibilisierung des Fehlers durch die Beobachtung über die Versorgungsleitungen den Fehler als erkannt bewerten. Hierzu wird ein digitales Testmuster in Verbindung mit einer Stromauf-**

nahmemessung während des Produktionstests verwendet. Eine Fehlermodellierung eines Schalters mit hochohmig schaltbaren Invertern ist daher effektiver als mit *Pass* Transistoren.

Mit diesen zwei Schaltervarianten können durch Rückkopplungen Latches entwickelt werden. Das Latch kann somit mit *Pass* Transistoren oder Tristate-Invertern aufgebaut werden. Um die Fläche und Geschwindigkeit zu optimieren, können die Speicherelemente invertierend aufgebaut werden. Durch die Master-Slave Struktur wird das invertierende Verhalten aufgehoben. Der Aufbau des Latches für beide Versionen ist in **Abb. 31** zu sehen. Latches werden auch als Speicher für die *Four-Phase Bundled Data Convention* eingesetzt. Die Fehlermodellierung entspricht der des Schalters mit einem rückgekoppelten Inverter. Darüber hinaus gibt es noch weitere Realisierungen von Speichern, die in Registerzellen häufig verwendet werden.

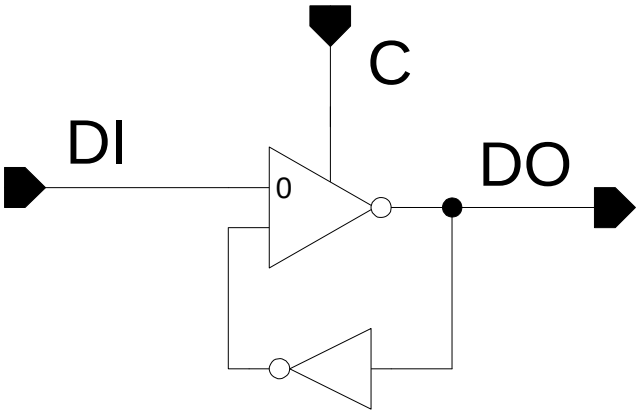

Abb. 31: Ein Latch mit einem Multiplexer mit hochohmig schaltbaren Invertern.

Abb. 32 zeigt die Realisierung eines Latches mit Reset[36]. Links befindet sich eine Eingangsstufe mit einem hochohmig schaltbaren Treiber, der die Speicherzelle mit einem neuen Datum beschreiben kann. Das Datum wird in einem rückgekoppelten Inverter gespeichert. Beim Schreiben eines neuen Datums treiben aber

[36] Das Latch ist wie ein pseudostatisches *Muller C-Element* aufgebaut und besitzt daher auch die gleiche Fehlermodellierungscharakteristik.

die Transistoren der Eingangsstufe gegen die der Speicherstufe, wodurch ein erhöhter Querstrom beim Umlagevorgang durch die Transistoren fließt.

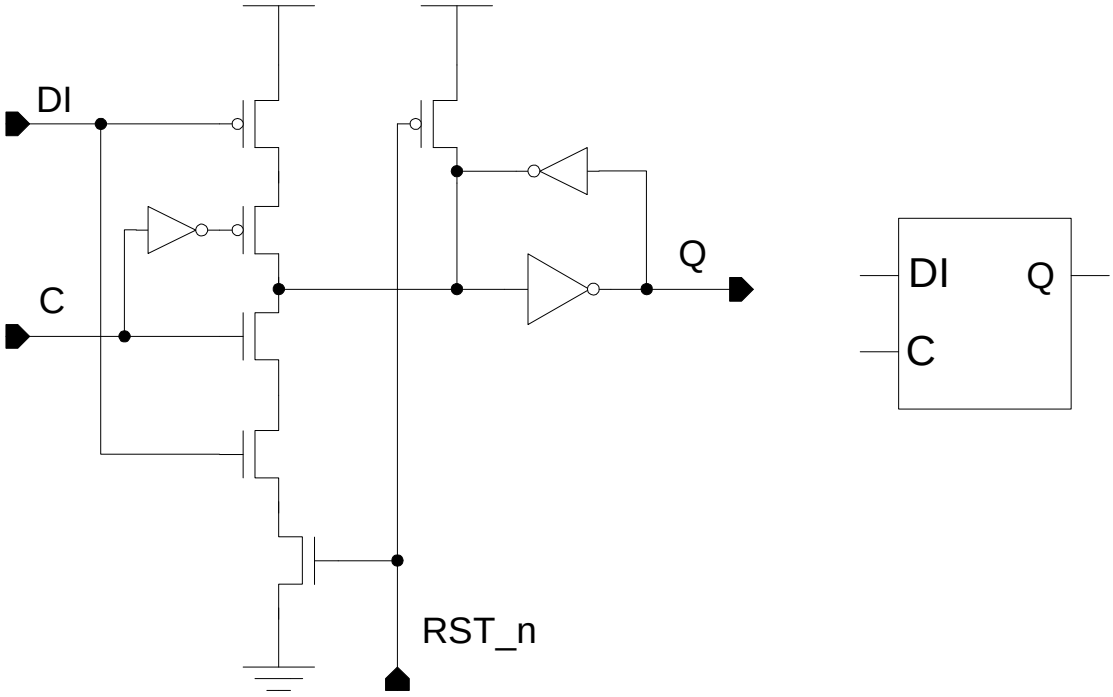

Abb. 32: Latch-t auf Transistorebene.

Eine weitere Möglichkeit besteht darin, einen invertierten Schalter mit den Ausgängen von zwei parallel angeordneten transparenten Speichern zu verbinden. Die Steuereingänge der Speicher ermöglichen das Umschalten des Haltezustands (*Capture*) und der Multiplexer ermöglicht das Weiterschalten des gültigen Signals zum Datenausgang **DO** (*Pass*). Bemerkenswert ist, dass das Register transparent schaltbar ist, obwohl es eine Art Master-Slave Struktur besitzt. **Abb. 33** zeigt eine mögliche Realisierungsform des *Capture-Pass* Registers **[89Suth]**$_{L.2}$. Das *Capture* Signal, **C**, schaltet eine der Speicherzellen transparent. Gleichzeitig speichert die andere Speicherzelle das Datum vom Eingang **DI**. Das *Pass*-Signal entscheidet, welches der Speicherelementausgänge auf den Ausgang **DO** weitergeleitet wird. Eine Fehlermodellierung ist hier hierarchisch möglich, da dieses Register aus Standardzellen besteht.

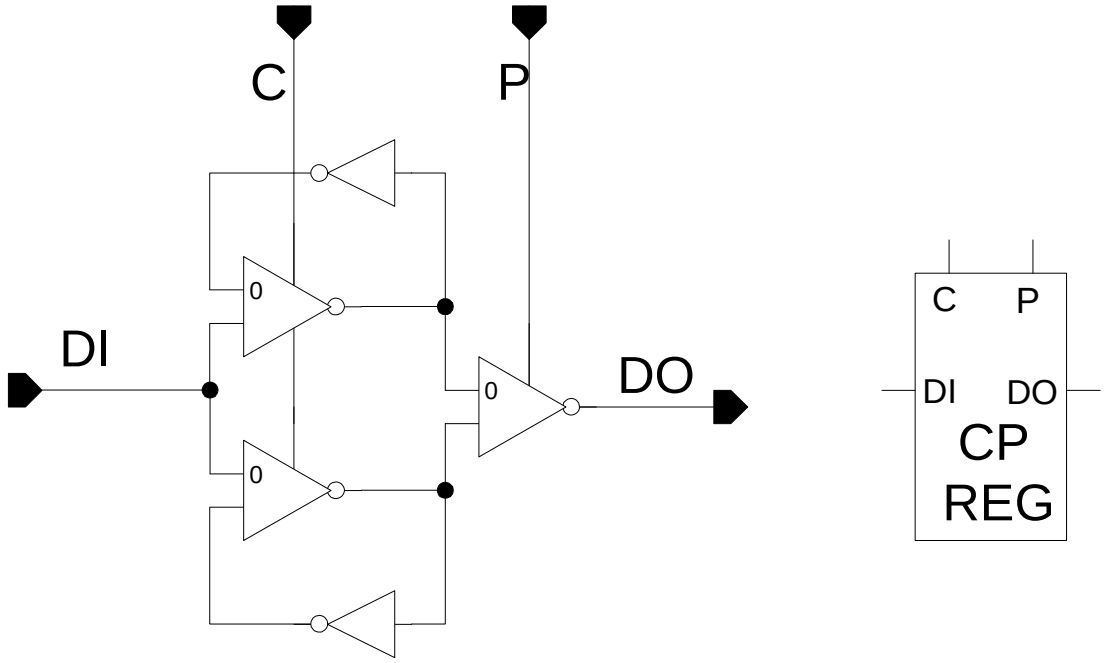

Abb. 33: Eine Realisierung eines nichttransparenten Speichers für *Two-Phase* Signale.

4.2 Standardzellen für den testfreundlichen Entwurf

Im folgenden werden neue Zellen beschrieben, die für den entwickelten testfreundlichen Entwurf notwendig sind. Diese Zellen ergeben sich aus den Konzepten, die in **Kapitel 5** vorgeschlagen werden[37]. Mit der Beschreibung der Funktionsweise und einer Realisierung entstehen somit weitere Module einer asynchronen Schaltungsbibliothek. Durch das Fehlen vergleichbarer Module in asynchronen Schaltungsbibliotheken wird die mangelnde Berücksichtigung des testfreundlichen Entwurfs im Bereich der asynchronen Schaltungsentwicklung deutlich[38].

Die Module zum testfreundlichen Entwurf teilen sich hierbei in zwei Anwendungsbereiche auf. Zuerst werden Elemente zur Datenspeicherung in den Registerstrukturen beschrieben. Hierdurch kann die Steuer- und Beobachtbarkeit tief eingebetteter Speicherelemente erhöht werden. Die Module zur Datenspeicherung sind unabhängig von der Art der Synchronisation. Die Möglichkeit der Verwendung von synchronen Datenpfadelementen und Datenstrukturen in der *Micropipeline* Schaltungstechnik bedeutet gleichzeitig, dass asynchrone Datenpfadelemente auch in synchronen Schaltungen verwendet werden können. Der zweite Teil enthält Module für die asynchrone Steuerung des Datenpfads im Testbetrieb. Hier müssen Signale aus dem Steuerpfad des Datenpfads aus- und eingekoppelt werden. Diese Auskopplung erfolgt nach der *Bundled Data Convention*.

---

[37] Ein synchroner testfreundlicher Entwurf asynchroner Schaltungen, wie er zur Zeit häufig vorgeschlagen wird, ist prinzipiell mit diesen Registerstufen möglich. Er ist jedoch nicht Gegenstand dieser Untersuchung.

[38] Bei bestehenden Arbeiten werden häufig Module und Konzepte aus dem Bereich des synchronen testfreundlichen Entwurfs verwendet. Siehe hierzu **Kapitel 3**.

4.2.1 Registerzellen für passive Testhilfen

Zum testfreundlichen Entwurf werden in der Regel Multiplexer vor den Speicherelementen plaziert, die das Anlegen eines Testmusters ermöglichen. Scanfähige Speicherzellen weisen daher zwei Nachteile gegenüber den nicht scanfähigen Zellen auf. Sie verursachen eine zusätzliche Signalverzögerung und benötigen eine größere Fläche aufgrund ihrer erweiterten Funktionalität. Insbesondere die zusätzliche Signalverzögerung bedeutet für schnelle Schaltungen einen großen Nachteil, da sich häufig die testfreundlichen Entwurfsmaßnahmen im kritischen Pfad der Schaltung befinden und damit die Performance direkt beeinflussen. Dieses führt zu Akzeptanzproblemen des testfreundlichen Entwurfs. Es soll versucht werden, diese Performanceverluste zu minimieren.

Die Einkopplung und damit die Steuerung der Registerzelle erfolgt über Multiplexer am Eingang **DI**. Durch den Einbau eines Multiplexers entsteht eine zusätzliche Verzögerung im Datenpfad, die auch den größten Teil der Performanceverluste verursacht. Beobachtet wird der Inhalt des Registers am Ausgang des Speicherelements, welches eine zusätzliche kapazitive Last bedeutet. Bei klassischen Speicherzellen wird das steuernde Testsignal durch einen zusätzlichen Multiplexer in den kritischen Datenpfad vor der eigentlichen Speicherzelle eingekoppelt. Dies ist in **Abb. 34** für zwei Speichervarianten dargestellt.

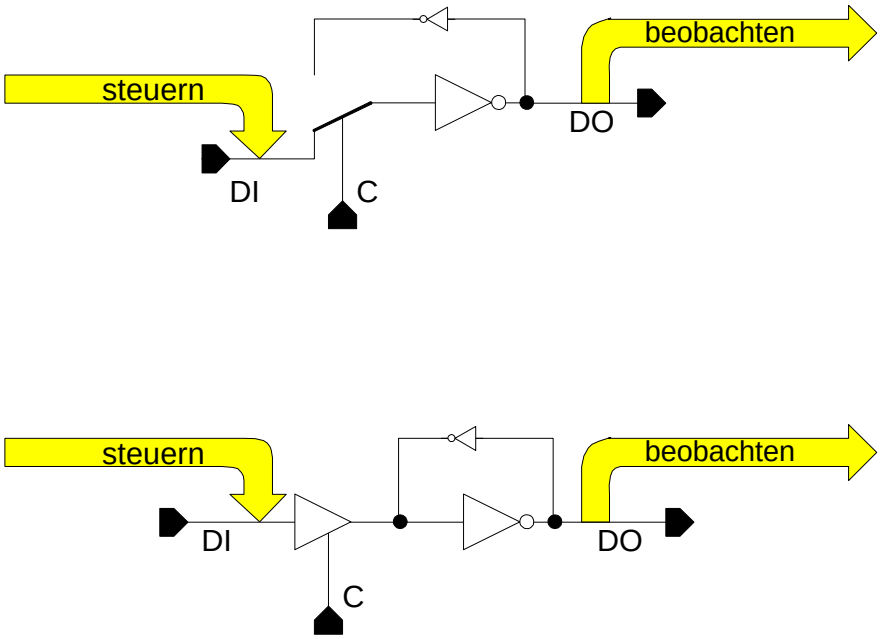

Abb. 34: Bei klassischen Scanzellen werden die Testsignale im kritischen Datenpfad eingekoppelt.

Zur Neutralisierung der Performanceverluste dürfen die Testsignale nicht im kritischen Datenpfad geschaltet werden. Die Datensignale der neu entwickelten Speicherzellen werden daher über den Rückkopplungspfad gesteuert und beobachtet. Dies ist in **Abb. 35** zu sehen. Der kritische Datenpfad ist nicht mehr durch einen zusätzlichen Multiplexer verlängert oder durch zusätzliche Kapazitäten am Ausgang belastet. Es können sich lediglich erhöhte kapazitive Lasten des Rückkopplungspfads ergeben. Obwohl hier für asynchrone Schaltungen vorgestellt, gilt dieses Prinzip auch für synchrone Speicherzellen.

- **Der Nachteil, dass scanfähige Registerzellen die Performance der Schaltung negativ beeinflussen, ist somit gelöst.**

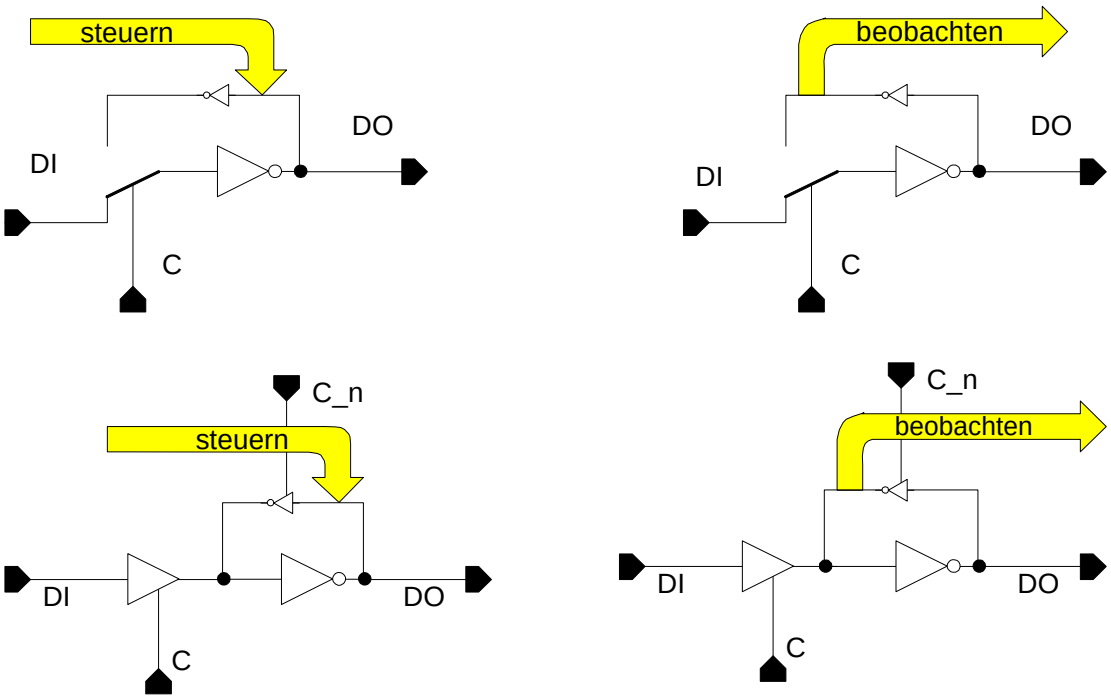

Abb. 35: Der neue Ansatz für Scanzellen steuert und beobachtet den Rückkopplungspfad der Zelle.

Das in **Abb. 35** dargestellte Prinzip soll nun auf Speicherzellen angewendet werden. **Abb. 36** zeigt die Realisierung der Speicherzelle mit invertierten Schaltern. Ist Test Mode Select (**TMS**) in der unteren Position, verhält sich das Register als normaler Datenspeicher. Mit **TMS** können das Latch in den Testbetrieb umgeschaltet und Daten im oberen Register gespeichert werden. Über Test Data Input (**TDI**) werden neue Testdaten in die Registerzelle eingelesen sowie über **TDO** ausgelesen. **TMS** dient auch zum Einlesen eines neuen Testmusters in den Rückkopplungspfad der Speicherzelle. Zum Beobachten des Speicherinhalts sowie zum Weiterleiten des Testsignals wird das Signal Test Capture (**TC**) verwendet. Es dient auch bei der seriellen Schieberegisteroperation als Signal zum Öffnen und Schließen der Scanzelle. Der Rückkopplungsinverter wird für das Speicherelement des Scanpfads verwendet. Die beiden Speicherelemente sind somit nicht entkoppelt. Es kann daher nur ein Speicherelement aktiv sein. Das andere Element besitzt das Verhalten eines Multiplexers. Hierdurch wird die Fläche - im Vergleich zu zwei vollständigen Registerzellen - reduziert.

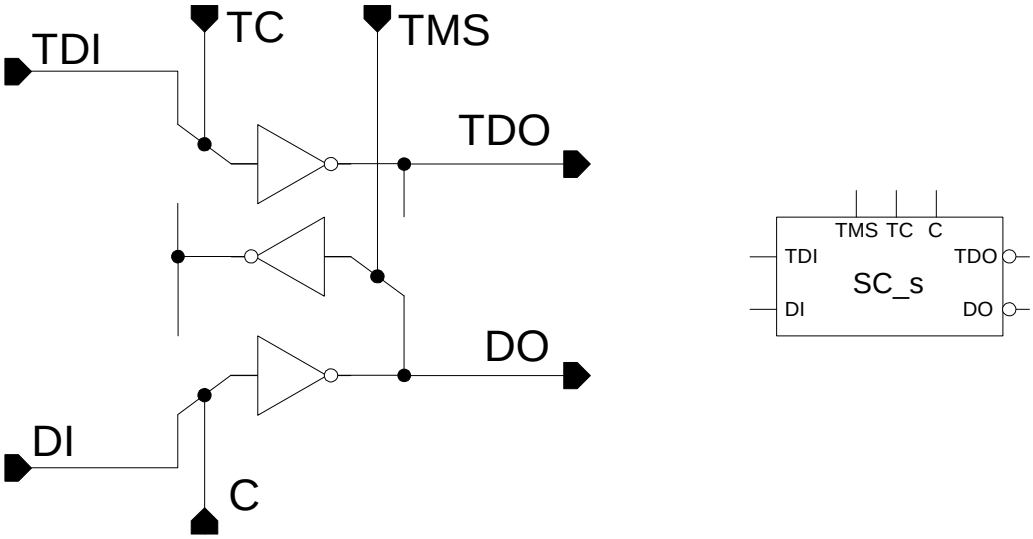

Abb. 36: Die Scan Cell ermöglicht die Steuer- und Beobachtbarkeit mit minimalen Verzögerungen.

Eine Fehlermodellierung erweist sich als sehr günstig, da die Transistoren über wenige Stufen mit **TDI** und **TDO** verbunden sind und keine rekonvergierenden Pfade existieren. Sie sind somit gut zu steuern und zu beobachten. Der hierarchische Fehlermodellierungsansatz kann auch hier erfolgen. **Abb. 37** zeigt die Realisierung der asynchronen Speicherzelle mit den invertierten Schaltern im Testpfad und *Pass* Transistoren im Datenpfad. Diese kompakte Realisierung erzeugt jedoch durch die *Pass* Transistortechnik Probleme bei der Fehlermodellierung.

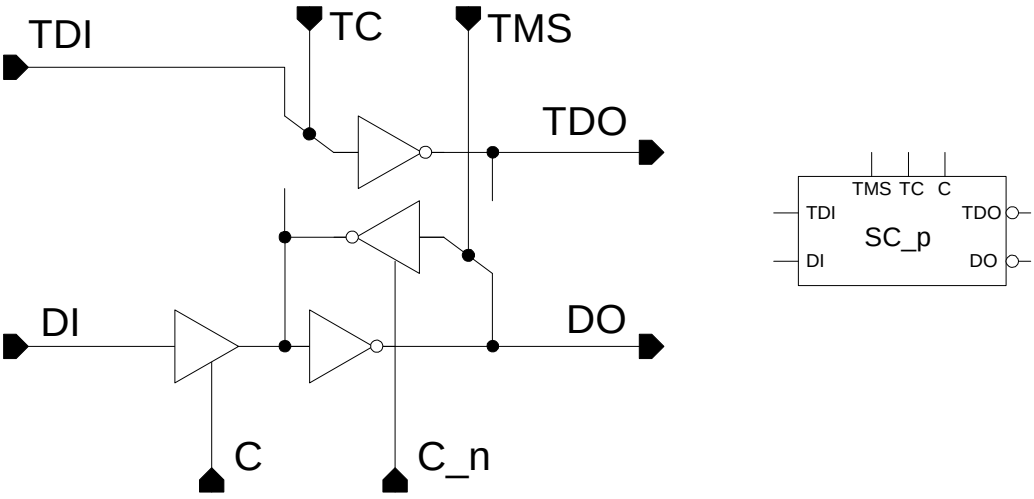

Abb. 37: Ein scanfähiges Latch mit *Pass* Transistoren.

Der invertierte Schalter im Testpfad kann auch durch einen *Pass* Transistor ersetzt werden, um die Transistoranzahl der Speicherzelle zu reduzieren. Dies erhöht, wie bei der LSSD Technik schon erwähnt, den Aufwand zur Erzeugung der Steuersignale, um zu verhindern, dass ein Querstrom zwischen VDD und GND entsteht. Die *Pass* Transistoren sind in gegenseitiger Abhängigkeit. Hieraus folgt, dass Teststeuersignale mit Steuersignalen im Datenpfad verknüpft werden müssen. Diese können eine zusätzliche Signalverzögerung der Steuersignale des Normalbetriebs erzeugen. **Abb. 38** zeigt eine weitere Realisierung der testfreundlichen Speicherzelle mit *Pass* Transistoren im Test und Datenpfad.

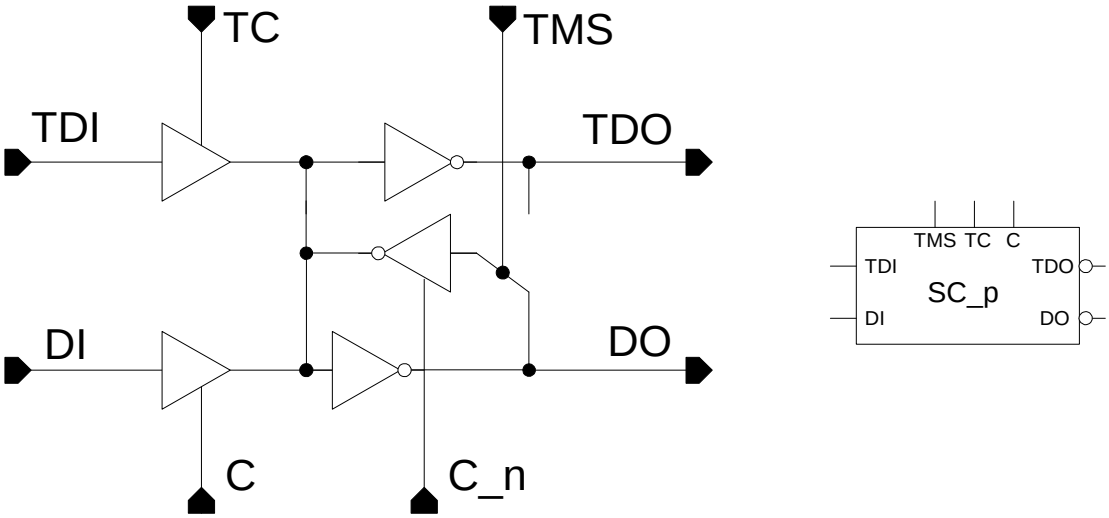

Abb. 38: Eine weitere Realisierung mit *Pass* Transistoren.

Die Schalter, die für die Testlogik integriert wurden, können mit minimalen Abmessungen dimensioniert werden, um den zusätzlichen Flächenaufwand zu reduzieren. Dies wird möglich, da die Scanpfad-Technik ein statisches Testverfahren ist, bei der die Performance eine untergeordnete Rolle spielt. Darüber hinaus muß **TDO** in der Regel nur ein weiteres Latch treiben. Soll die gemeinsame Rückkopplung umgangen werden, müssen zwei Pfade integriert werden. Eine Realisierungsform ist in **Abb. 39** zu sehen. Über das PRELOAD-Signal (**PRE**) können neue Testmuster in dem Register gespeichert werden. Mit SAMPLE (**S**) und **TC** wird ein Testergebnis im Scanpfad gespeichert.

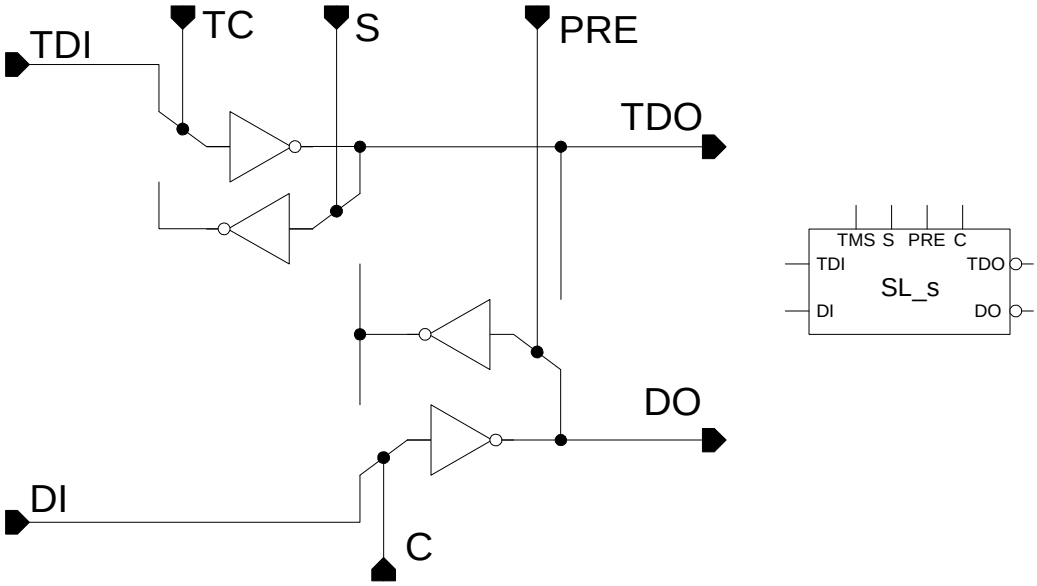

Abb. 39: Ein Scan Latch mit zwei voneinander entkoppelten Speicherzellen.

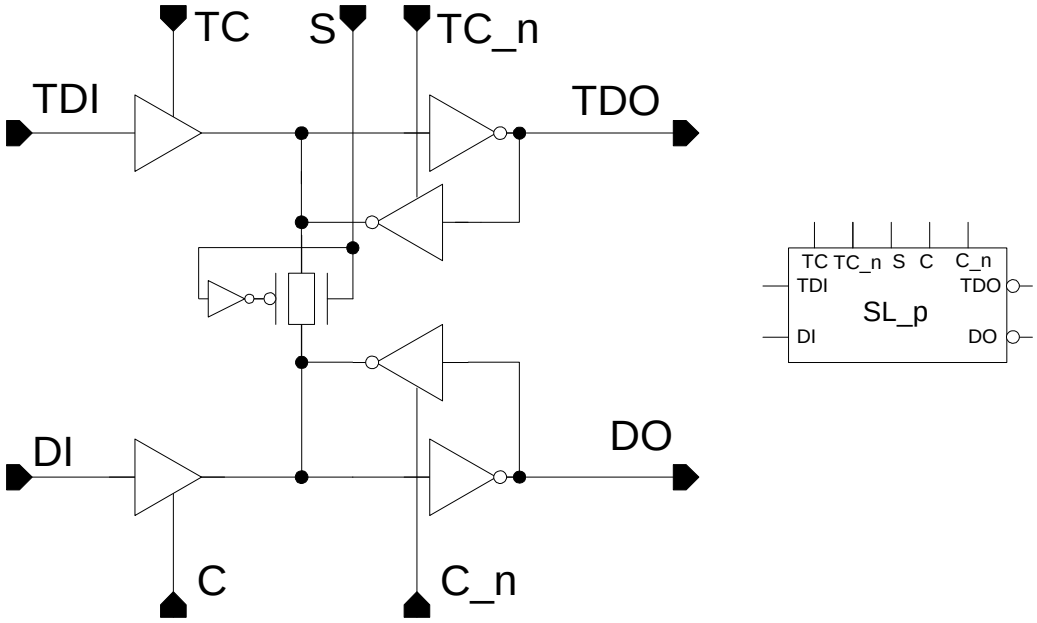

Abb. 40: Entkoppeltes Scanregister mit *Pass* Transistoren am Eingang.

Abb. 40 zeigt eine Darstellung des Scanregisters mit *Pass* Transistoren. Auch hier entstehen bei der Haftfehlermodellierung Probleme durch die parallele Anordnung der *Pass* Transistoren. Es stellt jedoch eine kompaktere Realisierung des Datenpfadelements dar.

Unterschiedliche Schalterstellungen in den Scan Zellen werden für die richtige Initialisierung der Scanpfade in asynchronen Schaltungen benötigt[39]. In **Abb. 41** sind Scanzellen mit unterschiedlichen Schalterstellungen zu sehen, die die modulare Entwicklung eines asynchronen Scanpfads stark vereinfachen.

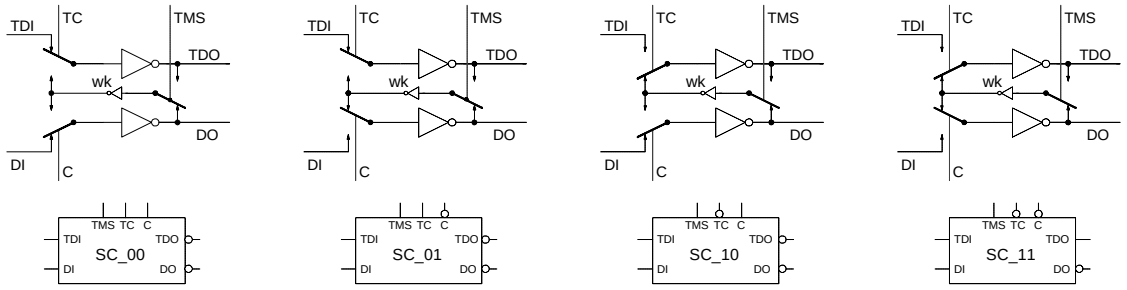

Abb. 41: Unterschiedliche Initialisierungszustände der Scan-Elemente reduzieren den Schaltungsaufwand.

Im folgenden soll der Aufbau von *Master-Slave* Scan-Registern für asynchrone sowie synchrone Schaltungen betrachtet werden. Mit zwei Scan-Elementen (SC) kann ein *Capture-Pass* Register für einen asynchronen Scanpfad realisiert werden. Das Prinzip des Registeraufbaus für den Datenpfad wurde dabei aus **[89Suth]**$_{L.2}$ entnommen[40]. Die Struktur ist in **Abb. 42** zu sehen. Die Signale *Pass* (**P**) und TEST PASS (**TP**) dienen zur Fortschaltung eines gültigen Signals über **DO** und **TDO** an die nächst folgende Stufe. Ist die Umschaltung erfolgt, kann ein neues Signal an **DI** oder **TDI** angelegt werden.

[39] Dies wird auch in **Kapitel 2.2** und **Kapitel 5.2** näher erklärt.

[40] Das *Capture-Pass* Register ist in **Abb. 14** dargestellt.

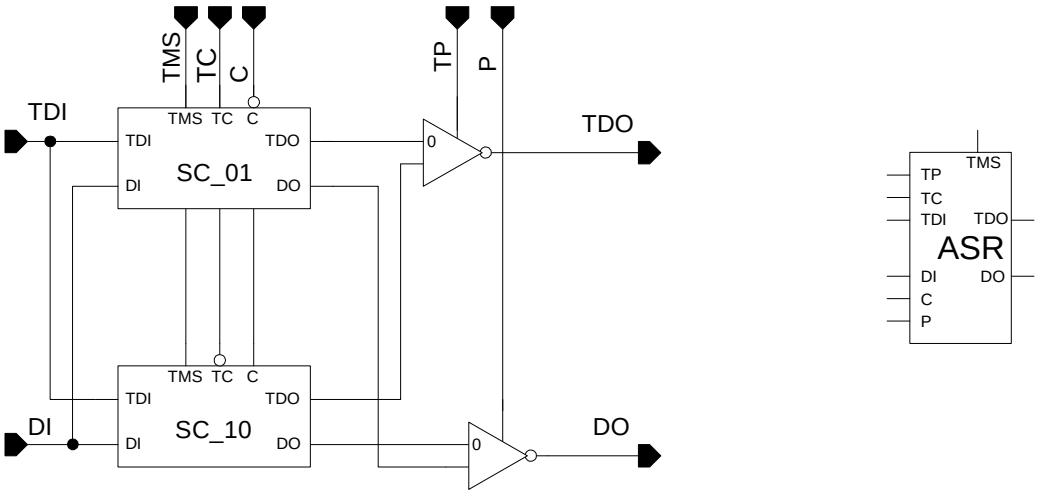

Abb. 42: Das scanfähige *Capture-Pass* Register nach dem Sutherland-Prinzip.

Für synchrone Schaltungen werden in der Regel zwei Latches hintereinander geschaltet, um ein Register zu erzeugen. Für ein Scan Register mit der hier bevorzugten Methode ergibt sich eine Anordnung, wie in **Abb. 43** zu sehen ist. Auch hier gilt, dass die zusätzlichen Multiplexer sich nicht im kritischen Pfad befinden, sondern im Rückkopplungspfad der Scanzelle, SC. Das Scan-Register hat somit keine zusätzliche Signalverzögerung.

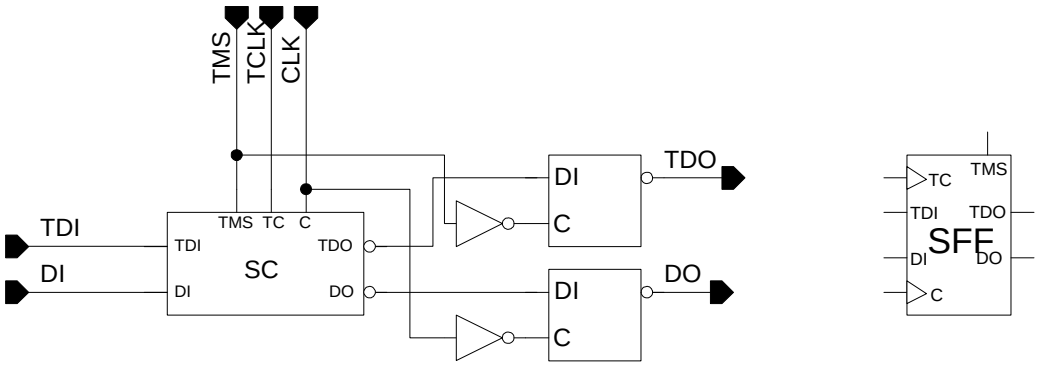

Abb. 43: Eine scanfähige Speicherzelle für synchrones Design.

Durch den Austausch von SC mit der Scan Zelle SL ergibt sich ein etwas anderes Verhalten. Die scanfähige Speicherzelle kann unabhängig von der Steuerung des Datenpfads neue Testmuster einlesen und auslesen, da für Daten- und Testpfad ein eigenes Latch zur Verfügung steht. Mit diesem Register ist es auch möglich, Testmusterpaare an die zu testenden Module anzulegen und somit dynamisches Verhalten zu überprüfen.

4.2.2 Registerzellen für den Built-In Self-Test

Es wird ein Konzept für die asynchrone Generierung und Komprimierung von Testmustern in **Kapitel 5** vorgestellt. Damit können Testmuster mit der Geschwindigkeit der Schaltung generiert und an die Module angelegt werden, ohne diese über einen Testautomaten zuzuführen. Im folgenden werden die notwendigen Speicherelemente für die aktiven testfreundlichen Entwurfshilfen entwickelt.

Zur schnellen Testmusterkomprimierung wird eine zusätzliche Speicherstufe integriert, die die bereits komprimierte Signatur mit dem aktuell anliegenden Testmuster verknüpft. **Abb. 44** zeigt diesen Speicher, der zwei Testmuster an **TDIe** und **TDIo** XOR verknüpft und das Ergebnis mit **TAI** in einem Latch speichert, wie es für die Testdatenkompression nötig ist. In den anderen Testbetriebsarten arbeitet die Logik vor dem Latch als Schalter mit TP als Steuersignal. Zur Testdatenkompression wird das Signal TX eingesetzt[41].

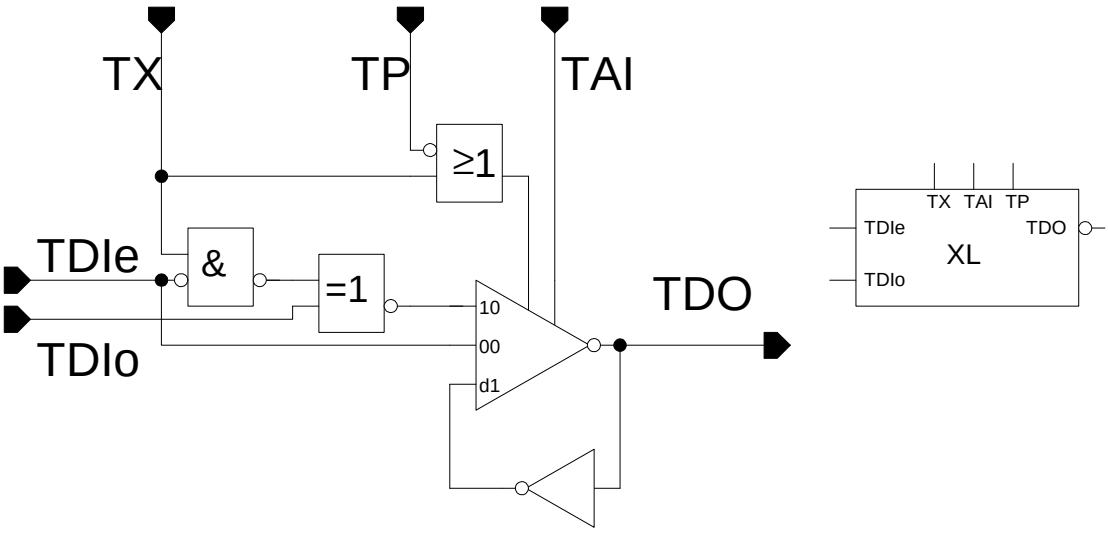

Abb. 44: Das Register zur Speicherung der komprimierten Testantwort.

[41] Siehe hierzu **Kapitel 5.2**.

Um die Master-Slave Struktur nach Sutherland zu erhalten, erfolgt die Anordnung, bei der der Schalter des Testdatenpfads durch das XL Modul ersetzt wird. **Abb. 45** stellt die Realisierung einer asynchronen BILBO Zelle, ABC, dar. Sie zeigt die Speicherzelle XL zur Unterstützung eines asynchronen BIST Entwurfs für asynchrone Schaltungen, analog zum BILBO Verfahren für synchrone Schaltungen **[79KonnMZ]**$_{L.2}$. Der Inhalt des BILBOs und das neue Testmuster werden dabei in den beiden vorderen Latches gespeichert. Die Berechnung und Speicherung der neuen Signatur erfolgt im XL Latch.

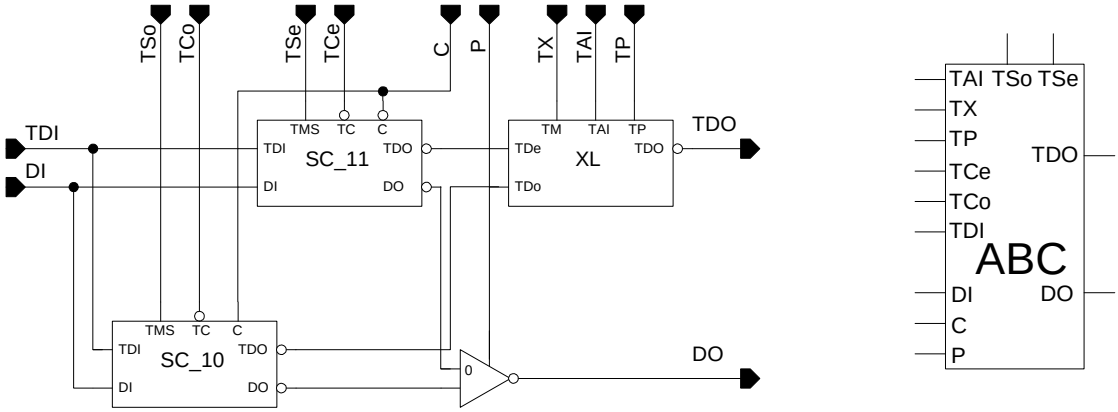

Abb. 45: Die Speicherzelle für das asynchrone BILBO Konzept.

Ist es das Ziel, das für asynchrone Schaltungen entwickelte Prinzip auf synchrone Schaltungen zu übertragen, ergibt sich eine ähnliche Realisierung wie beim Scanpfad. **Abb. 46** zeigt, dass das Latch im Scanpfad durch die XL Zelle ersetzt wird, um die Signatur zu berechnen. Während ein neues Testmuster an den Eingang des Registers angelegt wird, kann im SC Latch die alte Signatur gespeichert werden. Mit der Taktflanke von TLCK kann die neue berechnete Signatur in XL gespeichert werden. Mit der fallenden Flanke wird die berechnete Signatur nach SC der folgenden Stufe über TDO weitergeleitet. Das SC-Element kann durch ein SL-Element ausgetauscht werden. Somit kann der Testdatenpfad mit einem neuen Testmuster initialisiert werden, ohne den Datenpfad zu unterbrechen.

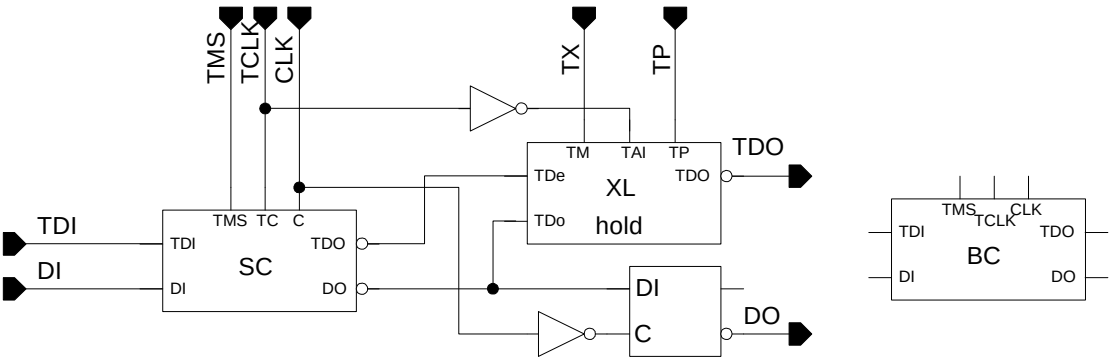

Abb. 46: Die Speicherzelle für das synchrone BILBO Konzept.

- **Die vorgestellte BILBO-Zelle für synchrone Schaltungen hat den Vorteil, dass der kritische Pfad nicht durch Multiplexer unterbrochen wird. Performanceverluste durch einen testfreundlichen Entwurf bei synchronen Schaltungen werden dadurch vernachlässigbar.**

4.2.3 Elemente zur Steuerung der Testdatenregister

Es werden die notwendigen Grundelemente für die Ablaufsteuerung erklärt. Sie enthalten das *Muller C-Element* zum testfreundlichen Entwurf, hier TMC-Element genannt[42]. Sie wurden in **[96Burd]$_{L.2}$** für unterschiedliche Transistorrealisierungen auf ihr dynamisches Verhalten mit Spice untersucht. Angaben zu Verzögerungswerten basieren auf dieser Untersuchung.

Da die seriellen Schieberegisteroperationen im passiven Scanpfad asynchron ablaufen sollen, werden Elemente zur Zustandserkennung benötigt. Sie sollen

[42] Die Wirkungsweise des TMC-Elements und der Einsatz für den asynchronen testfreundlichen Entwurf sind in **Kapitel 5** näher erklärt.

anzeigen, wann das Scanregister leer oder gefüllt ist. Somit erfolgt eine Berechnung des speichernden und transparenten Zustand eines *Capture-Pass* Registers anhand der Ein- und Ausgangssignale des *Muller C-Elements*. **Abb. 47** enthält eine Gatterrealisierung dieses Moduls (MCC, Muller C Control). **TRI** und **TAO** werden von den Eingängen und **TRO** vom Ausgang des *Muller C-Elements* abgegriffen. Danach erfolgt eine Verknüpfung mit den bereits ermittelten Werten aus den benachbarten Scanregisterzellen[43].

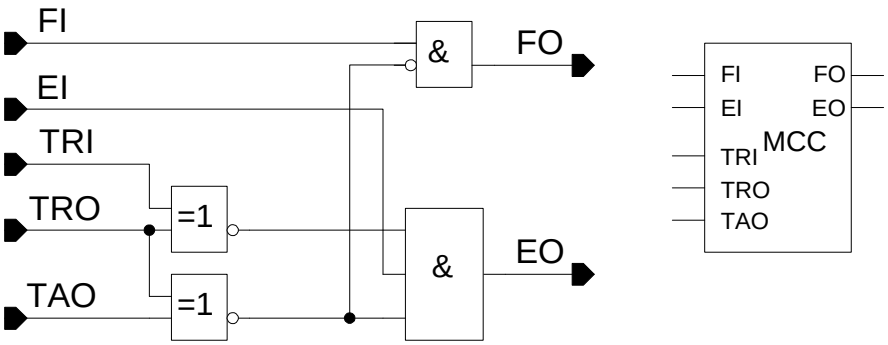

Abb. 47: Ein Modul erkennt, ob eine Registerzelle ein gültiges Datum enthält oder transparent ist.

Ein weiteres Modul für die asynchrone Testablaufsteuerung ist das TMC-Element, mit dessen Hilfe die *Bundled Data* Signale von dem Datenpfad in den Testpfad umgeschaltet werden. Sie ersetzen die *Muller C-Elemente* des Datenpfads. Es soll damit eine Minimierung der Performanceverluste bei der Verwendung eines Komplexgatters erreicht werden. Dazu werden zwei Multiplexer zum Umschalten der Eingangssignale des *Muller C-Elements* integriert. **Abb. 48** stellt eine mögliche Transistorrealisierung mit einer einstufigen Eingangsschaltung dar[44].

[43] Für *Four-Phase* Signale vereinfachen sich diese Elemente, da nur der Pegel des *Capture* Signals überprüft werden muß.

[44] Die Inverter zur Erzeugung der Signale **TSA_n** und **TSR_n** sind aus Übersichtsgründen nicht dargestellt.

