



Universität Stuttgart



Institut für Technische Informatik
Pfaffenwaldring 47, 70569 Stuttgart

Diplomarbeit Nr. 3451

Micro Architecture for Fault Tolerant NoCs

Stefan Zimmermann

Studiengang: Elektrotechnik und Informationstechnik

Prüfer: Prof. Dr. rer. nat. habil.
Hans-Joachim Wunderlich
Prof. Dr.-Ing. Andreas Kirstädter

Betreuer: M.Sc. Atefe Dalirsani
Dipl.-Inf. Michael Imhof

begonnen am: 21. Januar 2013

beendet am: 23. Juli 2013

CR-Klassifikation: B.8.1, C.1.2, C.2.1, C.2.3, E.1

Kurzfassung

Durch die Skalierung der Technologie ist es möglich andere Architekturen umzusetzen. So werden immer mehr Kerne auf einem Chip untergebracht. Mit der steigenden Anzahl an Kernen steigt der Kommunikationsbedarf. Die Alternative zu busbasierten Kommunikationen eines Ein-Chip-Systems ist ein Network-on-Chip. Ein Network-on-Chip basiertes System mit hunderten oder tausenden an Kernen hat bessere Performanceeigenschaften und einen besseren Datendurchsatz als ein vergleichbares busbasiertes Ein-Chip-System. Das Netzwerk auf einem Chip wird durch Switche aufgespannt. An jeden dieser Switche ist jeweils ein Kern angeschlossen.

Durch Produktionsschwankungen oder nach einer gewissen Zeit kann der Chip defekt werden. Die dadurch auftretenden Defekte können einen wesentlichen Einfluss auf die Systemperformance und die Systemverfügbarkeit haben. Es muss sichergestellt werden, dass eine fehlerhafte Verbindung zwischen einem Switch und einem Kern oder ein defekter Kern den Systembetrieb nicht beeinflusst. Dies ist der Grund, dass diese Fehler erkannt und toleriert werden müssen.

Um fehlerhafte Verbindungen zwischen dem Switch und dem Kern zu erkennen, wird die Anschlussfunktionalität bei Auftreten eines Fehlers überprüft. Informationen über die fehlerhaften Anschlüsse werden lokal in jedem Switch gespeichert.

Eine redundante Verbindung zwischen dem Kern und den Switchen hält die Kernverbindung aufrecht, wenn ein Switch oder eine Verbindung zu dem Kern beschädigt ist. Drei Konfigurationen, mit zwei, mit drei und mit vier Switchverbindungen zu einem Kern, werden durch eine numerische Verfügbarkeitsberechnung untersucht.

Die fehlertolerante Architektur modifiziert außerdem den Routingalgorithmus. Die Pakete müssen zu jedem Kern auch durch die alternative Verbindung zugestellt werden. Durch diese Erweiterungen kann die Verfügbarkeit und die Performance erhöht werden.

Um die Zuverlässigkeit des Systems zu erhöhen, werden transiente Fehler von permanenten Fehlern unterschieden. Hierfür wird die Überprüfung der Verbindungen erweitert.

Die Architektur wird dazu verwendet dass fehlerhafte Kerne erkannt werden. Die Operationen werden auf drei identischen Kernen, die an den gleichen Switch angeschlossen sind, ausgeführt. Ist das Ergebnis eines Kerns anders als das von den anderen Kernen, dann

wird der fehlerhafte Kern von diesem Switch getrennt. Durch diese dreifach modulare Redundanz steigt die Zuverlässigkeit des Systems.

Abstract

Due to the scaling of technology, it is possible to implement other architectures. Thus, more and more cores are placed on a chip. With the increasing number of cores is increasing the demand for communication. The alternative to the bus-based communication of a system-on-chip is a network-on-chip. A network-on-chip based system with hundreds or thousands of cores has a better performance and higher throughput than a comparable bus-based system-on-chip. The network on a chip is spanned by the switches. To each of these switches is connected to a core each.

With the increasingly complex systems, the error rate of a system increases. The defects occurring thereby can have a significant impact on system performance and system availability. It must be ensured that a faulty connection between a switch and a core or a defective core will not affect the system operation. For this reason that these faults must be detected and tolerated.

To detect faulty connections between the switch and the core, the port functionality of the connection is checked when an error occurs. Information about the faulty port is stored locally in the switch.

A redundant connection between the core and the switches keeps the core connected if a switch breaks down or the connection to the core is broken. Three configurations, with two, three and four switches connected to a core are examined by numerical reliability calculations.

The fault-tolerant architecture also modifies the routing algorithm. The packets must be delivered to each core through alternative connections too. Through these extensions, the availability and performance can be increased.

In order to increase the reliability of the system transient errors of permanent errors distinguish. For this purpose, the verification of connections is expanded.

The architecture is used to detect the faulty cores. The operations are scheduled to be performed on three identical cores connected to the same switch. If the result of one core is different to the other cores then the faulty core is disconnected from that switch. Through this triple modular redundancy, the reliability of the system increases.

Inhaltsverzeichnis

Kurzfassung	I
Abstract	III
Abkürzungen, Symbole und Formelzeichen	IX
1. Einleitung	1
1.1. Motivation	1
1.2. Kapitelorganisation	2
2. Grundlagen	3
2.1. System-on-Chip	3
2.1.1. Topologien eines System-on-Chip	3
2.2. Network-on-Chip	6
2.2.1. Topologien eines Network-on-Chip	7
2.3. Zyklische Blocksicherung	8
2.3.1. Hamming-Distanz	8
2.3.2. Detektierende und korrigierende Codes	9
2.3.3. Beispiel	9
2.4. Fehlertoleranz und Zuverlässigkeitsanalyse	10
2.4.1. Zuverlässigkeit und Kosten	11
2.4.2. Reparierbare und nicht reparierbare Systeme	11
2.4.3. Zeitabhängigkeit von Ausfällen bei nicht reparierbaren Systemen - Badewannenkurve	12
2.4.4. Zeitabhängigkeit von Ausfällen bei reparierbaren Systemen	12
2.4.5. Skalierbare Leistung im Vergleich zur Systemverfügbarkeit	13
2.4.6. Quantitative Zuverlässigkeitsanalyse	13
2.4.7. Zuverlässigkeitsbetrachtungen für redundante Systeme	15
2.4.8. Modulare Bauweise	17
3. Zugrunde liegende Architektur des NoCs	19
3.1. Überblick auf die zugrunde liegende Architektur des NoCs	19

3.2.	Flittypen	19
3.2.1.	Kopfflit	20
3.2.2.	Datenflit	20
3.2.3.	Endeflit	20
3.3.	Switch	21
3.3.1.	Router	21
3.3.2.	Paketaufkommen im NoC	22
3.4.	Netzwerkschnittstelle	22
4.	Numerische Bestimmung der Erreichbarkeit des Systems	25
4.1.	Erreichbarkeit eines NoCs mit redundanten Anbindungen	27
4.1.1.	Bei Ausfällen von gesamten Switchen oder IP-Cores	27
4.1.2.	Bei Ausfällen von Switch-Switch-Verbindungen	27
4.1.3.	Bei Ausfällen von Switch-Switch- und Switch-IP-Core-Verbindungen	31
4.1.4.	Vergleich der Simulationen, wenn nur Switch-Switch-Verbindungen bzw. Switch-Switch- und Switch-IP-Core-Verbindungen ausfallen	34
4.2.	Erreichbarkeit eines NoCs mit redundanten Anbindungen und einer Erkennung von transienten Verbindungsdefekten	34
5.	Numerische Abschätzung der Systemzuverlässigkeit	39
5.1.	Ausfall von gesamten Komponenten (permanente Fehler)	40
5.2.	Ausfall von Switch-Switch- und Switch-IP-Core-Verbindungen (permanente Fehler)	42
5.3.	Ausfall von Switch-Switch- und Switch-IP-Core-Verbindungen (permanente und transiente Fehler)	44
6.	Erkennung und Lokalisierung von fehlerhaften Verbindungen	45
6.1.	Fehlererkennung (Ende zu Ende)	46
6.2.	Fehlerlokalisierung (Komponente zu Komponente)	46
6.3.	Überprüfung von Steuersignalen	49
6.4.	Erweiterungen für die Fehlererkennung und Fehlerlokalisierung	50
6.4.1.	Flittypen	50
6.4.2.	Switch	50
6.4.3.	Router	51
6.4.4.	Netzwerkschnittstelle	51
6.4.5.	Erweiterter Routingalgorithmus	52
7.	Redundante Anbindung von IP-Cores an die Switches	59
7.1.	Konzept	59
7.2.	Erweiterungen für die redundante Anbindung	60
7.2.1.	Flittypen	60
7.2.2.	Switch	60
7.2.3.	Router	61
7.2.4.	Netzwerkschnittstelle	62

8. Unterscheidung zwischen permanenten und transienten Fehlern	65
8.1. Konzept	65
8.2. Erweiterungen für die Unterscheidung zwischen permanenten und transienten Fehlern	66
8.2.1. Router	66
8.2.2. Netzwerkschnittstelle	67
9. Dynamische dreifach modulare Redundanz	69
9.1. Konzept	69
9.2. Erweiterungen für die dynamische dreifach modulare Redundanz	70
9.2.1. Flittypen	70
9.2.2. Router	72
9.2.3. Netzwerkschnittstelle	72
10. Validierung des implementierten Systems	73
10.1. Simulationsalgorithmus für die Validierung	73
10.2. Gatterebensimulation mit Fehlerdetektierung, Fehlerlokalisierung und redundanter Anbindung	73
10.2.1. Mit Weiterleitung zu anderen IP-Cores	75
10.3. Gatterebensimulation mit transienten Fehlern	78
10.4. Gatterebensimulation mit dynamischer dreifach modularer Redundanz	78
11. Zusammenfassung und Ausblick	83
A. Weitere Ergebnisse von der Bestimmung der maximalen Metrik des Systems	85
A.1. Bei Ausfällen von gesamten Komponenten	85
A.2. Bei Ausfällen von Switch-Switch-Verbindungen	86
A.3. Bei Ausfällen von Switch-Switch- oder Switch-IP-Core-Verbindungen	87
A.4. Bei Ausfällen von Switch-Switch- oder Switch-IP-Core-Verbindungen mit transienten Fehlern	88
B. Weitere Algorithmen	93
B.1. Weitere Routingalgorithmen (Komponente zu Komponente)	93
B.1.1. Routingalgorithmen inklusive der Erweiterung für das yx-Routing	93
C. Weitere Konfigurationen für die redundante Anbindung	97
C.1. Zweidimensionales Gitter inkl. der redundanten Anbindung von IP-Cores	97
C.2. Switch	98
C.3. Netzwerkschnittstelle	99
Abbildungsverzeichnis	104
Tabellenverzeichnis	105
Literaturverzeichnis	107




Abkürzungen, Symbole und Formelzeichen

Abkürzungen

<i>Abkürzung</i>	<i>Erklärung</i>
buf_full_in	Puffer des Ports ist voll (Eingangssignal) (engl.: <u>B</u> uffer is <u>f</u> ull (<u>i</u> n))
buf_full_out	Puffer des Ports ist voll (Ausgangssignal) (engl.: <u>B</u> uffer is <u>f</u> ull (<u>o</u> ut))
CRC	Zyklische Blocksicherung (engl.: <u>C</u> yclic <u>R</u> edundancy <u>C</u> heck)
Controller	Steuereinheit der Netzwerkschnittstelle (engl.: <u>N</u> etwork <u>I</u> nterface <u>C</u> ontroller)
DEC	Zweifache Fehlerkorrektur (engl.: <u>D</u> ouble <u>E</u> rror <u>C</u> orrecting)
DED	Zweifache Fehlererkennung (engl.: <u>D</u> ouble <u>E</u> rror <u>D</u> etecting)
DTMR	Dynamische dreifach modulare Redundanz (engl.: <u>D</u> ynamic <u>T</u> riple <u>M</u> odular <u>R</u> edundancy)
FIFO	Pufferspeicher nach dem Prinzip einer Warteschlange (engl.: <u>F</u> irst <u>I</u> n – <u>F</u> irst <u>O</u> ut)
Flit	Elementares Datenpaket der Flusskontrolle
IC	Integrierte Schaltung (engl.: <u>I</u> ntegrated <u>C</u> ircuit)
IP-Core	Wiederverwendbarer Teil eines Chipdesigns (engl.: <u>I</u> ntellectual <u>P</u> roperty <u>C</u> ore)
MHD	<u>M</u> inimale <u>H</u> amming- <u>D</u> istanz
MPP	Massive Parallelverarbeitung (engl.: <u>M</u> assively <u>P</u> arallel <u>P</u> rocessing)
MTBF	Mittlere Zeit zwischen Ausfällen (engl.: <u>M</u> ean <u>T</u> ime <u>B</u> etween <u>F</u> ailures)
MTTF	Mittlere Zeit bis zum Ausfall (engl.: <u>M</u> ean <u>T</u> ime <u>T</u> o <u>F</u> ailure)
MUX	<u>M</u> ultiplexer
NI	Netzwerkschnittstelle (engl.: <u>N</u> etwork <u>I</u> nterface)
NoC	<u>N</u> etwork- <u>o</u> n- <u>C</u> hip
Port	Anschluss (engl.: <u>P</u> ort)
rec	Empfangssignal (engl.: <u>R</u> eception <u>S</u> ignal)
SEC	Einfache Fehlerkorrektur (engl.: <u>S</u> ingle <u>E</u> rror <u>C</u> orrecting)
SED	Einfache Fehlererkennung (engl.: <u>S</u> ingle <u>E</u> rror <u>D</u> etecting)

send	Sendesignal (engl.: <u>S</u> ending <u>S</u> ignal)
SMP	Symmetrische Mehrprozessorverarbeitung (engl.: <u>S</u> ymmetric <u>M</u> ultiprocessing)
SoC	Ein-Chip-System (engl.: <u>S</u> ystem- <u>o</u> n-a- <u>C</u> hip)
TED	Dreifache Fehlererkennung (engl.: <u>T</u> riple <u>E</u> rro <u>r</u> <u>D</u> etecting)
TMR	Dreifach modulare Redundanz (engl.: <u>T</u> riple <u>M</u> odular <u>R</u> edundancy)
XOR	Logisch <u>E</u> xklusiv- <u>O</u> der-Verknüpfung

Symbole

<i>Symbol</i>	<i>Erklärung</i>
	IP-Core
	Netzwerkschnittstelle
	Switch

Formelzeichen

<i>Formelzeichen</i>	<i>Erklärung</i>	<i>Einheit</i>
Lateinische Buchstaben		
a_p	Erreichbarkeit der IP-Cores in Prozent (engl.: <u>a</u> ccessibility <u>p</u> ercent)	-
B	Polynom des CRC-Verfahren	-
$B_{\text{erweitert}}$	Polynom des CRC-Verfahren inklusive der angehängten CRC-Bits	-
d	Anzahl der fehlerhaften Bits einer Bitfolge	-
D_{max}	Anzahl der Verbindungen die <u>m</u> aximal <u>d</u> efekt werden können	-
$f(t)$	Wahrscheinlichkeitsdichtefunktion	-
E_p	Prozentualer Anteil von der Geasmtdefektanzahl	-
G	Generatorpolynom für das CRC-Verfahren	-
MTBF	Mittlere Zeit zwischen Ausfällen des Systems (bei reparierbaren Systemen)	s
MTTF	Mittlere Zeit bis zum Ausfall des Systems (bei nichtreparierbaren Systemen)	s
$MTTF_{\text{Sim}}$	Mittlere Zeit bis zum Ausfall des Systems bei numerischen Simulationen	s

P_f	Wahrscheinlichkeit nach wie vielen fehlerhaften Komponenten beziehungsweise Verbindungen das System gerade nicht mehr funktionsfähig ist (engl.: <u>f</u> unctional <u>P</u> robability)	%
P_{MTTF}	<u>P</u> roportionalitätsfaktor von $MTTF_{Sim}$ der vierfachen Anbindung und $MTTF_{Sim}$ der einfachen Anbindung	-
$R(t)$	Zuverlässigkeit (engl.: <u>R</u> eliability)	-
$R_{IP-Core}$	Zuverlässigkeit eines <u>I</u> P- <u>C</u> ores	-
R_i	Zuverlässigkeit der <u>i</u> -ten Komponente	-
R_P	Rest der <u>P</u> olynomdivision	-
R_{TMR}	Zuverlässigkeit des <u>T</u> MRs	-
$R_{Mehrheitsentscheider}$	Zuverlässigkeit des <u>M</u> ehrheitsentscheiders beim TMR	-
t	Zeitliche Variable	s

Griechische Buchstaben

β	Gestalt-Parameter der Weibull-Verteilung	-
η	Charakteristische Lebensdauer bzw. Maßstab-Parameter der Weibull-Verteilung	s
λ	Konstante Ausfallrate	1/s
λ_i	Konstante Ausfallrate der <u>i</u> -ten Komponente	1/s
$\lambda_{IP-Core}$	Ausfallrate eines <u>I</u> P- <u>C</u> ores	1/s
λ_R	Ausfallrate von <u>r</u> eparierbaren Systemen	1/s
$\lambda_{Mehrheitsentscheider}$	Ausfallrate des <u>M</u> ehrheitsentscheiders des TMRs	1/s
λ_{Sim}	Ausfallrate bei den numerischen <u>S</u> imulationen	1/s

Einleitung

1.1. Motivation

Die kleinste Strukturgröße auf einem Chip kann nicht unbegrenzt kleiner werden. Deshalb wird nach weiteren Architekturen gesucht, um die Rechenleistung weiter zu erhöhen. Eine Möglichkeit ist, dass mehrere Recheneinheiten auf einem Chip untergebracht werden.

Zum Zeitpunkt der Erstellung der vorliegenden Arbeit existieren bereits auf einem Chip teilweise mehrere tausende Recheneinheiten. So sind auf aktuellen Graphikkarten über 3000 Kerne vorhanden [nvi13]. Die große Anzahl an Recheneinheiten führt zu einem großen Kommunikationsbedarf. Dieser Kommunikationsbedarf wird bisher über ein busbasiertes System verarbeitet. Jedoch kann bei einem busbasierten System, zu einem Zeitpunkt, nur eine der angebotenen Komponenten senden. Eine Alternative zum busbasierten System ist, dass ein Netzwerk auf dem Chip aufgespannt wird. Das Netzwerk auf dem Chip wird durch Switche aufgespannt. An jedem dieser Switche ist jeweils ein Kern angebunden. Hierdurch können die Komponenten besser eine parallele Kommunikation durchführen.

Nach einer gewissen Zeit oder durch Produktionsschwankungen kann der Chip defekt werden. Ein Defekt kann erhebliche Auswirkungen auf das System haben, die bis zum Systemausfall führen können. Als Beispiel seien die Bereiche der Luft- und Raumfahrt oder der Medizin genannt, in denen fehlertolerante Systeme von großer Wichtigkeit sind.

Durch ein Network-on-Chip steigt die Wahrscheinlichkeit, dass ein Kern nicht so schnell von dem Netzwerk getrennt wird. Um die Zuverlässigkeit des Systems zu erhöhen, werden die Kerne mehrfach an die Switche angebunden. Um diese Erweiterung sinnvoll zu nutzen, werden die fehlerhaften Verbindungen detektiert und lokalisiert. Da es neben den permanenten Fehlern auch transiente Fehler gibt, ist es sinnvoll, diese Fehlerarten zu unterscheiden. Sofern die transienten Fehler als permanente Fehler erkannt werden, ist das System schneller nicht mehr funktionsfähig.

Durch die erweiterte Architektur mit der redundanten Anbindung ist es durch einen geringen Aufwand möglich, dass eine Berechnung nicht nur auf einem Kern ausgeführt wird, sondern auf drei parallel. Die Ergebnisse dieser Berechnungen werden dann untereinander

verglichen und entschieden, welche der Ergebnisse verwendet wird. Dadurch, dass die Berechnungen auf mehreren Kernen durchgeführt werden, können zusätzlich die Kerne selbst auf Fehler untersucht werden.

[O’C90] [Scho5]

1.2. Kapitelorganisation

In dem Kapitel 2 wird auf diverse theoretische Grundlagen zu diesem Kontext eingegangen. Der grundlegende Aufbau des bearbeiteten Network-on-Chip wird in dem Kapitel 3 erklärt. Wie groß die maximal Erreichbarkeit bei zufälligen Ausfällen von Verbindungen ist wird in dem Kapitel 4 durch numerische Simulationen diskutiert. Dabei werden unterschiedliche Erweiterungen betrachtet.

In den folgenden Kapiteln 6 bis 9 wird erklärt wie die in Kapitel 4 simulierten Erweiterungen implementiert werden. In Kapitel 6 wird darauf eingegangen wie Fehler detektiert und lokalisiert werden. Auf die redundante Anbindung der Kerne zu den Switchen wird in Kapitel 7 eingegangen. Die bis dahin beschriebene Architektur ist der Ausgangspunkt für die zwei zusätzliche Erweiterungen. Auf die Unterscheidung von permanenten und transienten Fehlern wird in dem Kapitel 8 eingegangen. Auf der Basis des Network-on-Chip wird eine dreifach modulare Redundanz umgesetzt, was in Kapitel 9 erklärt wird. Alle Teile des implementierten Systems werden zusammen in Kapitel 10 validiert.

Kapitel 11 fasst die Arbeit zusammen und gibt einen Ausblick, was noch gemacht werden kann.

Grundlagen

2.1. System-on-Chip

Bei einem System-on-Chip (SoC) gibt es unterschiedliche Komponenten. Alle Komponenten, die Daten speichern oder verarbeiten können, werden zu IP-Cores zusammengefasst. Jeder dieser IP-Cores hat eine spezielle Funktion, so kann es sich zum Beispiel um eine Recheneinheit, eine Renderingeinheit oder eine Speichereinheit handeln. Die verschiedenen IP-Cores sind untereinander verbunden. Dafür gibt es eine ganze Reihe an möglichen Topologien (Abschnitt 2.1.1), wobei meistens die Bustopologie verwendet wird. Die Kommunikation zwischen den Komponenten kann synchron oder asynchron implementiert werden.

2.1.1. Topologien eines System-on-Chip

Eine Auswahl an Topologien die für ein SoC verwendet werden können, sind nachfolgend zu finden. Die Topologie des Netzes kann beliebig komplizierter gestaltet werden.

Ringtopologie

In der Ringtopologie (Abbildung 2.1) sind die Komponenten zu einem Ring verbunden. Bei dieser Topologie gibt es sowohl Realisierungen, die nur einen unidirektionalen Informationsfluss haben, als auch Realisierungen, die einen bidirektionalen Informationsfluss aufweisen. Bei der Ringtopologie ist die Geschwindigkeit durch die Ringverbindung zwischen den Komponenten begrenzt.

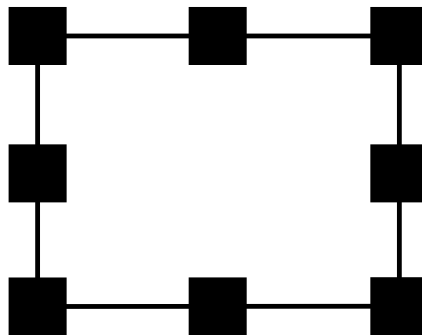


Abbildung 2.1.: Ringtopologie (nach [Chao5])

Sterntopologie

In der Abbildung 2.2 ist eine Sterntopologie zu sehen. Bei dieser Topologie gibt es eine zentrale Komponente, die mit allen anderen Komponenten direkt verbunden ist. Diese Komponente muss den gesamten Informationsfluss zu den anderen Komponenten verarbeiten. Falls nun die zentrale Komponente ausfällt, so kann kein Informationsfluss mehr zwischen den anderen Komponenten des Systems stattfinden.

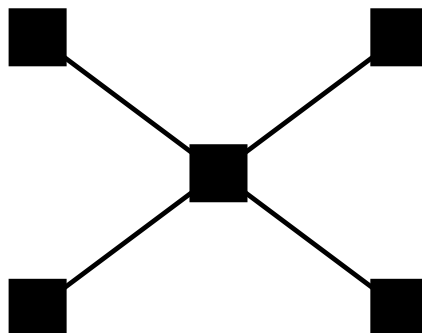


Abbildung 2.2.: Sterntopologie (nach [Scho5])

Baumtopologie

Bei der Baumtopologie (Abbildung 2.3) gibt es eine zentrale Komponente (Stamm), mit der beliebig viele Komponenten (Zweige) verbunden sind. An jede Komponente können dann wiederum beliebig viele Komponenten angebunden werden. Dies kann beliebig oft wiederholt werden. Jedoch darf keine Komponente mit einer anderen Komponente aus einem anderen Zweig verbunden sein. Sollte eine Komponente ausfallen, sind alle tiefer liegenden Komponenten von dem Rest des Systems abgeschnitten.

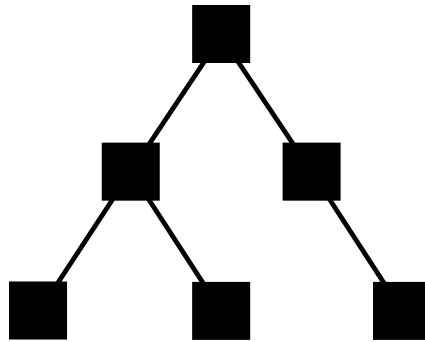


Abbildung 2.3.: Baumtopologie (nach [Scho5])

Vollständig verbundene Topologie

Bei der vollständig verbundenen Topologie (Abbildung 2.4) sind die verschiedenen Komponenten jeweils mit jeder anderen Komponente direkt verbunden. Diese Topologie zu realisieren, verlangt einen hohen Verdrahtungsaufwand.

