TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik – Institut für Technische Informatik – VLSI-Entwurfssysteme, Diagnostik und Entwurf

Test & Diagnose digitaler Systeme, Prüffreundlicher Entwurf.

Norman Seßler

Dresden, 1.7.2009

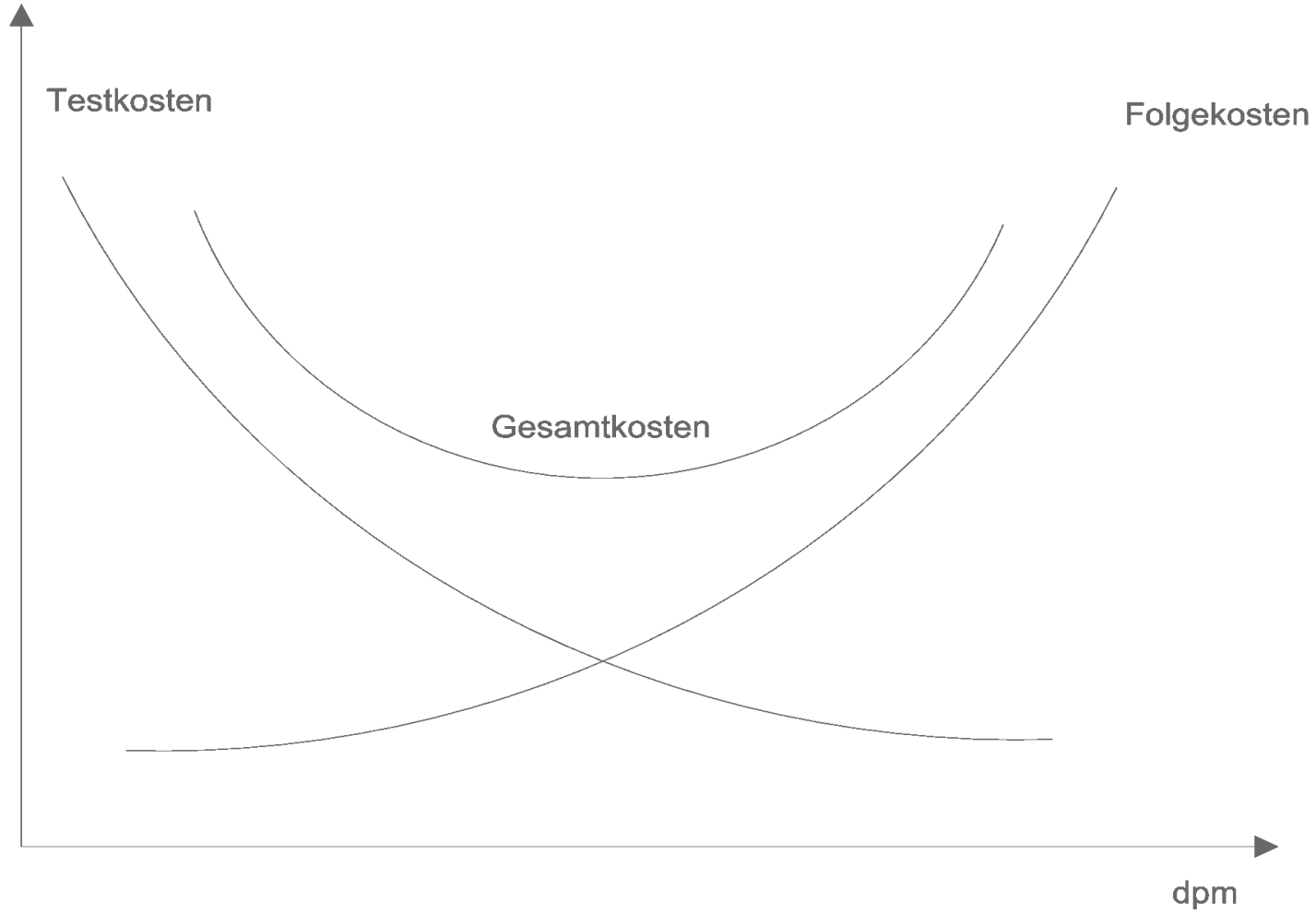
Gliederung

1. **Allgemeine Testprinzipien**
2. **Defekte und Fehlermodelle**
3. **ATPG**
4. **Sequentielle Schaltungen**
5. **DFT**