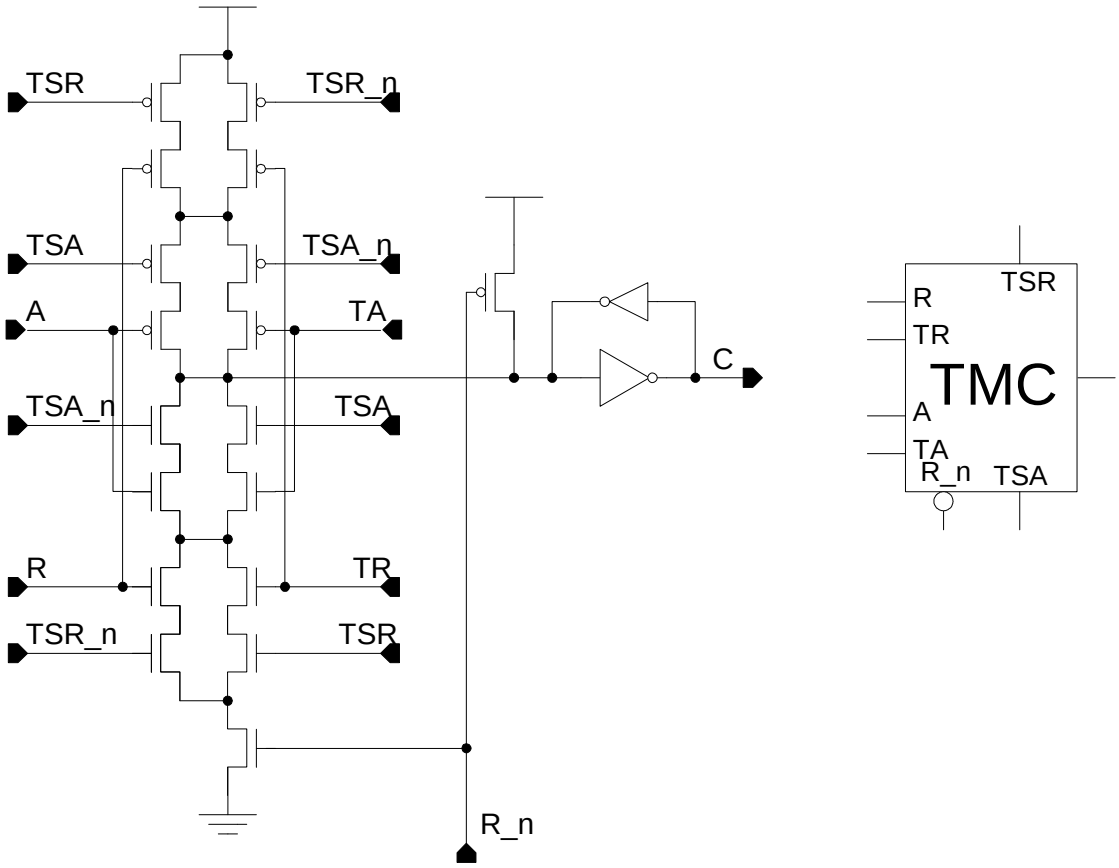

Abb. 48: Das TMC-Element mit einer einstufigen Eingangsschaltung (TMC1).

Abb. 49 zeigt eine weitere Transistorrealisierung von TMC mit *Pass* Transistoren. Die zweistufige Eingangsschaltung reduziert dabei die Anzahl der Transistoren bei mehr als zwei Eingangssignalen. Als Nachteil erweist sich die schlechtere Testbarkeit der *Pass* Transistoren durch ihre parallele Anordnung zu der ersten, einstufigen Realisierung. Ein Fehler an einem der *Pass* Transistoren verursacht kein logisches, sondern ein dynamisches Fehlverhalten des TMC-Elements[45]. Dieser Fehler reduziert die Schaltungsgeschwindigkeit, kann aber nicht zur Verletzung der *Bundled Data Convention* führen.

[45] Dieses Fehlverhalten tritt dann auf, wenn einer der *Pass* Transistoren immer sperrt.

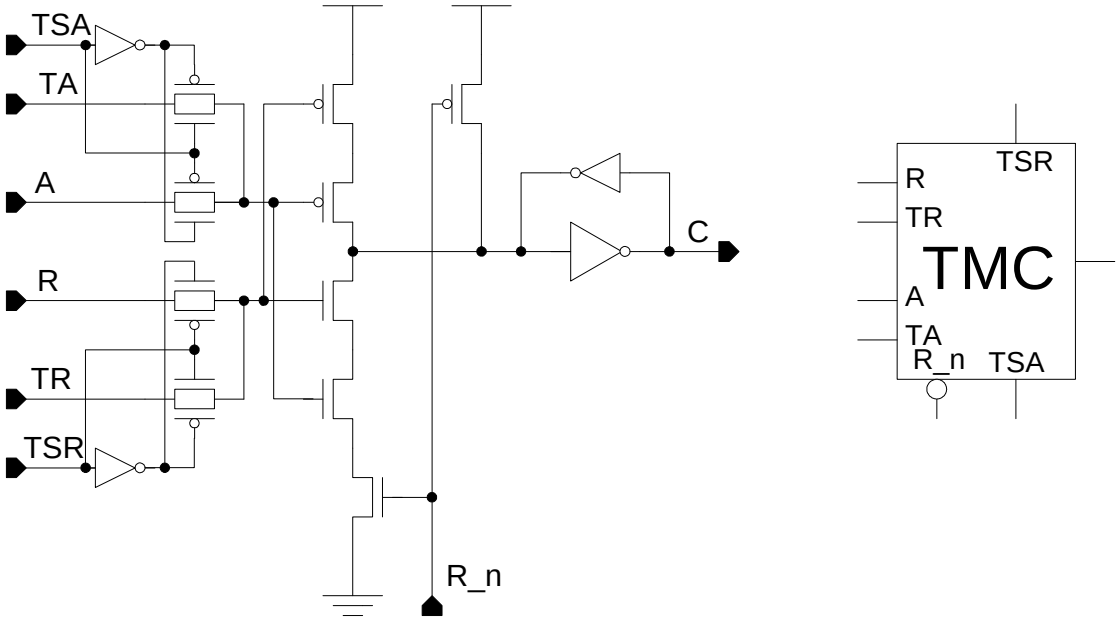

Abb. 49: Das TMC2-Element zum Umschalten der Steuersignale im Datenpfad.

Für das Speichern des Ausgangswerts wird in beiden Realisierungen ein schwach treibender Rückkopplungsinverter eingesetzt. Dieser hält das Potential, wenn keiner der Pfade zu VDD oder GND durchgeschaltet ist. Während des Umladens des Ausgangspotentials fließt für kurze Zeit ein Querstrom von VDD nach GND, da zwei invertierte Pfade auf den internen Knoten treiben. Hat der Ausgang seinen neuen Wert erreicht, stellt sich der Ruhestrom in CMOS Technik ein. Dieser Querstrom ist in Low Power Anwendungen unerwünscht und kann durch zwei Verfahren reduziert werden. Das erste Verfahren sieht einen hochohmig schaltbaren Rückkopplungsinverter vor. Bei dem TMC-Element werden für die Erzeugung des Signals für den hochohmigen Zustand eine große Anzahl von Transistoren benötigt.

Eine weitere Methode wurde für *Muller C-Elemente* eingesetzt **[95FarnEL]**$_{L.2}$. Hier werden Transistoren in den Pfad nach VDD bzw. GND des Inverters eingeführt. Die zusätzlichen Transistoren sind als hochohmige Widerstände angeschlossen und dienen der Reduktion der Treiberleistung des rückgekoppelten Inverters. Mit Hilfe einer dieser Methoden kann die Stromaufnahme des TMC-Elements nochmals reduziert werden. Das Zeitverhalten beider TMC-Elemente im Vergleich zu einem *Muller C-Element* ist in **Tabelle 3** aufgeführt.

Tabelle 3: Transistoranzahl und Geschwindigkeit des TMC-Elements

Name	Transistoren	Verzögerung	Mittlere Verzögerung
MC	10	0,57 ns - 0,36 ns	0,48 ns
TMC1	26	0,9 ns - 0,44 ns	0,5 ns
TMC2	22	0,9 ns - 0,58 ns	0,72 ns

4.3 Ergebnisse zu den asynchronen Standardzellen

Asynchrone Standardzellen für einen testfreundlichen Entwurf sind für eine asynchrone Schaltungsbibliothek notwendig. Daher wurden Standardzellenvorschläge für den Steuer- und Datenpfad entwickelt. Als Grundlage zum Entwurf dienten dabei die Anforderungen an Schaltungselemente, die aus den in **Kapitel 5** vorgestellten asynchronen Testkonzepten entwickelt wurden.

- **Es wurden Scanregisterzellen und BILBO-Registerzellen für einen testfreundlichen Entwurf mit einer asynchronen Steuerung vorgestellt.**

- **Beim testfreundlichen Entwurf der Registerzellen treten keine Performanceverluste im Datenpfad auf. Vielmehr steuern sie die Inhalte über den Rückkopplungspfad des Speicherelements.**

- **Datenausgang und Testdatensausgang sind bei den testfreundlichen Registerzellen getrennt. Zusätzliche kapazitive Lasten des Testpfads reduzieren die Performance der Schaltung nicht.**

- **Im asynchronen Steuerpfad können zusätzliche Signalverzögerungen im *Muller C-Element* auf ein Minimum reduziert werden.**

Je nach Realisierung des TMC-Elements wird keine bzw. eine neue Stufe in das *Muller C-Element* eingefügt. Die sich ergebenden Performanceverluste durch die zusätzliche Stufe können durch eine andere Dimensionierung des Verzögerungselements im Steuerpfad erreicht werden. Lediglich Verzögerungen im *Acknowledge* Pfad reduzieren die Geschwindigkeit der Schaltung. Darüber hinaus wurde die Funktionsweise eines Elements zur Überprüfung, ob ein ereignisgesteuertes Register ein gültiges Datum enthält, vorgestellt. Dieses Element bildet die Basis zur Umschaltung eines asynchronen Schieberegisters von dem seriellen in den parallelen Betrieb mit lokalen Steuersignalen.

Da die testfreundlichen Register unter dem Gesichtspunkt möglichst geringer Performanceverluste entwickelt wurden, sind einige dieser Elemente auch interessant für den synchronen testfreundlichen Entwurf. Die gewonnenen Erkenntnisse im asynchronen testfreundlichen Entwurf wurden auf synchrone Standardzellen übertragen. Auch bei diesen Elementen befindet sich die Einkopplung nicht im kritischen Datenpfad. Im Vergleich zu bisherigen getakteten Scanpfadzellen zeigt sich der Vorteil der Einkopplung der Testsignale über den Rückkopplungspfad, da auch hier bisher Multiplexer in den kritischen Pfad eingefügt werden, um Testsignale einzukoppeln.

- **Die Performanceverluste des testfreundlichen Entwurfs bei synchronen Schaltungen können mit den vorgestellten Registerzellen vernachlässigt werden.**

Für sogenannte primitive asynchrone Standardzellen des Steuer- und Datenpfads - wie *Muller C-Element*, XOR, *Mutual Exclusion* Element und Multiplexer - wurde eine Lösung zur Fehlermodellierung auf der Gatterebene angeboten. Als Basis diente dabei die Annahme, dass sich Defekte einer digitalen Schaltung auf der Transistorebene als *Shorts* zwischen den Anschlüssen oder als *Stuck-Opens* an den Ein- und Ausgängen der Transistoren manifestieren. Die Abbildung der Fehler von Transistorebene auf Gatterebene zeigte, dass sich die Fehler nicht ausschließlich in Form von Haftfehlern abbilden lassen. Zusätzlich zu dem klassischen Einzelhaftfehlermodell wurden daher die Fehlermodelle Funktionskonvertierung, Fehler mit Bedingung, Leckstromfehler und dynamische Fehler verwendet, um alle Fehlerauswirkungen auf die digitale Ebene abzubilden. Es wurde dabei versucht, die Anzahl der zu testenden Leckstromfehler zu minimie-

ren, da sich die Zeitpunkte, in denen sich die Schaltung in Ruhe befindet, bei asynchronen Schaltungen schwer einstellen lassen. Hier macht sich das Fehlen eines Takts bemerkbar. Die Ergebnisse der Fehlermodellierung sind in **Tabelle 4** zusammengefaßt[46].

Tabelle 4: Ergebnisse der Fehlermodellierung der primitiven Elemente

Fehlermodell	XOR	Muller C	MUTEX	MUX pass	MUX tri	Min	Max	Mittel
Haftfehler an den Ein- und Ausgängen	26%	50%	9%	7%	15%	7%	50%	21%
Funktionskonvertierung	21%	27%	16%	25%	15%	15%	27%	21%
Fehler mit Bedingung	38%	13%	75%	43%	30%	13%	75%	40%
IDDQ	10%	0%	0%	25%	30%	0%	30%	13%
Dynamisch	5%	10%	0%	0%	10%	0%	10%	5%

- **Ein systematischer Ansatz zur hierarchischen Fehlermodellierung für komplexe asynchrone Standardzellen wurde vorgeschlagen. Diese Standardzellen lassen sich aus den zuvor entwickelten primitiven Elementen und boolschen Funktionen wie INV, AND und OR aufbauen. Somit wurde ein effektiver Weg aufgezeigt, um eine Fehlerbibliothek für asynchrone Schaltungselemente für eine Fehlersimulation bzw. eine Testmustergenerierung vorzubereiten.**

[46] Bei MUTEX sind nur die Ergebnisse der Transistorfehlermodellierung untersucht. Das Speicherelement vor der Inverterkette ist in diesem Ergebnis nicht enthalten.

Kapitel 5

Testfreundliche Entwurfsmaßnahmen für asynchrone Schaltungen

Bevor die testfreundlichen Entwurfsmaßnahmen näher beschrieben werden, erfolgt eine allgemeine Betrachtung zum Steuer- und Datenpfad der zum testfreundlichen Entwurf vorgesehenen *Micropipeline*[47]. Die daran anschließenden Ergebnisse werden in zwei Unterkapitel gegliedert. Zunächst wird ein passives testfreundliches Entwurfskonzept für asynchrone Schaltungen vorgestellt, mit dessen Hilfe man die Steuer- und Beobachtbarkeit erhöhen kann. Aufbauend auf diesen Ergebnissen wird anschließend ein aktives testfreundliches Entwurfskonzept entwickelt, um Testmuster zusätzlich in den Schaltungen generieren und kompaktieren zu können.

Das Verzögerungselement im Steuerpfad zwischen zwei Registerstufen muß so dimensioniert sein, dass die Laufzeit der Signale im Steuerpfad stets größer ist als die der Signale im Datenpfad[48]. Der Schaltungsentwickler muß das Steuersignal des Senders so lange verzögern, bis die Daten an der nächsten Registers-

[47] Vergleiche hierzu **Kapitel 3** und **Kapitel 4**.

[48] Siehe hierzu auch **Abb. 18**.

tufe stabil anliegen. Die Überprüfung stellt das größte Testproblem dar[49]. Die kombinatorischen Module haben in der Regel kein Signal, mit dem der Abschluss einer Berechnung bzw. die Verfügbarkeit eines neuen Ausgangssignals quittiert wird. Dies ist ein großer Nachteil der *Single-Rail* Technik.

Im einfachsten Fall wird ein Verzögerungselement eingebaut, welches eine größere Verzögerung als der längste funktionale Pfad des aktuellen Datenpfads besitzen muß. Hieraus folgt, dass der Schaltungsentwickler die Möglichkeit besitzen muß, den längsten funktionalen Pfad bestimmen und ihn von den strukturell möglichen Pfaden unterscheiden zu können, welches einen erheblichen Aufwand bedeuten kann **[95Mahl]**$_{L.1}$. Eine spätere Korrektur dieses Verzögerungselements nach der Produktion ist nicht möglich, da eine Kalibrierung der Verzögerungselemente zum heutigen Zeitpunkt nicht vorgesehen ist. Dieses Verzögerungselement erzeugt ein großes Testproblem in der Schaltung mit *Bundled Data Convention*. Der Schaltungsentwickler wird versuchen, die zusätzliche Sicherheit bei der Verzögerung des Steuersignals zu minimieren. Hierdurch wird die Schaltung sehr sensibel auf Verzögerungsfehler im Datenpfad. Da Steuer- und Datenpfad nicht dicht beieinander auf der Schaltung platziert sein müssen, können auch lokale Prozessschwankungen eine Rolle spielen. Um die Schaltungen mit hoher Qualität zu testen, sollte man folgende Punkte beim testfreundlichen Entwurf asynchroner Schaltungen berücksichtigen:

Es müssen Testmuster angelegt werden können, die den Steuerpfad bei Haftfehlern "zum Halten bringen", um ein *Fail-stop* Verhalten zu erzeugen.

Testfreundliche Entwurfshilfen müssen Tests für nichtklassische Fehler berücksichtigen.

Es müssen Testmuster für statische und dynamische Fehler des Datenpfads über die Testhilfen angelegt werden können.

Der testfreundliche Entwurf muß den Test der *Bundled Data Convention* unterstützen.

[49] Diese Annahme bildet die Basis für *Bundled Data Convention* und muß unter allen Bedingungen erfüllt sein.

Beim synchronen Test können Scanregister seriell Daten ein- und auslesen oder parallel Daten verarbeiten. Eine Testablaufsteuerung schaltet zwischen diesen beiden Betriebsarten während eines Testablaufs um. Durch die lokale Steuerung asynchroner Schaltungen besteht zudem die Möglichkeit, den Umschaltvorgang nicht von außen zu steuern. Die Testmuster werden dabei seriell in das Scanregister eingelesen und parallel ausgegeben bzw. parallel eingelesen und seriell ausgegeben, ohne dass Signale von außen geändert werden müssen. Lokale Steuerungselemente entscheiden dabei, wann der Umschaltvorgang erfolgt. Zur Verdeutlichung dieses Prinzips werden die Betriebsarten in **Abb. 50** dargestellt.

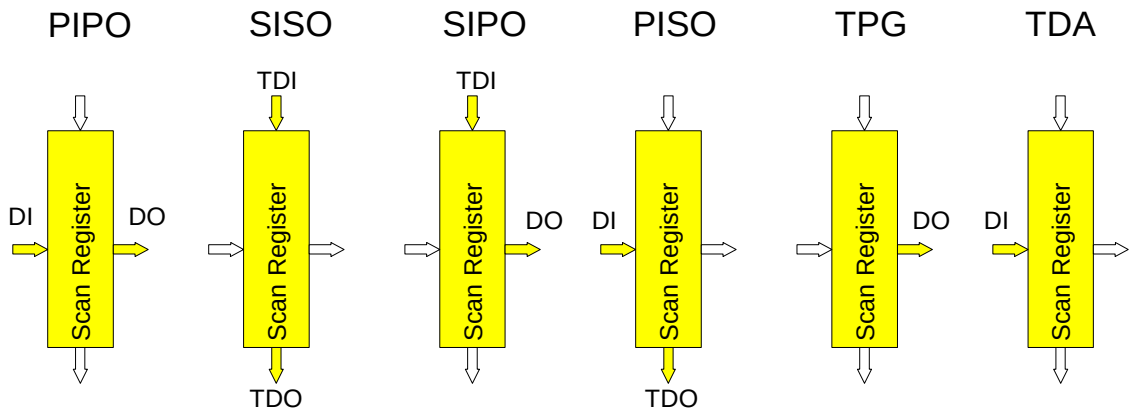

Abb. 50: Die unterschiedlichen Betriebsarten für testfreundliche Registerstrukturen.

Der Operationsbetrieb einer Schaltung wird dabei als PIPO (**P**arallel **I**n **P**arallel **O**ut) Betriebsart bezeichnet, der serielle Scanbetrieb als SISO (**S**erial **I**n **S**erial **O**ut) Betriebsart. Dieses sind die Betriebsarten, die auch bei synchronen testfreundlichen Entwurfsmethoden in Schaltungen und Systemen verwendet werden. Durch die lokale Steuerung der einzelnen Registerzellen sind zwei weitere Betriebsarten möglich. Diese werden mit SIPO (**S**erial **I**n **P**arallel **O**ut) für das Anlegen neuer Testmuster und PISO (**P**arallel **I**n **S**erial **O**ut) für das Auslesen von Testantworten bezeichnet. Der SIPO Betrieb ermöglicht es, seriell Daten in das Register mit Hilfe der *Bundled Data Convention* einzulesen. Sind alle Registerzellen mit neuen Testdaten über TDI gefüllt, schaltet eine lokale Testablaufsteuerung das Register in den parallelen Datenbetrieb[50]. Die Testdaten können dann über DO mit Hilfe der *Bundled Data Convention* durch ein *Request* Signal an den Datenpfad weitergegeben werden. Hierzu werden keine äußeren Signale benötigt. Analog können die Testdaten am Empfänger parallel

[50] Siehe hierzu **Abb. 47**.

über DI eingelesen, in den seriellen Pfad des Registers übertragen und über TDO mit der *Bundled Data Convention* ausgelesen werden. Mit *TPG* (Test Pattern Generation) wird die Betriebsart Generierung von Testmustern auf dem Chip bezeichnet. *TDA* (Test Data Analysis) bezeichnet den Testmodus zum Komprimieren der Testantworten innerhalb der Schaltung. Diese beiden Betriebsarten werden bei aktiven Testhilfen nach einer Initialisierungsphase eingestellt.

Um im Testbetrieb ein Register mit den nötigen Signalen zu steuern, müssen die Signale durch die Teststeuerlogik erzeugt werden, die im Operationsbetrieb von den vorherigen und den nachfolgenden Registern verarbeitet werden. Hierzu erfolgt eine Umschaltung. Das *Muller C-Element* des Datenpfads wird daher um zwei Multiplexer erweitert, wie in **Abb. 51** zu sehen ist[51]. Diese Modifikation erlaubt die Steuer- und Beobachtbarkeit der Registerinhalte mit der Synchronisationslogik des Datenpfads. Der Inverter an einem Eingang des TMC-Elements ergibt sich aus der Initialisierung einer *Micropipeline*. Der Ausgang C wird dabei für die *Bundled Data* Signale RO und AI verwendet, wie auch in **Abb. 54** zu sehen ist.

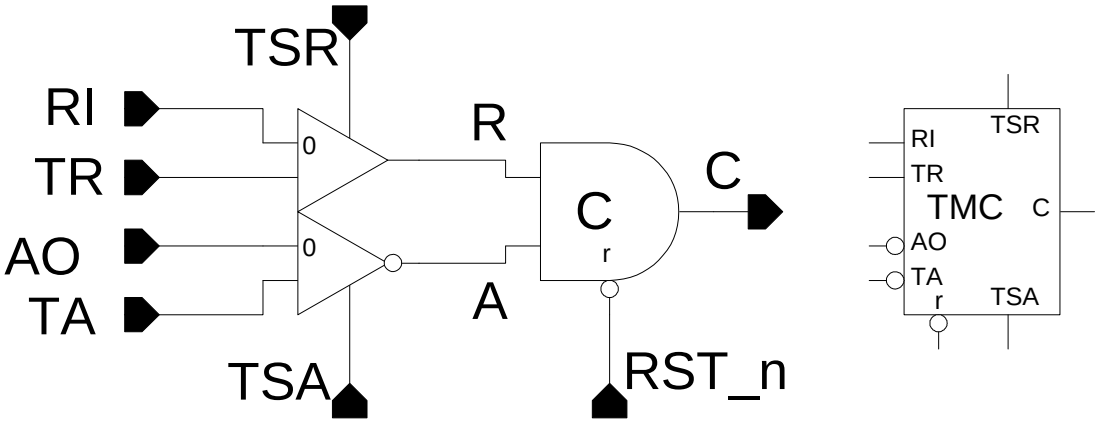

Abb. 51: Das testfreundliche *Muller C-Element* kann seine Steuereingänge durch Schalter multiplexen.

Zur näheren Erläuterung zeigt **Abb. 52** den Signalablauf des TMC-Elements in allen vier Betriebsarten. Der Multiplexer am Eingang R des *Muller C-Elements* erlaubt es, die *Request* Signale der vorhergehenden Stufe durch die der Testlo-

[51] Um den Schaltungsaufwand und die zusätzlichen Signalverzögerungen zu minimieren, ist ein Vorschlag für eine Standardzelle in **Kapitel 4** zu finden, der diese schematische Darstellung ersetzt.

gik zu ersetzen. Damit können Testdaten aus dem seriellen Scanpfad an den Datenpfad angelegt und das zugehörige Steuersignal für den Datenpfad generiert werden. Der Multiplexer an Eingang A dient zur Umschaltung des Registers für das Empfangen von Testdaten. Die Testablaufsteuerung des Empfängers speichert das Datum des parallelen Eingangs, wenn das *Request* Steuersignal mit einem Signalwechsel ein neues Testmuster am Empfängereingang anzeigt. Die Pfeile deuten dabei an, welche Signalwechsel zuvor erfolgt sind. Die vertikalen Balken gliedern die Abbildung in die verschiedenen Betriebsarten. Sie werden über die externen Signale TSA und TSR umgeschaltet.

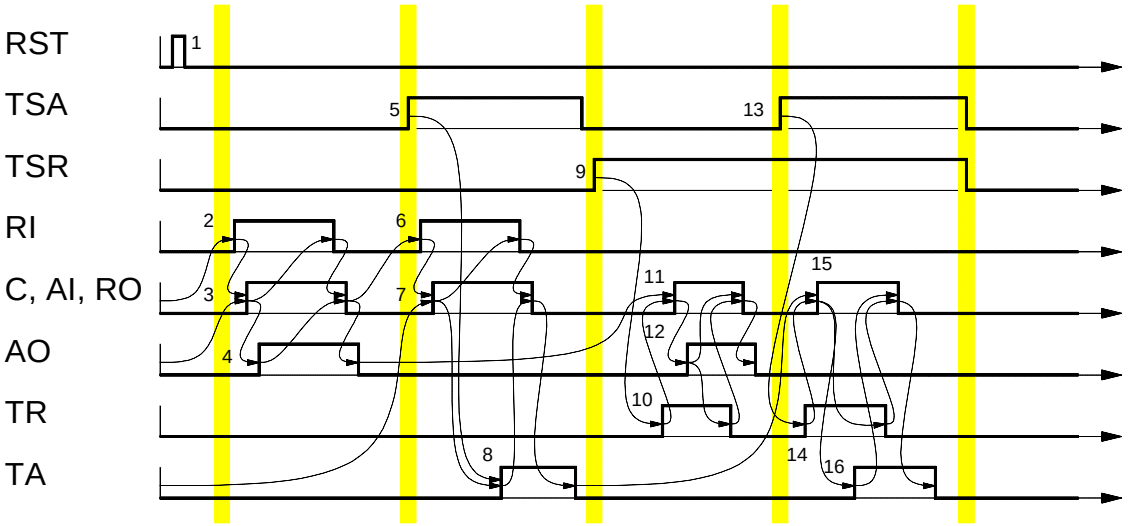

Abb. 52: Das Signalablaufdiagramm für das TMC-Element mit den unterschiedlichen Betriebsarten.

Nach der Initialisierung von TMC (**1**)[52] beginnt der Operationsbetrieb (PIPO Betriebsart) mit einem Signalwechsel von RI+ (**2**), welcher ein gültiges Datum am Eingang des Registers anzeigt. Mit einer positiven Signalflanke von C+ wird die Speicherung des Datums quittiert und zugleich das Signal AI+ an die vorhergehende und RO+ an die nächste Stufe weitergereicht (**3**). Mit AO+ (**4**) quittiert die nächste Stufe die Übernahme des Datums. Nach der parallelen Betriebsart PIPO wird der PISO Modus mit einer positiven Flanke des externen Signals TSA (**5**) eingeleitet. Nach RI+ (**6**) wird das Datum in dem Scanregister mit C+ (**7**) gespeichert. Zusätzliche Teststeuersignale quittieren den Vorgang des seriellen Ausleseprozesses mit einer positiven Flanke von TA+ (**8**). Mit den Signalwechseln TSA und TSR (**9**) wird die SIPO Betriebsart erreicht. Hier nicht eingezeichnete Steuersignale schreiben ein Testmuster in das Register und beenden

[52] Der Ausgang wird hierbei auf "0" gesetzt.

diesen Vorgang mit dem Signalwechsel TR+ (**10**). TR ersetzt dabei das Signal RI. Daraufhin wird das Signal C+ (**11**) am Ausgang des TMC-Elements erzeugt, das ein gültiges Datum am Registerausgang anzeigt. Mit AO+ (**12**) wird das Speichern der Testantwort in der nächsten Stufe quittiert. Sind TSA und TSR auf "1", ist der Steuerpfad im SISO Betrieb (**13**). Die Steuerung des Scanbetriebs erfolgt nur über Teststeuersignale TR und TA, wie es mit (**14-16**) beschrieben ist.

Der STG von TMC ist in **Abb. 53** dargestellt. Es ist mit der Multiplexerschaltung möglich, den Signalwechsel von RI+ und RI- im Testsendebetrieb durch TR+ und TR- zu ersetzen, wie im Signalablaufdiagramm in **Abb. 52** gezeigt wurde. Im Testempfängerbetrieb kann analog der Signalwechsel von AO+ und AO- durch TA+ und TA- ersetzt werden. In dem STG sind einige Signalwechsel nicht dargestellt, da sie für die *Bundled Data Convention* nicht benötigt werden. Die Umschaltung der Steuersignale TSA und TSR erfolgt in den Anfangszuständen von TMC, wenn alle Eingangssignale R=TR=0 und A=TA=0 sind.

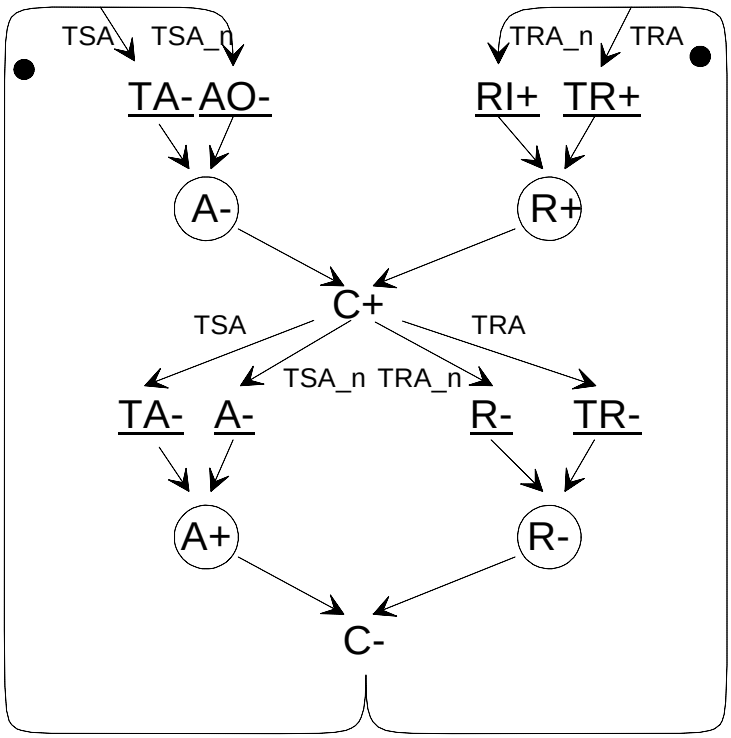

Abb. 53: Der Signal TransitionGraph des TMC-Elements für die *Bundled Data Convention*.

In dieser Abbildung ist eine neue Darstellung für Signalwechsel eingeführt. Sie beschreibt den Signalwechsel bei einer OR Verknüpfung von Signalwechseln. Der neue Signalwechsel wird dabei direkt in den Kreis eingetragen. Ein solches Modul kann mit einem XOR Gatter realisiert werden, wenn keine Bedingungen für die Signalwechsel angegeben sind[53]. In diesem Fall beschreibt die Funktion den Multiplexer am Eingang des *Muller C-Elements*. Dies ist möglich, da die Umschaltung der Steuereingänge TSA und TSR bei definierten Eingangspegeln erfolgt. Eine Umschaltung zwischen Daten- und Testbetrieb ohne Verlust der Initialisierungen des Steuerpfads ist bei diesem Konzept nicht möglich. Es würde einen enormen Aufwand an Steuerlogik erfordern und die Struktur des Datenpfads und die Geschwindigkeit der Schaltung stark beeinflussen, da für die Speicherung von Signalwechseln ein großer Schaltungsaufwand und eine detaillierte Analyse der neuen Signalwechsel nötig sind[54].

Integriert man die TMC-Elemente in den Steuerpfad, ergibt sich eine Anordnung mit zwei Registerstufen, wie sie in **Abb. 54** dargestellt ist. Die Steuersignale mit gestrichelten Linien sind in dieser Konfiguration des Tests inaktiv. Die fett dargestellten Signale werden zur lokalen Synchronisation der Schaltungsblöcke im Testbetrieb verwendet. Durch die TMC-Elemente werden die beiden Steuerleitungen des Operationsmodus' zwischen den Registerstufen zur Synchronisation eingesetzt. Darüber hinaus wird einer der beiden Eingänge der Ablaufsteuerung des Datenpfads (RI oder AO) im Testbetrieb verwendet und somit in den Test integriert. Im Testbetrieb und im Operationsbetrieb erfolgt dies mit der *Bundled Data Convention*. Es hat den Vorteil, dass der Steuerpfad parallel mit dem Datenpfad und der *Bundled Data Convention* getestet werden kann.

[53] Dies führt zu einer kompakten Darstellung der STGs, da auf vergleichbare Weise die AND-Verknüpfung bei Signalwechseln dargestellt ist.

[54] Hierfür eignet sich eher ein Schaltungsdesign mit *Four-Phase* Steuersignalen, da dort nur Zustände und keine Ereignisse gespeichert werden.

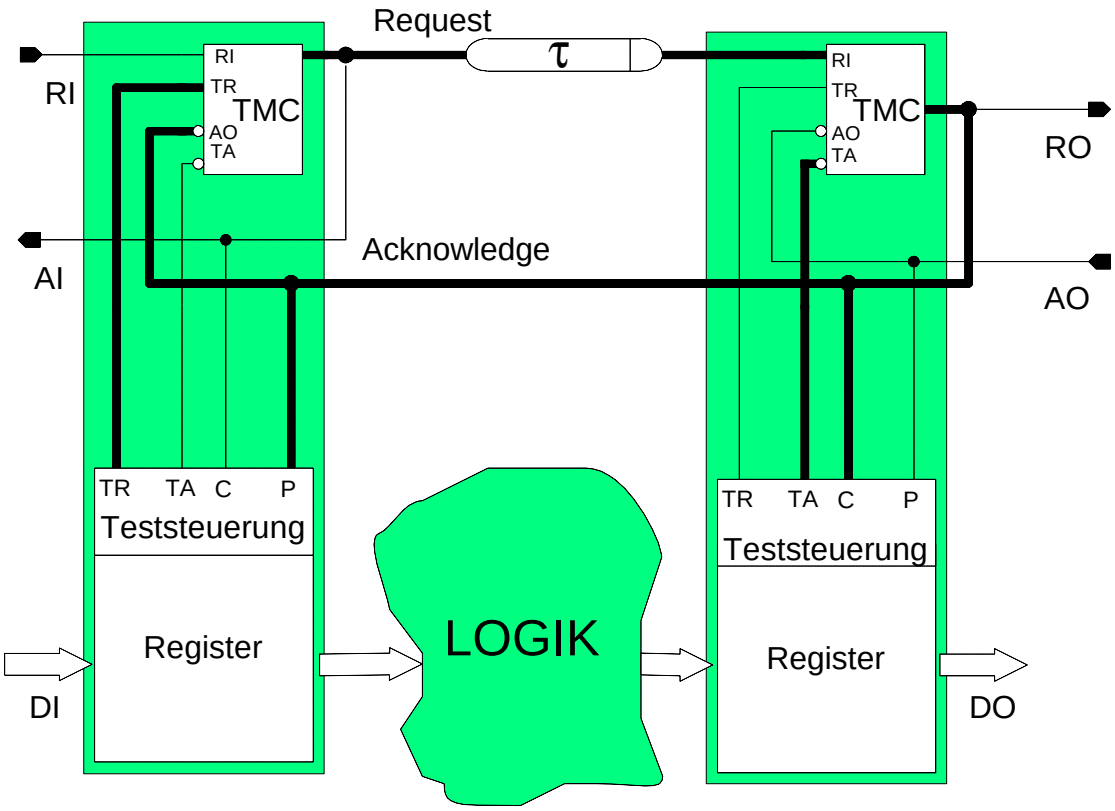

Abb. 54: Eine Übersicht zum Testablauf mit Hilfe der testfreundlichen asynchronen Register.

Alle weiteren Steuerelemente für die Testablaufsteuerung sind in dem Modul Namens Teststeuerung enthalten. Das Modul Register enthält die Erweiterungen zum testfreundlichen Entwurf des Datenpfads. Sie werden im folgenden in zwei verschiedenen Konzepten vorgestellt. Das erste Konzept stellt ein Scandesign für den passiven testfreundlichen Entwurf mit asynchronen Steuersignalen nach der *Bundled Data Convention* sowie im parallelen Daten- und im seriellen Testpfad dar. Das zweite Konzept beschreibt einen testfreundlichen Entwurf für asynchrone Schaltungen für den Built-In Self-Test. Auch hier wird der Datenpfad im Testbetrieb mit der *Bundled Data Convention* synchronisiert. Die Vor- und Nachteile der beiden Konzepte werden im Anschluss dargelegt.

5.1 Passive testfreundliche Entwurfshilfen

Unter passiven testfreundlichen Entwurfshilfen wird hier die Möglichkeit verstanden, die Steuer- und Beobachtbarkeit in Modulen innerhalb einer Schaltung zu erhöhen, um mit Hilfe von externen Geräten gesteuerte Tests und Diagnosen durchzuführen. Testmuster können an die Eingänge der Module über zusätzliche Testdatenkanäle angelegt und die Ergebnisse an Testdatenausgängen beobachtet werden. Test- und Diagnosemuster können dadurch an eingebettete Module angelegt werden[55].

In der synchronen Schaltungstechnik hat sich dafür der Scanpfad als effektives Entwurfskonzept durchgesetzt. Dies konnte durch eine globale synchrone Taktsteuerung der Speicherelemente ermöglicht werden. Durch serielle Datenpfade werden Testmuster bereitgestellt. Multiplexer an den Speicherelementen applizieren sie an die gewünschte Logik. Ziel dieses Kapitels ist es, eine vergleichbare Architektur für asynchrone Schaltungen zu entwickeln. Sie soll jedoch im Gegensatz zu einer globalen Taktsteuerung eine lokale Steuerung mit *Handshake* Signalen besitzen. Diese testfreundliche Registerarchitektur wird im folgenden als HIOB Modul (**H**andshake **I**nterface **O**bserver **B**lock) bezeichnet[56].

Die Erhöhung der Steuerbarkeit und Beobachtbarkeit durch parallele Multiplexer, die die Signale direkt mit den Ein- und Ausgängen verbinden, wird hier nicht betrachtet. Die Nachteile durch den zusätzlichen Flächenaufwand, die zusätzlichen primären Eingänge und der Performanceverlust bei diesem Prinzip des Freischaltens sind zu groß. Desweiteren können elementare asynchrone Schaltungsfunktionen mit dieser Methode nicht überprüft werden. Die Erhöhung der Steuer- und Beobachtbarkeit erfolgt bei diesem Entwurf durch die Modifikation von Registerstufen. Um den Verdrahtungsaufwand zu minimieren, werden die Testmuster hierbei seriell in die Schaltung ein- und ausgelesen. Die Umschaltung

[55] Hierzu wurde in **Kapitel 2.2.2** die Initialisierung der *Micropipelines* eingehend beschrieben.

[56] Der Name wurde in Anlehnung an den gleichnamigen Roman von Joseph Roth gewählt **[30Roth]**$_{L.3}$.

zwischen Test- und Datenpfad erfolgt innerhalb der Registerzellen. Hierfür wird eine Ablaufsteuerung benötigt, die sich in die komplexe Steuerung asynchroner Schaltungen einbinden läßt. Es soll dabei für die Synchronisation der Registerzellen im Daten- und im seriellen Testpfad die *Bundled Data Convention* verwendet werden.

Bei einem 16-Bit Testmuster würde das Scanregister im SIPO Modus 16 Schiebeoperationen ausführen, bevor ein neues Testmuster an den Ausgang weitergeleitet wird. Im PISO Modus erfolgen 16 serielle Schiebeoperationen, bevor eine neue Testantwort über den parallelen Eingang eingelesen wird. Durch die *Bundled Data Convention* ist eine flexible Art der Datenübertragung von parallelen auf serielle Datenpfade und wieder zurück möglich. Die Registerketten können somit aneinandergereiht werden, ohne dass globale Steuersignale für die Umschaltung zwischen den Übertragungsarten seriell und parallel notwendig sind.

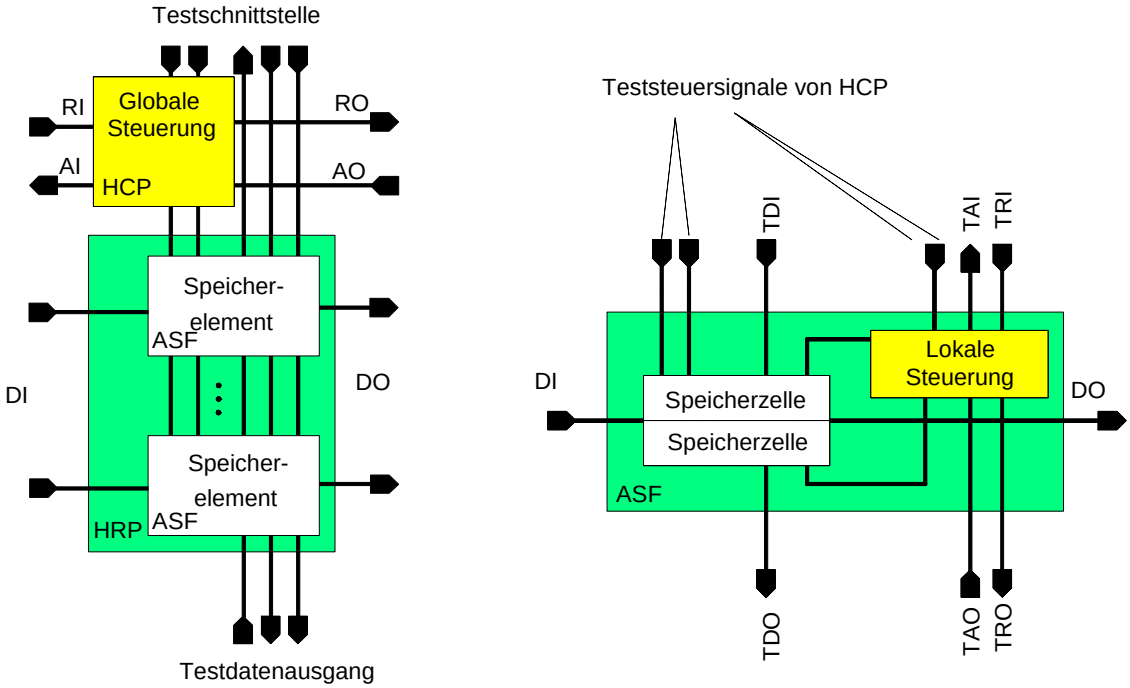

Abb. 55: Der prinzipielle Aufbau des passiven HIOB Scanpfadkonzepts.

Um dies zu ermöglichen, ist neben einer lokalen Steuerung der einzelnen Registerstufen ein globales Steuerelement für das ganze Scanregister notwendig. **Abb. 55** zeigt die prinzipielle Struktur des asynchronen Scanpfads. Die Steuerung für den Datenpfad (HCP) enthält die Steuerelemente für den Operations-

modus, die Umschaltung zwischen seriellem Testbetrieb und parallelem Datenbetrieb sowie die Ansteuerung zur Initialisierung und die Beobachtung des seriellen Schieberegisters (HRP). Das Speicherelement (ASF) enthält ein Registerelement mit zwei Speicherzellen sowie das Steuerelement zur seriellen Schiebeoperation. Kontrolliert von TAI und TRI werden über TDI neue Testdaten eingelesen und über TDO abgegriffen. Hierbei werden die Steuersignale TRO und TAO verwendet. Zur Umschaltung zwischen den Testbetriebsarten werden zwei Signale benötigt, TSI und TSO. **Tabelle 5** beschreibt die Teststeuersignale, mit denen der Scanpfad angesteuert wird. Hier sind die eingangs erwähnten Betriebsarten mit einer kurzen Erklärung wiederzufinden. Das Zusammenwirken der Ein- und Ausgänge der Steuer- und Datensignale wird nun näher untersucht.

Tabelle 5: Die Teststeuersignale der passiven Teststeuerung.

TSI	TSO	Betriebsart	Erklärung
0	0	PIPO Modus	Operationsbetrieb, kein Testbetrieb
1	0	SIPO Modus	Testmuster anlegen
0	1	PISO Modus	Testdaten auslesen
1	1	SISO Modus	Schieberegisterbetrieb

Abb. 56 zeigt den Signalablauf der Steuersignale im Zusammenhang mit den Datensignalen für den Testbetrieb. Nach dem Operationsbetrieb, in dem die Daten (**D1**) von DI nach DO übertragen werden, wird der Testbetrieb mit TSO+ (**1**) in den PISO Modus eingeleitet. Es wird ein neues Datum mit RI+ angezeigt. Nachdem das Datum (**D2**) im Register mit dem Quittungssignal AI gespeichert wurde, schreibt der Scanpfad über TDO mit Hilfe von TRO und TAO die Testdaten seriell aus, bis die Pipeline geleert ist (**2-3**). Es wird das Quittungssignal TA (**4**) für den Abschluß der Operation erzeugt. Dieses Signal ersetzt somit das Quittungssignal der nachfolgenden Registerstufe des Datenpfads. Anschließend kann das nächste anliegende Datum (**D3**) über den Testdatenausgang ausgelesen werden. Wird die SIPO Betriebsart ausgeführt, muß zuerst das Testsignal TSI+ (**5**) wechseln, um Testdaten (**D4**) über TDI mit Hilfe von TRI und TAI einzulesen (**6-7**). Danach erfolgt das Quittungssignal TR+ (**8**), welches dem TMC-Element ermöglicht, den Signalwechsel RO+ an die nachfolgende Stufe zu senden. TR ersetzt somit das RI Signal von der vorhergehenden Registerstufe des Datenpfads. Ist das Datum der Stufe gespeichert, kann das nächste Test-

muster (**D5**) eingelesen werden. Wird der SISO Modus aktiviert, müssen beide Teststeuersignale auf "1" geschaltet werden (**9**). Danach können Testmuster seriell durch das Register geschoben werden. Die Daten (**D6**) werden über den Testdateneingang gelesen (**10-11**) und zeitversetzt am Testdatenausgang mit den zugehörigen Steuersignalen (**12-13**) abgegriffen. Im folgenden werden die Arbeitsweise und Struktur des Registers näher untersucht.

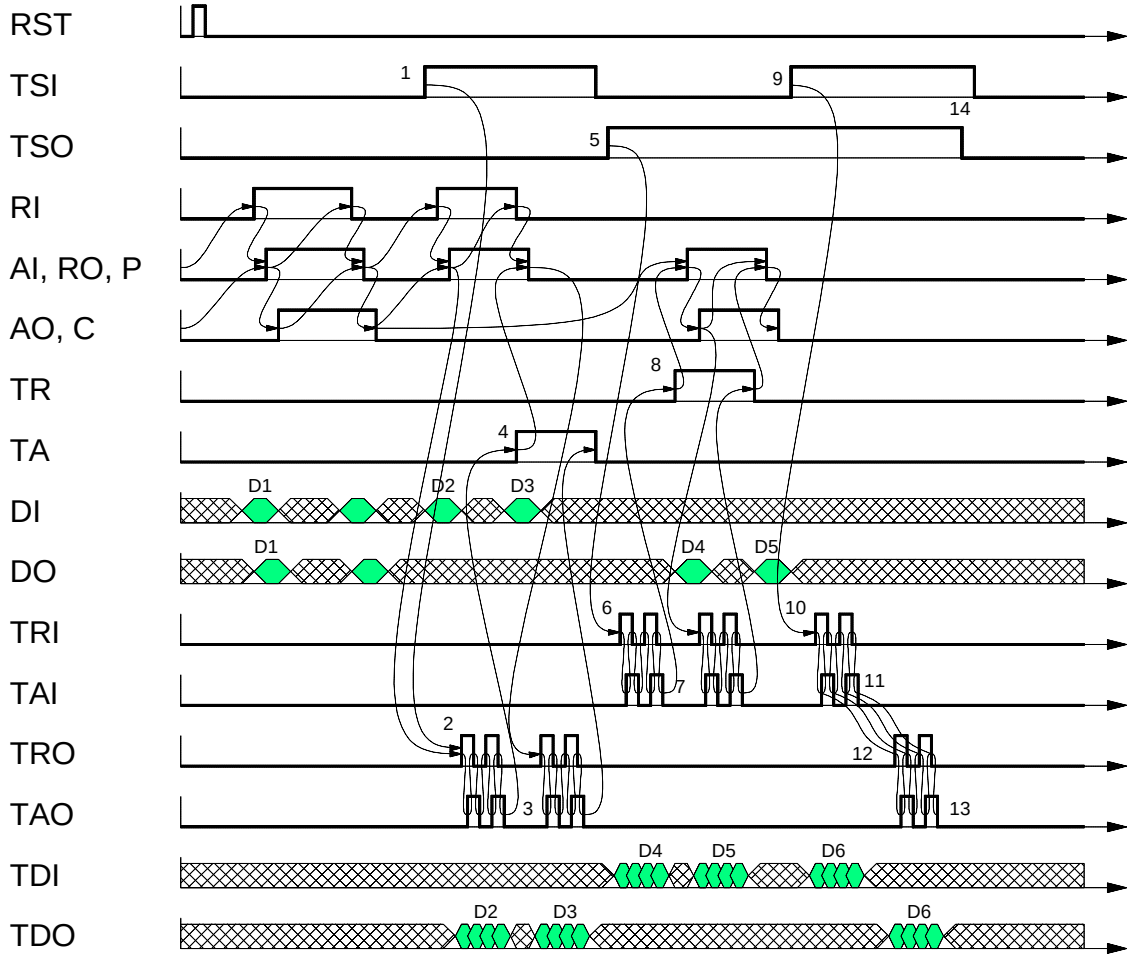

Abb. 56: Das Signalablaufdiagramm des passiven HIOB Konzepts.