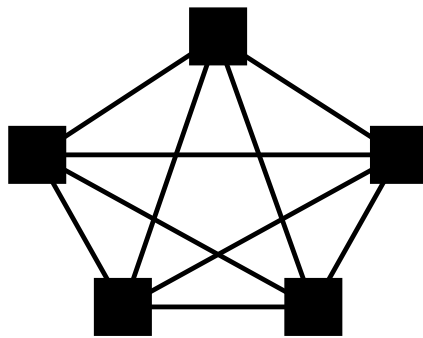


Abbildung 2.4.: Vollständig verbundene Topologie (nach [Scho5])

Unregelmäßige Gittertopologie

Bei der unregelmäßigen Gittertopologie, (Abbildung 2.5) sind die einzelnen Komponenten regelmäßig angeordnet. Jedoch sind die Verbindungen zwischen den einzelnen Komponenten unregelmäßig. Da es eine unregelmäßige Verbindungsstruktur gibt, ist die Implementierung komplexer.

Bustopologie

In der Abbildung 2.6 ist eine Bustopologie zu sehen. Dort sind immer einige der Komponenten über einen gemeinsamen Bus verbunden. Die verschiedenen Busse sind untereinander

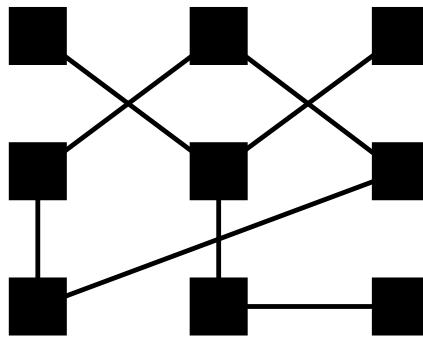


Abbildung 2.5.: Unregelmäßige Gittertopologie (nach [Chao5])

verbunden. Bei der Bustopologie muss immer darauf geachtet werden, dass keine andere Komponente zum gleichen Zeitpunkt auf dem Bus sendet. Dies macht diese Topologie langsam.

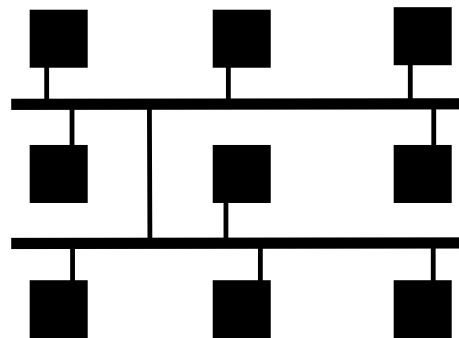


Abbildung 2.6.: Bustopologie (nach [Chao5])

2.2. Network-on-Chip

Bei einem Network-on-Chip (NoC) wird aus einzelnen Switchen ein Netzwerk aufgebaut. Das Netzwerk kann zum Beispiel die Form eines regelmäßigen Gitters oder eines Torusgitters haben (Abschnitt 2.2.1). Über diese Switches sind die einzelnen IP-Cores über die Netzwerkschnittstellen (NI) mit dem Netzwerk verbunden.

Durch diese Architektur des NoCs wird es möglich eine Netzwerkarchitektur aufzubauen, wie sie unter anderem aus dem Bereich des Ethernet bekannt ist. Dadurch wird es möglich verschiedene Performancebereiche zu haben. Des Weiteren werden die Performanceeigenschaften, Datenübertragungsrate, Latenz und Synchronisation verbessert. Durch die regelmäßige Anordnung und Verbindung der Komponenten wird das System einfacher zu skalieren. Des Weiteren unterstützt ein NoC die parallele Kommunikation zwischen verschiedenen IP-Cores. Dabei können die einzelnen Komponenten eines NoCs

so implementiert werden, dass sie wiederverwendbar sind. Mittels eines NoCs kann ein fehlertolerantes System entwickelt werden, das Defekte erkennt, lokalisiert und das System rekonfiguriert. Dadurch wird die Verfügbarkeit und damit auch die Lebenszeit des Systems erhöht. [Wun10] [Cot12] [Chao5] [Kir12]

2.2.1. Topologien eines Network-on-Chip

Eine Auswahl an Topologien die für ein NoC verwendet werden können, ist nachfolgend zu finden. Die Topologie des Netzes kann beliebig komplizierter gestaltet werden. So ist es zum Beispiel möglich, eine Würfeltopologie zu realisieren. Meistens wird eine regelmäßige Gittertopologie oder eine Torustopologie verwendet. In dieser Arbeit wird die regelmäßige Gittertopologie verwendet.

Bei einem NoC stellt jeder dieser Komponenten ein Switch dar. An jeden dieser Switches ist über eine Netzwerkschnittstelle ein IP-Core angeschlossen. Die Verbindungen unter den Switches selber wie auch zwischen den Switches und dem IP-Cores können sowohl unidirektional wie auch bidirektional implementiert werden.

Regelmäßige Gittertopologie

Bei der regelmäßigen Gittertopologie (Abbildung 2.7) sind die einzelnen Komponenten und die einzelnen Verbindungen zwischen den Komponenten in einer regelmäßigen Struktur angeordnet.

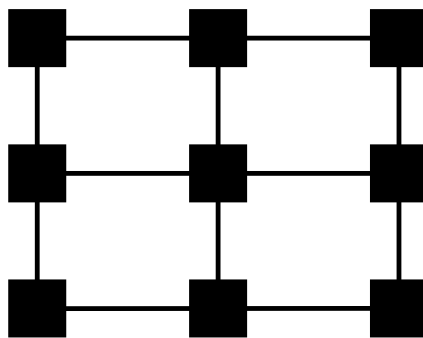


Abbildung 2.7.: Regelmäßige Gittertopologie (nach [Chao5])

Torustopologie

Die Torustopologie (Abbildung 2.8) ist eine Erweiterung der regelmäßigen Gittertopologie. Es werden die außen liegenden Komponenten mit den auf der gegenüberliegenden Seite außen liegenden Komponenten verbunden. Dies erhöht den Verdrahtungsaufwand. Jedoch wird dadurch auch die Datenübertragungsrates erhöht.

[Wun10] [Cha05] [Scho5]

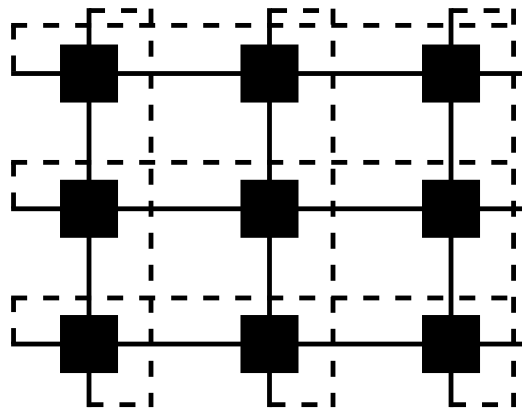


Abbildung 2.8.: Torustopologie (nach [Scho5])

2.3. Zyklische Blocksicherung

Das zyklische Blocksicherungsverfahren (CRC-Verfahren) kann bei Übertragungen eingesetzt werden, die auf einer blockweisen Übertragung beruhen. Mittels dieses Verfahrens können die Blöcke nach Fehlern untersucht werden. Bei diesem Verfahren wird eine Prüfsumme des zu übertragenden Blockes ermittelt und an diesen angehängt. Wenn nun bei der Übertragung ein Bit verändert wird, so kann dieses erkannt und korrigiert werden. Das CRC-Verfahren basiert auf der Modulo-2-Arithmetik.

Bei dem CRC-Verfahren werden die zu übertragenden Bits als Polynom $B(x)$ betrachtet. Dieses Polynom wird nun durch ein Generatorpolynom $G(x)$ dividiert. Das Generatorpolynom für das CRC-16 hat zum Beispiel die Gestalt $x^{16} + x^{15} + x^2 + 1$. Das entstandene Restpolynom wird an den zu übertragenen Block angehängt. Da das übertragene Polynom nun ein Vielfaches des Generatorpolynoms ist, muss das empfangene Polynom noch durch das Generatorpolynom dividiert werden. Sofern nun kein Fehler bei der Übertragung aufgetreten ist, ist der Rest $R_P(x) = 0$. Bei dem CRC-16 können maximal 16 Bitfehler in einem Übertragungsblog sicher erkannt werden. Alle anderen Bitfehler werden mit einer Wahrscheinlichkeit von bis zu 99,997% erkannt. [Kir12] [Scho5]

2.3.1. Hamming-Distanz

Ein wichtiger Wert in diesem Kontext ist die Hamming-Distanz, die nach dem amerikanischen Mathematiker Richard Wesley Hamming (1915 – 1998) benannt ist. Dabei werden

von zwei konkreten Bitfolgen die Anzahl unterschiedlichen Bitpositionen ermittelt. Dies geschieht mittels einer Exklusiv-Oder-Verknüpfung (XOR-Verknüpfung) auf Bitebene. Danach müssen noch die Einsen gezählt werden.

Bei einem Code ist immer die minimale Hamming-Distanz (MHD) zwischen zwei beliebigen Bitfolgen zu betrachten. Wenn bei einer Bitfolge mit n Bits d fehlerhafte Bits erkannt werden sollen, so muss darauf geachtet werden, dass keine andere mögliche Kombination entsteht. Dafür ist die Bedingung $MHD \geq d + 1$ ausreichend. Sollen die d fehlerhaften Bits der Bitfolge korrigiert werden, so muss darauf geachtet werden dass die Bedingung $MHD \geq 2d + 1$ erfüllt ist.

Beispiel

Bitfolge 1: 10001001
 Bitfolge 2: 10110001
 XOR-Ergebnis: 00111000
 → Hamming-Distanz = 3

2.3.2. Detektierende und korrigierende Codes

Es gibt diverse Codes mittels denen man eine bestimmte Anzahl an Fehlern detektieren und eine geringere Anzahl korrigieren kann. Dies sind unter anderem die SEC-DED-Codes (Single-Error-Correcting (SEC) Double-Error-Detecting (DED) Codes) und die DEC-TED-Codes (Double-Error-Correcting (DEC) Triple-Error-Detecting (TED) Codes). Bei den SEC-DED-Codes können zwei Fehler erkannt und ein Fehler korrigiert werden. Ein Beispiel für solch ein Code ist der Hamming-Code, für diesen Code ist eine MHD von drei erforderlich. Bei den DEC-TED-Codes können drei Fehler erkannt und zwei Fehler korrigiert werden. [Hed84]

2.3.3. Beispiel

Wenn das Polynom $B(x) = x^6 + x^2 + x$ und das Generatorpolynom $G(x) = x^7 + x^3 + 1$ ist, so muss $B(x)$ noch erweitert werden zu $B_{\text{erweitert}}(x) = x^{13} + x^9 + x^8$. Durch die Berechnung

$$\begin{array}{r}
 x^{13} + x^9 + x^8 \\
 - x^{13} - x^9 \\
 \hline
 x^8 - x^6 \\
 - x^8 \\
 \hline
 - x^6 - x^4 - x
 \end{array}
 = (x^7 + x^3 + 1) (x^6 + x) - x^6 - x^4 - x \tag{2.1}$$

erhält man den Rest $R_P(x) = -x^6 - x^4 - x$, dieser Rest wird an das Polynom $B(x)$ angehängt. Somit erhält man das Polynom $x^{13} + x^9 + x^8 + x^6 + x^4 + x$.

Um zu ermitteln ob ein Fehler bei der Übertragung aufgetreten ist, wird die Berechnung

$$\begin{array}{r} x^{13} + x^9 + x^8 + x^6 + x^4 + x \\ - x^{13} - x^9 \qquad \qquad - x^6 \\ \hline \qquad \qquad \qquad x^8 \qquad \qquad + x^4 + x \\ \qquad \qquad \qquad - x^8 \qquad \qquad - x^4 - x \\ \hline \qquad \qquad \qquad \qquad \qquad \qquad 0 \end{array} \quad (2.2)$$

durchgeführt. Sofern sich der Rest $R_P(x) = 0$ ergibt, ist die Übertragung fehlerfrei. Sollte jedoch ein Fehler wie in der folgenden Berechnung auftreten,

$$\begin{array}{r} x^{13} + x^9 + x^8 + x^6 + x^4 \\ - x^{13} - x^9 \qquad \qquad - x^6 \\ \hline \qquad \qquad \qquad x^8 \qquad \qquad + x^4 \\ \qquad \qquad \qquad - x^8 \qquad \qquad - x^4 - x \\ \hline \qquad \qquad \qquad \qquad \qquad \qquad - x \end{array} \quad (2.3)$$

so ist in diesem Beispiel der Rest $R_P(x) = -x$ und somit ist $R_P(x) \neq 0$.

Die MHD des Generatorpolynoms $G(x) = x^7 + x^3 + 1$ ist drei. Durch die Verwendung dieses Generatorpolynoms kann der Hamming-Code angewendet werden. Damit können zwei fehlerhafte Bits detektiert und ein fehlerhaftes Bit korrigiert werden.

2.4. Fehlertoleranz und Zuverlässigkeitsanalyse

Die Zuverlässigkeit ist eine Eigenschaft eines Systems. Diese gibt an, wie verlässlich das System zu einer bestimmten Zeit ist. Wie lange ein System funktionsfähig ist, beruht auf einem stochastischen Prozess. Die Zuverlässigkeit eines Systems kann empirisch oder analytisch bestimmt werden.

In integrierten Schaltungen (IC) gibt es eine Vielzahl von Gründen, die einen Fehler auslösen können. Dazu zählen Oxidfehler, die durch ein zu dünnes oder ein verunreinigtes Gateoxid hervorgerufen werden. Ein weiterer Grund für Fehler sind Metallisierungsdefekte, die durch Leitungsbrüche oder durch Diffusion auftreten können. Ein anderer Aspekt sind die Oberflächendefekte. Diese treten auf, wenn bei der Produktion die Oberflächen verunreinigt werden oder im Betrieb, wenn zum Beispiel das Gehäuse nicht dicht ist. Des Weiteren gibt es bei der Produktion der ICs noch viele weitere Gründe für Fehler, zum Beispiel Goldverunreinigungen, Maskenfehler oder Prozessfehler.

Eine andere Art von Fehlern sind die transienten Fehler. Diese Fehler können mehrere Ursachen haben. Dazu gehören, wenn die Temperatur die Gatterlaufzeiten unterschiedlich beeinflusst, wenn Höhenstrahlung oder radioaktive Strahlung die Signale beeinflussen oder

wenn Signalrauschen auftritt. Eine andere Möglichkeit ist, dass die Schaltung nicht überall mit der nötigen Spannung versorgt werden kann.

Durch einen Fehler in der Schaltung ist alles möglich. Beginnend damit, dass die Funktionalität nur unwesentlich beeinflusst wird, bis dahin, dass es bei dem IC zum Totalausfall kommt. Diese Fehler zu detektieren, dann zu lokalisieren und dann diese zu isolieren, beziehungsweise zu reparieren, ist die Aufgabe eines fehlertoleranten Systems. [Hed84]

2.4.1. Zuverlässigkeit und Kosten

Systeme, die eine geringe Zuverlässigkeit besitzen, müssen oft repariert oder ersetzt werden. Dies führt aufgrund der Reparaturen beziehungsweise der Neuanschaffungen zu erhöhten Kosten, denn die Gesamtkosten setzen sich aus den Anschaffungskosten und den Betriebskosten zusammen. Auf der anderen Seite ist die Anschaffung von hochzuverlässigen Systemen relativ teuer. Zur Entwicklung solcher Systeme wird zum Beispiel mehr Personal benötigt. Außerdem sind in solchen Systemen meist auch teurere Bauelemente verbaut. Diese Thematik der Kosten in Abhängigkeit der Zuverlässigkeit ist in der Abbildung 2.9 zu sehen. [Hed84]

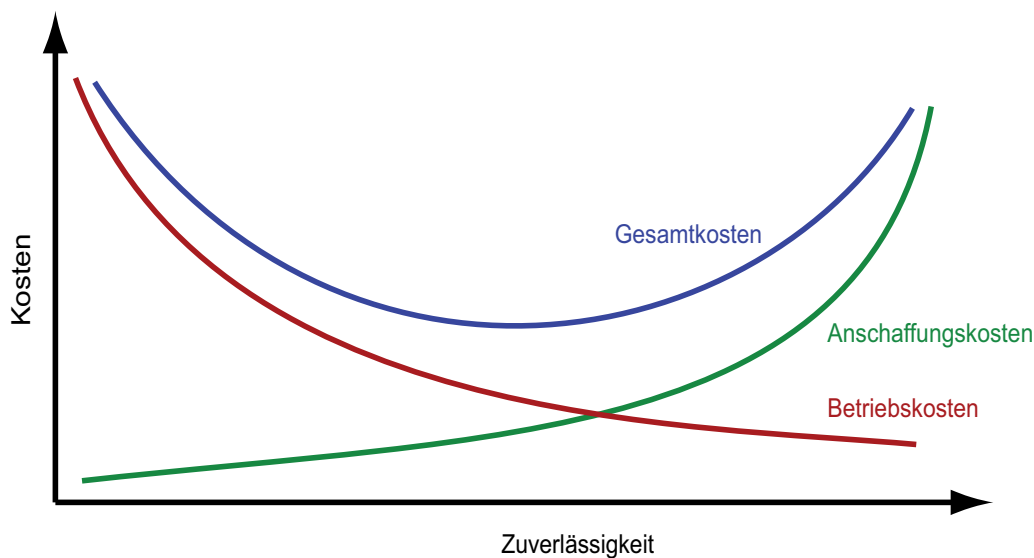


Abbildung 2.9.: Gesamtkosten eines Systems in Abhängigkeit von seiner Zuverlässigkeit (nach [Hed84])

2.4.2. Reparierbare und nicht reparierbare Systeme

Bei der Betrachtung der Zuverlässigkeit eines Systems ist darauf zu achten, ob das System repariert werden kann. Beispiele für nicht reparierbare Teile sind Transistoren oder Glühbirnen. Jedoch können diese Bauelemente gegebenenfalls ausgetauscht werden, wodurch

die Zuverlässigkeit des Systems steigt. Kann ein Bauelement nicht ausgetauscht werden, so kann es sein, dass das System nicht mehr funktionsfähig ist. Um ein System charakterisieren zu können, werden diverse Werte benötigt. Dazu zählt die mittlere Zeit bis zum Ausfall (MTTF) beziehungsweise die erwartete Lebensdauer, bis zu dem Zeitpunkt ein gewisser Prozentsatz versagt hat. Die MTTF kann über die Zuverlässigkeitsfunktion $R(t)$ oder die konstante Ausfallrate λ mittels der Formel

$$\text{MTTF} = \int R(t)dt = \frac{1}{\lambda} \quad (2.4)$$

bestimmt werden.

Bei einem reparierbaren System gibt es die mittlere Zeit zwischen Ausfällen (MTBF). Des Weiteren gibt es die Ausfallrate λ . Bei nicht reparierbaren Systemen ist λ die Wahrscheinlichkeit, dass ein System zu einem gewissen Zeitpunkt ausfällt. Bei einem reparierbaren System gibt $\lambda_R = 1/\text{MTBF}$ die Häufigkeit an, wie oft Ausfälle auftreten. [O'C90]

2.4.3. Zeitabhängigkeit von Ausfällen bei nicht reparierbaren Systemen - Badewannenkurve

Wie sich die Ausfälle über die Zeit bei einem nicht reparierbaren System verteilen, ist in der Abbildung 2.10 zu sehen. Dort sind auch die unterschiedlichen Phasen der Ausfälle zu sehen. In der ersten Phase, der Phase der Frühausfälle (Säuglingssterblichkeit), treten Defekte auf, die in der Fertigung entstanden sind, dort jedoch nicht erkannt werden konnten. Über die Zeit wird diese Anzahl der Ausfälle immer geringer und die Produkte gehen in die Phase der nutzbaren Lebenszeit über. In dieser zweiten Phase treten Fehler auf, die lediglich Zufallsausfälle sind. Da es Zufallsausfälle sind, bleibt die Ausfallrate relativ konstant. Diese Ausfälle können zum Beispiel durch Schmutzpartikel auftreten, die in das Gehäuse eindringen. In der dritten Phase kommen die Ermüdungserscheinungen zum Tragen. Diese führen dann auch zu den Spätausfällen. Gründe dafür können unter anderem zu hohe Temperaturen oder zu hohe Spannungen sein. In dieser Phase steigt die Ausfallrate wieder an. [O'C90] [Hed84] [sim10] [Liu11]

2.4.4. Zeitabhängigkeit von Ausfällen bei reparierbaren Systemen

Bei reparierbaren Systemen, vor allem bei komplexen Systemen, gibt es eine konstante Ausfallrate der einzelnen Komponenten. Dies kommt daher, dass die unterschiedlichen Komponenten unterschiedliche Ausfallwahrscheinlichkeiten haben. Die ausgefallenen Komponenten werden ersetzt. Somit ist das System wieder voll funktionsfähig. Ab einem gewissen Zeitpunkt nehmen die verschleißbedingten Ausfälle jedoch überhand. Somit kann auch bei reparierbaren Systemen ein Verhalten wie in der Abbildung 2.10 beobachtet werden. [O'C90]

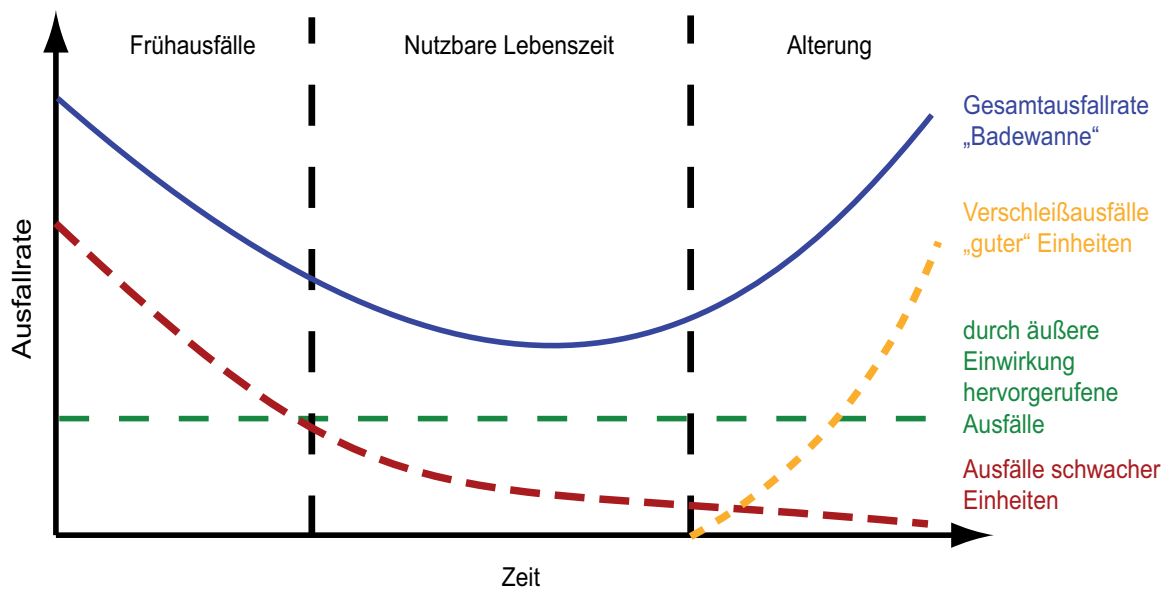


Abbildung 2.10.: Zeitlicher Verlauf der Ausfallrate („Badewannenkurve“) (nach [sim10])

2.4.5. Skalierbare Leistung im Vergleich zur Systemverfügbarkeit

Wie aus der Abbildung 2.11 entnommen werden kann, sind symmetrische Mehrprozesserverarbeitungen (Symmetric Multiprocessing, SMP) weder gut für die skalierbare Leistung noch für die Ausfallsicherheit des Systems. Eine massiv parallele Verarbeitung (Massively Parallel Processing, MPP) ist dagegen für die skalierbare Leistung gut. Eine gute Kombination zwischen skalierbarer Leistung und Ausfallsicherheit bietet ein Clustersystem. Ein anderes Extremum sind fehlertolerante Systeme. Diese Systeme wurden auf Ausfallsicherheit ausgelegt. [Hwaoo]

2.4.6. Quantitative Zuverlässigkeitsanalyse

Mittels einer quantitativen Zuverlässigkeitsanalyse können Prognosen aufgestellt werden, wie lange zum Beispiel die Lebensdauer eines Systems ist. Es gibt eine Vielzahl an Wahrscheinlichkeitsverteilungen. Auf die Exponentialverteilung und die Weibull-Verteilung soll etwas genauer eingegangen werden.