01 Allgemeine Testprinzipien

- Neue Produkte müssen geprüft werden, ob sie funktionsfähig sind und ihren Zweck erfüllen
- Getestet wird in verschiedenen Stadien der Hardwareentwicklung
- Ziel: fehlerfreier Entwurf, fehler- und defektfreies Produkt, damit wenige Ausfälle auftreten und geringe Folgekosten entstehen
- Fehler erzeugen höhere Kosten je später sie entdeckt werden
- Erschöpfender Test fast nie möglich, deshalb Konstruktion von sinnvollen Testfällen

01 Allgemeine Testprinzipien



02 Defekte und Fehlermodell

- Defekte sind Fehler, die während der Produktion im Bauteil auftreten
- Fehlermodell ist eine mathematische Beschreibung wie ein Defekt das Verhalten des Designs verändert
- Stuck-at-Fault: jede Leitung eines Schaltkreises kann auf einem bestimmten logischen Wert „feststecken", bei n Leitungen 2n mögliche Fehler
- Brückenfehler: 2 Leitungen sind verbunden, obwohl sie es nicht sein sollten, dies führt zu wired-or oder wired-and Logikfunktion, nur physikalisch benachbarte Leitungen müssen betrachtet werden
- Stuck-open-Fehler: eine Leitung ist durchtrennt und weder mit Vcc noch mit Gnd verbunden, ein oder mehrere Inputs abgetrennt
- Delay – Fehler: Signal hat korrekten Wert, aber langsamer als normal

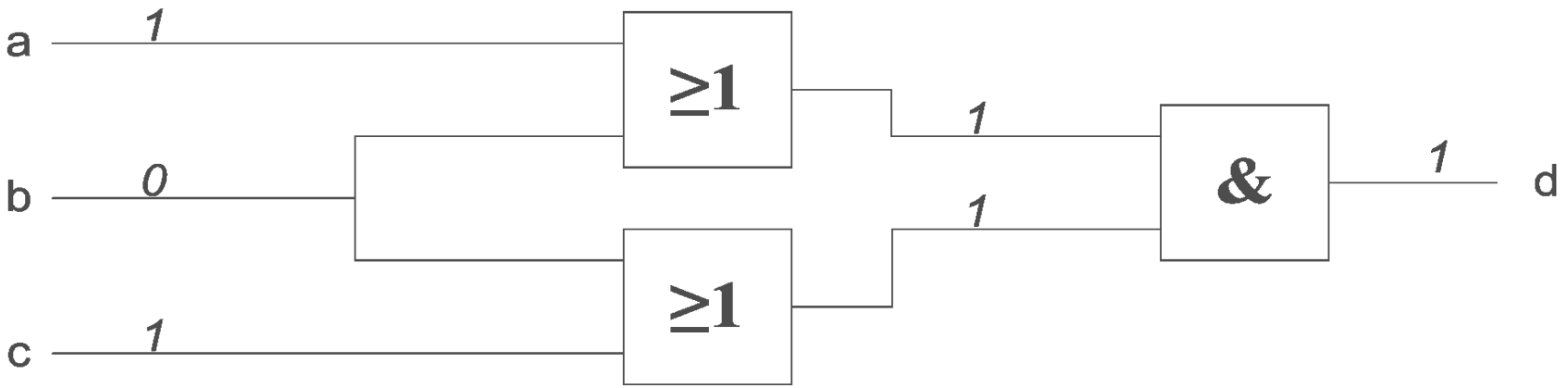
03 ATPG

Test-Pattern-Generierung

- Aus Komplexitätsgründen werden meist nur Einzellfehler angenommen
- Defekte müssen als elektrisches oder logisches Verhalten modellierbar sein
- Ein Fehler muss durch beschränkten Aufwand durch Änderungen an den Primären Eingängen einstellbar und gleichzeitig an den primären Ausgängen sichtbar sein
- Ein Testvektor sollte dabei möglichst viele Fehler erkennen