5.1.1 Die Registerstruktur des Scanpfads

Das Register (HRP) besitzt zwei Dateneingänge, einen parallelen Eingang für den Operationsbetrieb (**DI**) und einen seriellen Eingang für den Testbetrieb (**TDI**). Hinzu kommen die Datenausgänge für den parallelen Datenpfad (**DO**) und den seriellen Testbetrieb (**TDO**). Die Verknüpfung des seriellen Scanpfads und des parallelen Datenbusses erfolgt in Registerstufen (**ASF**). HRP wird von der globalen Steuereinheit, HCP, des Datenpfads kontrolliert und beobachtet. Eine mögliche Realisierungsform ist in **Abb. 57** aufgebaut. Im Operationsbetrieb sind die Signale C (Capture) zum Speichern und P (*Pass*) zum Weiterleiten aktiv. Dies erfolgt analog zu der Beschreibung in **[89Suth]**$_{L.2}$.

Die Steuereingänge zum Setzen (**So_n, Se_n**) und Rücksetzen (**Ro_n, Re_n**) der *Muller C-Elemente* dienen zum Initialisieren des seriellen Pfads. Es wechseln sich dabei zwei unterschiedliche Registerarten ab, ASFe für die geraden und ASFo für die ungeraden Stellen des seriellen Pfads. Die Unterscheidung zwischen geraden und ungeraden Stellen ist notwendig, um das ganze Testmuster auf einer Seite im Register zu speichern[57]. An den Ein- und Ausgängen sind weitere Module in die Steuerung des Testpfads integriert. Die XORs dienen zur Pegelumwandlung der Steuersignale. Die *Muller C-Elemente* werden zum Halten des internen Zustands bei externen Signalwechseln verwendet, wenn die Steuerung von HRP beim Umschalten in dem seriellen Betrieb neu initialisiert wird. Mit TMS wird neben den Signalen Re_n, Ro_n, Se_n und So_n das Register zwischen den Daten und Testmodi umgeschaltet. Im Testbetrieb wird je nach Betriebsart der Testdateneingang mit den Signalen TDI, TRI und TAI sowie der Testdatenausgang mit den Signalen TDO, TRO und TAO verwendet. Im seriellen Pfad werden die Daten- und die Steuerleitungen zwischen den Registerstufen verbunden, wohingegen die Leitungen während des Operationsmodus' parallel angeschlossen sind. Mit den Signalen ED und FD wird der globalen Steuerung des Datenpfads signalisiert, ob ein Register mit neuen Daten gefüllt oder geleert ist. Die Signale Re_n, Ro_n, Se_n und So_n dienen zur Initialisierung der *Muller C-Elemente* des seriellen Datenpfads.

[57] Wie in **Kapitel 2.2.2** gezeigt wurde, ist das jeweilige *Muller C-Element* auf "0" oder "1" zu initialisieren. Dies ist von der Position innerhalb der Reihenfolge der *Micropipelines* sowie der Zustände der angrenzenden Register abhängig.

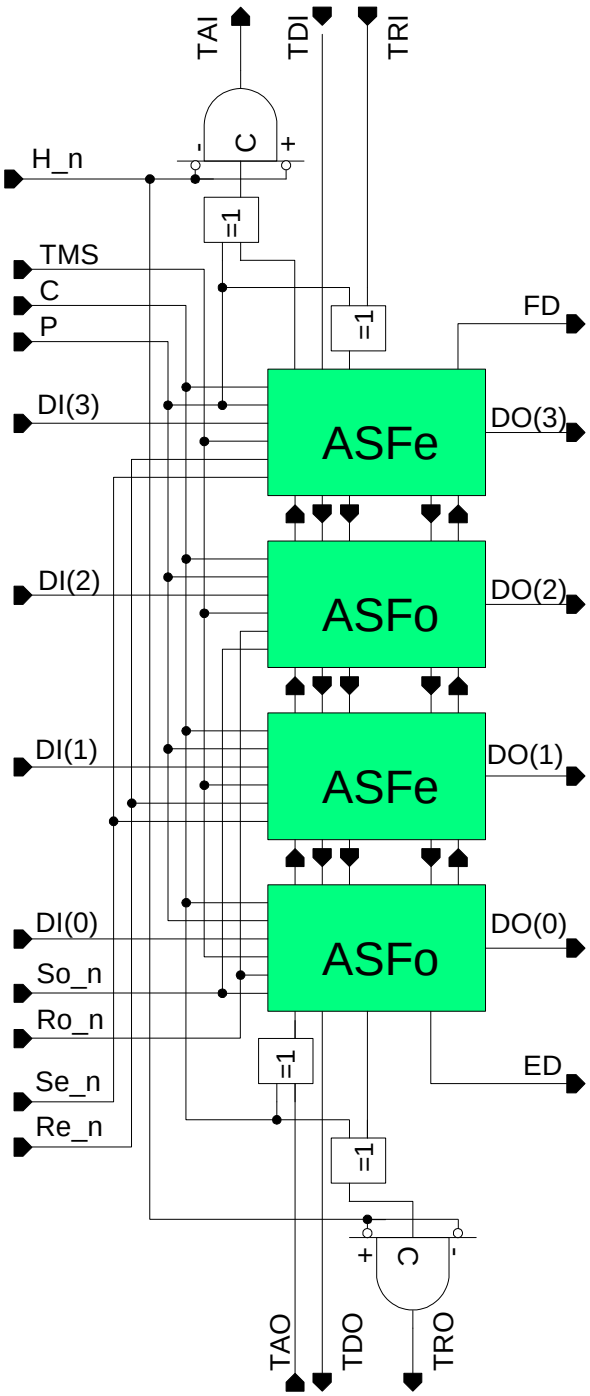

Abb. 57: Der Aufbau des asynchronen Scanregisters, HRP, des passiven HIOB Konzepts.

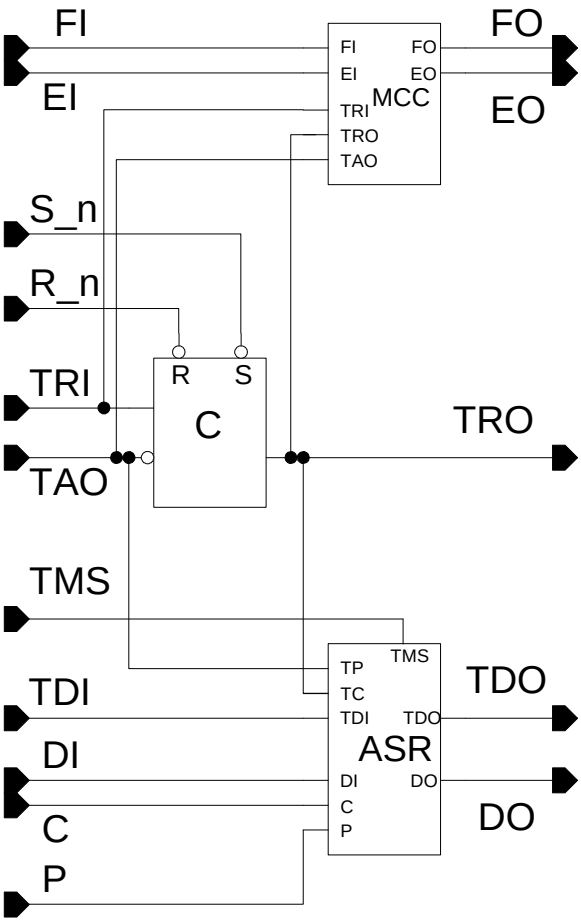

Abb. 58: Das asynchrone Scan Flip-Flop einer Registerstufe besitzt Datenspeicher und Steuerlogik.

Die ASF Zelle enthält die Speicherzelle (ASR) sowie die lokale Steuerung des seriellen Datenpfads. Die lokale Steuerlogik ist für die Generierung der Steuersignale der Testbetriebsarten notwendig. Sie teilt sich dabei in ein *Muller C-Element* und einen Logikblock (MCC) auf[58]. Mit dem Register (ASR) werden im Operationsbetrieb Daten gespeichert. Im Testbetrieb werden Testmuster eingelesen, in den Datenpfad geschaltet und wieder ausgelesen[59]. Da der Scanpfad auch eine asynchrone Ablaufsteuerung für den seriellen Betrieb besitzt, benötigt er für jede Scanzelle eine Ansteuerung, wie in **Abb. 58** gezeigt ist.

[58] Das MCC-Element wurde in **Kapitel 4** näher erläutert.

[59] Siehe hierzu **Kapitel 4**.

Das MCC Modul wird mit den Ein- und Ausgängen des *Muller C-Elements* verbunden. Über das Beobachten der Eingänge in Abhängigkeit von dem Ausgangssignal kann entschieden werden, ob ein aktuelles Datum gespeichert ist. Zwei Steuereingänge zeigen an, ob die vorherigen Scanzellen speichern (**FI**) oder geöffnet (**EI**) sind. Hieraus wird ein Signal erzeugt, welches anzeigt, dass diese und alle vorherigen Speicherzellen ein Datum gespeichert haben (**FO**). Ein weiteres Signal zeigt an, dass diese und alle vorherigen Speicherzellen leer sind (**EO**). Diese Informationen werden weitergegeben und über alle Elemente summiert. Hierdurch können Signale generiert werden, wenn der serielle Pfad des Registers vollständig gefüllt (**FD=1**), oder das Register zum Speichern von neuen Daten bereit ist (**ED=1**).

Zur Verdeutlichung des Prinzips der lokalen Steuerung ist in **Abb. 59** der Zustandsgraph des *Muller C-Elements* mit seinen instabilen und stabilen Zuständen dargestellt. Die stabilen Zustände sind dabei in speichernde und transparente Zustände des daran angeschlossenen asynchronen Registers aufgeteilt. Rechts ist die zugehörige Verdrahtung der beiden Elemente angegeben. Das MCC-Element soll dabei die Information, ob die vorhergehenden Stufen gefüllt oder transparent sind, an die nächste Registerstufe weitergeben. Beim Wechsel von einer transparenten zu einer vollen Speicherzelle werden die Eingangssignale EI oder FI betrachtet und gegebenenfalls ein FO oder EO Signal generiert.

Um alle Inhalte der Scanzellen des seriellen Pfads in den parallelen Datenpfad auslesen zu können, müssen alle in der "gleichen" Speicherzelle des Registerelements gespeichert sein. Dieses bezieht sich auf die "gerade" oder "ungerade" Speicherposition im Register. Im Gegensatz hierzu müssen alle *Muller C-Elemente* der seriellen Steuerung auf wechselnde interne Zustände gesetzt werden, um ein gültiges Datum im Scanpfad speichern und auslesen zu können. Dies erzwingt normalerweise aber eine unterschiedliche Speicherung der Testdaten in den geraden (even) und den ungeraden (odd) Speicherpositionen der Scanzellen und führt somit zum Widerspruch beim Umschalten zwischen dem seriellen und parallelen Betrieb. Um den Widerspruch zu lösen, werden die SC-Elemente der geraden und der ungeraden Binärstellen gegensätzlich verknüpft[60].

[60] Siehe hierzu auch **Kapitel 2.2.2**, in dem das Prinzip erläutert wird und **Kapitel 4**, in dem die Speicherzellen beschrieben werden.

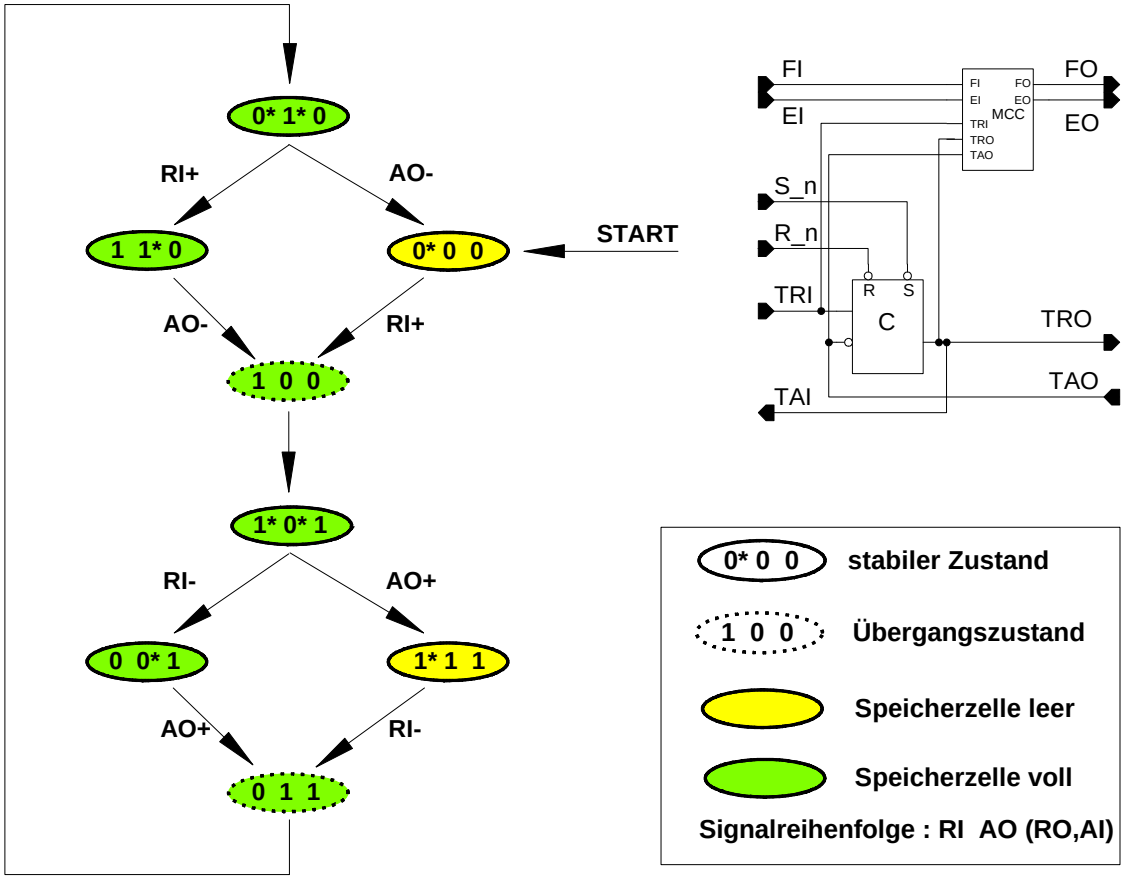

Abb. 59: Der Speicherzustand eines Registers wird über den Zustand seines *Muller C-Elements* ermittelt.

Bei der Umschaltung zwischen dem Test- und Datenpfad einer *Micropipeline* entsteht ein zusätzliches Problem bei *Two-Phase* Signalen. Besitzen der Testpfad eine gerade und der Datenpfad eine ungerade Anzahl von Registerstufen, muß eine korrekte Initialisierung der Register gewährleistet werden. Der Grund hierfür liegt in der Umsetzung von *Two-Phase* Signalen. Eine korrekte Ereignissteuerung ist in der CMOS Technik von Pegeln abhängig. Die Initialisierung des Steuerpfads mit definierten Pegeln ist dabei entscheidend, um nach der Initialisierung nur noch eine Ereignissteuerung zu betrachten. Bei den Betriebsarten SIPO und PISO soll diese Initialisierung automatisch durch die Testablaufsteuerung nach jedem Testmuster erfolgen. Dies würde zu einem Konflikt bei jedem zweiten Initialisierungsvorgang führen, da nur in diesen Fällen die Daten in der richtigen Zelle der Master Slave Struktur des Registers gespeichert werden. Um dieses Problem zu lösen, müssen die Pegel in geeigneter Form umgewandelt werden. Erfolgt das Einlesen eines neuen Testmusters seriell in ein Register bei einer geraden Anzahl von Speicherzellen, ist das gültige Datum immer in dem gleichen Latch des Master-Slave Flip-Flops gespeichert, da hierfür

eine gerade Anzahl von Schieberegisteroperationen notwendig ist. Der parallele Datenpfad benötigt jedoch die aufeinander folgenden Testmuster immer abwechselnd in den Latches des Flip-Flops. Dies muß beim Initialisieren des seriellen Pfads berücksichtigt werden und erfolgt über eine Pegelumwandlung mit Hilfe eines XOR Gatters in Verbindung mit einem *Muller C-Element*, wie in **Abb. 60** gezeigt.

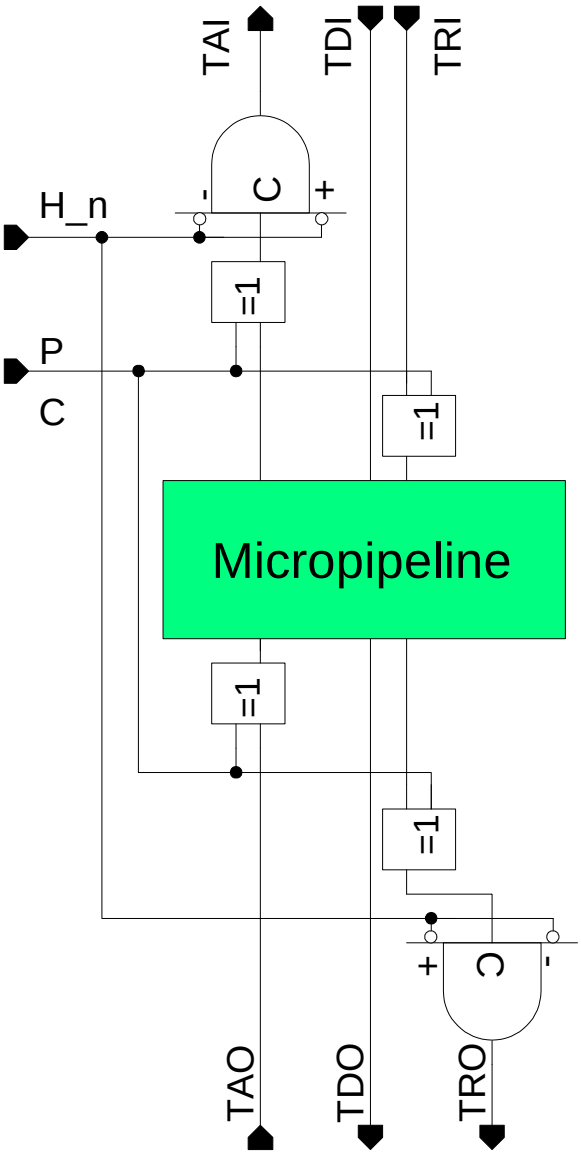

Abb. 60: Zur Pegelumwandlung bei *Two-Phase* Signalen ist zusätzliche Logik notwendig.

Das *Muller C-Element* ermöglicht das Halten der Ausgangssignale, bis der Initialisierungsvorgang abgeschlossen ist. Hierzu wird das Signal (H_n) verwendet, welches in der Zeit des Übergangs die Ausgänge der *Muller C-Elemente* stabilisiert. Das Signal H_n wird dabei aus den Setz- und Rücksetzsignalen generiert. Besitzen der serielle und der parallele Pfad beide eine gerade oder eine ungerade Anzahl von Registerstufen, ist keine Pegelumwandlung notwendig. H_n ermittelt sich somit aus (Ee + Eo + Fe + Fo)_n

Beim PISO Modus sind die Signale TRO und TAO von Bedeutung. TRO darf bei einer internen Pegelumwandlung keinen falschen Zustand am Ausgang erzeugen. Im SIPO Modus sind die TRI und TAI Signale von Bedeutung. Deshalb darf das TAI Testsignal bei einer Pegelumwandlung keinen Einfluß auf den seriellen Testpfad haben. So werden TRO im PISO Modus und TAI im SIPO Modus des seriellen Pfads so lange gehalten, bis die Initialisierung abgeschlossen ist. TRI und TAO müssen nicht auf gleiche Weise stabilisiert werden, da die Invertierung der Eingangssignale keine Auswirkung auf die *Micropipeline* hat, solange eines der Signale Re_n, Ro_n, Se_n aktiv ist[61].

5.1.2 Die Ereignissteuerung im Datenpfad

Die Ereignissteuerung des Scanregisters teilt sich in zwei voneinander getrennte Bereiche. Es wird eine globale Steuerung für das ganze Register benötigt, welche in **Abb. 55** mit "Globale Steuerung" bezeichnet ist. Hinzu kommt eine "Lokale Steuerung" für jede einzelne Registerzelle, welche die Taktsteuerung im seriellen Schieberegisterbetrieb ersetzt. Sie ist im vorherigen Kapitel beschrieben. Im folgenden wird die Ablaufsteuerung, mit der das Scanregister im Datenpfad gesteuert wird, dargestellt.

Zum Überblick über die verwendeten Teststeuersignale im passiven HIOB Konzept und deren Verbindung zur Registerstruktur, sind die Namen und deren Verschaltung in **Abb. 61** dargestellt. HCP enthält die Steuerung des Test- und des Datenpfads und teilt sich in das TMC-Element und eine globale Teststeuerung auf. TA und TR ersetzen im Testbetrieb die Steuersignale des Datenpfads

[61] Dies erfolgt analog zu den Überlegungen in **Kapitel 2.2.2**, in dem die Initialisierung näher untersucht wurde.

der vorhergehenden und der folgenden Registerstufe. Die Signale FD und ED sind die Quittungssignale der lokalen Steuerung, mit deren Hilfe die globale Steuerung erkennt, ob die Registerstufe voll (FD, Full Detection) oder leer (ED, Empty Detection) ist. Diese Signale werden mit Hilfe der Zustandselemente, MCC, generiert. ED und FD wechseln ihre Zustände bei der Initialisierung und beim Beenden einer Schiebeoperation. Im folgenden werden die Testablaufsteuerung (HCP) des Datenpfads und die Wirkungsweise der neuen Steuersignale erklärt.

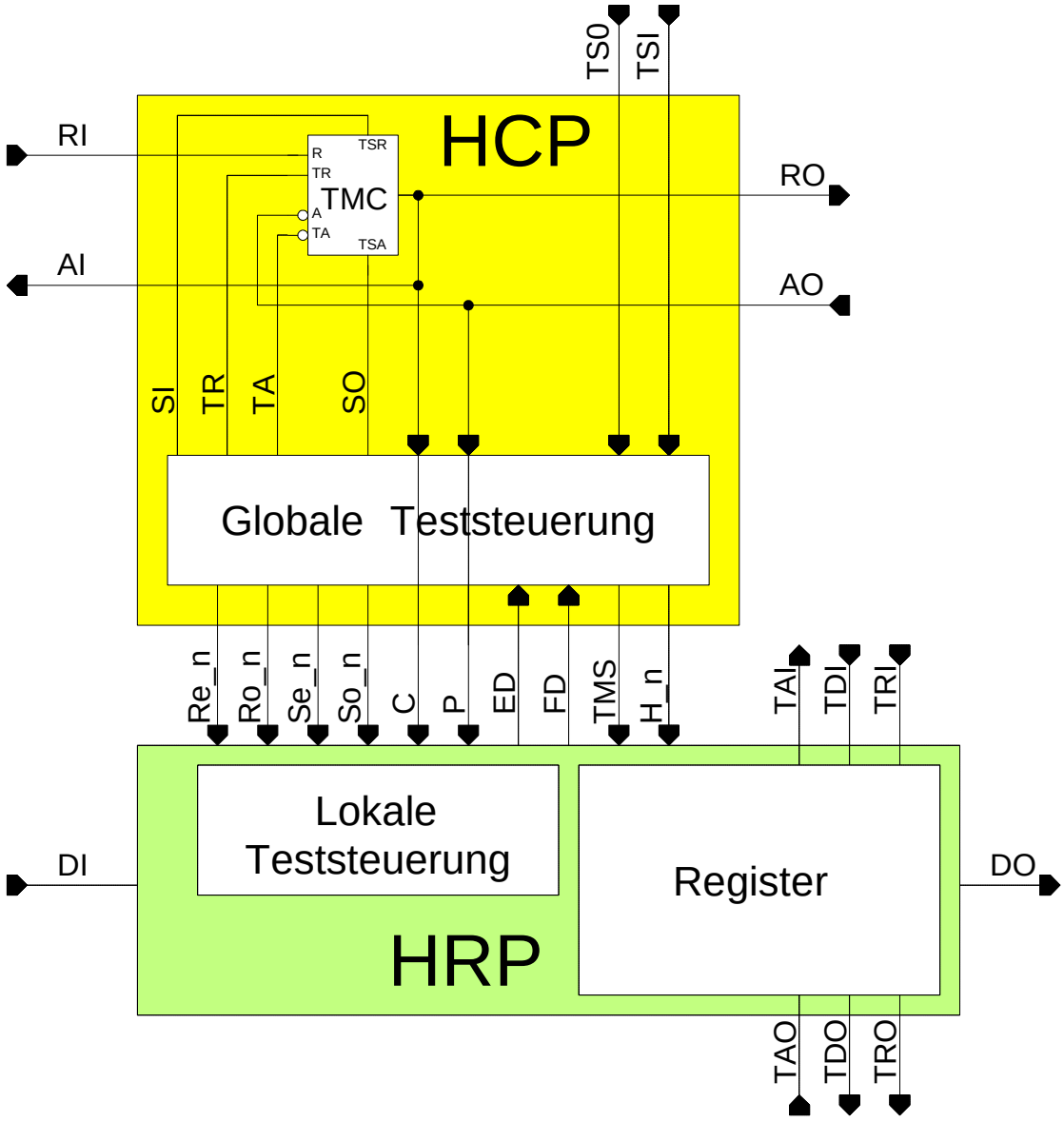

Abb. 61: Zur Steuerung der Registerzellen werden zusätzliche Signale benötigt.

Die Testablaufsteuerung muß die *Bundled Data Convention* berücksichtigen. Problematisch hierbei ist, dass alle Zellen eines Registers im Operationsbetrieb gleichzeitig schalten, während im Testbetrieb jede Registerzelle durch die serielle Schiebeoperation einzeln schaltet. In dieser lokalen Steuerung des Testpfads ist der Zustand enthalten, ob eine Registerzelle ein gültiges Datum gespeichert hat oder transparent geschaltet und somit bereit zur Speicherung eines neuen Datums ist.

Zur Verwendung einer Testablaufsteuerung für alle Betriebsarten werden interne Zustände benötigt. In **Tabelle 6** sind diese Zustände aufgeführt. Die Signale INIT, SO und SI werden eingesetzt, um Module der globalen Testablaufsteuerung in Abhängigkeit der Betriebsart in einem definierten Zustand zu halten. Sie setzen die Ausgänge der an sie angeschlossenen Elemente bei einer Schaltungsrealisierung zurück und halten sie auf einem definierten Potential. In den danach folgenden Signalablaufdiagrammen werden sie dargestellt, um die Initialisierung der Testablaufsteuerung zu verdeutlichen.

Tabelle 6: Interne Zustände der globalen Teststeuerung

TSI	TSO	INIT	SO	SI	Erklärung
0	0	1	1	1	Initialisierung, der parallele Operationsbetrieb ist aktiv
0	1	0	1	0	Der serielle Testdateneingang ist aktiv
1	0	0	0	1	Der serielle Testdatenausgang ist aktiv
1	1	0	0	0	Datenpfad ist inaktiv, der serielle Scanpfad ist aktiv

In **Abb. 62** ist das Signalablaufdiagramm der PISO Betriebsart dargestellt. TO und TMO sind, wie auch INIT, SI und SO interne Signale zur Testablaufsteuerung, die zur Umwandlung zwischen den *Two-Phase* der *Bundled Data Convention* und den internen *Four-Phase* Signalen dienen. TO steuert im PISO Modus das TMS Signal des Scanregisters, da dieses in dieser Betriebsart bei jedem Testmuster umgeschaltet werden muß, um die Testantworten aus dem Datenpfad in den Schieberegisterbetrieb zu übernehmen[62].

[62] Sie sind in dieser Betriebsart identisch.

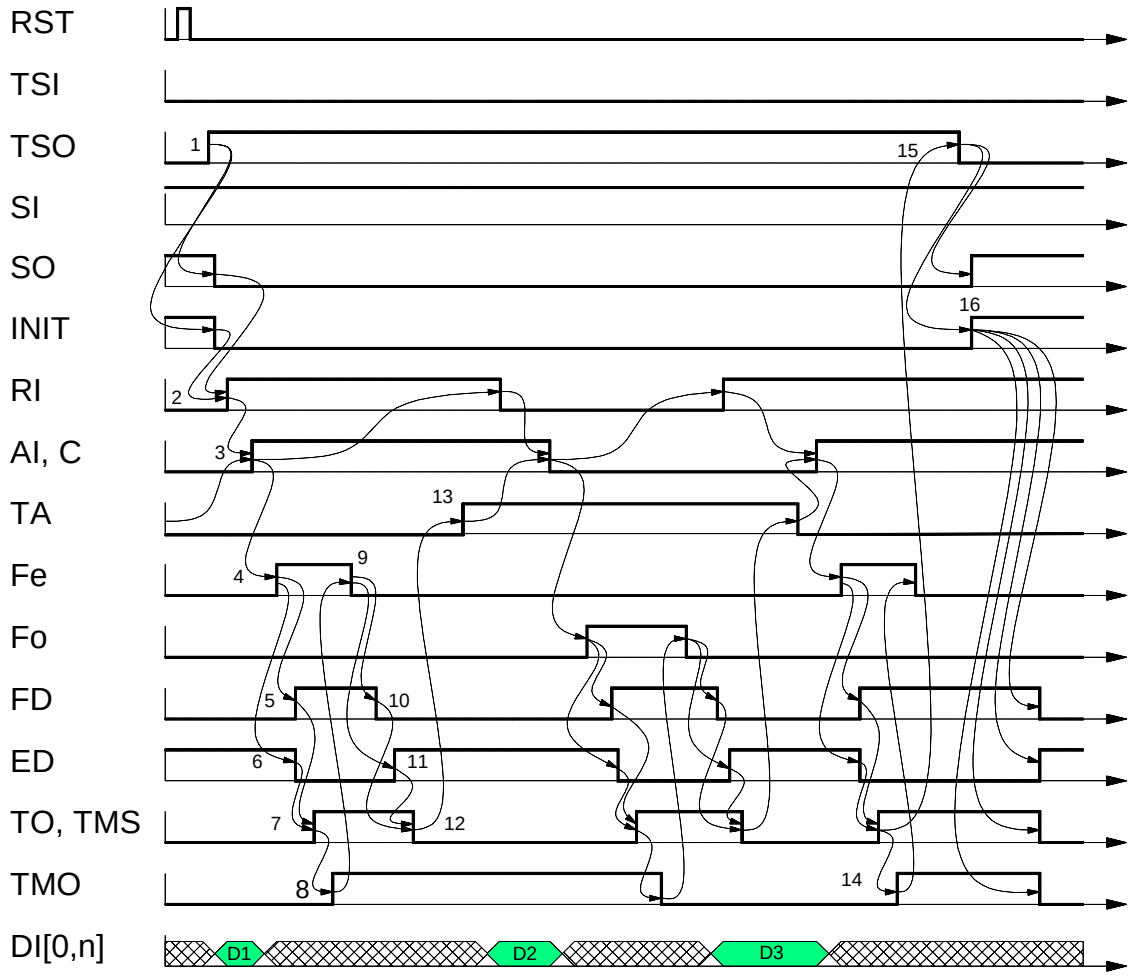

Abb. 62: Das Signalablaufdiagramm des PISO Modus´.

Nach dem Start des PISO Modus´ (**1**) wartet das Register auf das nächste RI Signal (**2**), um das Datum (**D1**) im Register zu speichern und das Quittungssignal, AI+, zurückzusenden (**3**)[63]. Danach schaltet die globale Teststeuerung das Register in den internen Schieberegisterbetrieb. Hierfür werden die lokalen Steuerelemente der seriellen *Micropipeline* mit dem Signal Fe+ gesetzt[64]. Das serielle Schieberegister wird für den Ausleseprozeß mit einem neuen Datum belegt (**4**). Die lokale Steuerung der Register quittiert dies mit den Signalwechseln FD+ (**5**) und ED- (**6**)[65]. Danach können die Signale TO und TMS (**7**), die das

[63] Die Daten sind entsprechend der Steuersignale in den Signalablaufdiagrammen eingezeichnet.

[64] Die Signale Fe und Fo setzen den seriellen Scanpfad in einen speichernden Zustand.

[65] FD+ bedeutet, dass das Register keine transparente Zelle besitzt. ED- zeigt an, dass mindestens eine Zelle ein gültiges Datum beinhaltet. Leert sich ein Scanregister, wechselt nach dem Auslesen der ersten Binärstelle das Signal FD-. Erst

Testdatum in einen den Schieberegisterbetrieb übertragen, ausgelöst werden. Mit TMO+ (**8**) ist dieser Vorgang abgeschlossen, und der serielle Schieberegisterbetrieb kann beginnen. Dafür muß das Signal Fe- (**9**) den Initialisierungsvorgang abschließen[66]. Die Testdaten werden jetzt durch die lokale Testablaufsteuerung ausgelesen. Das Signal FD schaltet während des Auslesevorgangs wieder zurück in den Ausgangszustand (**10**). Ist die Pipeline leer, erfolgt das ED+ Signal (**11**). Der serielle Schieberegisterbetrieb ist damit abgeschlossen. TMS wird zum Einlesen der nächsten Testantwort zurückgesetzt (**12**). Mit TA+ (**13**) und RI+ ist das TMC-Element bereit für einen weiteren Lesezyklus, um das neue Datum (**D2**) im Schieberegister zu speichern. Man erkennt, dass TMO bei jedem zweiten Lesezugriff invertiert wird. Es ist ein *Two-Phase* Signal in der Testablaufsteuerung. Sonst arbeitet es mit *Four-Phase* Signalen. Es wird zur Umwandlung der Signale für den Datenpfad benötigt. Mit dem Wechsel von TMO können Ausleseprozesse beendet werden (**8, 14**). Danach besteht die Möglichkeit, mit TSO- (**15**) den Testbetrieb zu beenden. SO+ und INIT+ (**16**) schalten danach die Testablaufsteuerung in einen definierten Zustand. Die Signale des seriellen Testdatenausgangs sind hier aus Übersichtsgründen nicht dargestellt.

Abb. 63 zeigt den STG zur Steuerung des Datenpfads für den PISO Betrieb. Man erkennt hierbei die Signalwechsel der *Bundled Data Convention* des Datenpfads, RI und AI. Auf der anderen Seite sind Signalwechsel zur Ablaufsteuerung des seriellen Testpfads mit den Namen TO (TMS), TMO, FD und ED zu erkennen. Die Signalwechsel TA-, TA+, Fo+ und Fe+ verbinden die beiden Signalübergangsgraphen (STGs). Mit TA, RI und AI wird die *Bundled Data Convention* des Datenpfads und somit die Ansteuerung des TMC-Elements ermöglicht[67]. Die Synchronisation von der lokalen zur globalen Steuerung erfolgt über Fo- und Fe-. Mit TSO=1 und TSI=0 erfolgt der Start, dessen Signalwechsel im STG mit ausgefüllten Punkten gekennzeichnet sind.

bei der letzten Binärstelle erfolgt das Signal ED+.

[66] Danach können die Elemente aus ihren gesetzten Zuständen die Daten seriell auslesen.

[67] Die Steuersignale des seriellen Schieberegisters sind ebenfalls nicht dargestellt.

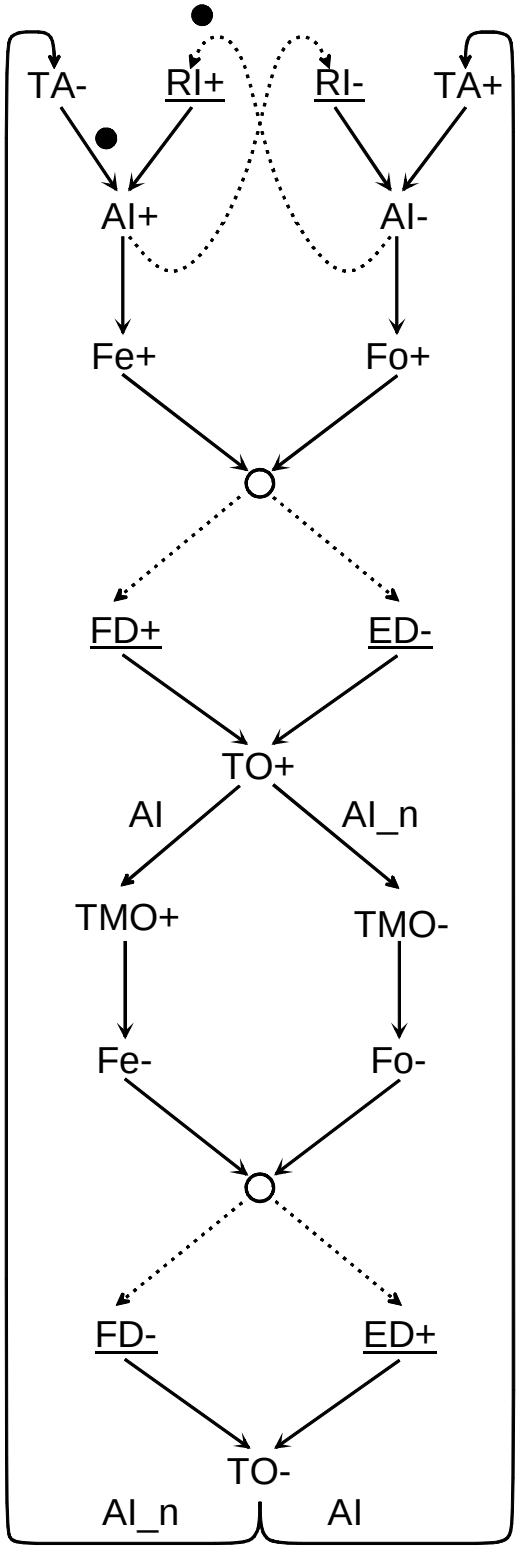

Abb. 63: Der STG des Scanregisters im PISO Modus.

Nach dem Signalwechsel RI+ erfolgt die Speicherung des Datums in eine der Registerzellen des Empfängers durch den Signalwechsel AI+, der zugleich auch mit dem Signal C des Registers verbunden ist. AI quittiert das Speichern des Datums mit dem Signalwechsel für den Sender, der seinerseits das alte Datum löschen, ein neues Datum generieren und einen weiteren Signalwechsel an RI erzeugen kann. Der *Request* für die nächste Pipelinestufe des Datenpfads wird durch die Signalwechsel Fe oder Fo ersetzt, wodurch der serielle Testpfad initialisiert wird.

Alle *Muller C-Elemente* des seriellen Scanpfads werden so gesetzt, dass die parallele Pipelinestufe gefüllt ist[68]. Dabei werden die Signalwechsel FD+ und ED- in den einzelnen Registerzellen berechnet und an die Testablaufsteuerung zurückgegeben. Anschließend schaltet die Steuerung TMS mit Hilfe von TO den Scanpfad in die notwendigen Positionen, damit dort das Testdatum gespeichert und das TMO Signal generiert werden. Dieses löst den Wechsel Fe- aus und schließt die Speicherung in den geraden Registerzellen ab. Bei der Speicherung der Testdaten in den ungeraden Registerzellen würde das Fo Signal gesetzt werden. Das Auslesen der Daten des Scanpfads erfolgt über TDO, TRO und TAO mit Hilfe der *Bundled Data Convention*. Das wird mit der für den Test eingebauten *Muller C-Elemente* des Scanregisters erreicht, die durch das Rücksetzen des Reset und Set Signals den Pipelinebetrieb aufnehmen können. Nach Auslesen der ersten Speicherzelle erfolgt der Signalwechsel FD-, der anzeigt, dass mindestens eine Registerzelle transparent geschaltet ist. Ist auch die letzte Speicherzelle ausgelesen, erfolgt der Signalwechsel ED+, der anzeigt, dass keine Registerstufe ein gültiges Datum enthält. Daraufhin schalten TO- und damit TMS- das Register in den Zustand, um parallele Daten aufzunehmen. Mit einem Signalwechsel von TA wird das dazugehörige Bereitschaftssignal für den Datenpfad generiert.

Man erkennt im STG, dass die Teilgraphen für den Datenpfad und den Scanpfad jeweils ereignis- und pegelgesteuert arbeiten. Während das Interface zum Datenpfad mit den *Two-Phase* Signalen arbeitet, wird die Steuerlogik zum Ansteuern des Scanpfads mit *Four-Phase* Signalen ermöglicht. Um von dem einen Teilgraphen in den anderen Teilgraphen zu gelangen, erfolgt eine Pegelumwand-

[68] Siehe hierzu auch **Kapitel 2.2**.

lung, wie sie in **Kapitel 2.1** dargestellt wurde. Hierzu eignen sich XOR und *Toggle* Elemente. Fe und Fo empfangen hierbei *Two-Phase* Signale aus dem Datenpfad und TA gibt ein *Two-Phase* Signal zurück. Die Pegelumwandlung erzeugt einen erhöhten Steuersignalaufwand, der sich durch zusätzliche Logik bei einer ereignisgesteuerten Signalverarbeitung im Datenpfad niederschlägt.

Das Signalablaufdiagramm der SIPO Betriebsart ist in **Abb. 64** dargestellt. Hier werden statt TO und TMO die internen Zustände TI und TMI eingesetzt. Das Signal TMS kann dabei kontinuierlich auf "1" gehalten werden und muß nicht, wie im PISO Modus, für jedes neue Testmuster schalten. Dies erfolgt nach dem Schaltvorgang von TSI+ (**1**). Nach dem Start durch den positiven Signalwechsel von TSI werden mit Ee- die *Muller C-Elemente* der Schieberegisterzellen für Signalabläufe freigegeben, und das Schieberegister kann mit einem neuen Testmuster gefüllt werden (**2**)[69]. Mit dem Einlesevorgang des ersten Bits schaltet ED- (**3**). Mit dem letzten einzulesendem Bit des Testmusters erfolgt FD+ (**4**)[70]. Danach ist die *Micropipeline* vollständig gefüllt, und der Wechsel von TI+ (**5**) signalisiert das gültige Datum (**D1**) am Ausgang. Danach erfolgt der Wechsel von TR+ (**6**). Es stellt die Umwandlung von TI auf ein *Two-Phase* Signal dar. Dieses übergibt den Steuerwechsel an den Datenpfad, und ein *Request* RO (**7**) erscheint am Steuerausgang des Registers. Danach erfolgt das Schalten des Transparentmodus' für die Schieberegisterzellen des Scanpfads (**9-12**). Abhängig vom Zustand von TMI werden die gerade oder die ungerade Seite der Schieberegisterkette geladen (**13**). Danach wird das Register mit einem neuen Testmuster geladen und mit TI das Quittungssignal an den Datenpfad weitergeleitet (**14**). Mit (**15**) wird der SIPO Betrieb abgeschlossen. Zum Einlesen eines Testmusters in die ungeraden Speicherzellen des Scanpfads erfolgen entsprechende Signalwechsel. Wie auch beim PISO Modus verwendet diese Betriebsart für das Interface des Datenpfads *Two-Phase* Signale und für die interne Ablaufsteuerung *Four-Phase* Signale.

[69] Die Signale Ee und Eo setzen den seriellen Scanpfad in einen transparenten Zustand.

[70] Beide Signalwechsel (ED und FD) sind beim PISO und SIPO Betrieb nötig, um alle Testbetriebsarten von einer Testablaufsteuerung zu kontrollieren. Soll nur eine Betriebsart implementiert werden, reichen ED für den PISO und FD für den SIPO Betrieb aus.

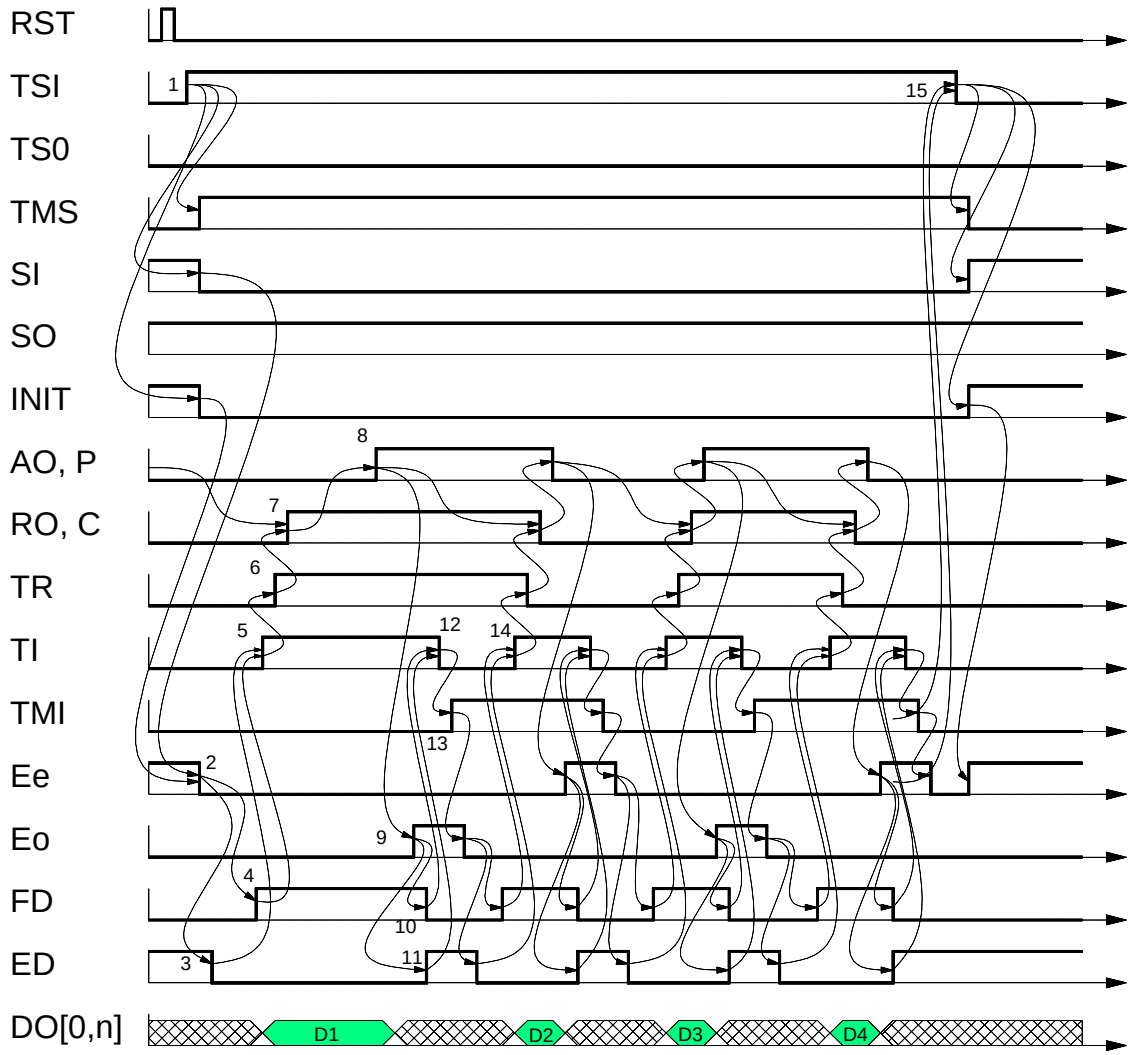

Abb. 64: Das Signalablaufdiagramm des SIPO Modus´.

Abb. 65 zeigt den zugehörigen STG der SIPO Betriebsart. Er ist im Aufbau vergleichbar mit dem STG im PISO Modus. Oben befindet sich die Steuerung der *Bundled Data Convention* des Datenpfads. Der untere Teil des STGs beschreibt den Ablauf des seriellen Einlesens eines neuen Testmusters. Die Synchronisation im SIPO Modus zwischen der lokalen und globalen Steuerung erfolgt über die Initialisierungssignale Eo und Ee. Das Register quittiert mit FD und ED den Abschluß der Initialisierung und den Füllstatus der Zellen.