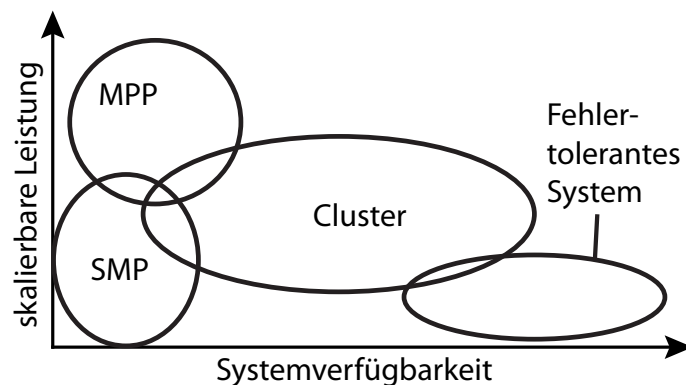


Abbildung 2.11.: Skalierbare Leistung im Vergleich zur Systemverfügbarkeit (nach [Hwa00])

Exponentialverteilung

Die Exponentialverteilung kann gut verwendet werden um das Verhalten einer konstanten Ausfallrate zu modellieren. Die Wahrscheinlichkeitsdichtefunktion wird durch die Funktion

$$f(t) = \begin{cases} \lambda \exp(-\lambda t), & \text{für } t \geq 0 \\ 0, & \text{sonst} \end{cases} \quad (2.5)$$

bestimmt. Die Variable t ist die zeitliche Veränderliche. Die konstante Ausfallrate wird durch λ und die MTTF mit $1/\lambda$ angegeben. Die Wahrscheinlichkeit, dass bis zum Zeitpunkt t kein Ausfall aufgetreten ist, berechnet sich durch

$$R(t) = 1 - \int_0^t f(t)dt = \exp(-\lambda t) \quad (2.6)$$

$R(t)$ wird auch als Zuverlässigkeitsfunktion bezeichnet.

Bei reparierbaren Systemen ist λ die Ausfallrate und $1/\lambda$ ist die mittlere Zeit zwischen zwei Ausfällen (MTBF).[O'C90]

Weibull-Verteilung

Die Weibull-Verteilung ist eine anpassungsfähige Verteilung. Durch den Parameter β und den Parameter der charakteristischen Lebensdauer η kann die Verteilung angepasst werden. Die Wahrscheinlichkeitsdichtefunktion der Weibull-Verteilung ist

$$f(t) = \begin{cases} \frac{\beta}{\eta^\beta t^{\beta-1}} \exp[-(t/\eta)^\beta], & \text{für } t \geq 0 \\ 0, & \text{sonst} \end{cases} \quad (2.7)$$

Die Zuverlässigkeit lässt sich durch

$$R(t) = \exp[-(t/\eta)^\beta] \quad (2.8)$$

berechnen. Die Ausfallrate ist mittels

$$\lambda = \frac{\beta}{\eta^\beta} t^{\beta-1} \quad (2.9)$$

zu bestimmen. Falls $\beta = 1$ ist, so erhält man eine konstante Ausfallrate. Dies ist eine besondere Version der Weibull-Verteilung, denn sie ist identisch mit der Exponentialverteilung. Somit ist dann die mittlere Lebensdauer $\eta = 1/\lambda$. Wenn $\beta < 1$ ist, so erhält man eine abfallende Ausfallrate. Sollte $\beta > 1$ sein erhält man eine ansteigende Ausfallrate. [O'C90]

2.4.7. Zuverlässigkeitsbetrachtungen für redundante Systeme

Ein System, das aus zwei statistisch unabhängigen Komponenten besteht, kann unterschiedlich angeordnet werden. Dadurch erhält man entsprechend unterschiedliche Wahrscheinlichkeiten für das System.

Serienverschaltung

In der ersten Variante sind die Komponenten in Serie verschaltet. Diese Verschaltung ist in der Abbildung 2.12 zu sehen. Die Zuverlässigkeit dieses Systems für konstante Ausfallraten erhält man durch

$$R_1 R_2 = \exp[-(\lambda_1 + \lambda_2)t] \quad (2.10)$$

Verallgemeinert auf n statistisch unabhängige in Serie verschaltete Komponenten erhält man

$$R = \prod_{i=1}^n R_i \quad (2.11)$$



Abbildung 2.12.: Serienschaltung der Komponenten R_1 und R_2 (nach [O'C90])

Parallelverschaltung

Eine andere Variante ist die Parallelverschaltung (siehe Abbildung 2.13). Durch diese Art der Verschaltung erhält man außerdem noch eine Redundanz.

Heiße Redundanz

Wenn beide Komponenten benutzt werden, so nennt man dies heiße Redundanz. Es kann

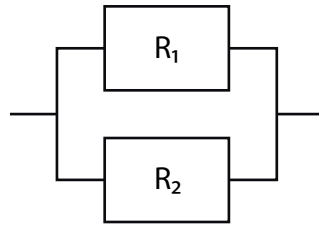


Abbildung 2.13.: Parallelverschaltung der Komponenten R_1 und R_2 (nach [O'CG90])

eine Komponente ausfallen und das System funktioniert immer noch problemlos. Die Zuverlässigkeit für ein solches System berechnet sich durch

$$(R_1 + R_2) = R_1 + R_2 - R_1R_2 = 1 - (1 - R_1)(1 - R_2) \quad (2.12)$$

Für konstante Ausfallraten gilt

$$R = \exp(-\lambda_1 t) + \exp(-\lambda_2 t) - \exp[-(\lambda_1 + \lambda_2)t] \quad (2.13)$$

Verallgemeinert für ein System mit n Komponenten, die alle parallel verwendet werden, gilt

$$R = 1 - \prod_{i=1}^n (1 - R_i) \quad (2.14)$$

M-aus-n Redundanz

In manchen Systemen ist es notwendig, dass mindestens m der n statistisch unabhängigen Komponenten funktionsfähig sind. Dies führt zu einer Zuverlässigkeit, die mittels der Gleichung

$$R = 1 - \sum_{i=0}^{m-1} \binom{n}{i} R^i (1 - R)^{n-i} \quad (2.15)$$

berechnet werden kann.

Dreifach modulare Redundanz

Bei der dreifach modularen Redundanz (TMR) werden, wie in der Abbildung 2.14 zu sehen, die Ergebnisse von drei verschiedenen IP-Cores verglichen. Dabei entscheidet der Mehrheitsentscheider welches Ergebnis weitergegeben wird. Wären es nur zwei IP-Cores, kann man erkennen, dass ein Fehler aufgetreten ist. Jedoch kann nicht herausgefunden werden, bei welchem IP-Core dieser Fehler aufgetreten ist. Deshalb werden drei IP-Cores verwendet. Bei einem System, bei dem immer drei aus zwei IP-Cores richtig sein müssen, berechnet sich die Wahrscheinlichkeit durch

$$\begin{aligned} R_{\text{TMR}} &= R_{\text{Mehrheitsentscheider}} [3(R_{\text{IP-Core}})^2 - 2(R_{\text{IP-Core}})^3] \\ &= \exp(-\lambda_{\text{Mehrheitsentscheider}} t) [3 \exp(-2\lambda_{\text{IP-Core}} t) - 2 \exp(-3\lambda_{\text{IP-Core}} t)] \end{aligned} \quad (2.16)$$

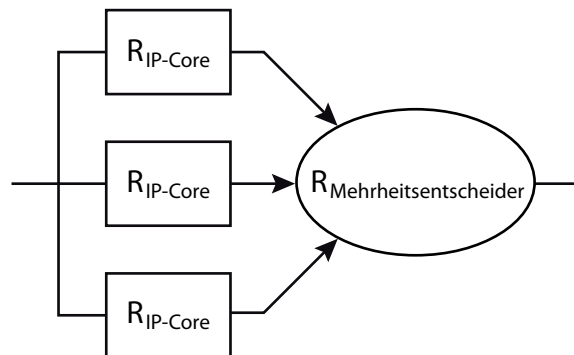


Abbildung 2.14.: Dreifach modulare Redundanz (TMR) (nach [Hed84])

Kalte Redundanz

Bei einem System mit einer kalten Redundanz gibt es eine primäre Komponente und eine Komponente, die verwendet werden kann, falls die primäre Komponente ausfällt. Durch die Reservekomponente erhält man ebenfalls ein redundantes System. Bei einem nicht gewarteten System mit der gleichen Ausfallrate je Komponente erhält man die Zuverlässigkeit

$$R = \exp(-\lambda t) + \lambda t \exp(-\lambda t) \quad (2.17)$$

Verallgemeinert man diese, so erhält man für ein System mit n gleichen Komponenten in einer redundanten Anordnung bei einer kalten Redundanz eine Zuverlässigkeit von

$$R = \sum_{i=0}^{n-1} \frac{(\lambda t)^i}{i!} \exp(-\lambda t) \quad (2.18)$$

2.4.8. Modulare Bauweise

Die Kosten können durch eine modulare Bauweise gesenkt werden. Denn teilweise können Komponenten aus anderen Projekten wieder verwendet werden. Außerdem wird dadurch auch die Zuverlässigkeit des Systems erhöht. Durch die Verwendung von Komponenten aus anderen Projekten sind diese schon gut getestet. Ein Beispiel aus der Industrie sind Speicherkomponenten oder Verarbeitungseinheiten. [O'C90]

Zugrunde liegende Architektur des NoCs

3.1. Überblick auf die zugrunde liegende Architektur des NoCs

Das bearbeitete NoC ist ein zweidimensionales Gitter. An jedem Knotenpunkt des Gitters befindet sich ein Switch. Jeder dieser Switches ist an eine Netzwerkschnittstelle (NI) angebunden. Jede dieser NI hat eine Verbindung zu einem IP-Core. Dieser Aufbau ist in der Abbildung 3.1 dargestellt. Die Verbindung zwischen den Komponenten sind zwei unidirektionale Verbindungen.

3.2. Flittypen

Das zuvor schon implementierte Ausgangsprotokoll ist ein paketbasiertes Protokoll mit fester Länge je Paketeil (Flit). Die Größe eines Flits kann nach eigenen Bedürfnissen angepasst werden. Dabei ist die Anzahl der Flits nicht begrenzt. Somit gibt es ein Kopfflit, Datenflits und ein Endflit. Bei jedem Flit wird durch die zwei höchstwertigsten Bits bestimmt, was für eine Art von Flit es ist. Aus der Tabelle 3.1 ist ersichtlich, was der jeweiligen Bitkombination zugeordnet ist.

Tabelle 3.1.: Flittypen (Ausgangssystem)

Bitkombination der Steuerbits	Funktion
00	Datenflit
01	Kopfflit
10	Endflit
11	ungenutzt

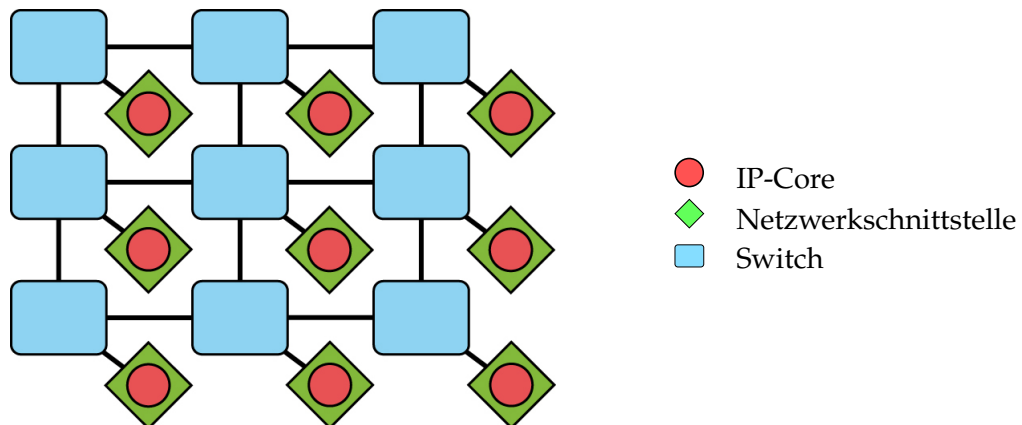


Abbildung 3.1.: Zweidimensionales regelmäßiges Gitter mit angeschlossenen IP-Cores (nach [Chao5])

3.2.1. Kopfflit

Im Kopfflit ist außerdem die Zieladresse enthalten. Diese ist in die x-Koordinate und die y-Koordinate aufgeteilt. In der Abbildung 3.2 ist das Kopfflit zu sehen.

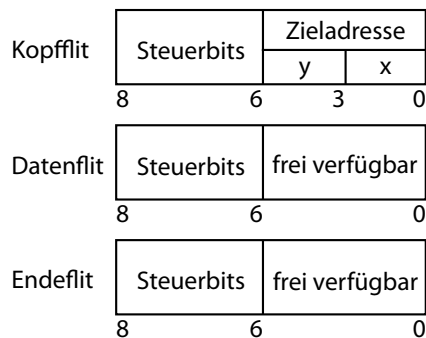


Abbildung 3.2.: Flittypen des Ausgangssystems

3.2.2. Datenflit

Die Anzahl der Datenflits ist beliebig groß. Dabei ist die Anzahl der frei verfügbaren Bits so groß wie die Anzahl der Adressbits. Die Abbildung 3.2 zeigt die Struktur eines Datenflits.

3.2.3. Endflit

Das Endflit schließt das Paket ab. Dabei stehen die restlichen Bits des Flits, außer den Steuerbits, frei zur Verfügung (Abbildung 3.2). [Kir12]

3.3. Switch

In dieser Arbeit wird auf einen am Institut bereits vorhandenen einfachen Switch zurückgegriffen. Der Aufbau dieses Switches ist in der Abbildung 3.3 zu sehen. Dieser Switch verfügt über fünf Anschlüsse (Ports). Der Port null ist mit der NI, respektive dem IP-Core verbunden. Die Ports eins bis vier sind mit den nebenliegenden Switches verbunden. Jeder der Ports besitzt einen Eingangspufferspeicher (Eingang-FIFO), um das fluktuierende Paketaufkommen kompensieren zu können. Im Router wird entschieden, an welchen Port das Paket weitergeleitet wird. Diese Entscheidung wird an den Steuereingang des Multiplexers (MUX) weitergegeben. An diesen MUX sind die Ausgänge aller Eingang-FIFOs angeschlossen.

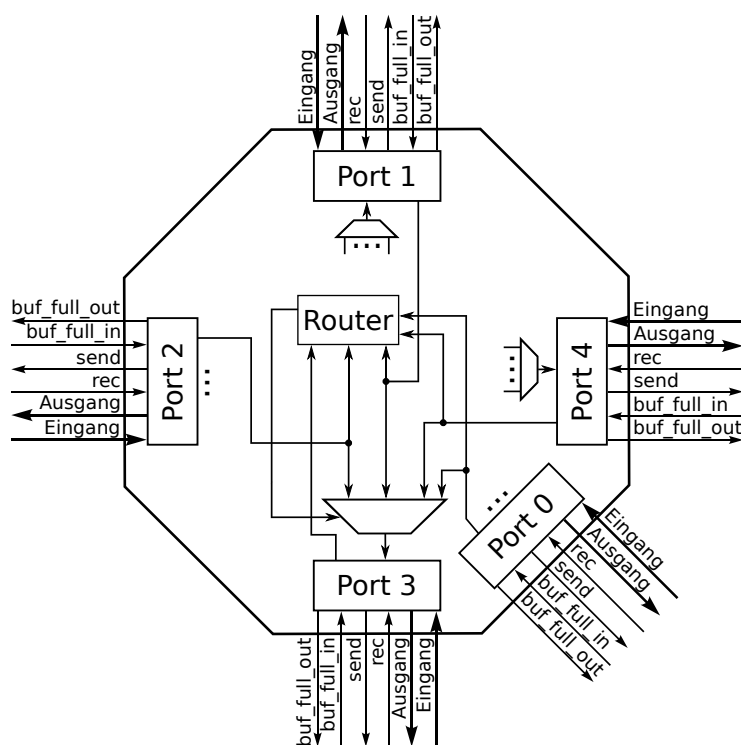


Abbildung 3.3.: Architektur des Switches (ohne Erweiterungen) (nach [Hoso7])

3.3.1. Router

Der Router folgt dem xy-Wurmlochrouting [Dal87a]. Dies heißt, dass durch das Kopfflit festgelegt wird, welcher der Ports verwendet wird. Der Eingangsteil des Ports, an dem das Paket ankommt, und der Ausgangsteil des Ports, an den das Paket gesendet wird, werden in der Zeit der Paketübertragung blockiert. Die Datenflits und das Endeflit folgen dem Pfad des Kopfflits. Das Endeflit gibt die jeweiligen Portteile wieder frei. Bei dem

xy-Routing wird das Paket erst in die gewünschte x-Koordinate geroutet und dann erst in gewünschte y-Koordinate geroutet (Abbildung 3.4). Sofern kein Defekt auftritt, ist der xy-Routingalgorithmus deadlockfrei [Dal87b].

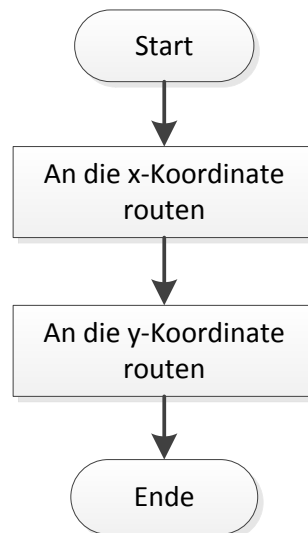


Abbildung 3.4.: Xy-Routing ohne Fehlertoleranz oder das Löschen von Paketen (nach [Dal87a])

Der Router kann, sofern der gewünschte Eingangsteil des Ports und der gewünschte Ausgangsteil des Ports noch nicht durch ein Routing eines anderen Paketes blockiert ist, mehrere Pakete gleichzeitig routen.

3.3.2. Paketaufkommen im NoC

Bei dem xy-Routingalgorithmus ist der Datenverkehr in der Mitte des NoC am größten. Dies ist schon bei einem kleinen NoC mit einem 3x3-Switchgitter ersichtlich. In der Abbildung 3.5 wird an jeden Port gezählt wie viele Pakete dort eintreffen. Bei den Switchen ist zusätzlich noch die Gesamtzahl der eintreffenden Pakete an diesem Switch vermerkt. Bei dieser Simulation sendet jeder IP-Core an jeden anderen IP-Core ein Paket.

3.4. Netzwerkschnittstelle

Die NI stellt die Verbindung zwischen dem IP-Core mit dem Switch, respektive dem Netzwerk, her. Außerdem fügt die NI die Adresse des IP-Cores, respektive des Switches, hinzu. Die NI verfügt über eine zentrale Steuereinheit (Controller). Dieser Controller leitet

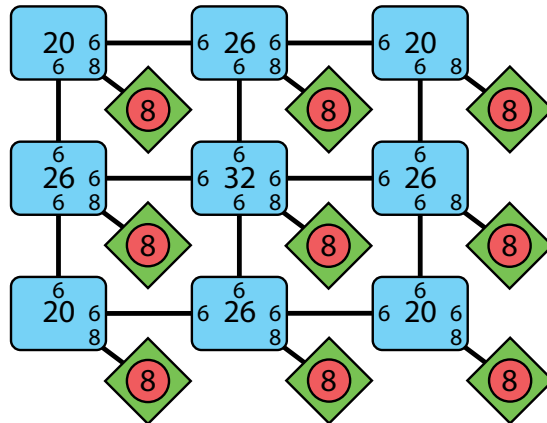


Abbildung 3.5.: Anzahl der eintreffenden Pakete bei einem defektfreien NoC

die Pakete die von dem Switch kommen an den IP-Core weiter. Ebenso leitet er auch die Pakete von dem IP-Core an den Switch weiter. Beim Senden eines Paketes wird immer geschaut ob der Ziel-FIFO voll ist, sollte er voll sein wird das Paket erst später weitergeleitet. Außerdem ist jedem Controllerport ein FIFO vorgeschaltet. Die NI ist der Abbildung 3.6 dargestellt.

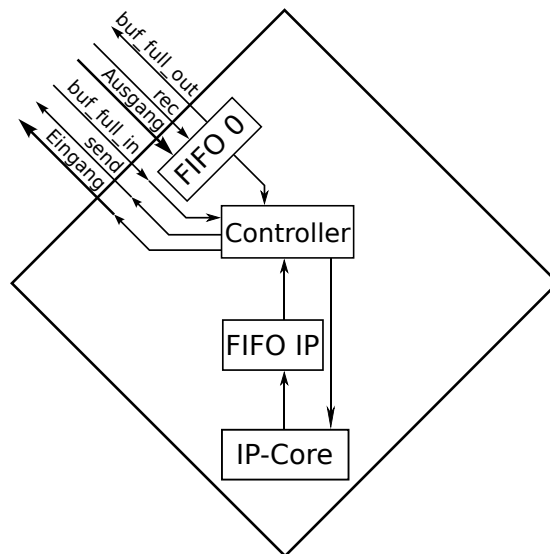


Abbildung 3.6.: Architektur der Netzwerkschnittstelle (ohne Erweiterungen)

Numerische Bestimmung der Erreichbarkeit des Systems

Mittels numerischer Simulationen wird die Erreichbarkeit bestimmt, was mit den Erweiterungen der Architektur maximal möglich ist. Diese Erweiterungen werden für das implementierte System auf Gatterebene, ab dem Kapitel 6, genauer erklärt. Dazu wird in den folgenden Simulationen festgestellt, wie viele der IP-Cores noch zu erreichen sind. Dabei laufen die Simulationen unabhängig vom Routingalgorithmus ab. Bei den Simulationen gibt es unterschiedliche Arten von Ausfällen, für die jeweils separate Simulationen durchgeführt wurden. Hintergrund ist, dass bei den Simulationen einzelne, unidirektionale oder bidirektionale, Verbindungen oder gesamte Komponenten ausfallen können. Welche Verbindung oder welche Komponente ausfällt, wird zufällig bestimmt. Ebenso wird darauf geachtet, dass die Verteilung, welche Verbindung oder Komponente ausfällt, entsprechend der vorkommenden Anzahl ist. Zum Beispiel ist die Zahl der Switch-Switch-Verbindungen anders als die Anzahl der Switch-IP-Core-Verbindungen.

Der Suchalgorithmus, welche IP-Cores noch zu erreichen sind, ist bei den Simulationen immer der gleiche. Mit der Einschränkung, dass wenn die Komponenten defekt werden können, dass dann diese untersucht werden. Wenn die Verbindungen zwischen den Komponenten defekt werden können, dann werden diese untersucht. Dabei wird von jedem IP-Core aus untersucht, zu wie vielen anderen IP-Cores noch eine funktionsfähige Verbindung besteht. Wenn nun die entsprechende Komponente, beziehungsweise die Verbindung, defekt ist, wird dort nicht genauer weitergesucht. Sollte sie jedoch intakt sein, wird der Switch als zu besuchend markiert. Von dort aus startet dann wieder eine Untersuchung zu den benachbarten Switchen. Wenn nun ein IP-Core erreicht wurde, dann wird dieser als erreichbar markiert. Nach dem Suchlauf wird gezählt, wie viele IP-Cores als erreichbar markiert sind. Der Algorithmus ist auch in der Abbildung 4.1 zu sehen.

Um diese Simulationsergebnisse mit den Ergebnissen für das implementierte System vergleichen zu können (Kapitel 10), werden Simulationen mit einer Gittergröße von 3×3 Switchen untersucht. Des Weiteren werden noch unter anderem 20×20 Gitter untersucht. Dadurch kann man sehen, wie sich ein entsprechendes System bei einer größeren Dimensionierung verhält.

4. Numerische Bestimmung der Erreichbarkeit des Systems

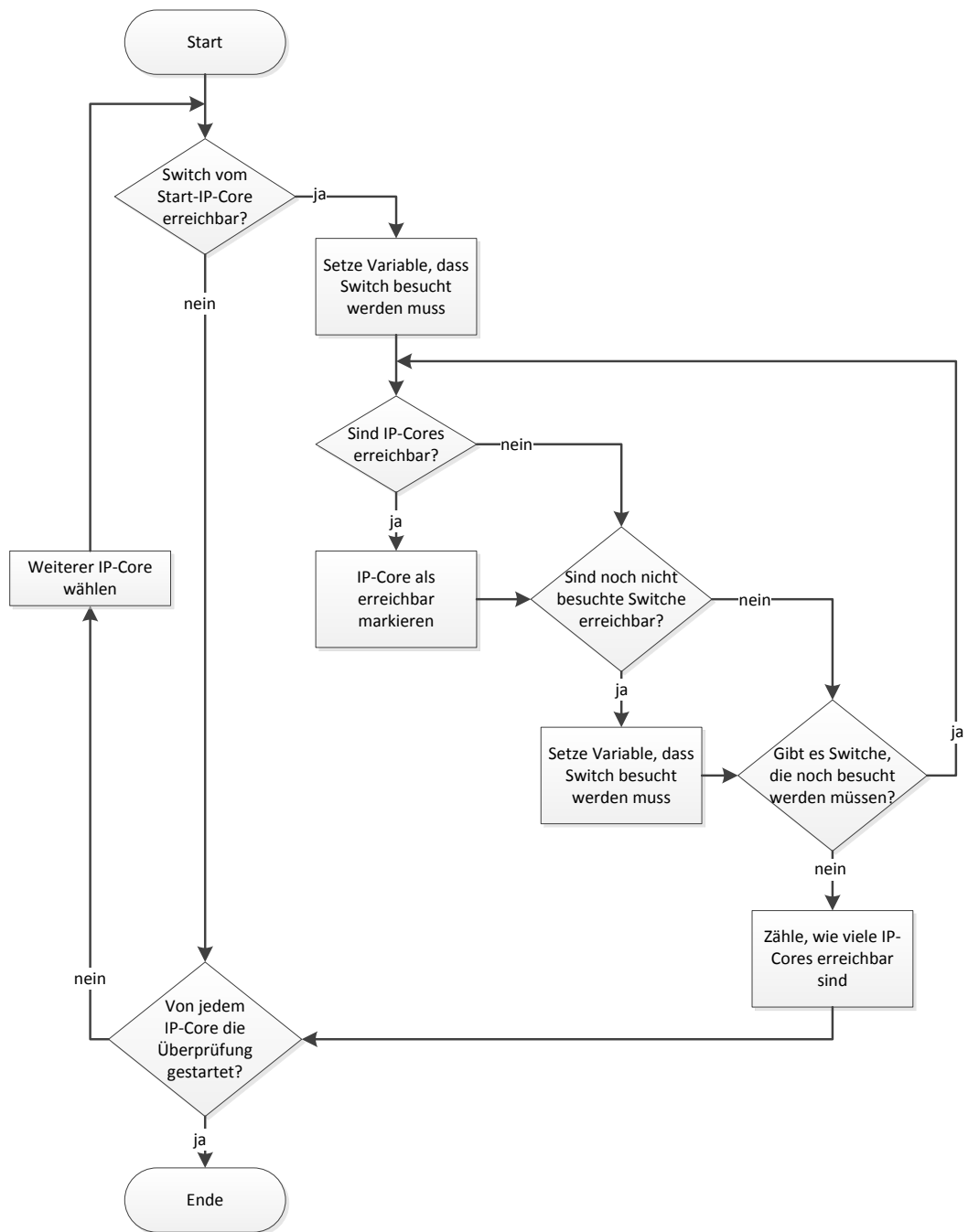


Abbildung 4.1.: Algorithmus zum Bestimmen der globalen Erreichbarkeit (numerische Simulation)

Bei den Abbildungen handelt es sich bei der Abszissenachse um den prozentualen Anteil der defekten Verbindungen respektive den defekten Komponenten. Bei der Ordinatenachse handelt es sich um den prozentuellen Anteil der noch erreichbaren IP-Cores nach dem oben beschriebenen Algorithmus. Es ist zu beachten, dass bei einer Simulation, bei der alle Verbindungen ausfallen können und eine redundante Anbindung der IP-Cores vorhanden ist, die Anzahl der Verbindungen, die defekt werden können, nicht konstant ist. Das liegt daran, dass ein System mit einer vierfachen Anbindung der IP-Cores an die Switche eine höhere Anzahl an Verbindungen im Vergleich zu einem System mit einer einfachen Switch-IP-Core-Verbindung hat.