03 ATPG

Beispiel



03 ATPG

Beispiel

- Ein vollständiger Testsatz für dieses Beispiel

a	b	c	d
0	0	1	0
0	1	0	1
1	0	0	0
1	0	1	1

03 ATPG

Automatisierte-Test-Pattern-Generierung

Ablaufskizze

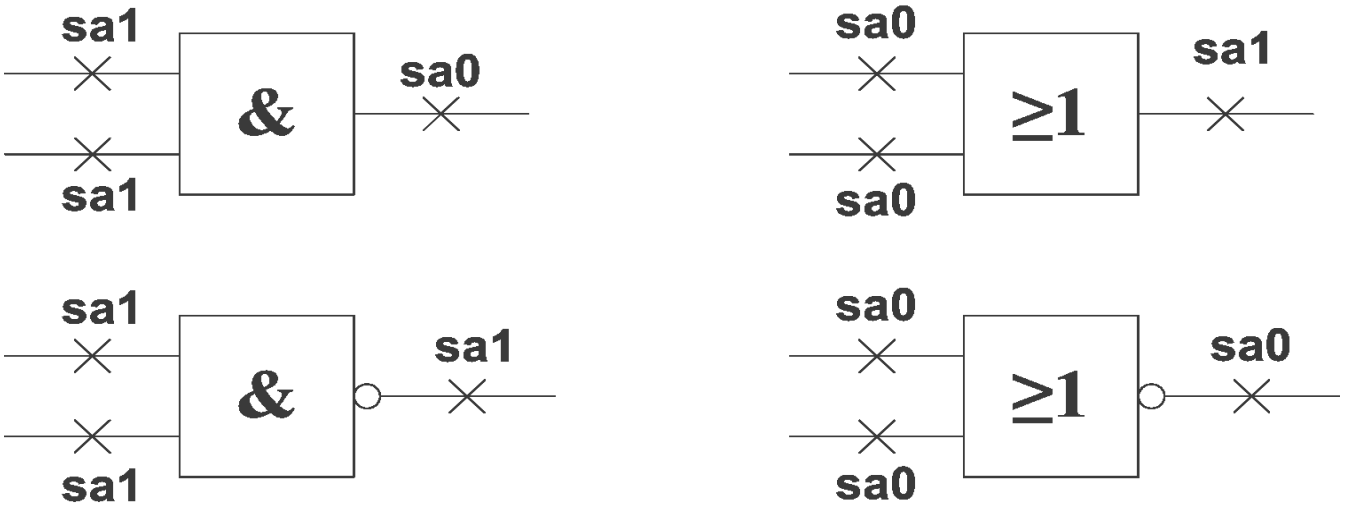
- Erstellen einer Fehlerliste aufgrund des Fehlermodells
- Gutsimulation mit bestimmten Testvektoren
- Simulation des fehlerbehafteten Netzwerks
- Vergleich der Werte an korrespondierenden Ausgängen Gutnetzwerks und des fehlerbehafteten Netzwerks
- Markieren (oder streichen) der entdeckten Fehler aus der Fehlerliste
- Solange bis alle Fehler entdeckt wurden oder Fehlerüberdeckungsgrad hoch genug ist
- Fehlersimulation kann als Modellierung und Simulation des Testvorganges angesehen werden

03 ATPG

Automatisierte-Test-Pattern-Generierung

Reduktion der Fehlerliste

- Die Fehlerliste kann verringert werden indem äquivalente und dominierte Fehler gestrichen werden