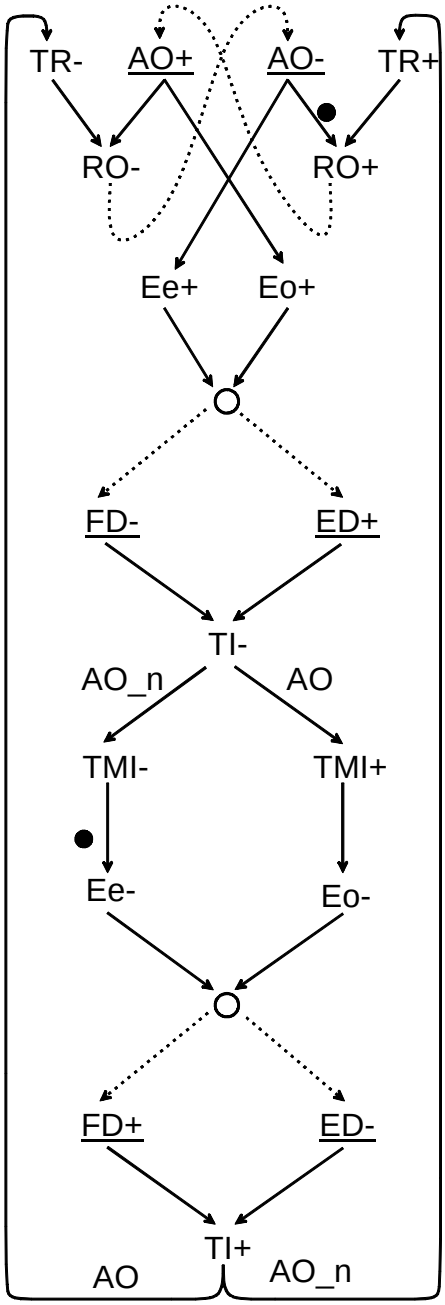

Abb. 65: Der STG des Scanregisters in der SIPO Betriebsart.

Nach der Initialisierung der Pipeline durch das Steuersignal TSI+ wird als erster Signalwechsel Ee- ausgeführt. Der Startpunkt im STG ist mit Punkten gekennzeichnet. Dann erfolgt das serielle Einlesen eines neuen Testmusters in das Scanregister. Die Quittungssignale sind die Signalwechsel FD+ und ED-. TI+ ist das Four-Phase Signal, um das *Two-Phase* Signal TR+ zu erzeugen. Mit dem weiteren Initialisierungspunkt kann das Signal RO+ für die zweite Pipelinestufe

erzeugt werden[71]. Erfolgt AO+ von der zweiten Pipelinestufe, ist das Datum dort gespeichert. Ein weiteres Datum kann über den seriellen Eingang der ersten Stufe eingelesen werden. AO ist dabei identisch mit dem P Signal des Registers. Das Einlesen des neuen Datums erfolgt durch Eo+, welches die Initialisierung der lokalen Steuerung der seriellen Registerstufen ausführt. Das Register quittiert diesen Vorgang mit den Signalen FD- und ED+. Daraufhin wird über die Signale TI- und TMI+ mit Eo- die Initialisierung beendet. Das Füllen der Micropipeline über TDI, TRI und TAI wird mit FD+ abgeschlossen. Wird das erste neue Bit eingelesen, erfolgt schon der Signalwechsel ED-, der anzeigt, dass das Register nicht mehr vollständig leer ist. Mit FD+ ist der Speichervorgang erfolgt.

In **Abb. 66** ist der serielle Schieberegisterbetrieb als Signalablaufdiagramm für die SISO Betriebsart dargestellt. Hier werden die *Muller C-Elemente* des seriellen Scanpfads auf "0" initialisiert (*even mode*). TRI und TAI sind die Steuersignale des Testdateneingangs TDI. TRO und TAO sind die Steuersignale des Testdatenausgangs TDO. Mit Hilfe der *Bundled Data Convention* werden die Testdaten seriell in die elastische Pipeline ein- und ausgelesen[72]. Steuersignale für den Datenpfad werden nicht generiert. Diese Betriebsart dient zum Weiterleiten von Testmustern an eine folgende Stufe. Zum Einlesen werden die Signalwechsel (**3-7**) benötigt. (**8, 11, 14**) zeigen den Füllgrad des Schieberegisters an. Mit (**9, 10, 12, 13**) werden die zuvor eingelesenen Daten ausgelesen.

[71] Hierfür ist das Signal AO- der Initialisierung gesetzt.

[72] Der entgegengesetzte Fall, bei dem alle *Muller C-Elemente* auf "1" initialisiert sind (*odd mode*), ist hier nicht dargestellt, kann aber analog entwickelt werden. Siehe hierzu auch **Kapitel 2.2.2**. Auf die Darstellung des STGs dieser Betriebsart wird hier verzichtet, da das Register als konventionelle *Micropipeline* arbeitet. Die Zeitpunkte zur Umschaltung in eine andere Betriebsart erfolgen über die Signale FD (**8**) und ED (**14**), die den Füllgrad der Stufen anzeigen.

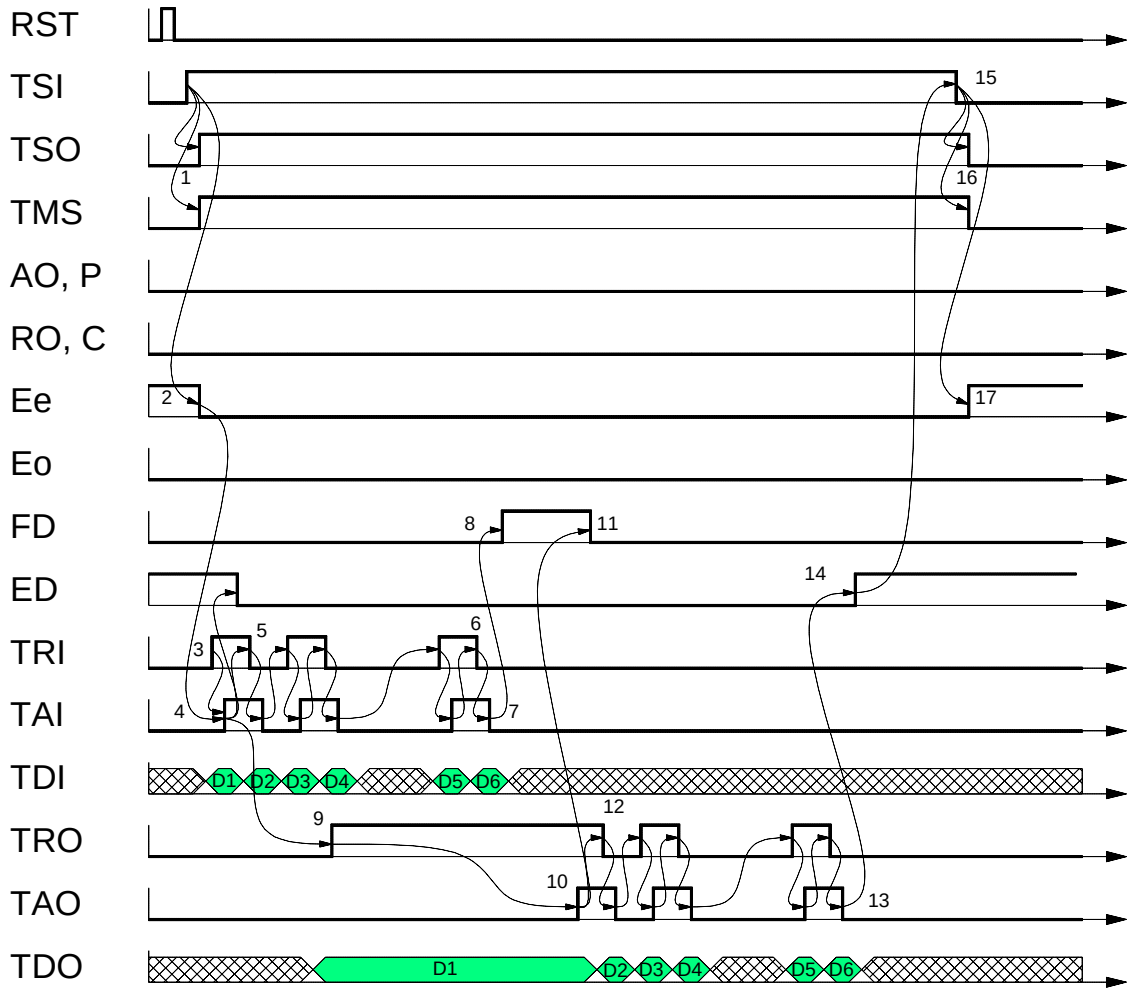

Abb. 66: Der serielle Schieberegisterbetrieb.

Es folgt nun eine nähere Untersuchung zur Initialisierung der *Muller C-Elemente* des seriellen Schieberegisterpfads durch die globale Testablaufsteuerung. Um die Steuersignale zur Initialisierung der *Muller C-Elemente* in den seriellen Registerzellen zu generieren, müssen die Signale Ee (*Empty even*) und Eo (*Empty odd*) das Leeren sowie Fe (*Full even*) und Fo (*Full odd*) das Füllen der Pipeline einleiten[73]. Mit dem definierten Setzen der *Muller C-Elemente* werden die Registerzellen so geschaltet, dass das gültige Datum im Scanpfad gespeichert wird, bzw. das Scanregister bereit für neue Daten ist. Zusätzlich kann unterschieden werden, welche der beiden Zellen des Registers geladen werden sollen. Die Steuersignale AO und RO des Datenpfads leiten eine neue Initialisierung des Scanpfads ein. Die Signale TMI und TMO der globalen Steuerung

[73] Die Signalnamen sollen anzeigen, ob der Steuerpfad eines Scanregisters für den Transparent- oder Speichermodus initialisiert wird.

sowie AO und RO des parallelen Datenpfads werden ausgewertet, um das Ende der Initialisierung zu erkennen oder um eine neue Initialisierung zu beginnen. Mit diesen Signalen werden Re_n, Ro_n, Se_n und So_n gesetzt, die in **Abb. 55** als Eingangssignale für HRP zu sehen sind. Sie setzen die *Muller C-Elemente* des Scanpfads und damit die Registerzellen in die gewünschte Schalterstellung. Die Signale Re_n (*Reset even not*) und Ro_n (*Reset odd not*) sind mit den Reset Eingängen der geraden (*even*) und der ungeraden (*odd*) *Muller C-Elemente* verbunden. Die Signale Se_n (*Set even not*) und So_n (*Set odd not*) sind mit den Set Eingängen der geraden (*even*) und der ungeraden (*odd*) *Muller C-Elemente* verbunden[74].

In **Abb. 67** ist das Signalübergangsdiagramm für die Initialisierung der Registerzellen des Scanpfads dargestellt[75]. Es ist die gleichzeitige Aktivierung der Steuersignale des Registers der Übergänge (**6**) durch (**5**), (**12**) durch (**11**) und (**16**) durch (**15**) und (**18**) durch (**17**) zu erkennen. Ist das Register initialisiert, werden die Signale Ee, Eo, Fe und Fo wieder in den Ruhezustand gesetzt. Der Übergang (**3**) erfolgt durch (**2**) sowie (**9**) durch (**8**). Im Anschluß daran erfolgen die Signalwechsel TMI oder TMO (**7, 19**). Nachdem der Initialisierungsvorgang abgeschlossen ist (**3, 9**) werden die Signale Re_n, Ro_n, Se_n und So_n auf "1" in den Ruhezustand gesetzt. Der parallele Datenpfad quittiert die Ausführung seines Signalwechsel mit (**4, 10**) von der nachfolgenden Stufe und (**14, 20**) von der vorhergehenden Stufe. Da der Datenpfad mit *Two-Phase* Signalen arbeitet und die Registerzellen pegelgesteuerte Signale benötigen, werden TMI und TMO zur Umwandlung der Pegel von *Two-Phase* auf *Four-Phase* verwendet.

[74] Siehe hierzu auch **Kapitel 2.2.2**, welches näher auf die Initialisierung von *Micropipelines* eingeht.

[75] Die Verbindungen des Signals H_n zu den restlichen Signalen sind hier aus Übersichtsgründen nicht dargestellt. Es wird immer dann geschaltet, wenn eines der vier Signale Re_n, Ro_n, Se_n und So_n aktiv ist. Hier zeigt sich ein Nachteil der graphischen Darstellung von STGs, wenn sie eine hohe Signalaktivität aufweisen.

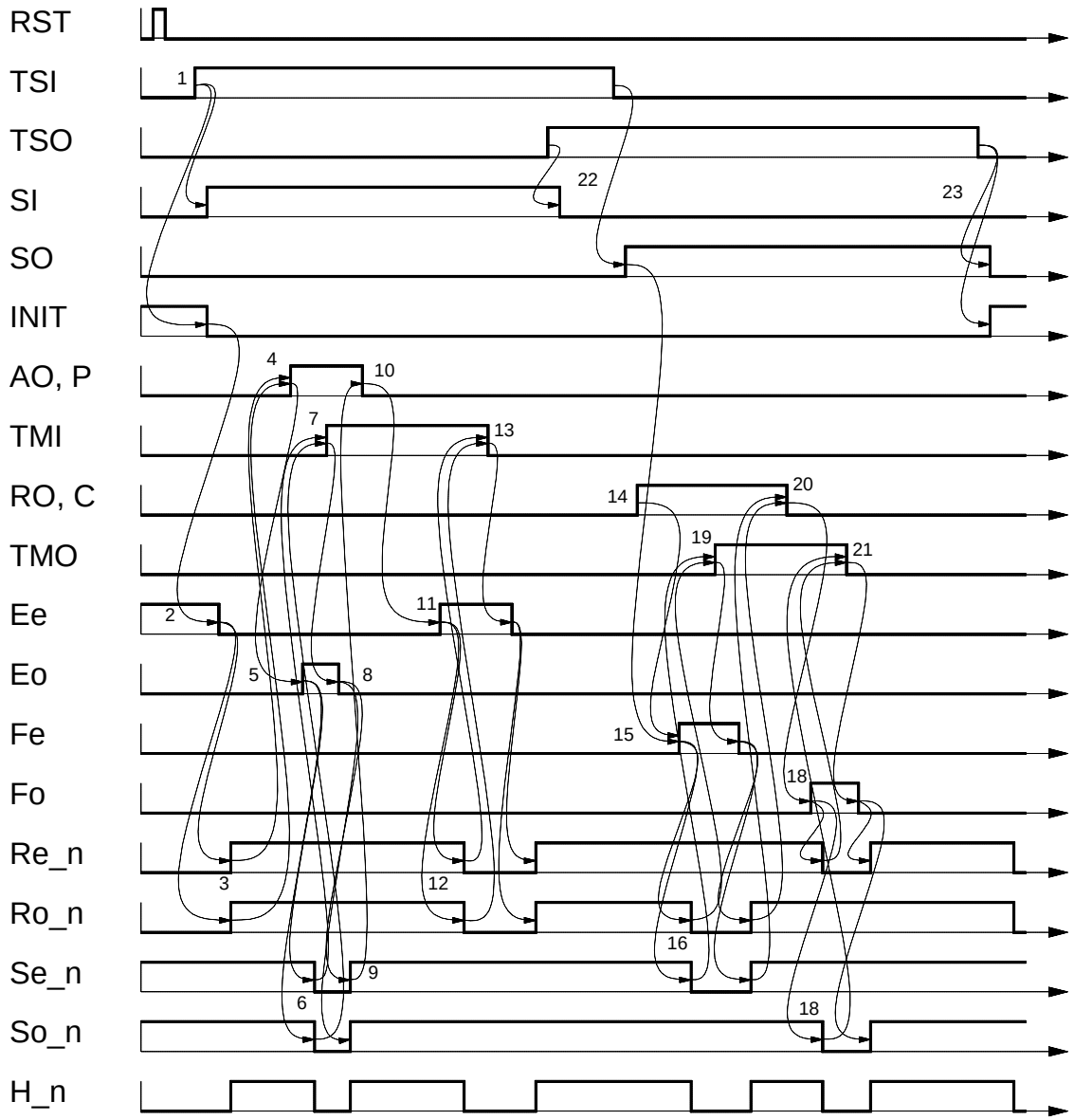

Abb. 67: Das Signalablaufdiagramm für die Initialisierungssignale der Testablaufsteuerung.

Abb. 68 zeigt den zugehörigen STG zur Erzeugung der Initialzustände der lokalen Steuerung. Ausgehend von den Anfangszuständen und der Einstellung der Betriebsart des Registers werden Zwischenzustände generiert, die mit Ee, Eo, Fe und Fo bezeichnet sind. Für den Leerzustand gibt es zwei Möglichkeiten. Alle *Muller C-Elemente* werden auf "0" oder alle auf "1" gesetzt. Dies ist abhängig davon, ob das neue gültige Datum in der oberen oder unteren Registerzelle gespeichert sein soll. Analog dazu gibt es auch zwei Möglichkeiten zum Initialisieren des Vollzustands mit einer Reihe von gegensätzlich initialisierten *Muller C-Elementen*. Der Initialisierungswert der *Muller C-Elemente* beginnt

entweder mit "0" oder "1". Ist die Initialisierung der Testablaufsteuerung erfolgt, wechselt der mit den Punkten gekennzeichnete Signalpegel. Dieser ist für den PISO und SIPO Modus unterschiedlich.

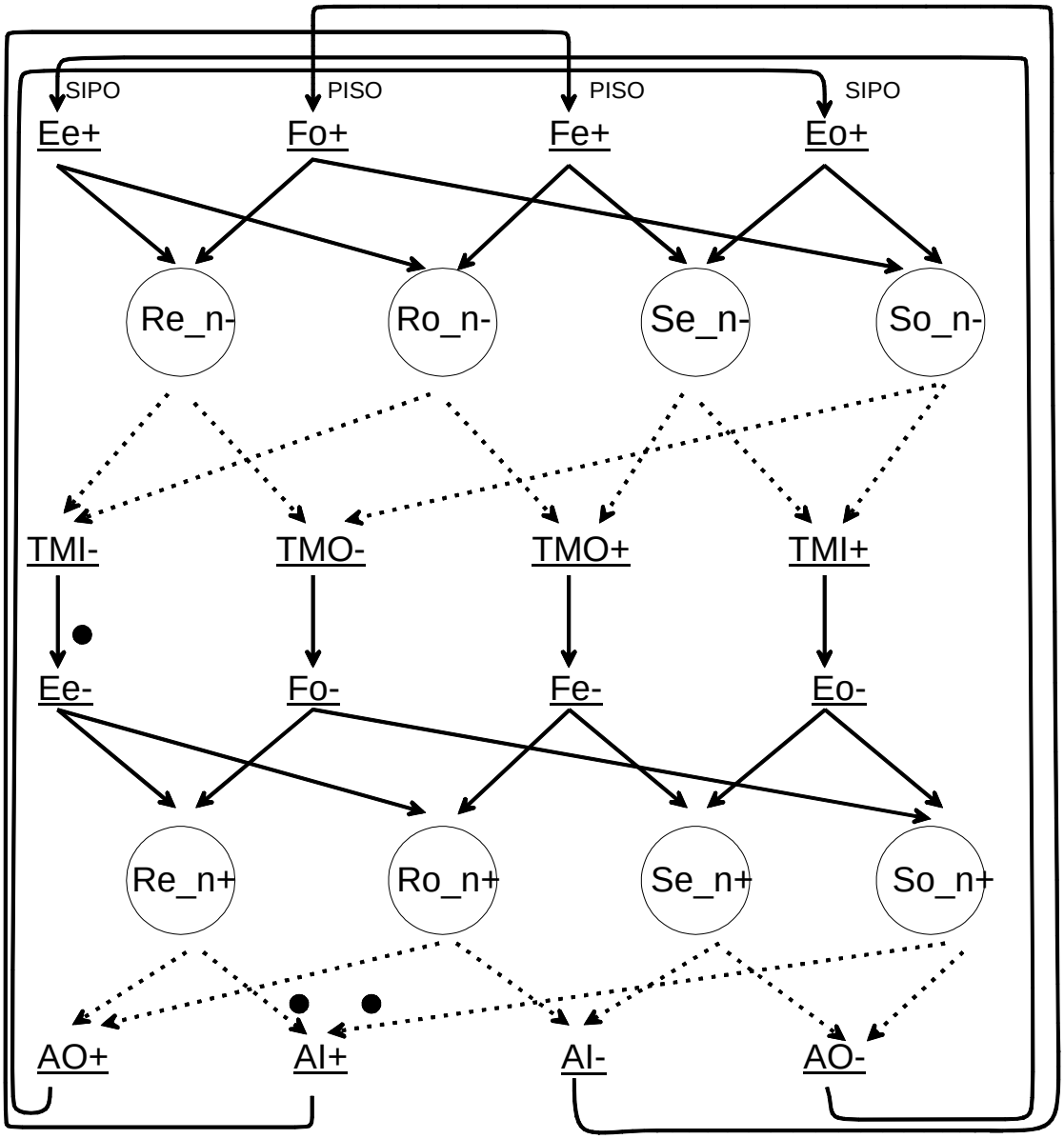

Abb. 68: Der STG zur Erzeugung des Initialzustands des Scanpfads.

Der erste Signalwechsel für den SIPO Modus erfolgt mit Ee-, der für den PISO Modus mit AI+. Im SIPO Modus löst die Initialisierung die Signalwechsel Re_n+, Ro_n+ aus. Danach kann das neue Testmuster eingelesen und über den Datenpfad in die nachfolgende Registerstufe weitergeleitet werden. Diese Stufe quittiert das Speichern mit dem AO+ Signal. Danach erfolgt die Initialisierung des Datenpfads für das nächste Testmuster mit Eo+. Das erzeugt das Rücksetzen

von Se_n- und So_n-. Die Registerstufe quittiert dies mit dem Signalwechsel von TMI. Daraufhin können die Setzsignale zurückgenommen (Se_n+ und So_n+), und das nächste Testmuster kann eingelesen werden. Die PISO Betriebsart erfolgt analog. Wurde ein gültiges Datum im Register gespeichert, schaltet das Signal AI. Aus dem seriellen Scanpfad kann ein aktuelles Datum über TDO ausgelesen und mit TMO quittiert werden. Danach folgt die Speicherung eines neuen Testmusters über den parallelen Dateneingang. Im seriellen Pfad entsteht durch die unterschiedliche Initialisierung der Register das Problem, dass die Signale TAI und TRI sowie TAO und TRO invertiert werden müssen.

Im folgenden wird ein Vorschlag für alle Betriebsarten der Testablaufsteuerung des passiven HIOB Konzepts vorgestellt. Zusätzlich zu dem TMC-Element[76] des Datenpfads wird eine Testablaufsteuerung für den Scanpfad benötigt. Die Logik zur Testablaufsteuerung wird mit HIOB Control passiv (HCP) bezeichnet und eine mögliche Realisierung in **Abb. 69** vorgestellt. Sie dient zur Konvertierung von *Two-Phase* auf *Four-Phase* Signale und zurück auf *Two-Phase* Signale. Mit den *Four-Phase* Signalen erfolgt die Anbindung der lokalen Steuerung des seriellen Scanpfads an den Datenpfad. In der Abbildung sind die Anschlüsse zur Steuerung des Datenpfads (RI, AI, Ro_n, AO, R), zur Steuerung und Beobachtung des asynchronen Scanregisters (TMS, C, P, F, ED, FD) sowie zur Steuerung und Initialisierung des HIOB Control Registers (TSO, TSI, Re_n, Ro_n, Se_n, So_n) zu sehen. HCP läßt sich in mehrere Bereiche aufteilen. Das TMC-Element wird für die *Bundled Data Convention* und die Umschaltung in die Testbetriebsarten PISO, SIPO und SISO verwendet. Direkt darunter befinden sich Module mit den Signalen TI, TMI, Ee, Eo, Re_n und Ro_n, die die Ablaufsteuerung der SIPO Betriebsart ermöglichen. Für den PISO Modus werden die Module mit den Steuersignalen TO, TMO, TA, Fe, Fo, Se_n und So_n verwendet, die sich unterhalb der SIPO Steuerung befinden. Die Setz- und Rücksetzeingänge des Scanregisters werden abhängig von den Steuersignalen Ee, Eo, Fe und Fo aktiviert.

Zur Initialisierung der globalen Steuerung werden mehrere Elemente und Signale benötigt. Diese werden mit den Signalen TSI und TSO erzeugt. Die Gatter auf der linken Seite generieren aus zwei Eingangssignalen die Signale INIT, SI und SO. Rechts befinden sich Logikgatter zur Initialisierung der seriellen Registers-

[76] Hierzu wird das *Muller C-Element* aus **Abb. 51** verwendet.

teuerung. Das TMS Signal für das Register wird dann erzeugt, wenn eine der Testbetriebsarten aktiv ist. H_n wird dann aktiviert, wenn sich das Register in einer der Initialisierungsphasen befindet. C und P werden abhängig von den Signalen Ro_n und AO geschaltet. Die Schaltgeschwindigkeit dieser Signale wird durch die Testlogik nicht beeinträchtigt. Auch die anderen Steuersignale werden nur geringfügig durch das Timing des TMC-Elements beeinflußt.

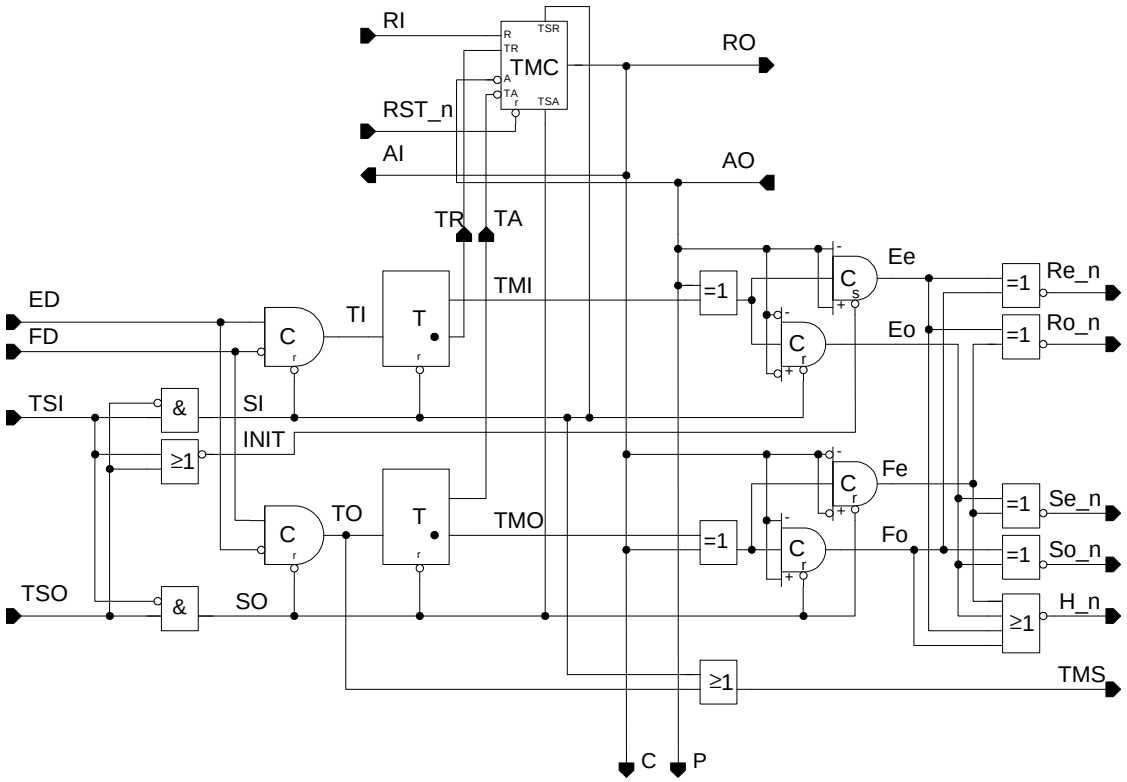

Abb. 69: Die Schaltung zur Steuerung des Datenpfads.

5.2 Aktive testfreundliche Entwurfshilfen

Im folgenden wird ein Testkonzept vorgestellt, das neben der seriellen Schiebeoperation zusätzlich eine Testmustergenerierung und Testdatenkompression innerhalb der Testlogik erlaubt. Als Basis für die Funktionsweise des Bausteins werden die Betriebsarten des BILBOs verwendet **[79KönnMZ]**$_{L.2}$.

Ziel bei diesem, wie auch bei dem bereits vorgestellten Verfahren ist es, den Datenpfad während des Tests weiterhin mit der *Bundled Data Convention* zu synchronisieren. Der Unterschied zwischen den Konzepten liegt darin, dass hier keine lokale Steuerung im seriellen Schiebeoperationsbetrieb enthalten ist. Daraus ergibt sich, dass die serielle Schieberegisteroperation nicht als elastische *Micropipeline* konzipiert ist. Dies ermöglicht eine schnellere Testmustergenerierung und Kompression, da die Signalverzögerung zwischen den Flip-Flops bekannt ist. Auch der Schiebeoperationsbetrieb kann bei langen Registerketten deutlich beschleunigt werden. **Abb. 70** zeigt das Konzept des aktiven testfreundlichen Entwurfs. Die Testablaufsteuerung (HCA) im Datenpfad schaltet zwischen dem Operations- und Testbetrieb und generiert die Signale zur Ansteuerung der Speicherzellen, die sich im Register HRA befinden. Mit Hilfe eines Multiplexers und der linearen Rückkopplung von Speicherelementen können im Testbetrieb die unterschiedlichen Betriebsarten eingestellt werden. Die Auswahl der rückgekoppelten Signale entscheidet darüber, ob eine maximale Zyklendauer erreicht wird, wie dies auch bei der BILBO Technik erfolgt **[90AbraBF]**$_{L.1}$. Mit einem *Request* Signal von TRI am Testeingang werden die Inhalte der Speicherzellen synchron um eine Stelle verschoben. Dies ist möglich und sinnvoll, da bei allen Registerzellen die Datenverarbeitungsgeschwindigkeit gleich ist. Hierdurch ist die Steuerung des Testpfads verschieden zu dem ersten, passiven Testkonzept aufgebaut. Das Prinzip der *Micropipeline* im seriellen Pfad ist dadurch nicht mehr möglich. Die *Two-Phase Bundled Data Convention* des Datenpfads bleibt davon unbeeinflußt.

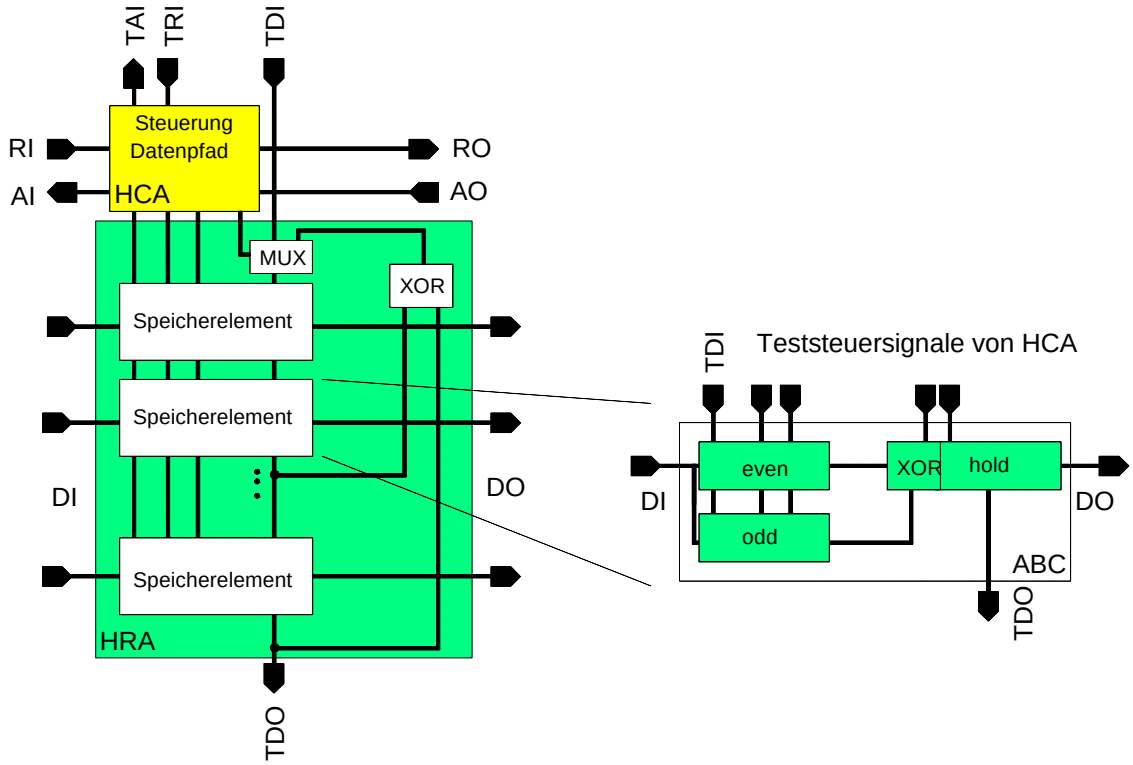

Abb. 70: Die HIOB-Struktur der aktiven Testhilfen enthält keine lokale Steuerung.

Dieser neue Ansatz reduziert durch den Wegfall der lokalen Steuerelemente die Anzahl der Transistoren im Datenregister auf Kosten einer erhöhten Anzahl von Signalleitungen zwischen den Speicherelementen und einer Testablaufsteuerung. TRI und TAI sind *Four-Phase* Signale. TRO und TAO werden hier nicht benötigt, da die Schieberegisteroperation nicht mehr nach der *Bundled Data Convention* erfolgt[77]. Die Signale im Datenpfad werden hingegen weiterhin mit der *Two-Phase Bundled Data Convention* gesteuert. Die Umwandlung der Signale von *Four-Phase* in *Two-Phase* erfolgt durch die Testablaufsteuerung.

Zur schnellen Generierung und Kompaktierung von Testdaten ist es notwendig, zu den zwei scanfähigen transparenten Speicherzellen ein drittes Speicherelement einzufügen. Dies ergibt sich aus folgender Betrachtung: Kompaktiert man eine Testantwort, müssen die zuvor ermittelte Signatur und die neue Testantwort mit einem EXOR verknüpft und als neue Signatur gespeichert werden[78].

[77] Dies verdeutlicht auch die unterschiedliche Ansteuerung beider Verfahren.

[78] Rechts in **Abb. 70** erkennt man die Struktur des benötigten Speicherelements.

Wenn in der unteren Speicherzelle (*odd*) die aktuelle Signatur gespeichert ist, wird für das nächste Testmuster die obere Speicherzelle (*even*) transparent geschaltet. Mit Speicherung der Testantwort erfolgt das *Acknowledge* Signal für den Sender des Testmusters. Die Signatur wird dabei in der anderen Speicherzelle gehalten, bis die neue Signatur gebildet ist. Im Scanbetrieb wird diese Speicherzelle (*hold*) transparent geschaltet, und die XOR-Verknüpfung der Eingänge wird durch einen Schalter überbrückt. Im nächsten Schritt muß die neue Signatur aus den Inhalten der beiden Speicherzellen ermittelt und in einem Zwischenspeicher (*hold*) abgelegt werden. Im nächsten Schritt werden die neue Signatur mit einer Schiebeoperation in der oberen Speicherzelle (*even*) der nächsten Binärstelle des HIOB-Registers als aktuelle Signatur gespeichert und die untere Speicherzelle (*odd*) transparent geschaltet. Somit ist sie bereit zur Speicherung eines neuen Testmusters. Die Speicherung der Testmuster, abwechselnd in der oberen und unteren Zelle, ergibt sich aus dem Prinzip der *Micropipeline*[79].

Mit diesem Prinzip können Testdaten ohne eine zusätzliche Pipelinestufe im Testempfänger gespeichert und ein *Acknowledge* Signal an die vorhergehende Stufe gesendet werden. Die Berechnung der neuen Signatur erfolgt in einem internen Testmodus und ist von der *Bundled Data* Synchronisation unabhängig. Der Testsender kann sofort nach Senden des *Request* Signals das *Acknowledge* Signal vom Empfänger erhalten, ein neues Testmuster berechnen, in den Datenpfad schalten und ein weiteres *Request* Signal erzeugen. Die Erzeugung des nächsten Testmusters erfolgt unabhängig von der Berechnung der Signatur in der Empfängerstufe. Die Betriebsart zur Generierung von Pseudozufallsmustern benötigt zur Berechnung des neuen Testmusters nur das aktuelle Testmuster sowie eine XOR-Rückkopplung von bestimmten Speicherzellen der Schieberegisterkette, mit deren Hilfe der serielle TDI Eingang des HIOB-Registers mit neuen Daten geladen wird. Bei der Schieberegisteroperation zur Speicherung des nächsten Testmusters wird dieses berechnete Datum in das Register eingelesen.

Da beide Betriebsarten mit einer vergleichbaren Geschwindigkeit die nächsten Testmuster berechnen und nur einen Signalwechsel für eine Schiebeoperation

[79] Siehe hierzu auch die Betrachtungen zur Speicherung und Fortschaltung von Daten in *Micropipelines* aus **Kapitel 2.2.2**.

für alle Speicherzellen benötigen, ist mit diesem Verfahren eine hohe Verarbeitungsgeschwindigkeit im Testbetrieb möglich. Insbesondere kritisches Zeitverhalten bei größter Verarbeitungsgeschwindigkeit der Schaltung kann somit überprüft werden, da die Testmuster innerhalb der Schaltung erzeugt werden. Sie werden nicht von einem Testautomaten von außen mit großen Verzögerungen in den Schaltungskern ein- und ausgelesen. Große Vorteile erreicht man mit diesem Konzept, wenn, wie in diesem Fall, Testmuster generiert und kompaktiert werden. Ein Nachteil ist, dass durch die fehlende lokale Steuerung im seriellen Schieberegisterbetrieb vergleichbare SIPO und PISO Scan-Betriebsarten des passiven HIOB Entwurfs nicht mehr möglich sind. Es muß also von außen zwischen der seriellen Schiebeoperation und dem parallelen Datenbetrieb umgeschaltet werden. Die Testmustergenerierung und die Testdatenkompression ersetzen diese Modi.

Es ist möglich, auf das neue, dritte Speicherelement in der Registerstruktur zu verzichten. Dafür muß das *Acknowledge* Signal des Empfängers im Testbetrieb verzögert werden, bis die neue Signatur berechnet und als Testantwort gespeichert ist. Der Testsender dient dabei als Zwischenspeicher und darf solange seine Ausgangssignale nicht verändern. Dies bedingt im Steuerpfad eine zusätzliche Logik zum Halten des *Acknowledge* Signals, die auch im Operationsbetrieb die Geschwindigkeit der Schaltung reduziert. Hinzu kommt eine zusätzliche Verzögerung bei der Generierung und Kompaktierung der Testmuster. Die Prozesse zur Generierung und Kompaktierung des nächsten Testmusters laufen nicht mehr parallel, sondern sequentiell ab. Das XOR wird dabei mit der gespeicherten Signatur und der neuen Testantwort im zeitkritischen Datenpfad verknüpft. Diese Realisierung wurde daher aus Performancegründen nicht weiter verfolgt.

Die Struktur des Datenpfads wird anschließend beschrieben. Danach erfolgt die Beschreibung der Steuerlogik. Anhand von STGs werden die Signalwechsel der unterschiedlichen Betriebsarten ermittelt und eine Schaltungsrealisierung vorgestellt. Es ist mit Hilfe der STGs möglich, weitere Realisierungen zu entwerfen.

5.2.1 Der Datenpfad des HIOB Moduls für alle Betriebsarten

Der Datenpfad ist in Bitslice Technik aufgebaut. Eine EXOR-Rückkopplung vom TDO des letzten Bits mit einem Ausgang oder mehreren Ausgängen von Speicherzellen auf den Eingang ermöglicht die Erzeugung von pseudozufälligen Testmustern oder dient der Testdatenkompression. **Abb. 71** zeigt diese Struktur, in der der Datenpfad parallel an DI und DO sowie der Testbus seriell durch alle Registerzellen verbunden sind. Durch das Fehlen der lokalen Steuerung in den Registerstufen müssen neben den Initialisierungssignalen, *Capture* und *Pass*, weitere Steuersignale von der globalen Steuerung des Datenpfads im Testbetrieb erzeugt werden. Diese Steuersignale werden parallel an alle Speicherelemente (ABC) angeschlossen. Mit einem Multiplexer an TDI von HRA wird zwischen dem seriellen Scanbetrieb und den Betriebsarten der Testmustergenerierung und der Testdatenkompression gewechselt.

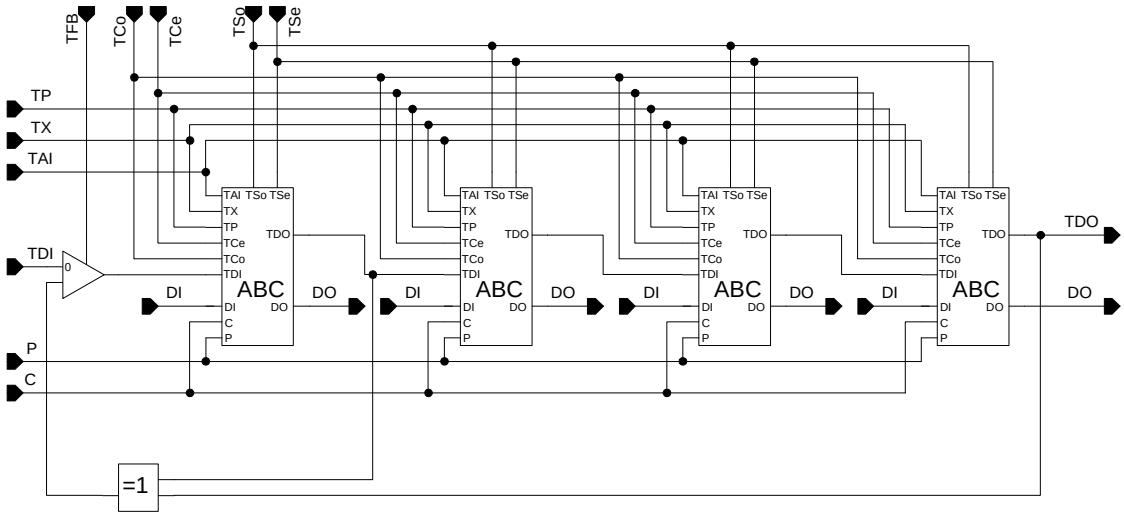

Abb. 71: Die Struktur des HIOB Registers HRA für das aktive Testkonzept mit seinen Speicherelementen.

Jedes ABC Modul enthält dabei eine scanfähige *Capture-Pass* Speicherzelle, die aus zwei SC-Elementen sowie einem Schalter aufgebaut ist, um einen der beiden Ausgänge DO der SC-Elemente auf den Ausgang DO zu schalten[80]. SC_11 wird für die geraden (*even*) und SC_10 für die ungeraden Speicherzellen verwendet (*odd*). Zusätzlich zu den zwei Speicherelementen ist ein weiteres

[80] Siehe hierzu **Abb. 72**.

Speicherelement, XL, integriert[81]. Es ermöglicht zu der Schalterfunktion[82] eine XOR-Verknüpfung der Eingänge für den MISR Betrieb sowie die Speicherung des Zwischenzustands bei der seriellen Schiebeoperation.

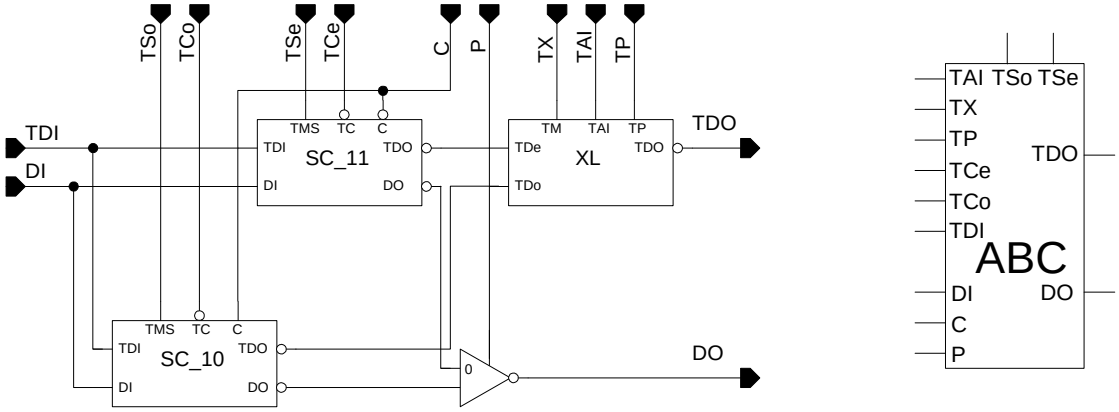

Abb. 72: Die asynchrone Speicherzelle für den MISR Betrieb.

Abb. 73 zeigt das Prinzip des XL Elements. Die rechte Anordnung ist im MISR Betrieb aktiv (TX=1). Die Signatur wird mit dem aktuellen Testmuster XNOR verknüpft und im Latch als neue Signatur gespeichert. Durch das invertierte Speicherelement, welches durch das Signal TRI gesteuert wird, ergibt sich die gewünschte logische XOR-Verknüpfung der beiden Speicherzellen (*even* und *odd*) der linearen Rückkopplung. Im LFSR Modus und Schieberegisterbetrieb (TX=0) ist das XOR nicht erforderlich. Es wird durch einen Schalter ersetzt, um eines der beiden SC-Elemente auf TDO zu schalten und dort zu speichern[83].

[81] Es ist mit *hold* in dem ABC Register in **Abb. 70** bezeichnet.

[82] Siehe hierzu den Aufbau des *Capture-Pass* Registers nach Sutherland und das testfreundliche *Capture-Pass* Register der passiven Teststruktur (ASR).

[83] Eine mögliche Realisierungsform für eine Standardzelle wurde in **Kapitel 4** vorgeschlagen.

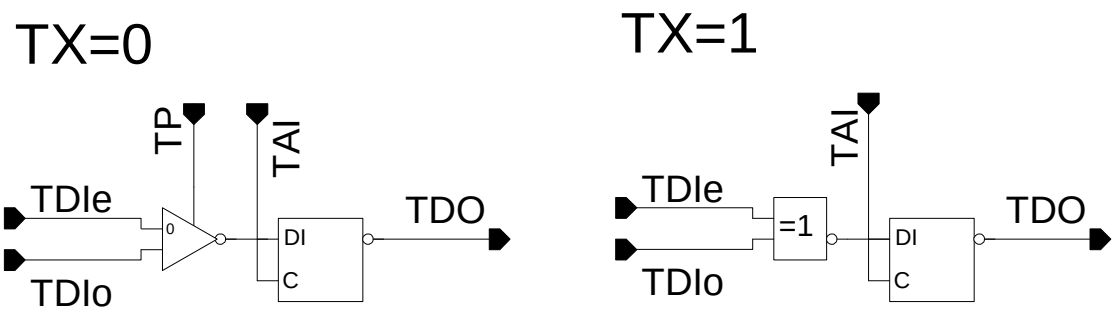

Abb. 73: Das MISR-Register zwischen den Pipelinestufen muß nach Betriebsart die Eingänge XOR verknüpfen.

5.2.2 Die Ereignissteuerung für alle Betriebsarten

Die Ereignissteuerung des HIOB-Konzepts für aktive Testhilfen ist in einer globalen Steuerung für den Datenpfad zusammengefaßt. Sie enthält das TMC-Element zum Umschalten vom Operations- in den Testbetrieb sowie ein weiteres Modul zur Generierung der Steuersignale für die ABC-Elemente des HIOB Registers. Dieses zusätzliche Modul, HCA, generiert auch die Steuersignale für die Synchronisation des Datenpfads, HRA. In **Abb. 74** sind die Signale zur Steuerung des Registers dargestellt.

Auch dieses Testkonzept sieht keine Änderungen im Steuer- und Datenpfad zwischen den Registerstufen vor. Das *Muller C-Element* der Registerstufe wird durch das TMC-Element ersetzt. Drei externe Teststeuersignale, TSM, TPR und TSH, kontrollieren die Testmodi und schalten zwischen unterschiedlichen Betriebsarten. Mit TRI und TAI werden neue Testdaten über TDI eingelesen und über TDO ausgelesen. Im TPG und TDA Betrieb dienen TRI und TAI zur lokalen Synchronisation[84]. Sie ersetzen dabei die Steuersignale des Datenbetriebs, RI und AI. TRI und TAI sind asynchrone, pegelgesteuerte Signale. Mit TAI kann extern die Anzahl der erfolgten Schieberegisteroperationen gezählt und dann gegebenenfalls eine neue Betriebsart durch die externen Teststeuersignale eingeleitet werden.

[84] Mit einem Inverter zwischen TRI und TAI kann die maximale Geschwindigkeit des Bausteins in diesem Betrieb ermittelt werden.