4.1. Erreichbarkeit eines NoCs mit redundanten Anbindungen

Wie die Erreichbarkeit bei einem NoC ist, bei dem die IP-Cores über ein NI mehrfach an Switche angebunden sind, wird in diesem Abschnitt untersucht.

4.1.1. Bei Ausfällen von gesamten Switchen oder IP-Cores

Bei zufälligen Ausfällen von gesamten Switchen oder IP-Cores erhält man für ein 3×3 -Gitter mit 500 Wiederholungen das Resultat aus Abbildung 4.2. Wenn diese Simulation für ein 20×20 -Gitter mit 100 Wiederholungen durchgeführt wird, so erhält man als Ergebnis die Abbildung 4.3. Eine weiteres Simulationsergebnis für ein 5×5 -Gitter ist im Anhang A.1 zu finden.

Aus den Simulationen ist ersichtlich, dass bei einer doppelten Anbindung im Vergleich zu einer einfachen Anbindung der NIs an die Switche erst ab einem Ausfall von zirka 20 % der Komponenten eine merkliche Verbesserung auftritt. Wenn es eine dreifache oder vierfache Anbindung der NIs an die Switche gibt, ist ein Verbesserung der Erreichbarkeit im gesamten Bereich zu sehen. Eine weitere Verbesserung ist zwischen der dreifachen und der vierfachen Anbindung, wenn nur gesamte Komponenten ausfallen, zu beachten.

Wenn 20 % der Komponenten defekt sind, dann ist der Unterschied zwischen der einfachen und der vierfachen Anbindung der IP-Cores bei einem 3×3 -Gitter 13 Prozentpunkte und bei einem 20×20 -Gitter 15 Prozentpunkte.

4.1.2. Bei Ausfällen von Switch-Switch-Verbindungen

Wenn nur die Switch-Switch-Verbindungen ausfallen erhält, man bei dem 3×3 -Gitter und 500 Wiederholungen die Ergebnisse in der Abbildung 4.4 und in der Abbildung 4.5. Bei einem 20×20 -Gitter und 100 Wiederholungen erhält man bei dieser Simulation die Resultate aus der Abbildung 4.6 und der Abbildung 4.7. Diese Simulation ist ebenso für ein 5×5 -Gitter durchgeführt worden, die Ergebnisse sind in dem Anhang A.2 zu finden.

4. Numerische Bestimmung der Erreichbarkeit des Systems

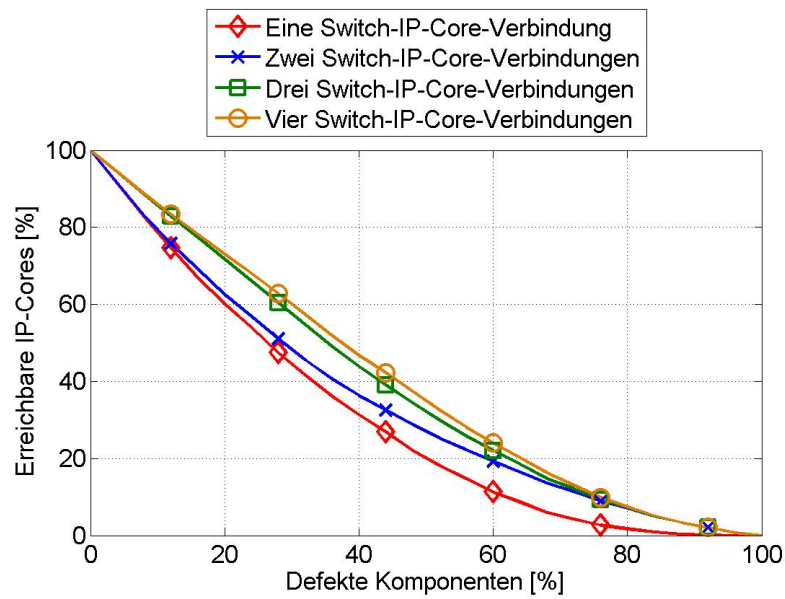


Abbildung 4.2.: Numerische Simulation eines 3×3 -Gitters mit redundanten Anbindungen bei dem gesamte Komponenten ausfallen

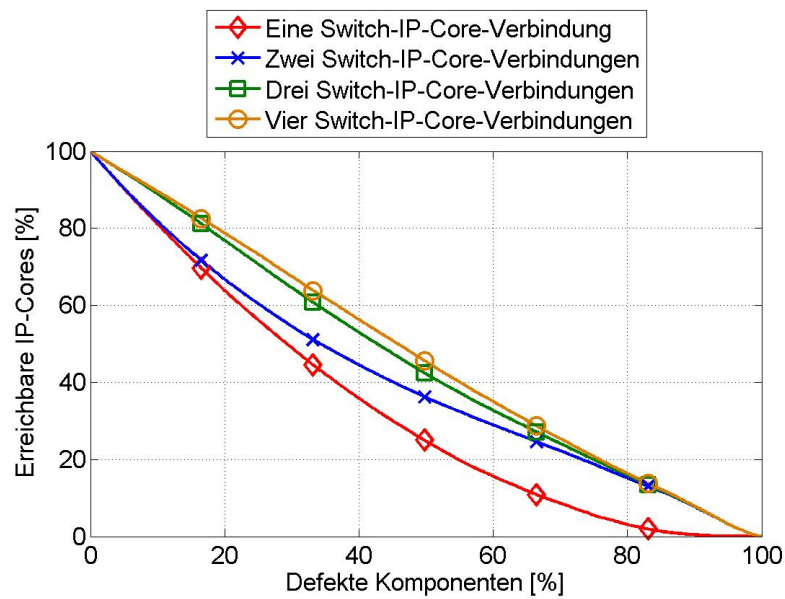


Abbildung 4.3.: Numerische Simulation eines 20×20 -Gitters mit redundanten Anbindungen bei dem gesamte Komponenten ausfallen

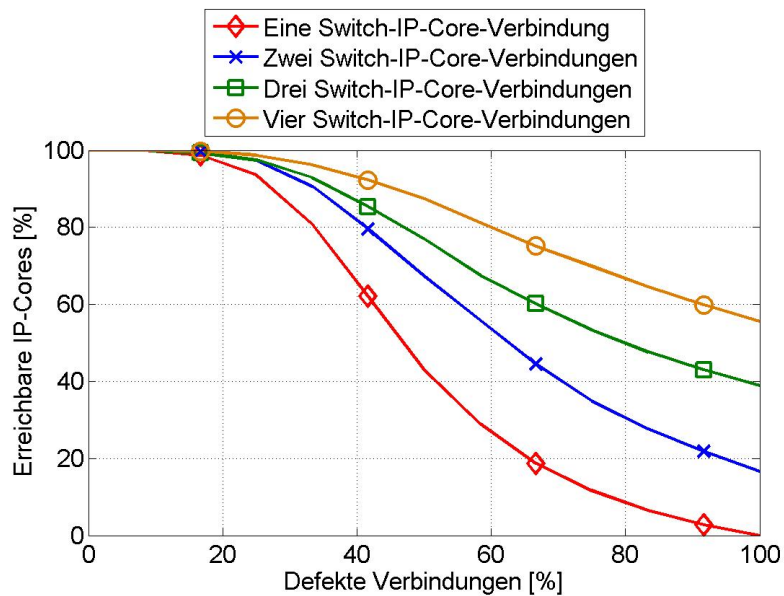


Abbildung 4.4.: Numerische Simulation eines 3x3-Gitters mit redundanten Anbindungen bei dem bidirektionale Switch-Switch-Verbindungen ausfallen

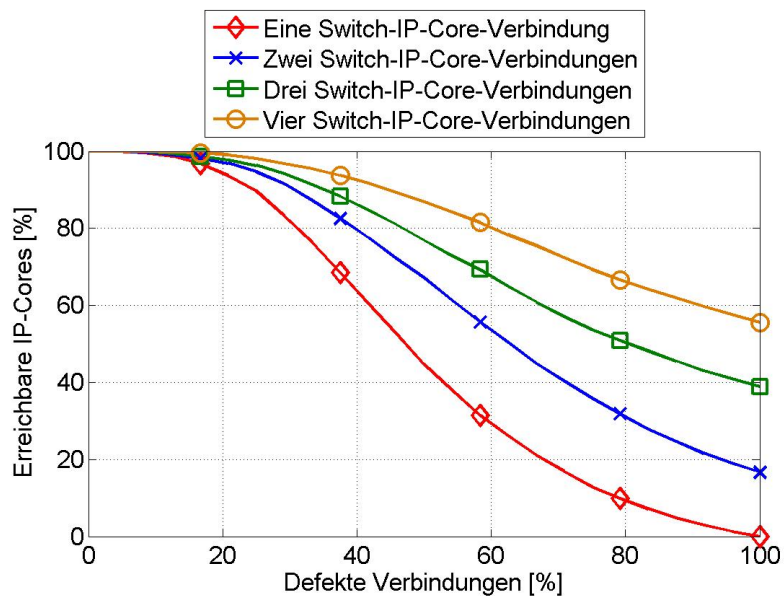


Abbildung 4.5.: Numerische Simulation eines 3x3-Gitters mit redundanten Anbindungen bei dem unidirektionale Switch-Switch-Verbindungen ausfallen

4. Numerische Bestimmung der Erreichbarkeit des Systems

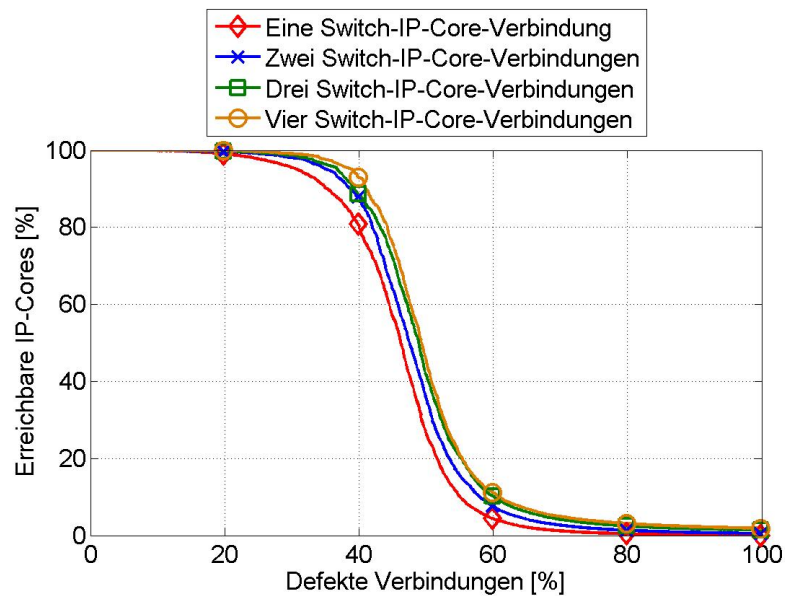


Abbildung 4.6.: Numerische Simulation eines 20x20-Gitters mit redundanten Anbindungen bei dem bidirektionale Switch-Switch-Verbindungen ausfallen

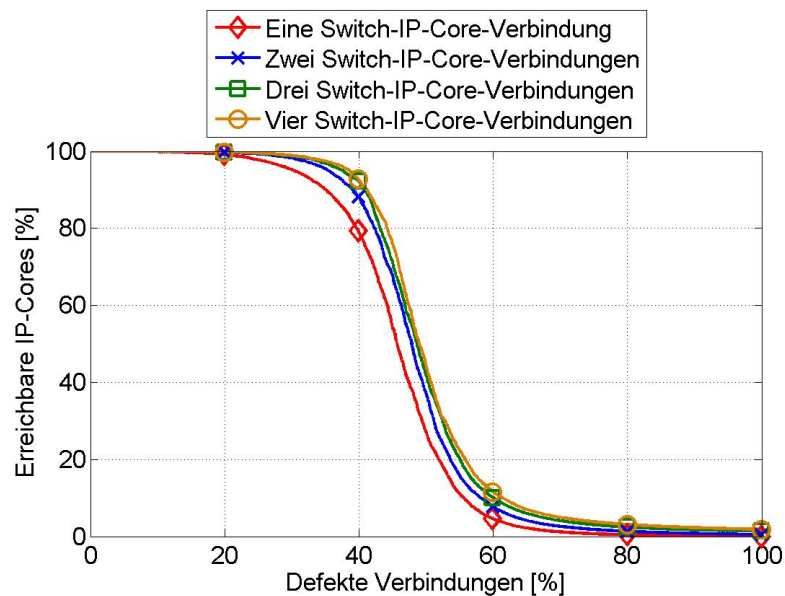


Abbildung 4.7.: Numerische Simulation eines 20x20-Gitters mit redundanten Anbindungen bei dem unidirektionale Switch-Switch-Verbindungen ausfallen

Wenn außer Betracht gelassen wird, dass die Anzahl der Verbindungen bei einer Simulation mit unidirektionalen Verbindungen deutlich höher ist als die Anzahl der Verbindungen bei einer Simulation mit bidirektionalen Verbindungen, sind die Unterschiede bei den Simulationen relativ gering.

Bei einem Ausfall aller Switch-Switch-Verbindungen enden die Kurven nicht in dem gleichen Punkt. Der Grund dafür ist, dass an einem Switch teilweise mehrere IP-Cores angebunden sind und die Switch-IP-Core-Verbindungen nicht ausfallen. Wie viele IP-Cores bei einem 3x3-Gitter noch zu erreichen sind, wenn der Algorithmus aus Abbildung 4.1 verwendet wird, kann der Tabelle 4.1 entnommen werden. Es ist ersichtlich, dass die Anzahl der immer noch erreichbaren IP-Cores nicht linear steigt.

Bei der Simulation mit einem 20x20-Gitter ist ab zirka 40 % ein starker Abfall sichtbar. Dieser Abfall ist auf die Partitionierung des Netzwerkes in einzelne Gruppen von Komponenten zurückzuführen.

4.1.3. Bei Ausfällen von Switch-Switch- und Switch-IP-Core-Verbindungen

Fallen die Switch-Switch-Verbindungen und die Switch-IP-Core-Verbindungen aus, dann erhält man für das 3x3-Gitter und 500 Wiederholungen die Resultate aus der Abbildung 4.8 und der Abbildung 4.9. Wenn diese Simulation für das 20x20-Gitter und 100 Wiederholungen ausgeführt wird, dann ergeben sich die Ergebnisse aus der Abbildung 4.10 und der Abbildung 4.11. Diese Simulation ist ebenso für ein 5x5-Gitter durchgeführt, die Simulationsergebnisse sind im Anhang A.3 zu finden.

Zwischen den Simulationen mit unidirektionalen Verbindungen und den Simulationen mit bidirektionalen Verbindungen hat, bei einer Gittergröße von 3x3, die Simulation mit bidirektionalen Verbindungen immer eine zwischen zwei und vier Prozentpunkte höhere Erreichbarkeit. Bei den Simulationen des 20x20-Gitters ist jedoch kein Unterschied zu sehen.

Tabelle 4.1.: Erreichbarkeit der IP-Cores nach Ausfall von allen Switch-Switch-Verbindungen bei einem 3x3-Gitter (mit dem Algorithmus aus Abbildung 4.1)

Anzahl der Switch-IP-Core-Verbindungen	Anzahl der noch erreichbaren IP-Cores	Prozentueller Anteil der noch erreichbaren IP-Cores
1	0	0
2	12	16,67
3	28	38,89
4	40	55,56

4. Numerische Bestimmung der Erreichbarkeit des Systems

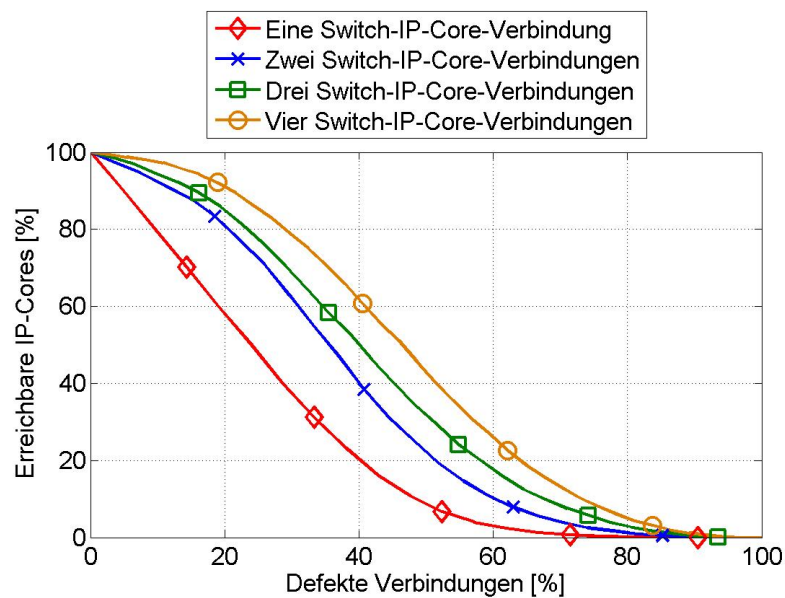


Abbildung 4.8.: Numerische Simulation eines 3×3 -Gitters mit redundanten Anbindungen bei dem alle bidirektionalen Verbindungen ausfallen

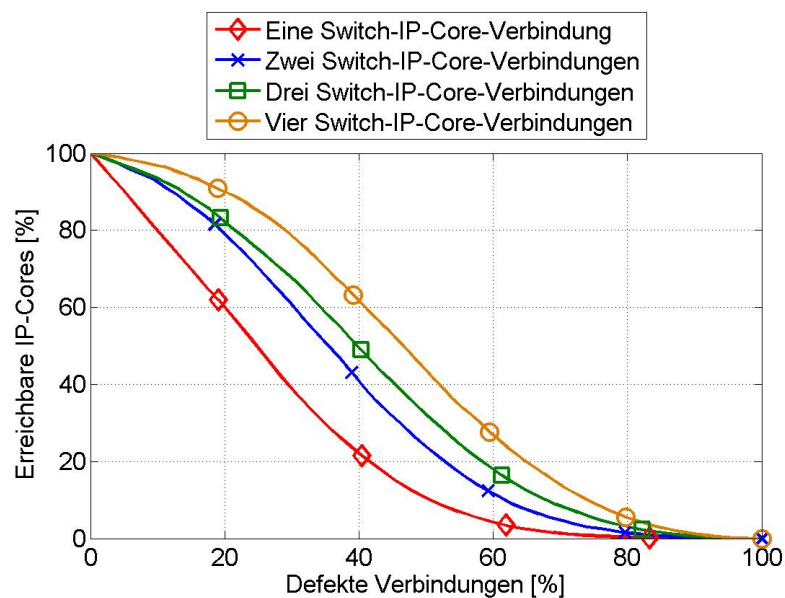


Abbildung 4.9.: Numerische Simulation eines 3×3 -Gitters mit redundanten Anbindungen bei dem alle unidirektionalen Verbindungen ausfallen

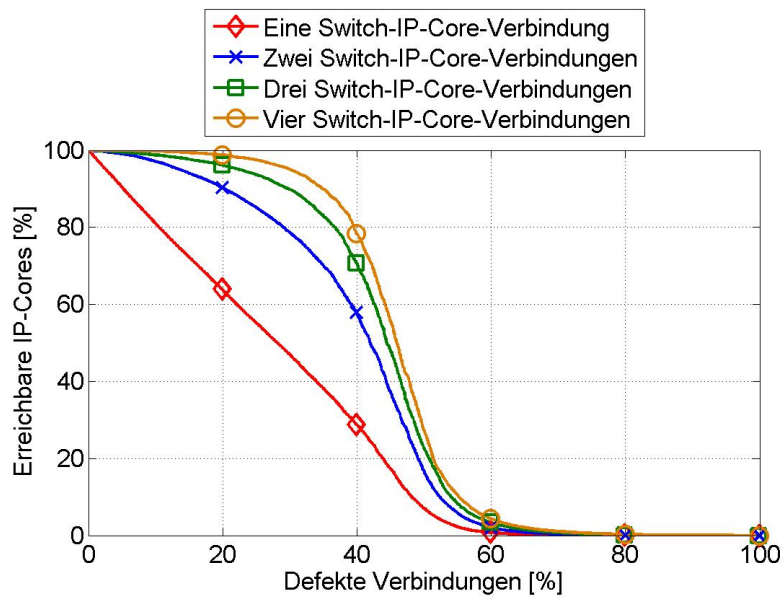


Abbildung 4.10.: Numerische Simulation eines 20x20-Gitters mit redundanten Anbindungen bei dem alle bidirektionalen Verbindungen ausfallen

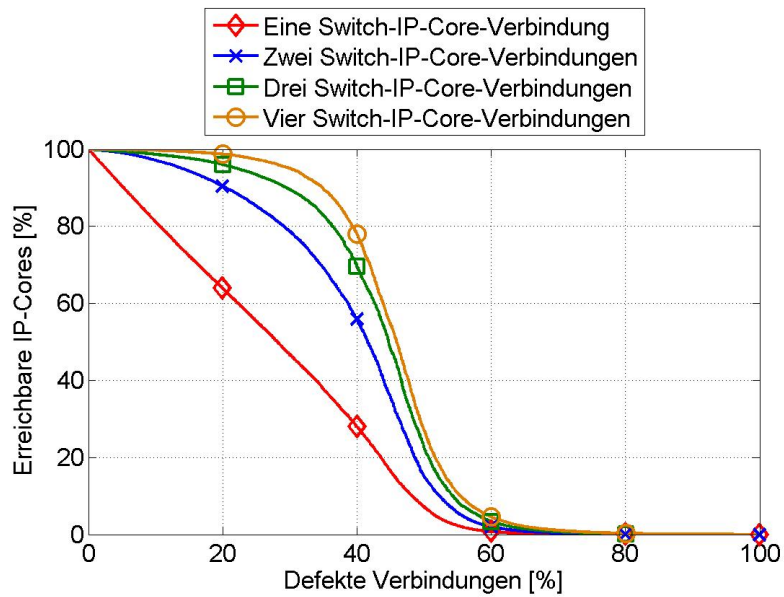


Abbildung 4.11.: Numerische Simulation eines 20x20-Gitters mit redundanten Anbindungen bei dem alle unidirektionalen Verbindungen ausfallen

Bei diesen Simulationen, bei denen alle Switch-Switch- und Switch-IP-Core-Verbindungen ausfallen, fällt die Kurve mit einer Switch-IP-Core-Verbindung zu Beginn immer linear ab. Dies liegt daran, dass, sobald eine der beiden unidirektionalen Switch-IP-Core-Verbindungen bei einem IP-Core defekt ist, der IP-Core nicht mehr senden beziehungsweise empfangen kann. Wenn es bidirektionale Verbindungen sind, dann kann nicht mehr gesendet und nicht mehr empfangen werden. Zu Beginn ist die Partitionierung der einzelnen IP-Cores noch nicht relevant. Bei einer einfachen Switch-IP-Core-Verbindungen ist ab zirka 40 % zu erkennen, dass der lineare Abfall in einen exponentiellen Abfall übergeht. Dieser Übergang ist durch die Partitionierung der Komponenten zu erklären.

Bei der Simulation mit einem 20x20-Gitter ist ab zirka 40 % ein starker Abfall sichtbar. Dieser Abfall ist auf die Partitionierung in einzelne Gruppen von Komponenten zurückzuführen. Dieser Abfall ist umso extremer je mehr Switch-IP-Core-Verbindungen vorhanden sind.