03 ATPG

Automatisierte-Test-Pattern-Generierung

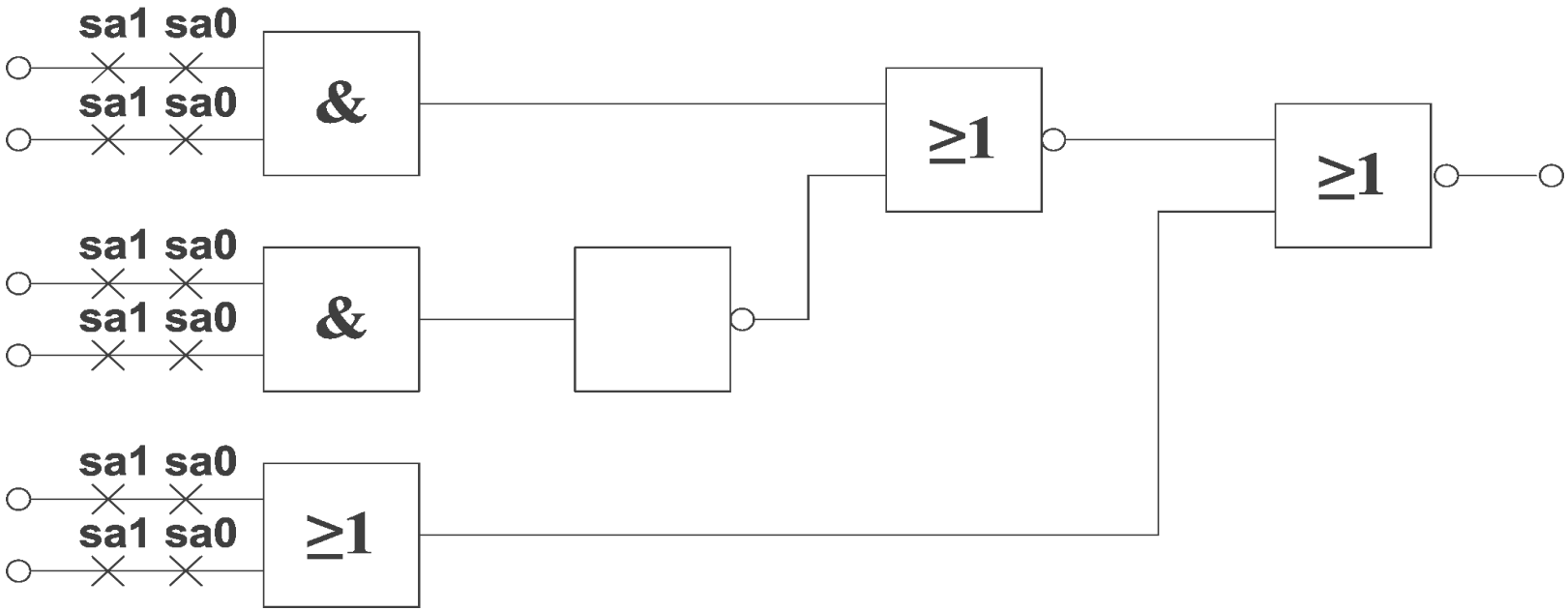
Reduktion der Fehlerliste

- Statt des Fehlers auf der Ausgangsleitung können die äquivalenten Fehler auf der Eingangsleitung betrachtet werden
- Da innerhalb eines Netzwerkes eine Gatter-Eingangsleitung gleichzeitig Ausgangsleitung eines davorliegenden Gatters ist, können Fehler „zurückgeschoben" werden, jeweils unter Beachtung der Funktion des Gatters, bis zu den primären Eingängen oder Fan-out-Leitungen