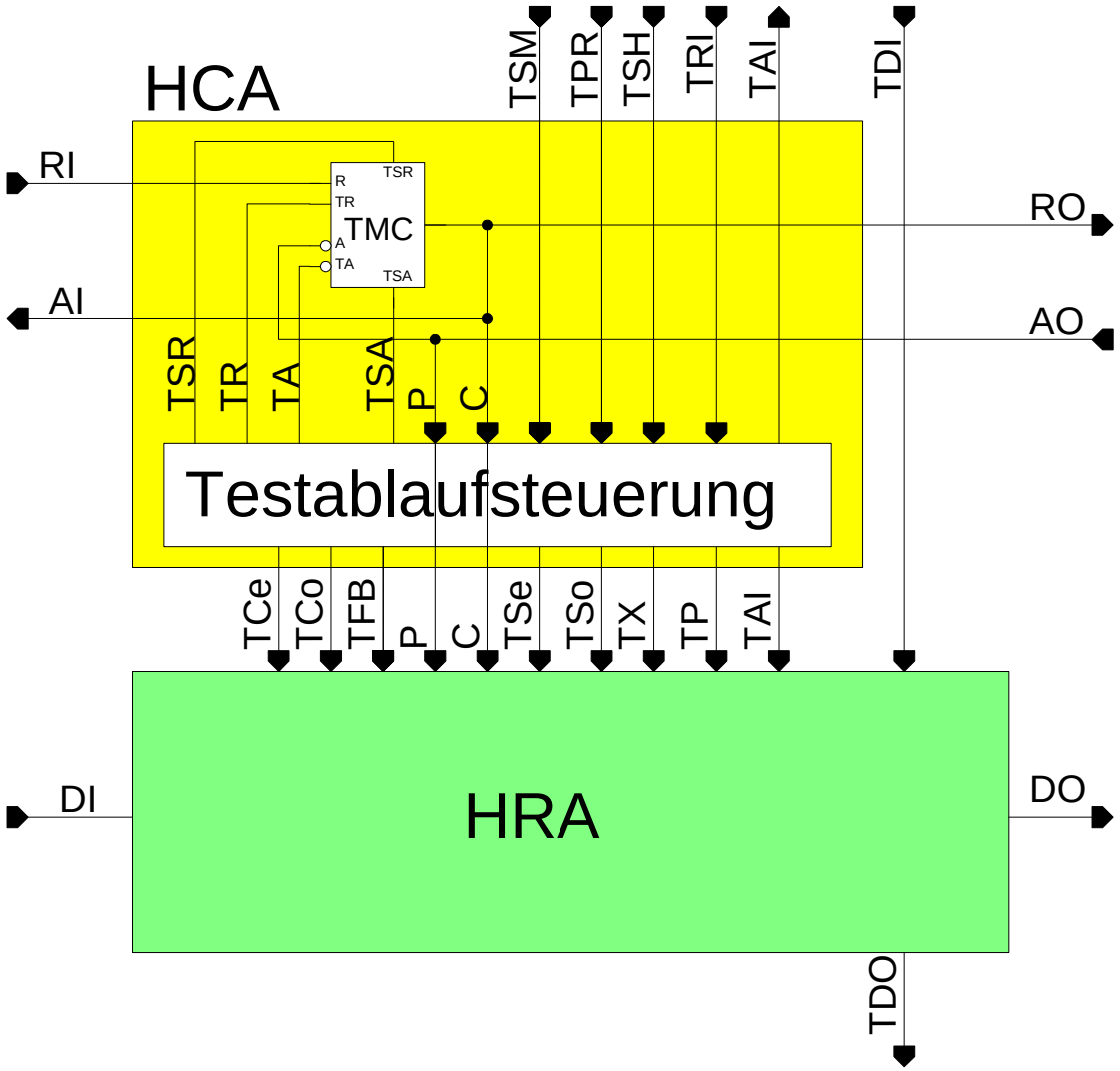

Abb. 74: Die Signale zur Steuerung des Registers für den aktiven testfreundlichen Entwurf.

Die Testablaufsteuerung generiert, abhängig von RO, AO und RI, die Steuersignale für das TMC-Element und die Registerstruktur HRA. TCe und TCo schalten die Speicherelemente (SCe und SCo), die an TDI angeschlossen sind, getrennt voneinander[85]. Die Signale TSe und TSo entscheiden, ob der Inhalt einer Speicherzelle beobachtet oder gesteuert wird. TAI, TP und TX steuern das dritte Register (XL) der asynchronen Speicherzelle ABC. TP dient dabei als Schalter des Dateneingangs in der SCAN und TPG Betriebsart (TX=0). In der Betriebsart zur Testdatenkomprimierung wird mit TX der Schalter in ein XOR

[85] Hierbei ist das Signal RO identisch zu AI. C, AO ist mit P zusammengeschlossen. Siehe hierzu auch **Abb. 72**.

ersetzt[86]. Mit TDI, TDO, TAI und TRI werden Testmuster in das Register ein- und ausgelesen. Sie bilden die Schnittstelle zu externen Automaten, die den Test starten und beenden. Die Betriebsarten sind in **Tabelle 7** dargestellt.

Tabelle 7: Die Teststeuersignale der aktiven Teststeuerung.

TSM	TPR	TSH	Betriebsart	Erklärung
0	0	0	OPERATE	Operationsbetrieb, kein Testbetrieb
0	0	1	SCAN	Serielle Schiebeoperation
0	1	0	PRELOAD	Anlegen einer neuen Testantwort an den Datenpfad
0	1	1	TPG	Testmustergenerierungsbetrieb
1	0	0	SAMPLE	Einlesen einer Testantwort in das Schieberegister
1	0	1	TDA	Testdatenkomprimierungsbetrieb
1	1	0	S2P	Übergang zwischen den beiden Modi SAMPLE und PRELOAD
1	1	1	SCANI	Serielle Schiebeoperation für die Testdatenanalyse

Die Steuersignale aktivieren unterschiedliche Signale. Sind die drei Teststeuersignale deaktiviert, wird der Operationsbetrieb mit Betriebsgeschwindigkeit ausgeführt (**OPERATE**). Mit TSM = "1" wird mit dem nächsten Signalwechsel von TRI eine Testantwort aus dem Datenpfad abgegriffen (**SAMPLE**), welche über den Scanpfad ausgelesen wird. Mit TPR="1" wird in Zusammenhang mit TRI ein neues Datum in den Datenpfad geschrieben (**PRELOAD**), nachdem es in das Register geladen wurde. Ist eine Schiebeoperation aktiv, wird das Signal TSH="1" (**SHIFT**) gesetzt. Im Schieberegisterbetrieb wird dabei nur eine SC Zelle benötigt, die sogenannte "even"-Zelle. Die andere Zelle bleibt in ihrem Initialisierungzustand. Neben diesen drei Betriebsarten gibt es noch die Testbetriebsarten zur Testmustergenerierung (**TPG**) und zur Testdatenkompression (**TDA**). Diese sind jeweils Kombinationen aus den zwei oben beschriebenen Befehlen. Bei der Testmustergenerierung werden einerseits eine rückgekoppelte Schieberegisteroperation, andererseits mit jeder positiven Flanke von TRI ein neues Testmuster in den Datenpfad geschrieben. Analog hierzu erfolgt bei der Testdatenkompression das Speichern eines Datums in dem Schieberegister, verbunden mit einer rückgekoppelten Schiebeoperation des Registers. Der letzte Befehl ist ein

[86] Siehe hierzu **Abb. 44**.

weiterer Schieberegisterbetrieb (**SCANI**), der für die Testdatenkompression benötigt wird. Er verwendet die andere Schieberegisterzelle, die sogenannte "odd"-Zelle.

Zum Ablauf der Signalwechsel während einer Betriebsart werden die einzelnen Signalablaufdiagramme dargestellt. Darauf aufbauend werden STGs entwickelt. Es ergeben sich dadurch STGs für die in **Tabelle 7** beschriebenen Betriebsarten. Aus allen STGs wird dann ein gemeinsamer STG gewonnen. Dieser bildet die Grundlage zur Erstellung des Schaltungsdesigns.

In **Abb. 75** sind die Signalwechsel des aktiven HIOB Konzepts beschrieben. Nach einer Initialisierung (**1**) springt man durch den Signalwechsel von TSM in den SAMPLE Modus (**2**). Die *Bundled Data* Signale für den SAMPLE Betrieb können danach erfolgen (**3, 4**). Über die Aktivierung des TDA Modus' (**5**) können der SCAN Betrieb eingestellt (**6**) und D1 ausgelesen werden[87]. Das Zurückschalten in den TDA Modus (**7**) ermöglicht die Kompression der Daten D2 - D5 (**8 - 9**). Nach Aktivieren des SCANI Betriebs (**10**) kann die Signatur ausgelesen werden. SCANI ist notwendig, da bei der Testdatenkompression die Signatur anders als im SCAN und TPG Betrieb gespeichert ist. Mit (**11**) ist diese Testbetriebsart abgeschlossen. Nach einer neuen Initialisierung (**12**) erfolgt das Anlegen von Testmustern in den Datenpfad. Mit SCAN als Zwischenstufe kann der TPG Modus eingestellt werden (**13, 14**). Im Testmustergenerierungsbetrieb werden Testmuster an den Datenausgang DO angelegt und können über RO und AO an die nächste Pipelinestufe weitergeleitet werden (**15, 16**). Das Einlesen eines Testmusters kann nach den Übergängen (**13, 17**) erfolgen. Zum Anlegen des Testmusters an den Datenpfad (PRELOAD) erfolgen die Übergänge (**18-19**). Die *Bundled Data* Signale des Datenpfads synchronisieren die Datenübertragung mit der nächsten Pipelinestufe (**20, 21**).

[87] Der serielle Ausleseprozeß ist nicht dargestellt.

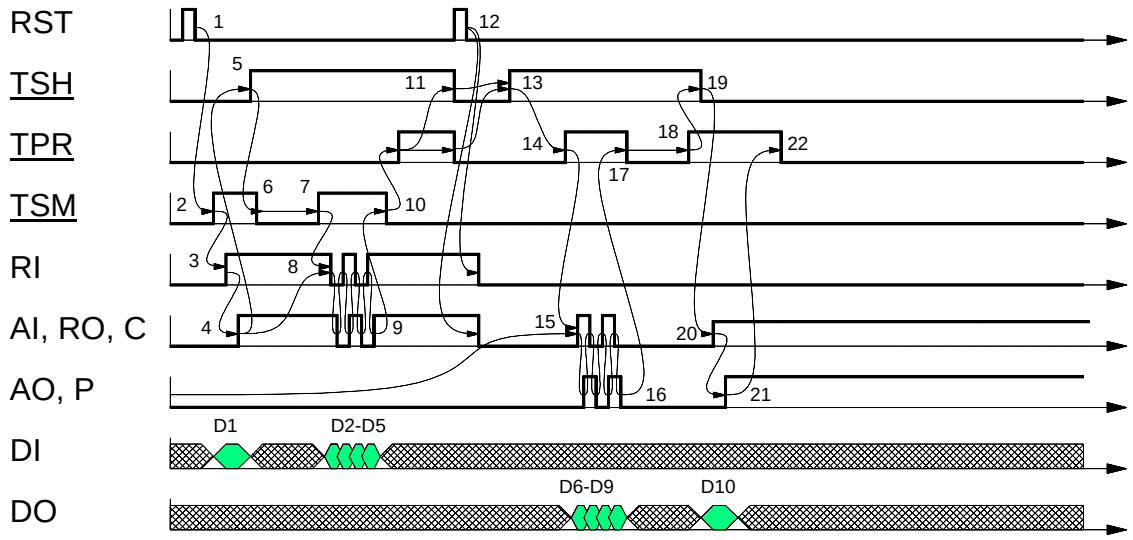

Abb. 75: Die Signalablauffolge zwischen den unterschiedlichen Testbetriebsarten.

Abb. 76 zeigt einen reduzierten STG für das aktive HIOB Konzept. Alle Bedingungen, die zu Signalwechseln führen, sind hier nicht dargestellt, werden jedoch in den folgenden STGs ergänzt[88]. Links befinden sich die Übergänge, die sich aus dem TMC-Element mit seinen *Two-Phase* Signalen ergeben. Die externen *Four-Phase* Signale, TRI und TAI des HIOBA Registers sind rechts dargestellt. Mit TAI- wird die Kommunikation des Datenpfads abgeschlossen, was an den Signalwechseln des TMC-Elements zu erkennen ist. Im SCAN Modus wird direkt der Signalwechsel von TRI- an TAI- weitergeleitet. Die interne Testdatenverarbeitung wird mit dem TRI+ Signalwechsel eingeleitet. Ist eine Schieberegisteroperation erfolgt, quittiert das Register diesen Vorgang mit einem TAI+ Signalwechsel. Die interne Testablaufsteuerung wartet auf den nächsten Signalwechsel von TRI-. TRI+ und TRI- initiieren sowohl die Kommunikation mit dem Datenpfad als auch die interne Signalverarbeitung des Registers. Zwischen den Signalen TRI- und TRI+ erfolgen die Umschaltprozesse von der einen in die andere Testbetriebsart[89]. Nach der allgemeinen Beschreibung werden nun die Signalwechsel für jede einzelne Testbetriebsart entwickelt. Im Anschluß wird ein gemeinsames Modell für die Testablaufsteuerung vorgestellt.

[88] Es wurde hier bewußt auf eine detaillierte Beschreibung aller Signale verzichtet, um die prinzipielle Struktur der Befehle darzulegen.

[89] Im folgenden Verlauf werden die einzelnen Signalübergangsgraphen und die STGs in den Betriebsarten nach **Tabelle 7** dargestellt und deren Ablauf erklärt.

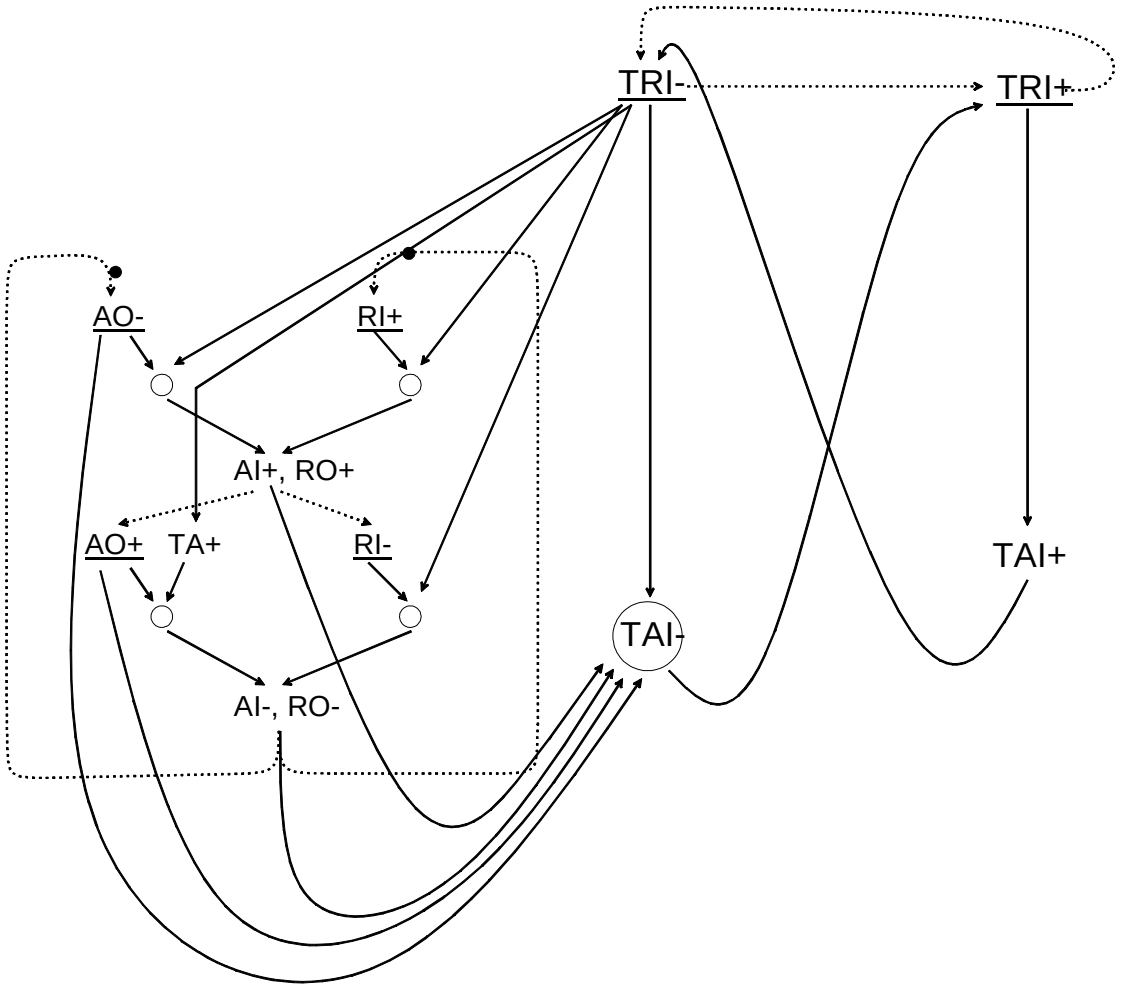

Abb. 76: Der vereinfachte STG des aktiven HIOB Konzepts.

Abb. 77 zeigt das Signalablaufdiagramm für den seriellen Schieberegisterbetrieb. Nach der Initialisierung (**1**) erfolgt mit (**2**) das Einleiten des SCAN Modus', der die Testsignale TSe und TSo schaltet (**3**). Danach wird mit TRI+ (**4**) die Schiebeoperation in die geraden (*even*) Registerzellen eingeleitet. Mit dem Signalwechsel TCe+ (**5**) öffnet sich die Zelle und speichert mit (**6**) das Datum. Dieses erfolgt nur dann, wenn auch der Wechsel TSe+ zuvor erfolgt ist. Mit TAI+ (**7**) quittiert die Steuerung die Speicherung an die globale Steuerung. Das Signal TRI- (**8**) öffnet die zweite Speicherstufe des Registers und TAI- (**9**) quittiert diesen Vorgang. Danach beginnt mit dem Speichern des Datums in der zweiten Registerstufe der nächste Auslesevorgang (**10**). Sind alle Registerstufen gefüllt und ausgelesen, erfolgt mit dem letzten TAI- Signalwechsel (**11**) ein neuer Testmodus. In der Abbildung ist darüber hinaus die andere Schieberegisterbetriebsart (SCANI) dargestellt. Sie wird mit den Signalwechseln TPR+ und TSM+ (**12, 13**)

eingeleitet. Daraufhin invertiert das Signal TP (**14**), um die anderen, die ungeraden Registerzellen zu verwenden. Das Laden der Registerzellen wird mit dem Signal TRI+ (**15**) ausgelöst und mit TCo+ (**16**) das Öffnen der vorderen Speicherzellen der einzelnen Register. Mit (**17**) ist der Vorgang abgeschlossen, und die Signalwechsel TSM-, TPR- und TSH- (**18**) schalten das Register in den Operationsbetrieb. Hierbei werden auch die Signale TSE und TSo zurückgesetzt (**19**).

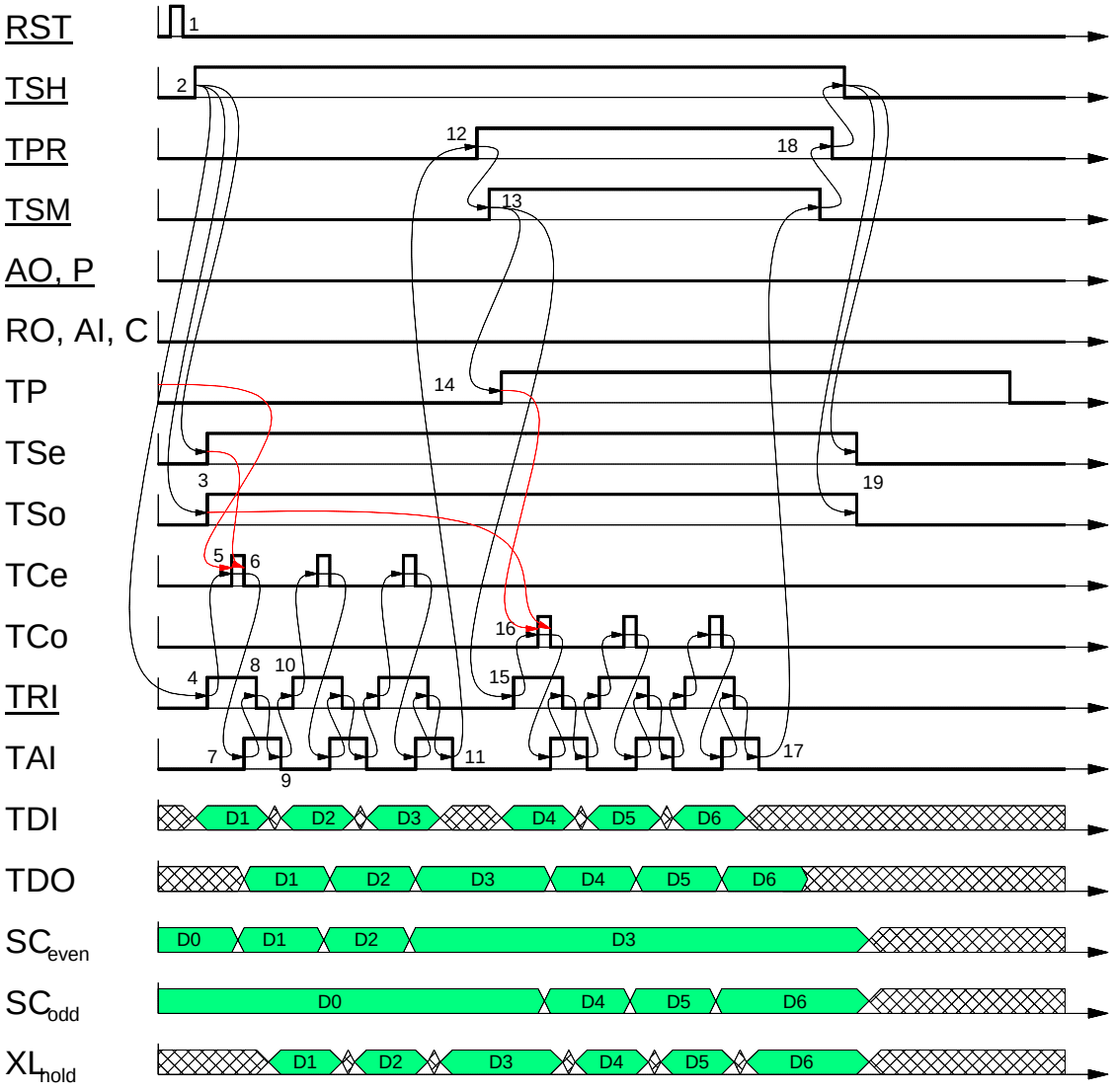

Abb. 77: Das Signalablaufdiagramm für den seriellen Schieberegisterbetrieb.

Abb. 78 zeigt den STG für den seriellen Schieberegisterbetrieb. Im Schieberegisterbetrieb gibt es zwei Arten von Schiebeoperationen, den SCAN und den SCANI Modus. Diese werden abhängig von den Signalen TP, TSe und TSo gestartet. Nach der Initialisierung wird mit TRI+ die Speicherung des Datums am

Eingang des Schieberegisters eingeleitet. Mit TCe+ öffnet die vordere Zelle die Master-Slave Struktur. Mit TCe- wird das Datum gespeichert. TAI+ öffnet die zweite Stufe des Registers. Das Datum liegt am Ausgang des Registers an. Nach TRI- speichert der Signalwechsel TAI- das Datum in der zweiten Stufe. Eine weitere Schieberegisteroperation kann nun eingeleitet werden.

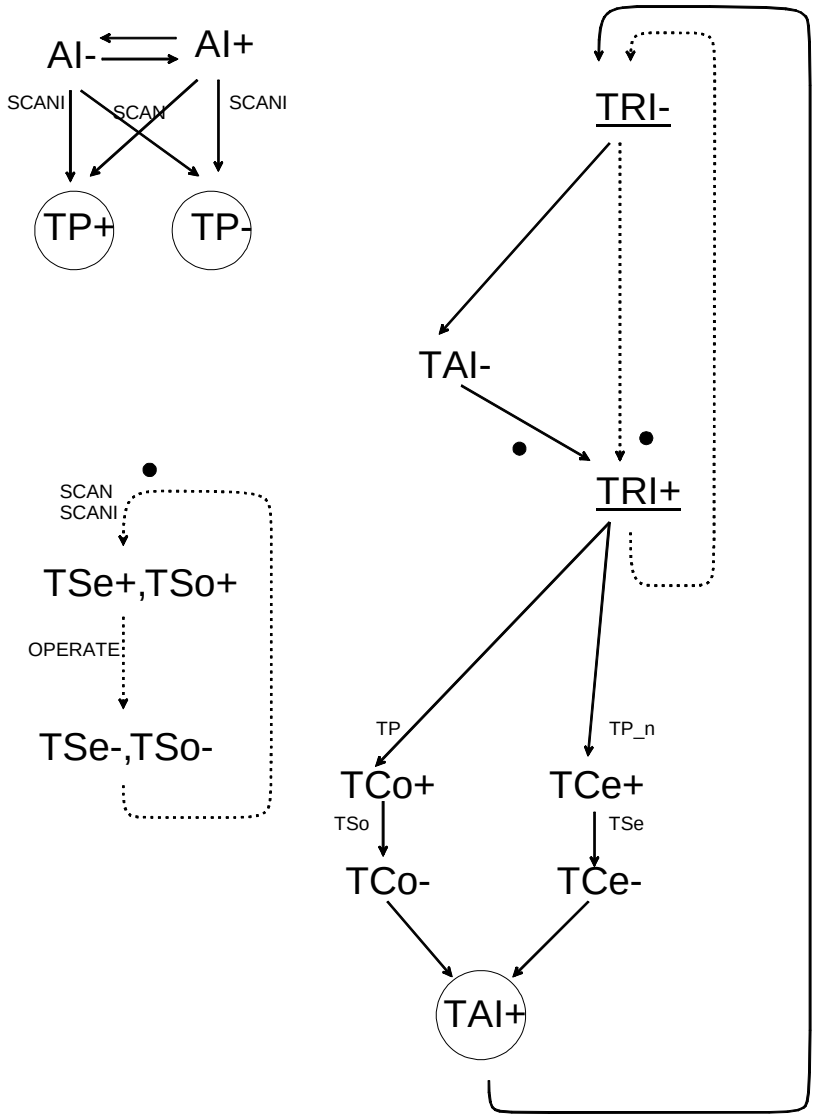

Abb. 78: Der STG für den seriellen Schieberegisterbetrieb.

TP ist hier von den Schalterstellungen AI abhängig, da diese anzeigen, in welcher der Registerzellen das nächste gültige Datum erwartet wird. Dies ergibt sich aus der Notwendigkeit, dass das Datum entweder in die gerade oder ungerade Speicherzelle zu laden oder auszulesen ist. Im Testbetrieb wird über TP entschieden, welches der Speicherzellen mit dem seriellen Datenpfad verbunden

ist. Mit AI wird die Speicherzelle für das nächste Datum ausgewählt. Diese Schieberegisteroperation wird auch während der Betriebsarten TPG und TDA verwendet. Die Signale TSe und TSo bleiben im ganzen SCAN Modus auf "1". Dies wird durch den dritten STG in der Abbildung verdeutlicht[90].

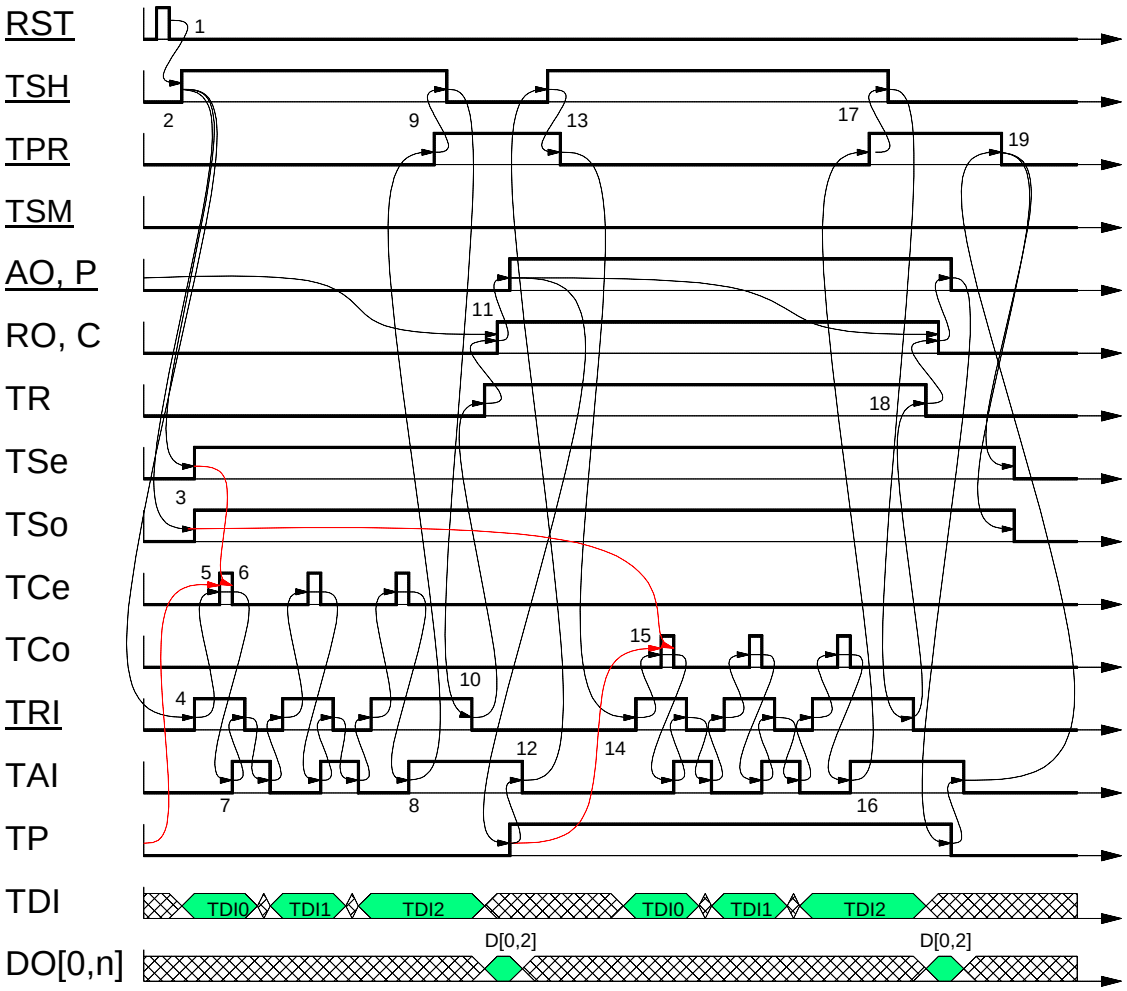

Abb. 79: Das Signalablaufdiagramm des Schieberegisters zum Anlegen von Testdaten in den Datenpfad.

Abb. 79 zeigt das Signalablaufdiagramm des PRELOAD Modus´ im Zusammenhang mit dem seriellen Schieberegisterbetrieb. Die Schieberegisteroperation wird stets vor dem PRELOAD-Befehl ausgeführt, um Testdaten in das Register einzulesen. Nach der Initialisierung (**1**) wird das Register in den Schieberegisterbetrieb geschaltet (**2, 3**). Danach erfolgt das Einlesen der Testmuster (**4-8**). Ist das Scanregister gefüllt, wird mit TAI+ (**8**) der PRELOAD Modus (**9**) aktiviert. Mit

[90] Zum Umschalten zwischen den Betriebsarten müssen die drei externen Steuersignale geschaltet werden. Dies gilt auch für die folgenden Testbetriebsarten. STGs mit diesen Umschaltevorgängen sind im **Anhang** zu finden.

TRI- (**10**) wird das *Bundled Data* Signal (**11**) durch TR freigegeben. TR ersetzt dabei das *Request* der vorhergehenden *Micropipeline* Stufe. Der Signalwechsel von AO+ (identisch zu P+) speichert das Datum in der nachfolgenden Stufe. Die Testablaufsteuerung quittiert dies mit dem TAI- Signal (**12**). Nach dem Beenden von PRELOAD wechselt das Register in den Scanbetrieb (**13**) zurück, und ein erneutes Einlesen eines Testmusters (**14, 15**) erfolgt, um im Anschluß widerum in den PRELOAD Modus (**17**) zu wechseln. Danach wird wieder in den Operationsmodus (**19**) geschaltet.

In **Abb. 80** ist der STG für den PRELOAD Modus dargestellt. Er enthält den Scanbetrieb, da das Datum des Datenpfads zuvor seriell in das Register geladen wird. Die Signale TSe und TSo werden im ganzen PRELOAD Modus auf "1" gehalten. Zwischen SCAN und PRELOAD Betrieb wird durch externe Signale umgeschaltet, die widerum durch *Input choices* ermöglicht werden. Man erkennt auf der linken Seite des STGs die Signalwechsel des *Muller C-Elements*. Auf der rechten Seite befinden sich die internen Schaltvorgänge des Registers. Durch TAI und TRI werden Übergänge zwischen den beiden Teilgraphen gesteuert.

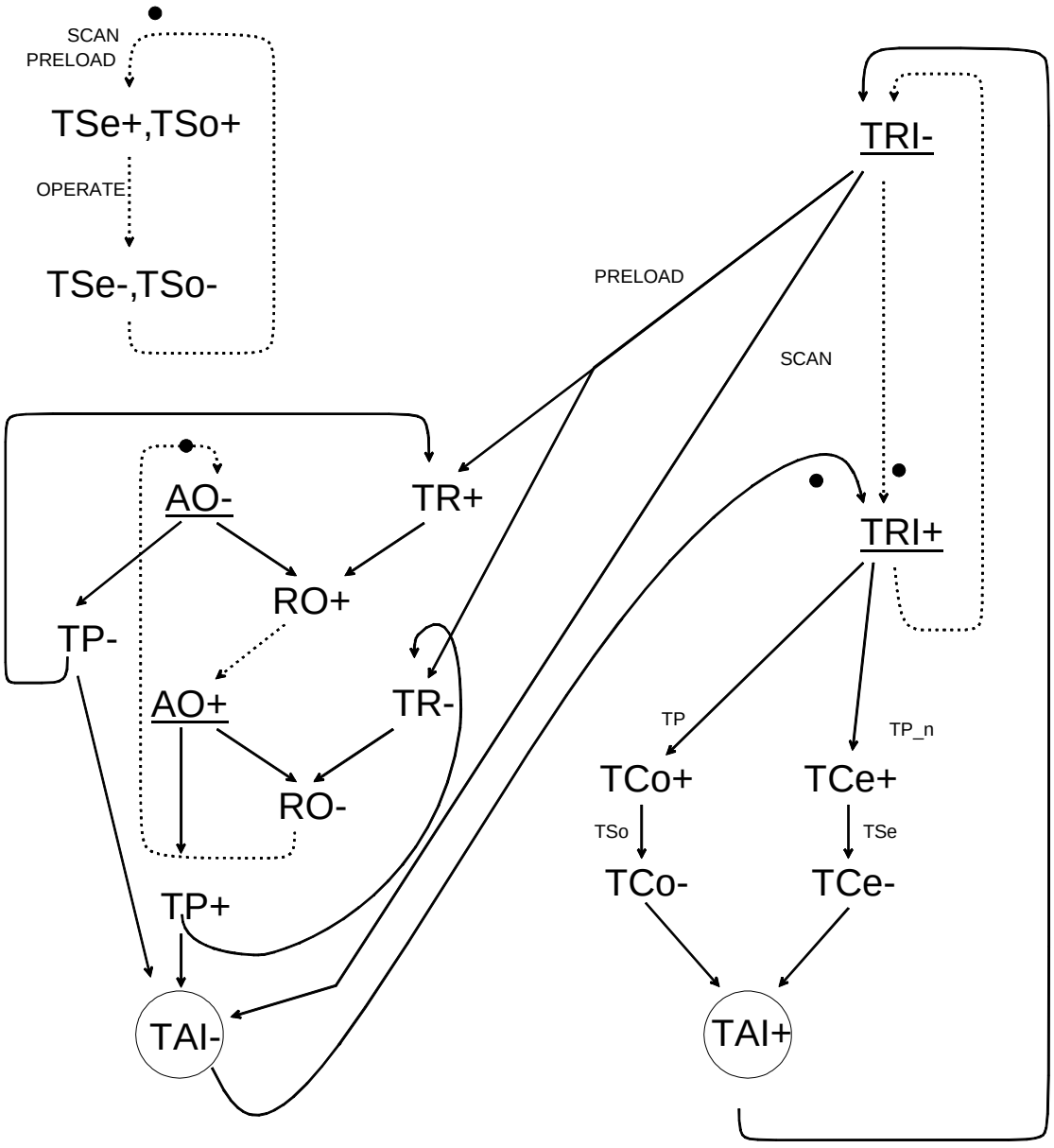

Abb. 80: Der STG zum Anlegen eines Testmusters aus dem Schieberegister an den Datenpfad.

Es wird nun der SAMPLE Betrieb untersucht. Hier ist, wie auch beim PRELOAD Befehl, der SCAN Modus notwendig. Es werden damit die Testdaten aus der Schaltung ausgelesen, um sie mit der Sollantwort zu vergleichen. **Abb. 81** zeigt das Signalablaufdiagramm für die Übernahme einer Testantwort in das Register sowie den anschließenden seriellen Schieberegisterbetrieb zum Auslesen des Testmusters. Nach der Initialisierung schaltet die Steuerung in den Sample Modus (**1**). Anschließend schalten die Signale TSo und TSe (**2**). Es folgt ein Schieberegisterzyklus (**3**), um den SAMPLE Modus zu initialisieren. Dies ist mit

TAI+ (**4**) abgeschlossen. Die externen Steuersignale TSM+ und TSH- (**5**) schalten, um in den SAMPLE Modus zu gelangen. Mit TRI+ (**6**) wird TA- ausgelöst, welches das TMC-Element für die *Bundled Data* Signale RI und AI (**7**) öffnet. TAI+ (**8**) quittiert das Speichern des Datums. Danach schaltet die Steuerung in den Scan Betrieb (**9**). Das Steuersignal TSe des Registers schaltet für den SCAN Betrieb zurück. Mit TRI+ beginnt das Auslesen der Testantwort (**10**). Mit (**11**) ist dieser Vorgang abgeschlossen, und ein weiteres Testmuster kann im Register gespeichert werden (**13-23**).

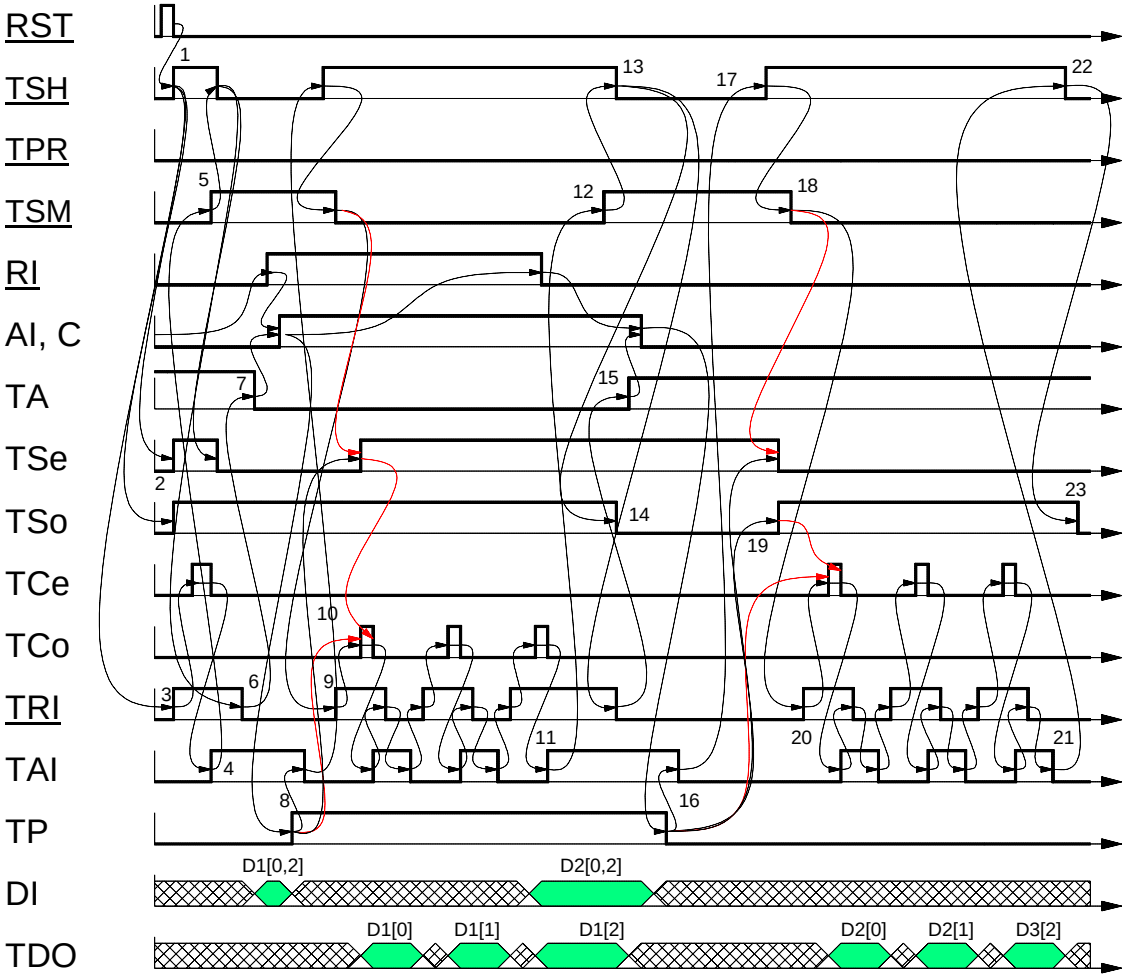

Abb. 81: Das Signalablaufdiagramm für die SAMPLE Betriebsart.

In **Abb. 82** ist der STG des SAMPLE Modus' dargestellt. Auch hier ist, wie schon bei dem PRELOAD Modus, eine Kombination zwischen Abgreifen des Datums aus dem Datenpfad und einem Schieberegisterbetrieb dargestellt, um das Datum außerhalb des Schaltkreises mit der Sollantwort zu vergleichen. Um dies zu

erreichen, wird durch die externe Steuerung zwischen SCAN, SCANI und SAMPLE geschaltet. Mit TAI- wird die Testantwort im Datenregister gespeichert, und der Auslesevorgang kann eingeleitet werden. Nachdem das Datenwort mit mehreren Durchläufen des Scanbetriebs ausgelesen ist, erfolgt die erneute Umschaltung zum Abgreifen einer neuen Testantwort. Die zusätzlichen Teilgraphen des STGs beschreiben die Steuerung der Schalter der Registerzellen. Die Trennung der Graphen wurde aus Übersichtsgründen gewählt.

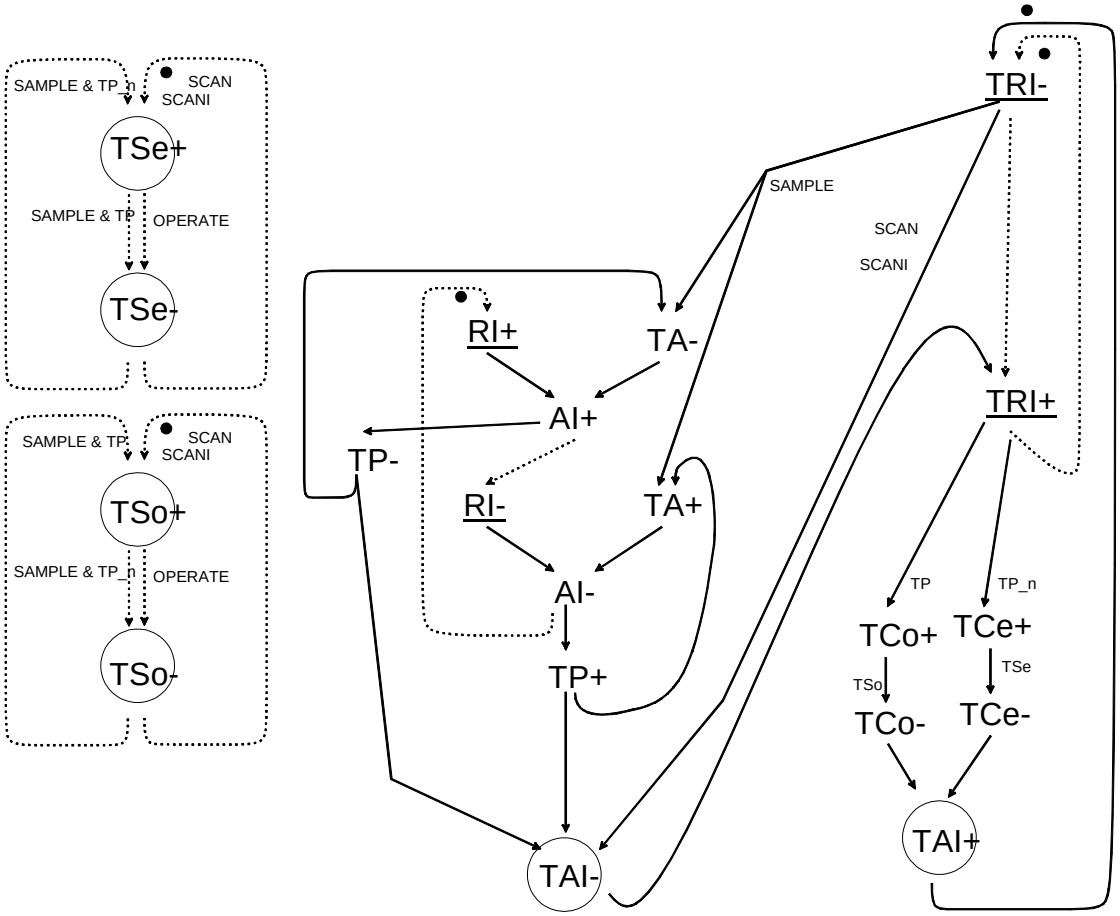

Abb. 82: Der STG zum Einlesen der Testantwort in den seriellen Schieberegisterpfad.

Im Anschluß an den SAMPLE Betrieb wird nun der TPG Modus erklärt. **Abb. 83** stellt das Signalablaufdiagramm für den Testmustergenerierungsbetrieb dar. Nach einer Initialisierung des Registers (**1**) und dem Signalwechsel TSH+ befindet sich das Register im Schieberegisterbetrieb, der mit den Signalwechseln TSe+ und TSo+ (**2**) beginnt und dem Einlesen des Startvektors (**3-4**) endet. Hierfür ist eine Anzahl von Schieberegisteroperationen notwendig, die nicht dargestellt sind. Danach schaltet das Register durch TPR+ (**5**) in den Test-

mustergenerierungsbetrieb und schaltet mit TRI+ (**6**) und TR+ (**7**) das TMC-Element in den Bereitschaftsmodus. Das Testmuster wird in den Datenpfad geschrieben und die Quittung der zweiten Pipelinestufe des Datenpfads erwartet (**8**). Ist dies erfolgt, quittiert das HIOBA Register mit TAI- (**9**). TRI+ (**10**) leitet die Berechnung des nächsten Testvektors ein. Mit der folgenden negativen Flanke von TRI wird TR+ (**11**) gesetzt. Die nächste *Bundled Data Convention* kann erfolgen (**12**), die das Ende mit einen TAI- Signalwechsel (**13**) abschließt. Danach kann das letzte generierte Testmuster ausgelesen werden (**15-19**).

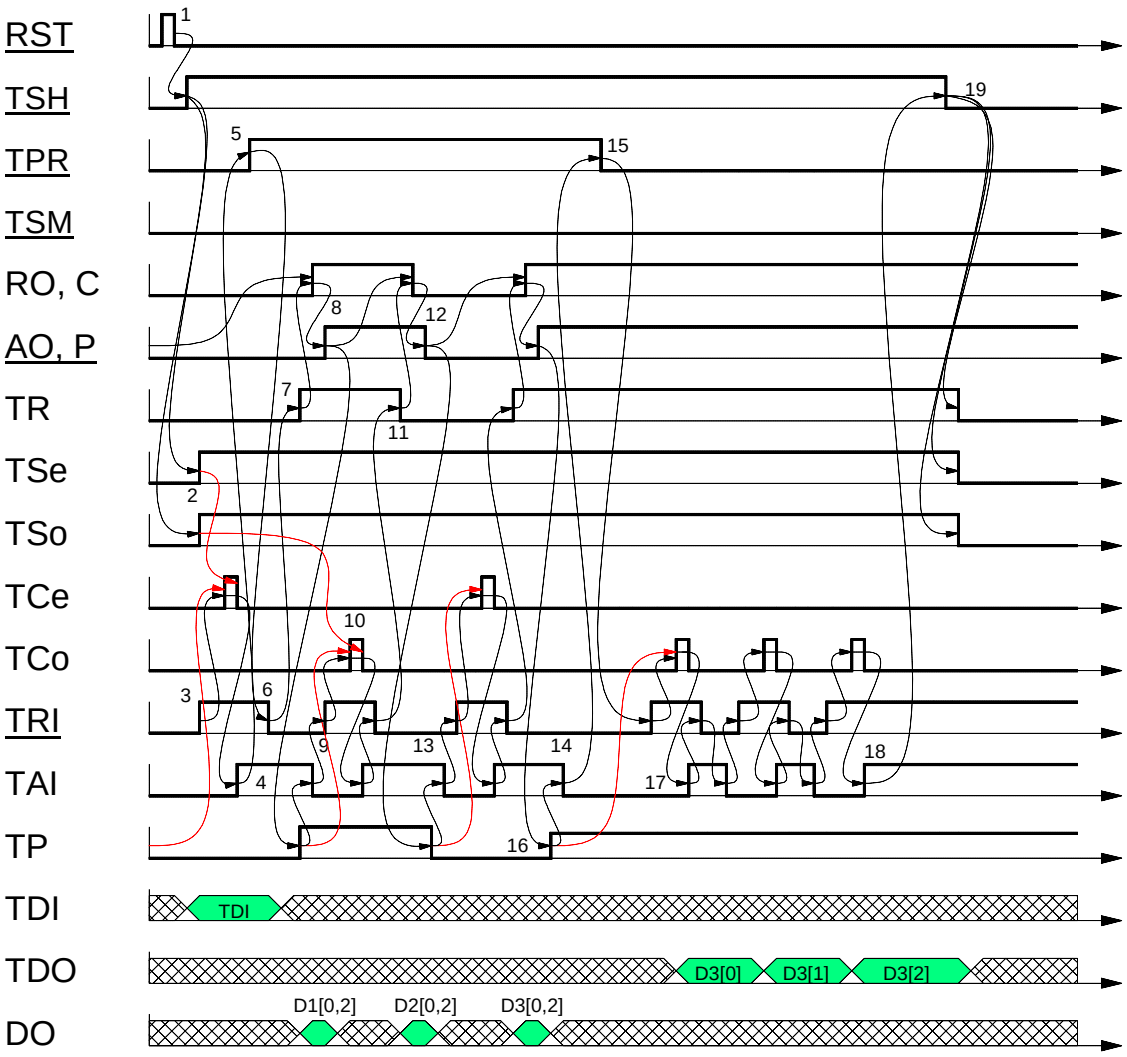

Abb. 83: Das Ablaufdiagramm der Steuersignale für den Testmustergenerierungsbetrieb.