Sind 20 % der Verbindungen defekt, dann ist zwischen der einfachen und der vierfachen Anbindung für das 3x3-Gitter und bidirektionalen Verbindungen ein Unterschied von 34 Prozentpunkten und für die unidirektionalen ist ein Unterschied von 29 Prozentpunkten zu sehen. Der Unterschied bei den 20x20-Gittern beträgt 35 Prozentpunkte.

4.1.4. Vergleich der Simulationen, wenn nur Switch-Switch-Verbindungen bzw. Switch-Switch- und Switch-IP-Core-Verbindungen ausfallen

In diesem Abschnitt werden die Simulationen aus Abschnitt 4.1.2 und Abschnitt 4.1.3 miteinander verglichen.

Da in den zweiten Simulationen auch die Switch-IP-Core-Verbindungen defekt werden, sind beim Ausfall aller Verbindungen keine IP-Cores mehr erreichbar. Des Weiteren ist bei dem 20x20-Gitter der ersten Simulation eine deutliche Partitionierung der Komponenten zu sehen.

Aufgrund dessen, dass in den zweiten Simulationen auch die Switch-IP-Core-Verbindungen ausfallen, ist es schon deutlich früher möglich, dass ein gewünschter IP-Core nicht mehr erreichbar ist. Dieser Unterschied wird bei der einfachen Anbindung eines IP-Cores an einen Switch am stärksten sichtbar.

4.2. Erreichbarkeit eines NoCs mit redundanten Anbindungen und einer Erkennung von transienten Verbindungsdefekten

Bei den bisherigen Simulationen wurden nur permanente Fehler betrachtet. In diesem Abschnitt werden Simulationen diskutiert, bei denen neben den permanenten Fehlern auch transiente Fehler vorkommen. Dabei wird immer ein entsprechender Fehler in das simulierte System eingepreßt und dann geschaut welche IP-Cores noch zu erreichen sind. Ist es eine transiente Störung gewesen, dann wird diese nun wieder entfernt. Dies heißt, eine Verbindung kann mehrmals mit einem Fehler belastet werden. Dennoch ist die Auswahl,

4.2. Erreichbarkeit eines NoCs mit redundanten Anbindungen und einer Erkennung von transienten Verbindungsdefekten

welche Verbindung defekt wird, zufällig und bei den Verbindungen handelt es sich um unidirektionale Verbindungen. Die Abszissenachse stellt weiter den prozentualen Anteil der fehlerhaften Verbindungen dar. Jedoch ist immer die Anzahl an fehlerhaften Verbindungen, wenn nur permanente Fehler vorhanden sind, gleich 100 % zu setzen. Dies heißt, wenn transiente Fehler vorhanden sind, dann ist der prozentuale Anteil höher als 100 %.

Für ein 3x3-Gitter mit 500 Wiederholungen erhält man mit einer Switch-IP-Core-Verbindung das Ergebnis aus Abbildung 4.12. Mit vier Switch-IP-Core-Verbindungen erhält man die Kurven in Abbildung 4.13.

Wenn die Simulation für ein 20x20-Gitter mit 50 Wiederholungen durchgeführt wird, erhält man mit einer Switch-IP-Core-Verbindung das Resultat aus Abbildung 4.14. Mit vier Switch-IP-Core-Verbindungen erhält man das Ergebnis in Abbildung 4.15.

Weitere Simulationsergebnisse für 3x3-, 5x5- und 20x20-Gitter sind im Anhang A.4 zu finden.

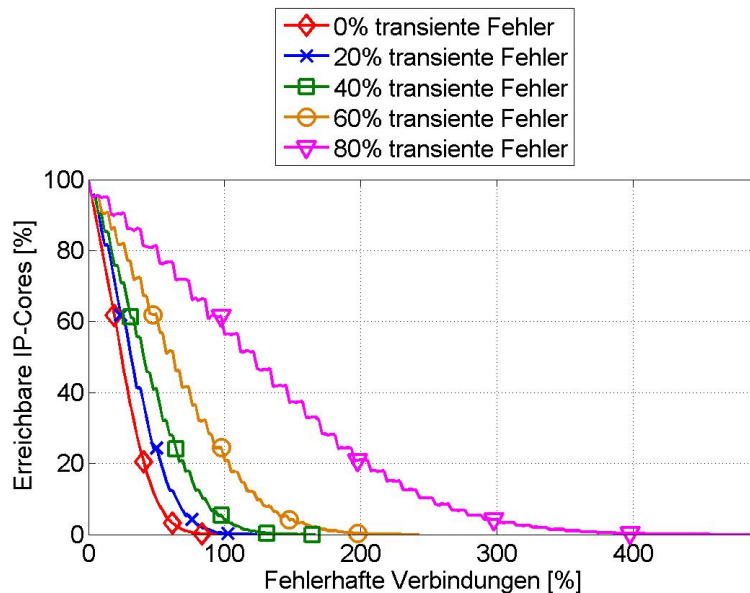


Abbildung 4.12.: Numerische Transientensimulation eines 3x3-Gitters mit einer Switch-IP-Core-Verbindung

Wenn die Anzahl der permanenten und der transienten Fehler addiert werden, ist zu erkennen, dass diese Anzahl mit dem Prozentsatz wie viele davon transiente Fehler sind exponentiell ansteigt. Hierdurch ist auch zu erklären, warum die Abstände zwischen den Kurven mit steigendem Prozentsatz immer größer werden.

An diesen Simulationen ist zu sehen, wenn alle Fehler als transiente beziehungsweise als permanente Fehler erkannt werden, dann ist das System entsprechend länger nutzbar, als wenn alle Fehler als permanent erkannt werden. Bei 20 % transienten Fehlern ist das

4. Numerische Bestimmung der Erreichbarkeit des Systems

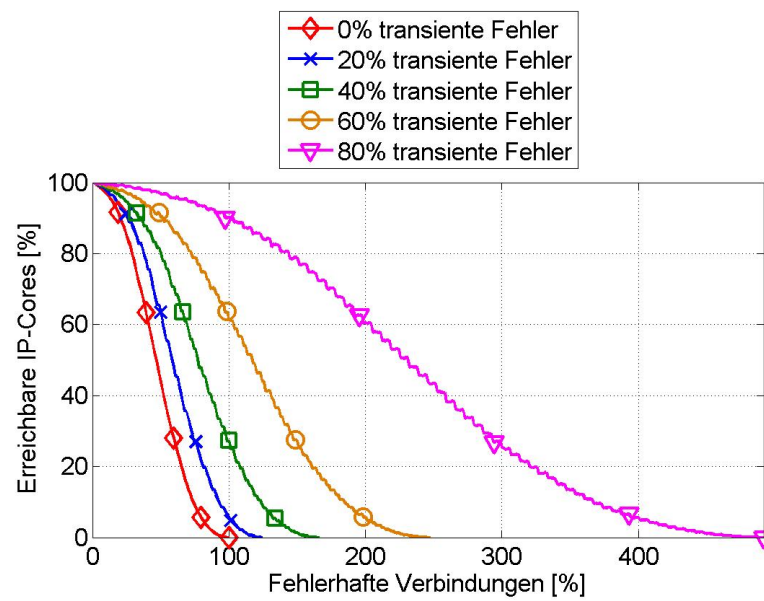


Abbildung 4.13.: Numerische Transientensimulation eines 3x3-Gitters mit vier Switch-IP-Core-Verbindungen

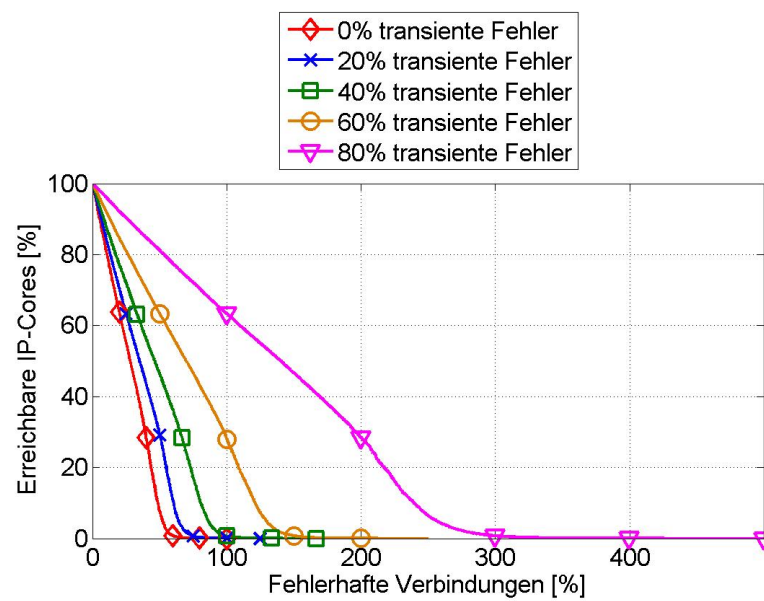


Abbildung 4.14.: Numerische Transientensimulation eines 20x20-Gitters mit einer Switch-IP-Core-Verbindung

4.2. Erreichbarkeit eines NoCs mit redundanten Anbindungen und einer Erkennung von transienten Verbindungsdefekten

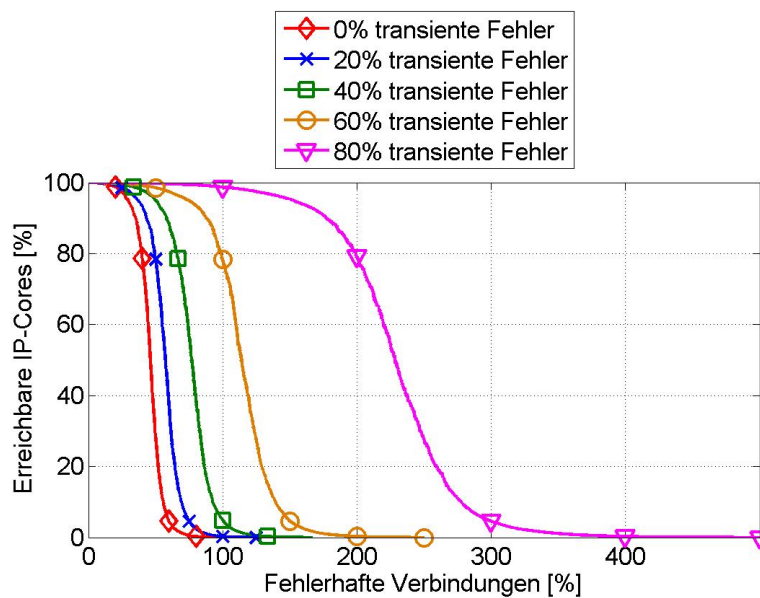


Abbildung 4.15.: Numerische Transientensimulation eines 20x20-Gitters mit vier Switch-IP-Core-Verbindungen

System um den Faktor 1,25 länger funktionsfähig. 40 % transiente Fehler ergeben einen Faktor von 1,67. Für 60 % transiente Fehler ist es ein Faktor von 2,5. Bei 80 % transiente Fehler ist es das Fünffache.

Die Unebenheiten bei dem 3x3-Gitter ist damit erklären, dass immer nach der gleichen Anzahl von permanenten Fehlern einer Simulationen ein transienter Fehler folgt. Bei dem 20x20-Gitter sind diese Unebenheiten nur noch schwach zu sehen, da die Anzahl der IP-Cores höher ist.

Numerische Abschätzung der Systemzuverlässigkeit

Wenn ein entsprechendes System verwendet wird, dann ist die zu erwartende Zuverlässigkeit des Systems von Interesse. Hierfür ist bei nicht reparierbaren Systemen der Parameter der mittlere Zeit bis zum Ausfall des Systems (MTTF) ein wichtiger Anhaltspunkt.

Die theoretischen Grundlagen der Zuverlässigkeit sind dem Abschnitt 2.4 zu entnehmen. Für die MTTF wird von vielen Simulationen aus Kapitel 4 bestimmt, nach welchem Prozentsatz von fehlerhaften Komponenten beziehungsweise Verbindungen das System gerade nicht mehr funktionsfähig ist. Diese Wahrscheinlichkeit P_f wird für den Ausfall nach 100 % der Komponenten beziehungsweise der Verbindungen ermittelt. Von ausgewählten Simulationen aus Kapitel 4 wird P_f , nach Ausfall von 20 %, 40 %, 60 %, 80 % und 100 % der Komponenten beziehungsweise der Verbindungen, ermittelt. P_f kann durch

$$P_f = \int_{\text{entsprechende Fehler}} a_p d(\text{Fehler}) \quad (5.1)$$

bestimmt werden. Dabei ist a_p der prozentuale Anteil, wie viele IP-Cores erreichbar sind. Hierbei wird nicht betrachtet, dass es unterschiedliche IP-Cores in einem NoC gibt. Aus den Angaben P_f , der Anzahl wie viele Komponenten beziehungsweise Verbindungen maximal defekt werden können D_{max} , des Prozentsatzes wie viele Fehler es von der Gesamtanzahl zu dieser Zeit sein dürfen F_p , und der Ausfallrate der Simulationen λ_{Sim} kann die $MTTF_{Sim}$ mit der Formel

$$MTTF_{Sim} = \frac{1}{\lambda_{Sim}} D_{max} F_p P_f \quad (5.2)$$

berechnet werden.

Diese Berechnungen werden für ein NoC, mit einer redundanten Anbindung der NIs, respektive der IP-Cores, an die Switche durchgeführt. Für die Ausfallrate von gesamten Komponenten beziehungsweise von einzelnen Verbindungen in den Simulationen λ_{Sim} wird exemplarisch ein Wert von 2×10^{-4} 1/h verwendet.

Anhand der Werte in der folgenden Abschnitte ist zu sehen, dass eine mehrfache Anbindung der IP-Cores an die Switche eine deutliche Erhöhung der Lebensdauer des Systems darstellt. Für einen Teil der $MTTF_{Sim}$ Werte wird der Proportionalitätsfaktor P_{MTTF} bestimmt, dieser berechnet sich durch

$$P_{MTTF} = \frac{MTTF_{Sim}(\text{vier Switch-IP-Core-Verbindungen})}{MTTF_{Sim}(\text{eine Switch-IP-Core-Verbindung})} \quad (5.3)$$

P_{MTTF} gibt an um was für einen Faktor die $MTTF_{Sim}$ der vierfachen Anbindung im Vergleich zur einfachen Anbindung besser ist. Aufgrund der folgenden Berechnungen ist davon auszugehen, dass eine Schwankung von P_{MTTF} im Bereich von $\pm 0,06$ an Simulationsschwankungen liegt.

5.1. Ausfall von gesamten Komponenten (permanente Fehler)

Wenn nur gesamte Komponenten ausfallen, dann erhält man für die $MTTF_{Sim}$ die Werte aus der Tabelle 5.1. Des Weiteren erhält man für das 3×3 -Gitter $P_{MTTF} = 1,30$ und für das 20×20 -Gitter $P_{MTTF} = 1,43$.

In der Abbildung 5.1 wird betrachtet, wie der Verlauf der Erreichbarkeit in Abhängigkeit der Anzahl der defekten Komponenten für ein 3×3 -Gitter ist. Diese Berechnung wurde ebenso für ein 20×20 -Gitter durchgeführt, das Ergebnis ist in der Abbildung 5.2 zu sehen. Der Verlauf der Erreichbarkeit bei diesen beiden Berechnungen ist, für die einfache und zweifache beziehungsweise die dreifache und vierfache Switch-IP-Core-Anbindung, bis zirka 20 % fast gleich. Dies kommt daher, dass die dazugehörigen Kurven der Numeriksimulationen in diesem Bereich einen fast identischen Verlauf aufweisen. An diesen Berechnungen ist zu sehen, dass der Verlauf der Kurve, in der dazugehörigen Numeriksimulation, mit einer zweifachen Switch-IP-Core-Anbindung sich dem Verlauf der Kurven mit einer dreifachen beziehungsweise einer vierfachen Switch-IP-Core-Anbindung annähert.

Tabelle 5.1.: $MTTF_{Sim}$ (in Jahren) beim Ausfall aller Komponenten

Gittergröße	Anzahl der Switch-IP-Core-Verbindungen			
	1	2	3	4
3x3	4,61	5,20	5,82	6,00
5x5	11,38	13,23	14,84	15,33
20x20	158,75	195,21	218,67	226,40

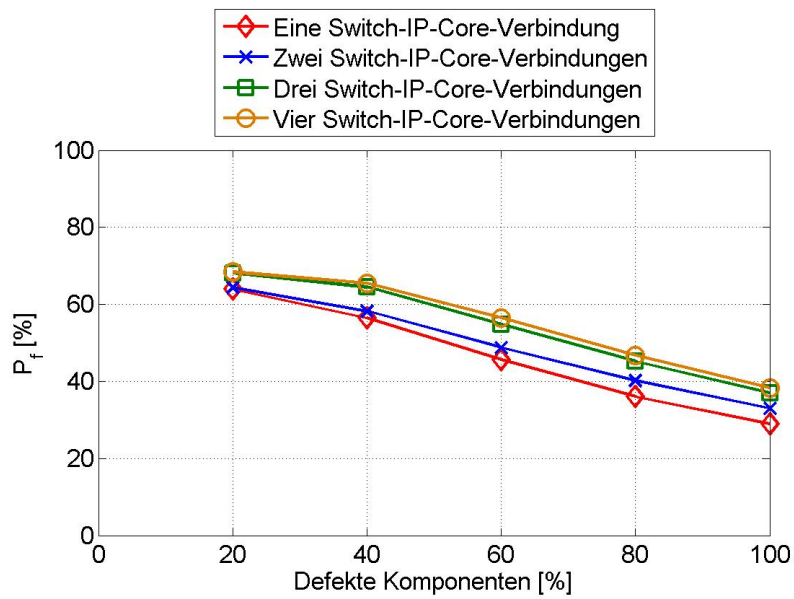


Abbildung 5.1.: Erreichbarkeitswahrscheinlichkeit wenn gesamte Komponenten ausfallen bei einem 3x3-Gitter

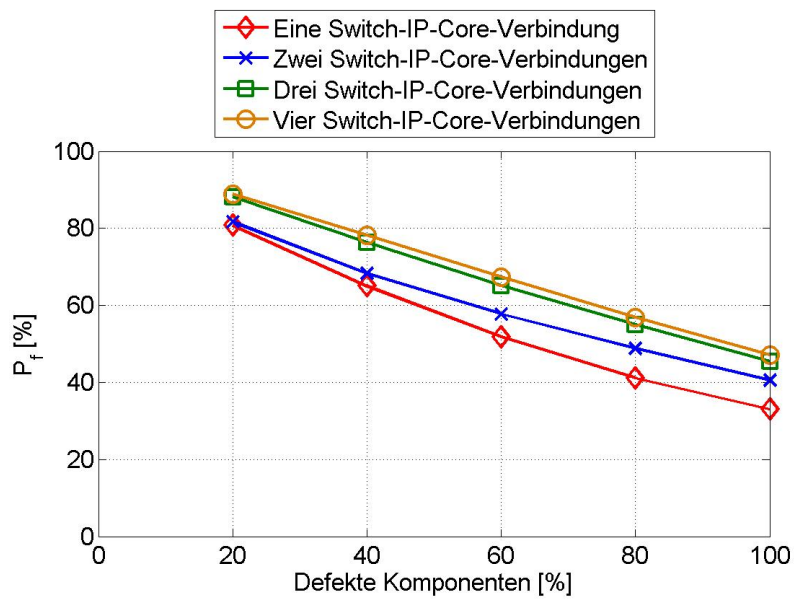


Abbildung 5.2.: Erreichbarkeitswahrscheinlichkeit wenn gesamte Komponenten ausfallen bei einem 20x20-Gitter

5.2. Ausfall von Switch-Switch- und Switch-IP-Core-Verbindungen (permanente Fehler)

Für die Simulationen, wenn die Switch-Switch- und die Switch-IP-Core-Verbindungen ausfallen, erhält man für die $MTTF_{Sim}$ -Werte aus der Tabelle 5.2 beziehungsweise der Tabelle 5.3. Die Simulationen ergeben für ein 3×3 -Gitter und bidirektionalen Verbindungen einen P_{MTTF} -Wert von 3,22 für bidirektionale Verbindungen liegt dieser bei 3,20. Dieser Unterschied ist durch die in Abschnitt 4.1.3 angesprochenen Differenz der simulierten Kurven zu erklären. Für die 20×20 -Gitter ergibt sich ein $P_{MTTF} = 3,24$.

In der Abbildung 5.3 wird betrachtet wie der Verlauf der Erreichbarkeit in Abhängigkeit der Anzahl der defekten Verbindungen für ein 3×3 -Gitter ist. Diese Berechnung wurde ebenso für ein 20×20 -Gitter durchgeführt, das Ergebnis ist in der Abbildung 5.4 zu sehen. Bei der Berechnung für das 3×3 -Gitter ist zu sehen, dass die Kurven, bis auf die Kurve mit der zweifachen und der dreifachen Switch-IP-Core-Anbindung, einen unterschiedlichen Verlauf aufweisen. In der Berechnung für das 20×20 -Gitter ist zu sehen, dass die Kurve mit einer einfachen Switch-IP-Core-Anbindung, im Vergleich zu den anderen relativ stark abfällt. Hierdurch kann der Abstand in der Abbildung 5.4 erklärt werden.

Tabelle 5.2.: $MTTF_{Sim}$ (in Jahren) beim Ausfall aller Verbindungen (bidirektional)

Gittergröße	Anzahl der Switch-IP-Core-Verbindungen			
	1	2	3	4
3x3	2,97	5,37	6,93	9,56
5x5	9,73	17,77	23,35	31,15
20x20	182,71	344,12	468,10	591,79

Tabelle 5.3.: $MTTF_{Sim}$ (in Jahren) beim Ausfall aller Verbindungen (unidirektional)

Gittergröße	Anzahl der Switch-IP-Core-Verbindungen			
	1	2	3	4
3x3	6,08	10,93	14,01	19,47
5x5	19,73	35,84	47,03	62,77
20x20	365,05	684,60	935,76	1181,67

5.2. Ausfall von Switch-Switch- und Switch-IP-Core-Verbindungen (permanente Fehler)

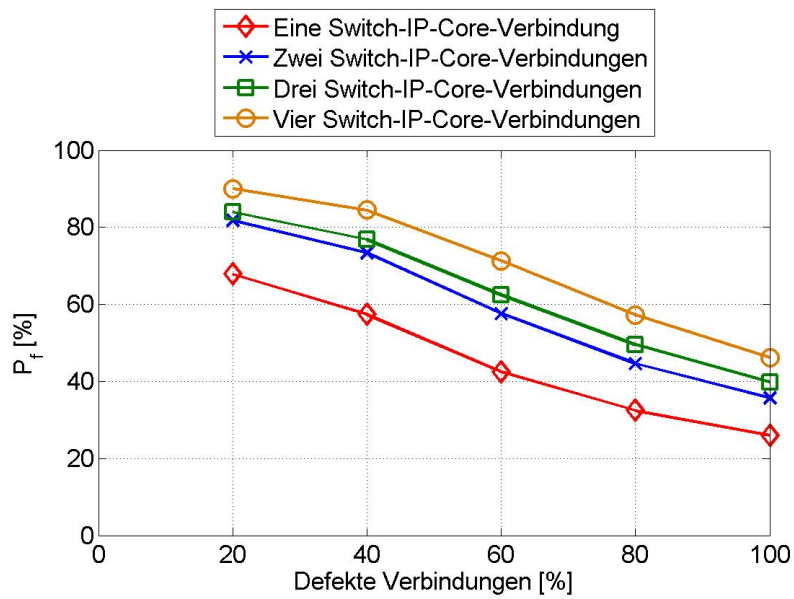


Abbildung 5.3.: Erreichbarkeitswahrscheinlichkeit wenn unidirektionale Verbindungen ausfallen bei einem 3x3-Gitter

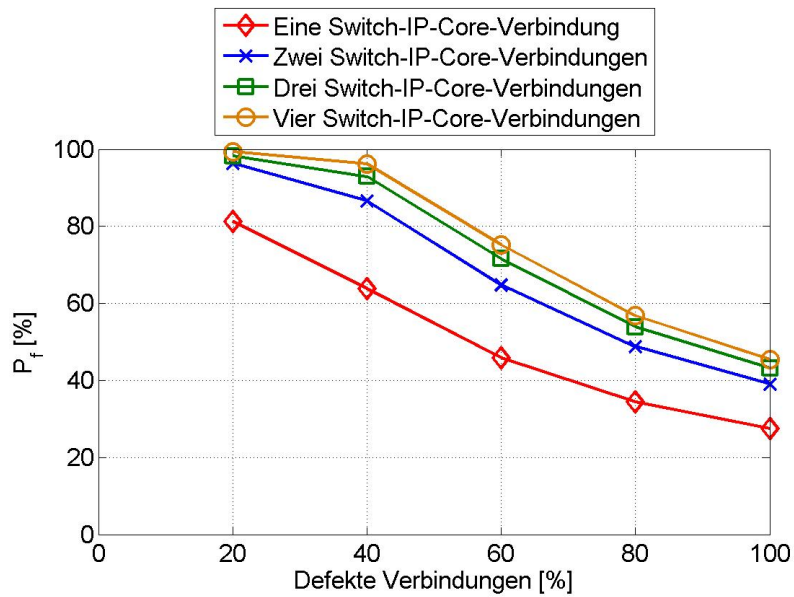


Abbildung 5.4.: Erreichbarkeitswahrscheinlichkeit wenn unidirektionale Verbindungen ausfallen bei einem 20x20-Gitter

5.3. Ausfall von Switch-Switch- und Switch-IP-Core-Verbindungen (permanente und transiente Fehler)

Für die transienten Numeriksimulationen sind die $MTTF_{Sim}$ -Werte den Tabellen 5.4 und 5.5 zu entnehmen. Sofern das System transiente und permanente Fehler unterscheiden kann ist das System entsprechend länger verwendbar. Dies ist an den steigenden Werten der zwei Tabellen 5.4 und 5.5 zu sehen. Für das 3x3-Gitter und null Prozent transienten Fehlern errechnet sich ein $P_{MTTF} = 3,18$ und für 80 % transiente Fehler ergibt sich ein Wert von 3,16. Diese Berechnung wird ebenso für das 20x20-Gitter durchgeführt, hier ergibt sich ein $P_{MTTF} = 3,24$ für null Prozent transiente Fehler und für 80 % transiente Fehler ergibt sich ein $P_{MTTF} = 3,26$.