03 ATPG

Automatisierte-Test-Pattern-Generierung

Reduktion der Fehlerliste

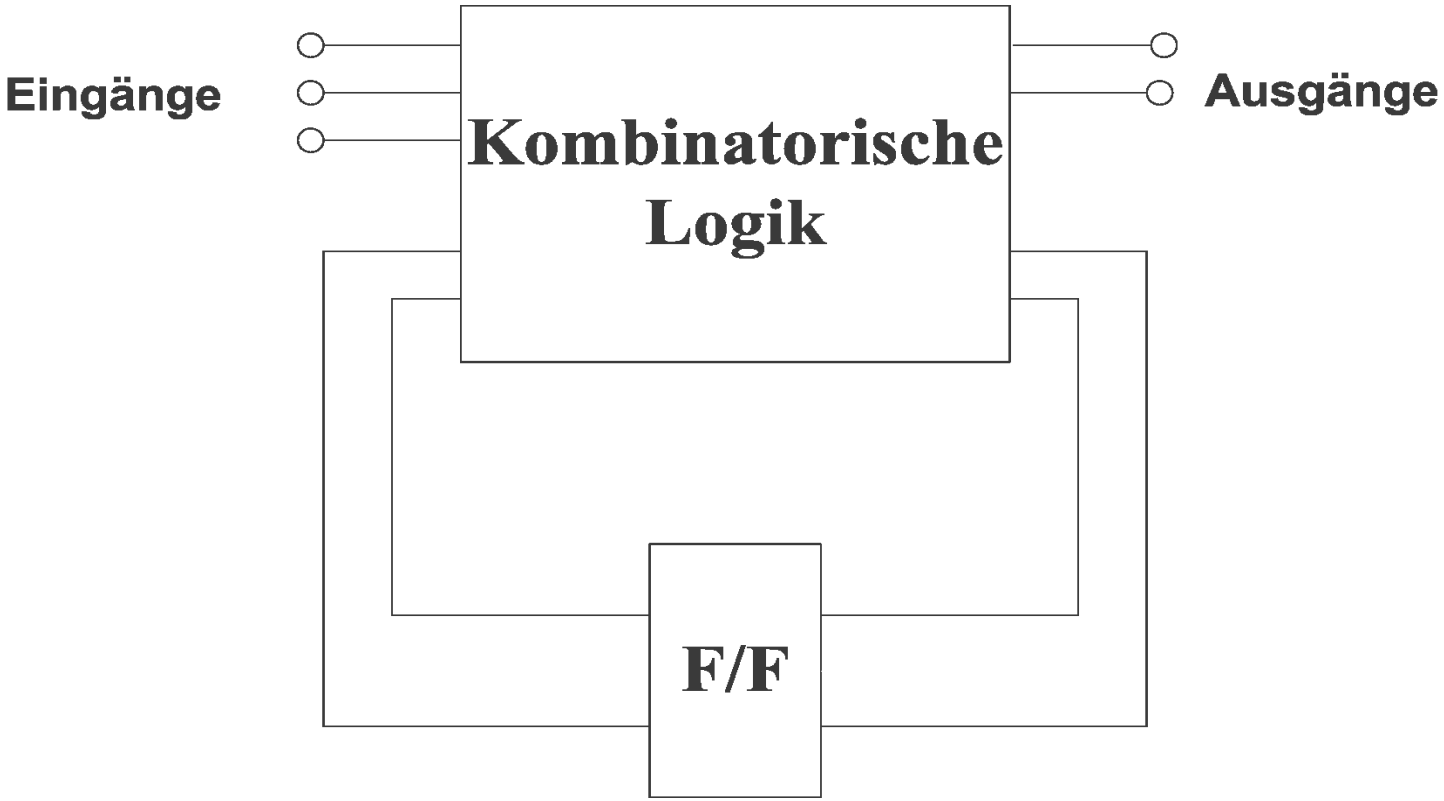


03 ATPG

Automatisierte-Test-Pattern-Generierung

- Nachdem alle pseudozufällig erzeugten Testvektoren simuliert wurden und noch Fehler in der Fehlerliste stehen, müssen dafür explizit Testvektoren generiert werden
- Für diese Aufgabe gibt es spezielle Verfahren, wie den D-Algorithmus
- Es gibt Weiterentwicklungen wie PODEM- oder FAN-Verfahren
- Entscheidungen werden aufgrund bestimmter Heuristiken getroffen, die zu schnelleren Lösungen führen
- Qualität der ATPG wird anhand der abgedeckten Fehlermodelle, Anzahl der Testvektoren, Fehlerabdeckung und der benötigten Zeit bestimmt

04 Sequentielle Schaltungen



04 Sequentielle Schaltungen

- Für kombinatorische Schaltungen gibt es einige sehr effektive TPG-Algorithmen
- Es ist aber nicht möglich diese unmittelbar auf sequentielle Schaltungen zu übertragen, da die Speicherzustände berücksichtigt werden müssen
- Um einen Fehler zu erregen und zu einem primären Ausgang zu propagieren sind oft sehr lange Folgen von Eingangsvektoren notwendig
- Kombinatorische Algorithmen nur anwendbar, wenn man die FFs gezielt setzen kann

05 DFT

Design for Testability

- Alle Maßnahmen, die zur Verbesserung von wenigstens einem der folgenden vier Punkte führen

1. Initialisierbarkeit
2. Fehlersteuerbarkeit
3. Fehlerbeobachtbarkeit
4. Vermeiden potentieller Defekte

- Ziele: Test zu vereinfachen oder gar erst zu ermöglichen, dabei aber möglichst wenig Pins, Logik und Verdratung zu verbrauchen
- Designänderungen, die den Zugang zu internen Schaltkreiselementen gewährleisten um so z.B. die internen Zustände zu verändern
- Scan-Path-Verfahren weit verbreitet
- Anzahl an Testpins ist begrenzt, deshalb kann man Testdaten für einzelne Scan-Chains komprimieren