Abb. 84 zeigt den STG des Testmustergenerierungsbetriebs (TPG). Man erkennt, dass alle Signale zur Testablaufsteuerung intern generiert werden. Das TRI Signal dient zur Synchronisation und Kontrolle des Testablaufs. Hiermit kann

ein Zähler angesteuert werden, der die Anzahl der generierten Testmuster zählt, um aus dem Testmustergenerierungsbetrieb wieder in eine andere Betriebsart zu springen.

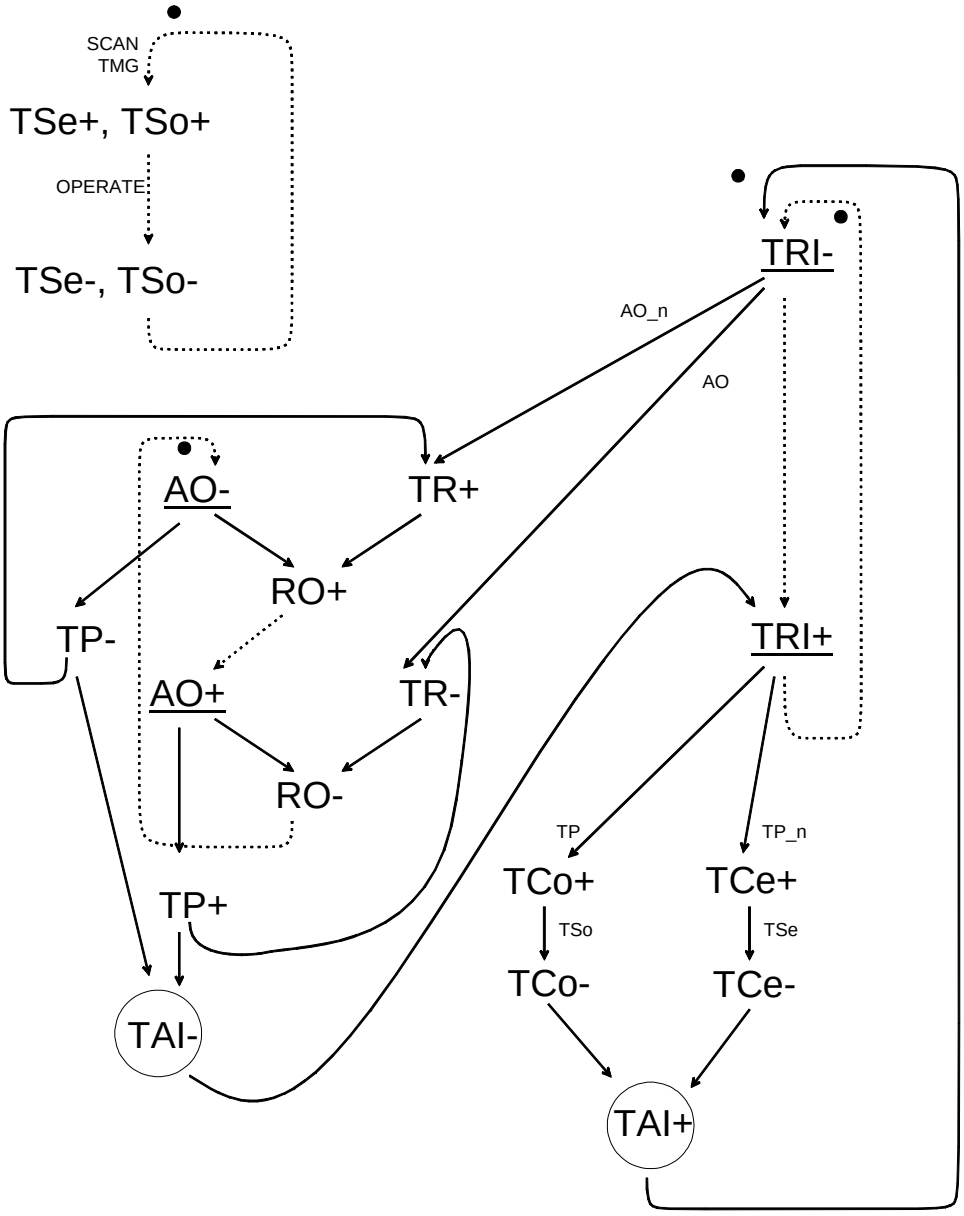

Abb. 84: Der STG zur Testmustergenerierung.

Die letzte und komplizierteste Betriebsart dient zur Testdatenkomprimierung (TDA). Sie ist aufwendiger als der TPG Betrieb, da neben der Generierung des

nächsten Testmusters auch das aktuelle, am parallelen Eingang anliegende Datum berücksichtigt werden muß. In **Abb. 85** ist das Signalablaufdiagramm für die Testdatenkompression dargestellt. Nach der Initialisierung (**1**) erfolgt der Wechsel in den SCANI Modus (**2**). Diese Schieberegisterbetriebsart ist hier erforderlich, da die zweite Speicherzelle für das folgende gültige Datum reserviert ist. Hierbei wird in der nicht benötigten vorderen Speicherzelle ein Startvektor geladen. Beide Signale sind notwendig, um die nächste Signatur mit Hilfe einer XOR-Verknüpfung zu speichern. Für den SCANI Modus werden zuerst TP, TSe und TSo in die entsprechenden Zustände gesetzt (**3**). Danach erfolgt die gewünschte Anzahl von Schieberegisteroperationen, welche mit TAI+ (**4**) abgeschlossen werden. TPR- erfolgt (**5**), damit die Testdatenkomprimierung beginnen kann. Dabei wird die XOR-Verknüpfung in der XL Speicherzelle aktiviert (**6**). Die Testdatenkompression wird mit RI+ und TRI+ ausgelöst. Die Wechsel TA+ und AI+ speichern das Testdatum im Register (**8**). Der Testdatensender kann danach bereits ein weiteres Datum berechnen. Mit (**9-16**) erfolgt die Testdatenkompression im HIOBA Register. Der SCANI Modus liest die Signatur des Registers aus (**17-23**). Diese Betriebsart verwendet beim Auslesen im Vergleich zum SCAN Modus die entgegengesetzten Registerzellen der seriellen Pipeline.

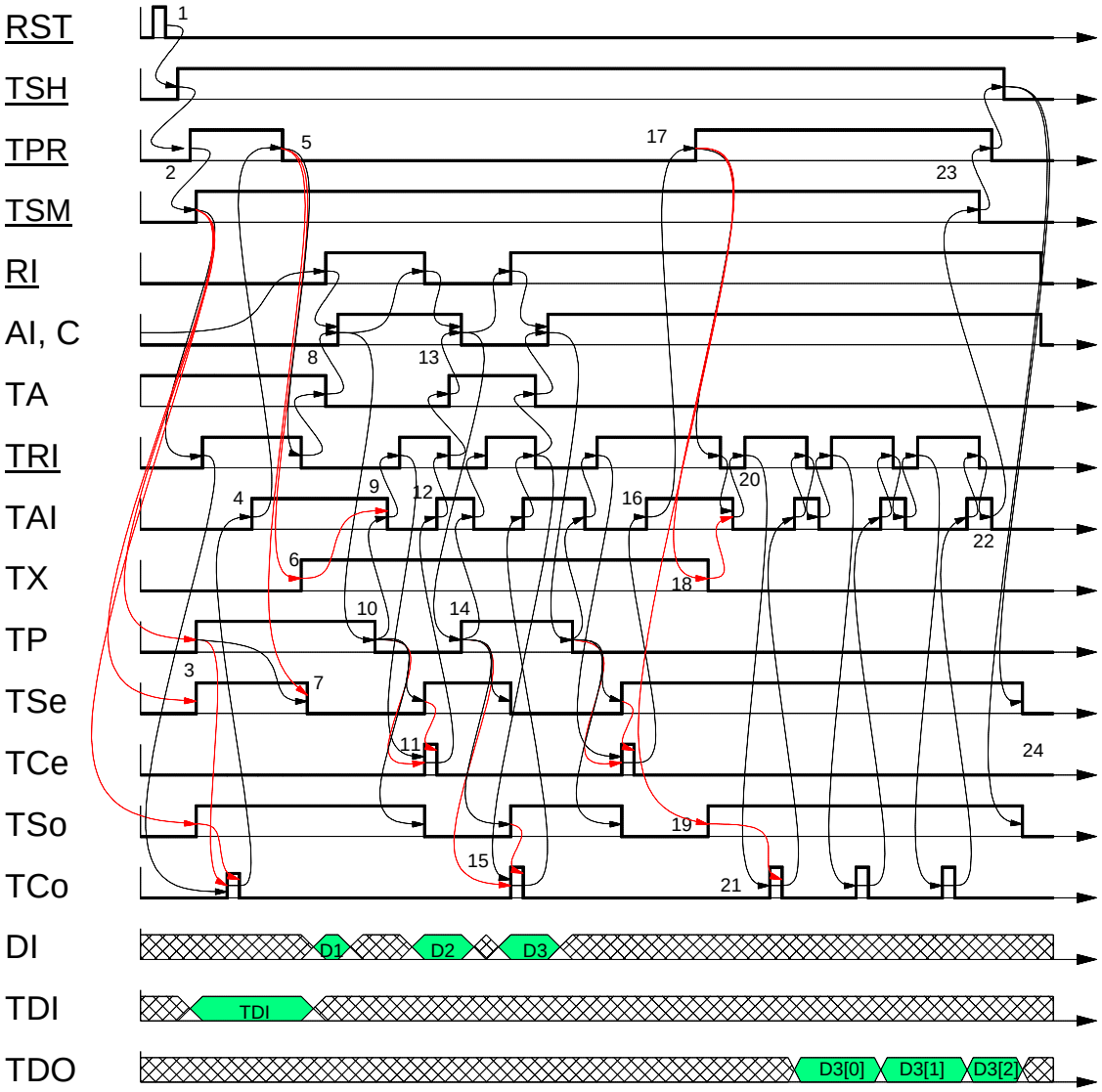

Abb. 85: Das Ablaufdiagramm für die Testdatenkomprimierung.

In **Abb. 86** ist der STG für die Testdatenkompression dargestellt. Man erkennt wieder die Signale des TMC-Elements, die internen Signale des Registers sowie das externe Testsignal TRI, mit dem die Testmustergenerierung extern gesteuert werden kann. Die zwei Teilgraphen auf der linken Seite beschreiben Signale bzw. Schalterstellungen des HIOBA Registers, die in Abhängigkeit der Testbetriebsarten ihren Zustand ändern.

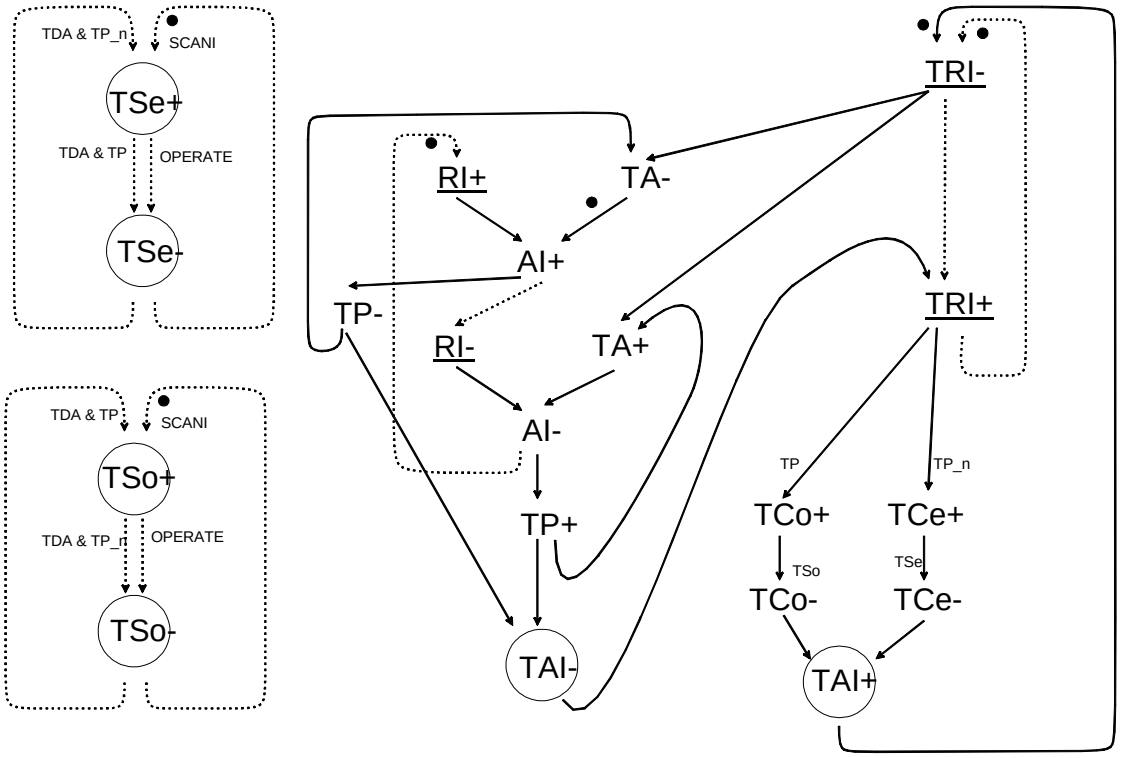

Abb. 86: Der STG für die Testdatenkomprimierung.

Nachdem alle unterschiedlichen Betriebsarten einzeln erklärt sind, werden sie in einen STG überführt, aus dem die gesamte Testablaufsteuerung ermittelt werden kann. In **Abb. 87** ist dieser STG der Testablaufsteuerung dargestellt. Aus Übersichtsgründen werden wieder die drei Teilgraphen der Schalter an den Registerzellen gesondert behandelt. TP, TSe und TSo dienen dabei als Bedingungen (*Input choice*) für die Verzweigungen und das Zusammenführen von Signalen des oberen STGs. Auf der linken Seite ist in dem STG das TMC-Element zu erkennen. Oben befindet sich das externe TRI Signal. Aus dem *Two-Phase* Signal TP wird das *Four-Phase* Signal TAI gewonnen, welches mit TAI- die interne Steuerung der Speicherzellen fortführt. Mit TAI+ ist die interne Signalverarbeitung abgeschlossen, und neue Daten können eingelesen oder an die nächste Stufe weitergegeben werden. TRI- leitet die Verarbeitung von Signalen mit TMC und somit die Synchronisation mit dem Datenpfad ein. Mit TRI+ wird die interne Steuerung des Registers, z.B. der Scanbetrieb, begonnen.

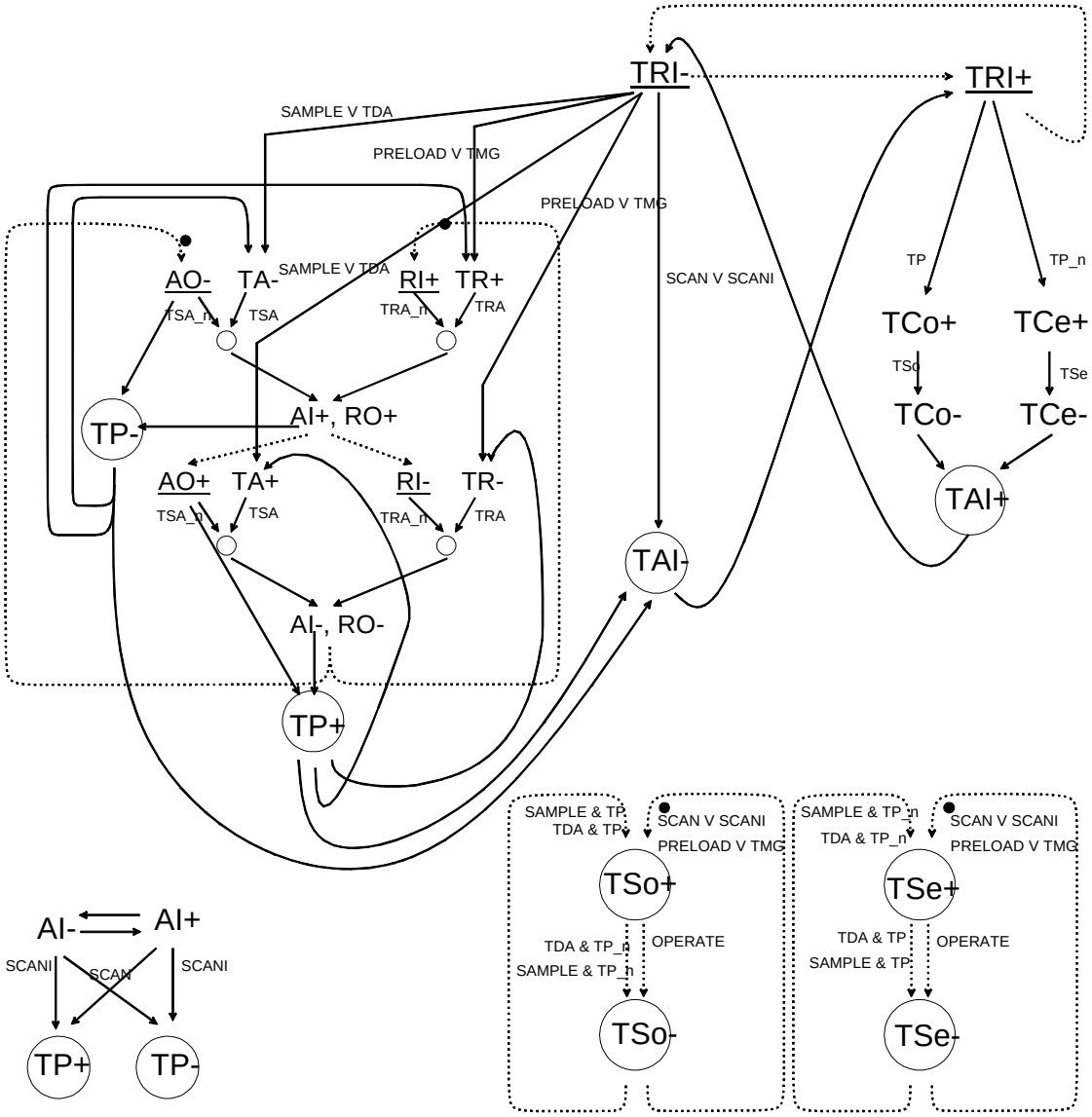

Abb. 87: Der STG der gesamten Testablaufsteuerung des aktiven HIOB Konzepts.

Zur Generierung aller Zustandsfolgen im Testbetrieb wird ein Modul vorgeschlagen, das in Abhängigkeit von den Betriebsarten die nötigen Signalwechsel für das HIOBA Register erzeugt. Eine mögliche Realisierung für eine Testablaufsteuerung eines asynchronen BILBOs wird in **Abb. 88** dargestellt[91]. Rechts oben befindet sich das TMC-Element zur Ablaufsteuerung des Datenpfads. Wie schon bei dem passiven testfreundlichen Entwurf sind die Signalleitungen des Datenpfads, C und P, nicht durch Logik unterbrochen. Dies bedeutet für den Datenpfad, dass im Operationsbetrieb keine Performanceverluste auftreten.

[91] Die Initialisierung über die Rücksetz- und Setzeingänge der Module ist aus Übersichtsgründen nicht dargestellt.

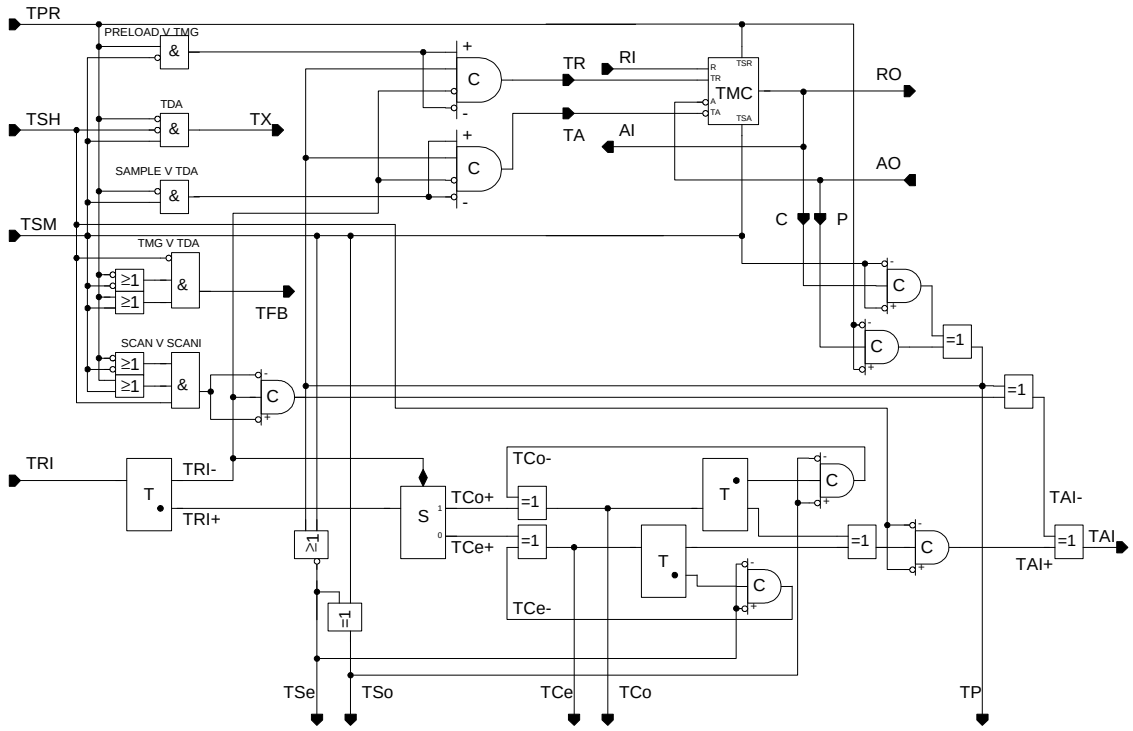

Abb. 88: Die globale Steuerung des HIOB Registers erfolgt durch einen konzentrierten Baustein.

Zur Reduktion der Anzahl der Eingänge wurden die Betriebsmodi kodiert und müssen zur Verarbeitung innerhalb der Testablaufsteuerung dekodiert werden. Diese Dekodierung ist auf der linken Seite zu finden. Mit diesen Signalen werden die *Muller C-Elemente* in einen Verharrungszustand versetzt. Unten ist die Ablaufsteuerung der Signale des Testregisters zu sehen. Man erkennt die Bereiche, in denen mit Hilfe eines XOR Gatters[92] von *Two-Phase* auf *Four-Phase* Signale geschaltet bzw. mit *Toggle* Elementen auf *Two-Phase* Signale zurückgesetzt wird.

[92] Die Signalleitungen des XORs sind mit TCe+ und TCe+ gekennzeichnet.

5.3 Diskussion der testfreundlichen Entwurfsverfahren

Es werden die Eigenschaften der vorgestellten testfreundlichen Entwurfshilfen für asynchrone Schaltungen zusammengefaßt. Mögliche Einsatzgebiete in der digitalen Schaltungstechnik, für die die asynchronen Testkonzepte neue Impulse geben können, werden beschrieben. Aus der Entwicklung beider Konzepte ergaben sich auch Bibliothekselemente für den testfreundlichen Entwurf von integrierten Schaltungen. Die Diskussion zu diesen Bibliothekselementen ist in **Kapitel 4.3** zu finden.

Micropipelines besitzen ein sehr komplexes Signalverhalten. Durch die Umsetzung der Signale zwischen seriellem und parallelem Testbetrieb nimmt die Komplexität der Steuersignale zu. Dieses führt bei einem asynchronen testfreundlichen Entwurf zu einem hohen Schaltungsaufwand bei der Steuerung des Tests. Durch die parallele Anordnung der *Capture-Pass* Register ist ein erhöhter Schaltungsaufwand für den testfreundlichen Entwurf nötig, da beide Zellen während des Betriebs abwechselnd gültige Daten speichern.

Der hohe Schaltungsaufwand ergibt sich aus dem *Two-Phase Signalling*, welches mit ereignisgesteuerten Signalen arbeitet. Bei der Verwendung von pegelgesteuerten Signalen kann neben der Größe der Registerzellen auch der Schaltungsaufwand zur Erzeugung der Steuersignale reduziert werden.

Das passive HIOB Konzept ermöglicht das serielle Einlesen von Testdaten mit asynchronen Signalwechseln. Das Umschalten in den parallelen Datenpfad des zu testenden Schaltungsmoduls erfolgt hier mit einer lokalen Steuerung. Für einen lokal gesteuerten Scanpfad mit der passiven HIOB Methode nimmt die Schaltungsfläche linear mit der Länge des Scanpfads zu. Die Steuerelemente zur Umschaltung zwischen dem parallelen und dem seriellen Betrieb können dabei als Makrozellen optimiert und nach dem Carry-Look-Ahead Prinzip angeschlossen werden. Das passive Testkonzept kann nur in begrenztem Maße

Verzögerungsfehler berücksichtigen. Das dynamische Verhalten des passiven HIOB Konzepts wurde in **[96Kiel]**$_{L.3}$ untersucht. Mit jedem Bit wurde dabei ein neues gültiges Datum eingelesen und an den parallelen Datenpfad angelegt. Dieses Verfahren reduziert deutlich die anzulegende Testdatenmenge. Hierfür können sogenannte periodische Muster nach dem Prinzip von **[94HeinG]**$_{L.1}$ mit Hilfe eines Testmustergenerierungsprogramms ermittelt werden.

- **Das passive HIOB Konzept ermöglicht erstmals neben dem Test des Daten- und des Steuerpfads den Test der *Bundled Data Convention* einer asynchronen Schaltung.**

Für den testfreundlichen Entwurf von A/D Wandlern eignet sich die PISO Betriebsart des asynchronen Scanpfads. Abhängig von der Zeitdauer der Signalumsetzung können unterschiedliche Genauigkeitsstufen des Signals erzeugt werden, wenn die internen Zustände der *Muller C-Elemente* der einzelnen Scanzellen beobachtet werden. In **[87Chua]**$_{L.2}$ wird der Aufbau eines A/D Wandlers mit asynchroner Schaltungstechnik beschrieben.

- **Mit dem aktiven HIOB Konzept können Testdaten seriell ein- und ausgelesen sowie Testmuster asynchron generiert und kompaktiert werden.**

Das aktive Testkonzept besitzt teilweise eine lokale Steuerung. Die serielle Schieberegisteroperation wird von einem globalen Signal und nicht mit lokalen Steuerelementen ausgelöst. Die Generierung der Testmuster kann durch die Signale TAI und TRI von einer äußeren Steuerung synchronisiert werden. Bei dem aktiven HIOB Konzept ist der Steuerungsaufwand unabhängig von der Anzahl der Registerstufen. Der Aufbau teilt sich dabei in eine Testablaufsteuerung und ein Datenregister auf. Da dieses Konzept selbständig Folgemuster bestimmt, ist eine Logik vorzusehen, mit deren Hilfe der Testmustergenerator angehalten werden kann. Ein Zähler kann hier verwendet werden, um die Anzahl der zu generierenden Testmuster zu protokollieren.

Das aktive HIOB Konzept eignet sich für eine synchrone Testumgebung, da der serielle Schieberegisterbetrieb mit einem synchronen Takt verbunden werden

kann. Dies wird dadurch erreicht, dass die Schnittstelle zum Testsender und Testempfänger mit pegelgesteuerten Signalen arbeitet. Der TRI Eingang wird dabei an die Taktversorgung angeschlossen. Durch Beobachten des *Acknowledge* Signals werden die Signalverzögerung ermittelt und die Taktrate variiert. Im Testmustergenerierungs- und Testdatenkompressionsbetrieb kann das *Acknowledge* Signal auf den *Request* Eingang rückgekoppelt werden. Somit wird die maximale Verarbeitungsgeschwindigkeit erreicht. Ein Zähler im Testsender oder im Empfänger beobachtet die Anzahl der Signalwechsel und unterbricht die Testbetriebsarten. Hierbei muß der Zähler die Signale schneller als der Testmustergenerator verarbeiten, um rechtzeitig die Testmustergenerierung unterbrechen zu können.

- **Das aktive HIOB Konzept ermöglicht die Verwendung von synchronen seriellen Testdaten von Prüfautomaten in asynchronen Schaltungen. Die Testmustergenerierung und -kompression erfolgt innerhalb der Schaltung mit der Verarbeitungsgeschwindigkeit des zu testenden Moduls.**

Zur Überprüfung der Schaltung auf dynamische Fehler ist das aktive HIOB Konzept zu verwenden. Durch die aktiven Testhilfen werden Testmuster mit der Geschwindigkeit des zu testenden Moduls generiert. Somit können auch nichtklassische Fehlermodelle bei der Testmusterberechnung berücksichtigt werden.

- **Das aktive HIOB Konzept ermöglicht den Test einer asynchronen Schaltung mit *Single-Rail* Signalen auch für Testmuster, die mit Hilfe von nichtklassischen Fehlermodellen bewertet wurden. Somit kann auch der Steuerpfad auf die Einhaltung der *Bundled Data Convention* auf dynamische Fehler überprüft werden.**

Die vorgestellten Konzepte basieren auf einer Ereignissteuerung im Datenpfad. Der Testsender und der Testempfänger arbeiten bei den Verfahren unterschiedlich. Die CMOS Technik verwendet in der heutigen Digitaltechnik Signalpegel zur Informationsverarbeitung. Testautomaten sind daher in der Regel pegelgesteu-

ert. Für einen pegelgesteuerten Testautomaten müssen die ereignisgesteuerten Signale von HIOBP auf *Four-Phase* Signale umgewandelt werden. Auch die Prüfprogramme gehen von einem synchronen Testablauf mit einer globalen Taktsteuerung aus, da es bei einem synchronen Schaltungstest keine lokalen Synchronisationselemente innerhalb der Schaltung gibt, wie mit den *Self-timed Circuits* von Seitz vorgeschlagen wurde. Testsender und -empfänger können sich dabei innerhalb des Chips, aber auch auf dem Load Board oder im Testautomaten befinden.

- **Beim Produktionstest asynchroner Schaltungen ist zu bedenken, dass die Testumgebung auf einen synchronen, pegelgesteuerten Schaltungstest ausgerichtet ist.**

In **Abb. 89** ist eine Konfiguration des passiven Testkonzepts zu sehen. Hier sind neben dem Datenpfad auch die Testsignale TRI, TAI, TRO und TAO ereignisgesteuerte Signale. Die Umschaltung zwischen seriellem und parallelem Betrieb erfolgt in der Testablaufsteuerung von HIOBP. Die Umschaltung wird in der ersten Stufe mit der SIPO Betriebsart, in der zweiten Stufe mit der PISO Betriebsart durchgeführt. Somit können Testmuster, die seriell im Testsender anliegen, in den Datenpfad eingelesen und über das zweite HIOB Register seriell ausgelesen werden. Die Auswertung der Ergebnisse erfolgt im Testempfänger. Neben den Testbetriebsarten, die eine Umsetzung zwischen parallelem und seriellem Betrieb durchführen, kann mit dem Schieberegisterbetrieb eine Kette von Scanpfaden ermöglicht werden. Diese können zwischen dem seriellen Testausgang der ersten Stufe und dem seriellem Testeingang der zweiten Stufe eingefügt werden. Auch ist es möglich, dass im DUT mehrere Pipelinestufen enthalten sind. Die *Request* und *Acknowledge* Signale sind dann dementsprechend verzögert.

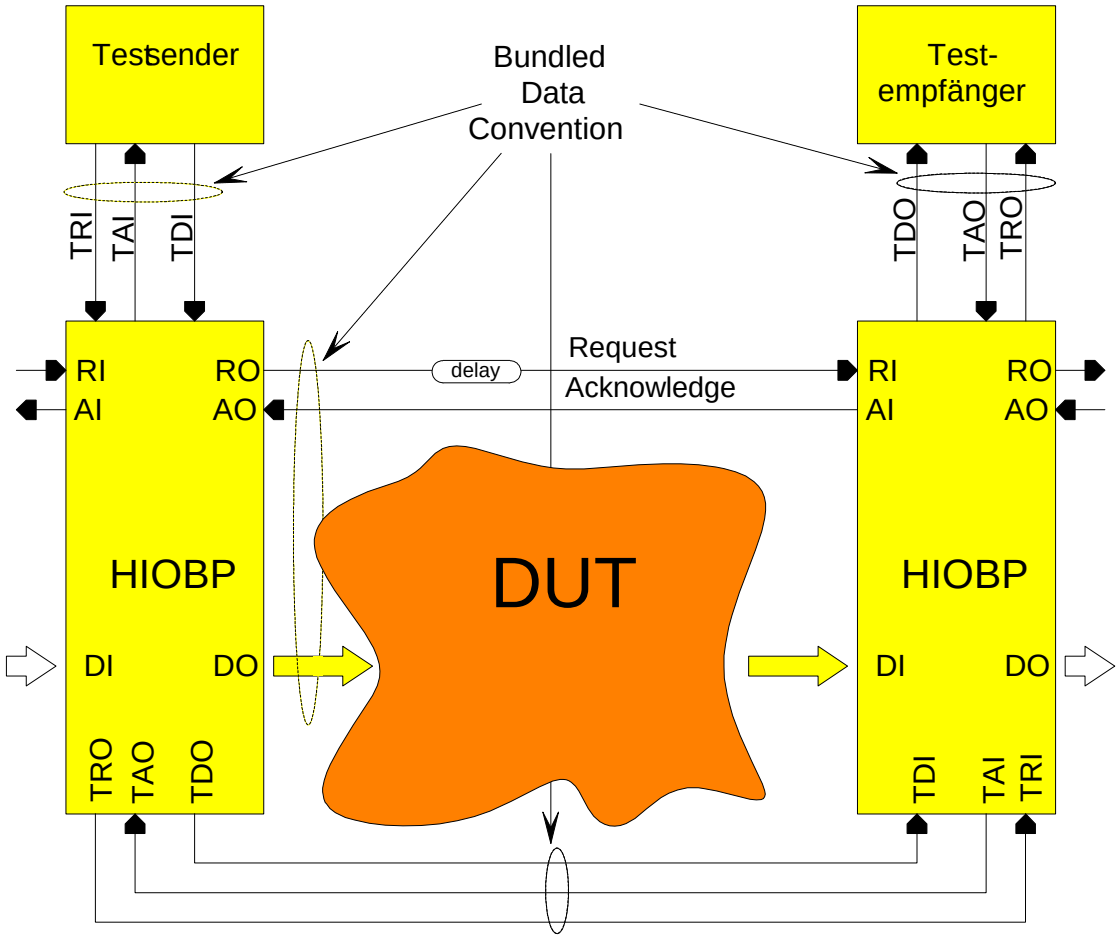

Abb. 89: Eine Konfiguration des passiven HIOB Testkonzepts.

Abb. 90 stellt eine Konfiguration des aktiven Testkonzepts mit *Four-Phase* Teststeuersignalen dar. Das aktive Konzept verwendet für TRI und TAI pegelgesteuerte Signale. Es benötigt eine externe Steuerung zum Umschalten zwischen seriellem Schieberegisterbetrieb und dem Anlegen an den Datenpfad. Der Testablauf beginnt mit dem seriellen Laden der ersten und dem Initialisieren der zweiten HIOBA Stufe. Danach wird in den Testmustergenerierungsbetrieb (TPG) und Testdatenkompressionsbetrieb (TDA) umgeschaltet. In der folgenden Phase werden Testmuster mit jedem positiven Pegelwechsel von TRI generiert. TAI quittiert diesen Prozeß. TRI kann somit als Taktsignal für einen Testautomaten verwendet werden. Es muß dabei gewährleistet sein, dass vor dem nächsten Taktsignal das TAI Signal erfolgt ist. Im letzten Schritt erfolgt die Auswertung der komprimierten Testdaten.

- **Mit der Reaktionszeit von TAI nach TRI erhält man beim aktiven Testkonzept die maximale mögliche Performance für die getestete asynchrone Schaltung.**

Bei *Capture-Pass* Registerstrukturen besteht die Möglichkeit, alle Pipelinestufen transparent zu schalten. Damit können die Pipelinestufen vergleichbar zur LSSD Technik getestet werden. Während die Registerstufen der Pipeline transparent geschaltet sind, kann die Pfadverzögerung über alle Stufen ermittelt werden. Somit können Aussagen zur Charakterisierung des aktuellen Fertigungsprozesses getroffen werden.

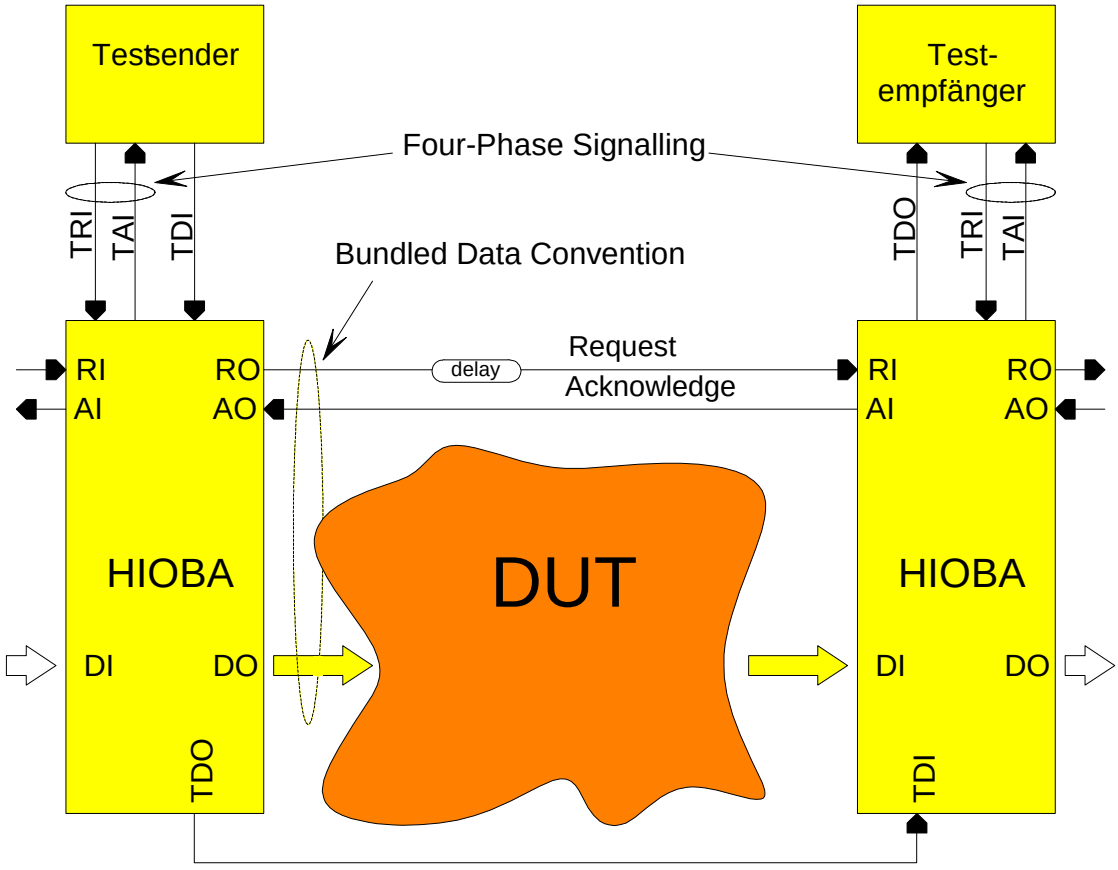

Abb. 90: Eine Konfiguration des aktiven HIOB Testkonzepts.

Die vorgestellte Testmethodik kann auf *Four-Phase* Signale im Datenpfad abgebildet werden. Pegelgesteuerte Signale verringern den Implementierungs-

aufwand im Vergleich zu ereignisgesteuerten Signalen unter Verwendung der CMOS Technik. Bei *Four-Phase* Signalen müssen nicht positive und negative Signalwechsel für einen Vorgang berücksichtigt werden. Die Auswertung der Signale erfolgt über Spannungspegel, was zu einer Vereinfachung der Signalgenerierung des Testablaufs und einer kompakteren Schaltung führt.

Digitale Speicher besitzen durch ihren internen analogen Aufbau ein asynchrones Verhalten an den Interfaces. Für das externe Zeitverhalten ist dieses asynchrone Verhalten in einer synchronen Umgebung jedoch unerwünscht. Asynchrone Interfaces können die Geschwindigkeit der Speicher erhöhen, wenn die Module zur Speicheransteuerung auch mit einem asynchronen Interface ausgerüstet sind. Da Speicher in der Regel *Dual-Rail* Signale intern verwenden, können daraus *Single-Rail* Signale, wie *Request* und *Acknowledge*, gewonnen werden[93].

- **Die hier vorgestellten testfreundlichen Entwurfskonzepte bilden einen Ansatz für den testfreundlichen Entwurf von Speichern - wie z.B. *embedded SRAM*, *DRAM* - in integrierten Schaltungen.**

Der Test asynchroner Interfacebausteine in Schaltungen und Systemen kann mit den vorgeschlagenen asynchronen Testkonzepten verbessert werden. Hierzu sind weitere Untersuchungen notwendig. Auch für die Anwendung des Prinzips der asynchronen Registerzellen auf Scanpfade und rückgekoppelte Schieberegister in synchronen Schaltungen sind zusätzliche Untersuchungen nötig. Zellulare Automaten sind aufgrund ihres Aufbaus in der asynchronen Schaltungstechnik eine interessante Alternative zu den rückgekoppelten Schieberegistern mit einer globalen Rückkopplung: sie besitzen nur Verknüpfungen zu ihren nächsten Nachbarn und entsprechen dadurch eher einer lokalen Steuerung.

Asynchrone Schaltungsmaßnahmen erfahren mit zunehmender Integrationsdichte eine immer größere Bedeutung in der Digitaltechnik. Insbesondere die

[93] Zur Zeit sind jedoch diese asynchronen Steuersignale bei Speichern extern nicht vorgesehen. Ein Test mit einem asynchronen Zeitverhalten könnte dynamische Fehler besser überprüfen und analysieren.

Berechnung einer synchronen Taktverteilung in einer Schaltung gestaltet sich sehr aufwendig. Die Schaltungen werden dabei in Taktdomänen aufgeteilt, in denen die Speicherelemente lokal synchronisiert werden. Die Übergänge zwischen den Taktdomänen müssen in einem zweiten Schritt behandelt und bei jedem Technologieschritt neu überprüft werden. Zur Reduzierung der Stromaufnahme werden Bereiche des Taktnetzwerks abgeschaltet. Bei der Betrachtung dieser Problematik ist die Verwandtschaft zu den Verfahren der asynchronen Signalverarbeitung zu erkennen. Bei der Verwendung asynchroner Schaltungen können die Fläche, die Entwicklungszeit und der Stromverbrauch des Taktnetzwerks für die Schaltungsfunktion gegeneinander aufgerechnet werden. Die Versorgungsleitungen für VDD und GND können deutlich reduziert werden, da die Register nie zum gleichen Zeitpunkt schalten. Bei der Verwendung von *Muller C-Elementen* können die Generierung und Balancierung eines Taktnetzwerks durch *Gated Clocks* vereinfacht werden. Dies wird mit wenigen zusätzlichen Transistoren erreicht.

Kapitel 6

Zusammenfassung

Diese Arbeit gibt zunächst einen Überblick zum Test asynchroner Schaltungen sowie zur Problematik des Testens von Schaltungsstrukturen mit *Single-Rail* Signalen, die auch als *Self-timed Circuits* bezeichnet werden. Die Stärken und Schwächen der bisherigen Vorschläge zum Test von asynchronen Schaltungen wurden diskutiert. Die bisher vorgestellten Verfahren zum Testen asynchroner Schaltungen benötigen dabei ein s y n c h r o n e s Taktnetzwerk und können das asynchrone Schaltverhalten nicht prüfen. In diesen Schaltungen kann auf eine dem Taktnetzwerk ähnliche Struktur nicht verzichtet werden.

Ausgehend von den gewonnenen Kenntnissen der Untersuchung zum Stand der Technik wurden erstmals Konzepte zum testfreundlichen Entwurf vorgestellt, die auch während der Testdurchführung mit a s y n c h r o n e n Signalen zur Ablaufsteuerung arbeiten. Diese Konzepte beschreiben eine passive scanbasierte Methode und ein aktives asynchrones Testkonzept, welches auf dem BILBO Prinzip basiert. Zur Umsetzung wurde dazu das Signalablaufdiagramm jeder Testbetriebsart beschrieben und der entsprechende *Signal Transition Graph* entwickelt. Die Betriebsarten in beiden Konzepten sind jeweils zu einer gemeinsamen Betriebsart zusammengefaßt und anschließend mit einer Realisierung auf Gatterebene verdeutlicht. Mit Hilfe dieser Konzepte besteht die Möglichkeit, Tests für Schaltungen zu entwickeln, die auch die lokalen asynchronen Synchronisationsprotokolle beim Test berücksichtigen.

Durch die vorgestellten testfreundlichen Registerelemente ist es möglich, einen testfreundlichen Entwurf in integrierten asynchronen und synchronen Schaltungen durchzuführen, ohne die Performance der Schaltung zu reduzieren. Hiermit kann die Akzeptanz des testfreundlichen Entwurfs in der modernen Schaltungsentwicklung stark erhöht werden, da sich der zur Zeit verwendete zusätzliche Multiplexer nicht mehr im Datenpfad befindet.

Als neue Bibliothekselemente wurden testfreundliche *Muller C-Elemente* sowie scanfähige Registerzellen vorgestellt, die speziell auf die asynchrone Schaltungsarchitektur abgestimmt sind. Das testfreundliche *Muller C-Element* ermöglicht dabei die Steuerung und Observation der asynchronen Steuersignale einer *Micropipeline*. Als herausragende Eigenschaft besitzen die asynchronen Scanzellen eine vernachlässigbare zusätzliche Signalverzögerung. Ohne den kritischen Signalpfad zwischen Datenein- und Datenausgang des Registers zu verlängern, ermöglichen sie das Steuern und Beobachten der Testmuster. Auch für die Testmustergenerierung und die Testdatenkompression wurden Registerzellen entwickelt, die, vergleichbar zu den Scanzellen, eine vernachlässigbare zusätzliche Signalverzögerung besitzen. Ausgehend von den Zellen für den asynchronen Datenpfad wurden Scanpfadzellen und BILBO Register vorgeschlagen, welche auch für synchrone Schaltungen eingesetzt werden können, ohne deren Performance zu reduzieren.

Ein weiteres Ergebnis ist die Implementierung der neuen Schaltungsfunktionen als Bibliothekselemente, die den testfreundlichen Entwurf asynchroner Schaltungen unterstützen. Die neuen Bibliothekszellen für den testfreundlichen Entwurf teilen sich in Elemente für die Steuerungslogik und für den Datenpfad auf. Sie vervollständigen somit asynchrone Schaltungsbibliotheken. Ein Konzept zur hierarchischen Fehlermodellierung einer asynchronen Schaltungsbibliothek wurde vorgestellt und beispielhaft angewendet. Hier zeigt sich, dass nichtklassische Fehlermodelle einen erheblichen Anteil der notwendigen Modellierungen auf Gatterebene bilden. Die Betrachtung von Haftfehlern an den Ein- und Ausgängen der Module reicht nicht zur Modellierung der Fehlerursachen aus, die auf Transistorebene angenommen werden können.

Der passive asynchrone Scanpfad berücksichtigt die *Two-Phase Bundled Data Convention* im parallelen Daten- sowie im seriellen Testpfad. Dadurch wird über eine lokale Steuerung die Umschaltung zwischen Schieberegisterbetrieb und

parallelem Datenbetrieb ohne eine Änderung der externen Steuersignale möglich. Mit Hilfe der testfreundlichen *Muller C-Elemente* und eines asynchronen Scanregisters ermöglicht der Testablauf erstmals den Test des Datenpfads, des Steuerpfads und der *Bundled Data Convention* auf Haftfehler gleichzeitig.