Die Zeilen der Tabellen, die jeweils null Prozent transiente Fehler haben, besitzen den gleichen Simulationsaufbau wie die Simulationen ohne transiente Fehler, somit sollten diese die gleichen Werte aufweisen. Da dies jedoch nicht der Fall ist (vergleiche Tabelle 5.3), ist von entsprechenden Simulationsschwankungen auszugehen.

Tabelle 5.4.: $MTTF_{Sim}$ (in Jahren) beim Ausfall aller Verbindungen (unidirektional, 3x3-Gitter) inklusive transienter Fehler

Anzahl der transiente Fehler [%]	Anzahl der Switch-IP-Core-Verbindungen			
	1	2	3	4
0	6,14	10,94	14,02	19,54
20	8,03	14,15	17,94	24,66
40	10,48	18,62	23,90	32,84
60	15,67	27,83	35,50	49,20
80	30,97	55,34	70,34	97,76

Tabelle 5.5.: $MTTF_{Sim}$ (in Jahren) beim Ausfall aller Verbindungen (unidirektional, 20x20-Gitter) inklusive transienter Fehler

Anzahl der transiente Fehler [%]	Anzahl der Switch-IP-Core-Verbindungen			
	1	2	3	4
0	366,42	688,65	937,84	1187,60
20	456,21	859,55	1173,59	1475,39
40	605,74	1152,26	1560,86	1981,85
60	911,94	1715,11	2329,41	2958,33
80	1819,91	3453,51	4663,84	5932,53

Erkennung und Lokalisierung von fehlerhaften Verbindungen

In den numerischen Simulationen und den Berechnungen zu diesen Simulationen aus den Kapiteln 4 und 5 wurde gezeigt, dass die redundante Anbindung der IP-Cores an die Switche eine deutliche Verbesserung für die Erreichbarkeit der IP-Cores ergibt. Der erste Schritt für ein fehlertolerantes System ist die Detektierung und Lokalisierung von Fehlern.

Bei dem Ausgangssystem (Kapitel 3) ist es nicht möglich fehlerhafte Ports zu detektieren, geschweige denn zu lokalisieren. Dies bedeutet, wenn eine Verbindung ausfällt, dann kann unter Umständen das gesamte System nicht mehr funktionsfähig sein.

In diesem Kapitel wird die Detektierung und Lokalisierung von fehlerhaften Verbindungen behandelt. Dazu werden bei jedem Datenpaket im Quell-NI CRC-Bits in das Endeflit eingefügt. Diese CRC-Bits werden aus dem gesamten Paket generiert.

Im Ziel-NI werden diese CRC-Bits dann wieder kontrolliert. Wenn hierdurch ein Fehler erkannt wird, dann wird ein Flit an die Quelle gesendet. Von dort aus wird dann eine Switch zu Switch beziehungsweise eine Switch zu IP-Core Überprüfung gestartet. Mit dieser Überprüfung wird der Fehler lokalisiert. Durch dieses Verfahren wird sichergestellt, dass der gleiche Weg verwendet wird, den das eingetroffene Datenpaket verwendet hat.

In den nächsten zwei Abschnitten, in denen die Überprüfung der Verbindungen erklärt wird, sind in den Abbildungen nur Switche dargestellt. Hierdurch werden die Abbildungen weniger komplex. Die Überprüfung der Switch-IP-Core-Verbindungen wird bei dieser Überprüfung jedoch ebenso durchgeführt.

Durch die Überprüfung der Verbindungen werden zusätzliche Pakete versendet. Jedoch werden diese Pakete nur nach einem detektierten Fehler versendet. Im Vergleich zu den Datenpaketen sind die Überprüfungspakete zu vernachlässigen.

6.1. Fehlererkennung (Ende zu Ende)

Bei jedem Paket werden im Quell-NI aus allen Paketflits die CRC-Bits erzeugt. Diese werden dann am Ende des Endeflits eingefügt. Im Ziel-NI wird dann eine CRC-Überprüfung des gesamten Paketes durchgeführt. Sollte bei dieser Überprüfung ein Fehler detektiert werden, so wird ein Paket aus einem Flit zurück an den Ausgangsswitch gesendet. Dieses Flit wird als Defektflit bezeichnet. Bei diesem wird dann eine Switch zu Switch beziehungsweise Switch zu NI Überprüfung gestartet, der Ablauf dieser Überprüfung ist in dem Abschnitt 6.2 beschrieben. Exemplarisch ist in der Abbildung 6.1 der Weg eines Datenpaketes und eines entsprechenden Defektflits dargestellt.

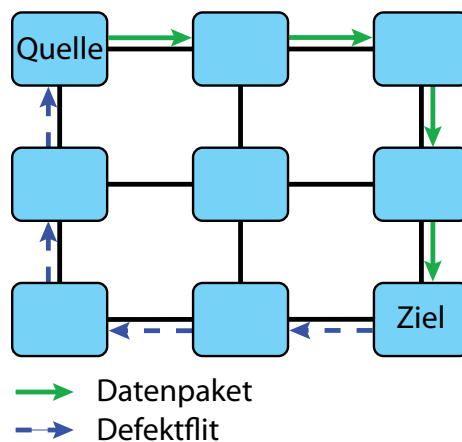


Abbildung 6.1.: CRC-basierte Fehlererkennung (Fehlerdetektion, Ende zu Ende)

Dadurch, dass nicht in jeder Komponente die CRC-Bits überprüft werden, kann Energie gespart werden.

6.2. Fehlerlokalisierung (Komponente zu Komponente)

Die Fehlerlokalisierung findet durch eine abfragebasierte Überprüfung statt. Bei der abfragebasierten Fehlererkennung wird eine Nachricht, bestehend aus einem Überprüfungsflit und der invertierten Version des Überprüfungsflits, zu der nächsten Komponente gesendet. Durch die invertierte Version des Flits können alle Haftfehler detektiert werden. Wenn zum Beispiel das Überprüfungsflit die Bitfolge 111010010001001 hat, dann können alle Haftfehler null bei den Stellen, an denen eine null im Überprüfungsflit vorhanden ist, nicht detektiert werden. Durch das invertierte Flit ist dies möglich

Beim Versenden dieser Überprüfungsflits wird in einer lokalen Defekttabelle für den entsprechenden Port das entsprechende Bit gesetzt, dass der Port defekt ist.

Wenn ein Switch solche Überprüfungsflits empfängt, so kontrolliert dieser, ob die zwei Flits immer noch die invertierten Versionen voneinander sind. Dadurch, dass erst die

Inversionsüberprüfung durchgeführt wird, ist die CRC-Einheit nicht so stark belastet. Sollte diese Überprüfung positiv sein, dann werden die CRC-Bits überprüft. Durch diese Überprüfung werden invertierende Fehler (line flip fault) erkannt, da diese bei beiden Flits das entsprechende Bit invertieren.

Sobald eine der beiden Überprüfungen einen Fehler ausgibt, wird kein weiteres Paket versendet. Dadurch wird der Eintrag in der lokalen Defekttabelle des Switches beziehungsweise der NI, von dem dieses Überprüfungs paket gekommen ist, nicht zurückgesetzt. Somit ist dieser Port als defekt erkannt. Sollten die beiden Überprüfungen positiv ausgefallen sein, dann wird eine Antwort an die vorherige Komponente gesendet. Diese Antwort ist gleich aufgebaut wie die Überprüfungsflits, nur ist bei der Antwort das Kontrollbit gesetzt. In dem Kontrollbit wird vermerkt, welcher Teil der Überprüfung es ist, das heißt ob es eine Anfrage oder eine Antwort ist.

Falls das Ziel-NI noch nicht erreicht ist, werden die Überprüfungsflits an den nächsten Switch beziehungsweise das NI gesendet. Für den entsprechenden Ausgangsport der Überprüfungsflits wird der entsprechende Eintrag in der lokalen Defekttabelle gesetzt.

Wird an einem Switch beziehungsweise einer NI eine Antwort empfangen, dann wird zuerst wieder die Inversionsüberprüfung durchgeführt. Wenn diese positiv ausfällt, dann werden die CRC-Bits der Flits überprüft. Sofern beide Überprüfungen ein positives Ergebnis ergeben, dann wird der Eintrag für den Port aus der lokalen Defekttabelle dieses Switches beziehungsweise der NI wieder zurückgesetzt. Dieses Verfahren der abfragebasierten Überprüfung ist in der Abbildung 6.2 und der Abbildung 6.3 zu sehen.

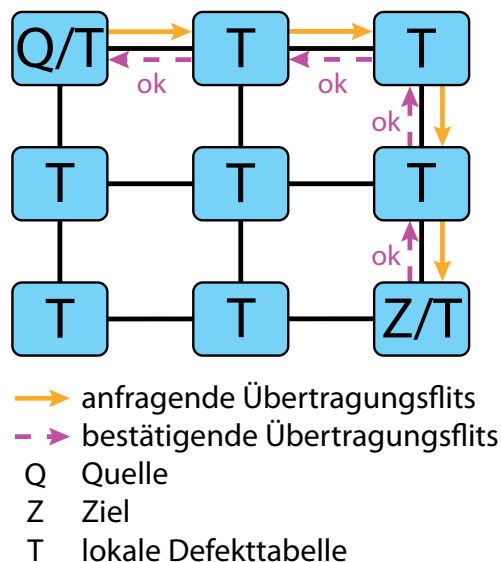


Abbildung 6.2.: Abfragebasierte Fehlererkennung (Fehlerlokalisierung, Komponente zu Komponente)

Es ist zu beachten, dass immer nur die unidirektionale Verbindung zwischen den Switchen als defekt erkannt und markiert wird. Sollte ein bidirektionaler Fehler auftreten, muss

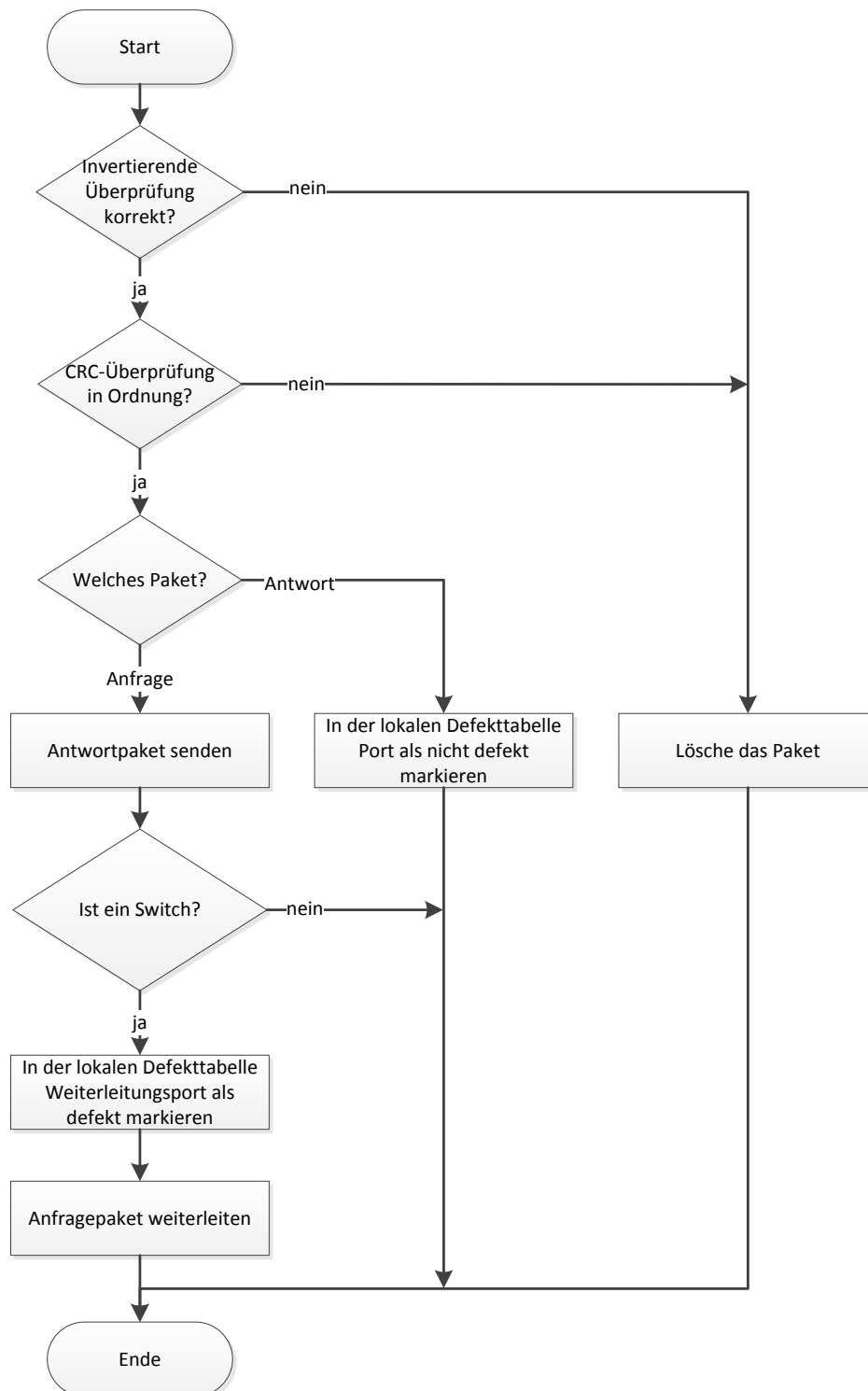


Abbildung 6.3.: Abfragebasierte Überprüfung der Verbindungen

es auch zwei Überprüfungen geben. Durch dieses Verfahren, wird nur die jeweilige Verbindung zwischen den Switchen beziehungsweise dem entsprechenden Switch und dem entsprechenden NI unidirektional blockiert. Dies bedeutet, dass nicht das gesamte NoC blockiert wird.

Wenn nun ein unidirektionaler Defekt lokalisiert wird und dann eine Überprüfung in die gegengesetzte Richtung des unidirektionalen Defektes überprüft wird, dann wird diese Verbindung ebenfalls als Defekt markiert, denn es kann keine Antwort zurückgesendet werden. Eine Lösung für dieses Problem ist, dass die Überprüfungsflits dann einen anderen nicht defekten Port der Komponente verwenden (Abbildung 6.4). Für diese Lösung müsste die Information, welcher Port mit dieser Überprüfung kontrolliert wird, zu den Überprüfungsflits hinzugefügt werden.

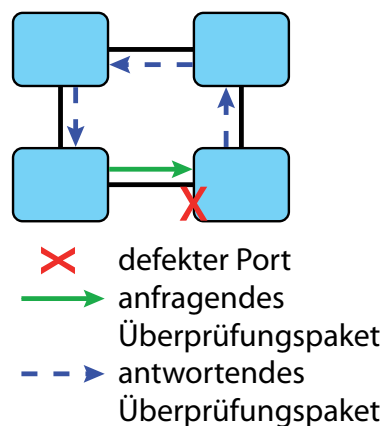


Abbildung 6.4.: Weiterleitung eines Überprüfungspaketes

6.3. Überprüfung von Steuersignalen

Die Steuersignale für das Senden (send) und das Empfangen (rec) sowie die Signale ob ein FIFO voll ist (buf_full_in, buf_full_out) werden bisher nicht überprüft. Dies könnte jedoch zum Beispiel durch die Implementierung eines differenziellen Signals sichergestellt werden.

6.4. Erweiterungen für die Fehlererkennung und Fehlerlokalisierung

6.4.1. Flittypen

Da weitere Flittypen benötigt werden, muss die Anzahl der Steuerbits auf drei erhöht werden. Wofür die entsprechende Bitkombination steht, kann der Tabelle 6.1 entnommen werden.

Tabelle 6.1.: Flittypen (mit Fehlererkennung)

Bitkombination der Steuerbits	Funktion
000	Invertiertes Überprüfungsflit (abfragebasierte Fehlererkennung) (Komponente zu Komponente)
001	Kopfflit
010	Endeflit
011	Defektfliit (Ende zu Ende)
100	Datenflit
101	ungenutzt
110	ungenutzt
111	Überprüfungsflit (abfragebasierte Fehlererkennung) (Komponente zu Komponente)

Das Kopfflit wird durch die Quelladresse erweitert. In dem Endeflit werden die CRC-Bits untergebracht. Das Defektfliit, das an den Ausgangsswitch gesendet wird, sobald ein Fehler erkannt wird, enthält die drei Steuerbits, die Adresse des Zielswitches und die Adresse des Quellswitches. Das Überprüfungsflit für die abfragebasierte Überprüfung umfasst die drei Steuerbits, die Zieladresse, ein Kontrollbit und die CRC-Bits. Die unterschiedlichen Pakettypen sind in der Abbildung 6.5 zu sehen.

6.4.2. Switch

Um zum Beispiel das Überprüfungs paket, das im Router erzeugt wird, versenden zu können, wird eine Verbindung zwischen dem Router und den MUXs benötigt (Abbildung 6.6). Die MUXs die den Ports vorgeschaltet sind, werden vergleichbar wie der MUX vor Port drei verschaltet. Da der Router den Paketfluss steuert, kann dieser entscheiden welcher MUX-Eingang an dem entsprechenden Port freigeschaltet wird.

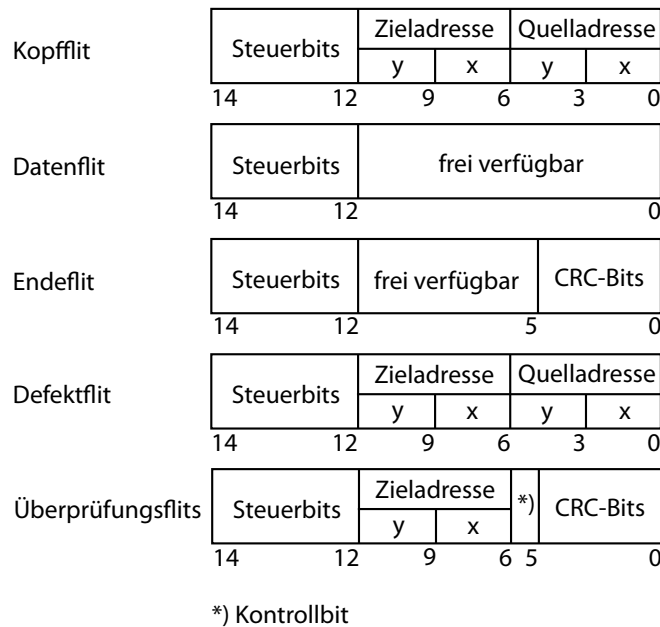


Abbildung 6.5.: Flittypen mit der Erweiterung zur Erkennung von fehlerhaften Verbindungen

6.4.3. Router

Der Router muss dahingehend erweitert werden, dass er die weiteren Flittypen verarbeiten kann. Außerdem wird eine zentrale Einheit für die CRC-Überprüfung hinzugefügt [ope]. Es ist bewusst nicht in jedem Port eine CRC-Einheit implementiert, um den Hardwareaufwand gering zu halten. Der Nachteil dieser Variante ist, dass immer überprüft werden muss, ob die CRC-Einheit aktuell genutzt wird. Falls die CRC-Einheit genutzt wird, dann werden die Daten zwischengespeichert. Hierdurch gibt es eine maximale Anzahl an Datenflits. Dies kann in der Implementierung durch Verändern einer Variable angepasst werden. In implementierten Version sind es maximal zwei Datenflits. Es ist auch möglich die Pakete so lange in den FIFOs zu belassen bis die CRC-Einheit zu Verfügung steht. Durch das Belassen der Pakete in den FIFOs ist davon auszugehen, dass der benötigte Hardwareaufwand geringer wird als bei der Lösung mit der Zwischenspeicherung. Mittels des Zwischenspeichern, der Pakete wird es ermöglicht, dass andere Pakete verarbeitet werden. Des Weiteren wird die lokale Defekttabelle im Router untergebracht.

6.4.4. Netzwerkschnittstelle

Die NI wird mit einer zentralen CRC-Komponente von [ope] und einer lokalen Defekttabelle ausgestattet (Abbildung 6.7). Außerdem wird die NI dahingehend erweitert, dass die Pakete untersucht werden und gegebenenfalls ein entsprechendes Defektfrit versenden wird. Des

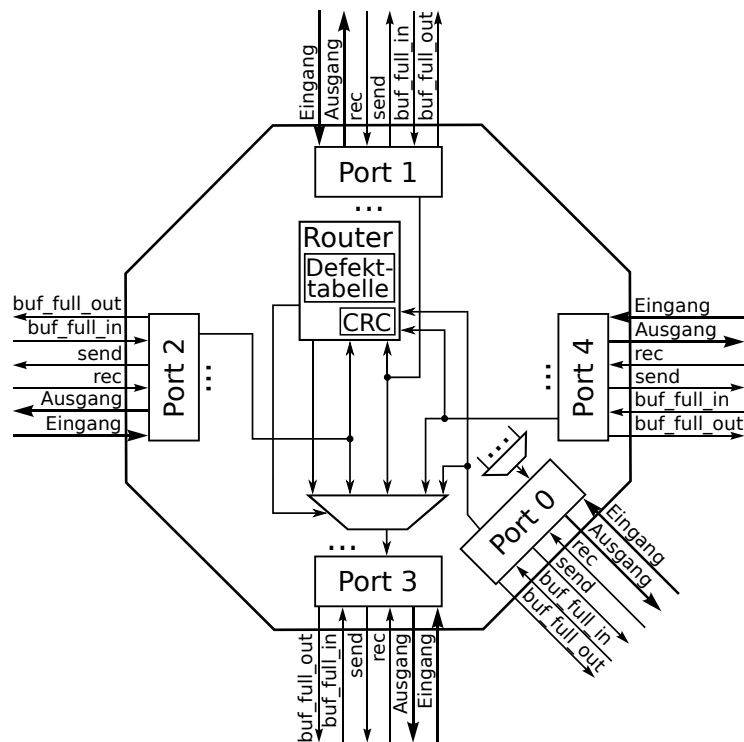


Abbildung 6.6.: Architektur des erweiterten Switches um fehlerhafte Ports erkennen zu können

Weiteren wird die Funktionalität zur Unterstützung des Überprüfungspaketes hinzugefügt.

6.4.5. Erweiterter Routingalgorithmus

Aufgrund dessen, dass ein Switch beziehungsweise eine NI Ports haben können, die als defekt markiert worden sind, können diese Ports nicht mehr verwendet werden. Somit müssen die Pakete auf einem alternativen Weg geroutet werden.

Um den Routingalgorithmus zu erweitern sind zwei Varianten untersucht worden. Jedoch ist die Variante mit einem zusätzlichen yx-Routing nicht deadlockfrei. Dieser Algorithmus mit dem zusätzlichen yx-Routing ist im Anhang B.1.1 zu finden. Die zweite Variante ist nachfolgend erklärt.

Der xy-Routingalgorithmus wurde durch eine Alternative erweitert. Wenn der gewünschte Port als defekt markiert sein sollte, so kann aus der Tabelle 6.2 der alternative Port entnommen werden. Ist der alternative Port ebenso defekt, dann wird das Paket gelöscht. Es gibt keine Alternative von der Alternative. Sollte es keine Alternative in der Tabelle 6.2 geben, so wird das Paket gelöscht. Ebenso wird das Paket gelöscht, wenn das Paket aus dem

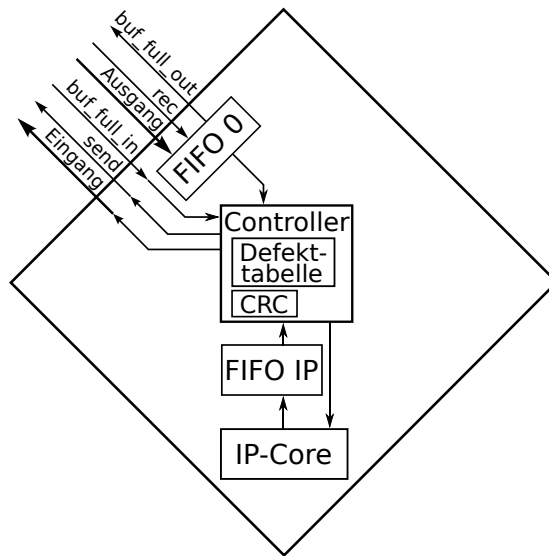


Abbildung 6.7.: Architektur der erweiterten Netzwerkschnittstelle, um fehlerhafte Ports erkennen zu können

gleichen Port geroutet werden soll, in den das Paket hereingekommen ist. Der Algorithmus ist in der Abbildung 6.8 zu sehen.