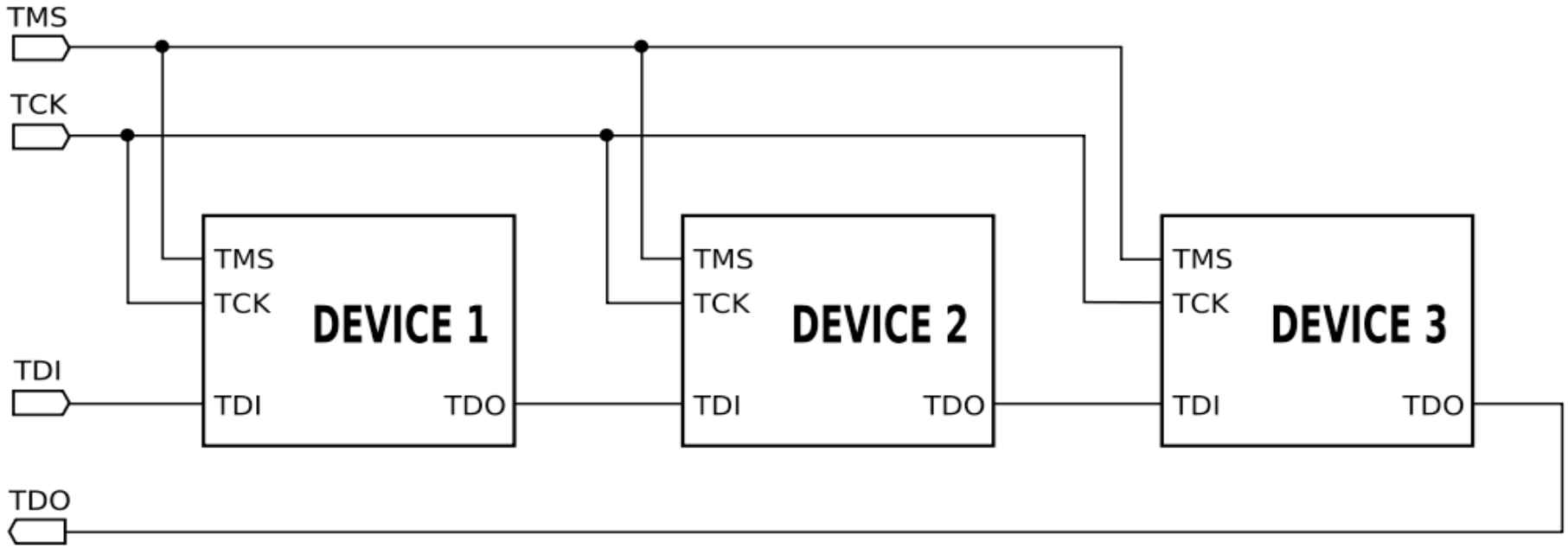
05 DFT

JTAG

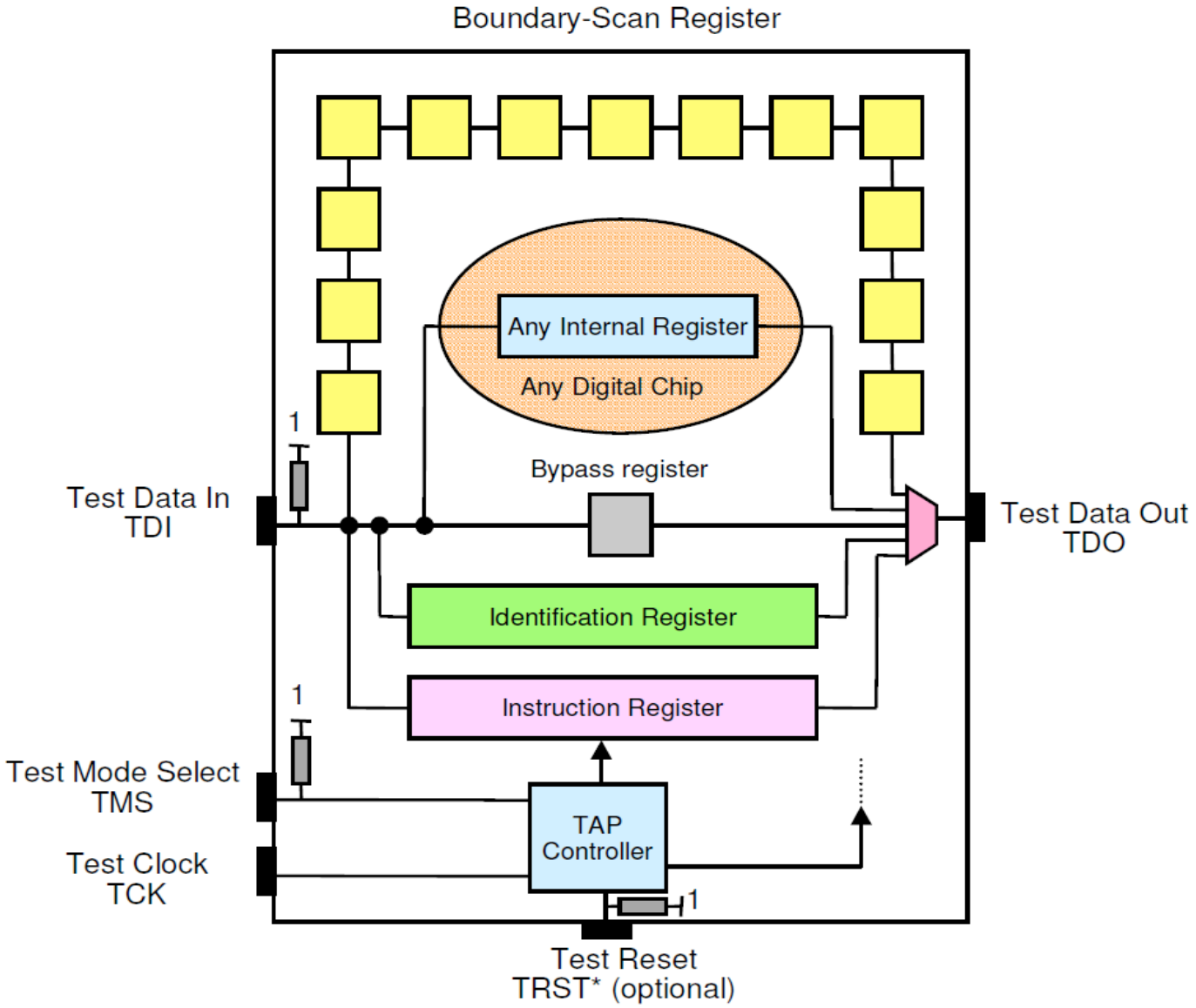
- Mittlerweile Synonym für den IEEE 1149.1 Standard (Standard Test Access Port und Boundary-Scan Architecture)
- BSDL (Boundary Scan Description Language) Teil des Standards
- TAP (Test Access Port) mit 4 I/O Pins und optional TRST
- TAP Controller
- Instruction Register
- Data Registers
- Inzwischen wird JTAG vermehrt auch zur Konfiguration von FPGAs und CPLDs sowie zum Programmieren und Debuggen von Mikrocontrollern verwendet

05 DFT

JTAG


TMS
TCK
TDI
TDO
TMS
TCK
DEVICE 1
TDI
TDO
TMS
TCK
DEVICE 2
TDI
TDO
TMS
TCK
DEVICE 3
TDI
TDO

05 DFT
JTAG


Boundary-Scan Register
Any Internal Register
Any Digital Chip
1
Bypass register
Test Data In
TDI
Test Data Out
TDO
Identification Register
1
Instruction Register
Test Mode Select
TMS
TAP
Controller
Test Clock
TCK
1
Test Reset
TRST* (optional)

Vielen Dank für Ihre Aufmerksamkeit!

Quellen

[1] Scheffer, Louis, EDA for IC system design, verification, and testing, 2006
[2] Prof. Dr.-Ing. habil. Bernd Straube, Vorlesung „Logiksimulation und Test“
[3] http://www.corelis.com/products/Boundary-Scan_Tutorial.htm
[4] http://www.asset-intertech.com/pdfs/Boundary-Scan_Tutorial_2007.pdf