Das zweite vorgestellte Testkonzept beinhaltet neben dem Schieberegisterbetrieb die Testmustergenerierung und die Testdatenkompression. Es können somit erstmals Testmuster auf dem Chip mit asynchronen Signalen generiert und komprimiert werden. Auch hierbei erfolgen die Testmustergenerierung und die Testdatenkompression mit der *Bundled Data Convention*. Es muß lediglich die Anzahl der generierten oder komprimierten Testmuster gezählt werden, um den Test an der gewünschten Stelle zu stoppen. Der Scanbetrieb des zweiten Konzepts verwendet im Gegensatz zu den anderen Betriebsarten einen synchronen pegelgesteuerten Schieberegisterbetrieb. Hierdurch wird die Ansteuerung durch Testautomaten oder externe Testlogik erheblich erleichtert. Ein weiterer Vorteil ist in dem reduzierten Flächenbedarf zum zuvor beschriebenen Scanpfadkonzept zu sehen. Die Teststeuersignale verwenden hierbei durchgängig *Four-Phase* Signale. Der Datenpfad arbeitet weiterhin mit *Two-Phase* Signalen, wie dies bei *Micropipelines* vorgesehen ist. Bei Verwendung des zweiten Konzepts können zusätzlich zu den Haftfehlern und Funktionskonvertierungen auch dynamische Fehler erkannt werden, da die Testmustergenerierung und -kompression auf dem Chip erfolgen.

Diese Untersuchung bildet somit eine Basis zum Test asynchroner Schaltungen. Sie behandelt im Kern den testfreundlichen Entwurf von Schaltungen mit *Two-Phase* Signalen. Schaltungen mit *Four-Phase* Signalen bieten sich als attraktive Alternative zu *Micropipelines* an, da die Register weniger Fläche benötigen. Die vorgestellten Testkonzepte können daran angepaßt werden, da das Signalverhalten durch die Pegelsteuerung stark vereinfacht wird. Beide Testkonzepte eignen sich auch in hohem Maße zum testfreundlichen Entwurf von Mixed-Signal Bausteinen. Weitere Untersuchungen sind im Bereich Simulations- und Testmustergenerierungstechniken für asynchrone Schaltungen notwendig, um die Qualität von Testmustern bestimmen und generieren zu können.

Literatur

Das Literaturverzeichnis teilt sich in drei Bereiche auf. Die Differenzierung in unterschiedliche Teilgebiete wurde vorgenommen, da sich diese Untersuchung mit drei verschiedenen Forschungsgebieten beschäftigt.

Der erste Teil enthält die Grundlagen zum testfreundlichen Entwurf und zum Testen digitaler synchroner Schaltungen, einschließlich der Untersuchungen zur Fehlermodellierung. Literaturstellen sind mit dem Index **[Literaturstelle]**$_{L.1}$ gekennzeichnet.

Der zweite Teil enthält den Entwurf, die Synthese und Analyse asynchroner Schaltungen. Sie erhalten den Index **[Literaturstelle]**$_{L.2}$.

Im dritten Bereich sind Veröffentlichungen verzeichnet, die den Test asynchroner Schaltungen untersuchen. Diese werden mit dem Index **[Literaturstelle]**$_{L.3}$ versehen.

Darüber hinaus haben sich im Internet Foren etabliert, die sich mit der Forschung und Entwicklung asynchroner Schaltungen beschäftigen. Hier sind einige Adressen aufgezählt.

[PHARAO]$_{L}$ mailto: asynchronous-private@pharaos.cs.columbia.edu

In einem Emailforum werden Themen zur asynchronen Schaltungsentwicklung und deren Analyse diskutiert. Insbesondere werden Workshops und Konferenzen angekündigt und Kontakte aufgebaut. Geleitet wird das Forum von Steven Novick. Die Anmeldung erfolgt über die Email-Adresse: mailto:asynchronous-request@pharaos.cs.columbia.edu.

[BIBTEX]$_L$ http://www.win.tue.nl/cs/pa/wsinap/async.html

Eine umfangreiche Literatursammlung zum Thema asynchrone Schaltungsentwicklung liegt in dem Datenbankformat von TEX vor. Es können, neben der Eingabe neuer Datensätze, Literaturstellen gesucht und die komplette Datenbank abgerufen werden. Sie ist unter dem Namen BIBTEX zu finden.

[EDIS]$_L$ http://edis.win.tue.nl/bibl.html

Eine "Encyclopedia of *Delay-insensitive* Systems" mit dem Namen EDIS ist dort ebenfalls verzeichnet.

[AMULET]$_L$ http://www.cs.man.ac.uk/amulet

Der Amulet Server enthält Veröffentlichungen zum Themenbereich Entwurf und Test von *Micropipelines*. Den Schwerpunkt bilden hierbei Veröffentlichungen, die sich mit dem Entwurf und der Analyse asynchroner Mikroprozessoren befassen. Auch ein Konzept zur formalen Beschreibung von asynchronen Schaltungen für einen synthesefähigen Schaltungsentwurf wird hier vorgestellt.

[TOB]$_L$ http://www.tet.uni-hannover.de/tob/tob.htm

Eine thematisch sortierte Literatursammlung zum Testen von integrierten Schaltungen des Instituts für Theoretische Elektrotechnik liegt im HTML-Format vor. Sie umfaßt neben der asynchronen Schaltungstechnik die Themenbereiche zum Test digitaler und analoger Schaltungen.

L.1 Test digitaler Schaltungen

[59Eldr] Richard D. Eldred: "Test Routines Based on Symbolic Logical Statements", Journal ACM, Vol. 6, No. 1, Jan 59, pp. 33-36.,

[73Breu] Breuer Melvin A.: "Testing for Intermittent Faults in Digital Circuits", IEEE Transactions on Computers, Vol. C-22, No. 3, pp. 241-246, March 1973.

[73WillA] M. J. Y. Williams, J. B. Angell: "Enhancing Testability of Large-Scale Integrated Circuits via Test Points and Additional Logic", IEEE Transactions on Computers, Vol. C-22, No. 1, pp. 46-60, January 1973.

[74Mei] Mei K.C.Y.: "Bridging and Stuck-At Faults", IEEE Transactions on Computers, VOL. C-23, NO.7, pp. 720-727, July 1974.

[75ShedM] J. J. Shedletsky, E. J. McCluskey: "The Error Latency of a Fault in a Combinational Digital Circuit". 1975.

[76Muth] P. Muth: "A Nine-Valued Circuit Model for Test Generation", IEEE Transactions on Computers, Vol. C-25, No. 6, pp. 630-636, June 1976.

[77EichW] E. Eichelberger: "A logic design structure for LSI testability", 14th Design Automation Conference, pp. 462-468, June 1977.

[77Seth] S. C. Seth: "Data Compression Techniques in Logic Testing: An Extension of Transition Counts", Journal of design automation & fault tolerant computing,77.

[77Will] T. Williams: "Random Patterns within a Structured Sequential Logic Design", Semiconductor Test Symposium, pp.19-27, 1977.

[77WillE] T. Williams, E. Eichelberger: "Random Patterns Within a Structured Sequential Logic Design", IEEE Semiconductor Test Symposium, Cherry Hill, PA, Digest of Papers, 19-27, 1977.

[79KoenMZ] B. Könemann, J. Mucha, G. Zwiehoff: "Built-in Logic Block Observation Techniques", Test Conference, Cherry Hill, PA, Digest of Papers, 37-41, 1979.

[80GaliCV] Galiay J., Crouzet Y., Vergniault M.: "Physical Versus Logical Fault Models MOS LSI Circuits: Impact on Their Testability", IEEE Transactions on Computers, Vol. C-29, No. 6, pp. 527-53, June 1980.

[82MalaS] Malaiya, Y.K. Su, S.Y.H.: "A New Fault Model and Testing Technique for CMOS Devices", IEEE Test Conference, pp. 25-34, November 1982.

[83BarzR] Z. Barzilai, B. Rosen: "Comparison of AC Selt-Testing Procedures", International Test Conference, ITC 1983, pp. 89-94, October 1983.

[83Elzi] Y. M. El-zig: "Classifying, Testing, and Eliminating VLSI MOS Failures", 1983.

[83RichF] B. Richards, P. Footner: "Failure Analysis in Semiconductor Devices - Rationale, Methodology and Practice", GEC Journal of Research, vol. 1, no. 2, 1983.

[84BardM] P.Bardell, W.McAnney: "Parallel pseudorandom sequences for built-in test", International Test Conference, pp. 302-308, October 1984.

[84HughM] Hughes J., McCluskey E.: "An Analysis of the Multiple Fault Detection Capabilities of Single Stuck-At Fault Test Sets", 1984 International Test Conference, pp. 52-58, October 1984.

[84Star] C. W. Starke: "Built-in Test for CMOS circuits", International Test Conference, Philadelphia, PA, Digest of Papers, 309-314, 1984.

[85Haye] Hayes, J.P.: "Fault Modeling", IEEE Design & Test of Computers, pp. 88-95, April 1985.

[86Abra] J. Abrabaham: "Fault Modeling in VLSI", VLSI Testing, Elsevier Science Publisher B.V., 1986.

[86AbraF] J. Abrabaham, W. Fuchs: "Fault and Error Models for VLSI", Proceedings of the IEEE, Vol. 74, No. 5, pp. 639-654, May 1986.

[86Koep] S. Koeppe: "Modeling and Simulation of Delay Faults in CMOS Logic Circuits", International Test Conference, ITC, pp. 530-536, 1986.

[86ShihAa] Shih Hsi-Ching, Abraham Jacob A.: "Fault Collapsing Techniques for MOS VLSI Circuits", IEEE Proc. International Symposium on Fault Tolerant Computing, Vienna, Austria, pp. 370-375, July 86.

[86WaicL] Waicukauski J. A., Lindbloom E.: "Transition Fault Simulation by Parrallel Pattern Single Fault Propagation", International Test Conference, pp. 542-549, 1986.

[86WalkD] Walker Hank, Director Stephen W.: "VLASIC: A Catastrophic Fault Yield Simulator for Integrated Circuits", IEEE Transactions on Computer-Aided Design, Vol. CAD-5, No.4, October 1986.

[87Koepa] Koeppe S.: "Layout Methods to Reduce CMOS Stuck-Open-Faults and Enhance Test- ability", IEEE International Solid-State Circuits Conference,, pp. 228-229 1987.

[87Koepb] Koeppe Siegmar: "Optimal Layout to Avoid CMOS Stuck-Open Faults", 24th ACM/IEEE Design Automation Conference, pp.829-835, 1987.

[87Magn] Magnhagen B.: "Fast Fault Analysis, a new Approach", Fifth European Workshop on Design for Testability, June 1987.

[87Maly] Maly Wojciech: "Realistic Fault Modelling for VLSI Testing", 24th Design Automation Conference, pp. 173-180, June 1987.

[88FergS] Ferguson F.J., Shen J.P.: "Extraction and Simulation of Realistic CMOS Faults using Inductive Fault Analsis", 1988 International Test Conference, pp.475-484.

[88IyenRW] Vijay S. Iyengar, Barry K. Rosen, and John A.Waicukauski: "A Quantitative Transition Fault Model", IBM Research Report, 1988.

[88MillM] Millman, S.D. McCluskey, E.J.: "Detecting Bridging Faults with Stuck-at Test Sets", International Test Conference, pp. 773-783, September 1988.

[88WillDG] T.Williams, W.Daehn, M.Gruetzner, C.Starke: "Bounds and Analysis of Aliasing Errors in Linear Feedback Shift Registers", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. CAD-7, No. 1, pp. 75-83, January 1988.

[90AbraBF] Miran Abramovici, Melvin A. Breuer, Arthur D. Friedman: "Digital Systems Testing and Testable Design", IEEE Press, 1990.

[90Bard]P. Bardell: "Design Considerations on Parallel Pseudorandom Pattern Generators", Jetta, 73-87, 1990.

[90ButlM] Butler, K.M. Mercer, M.R.: "The Influences of Fault Type and Topology on Fault Model Performance and the Implications to Test and Testable Design", 27th Design Automation Conference, vol. 27, pp. 673-678, June 1990.

[90JhaT] Jha Niraj K., Tong Qiao: "Detection of Multiple Input Bridging and Stuck-on Faults in CMOS Logic Circuits using Current Monitoring", EDAC, pp. 350-354, 1990.

[90McEu] S.McEuen: "Why IDDQ?", International Test Conference, pp. 252-253, September 1990.

[90McGeRS] P. C. McGeer, R. K. Brayton, R. Rudell, A. L. Sangiovanni-Vincentelli: "Extended Stuck-Fault Testability for Combinational Networks", Advanced Reasearch in VLSI, 1990.

[90NighM] P. Nigh, W. Maly: "Test Generation for Current Testing", IEEE Design & Test of Computers, pp. 26-38, February 1990.

[90PancRM] Pancholy Ashish, Rajski Janusz, McNaughton Larry J.: "Empirical Failure Analysis and Validation of Fault Models in CMOS VLSI", International Test Conference, pp. 938-947, September 1990.

[90StorM] T. M. Storey, W. Maly: "CMOS Bridging Fault Detection", International Test Conference, pp. 842-851, September 1990.

[91ChamRS] Champac, V.H. Rodriguez-Montanes, R. Segura, J.A. Rubio, J.A.: "Fault Modelling of Gate Oxide Short, Floating Gate and Bridging Failures in CMOS Circuits", ETC91, European Test Conference 1991, pp. 143-148, 1991.

[92MahlHA] Mahlstedt, U., Heinitz, M., Alt, J.: "Test Generation for IDDQ Testing and Leakage Fault Detection in CMOS Circuits", European Design Automation Conference, pp. 486-491, September 1992.

[93AltM] J. Alt, U. Mahlstedt: "Simulation of non-classical Fault on the Gate Level Fault Modeling", 11th IEEE VLSI Test Symposium, Atlantic City, New Jersey, USA, Digest of Papers, 351-354, 6. - 8.4.1993, 1993.

[93Heinb] Heinitz,Matthias: "Generation of Inherent Periodical Test Patterns", Institut für Theoretische Elektrotechnik, Universität Hannover, Appelst. 9A, 30167 Hannover, Germany, 1993.

[93LamBS] Lam, W.K.C., Brayton, R.K., Sangiovanni-Vincentelli, A.L.: "Circuit Delay Models and Their Exact Computation Using Timed Boolean Functions", 30th Design Automation Conference, pp. 128-134, June 1993.

[93XueDJ] Xue, H., Di, C., Jess, J.A.G.: "A Net-Oriented Method for Realistic Fault Analysis", International Conference on Computer-Aided Design, pp. 78-83, November 1993.

[94CalhSG] M.Calha, M.Santos, F.Goncalves, I.Teixeira, J.Teixeira: "Back annotation of physical defects int gate-level, realistic faults in digital ICs", International Test Conference, pp. 720-728, October 1994.

[94ChakT] Chakravarty, S., Thadikaran, P.: "A Study of IDDQ Subset Selection Algorithms for Bridging Faults", International Test Conference, pp. 403-412, October 1994.

[94HawkSR] C.F.Hawkins, J.M.Soden, A.W.Righter, F.J.Ferguson: "Defect Classes-An overdue paradigm for CMOS IC testing", International Test Conference, pp. 413-425, October 1994.

[94HeinG] M.Heinitz, T.Grünning: "Generation of Inherently Periodical Test Patterns for the E-Beam Test of Scan-based Designs", Microelectronic Engineering, Vol. 24, pp. 279-286, March 1994.

[94RenoHB] M. Renovell, P. Huc, Y. Bertrand: "A Unified Model for Inter-Gate and Intra-Gate CMOS Bridging Fault: The Configuration Ratio", 3rd Asian Test Symposium, pp. 170-175, November 1994.

[95Aitk] R. C. Aitken: "Finding Defects with Fault Models", International Test Conference, pp. 498-505, October 1995.

[95Alt] Alt, Jürgen: "Fehlersimulation synchroner Schaltungen unter Berücksichtigung nicht-klassischer Fehler", Fakultät für Maschinenwesen, Universität Hannover, Germany.1995.

[95LuT] D. Lu, C. Tong: "High Level Fault Modeling of Asynchronous Circuits", 13th VLSI Test Symposium, pp. 190-195, April 1995.

[95Mahl] Mahlstedt, Udo: "Deterministische Testgenerierung für Gattenverzögerungsfehler unter Berücksichtigung der minimal erkennbaren Fehlergröße", Fakultät für Maschinenwesen, Universität Hannover, Germany, 1995.

[95Maxwa] Maxwell, P.C.: "The Use Of IDDQ Testing in Low Stuck-at Coverage Situations", 13th VLSI Test Symposium, pp. 84-88, April 1995.

[95RenoHBa] Renovell, M., Huc, P., Bertrand, Y.: "The Concept of Resistance Interval: A New Parametric Model for Realistic Resistive Bridging Fault", 13th VLSI Test Symposium, pp. 184-189, April 1995.

[95Wein] Weiner, S.: "A Fault Model and a Test Method for Analog Fuzzy Logic Circuits", International Test Conference, pp. 282-291, October 1995.

[96FavaDO] M. Favalli, M. Dalpasso, P. Olivio: "Modeling and Simulation of Broken Connections in CMOS IC`s", Transactions on Computer-Aided Design of Integrated Circuits & Systems (004-1391), IEEE, Transaction, 7, 1996.

[96KharMT] J. Khare, W. Maly, N. Tiday: "Fault Characterization of Standard Cell Libraries Using Inductive Contamination Analysis (ICA)", 14th IEEE VLSI Test Symposium, Princeton, New Jersey, USA, Proceedings, 405-413, 28.04-01.05, 1996.

[96Powe] T. J. Powell: "Consistently Dominant Fault Model for Tristate Buffer Nets", 14th IEEE VLSI Test Symposium, Digest of Papers, 400-404, 1996.

[96PowePC] Theo J. Powell, James R. Pair, Bernard G. Carbajal III: "CORRELATING DEFECTS TO FUNCTIONAL AND I DDQ TESTS", International Test Conference, Washington, DC, Digest of Papers, 501-510, 20.-25.10., 1996.

L.2 Entwurf asynchroner Schaltungen

[49Shan] Shannon, Claude E.: "The Synthesis of Two-Terminal Switching Circuits", The Bell System Technical Journal, pp. 59-99, 1949.

[52Quin] Quine, W. V.: "The Problem of Simplifying Truth Functions", American Mathematical Monthly, Vol. 59 (ZA 2291, Z mat Z1), pp. 521-531 Oct. 1952.

[54Huff] D. A. Huffman: "The Synthesis of Sequential Switching Circuits", The Journal of the Franklin Institute (ZS 4155), Vol. 257, No. 3, pp. 161-190, pp. 275-304, March 1954.

[55Meal] G. H. Mealy: "A Method for Synthesizing Sequential Circuits", The Bell System Technical Journal, pp. 1045-1079, September 1955.

[56Moor] E. F. Moore: "Gedanken-Experiments on Sequential Machines", Automata Studies, Annals of Mathmatics Studies, Vol. 34 Princeton University Press, Princeton, New York, pp. 129-153, 1956.

[57Huff] D. A. Huffmann: "The Design and Use of Hazard-Free Switching Networks", Journal of the Association for Computing Machinery 4, Vol. 47, pp. 47-62, 1957.

[59Mull] D. E. Muller: "Treatment of Transition Signals in Electronic Switching Circuits by Algrebraic Methods", IRE Transactions on Electronic Computers Ec-8, pp.401, ZA 3430, 1959.

[59Unge] S. H. Unger: "Hazards and Delays in Asynchronous Sequential Switching Circuits", IRE Transaction on Circuit Theory, Vol. CT-6, pp. 12-25, March 1959.

[62AmaB] Amarel S., Brzozowski J. A.: "Theoretical Considerations on Reliability Properties of Recursive Triangular Switching Networks", Redundancy Techniques for Computing Systems, Wilcox R.H. Mann W.C., Spartan Books 1962, pp. 70-128, s1962.

[62Petr] C.A. Petri: "Kommunikation mit Automaten", Institut für instrumentale Mathematik, Bonn, 1962.

[62Tryo] Tryon J. G.: "Quadded Logic", Redundancy Techniques for Computing Systems, Wilcox R.H. Mann W.C., Spartan Books, pp. 205-228, 1962.

[64YoelR] M. Yoeli, S. Rinon: "Application of Ternary Algebra to the Study of Static Hazards", Journal of the Association for Computing Machinery, Vol. 11, No. 1, January 1964.

[65Eich] E. B. Eichelberger: "Hazard Detection in Combinational and Sequential Switching Circuits", IBM Journal of Research & Development, Journal, Vol. 9, No. 2, pp. 90-99, March 1965.

[65Mill] R. Miller: "Switching Theory, Volume 2: Sequential Circuits and Machines", John Wiley, New York, volume 2, 1965.

[74Traj] Tarjan R.: "Finding Dominators in Directed Graphs", SIAM J. Comp., vol.3, no.1, pp.62-89, march 1974.

[76GareJ] M. Garey, D. Johnson: "Some simplified NP-Complete Graph Problems", Theoretical Computer Science, North Holland, 237-267, 1976.

[78Koha] Z. Kohavi: "Switching and Finite Automata Theory", McGraw-Hill, 1978.

[78Lamp] L. Lamport: "Time, Clocks, and the Ordering of Events in a Distributed System", Communication of the ACM, Volume 21, No. 7, July 1978.

[79LengT] Lengauer T., Tarjan R.E.: "A Fast Algorithm for Finding Dominators in a Flowgraph", ACM Transactions on Programming Languages and Systems, vol.1, no.1, pp.121-141, July 1979.

[79Seit] C. Seitz: "Self-timed VLSI Systems", Caltech Conference on Very Large Scale Integration, California, USA, Proceedings, 345-354, 22.-24.1., 1979.

[80MeadC] Mead, C./Conway, L.: "Introduction to VLSI Systems", Addison-Wesley Publishing Co..

[80Seita] Charles Seitz: "System Timing", In: Introduction to VLSI-Systems, pp. 218-259, 1980.

[80Seitb] C. Seitz: "Ideas about Arbiters", Lambda, 10-14, 1. Quarter, 1980.

[81ChanM] K. Chandy, J. Misra: "Asynchronous Distributed Simulation via a Sequence of Parallel Computation", Communication of the ACM, Vol. 24, No. 11, April 1981.

[81Pete] J. Peterson: "Petri Net Theory and the Modelling of Systems", Prentice Hall, Inc., 1981.

[82Seth] R. Sethi: "Pebble Games for Studying Storage Sharing", Theoretical Computer Science, North Holland Publisching Company, pp. 69-84, 1982.

[83RoseS] A.L. Rosenberg, I.H. Sudborough: "Bandwidth and Pebbling", Computing, Springer Verlag, pp. 115-139, 1983.

[83Snep] J. Snepscheut: "Trace Theory and VLSI Design", University of Eindhoven, 1983.

[84GlasH] L. Glasser, L. Hoyte: "Delay and Power Optimization in VLSI Circuits", 21th ACM-IEEE Design Automation Conference, Proceedings, 529-535, 1984.

[84GolaA] G. Golapalakrishnan, V. Akella: "High-Level Optimization in Compiling Process Descriptions to Asynchronous Circuits", Journal of VLSI Signal Processing, Kluwer Academic Publishers, Boston, Journal, 33-45, 1994.

[85DillC] D. Dill, E. Clarke: "Automatic Verification of Asynchronous Circuits Using Temporal Logic", Chapel Hill Conference on VLSI, 127-142, 1985.

[85Mart] A. Martin: "The Design of a Self-timed Circuit for Distributed Mutual Exclusion", Chapel Hill Conference on VLSI, 245-260, 1985.

[85MolnFR] C. Molnar, T. Fang, F. Rosenberger: "Synthesis of Delay-Insensitive Modules", Chapel Hill Conference on VLSI, 67-86, 1985.

[86Blac] D. Black: "On the existence of the delay-insensitive fair arbiters: Trace theory and its implementations", Distributed computing 1, 1986.

[86DallS] W. Dally, C. Seitz: "The torus routing chip", Distributed computing 1, 1986.

[86Mart] A. Martin: "Compiling communicating processes into delay-insensitive VLSI-circuits", Distributed computing 1, 1986.

[86OrtoPT] G. Orton, L. Peppard, S. Tavares: "A Fast Asynchronous RSA Encryption Chip", Custom Integrated Circuit Conference, 439-442, 1986.

[86Uddi] J. Udding: "A formal model for defining and classifying delay-insensitive circuits and systems", Distributed computing 1, 1986.

[87Chua] T. Chu: "Synthesis of Selt-timed VLSI Circuits from Graph-theoretic Specifications", IEEE International Conference on Computer Design: VLSI in Computers, Proceedings, 220-223, 1987.

[87Chub] T. Chu: "Synthesis of Self-timed VLSI Circuits from Graph-theoretic Specifications", Massachusetts Institute of Technology, June 1987.

[87DallS] W. Dally, P. Song: "Design of a Self-Timed VLSI Multicomputer Communication Controller", IEEE International Conference on Computer Design: VLSI in Computers, Proceedings, 230-234, 1987.

[87HayeTL] N. Hayes, M. Tsay, F. LaRocca, P. Kempsey, S. Padnos, T. Chang, P. Thomas: "Asynchronous Design integrates Adress Translation and Physical Data Cache for the WE-32201 IMDC".

[87Mart] A. J. Martin: "A Synthesis Method for Self-timed VLSI Circuits", IEEE International Conference on Computer Design: VLSI in Computers, Proceedings, 224-229, 1987.

[88BerkRS] K. van Berkel, M. Rem, R. Saejis: "VLSI Programming", IEEE International Conference on Computer Design: VLSI in Computers, Proceedings, 152-156, 1988.

[88BerkS] K. van Berkel, R. Saeijs: "Compilation of Communicating Processes into Delay-Insensitive Circuits", IEEE International Conference on Computer Design: VLSI in Computers, Proceedings, 157-162, 1988.

[88BertC] C. Berthet, E. Cerny: "Synthesis of Speed-Independent Circuits from Algebraic Specification", IEEE International Symposium on Circuits and Systems, 1869-1872, 1988.

[88DattBC] P. Datta, S. Bandyopadhay, A. Choudhury: "A Graph Theoretic Approach for State Assignment od Asynchronous Sequential Machines", In. J. Electronoics, 1067-1075, 1988.

[88KomoTT] S. Komori, H. Takata, T. Tamura, F. Asai, T. Ohno, et. al.: "An Elastic Pipeline Mechanism by Self-Timed Circuits", Journal of Solid State Circuits (004-4101), IEEE, Journal, 111-117, 2, 1988.

[88LauRM] C. Lau, D. Renshaw, J. Mavor: "Data Flow Approach to Self-Timed Logic in VLSI", IEEE International Symposium on Circuits and Systems, 479-482, 1988.

[88Peng] Z. Peng: "Let's Design Asynchronous VLSI Systems", Microprocessing and Miroprogramming 24, North Holland, 1988.

[88RoseMC] F. Rosenberger, C. Molnar, T. Chaney. T. Fang: "Q-Modules: Internally Clocked Delay-Insensitive Modules", Transactions on Computers (016-1161), IEEE, Transaction, 1005-1018, 9, 1988.

[88SaabYH] Saab D.G., Yang A.T., Hajj I.N.: "Delay Modeling and Timing of Bipolar Circuits", Design Automation Conference, pp.288-293, 1988.

[88VarsKK] V. Varshavskiy, M. Kishnevskiy, A. Kondrat'yev, M. Rosenblyum, R. Taubin: "Models for Specification and Analysis of Processes in Asynchronous Circuits", Soviet Journal of computer and System Science, 1988.

[88ZhouPK] Zhou D., Preparata F.P., Kang S.M.: "Interconnection Delay In Very High-Speed VLSI", ICCD, pp. 52-55, 1988.

[89BrunS] E. Brunvand, R. Sproull: "Translating Concurrent Programs into Delay-Insensitive Circuits", IEEE/ACM International Conference on CAD-89, Santa Clara, California, Digest of technical papers, 262-265, 1989.

[89BrzoS] J. Brzozowski, C. Seger: "A Unified Framework for Race Analysis of Asynchronous Network", Journal of the Association for Comuting Machinery, pp. 20-45, January, 1989.

[89DillNS] D. Dill, S. Novick, R. Sproull: "Automatic Verification of Speed-independent Circuits with petri Net Specification", IEEE International Conference on Computer Design: VLSI in Computers & Processors, Proceedings, 212-216, 1989.

[89Mart] A. Martin: "Quick Estimation of Transient Currents in CMOS Integrated Circuits", Journal of Solid State Circuits (004-4101), IEEE, Journal, 520-531, 4, 1989.

[89MartBL] A. Martin, S. Burns, T. Lee, D. Borkovic, P. Hazewindus: "The First Asynchronous Microprocessor: The Test Results", Computer Architecture News, April 1989.

[89MartBLa] A. Martin, S. Burns, T. Lee, D. Borkovic, P. Hazewindus: "The Design of an Asynchronous Microprocessor", Proceedings of the dec. Caltech Conference, Proceedings, 351-373, 1989.

[89NowiD] S. Nowick, D. Dill: "Practicality of State-Machine Verification of Speed-Independent Circuits", IEEE/ACM International Conference on CAD-89, Santa Clara, California, Digest of technical papers, 266-269, 1989.

[89PateS] V. Patel, K. Steptoe: "Evaluation of Self-Timed Systems for VLSI", Electronic Letters, Journal, 215-218, 3, 1989.

[89Subr] P. Subrahmanyam: "Automated Synthesis of systems with Interacting Asynchronous (Self-timed) and Synchronous Components", IEEE International Conference on Computer Design: VLSI in Computers & Processors, Proceedings, 334-337, 1989.

[89Suth] I. Sutherland: "Micropipelines", Communications of the ACM, vol. 32, No. 6, June 1989.

[89WhitM] S. Whitakter, G. Maki: "Pass-Transistor Asynchronous Sequential Circuits", Journal of Solid State Circuits (004-4101), IEEE, Journal, 71-78, 2, 1989.

[89WongMF] D. Wong, G. De Micheli, M. Flynn: "Inserting Active Delay Elements to Achieve Wave Pipelining", IEEE/ACM International Conference on CAD-89, Santa Clara, California, Digest of technical papers, 270-273, 1989.

[90Baum]B. Baumgarten: "Petri-Netze, Grundlagen und Anwendungen", ISBN: 3-411-14291-X Kapitel 12,1990.

[90Brow] G. Brown: "Towards Truly Delay-Insensitive Circuit Realizations of Process Algebras", Designing Correct Circuits, 1990.

[90ChiaK] W. Chia, Y. Kuo: "Synthesis of Self-Timed Circuits without State Assignment", Conf. on Computer and Commu. Systems, 528-531, September 1990.

[90GopaM] S. Gopalakrishnan, G. Maki: "VLSI Asynchronous Sequential Circuit Design", IEEE International Conference on Computer Design: VLSI in Computers & Processors, Cambridge, Massachusetts, USA, Proceedings, 238-242, 17.-19.9., 1990.

[90HungM] A. Hung, T. Meng: "Asynchronous Self-timed Circuit Synthesis with Timing Constraints", IEEE International Symposium on Circuits and Systems, 1126-1130, 1990.

[90JacoB] G. Jacobs, R. Brodersen: "A Fully Asynchronous Digital Signal Processor using Self-Timed Circuits", Journal of Solid State Circuits (004-4101), IEEE, Journal, 1526-1537, 12, 1990.

[90LamL] P. Lam, H. Li: "Hierachical design of delay-insensitive systems", IEE Proceedings, vol. 137, Pt. E, no. 1, January 1990.

[90VanbCG] P. Vanbergen, F. Catthoor, G. Goossens, H. De Man: "Optimized Synthesis of Asynchronous Control Circuits from Graph-theoretic Specification", IEEE/ACM International Conference on CAD-90, Santa Clara, California, Digest of technical papers, 184-187, 1990.

[91BerkKR] K. van Berkel, J. Kessels, M. Roncken, R. Saeijs, F. Schalij: "The VLSI-programming language Tangram and its translation into handshake circuits", Europaen Conference on Design Automation, Amsterdam, Netherlands, Proceedings, 384-389, 25.-28.2., 1991.

[91ChoA] K. Cho, K. Asada: "VLSI Oriented Design Method of Asynchronous Sequential Circuits Bsaed on One-hot State Code and Two-transistor AND Logic", IEEE International Symposium on Circuits and Systems, 1793-1796, 1991.

[91DeanWD] M. Dean, T. Williams, D. Dill: "Self-Timing with Level-Encoded 2-Phase Dual-Rail (LEDR)", Conference on Advanced Research in VLSI, Santa Cruz, 55-70, 1991.

[91GeigMK] M. Geiger, T. Müller-Wipperfürth, J. Kepler: "FSM Decomposition Revisited: Algebraic Structure Theory Applied to MCNC Benchmark FSMs", 28th ACM-IEEE Design Automation Conference, Proceedings, 182-184, 1991.

[91LaddB] M. Ladd, W. Birmingham: "Synthesis of Multiple-Input Change Asynchronous Finite State Machines", 28th ACM-IEEE Design Automation Conference, Proceedings, 309-314, 1991.

[91LavaKS] L. Lavangno, K. Keutzer, A. Sangiovanni-Vincentelli: "Algorithms for synthesis of hazard-free asynchronous circuits", 28th ACM-IEEE Design Automation Conference, Proceedings, 302-308, 1991.

[91LinL] K. Lin, C. Lin: "Automatic Synthesis of Asynchronous Circuits", 28th ACM-IEEE Design Automation Conference, Proceedings, 296-301, 1991.

[91MengBM] T. Meng, R. Brodersen, D. Messerschmitt: "Asynchronous Design for Programmable Digital Signal Processors", Transactions on Signal Processing (001-1001), IEEE, Transaction, 939-952, 4, 1991.

[91MoghE] A. Moghaddamzadeh, M. Elmasry: "AGAM: Automatic Generation of Asynchronous Machines", IEEE International Symposium on Circuits and Systems, 3154-3157, 1991.

[91MoonSB] C. Moon, P. Stephan, R. Brayton: "Synthesis of Hazard-free Asynchronous Circuits from Graphical Specifications", IEEE/ACM International Conference on CAD-91, Santa Clara, California, Digest of technical papers, 322-325, 1991.

[91NoviD] S. Novick, D. Dill: "Synthesis of Asynchronous State Machines Using a Local Clock", IEEE International Conference on Computer Design: VLSI in Computers & Processors, Cambridge, Massachusetts, USA, Proceedings, 192-197, 14.-16.10., 1991.

[91NowiD] S. Nowick, D. Dill: "Automatic Synthesis of Locally-Clocked Asynchronous State Machines", IEEE/ACM International Conference on CAD-91, Santa Clara, California, Digest of technical papers, 318-321, 1991.

[91Sege] C. Seger: "On the existence of speed-independent circuits", Theoretical Computer Sience 86 (Elsevier), 1991.

[92AfghS] M. Afghahi, C. Svensson: "Performance of Synchronous and Asynchronous Schemes for VLSI Systems", Transactions on Computers (016-1161), IEEE, Transaction, 858-872, 7, 1992.

[92Berk] K. van Berkel: "Beware the isochronic fork", Integration, the VLSI Journal, Elsevier Sience Publisher B.V., Journal, 103-128, 1992.

[92BrunMS] E. Brunvand, N. Michell, K. Smith: "A Comparison of Self-Timed Design using FPGA, CMOS and GaAs Technologies", IEEE International Conference on Computer Design: VLSI in Computers & Processors, Cambridge, Massachusetts, USA, Proceedings, 76-80, 1992.

[92BrzoE] J. Brzozowski, J. Ebergen: "On the Delay-Sensitivity of Gate Networks", Transactions on Computers (016-1161), IEEE, Transaction, 1349-1360, 11, 1992.

[92Burc] J. Burch: "Delay Models for Verifying Speed-Dependent Asynchronous Circuits", IEEE International Conference on Computer Design: VLSI in Computers & Processors, Cambridge, Massachusetts, USA, Proceedings, 270-274, 1992.

[92CortB] J. Cortadella, R. Badia: "An Asynchronous Architecture Model for Behavioral Synthesis", Europaen Conference on Design Automation, Brüssel, Proceedings, 307-311, 10.-16.3., 1992.

[92DaviGY] I. David, R. Ginosar, M. Yoeli: "Implementing Sequential Machines as Self-Timed Circuits", Transactions on Computers (016-1161), IEEE, Transaction, 43070, 1, 1992.

[92DaviGYa] I. David, R. Ginosar, M. Yoeli: "An Efficient Implementation of Boolean Functions as Self-Timed Circuits", Transactions on Computers (016-1161), IEEE, Transaction, 34275, 1, 1992.

[92EberP] J. Ebergen, M. Peeters: "Modulo-N Counters: Design and Analysis of Delay-Insensitive Circuits", Designing correct Circuits, Elsevier Sciences Publishers B.V. 1992.

[92KessBB] J. Kessels, K. Barkel, R. Burgess, M. Roncken, F. Schalij: "An error decoderfor the compact disc player as an example of VLSI Programming", Europaen Conference on Design Automation, Brüssel, Proceedings, 69-74, 10.-16.3., 1992.

[92KishKT] M. Kishinevsky, A. Kondrateev, A. Taubin, V. Varshavsky: "Analysis and Identification of self-timed circuits", Designing correct Circuits, Elsevier Sciences Publishers B.V. 1992.

[92LeonB] Y. Leong, W. Birmingham: "The Automatic Generation of Bus-Interface Models", 29th ACM-IEEE Design Automation Conference, Anaheim, Californien, Proceedings, 634-637, 8-12.6., 1992.

[92LiebG] A. Liebchen, G. Gopalaksrishnan: "Dynamic Recordering of High Latency Transactions Using a Modified Micropipeline", IEEE International Conference on Computer Design: VLSI in Computers & Processors, Cambridge, Massachusetts, USA, Proceedings, 336-340, 1992.

[92LinL] K. Lin, C. Lin: "A Realization Algorithm of Asynchronous Circuit from STG", Europaen Conference on Design Automation, Brüssel, Proceedings, 322-326, 10.-16.3., 1992.

[92McAu] A. McAuley: "Four State Asynchronous Architectures", Transactions on Computers (016-1161), IEEE, Transaction, 129-142, 2, 1992.

[92Mehr] R. Mehra: "Micropipelined Cache Design Strategies for an Asynchronous Microprocessor", Thesis, University of Manchester, Oktober 1992.

[92MyerM] C. Myers, T. Meng: "Synthesis of Timed Asynchronus Circuits", IEEE International Conference on Computer Design: VLSI in Computers & Processors, Cambridge, Massachusetts, USA, Proceedings, 279-284, 1992.

[92NowiYD] S. Nowick, K. Yun, D. Dill: "Practical Asynchronous Controller Design", IEEE International Conference on Computer Design: VLSI in Computers & Processors, Cambridge, Massachusetts, USA, Proceedings, 341-345, 1992.

[92PaveDF] N. Paver, P. Day, S. Furber, J. Garside, J. Woods: "Register Locking in an Asynchronous Microprocessor", IEEE International Conference on Computer Design: VLSI in Computers & Processors, Cambridge, Massachusetts, USA, Proceedings, 351-355, 1992.

[92SentSL] E. Sentovich, K. Singh, L. Lavango, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. Stephan, R. Brayton, A. Sangiovanni-Vincentelli: "SIS: A System for Sequential Circuit Synthesis", University of California, Berkeley, CA 94720, May 1992.

[92VanbG] P. Vanbekbergen, Goossens: "Specification and Analysis of Timing Constraints in Signal Transistion Graphs", Europaen Conference on Design Automation, Brüssel, Proceedings, 302-306, 10.-16.3., 1992.

[92Yako] A. Yakolev: "A Structural Technique for Fault-Protection in Asynchronous Interfaces", The 22. Annual International Symposium on Fault-Tolerant Computing, Boston, Massachusetts, USA, Digest of Papers, 288-295, 8. - 10. 7., 1992.

[92YunDN] K. Yun, D. Dill, S. Nowick: "Synthesis of 3D Asynchronous State Machines", IEEE International Conference on Computer Design: VLSI in Computers & Processors, Cambridge, Massachusetts, USA, Proceedings, 346-350, 1992.

[92YuS] M. Yu, P. Subrahmanyam: "A Path-Oriented Approach for Reducing Hazards in Asynchronous Design", 29th ACM-IEEE Design Automation Conference, Anaheim, Californien, Proceedings, 239-244, 8-12.6., 1992.

[92YuSa] M. Yu, P. Subrahmanyam: "A New Approach for Checking the Unique State Coding Property of Signal Transition Graphs", Europaen Conference on Design Automation, Brüssel, Proceedings, 312-321, 10.-16.3., 1992.

[93BadiC] R. Badi, J. Cortadella: "High-Level Synthesis of Asynchronous Systems: Scheduling and Process Synchronization", Europaen Conference on Design Automation, Paris, France, Proceedings, 70-74, 22.-25.2., 1993.

[93BeerBM] P. Beerel, J. Burch, T. Meng: "Efficient Verification of the Determinate Speed-Independent Circuits", IEEE/ACM International Conference on CAD-93, Santa Clara, California, Digest of technical papers, 261-267, 7.-11.11., 1993.

[93BelhSY] H. Belhadj, G. Saucier, M. Yoeli: "Asynchronous VLSI Circuits Synthesis: State of the Art", Europaen Conference on Design Automation, Paris, France, Proceedings, 89-96, 22.-25.2., 1993.

[93DaviGY] I. David, R. Ginosar, M. Yoeli: "Self-Timed Architecture of a Reduced Instruction Set Computer", IFIP-Workshop, Manchester, 1993.

[93Ende] P. Endekott: "Processor Architectures for Power Efficiency and Asynchronous Implementation", Thesis, Manchester of University, 1993.

[93Furb] S. Furber: "The Return of Asynchronous Logic". **[Amulet]**$_L$, 1993.

[93FurbDG] S. Furber, P. Day, J. Garside, N. Paver, J. Woods: "A Micropipelined ARM", Int. Conference on VLSI, Grenoble, FR, 1993.

[93Gars] J. Garside: "A CMOS VLSI Implementation of an Asynchronous ALU", IFIP-Workshop, Manchester, 1993.

[93LavaS] L. Lavango, A. Sangiovanni-Vicentelli: "Automated synthesis of asynchronous interface circuits", IFIP- Workshop, Manchester, 1993.

[93LavaSa] Lavagno, L.; Sangiovanni-Vincentelli, A.: "Algorithms for Synthesis and Testing of Asynchronous Circuits", Kluwer Academic Publishers, Boston, Dortrecht, London, 1993.

[93Poun] D. Pountain: "Computing Without Clocks", Byte, 145-150, Jan. 1993.

[93SaloK] O. Salomon, H. Klar: "Self-Timed Fully Pipelined Multiplier", IFIP Workshop, Manchester, 1993.

[93SparNN] J. Sparso, C. Nielsen, L. Nielsen, J. Staunstrup: "Design of Self-timed Multipliers: A Comparison", IFIP-Workshop, Machester, 1993.

[93WuuV] T. Wuu, S. Vrudhula: "A Design of a Fast and Area Efficient Multi-Input Muller C-element", Transaction on Very Large Scale Integration (VLSI) Systems, IEEE, Transaction, 215-219, 2, 1994.

[93YkmaLG] C. Ykman-Couvreur, B. Lin, G. Goossens, H. De Man: "Synthesis and optimization of asynchronous controllers based on extended lock graph theory", Europaen Conference on Design Automation, Paris, France, Proceedings, 512-517, 22.-25.2., 1993.

[93YunD] K. Yun, D. Dill: "Unifying Synchronous/Asynchronous State Machine Synthesis", IEEE/ACM International Conference on CAD-93, Santa Clara, California, Digest of technical papers, 255-260, 7.-11.11., 1993.

[93YunDN] K. Yun, D. Dill, S. Nowick: "Practical Generalizations of Asynchronous State Machines", Europaen Conference on Design Automation, Paris, France, Proceedings, 525-530, 22.-25.2., 1993.

[94AshkEF] A. Ashkinazy, D. Edwards, C. Farnsworth, G. Gendel, S. Sikand: "Tools for Validating Asynchronous Digital Circuits", International Symposium on Advanced Research in Asynchronous Circuits and Systems, Salt Lake City, Utah, USA, 12-21, November, 3-5, 1994.

[94BerkBK] K. Berkel, R. Burgess, J. Kessels, M. Roncken, F. Schalij: "Asynchronous Circuits for Low Power: A DCC Error Corrector", Design & Test of Computers (016-3111), IEEE, Magazine, 22-32, 6, 1994.

[94Brun] E. Brunvand: "Designing Self-Timed Systems Using Concurrent Programs", Journal of VLSI Signal Processing, Kluwer Academic Publishers, Boston, Journal, 47-59, 7, 1994.

[94BrzoS] Brzozowski, J.A., Seger, C-H. H: "Asynchronous Circuits", J. Springer Verlag, Berlin, Heidelberg, New York, Tokyo, 1994.

[94DayFGW] P. Day, S. Furber, J. Garside, N. Paver, J. Woods: "Delay Matching in an Asynchronous Microprocessor", 20. European Solid State Circuits Conference, Ulm, Germany, Proceedings, 1994.

[94DeanDH] M. Dean, D. Dill, M. Horowitz: "Self-Timed Using Current-Sensing Completion Detection (CSCD)", Journal of VLSI Signal Processing, Kluwer Academic Publishers, Boston, Journal, 7-16, 7, 1994.

[94FarnES] C. Farnsworth, D. Edwards, S. Sikand: "Utilising Dynamic Logic for Low Power Consumption in Asynchronous Circuits", International Symposium on Advanced Research in Asynchronous Circuits and Systems, Salt Lake City, Utah, USA, ?, November, 3-5, 1994.

[94FurbDG] S. Furber, P. Day, J. Garside, N. Paver, J. Woods: "AMULET1: A Micropipelined ARM", COMPCON, Fan Fransico, US, March 1994.

[94FurbDGa] S. Furber, P. Day, J. Garside, N. Paver, S. Temple, D. Woods: "The Design and the Evalutation of an Asynchronous Microprocessor", IEEE International Conference on Computer Design: VLSI in Computers & Processors, Cambridge, Massachusetts, USA, Proceedings, 217-220, 1994.

[94GopaKB] G. Gopalakrishnan, P. Kudva, E. Brunvand: "Peephole Optimization of Asynchronous Macromodule Networks", IEEE International Conference on Computer Design: VLSI in Computers & Processors, Cambridge, Massachusetts, USA, Proceedings, 442-446,

[94HaucBB] S. Hauck, S. Burns, G. Borriello, C. Ebeling: "An FPGA for Implementing Asynchronous Circuits", 1994.

[94Heer] C. Heer: "Entwurf selbstgetakteter Schaltungen mit Hilfe einer Standardzellenbibliothek für Sea-of Gates Architekturen", ITG-Fachbericht 127, VDE-Fachtagung, 21.-23.3.94.

[94KishKT] M. Kishinevsky, A. Kondratyev, A. Taubin: "Specification and Analysis od Self-Timed Circuits", Journal of VLSI Signal Processing, Kluwer Academic Publishers, Boston, Journal, 117-135, 7, 1994.

[94KishKTa] M. Kishinevsky, A. Kondratyev, A. Taubin, V. Varshavsky: "Concurrent Hardware", John Wiley & Sons, Inc., NY, USA, 1994.

[94KudvGB] P. Kudva, G. Gopalakrishnan, E. Brunvand, V. Akella: "Performance Analysis and Optimization of Asynchronous Circuits", IEEE International Conference on Computer Design: VLSI in Computers & Processors, Cambridge, Massachusetts, USA, Proceedings, 221-224, 1994.

[94LeunL] S. Leung, H. Li: "A Syntax-Directed Translation for the Synthesis of Delay-Insensitive Circuits", Transaction on Very Large Scale Integration (VLSI) Systems, IEEE, Transaction, 196-210, 6, 1994.

[94LinYV] B. Lin, C. Ykman-Couvreur, P. Vanbekbergen: "A General Stat Graph Transformation Framework for Asynchronous Synthesis", European Design Automation Conference, Grenoble, France, Proceedings, 448-453, 19.-23.09, 1994.

[94MarcB] R. Marc, E. Bachar: "A low power,100Mz 12x18+30-b multiplier-accumulator operating in asynchronous and synchronus modes", 20. Europaen Solid State Circuits Conference, Ulm, Germany, Proceedings, 1994.

[94MarsCS] A. Marshall, B. Coates, P. Siegel: "Designing an Asynchronus Communications Chip", Design & Test of Computers (016-3111), IEEE, Magazine, 8-21, 6, 1994.

[94NanyUK] T.Nanya, Y.Ueno, H.Kagotani, A.Takamura: "TITAC: Design of a Quasi-Delay-Insensitive Microprocessor", Design & Test of Computers (016-3111), IEEE, Magazine, 50-63, 6, 1994.

[94Niel] C. Nielsen: "Evaluation of Function Blocks for Asynchronous Design", European Design Automation Conference, Grenoble, France, Proceedings, 454-459, 19.-23.09, 1994.

[94NowiC] S. Nowick, B. Coates: "UCLOCK: Automated Design of High-Performance Unclocked State Machines", IEEE International Conference on Computer Design: VLSI in Computers & Processors, Cambridge, Massachusetts, USA, Proceedings, 434-441, 1994.

[94PatrF] P. Patra, D. Fussell: "Efficient Building Blocks for Delay Insensitive Circuits", International Symposium on Advanced Research in Asynchronous Circuits and Systems, Salt Lake City, Utah, USA, 196-205, November, 3-5, 1994.

[94Pave] N. Paver: "The Design and Implementation of an Asynchronous Microprocessor", Thesis, University of Manchester, 1994.