Tabelle 6.2.: Ausgangsports für die Paketweiterleitung

Standardport	Alternativer Ausgang
0	-
1	2
2	1
3	-
4	1

Livelock

Wenn zum Beispiel der Port eins defekt ist, dann ist der Port zwei der alternative Port. Durch das xy-Routing würde das Paket wieder zurückgesendet werden (Abbildung 6.9). Wird dies nicht beachtet, entsteht eine Verklemmung (Livelock) [Scho5]. Um dies zu unterbinden, wird immer wenn ein Paket zum gleichen Port geleitet werden soll, von dem das Paket kommt, das Paket gelöscht.

6. Erkennung und Lokalisierung von fehlerhaften Verbindungen

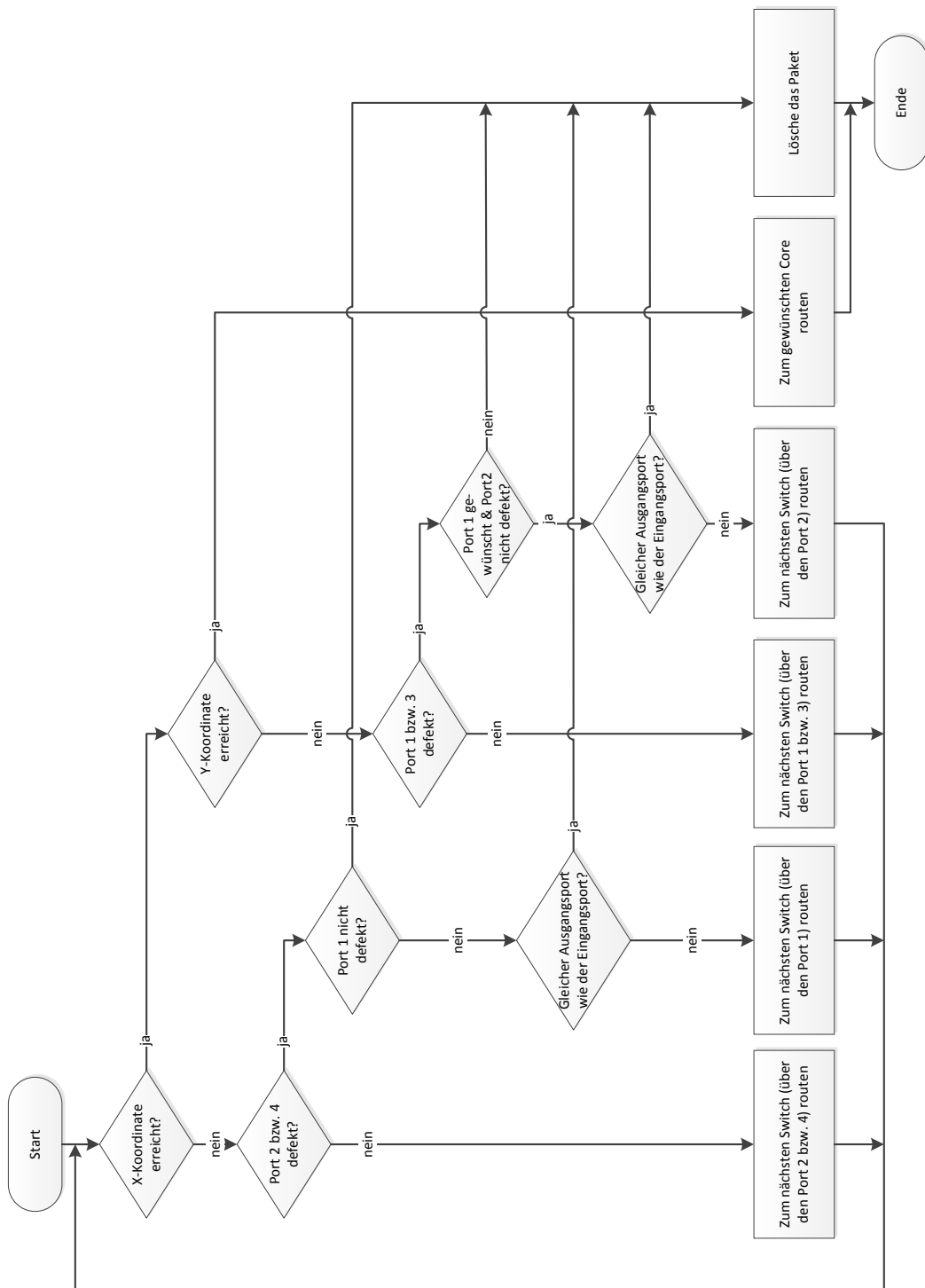


Abbildung 6.8.: Deadlockfreier fehlertoleranter Routingalgorithmus

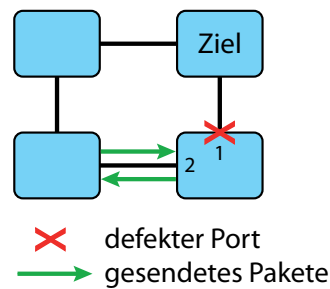


Abbildung 6.9.: Livelock beim xy-Routing

Verfahren zum Überprüfen der Deadlockfreiheit

Mittels des Verfahrens nach [Dal87b] kann überprüft werden ob ein Routingalgorithmus deadlockfrei ist.

Bei diesem Verfahren wird der Verbindungsgraph des Systems erstellt. Aus diesem Graphen werden die Abhängigkeiten der Verbindungen der Komponenten (Kanäle) in Kanalabhängigkeitsgraphen übertragen. Die Gewichtung eines Kanals muss immer ansteigend sein. Außerdem dürfen dabei keine Schleifen entstehen. Da diese Bedingungen eingehalten werden müssen, aber dennoch der gesamte Routingalgorithmus durch die Kanalabhängigkeitsgraphen abgebildet werden muss, sind meist mehrere Teilkanalabhängigkeitsgraphen erforderlich. Bei einem Routingvorgang darf dabei der Teilkanalabhängigkeitsgraph nicht gewechselt werden. Sind diese Vorgaben erfüllt, dann ist der Algorithmus deadlockfrei.

Beispiel

In der Abbildung 6.10 ist der Verbindungsgraph eines 3x2-Gitters zu sehen. Durch den gewählten xy-Routingalgorithmus erhält man vier Teilkanalabhängigkeitsgraphen (Abbildung 6.11). Diese einzelnen Teilkanalabhängigkeitsgraphen haben alle den gleichen Aufbau. Während eines Routingvorganges darf nicht zwischen den unterschiedlichen Teilkanalabhängigkeitsgraphen hin- und hergewechselt werden. Da keiner der Kanalabhängigkeitsgraphen eine Schleife aufweist, ist der xy-Routingalgorithmus deadlockfrei.

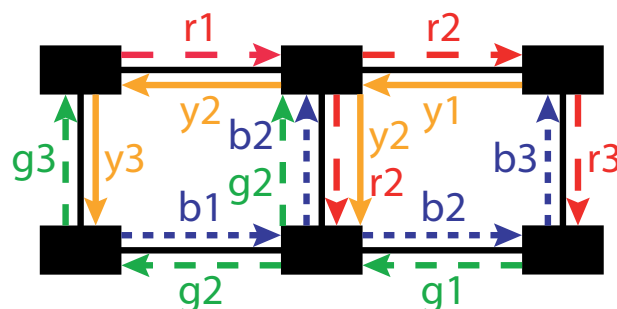


Abbildung 6.10.: Verbindungsgraph eines 3x2-Gitters mit xy-Routingalgorithmus

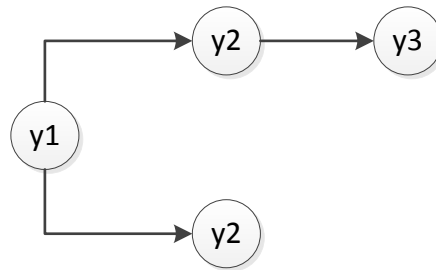


Abbildung 6.11.: Teilkanalabhängigkeitsgraph des gelben/durchgezogenen Kanals des 3x2-Gitters mit xy-Routingalgorithmus

Deadlockfreiheit

Dieser Routingalgorithmus wird durch eben eingeführte Verfahren auf Deadlockfreiheit geprüft. Der dazugehörige Verbindungsgraph ist in der Abbildung 6.12 und die entsprechenden Teilkanalabhängigkeitsgraphen sind in der Abbildung 6.13 und der Abbildung 6.14 zu sehen. Nach dieser Überprüfung ist der Routingalgorithmus deadlockfrei.

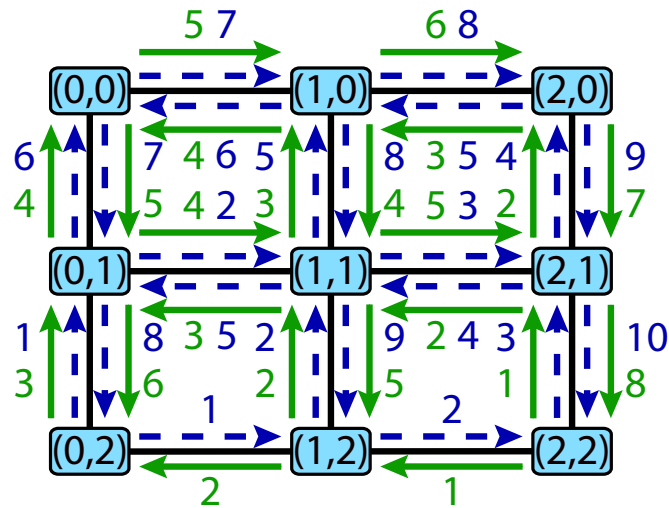


Abbildung 6.12.: Verbindungsgraph zum Überprüfen der Deadlockfreiheit des Routingalgorithmus mit einer Alternative

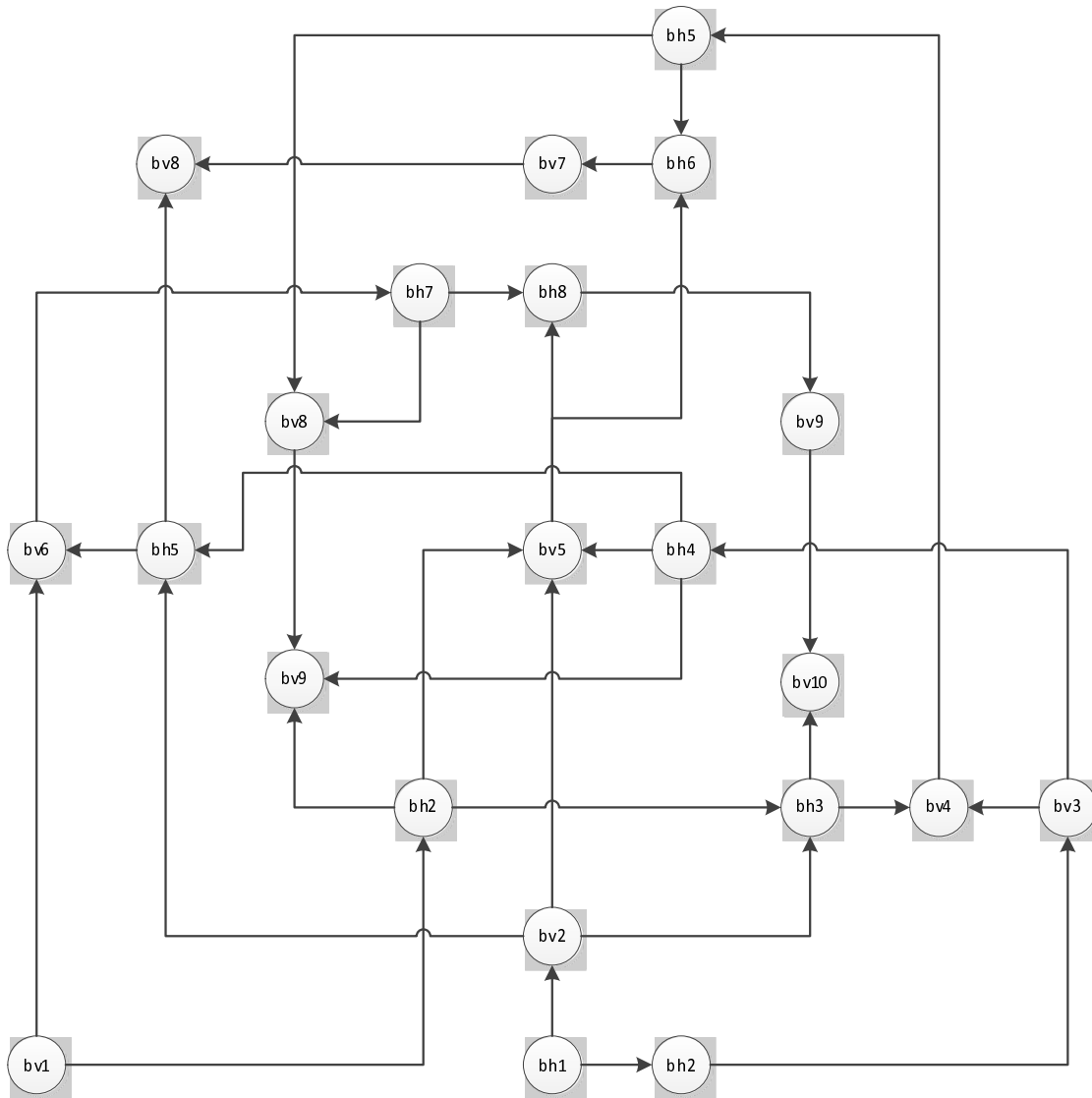


Abbildung 6.13.: Kanalabhängigkeitsgraph (blau/gestrichelt) zum Überprüfen der Deadlockfreiheit des Routingalgorithmus mit einer Alternative

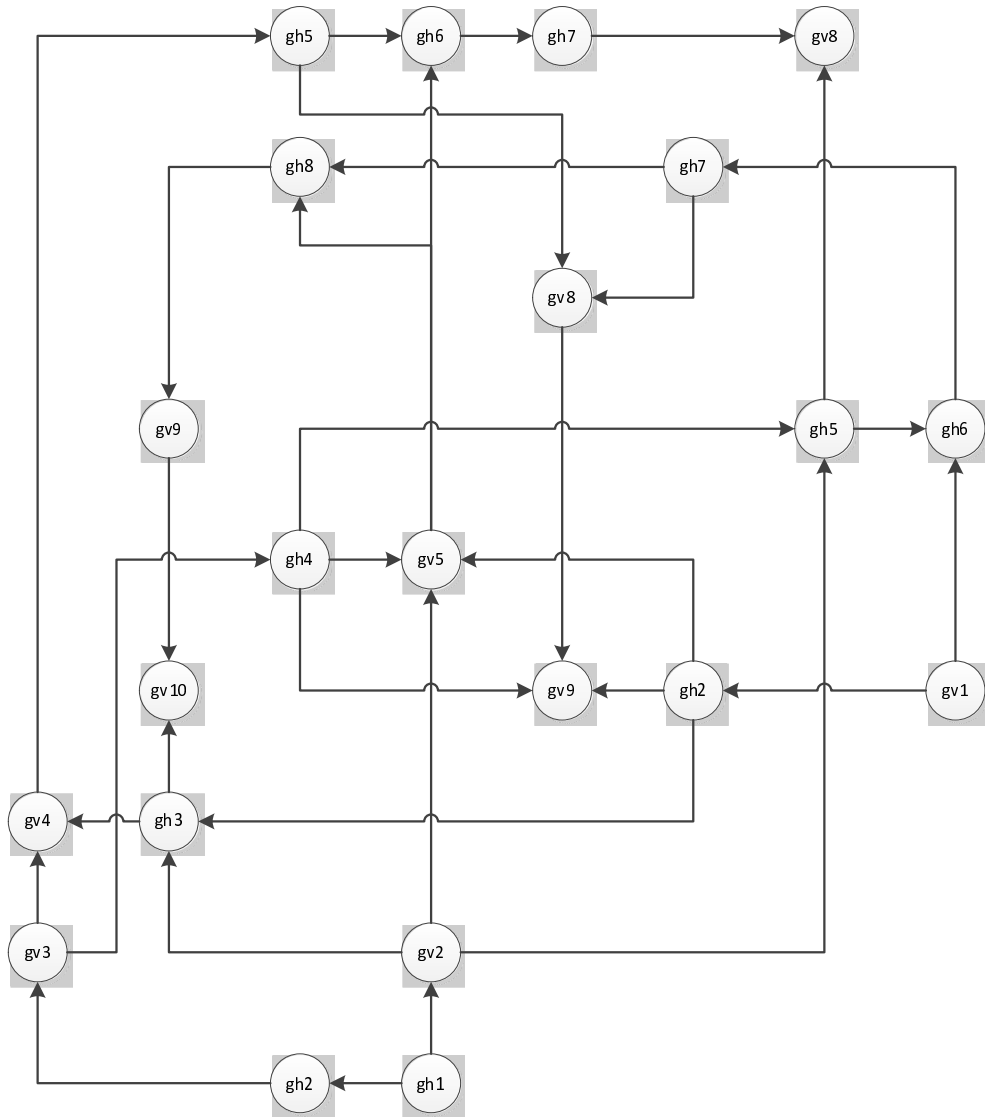


Abbildung 6.14.: Kanalabhängigkeitsgraph (grün/durchgezogen) zum Überprüfen der Deadlockfreiheit des Routingalgorithmus mit einer Alternative

Redundante Anbindung von IP-Cores an die Switche

7.1. Konzept

Bei einer Verbindung zwischen einem Switch und einem IP-Core, respektive einer NI, wie in der Abbildung 7.1, sind die IP-Cores schon bei dem Defekt dieser einen Verbindung unbrauchbar.

Um dies zu vermeiden werden die IP-Cores mehrfach angebinden. Durch die Verbindung eines IP-Cores an mehrere Switche kann die Erreichbarkeit erhöht werden. In der Abbildung 7.2 ist dargestellt, wie die Implementierung mit vier Verbindungen von der NI zu den Switchen aussieht. Weitere Konfigurationen sind im Anhang C.1 zu finden. Dadurch steigt die Wahrscheinlichkeit, dass ein IP-Core erreichbar ist, obgleich eine Komponente oder eine Verbindung zwischen zwei Komponenten ausfällt. Dies wurde in dem Kapitel 4 gezeigt.

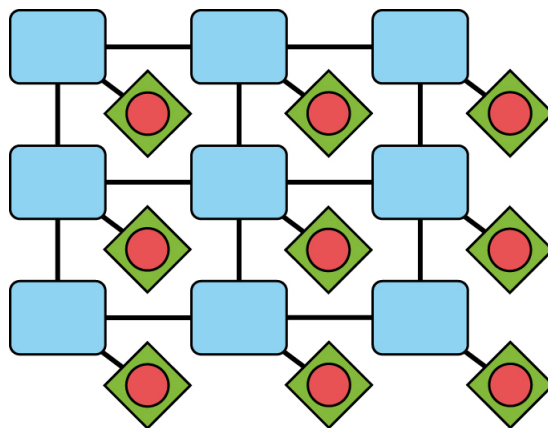


Abbildung 7.1.: Zweidimensionales Gitter ohne redundante Anbindung von IP-Cores

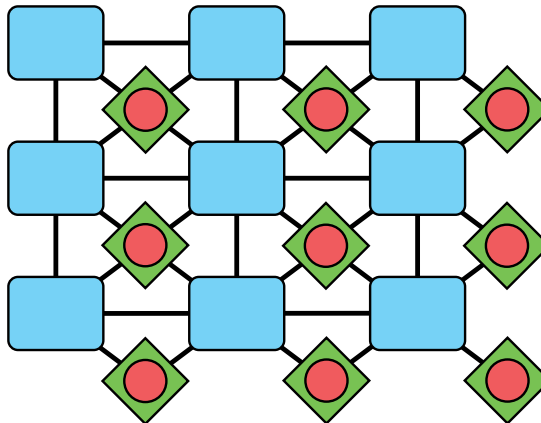


Abbildung 7.2.: Zweidimensionales Gitter inkl. der redundanten Anbindung von IP-Cores (vier verbindbare Switche an einen IP-Core)

Die Validierung der redundanten Anbindung von IP-Cores an die Switche wird in dem Abschnitt 10.2 behandelt.

7.2. Erweiterungen für die redundante Anbindung

7.2.1. Flittypen

Bei dem Kopfflit wird die Information hinzugefügt, welcher IP-Core an dem jeweiligen Switch gemeint ist (Adresse des IP-Cores). Dies wird für den Quell-IP-Core und den Ziel-IP-Core gemacht.

Bei der in dieser Arbeit verwendenden Größe des Gitters stehen im Datenpaketteil 16 Bit zur freien Verfügung. Die Anzahl der CRC-Bits beträgt sieben.

Beim Defektflit und bei dem Überprüfungspaket werden bei den Adressen ebenfalls noch die Adresse der IP-Cores hinzugefügt.

In der Abbildung 7.3 sind die verschiedenen Flittypen inklusive der Erweiterungen dargestellt.

7.2.2. Switch

Beim Switch werden entsprechend viele Switch-NI-Ports respektive Switch-IP-Core-Ports angefügt. Somit besitzt ein Switch mit einer vierfachen Anbindung an eine NI acht Ports. In der Abbildung 7.4 ist ein solcher Switch zu sehen. Weitere Konfigurationen von Switchen sind im Anhang C.2 zu finden.

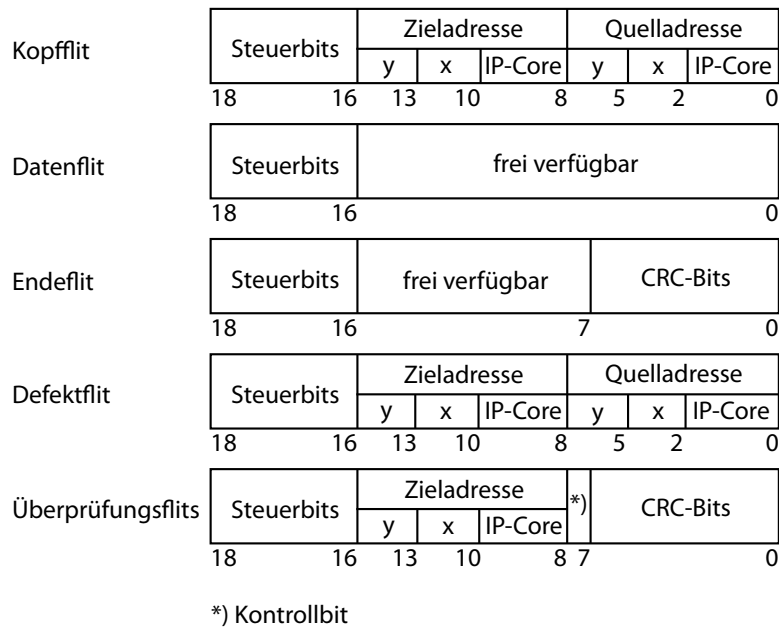


Abbildung 7.3.: Flittypen mit der Erweiterung der redundanten Anbindung

7.2.3. Router

Der Router wird so erweitert, dass er bei einer defekten Verbindung zwischen dem gewünschten Switch und der gewünschten NI das Paket zu einem anderen Switch weiterleitet, der ebenfalls eine Verbindung zu diesem NI besitzt. Zu welchem IP-Coreport weitergeroutet wird, kann der Tabelle 7.1 entnommen werden. Sollte zum Beispiel nur der Port null und sieben bei einem IP-Core vorhanden sein, so wird das Paket direkt an den Port sieben weitergeleitet. Der Algorithmus inklusive der folgenden Erweiterung ist in der Abbildung 7.5 zu sehen.

Tabelle 7.1.: Alternativer IP-Coreport

Standard IP-Coreport	Alternativer IP-Coreport
0	5
5	6
6	7
7	-

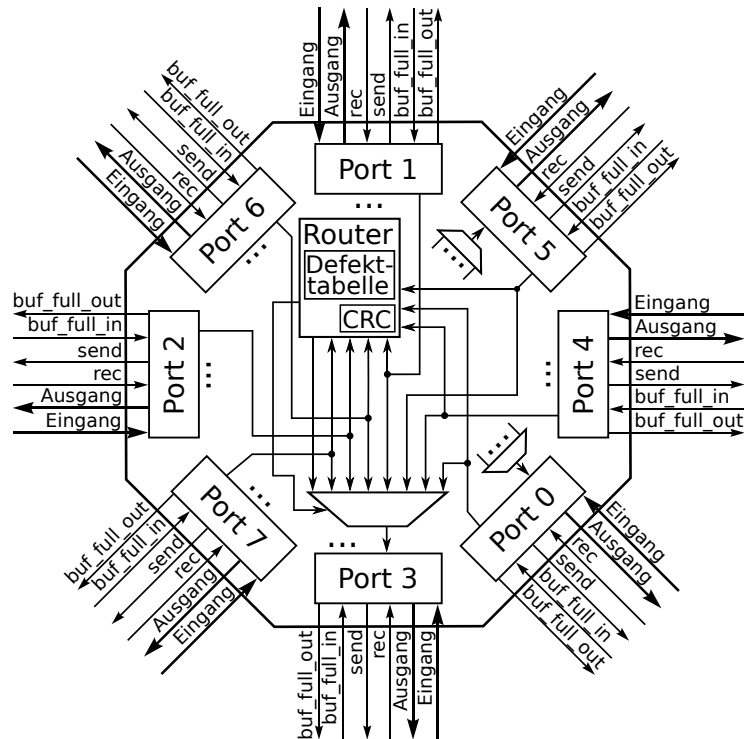


Abbildung 7.4.: Architektur des erweiterten Switches inkl. der redundanten Anbindung (vier verbindbare NI)

Erreichbarkeit eines alternativen IP-Cores

Sollte ein IP-Core durch keine seiner Ports mehr zu erreichen sein, so wird als Alternative der IP-Core verwendet, der an dem Port null des aktuellen Switches liegt. Das Paket wird solange weitergeleitet, bis das Paket am IP-Core mit der Adresse (x_{max}, y_{max}) angekommen ist, oder wenn das Paket nicht mehr weitergeleitet werden kann, da eine Verbindung defekt ist. Für den IP-Core mit der Adresse (x_{max}, y_{max}) gibt es keine Alternative, somit wird das Paket gelöscht. Wird das Paket nicht gelöscht, dann kann eine Verklemmung auftreten. In welcher Reihenfolge die IP-Cores versucht werden zu erreichen und mittels welcher Switch-IP-Core-Verbindung, ist der Abbildung 7.6 für eine vierfache Anbindung dargestellt.