[94PaveDF] N. Paver, P. Day, J. Garside, J. Woods: "On the Potential for Complex Deadlock in an Asynchronous Micrcoprocessor", International Symposium on Advanced Research in Asynchronous Circuits and Systems, Salt Lake City, Utah, USA, November, 3-5, 1994.

[94PuriG] R. Puri, J. Gu: "A Modular Partitioning Approach for Asynchronous Circuit Synthesis", 30th ACM-IEEE Design Automation Conference, Proceedings, 63-69, 1994.

[94Roin] P. Røine: "Building Fast Bundled Data Circuits with a Specialized Standard Cell Library", International Symposium on Advanced Research in Asynchronous Circuits and Systems, Salt Lake City, Utah, USA, 134-143, November, 3-5, 1994.

[94SproSM] R. Sproull, I. Sutherland, C. Molnar: "The Counterflow Pipeline Processor Architecture", Design & Test of Computers (016-3111), IEEE, Magazine, 48-59, 9, 1994.

[94SproSMa] R. Sproull, I. Sutherland, C. Molnar: "The Counterflow Pipeline Processor Architecture", Sun Microsystems Laboratories, Inc., SMLI TR-94-25, Mountain View, CA, 1994.

[94TierMB] J.A.Tier, A.J.Martin, D.Brokovic, T.K.Lee: "A 100-MIPS GaAs Asynchronus Microprocessor", Design & Test of Computers (016-3111), IEEE, Magazine, 43-49, 6, 1994.

[94VanbGL] P. Vanbekbergen, G. Goossens, B. Lin: "Modeling and Synthesis of Timed Asynchronous Circuits", European Design Automation Conference, 460-465, 1994.

[95D&T] Roundtable: "Low-Power Design", Design & Test of Computers (016-3111), IEEE, Magazine, 84-90, 12, 1995.

[95Dayw] P. Day, J. Woods: "Investigation into Micropipeline Latch Design Styles", Transaction on Very Large Scale Integration (VLSI) Systems, IEEE, Transaction, 1995.

[95EdwaF] D. Edwords, C. Farnsworth: "EXACT: Exploitation of Asynchronous Circuit Technologies", Technical Report, University of Manchester, Nov. 1995.

[95FarnEL] C. Farnsworth, D. Edwards, J. Liu, S. Sikand: "A Hybrid Asynchronous System Design Environment", 2nd Workshop on Asynchronous Design Methodologies, South Bank University, London, 91-98, 1995.

[95FurbD] S. Furber, P. Day: "Four-Phase Micropipeline Latch Control Circuits", Manchester University, 1995.

[95Gree] Mark R. Greenstreet: "Implementing a STARI Chip", IEEE International Conference on Computer Design: VLSI in Computers & Processors, Proceedings, 38-42, 1995.

[95Hauc] S. Hauck: "Asynchronous Design Methodologies: An Overview", Proceedings of the IEEE (500-5011), Vol. 83, No. 1, IEEE, Proceedings, 69-93, 1, 1995.

[95Heer] C. Heer: "Synchronous clocked and self-timed pipeline configurations", Technical Committee on Computer Architecture, IEEE Computer Society, Newsletter, 1995.

[95KinnGG] D. J. Kinniment, J. D. Garside, B. Gao: "A Comparison of Pover Consumption in Some CMOS Adder Circuits", PATMOS'95, Power and Timing Modeling for Performance of Integrated Circuits, Osnabrück, Workshop, 106-118, Oktober, 1995.

[95LeeGS] T.W.S. Lee, M.R. Greenstreet, C.-J. Seger: "Automatic Verification of Asynchronous Circuits", Design & Test of Computers (016-3111), IEEE, Magazine, 24-30, 1, 1995.

[95LeunL] S. Leung, H. Li: "On the Realizability and Synthesis of Delay-Insensitive Behaviors", Tran. on Computer-Aided Design of Integrated Circuits & Systems (004-1391), IEEE, Transaction, 833-848, 7, 1995.

[95MehrG] R. Mehra, J. Garside: "A Cache Line Fill Circuit for a Micropipelined, Asynchronous Microprocessor", Technical Committee on Computer Architecture, IEEE Computer Society, Newsletter, 1995.

[95MetrFR] C. Metra, M. Favalli, B. Riccò: "Glitch Power Dissipation Model", PATMOS'95, Workshop-Proceedings (Power and Timing Modeling for Performance of Integrated Circuits), pp. 175-189, October 1995.

[95RichB] W.F. Richardson, E. Brunvand: "Precise Exception Handling for a Self-Timed Processor", IEEE International Conference on Computer Design: VLSI in Computers & Processors, Proceedings, 32-37, 1995.

[95RogiV] B. Rogina, B. Vojnovic: "Metastability Evaluation Method by Propagation Delay Distribution Measurement", Fourth Asian Test Symposium, Proceedings, 40-44, 1995.

[95Röns] Rönsberg, J.: "Asynchrone Tricks beim Digital-Entwurf", Elektronik, Vol. 15, pp. 98-101, July 1995.

[95StotJA] B. Stott, D. Johnson, V. Akella: "Asynchronous 2-D Discrete Cosine Transform Core Processor", IEEE International Conference on Computer Design: VLSI in Computers & Processors, Proceedings, 380-385, 1995.

[95Theo] G. Theodoropoulos: "Strategies for Modelling and Simulation of Asynchronous Computer Architectures", Thesis, University of Manchester, Sep. 1995.

[95UedaK] Ueda, H., Kinoshita, K.: "Low Power Design and Its Testability", Fourth Asian Test Symposium, Proceedings, 361-366, 1995.

[95Unge] S. Unger: "Hazards, Critical Races, and Metastability", Transactions on Computers (016-1161), IEEE, Vol. 44, No. 6, Transaction, 754-768, 6, 1995.

[95WeyWW] Chin-Long Wey, H. Wang, Cheng-Ping Wang: "A Self-Timed Redundant-Binary Number Converter for Digital Arithmetic Processors", IEEE International Conference on Computer Design: VLSI in Computers & Processors, Proceedings, 386-391, 1995.

[95YakoKL] A.V. Yakovlev, A.M. Koelmans, L. Lavagno: "High-Level Modeling and Design of Asynchronous Interface Logic", Design & Test of Computers (016-3111), IEEE, Magazine, 32-40, 1, 1995.

[95YunD] K.Y. Yun, D.L. Dill: "A High-Performance Asynchronous SCSI Controller", IEEE International Conference on Computer Design: VLSI in Computers & Processors, Proceedings, 44-49, 1995.

[96BeerHW] Beerel, P.A., Hsieh, Ch.-T., Wadekar, S.: "Estimation of Energy Consumption in Speed-Independent Control Circuits", Transactions on Computer-Aided Design of Integrated Circuits & Systems (004-1391), IEEE, Transaction, 672-680, 6, 1996.

[96BerkB] K. van Berkel, A. Bink: "Single-Track Handshake Signalling with Application to Micropipelines and Handshake Circuits", Second International Symposium on Advanced Research in Asynchronous Circuits and Systems, Aizu-Wakamatsu, Fukushima, Japan, Proceedings, 122-133, 18.-21.03., 1996.

[96Burd] Burdiek, Bernhard: "Entwurf von Standardzellen für den testfreundlichen Entwurf asynchroner Schaltungen", Institut für Theoretische Elektrotechnik, Universität Hannover, Appelst. 9A, 30167 Hannover, Germany, 1996.

[96CortKK] J. Cortadella, M. Kishinevsky, A. Kondrateyev: "Complete state encoding based on the thorie of regions", Second International Symposium on Advanced Research in Asynchronous Circuits and Systems, Aizu-Wakamatsu, Fukushima, Japan, Proceedings, 36-47, 18.-21.03., 1996.

[96Fole] C. Foley: "Characterizing Metastability", Second International Symposium on Advanced Research in Asynchronous Circuits and Systems, Aizu-Wakamatsu, Fukushima, Japan, Proceedings, 175-184, 18.-21.03., 1996.

[96FurbL] S. Furber, J. Liu: "Dynamic Logic in Four-Phase Micropipelines", Second International Symposium on Advanced Research in Asynchronous Circuits and Systems, Aizu-Wakamatsu, Fukushima, Japan, Proceedings, 11-16, 18.-21.03., 1996.

[96GarsTM] J. Garside, S. Temple, R. Mehra: "The Amulet2e Cache System", Second International Symposium on Advanced Research in Asynchronous Circuits and Systems, Aizu-Wakamatsu, Fukushima, Japan, Proceedings, 208-217, 18.-21.03., 1996.

[96GrasMK] E. Grass, R. Morling, I. Kale: "Activity-Monitoring Completion-Detection (AMCD): A new Single-Rail approach to achieve self-testing", Second International Symposium on Advanced Research in Asynchronous Circuits and Systems, Aizu-Wakamatsu, Fukushima, Japan, Proceedings, 143-149, 18.-21.03., 1996.

[96Hulg] H. Hulgaard: "Timing Analysis and Verification of Timed Asynchronous Circuits", University of Washington, Thesis, 1996.

[96MaezK] M. Maezawa, I. Kurosawa: "Pulse-Driven Dual Rail Logic Gate Famaly Based on Rapid Single-Flux-Quantum (RSFQ) Devices for Asynchronous Circuits", Second International Symposium on Advanced Research in Asynchronous Circuits and Systems, Aizu-Wakamatsu, Fukushima, Japan, Proceedings, 134-142, 18.-21.03., 1996.

[96Mutz] M. Mutz: "Der Zeitbehaftete (Zustands-, Ereignis-) Überführungsgraph zur kombinatorierten Zeit- und Logikverifikation", 4. GI/ITG Workshop, "Methoden des Entwurfs digitaler Systeme, Kreischa, 101-110, 25.-27.03, 1996.

[96PenaC] M. A. Pena, J. Cortadella: "Combining Process Algebra and Petri Nets for the Specification and Synthesis of Asynchronous Circuits", Second International Symposium on Advanced Research in Asynchronous Circuits and Systems, Aizu-Wakamatsu, Fukushima, Japan, Proceedings, 222-232, 18.-21.03., 1996.

[96RenaHG] M. Renaudin, B.E. Hassan, A. Guyot: "A New Asynchronous Pipeline Scheme: Application to the Design of a Self-Timed Ring Divider", Journal of Solid State Circuits (004-4101), IEEE, Journal, 1001-1013, 7, 1996.

[96RichB] W.F. Richardson, E. Brunvand: "Fred: An Architecture for a Self-Timed Decoupled Computer", Second International Symposium on Advanced Research in Asynchronous Circuits and Systems, Aizu-Wakamatsu, Fukushima, Japan, Proceedings, 60-68, 18.-21.03., 1996.

[96Roen] H.J. Roensberg: "Asynchrone Tricks beim Digital-Entwurf", Elektronik, Journal, 98-101, 15, 1995.

[96SancS] P. Sancheti, S. Sapatnekar: "Optimal Design of Macrocells for Low Power and High Speed", Transactions on Computer-Aided Design of Integrated Circuits & Systems (004-1391), IEEE, Transaction, 1160-1166, 9, 1996.

[96TabrLE] N. Tabrizi, M.J. Liebelt, K. Eshraghian: "Dynamic Hazards and Speed Independent Delay Model", Second International Symposium on Advanced Research in Asynchronous Circuits and Systems, Aizu-Wakamatsu, Fukushima, Japan, Proceedings, 94-103, 18.-21.03., 1996.

[96YanoSR] K. Yano, Y. Sasaki, K. Rikino, K. Seki: "Top-Down Pass-Transistor Logic Design", Journal of Solid State Circuits (004-4101), IEEE, Journal, 792-803, 6, 1996.

[96YoneY] T. Yoneda, T. Yoshikawa: "Using partial orders for trace theoretic verification of asynchronous circuits", Second International Symposium on Advanced Research in Asynchronous Circuits and Systems, Aizu-Wakamatsu, Fukushima, Japan, Proceedings, 152-163, 18.-21.03., 1996.

[96YunBA] K. Yun, P. Beerel, J. Arceo: "High Performance Asynchronous Pipeline Circuits", Second International Symposium on Advanced Research in Asynchronous Circuits and Systems, Aizu-Wakamatsu, Fukushima, Japan, Proceedings, 17-28, 18.-21.03., 1996.

L.3 Test asynchroner Schaltungen

[30Roth] J. Roth: "HIOB", Kiepenheuer und Witsch, Köln, 1930.

[62SeshF] S. Seshu, D. Freeman: "The Diagnosis of Asynchronous Sequntial Switching Systems", IRE Transactions on Electronic Computers, pp. 459-465, August 1962.

[71PutzR] G. Putzolu, J. Roth: "A Heuristic Algorithm for the Testing of Asynchronous Circuits", Transactions on Computers (016-1161), IEEE, Transaction, 639-647, 6, 1971.

[74Chap] S. Chappell: "Automatic Test Generation for Asynchronous Digital Circuits", The Bell System Technical Journal, Vol. 83, No. 8, pp. 1477-1503, October 1974.

[76BreuF] M. Breuer, A. Friedmann: "Diagnosis & Reliable Design of Digital Systems", Pitman, pp. 109-131, 1976.

[84BellV] C. Bellon, R. Velazco: "Taking into account Asynchronous Signals in Functional Test of Complex Circuits", 21th ACM-IEEE Design Automation Conference, Proceedings, 490-496, 1984.

[84HearSD] B. Heard, R. Sheshadri, R. David, A. Sammuli: "Automatic Test Pattern Generation for Asynchronous Networks", 1984.

[84Suss] A. Susskind: "A Technique for Making Asynchronous sequential Circuits Readily Testable", International Test Conference, Philadelphia, PA, Digest of Papers, 842-846, 1984.

[86RanaLC] D. Rana, S. Levitan, D. Carlson: "A Testable, Asynchronous Systolic Array Implementation of an IIR Filter", Custom Integrated Circuit Conference, 90-93, 1986.

[88ChenAK] K. Cheng, V. Agrawal, E. Kuh: "A Sequential Circuit Test Generator Using Threshold-Value Simulation", 18. International Symposium on Fault-Tolerant Computing, Digest of Papers, 24-29, 1988.

[88Li] T. Li: "Design of VLSI asynchronous circuits for testability", Int. J. Electronics, vol.64, no. 6, pp. 859-868, 1988.

[89Cars] G. Carson: "A Fully Testable CMOS Asynchronous Counter", Technical Report # 89-10-07, University of Washington, Seattle, WA, Oct. 1989.

[89LeenSa] J. Leenstra, L. Spaanenburg: "On the design and test of asynchronous macros embedded in synchronous systems", International Test Conference, Washington, DC, Digest of Papers, 838-845, 1989.

[89SakaHO] K. Sakashita, T. Hashizume, T. Ohya, I. Takimoto, S. Kato: "Cell-Based Test Design Method", International Test Conference, Washington, DC, Digest of Papers, 909-916, 1989.

[90AshaGD] P. Ashar, A. Ghosh, S. Devadas, A. Newton: "Implicit State Transition Graphs: Application to Sequential Logic Synthesis and Test", IEEE/ACM International Conference on CAD-90, Santa Clara, California, Digest of technical papers, 84-87, 1990.

[90CarsB] G. Carson, G. Borriello: "A Testable CMOS Asynchronous Counter", IEEE International Solid-State Circuits Conference, Digest of technical papers, 952-960, 1990.

[91BeerM] P. Beerel, T. Meng: "Testability of Asynchronous Timed Control Circuits with Delay Assumptions", 28th ACM-IEEE Design Automation Conference, Proceedings, 446-451, 1991.

[91GuilSY] S. Guillory, D. Saab, A. Yang: "Fault Modeling and Testing of Self-timed Circuits", 9th IEEE VLSI Test Symposium, 62-66, 1991.

[91KeutLS] K. Keutzer, L. Lavagno, A. Sangiovanni-Vincentelli: "Synthesis for Testability Techniques for Asynchronous Circuits", IEEE/ACM International Conference on CAD-91, Santa Clara, California, Digest of technical papers, 326-329, 1991.

[91MartH] A. Marin, P. Hazewindus: "Testing Delay-Insensitive Circuits", Conference on Advanced Research in VLSI, Santa Cruz, 118-132, 1991.

[91ThorRB] P. Thorel, J. Rainard, A. Botta, A. Chemarin, J. Majos: "Implementing Boundary-Scan and Pseudo-Random BIST in an Asynchronous Transfer Mode Switch", International Test Conference, Washington, DC, Digest of Papers, 131-139, 1991.

[92BeerM] P. Beerel, T. Meng: "Semimodularity and Testability of Speed-Independent Circuits", Integration, the VLSI Journal, Elsevier Sience Publisher B.V., Journal, 301-323, 1992.

[92PageSV] S. Pagey, S. Sherlekar, G. Venkatesh: "Issues in Fault Modelling and Testing of Micropipelines", First Asian Test Symposium, Hiroshima, Japan, Proceedings, 107-111, November, 26-27., 1992.

[93LavaKL] L. Lavango, M. Kishinevsky, A. Lioy: "Testing Redundant Asynchronous Circuits", Technical Report ID-TR:1993-124, October 1993.

[93Leen] J. Leenstra: "Hierachical Test Development and DFT for Asynchronous Semi-Custom ASICs", Thesis, Eindhoven, Nederlands, Aprils 1993.

[93Mart] A. Martin: "Tomorrow's Digital Hardware will be Asynchronous and Verified", Caltech-CS-TR-93-26, 1993.

[93RoncS] M. Roncken, R. Saeijs: "Linear Test Times for Delay-Insensitive Circuits", IFIP Workshop, Manchester, 1993.

[93SaloK] O. Salomon, H. Klar: "Standard Cell Design for Testable Self-Timed Systems", 19. Europaen Solid State Circuits Conference, Sevilla, Spain, Proceedings, 22.-24.9., 1993.

[93Tang] Ulf Tangermann: "Eine Untersuchung zum testfreundlichen Entwurf von asynchronen Schaltungen", Universität Hannover, Institut für Theoretische Elektrotechnik, 1993.

[93VierMC] H.T. Vierhaus, W. Meyer, R. Camposano: "Fault Behavior and Testability of Asynchronous CMOS Circuits", Gesellschaft für Mathematik und Datenverarbeitung, GMD, Technical reports, 35034, 1, 1993.

[93WeySF] C. Wey, M. Shieh, P. Fisher: "ASLCScan: A Scan Design Technique for Asynchronous Sequential Logic Circuits", IEEE International Conference on Computer Design: VLSI in Computers & Processors, Cambridge, Massachusetts, USA, Proceedings, 159-162, 1993.

[94Feig] Ch. Feige: "Untersuchung zur Entwicklung eines asynchronen Boundary Scan Test Konzeptes", Universität Hannover, Institut für Theoretische Elektrotechnik, 1994.

[94KhocB] A. Khoche, E. Brunvand: "Testing of Micropipelines", International Symposium on Advanced Research in Asynchronous Circuits and Systems, Salt Lake City, Utah, USA, 239-246, November, 3-5, 1994.

[94LavaKL] L. Lavagno, M. Kishnevsky, A. Lioy: "Testing Redundant Asynchronous Circuits by Variable Phase Shifting", IEEE International Conference on Computer Design: VLSI in Computers & Processors, Cambridge, Massachusetts, USA, Proceedings, 328-333, 1994.

[94Petl] O. Petlin: "Random Testing of Asynchronous VLSI Circuits", Thesis, University of Manchester, 1994.

[94Ronc] M. Roncken: "Partial Scan Test for Asynchronous Circuits Illustrated on a DCC Error Corrector", International Symposium on Advanced Research in Asynchronous Circuits and Systems, Salt Lake City, Utah, USA, 247-256, November, 3-5, 1994.

[95BrzoR] J. Brzozowski, K. Raahemifar: "Testing C-Elements is not Elementary", Second Workshop on Asynchronous Design Methodologies, May 30-31, 1995, London.

[95DaviGY] I. David, R. Ginosar, M. Yoeli: "Self-Timed is Self-Checking", Journal of Electronic Testing Theory and Applications, Kluwer Academic Publishers, Journal, 219-228, 6, 1995.

[95EinsS] K. Einspahr, S. Seth: "A Switch-Level Test Generation System for Synchronous and Asynchronous Circuits", Jetta, Kluwer Academic Publishers, Journal, 59-73, 6, 1995.

[95GlorO] A. Gloria, M. Olivieri: "Efficient Semicustom Micropipeline Design", Transactions on Very Large Scale Integration (VLSI) Systems, IEEE, Transaction, 464-468, 9, 1995.

[95HessOE] S. Hessabi, M. Osman, M. Elamasry: "Differential BiCMOS Logic Circuits: Fault Characterization and Design-for-Testability", Transaction on VLSI Systems, 437-445, September, 1995.

[95HulgBB] H. Hulgaard, S. Burns, G. Borriello: "Testing Asynchronous Circuits: A Survey", Technical Report 94-03-06, University of Washington, Seattle, WA, March 1994.

[95KeutLS] K. Keutzer, L. Lavango, A. Sagiovanni-Vincentelli: "Synthesis for Testability Techniques for Asynchronous Circuits", Transactions on Computer-Aided Design of Integrated Circuits & Systems (004-1391), IEEE, Transaction, 1569-1577, 12, 1995.

[95KhocB] A. Khoche, E. Brunvand: "A Partial Scan Methodology for Testing Self-Timed Circuits", 13th IEEE VLSI Test Symposium, Princeton, New Jersey, USA, Digest of Papers, 283-289, 1. - 3.5., 1995.

[95LuT] D. Lu, C. Tong: "High Level Fault Modelling of Asynchronous Circuits", VLSI, 1995.

[95PageKB] S. Pagey, A. Khoche, E. Brunvand: "DFT and Fast Testing of Self-timed Control Circuits", Fourth Asian Test Symposium, Proceedings, 382-386, 11, 1995.

[95PetlF] O. Petlin, S. Furber: "Power Consumption and Testability of CMOS VLSI Circuits", University of Manchester, 1995.

[95PetlFa] O. Petlin, S. Furber: "Scan Testing of Asynchronous Sequential Circuits", 5'th great Lake Symp. on VLSI, 1995.

[95PetlFb] O. Petlin, S. Furber: "Scan Testing of Micropipelines", 13th IEEE VLSI Test Symposium, Princeton, New Jersey, USA, Digest of Papers, 1. - 3.5., 1995.

[95SchoH] V. Schöber, M. Heinitz: "An Asynchronous Scan Path for Micropipelined Modules", Technical Report, Institut für Theoretische Elektrotechnik, University of Hannover, 1995.

[96Kiel] B. Ohlau: "Entwicklung eines Scanpfades zum Testfreundlichen Entwurf von asynchronen Schaltungen", Institut für Theoretische Elektrotechnik, Universität Hannover, Appelst. 9A, 30167 Hannover, Germany, 1996.

[96Ohla] B. Ohlau: "Entwicklung eines asynchronen Testmustergenerators für den Testfreundlichen Entwurf von asynchronen Schaltungen", Institut für Theoretische Elektrotechnik, Universität Hannover, Appelst. 9A, 30167 Hannover, Germany, 1996.

[96Petl] O Petlin: "Design for Testability of Asynchronous VLSI circuits", University of Manchester, 1996.

[96Rick] Ricken, Dirk: "Eine Untersuchung zur Fehlermodellierung in asynchronen Schaltungen", Institut für Theoretische Elektrotechnik, Universität Hannover, 1996.

[96RoncB] M. Roncken, E. Bruls: "Test Quality of Asynchronous Circuits: A Defect-oriented Evaluation", International Test Conference, Washington, DC, Digest of Papers, 205-214, 20.-25.10., 1996.

[96SchoK] V. Schöber, T. Kiel: "An Asynchronous Scan Path for Micropipelined Modules", Europaen Conference on Design Automation, Paris, France, Proceedings, 28.2 - 3.3., 1996.

[96SchoKa] V. Schöber, T. Kiel: "An Asynchronous Scan Path Concept for Micropipelines using the Bundled Data Convention", International Test Conference, Washington, DC, Digest of Papers, 225-231, 1996.

[97RoigCP] O. Roig, J. Cortadella, M. Pena, E. Pastor: " Automatic Generation of Synchronous Patterns for Asynchronus Circuits", Design Automation Conference, 1997.

Anhang

Externe Steuersignale der aktiven Testhilfen

Bei dem aktiven HIOB Konzept werden über externe Steuersignale die Testbetriebsarten umgeschaltet. In diesem Anhang werden die Umschaltvorgänge beschrieben, die zur Einstellung einer neuen Testbetriebsart notwendig sind.

Abb. 91 zeigt die externe Steuerung des SCAN Betriebs. Die Signale TSM, TSH und TPR werden für die jeweilige Testbetriebsart neu eingestellt. Diese Signalwechsel erfolgen dabei jeweils nachdem die Registersteuerung mit einem TAI+ oder TAI- Signalwechsel einen Vorgang abgeschlossen hat. Sind die Signale gesetzt, erfolgt der nächste TRI+ oder TRI- Signalwechsel. Als Bedingung zum Wechsel zwischen den Zuständen sind Signale, TSM, TPR und TSH, angegeben, die zwischen TAI- und TRI+ wechseln sollen bzw. stabil bleiben. So gibt es neben den stabilen Signalen "0" und "1" die Bedingung für steigende (**R, Rising**) und fallende (**F, Falling**) Flanken. Als letzte Möglichkeit existiert ein D, Do not care, wenn ein externes Steuersignal in beiden Zuständen die Bedingung für den Signalwechsel erfüllt. Der SCAN Betrieb ist eine grundlegende Funktion der

Betriebsarten. Es wird wiederholt auf diese Betriebsart zugegriffen, um Daten in das Register ein- oder auszulesen. Das Anlegen der Daten in den Datenpfad erfolgt über gesonderte Befehle, dessen Erklärung auf den nächsten Seiten zu finden ist.

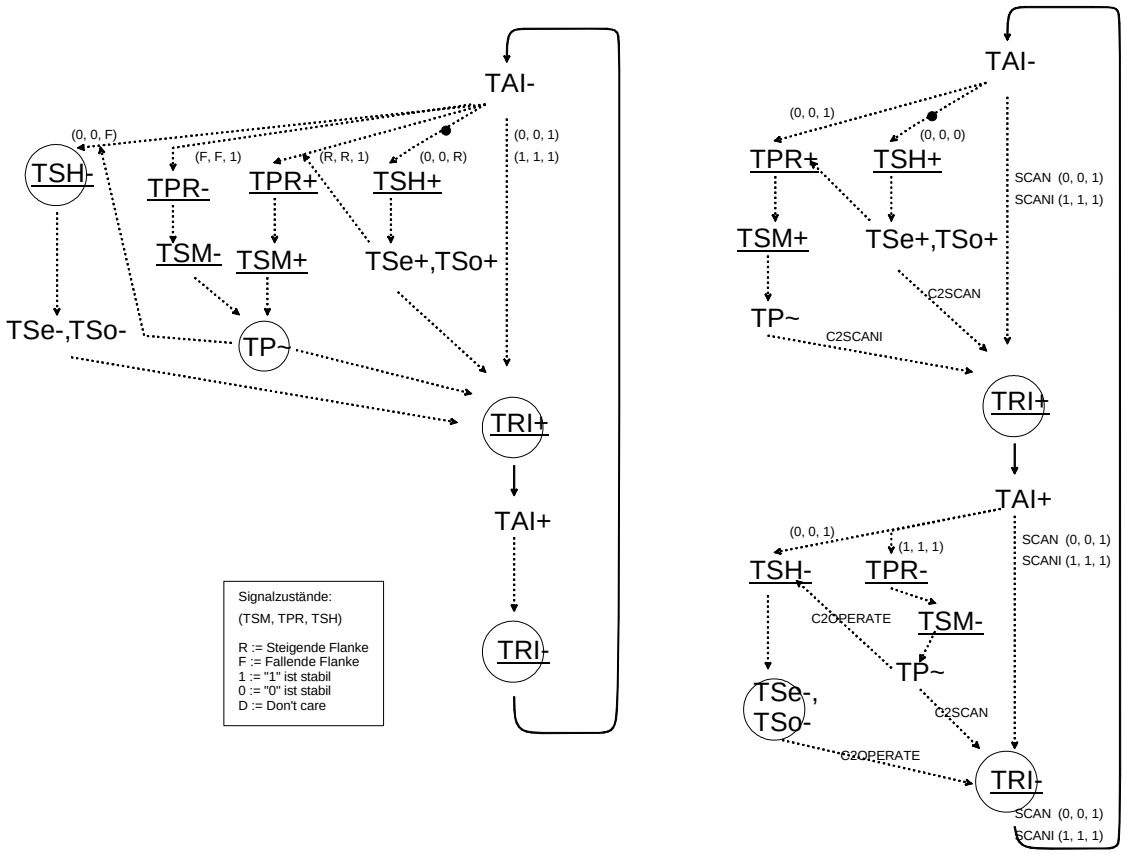

Abb. 91: Die externen Teststeuersignale für den SCAN-Betrieb.

In **Abb. 92** sind die externen Teststeuersignale dargestellt, die die Steuerung des HIOB Registers von außen ermöglichen. Sie erfolgen stets nachdem die interne Testablaufsteuerung mit einem TAI Signalwechsel anzeigt, dass sie die letzte Aktion ausgeführt hat.

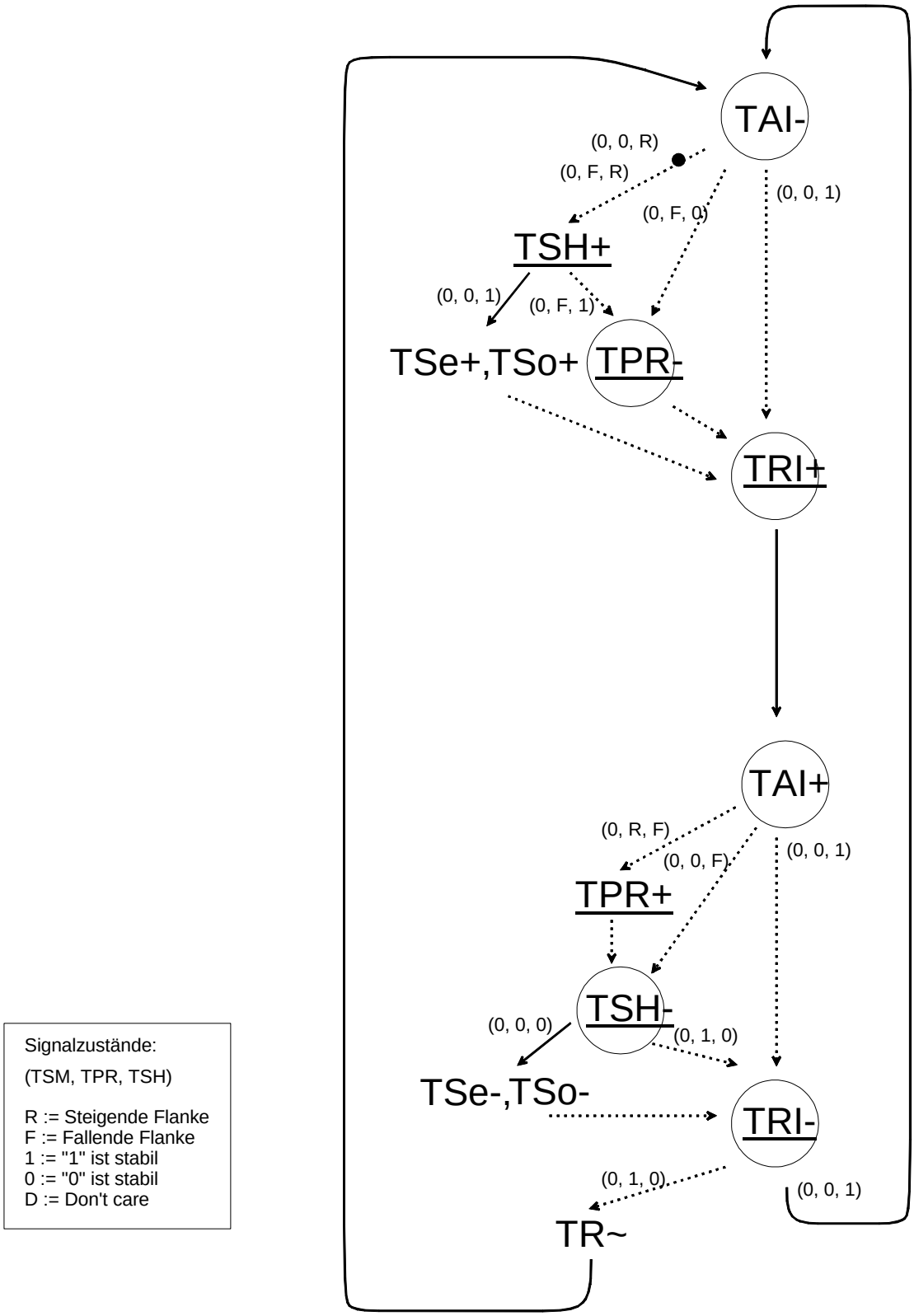

Abb. 92: Die externen Teststeuersignale für den PRELOAD Modus.

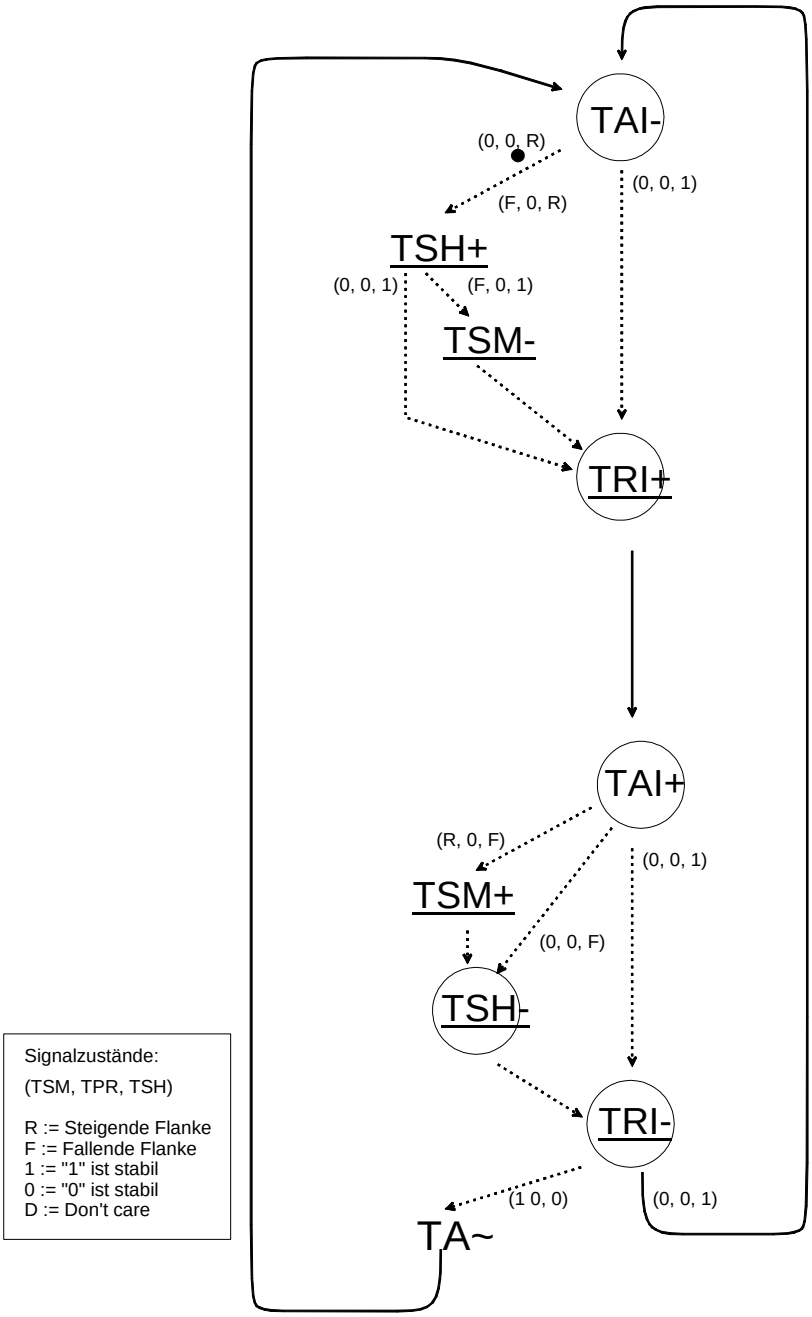

Abb. 93: Die externen Teststeuersignale für den SAMPLE Betrieb.

Abb. 93 zeigt die nötigen externen Signalwechsel zur Initialisierung und Ausführung des SAMPLE und SCAN Betriebs.

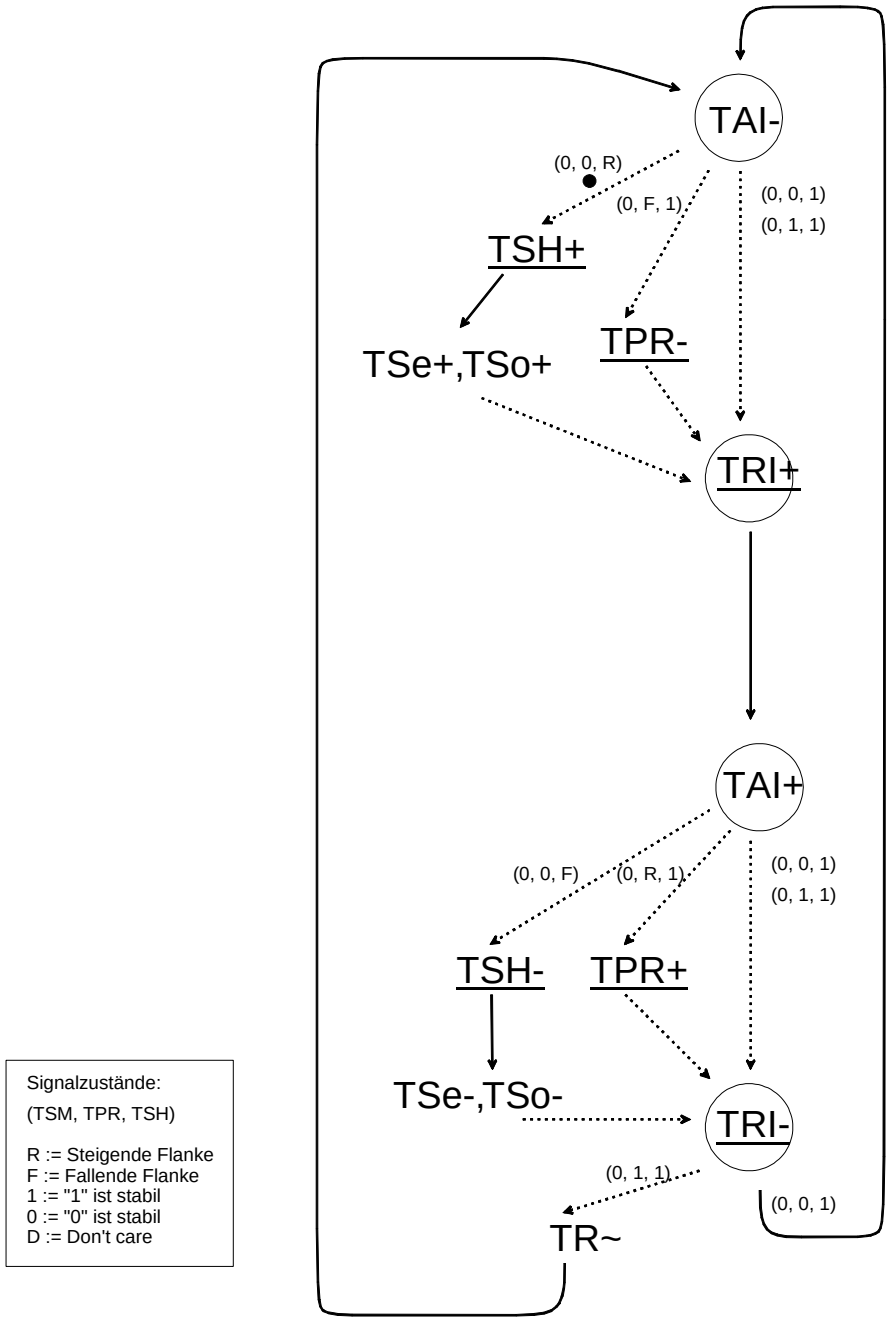

Abb. 94: Die externen Teststeuersignale für den TPG Modus.

Abb. 94 zeigt die externen Steuersignale der Testmustergenerierung. Auch hier ist der Scanbetrieb zu sehen, um ein Datum vor oder nach der Testmustergenerierung ein- oder auszulesen.

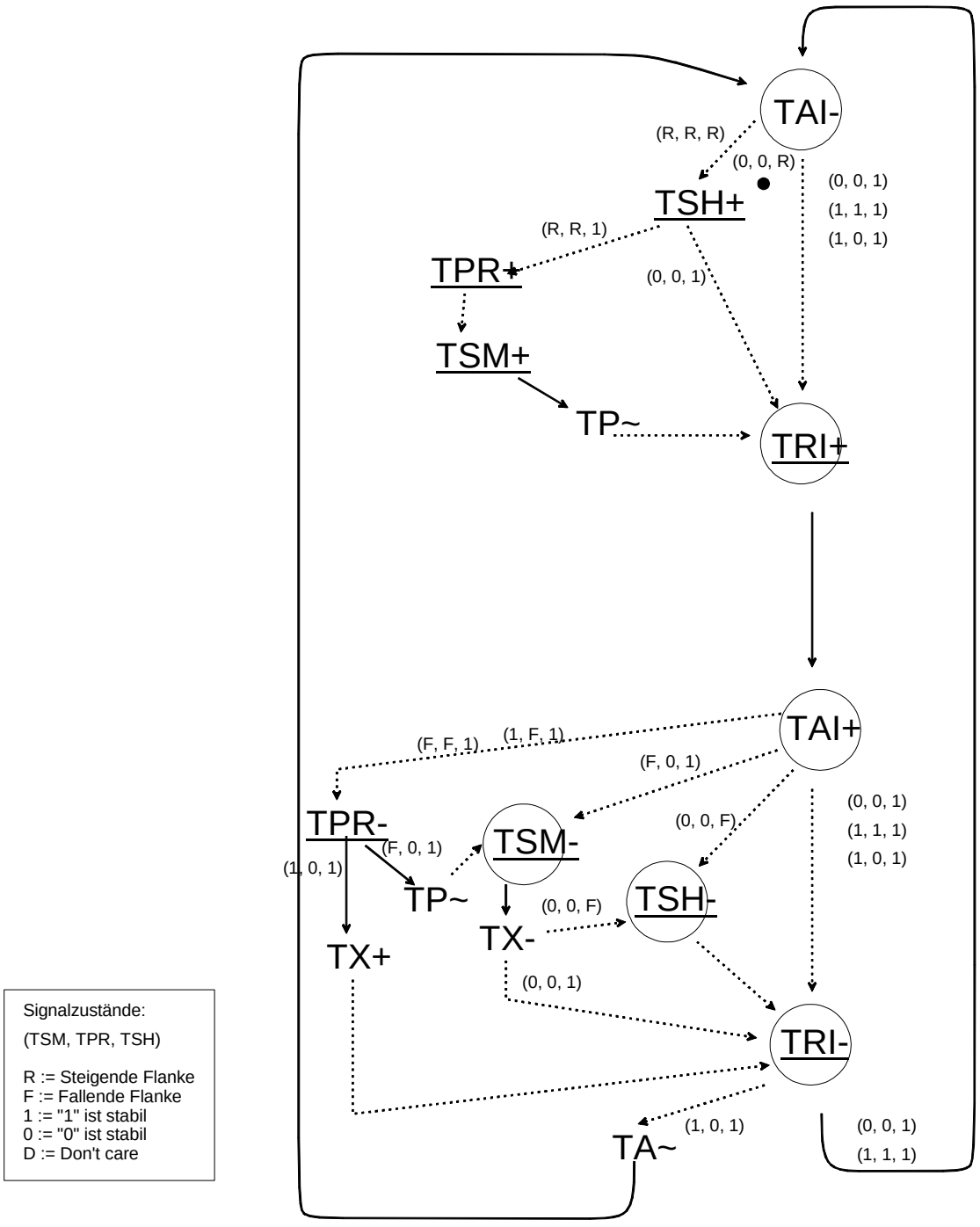

Abb. 95: Die externen Teststeuersignale für den TDA Modus.

Abb. 95 beschreibt für den Testdatenkomprimierungsbetrieb, wie externe Signale geschaltet werden müssen.

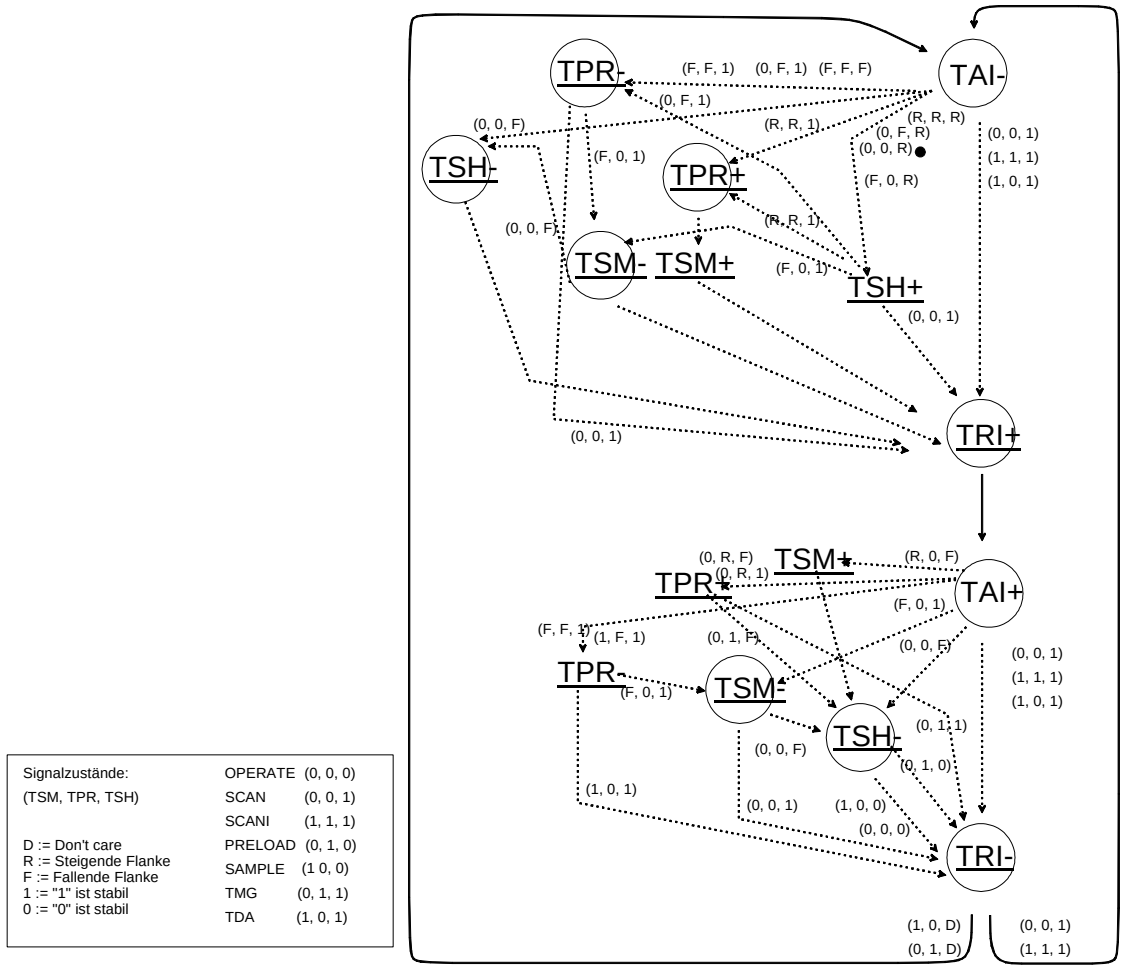

Abb. 96: Die externen Steuersignale der Testablaufsteuerung des aktiven HIOB Konzepts.

In **Abb. 96** sind die externen Steuersignale dargestellt, die für den Wechsel zwischen den Testbetriebsarten nötig sind. Diese Signalwechsel sind wichtig, um das Testregister von einer in die andere Betriebsart zu setzen, ohne einen falschen Folgezustand einzustellen[94]. Es muß dabei gewährleistet sein, dass sich immer nur ein Steuersignal zu einem Zeitpunkt ändert.

[94] Zur Übersichtlichkeit sind einige Signalwechsel zweimal eingezeichnet, da sie zwischen den Signalwechseln TAI- zu TRI+ und den Signalwechseln TAI+ zu TRI- benötigt werden.

Lebenslauf

Volker Schöber

9.1.1963	Geboren in Hannover als Sohn von Herbert und Ursula Schöber
1969-1973	Besuch der Volksschule Marienwerder in Hannover
1973-1982	Besuch des Gymnasium Garbsen
1982	Abschluß des Gymnasiums mit der allgemeinen Hochschulreife
1982-1991	Studium der Elektrotechnik an der Universität Hannover
1991	Abschluß des Studiums mit dem Diplom-Ing. Elektrotechnik
1991-1996	Wissenschaftlicher Mitarbeiter bei Prof. Dr.-Ing. J. Mucha am Institut für Theoretische Elektrotechnik der Universität Hannover
1997-1999	Mitarbeiter des Bereichs Halbleiter der Siemens AG
Seit 1999	Mitarbeiter von Infineon Technologies