Diese Erweiterung kann nur genutzt werden, wenn es sich um homogene IP-Cores handelt. Alternativ dazu wird das Paket gelöscht.

7.2.4. Netzwerkschnittstelle

Die NI fungiert bei der redundanten Anbindung als MUX. Dafür wird die Anzahl der Ports der NIs bis auf vier erhöht. Der Controller in der NI wird dahingehend erweitert, dass er

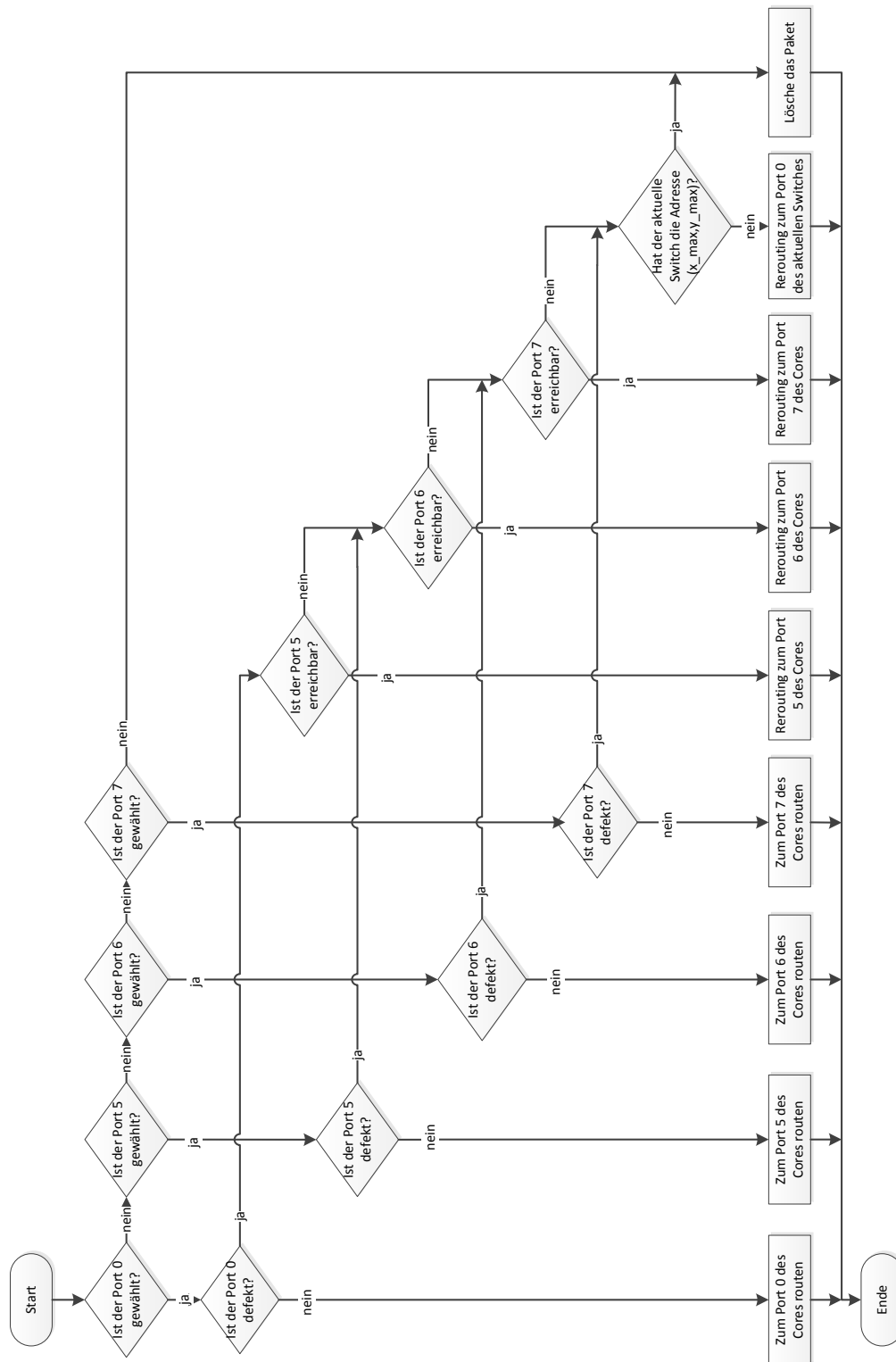


Abbildung 7.5.: Routingalgorithmus für die Erreichbarkeit von den IP-Cores bei einer redundanten Anbindung

7. Redundante Anbindung von IP-Cores an die Switche

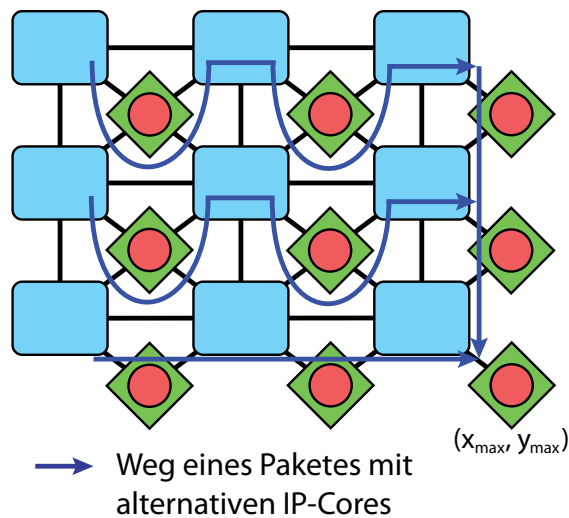


Abbildung 7.6.: Weiterleitung eines Paketes, wenn ein IP-Core nicht erreichbar ist

die Pakete von dem entsprechenden Switch an den IP-Core, oder umgekehrt, weiterleiten kann. In der Abbildung 7.7 ist der Aufbau der NI mit vier Switch-IP-Core-Verbindungen zu sehen. Weitere Konfigurationen von NIs sind im Anhang C.3 zu finden.

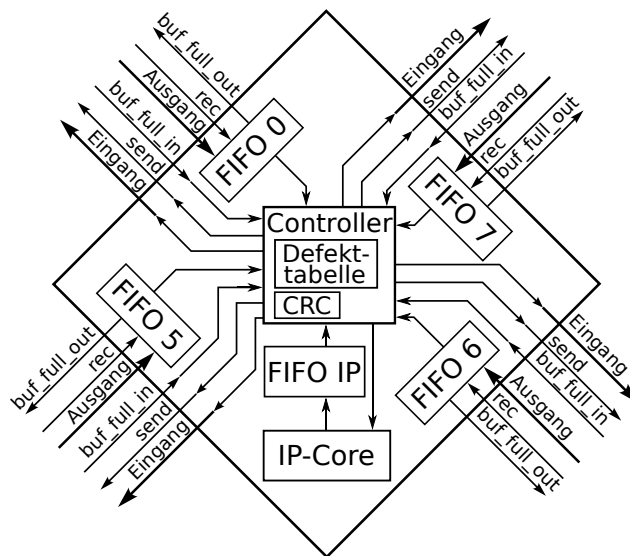


Abbildung 7.7.: Architektur der erweiterten Netzwerkschnittstelle inkl. der redundanten Anbindung (vier verbindbare Switche)

Unterscheidung zwischen permanenten und transienten Fehlern

Bisher sind alle lokalisierten Fehler direkt als permanente Fehler markiert worden. Somit werden aber auch transiente Fehler als permanent markiert. Bei transienten Fehlern ist die Auswirkung der Fehler auf die Schaltung zeitlich begrenzt. Diese Fehler werden durch unterschiedlichste Gründe hervorgerufen. Hierzu zählt unter anderem die radioaktive Strahlung und die Höhenstrahlung. Aber auch durch Temperaturschwankungen werden die Gatterlaufzeiten verändert. Diese Veränderungen können so groß sein, dass es Auswirkungen auf das Verhalten der Schaltung hat. Als weiterer Punkt ist zu beachten, dass die Versorgungsspannung teilweise nicht alle Teile der Schaltung mit der erforderlichen Spannung versorgen kann. Erkennt das System transiente Fehler auch als permanente, so kann eine Verbindung, bei der ein transienter Fehler als permanenter Fehler erkannt wurde, nicht mehr verwendet werden. Wenn so viele Fehler lokalisiert sind, wie viele Verbindungen vorhanden sind, dann hat das System keine als intakt markierten Verbindungen mehr. [Hed84]

8.1. Konzept

Um die transienten Fehler zu erkennen, wird die abfragebasierte Überprüfung aus Kapitel 6 erweitert.

Fehler werden bisher durch eine Überprüfung der CRC-Bits der Datenpakete erkannt. Wenn ein Fehler erkannt ist, dann wird ein Defektpaket an die Ausgangskomponente zurückgesendet. Dadurch wird sicher gestellt, dass die Überprüfung der Verbindungen auf dem gleichen Pfad gemacht wird den auch das Datenpaket verwendet hat. Diese Überprüfung ist eine Komponenten zu Komponenten Überprüfung. Um sicher zu stellen,

dass die transienten Fehler nicht als permanente Fehler erkannt werden, werden mehrere Überprüfungspakete gesendet.

Im implementierten System werden immer zwei Überprüfungspakete gesendet. Beim Versenden dieser Überprüfungspakete wird in der lokalen Defekttabelle ein Eintrag gemacht. Wenn eines der Überprüfungspakete an dem Switch oder an der NI angekommen ist, dann wird überprüft ob die zwei zueinander inversen Flits noch invers zueinander sind. Ist bei dieser Überprüfung kein Fehler aufgetreten, dann wird eine Überprüfung der CRC-Bits durchgeführt. Wurden die Überprüfungen ohne Fehler abgeschlossen, dann wird ein Überprüfungspaket als Antwort an die vorherige Komponente zurückgesendet. Dort wird dieses Antwortpaket auf Fehler untersucht. Ist es fehlerfrei, dann wird der Eintrag aus der lokalen Defekttabelle wieder entfernt. Wenn es das erste Überprüfungspaket ist und kein Fehler bei der Überprüfung aufgetaucht ist, dann wird noch untersucht ob es bereit am Ziel angekommen ist. Sollte dies nicht der Fall sein, dann werden wieder zwei Überprüfungspakete zur nächsten Komponente gesendet. Dabei wird der entsprechende Port in der lokalen Defekttabelle als defekt markiert. Bei dem zweiten Überprüfungspaket wird nur gegebenenfalls ein Antwortpaket versendet, es wird nicht untersucht ob es weiter geleitet werden muss.

So werden immer zwei Überprüfungspakete direkt hintereinander zur Überprüfung gesendet. Da transiente Fehler nur für relativ kurze Zeitspannen auftreten, können diese Fehler nicht mehr als permanent markiert werden. Hiermit wird eine sogenannte temporale Redundanz implementiert. Das heißt, sobald eine richtige Antwort zurückkommt, wird der Eintrag aus der lokalen Defekttabelle entfernt. Wenn nur eine Antwort richtig zurückkommt, dann ist ein transienter Fehler aufgetreten. Sollten beide Antworten nicht richtig oder überhaupt nicht zurückkommen, so ist die Verbindung mit einem permanenten Fehler belastet. Das Ausgangssystem für diese Implementierung stellt die Implementierung aus dem Kapitel 7 dar.

Das zusätzliche Paketaufkommen durch die temporale Redundanz beschränkt sich auf die zusätzlichen Überprüfungspakete. Im normalen Betrieb sind diese im Vergleich zu den Datenpaketen relativ gering.

Die Validierungssimulationen mit den transienten Fehlern sind im Abschnitt 10.3 zu finden.

8.2. Erweiterungen für die Unterscheidung zwischen permanenten und transienten Fehlern

8.2.1. Router

Der Router muss für die Unterscheidung von permanenten Fehlern und von transienten Fehlern zwei Überprüfungspakete versenden können. Des Weiteren wird der Router dahingehend erweitert, dass die Überprüfungspakete separat überprüft und gegebenenfalls separat beantwortet werden. Außerdem muss der Router in der Lage sein, das erste

Überprüfungspaket zur nächsten Komponente weiterleiten zu können. Dafür wird dieses Überprüfungspaket wiederum dupliziert.

8.2.2. Netzwerkschnittstelle

Die NI muss für die Unterscheidung von permanenten und von transienten Fehlern in der Lage sein, zwei Überprüfungspakete versenden zu können. Außerdem muss die NI beide Überprüfungspakete separat auf Fehler untersuchen können. Sollte kein Fehler aufgetaucht sein, dann wird ein Antwortpaket an die Komponente zurückgesendet.

Dynamische dreifach modulare Redundanz

Bisher werden defekte Verbindungen durch eine abfragebasierte Überprüfung detektiert und lokalisiert (Kapitel 6). In diesem Kapitel wird eine weitere Variante der Fehlererkennung diskutiert, die dreifach modulare Redundanz.

9.1. Konzept

Durch die redundante Anbindung der IP-Cores ist es möglich eine dreifach modulare Redundanz (TMR) umzusetzen. Bei der TMR werden die Berechnungen auf drei homogenen IP-Cores durchgeführt. Die Ergebnisse der drei IP-Cores werden dann mittels eines Mehrheitsentscheiders untersucht, welches Ergebnis richtig ist. Durch diese Erweiterung wird die Zuverlässigkeit des Systems weiter erhöht, denn eine Berechnung wird immer auf drei IP-Cores ausgeführt. Wenn nun ein IP-Core oder die Verbindung zu oder von diesem defekt ist, wird dieser Defekt direkt erkannt. Dennoch ist, sofern die Verbindungen der anderen beiden IP-Cores nicht defekt sind, ein richtiges Ergebnis vorhanden.

Da mehrere IP-Cores an einem Switch angebunden sind, kann der Mehrheitsentscheider im Router untergebracht werden. Ab einer Verbindungsanzahl von vier IP-Cores zu einem Switch ist es möglich, dass, wenn ein IP-Core oder die Verbindung dorthin ausfällt, automatisch auf die anderen drei verbleibenden IP-Cores ausgewichen wird. Dadurch erhält man ein dynamisches TMR (DTMR). Sollten nun ein weiterer IP-Core oder deren Verbindungen ausfallen, so wird das Datenpaket, ohne DTMR, an nur einen IP-Core gesendet. Bei dieser Arbeit wird nur ein lokales DTMR untersucht. Dies heißt, dass, wenn nicht mindestens drei IP-Cores von dem gewünschten Switch aus zu erreichen sind, das Datenpaket nur an einen IP-Core gesendet wird. Es ist jedoch möglich, dass die Ports eines IP-Cores einbezogen werden, die nicht direkt mit dem Switch verbunden sind. Eine weitere mögliche Variante ist, dass IP-Cores verwendet werden, die nicht direkt an den entsprechenden Switch angeschlossen sind. Dies ist dann ein globales DTMR.

Bevor ein Datenpaket an die drei entsprechenden IP-Cores weitergeleitet wird, muss das Datenpaket auf Fehler überprüft werden. Denn wenn ein DTMR-Datenpaket mit einem Fehler bei dem NI eintrifft, dann wird kein Datenpaket mit der Antwort zurückgesendet. Wenn jedoch ein DTMR-Datenpaket schon defekt bei dem Switch eintrifft und nicht untersucht wird, dann würde der Switch kein Datenpaket mit einer Antwort zurück erhalten.

Zu einem Zeitpunkt kann nur ein DTMR-Paket an die IP-Cores gesendet werden. Dies heißt, wenn ein anderes DTMR-Paket für diese Ports ankommt, dann wird es noch nicht bearbeitet bis die Antworten der IP-Cores eingetroffen sind oder die definierte Zeit abgelaufen ist. Jedoch werden DTMR-Pakete, die nicht für diese IP-Cores gedacht sind, weitergeleitet.

Die Synchronisation der Datenpakete, die von den IP-Cores zurückkommen, wird mittels der FIFOs bewerkstelligt. Sobald die Datenpakete zu den IP-Cores gesendet werden, startet ein Zähler. Wenn nun die definierte Zeit abgelaufen ist und noch nicht alle Pakete von den IP-Cores zurückgekommen sind, dann werden die Ports von denen noch keine Antwort gekommen ist, als defekt markiert. Sollten die Pakete alle in der vorgegebenen Zeit eintreffen, dann werden die Pakete Flit für Flit verglichen. Sollte dabei ein Fehler erkannt werden, dann wird der entsprechende Port als defekt markiert und bei den restlichen Paketen wird eine Überprüfung der CRC-Bits durchgeführt. Sollten weniger als drei Pakete zurückgekommen sein, dann startet bei den empfangenen Pakete eine Überprüfung der CRC-Bits. Bei den nicht empfangenen Pakete wird der entsprechende Port in der lokalen Defekttabelle als defekt markiert.

Der Algorithmus, wie das DTMR abläuft, ist auch in der Abbildung 9.1 zu sehen. Durch das DTMR wird das Paketaufkommen zwischen den Switchen und den IP-Cores deutlich erhöht. Wenn es ein globales DTMR ist, dann wird das gesamte Paketaufkommen entsprechend erhöht.

Diese Erweiterung ist aufbauend auf der Implementierung aus dem Kapitel 7.

Die Gattersimulationen, um diese Erweiterung zu validieren, ist im Abschnitt 10.4 zu finden.

9.2. Erweiterungen für die dynamische dreifach modulare Redundanz

9.2.1. Flittypen

Um die Funktionalitäten, die bis einschließlich dem Kapitel 6 implementiert wurden, erhalten zu können, werden zwei weitere Flittypen definiert. Durch diese zwei weiteren Flittypen kann das System auch ohne DTMR weiterarbeiten. Diese zwei Flittypen sind Kopfflits der DTMR-Pakete. Dabei ist eines das Kopfflit, das zu den drei unterschiedlichen IP-Cores gesendet werden soll. Der zweite Flittyp kennzeichnet die Antwort der IP-Cores, die an den Switch zurückgesendet wird. Beide Kopfflits eines DTMR-Paketes besitzen den

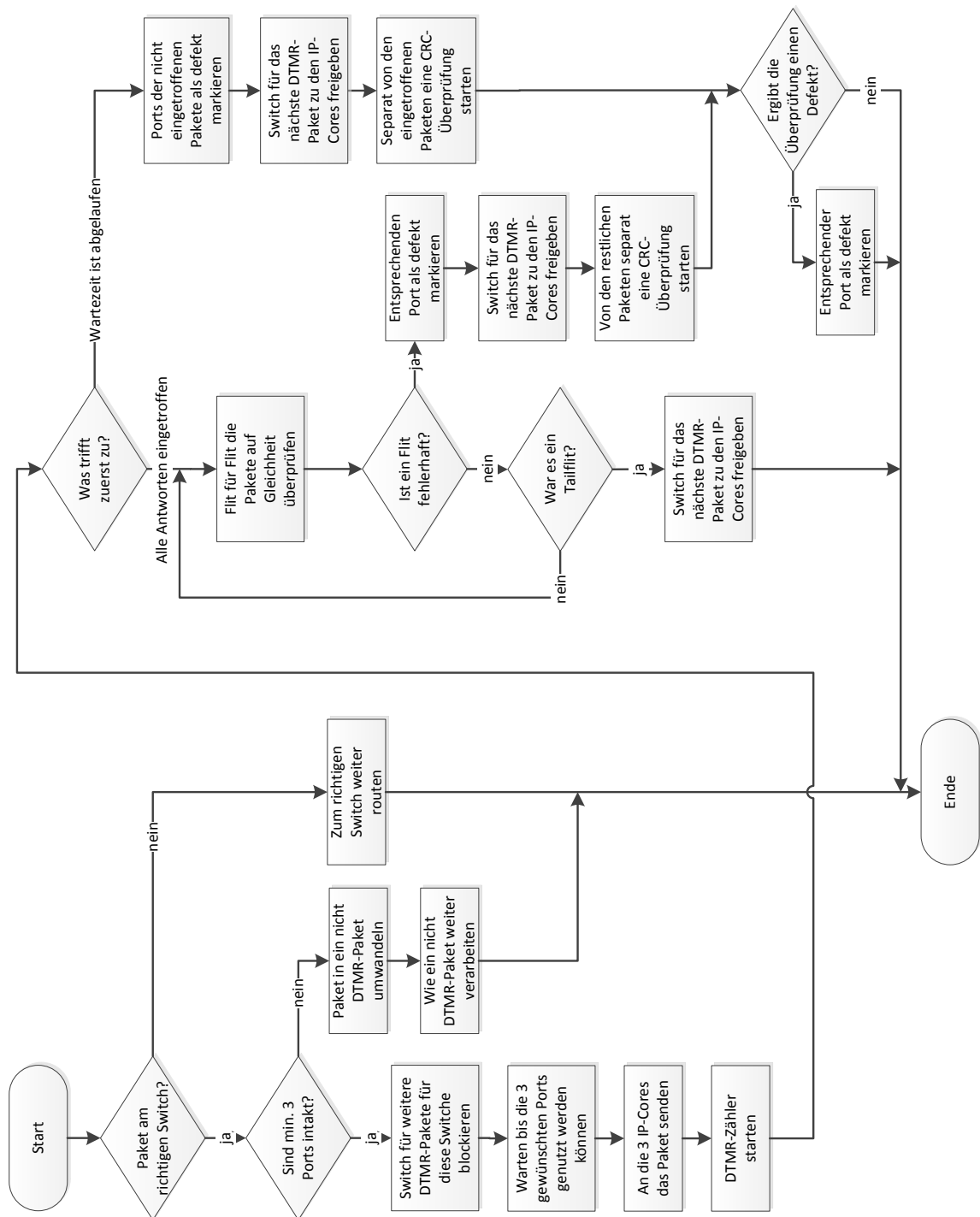


Abbildung 9.1.: Algorithmus des DTMR eines Switches

gleichen Aufbau wie ein Kopfflit eines nicht DTMR-Paketes. Welche Bitkombinationen für die unterschiedlichen Typen vergeben sind, kann der Tabelle 9.1 entnommen werden.

Tabelle 9.1.: Flittypen (mit dynamischer dreifach modularer Redundanz)

Bitkombination der Steuerbits	Funktion
000	Invertiertes Überprüfungsflit (abfragebasierte Fehlererkennung) (Komponente zu Komponente)
001	Kopfflit (nicht DTMR-Paket)
010	Endeflit
011	Defektfrit (Ende zu Ende)
100	Datenflit
101	Kopfflit (Antwort eines DTMR-Paketes)
110	Kopfflit (DTMR-Paket)
111	Überprüfungsflit (abfragebasierte Fehlererkennung) (Komponente zu Komponente)

9.2.2. Router

Der Router muss dahingehend erweitert werden, dass er die zusätzlichen Flittypen verarbeiten kann. Sofern nicht genügend IP-Cores mehr erreichbar sind, muss das DTMR-Paket in ein Datenpaket ohne DTMR umgewandelt werden. Des Weiteren muss noch ein Zähler und ein Mehrheitsentscheider für das DTMR eingebaut werden.

9.2.3. Netzwerkschnittstelle

Die NI wird dahingehend erweitert, dass die zusätzlichen Flittypen verarbeitet werden können. Außerdem ist sicherzustellen, dass das DTMR-Antwortpaket wieder den gleichen Port verwendet, in den das Paket hereingekommen ist. Hierfür wird beim Versenden eines DTMR-Antwortpaketes immer die Quelladresse untersucht. Es wird nicht die Zieladresse untersucht, denn in dem verwendeten IP-Core wird die Zieladresse mit der Quelladresse vertauscht.

Validierung des implementierten Systems

In diesem Kapitel werden alle implementierten Konzepte mittels Simulationen validiert.

Bei diesen Gatterebenenimulationen wird immer der folgende Simulationsalgorithmus verwendet. Außerdem sind die Verbindungen zwischen den Komponenten alle unidirektional und es werden immer nach und nach alle Verbindungen defekt. Um geglättete Kurven zu erhalten, werden je 50 Simulationswiederholungen durchgeführt. Aus deren Ergebnis wird dann das Mittel bestimmt.

10.1. Simulationsalgorithmus für die Validierung

Bei diesem Algorithmus wird eine Verbindung mit einem Defekt belastet. Dann sendet jeder IP-Core an jeden anderen IP-Core ein Datenpaket. Wenn ein Paket richtig empfangen wird, dann wird dies gezählt. Mit diesen Paketen kann nun überprüft werden, wie viele der IP-Cores noch zu erreichen sind. Wenn alle Pakete an alle IP-Cores gesendet wurden, dann werden die Paketzähler ausgelesen und es wird eine weitere Verbindung mit einem Defekt belastet. Dann startet erneut die Überprüfung, wie viele IP-Cores noch zu erreichen sind. Dieser Ablauf wird solange ausgeführt, bis alle Verbindungen defekt sind. Dieser Ablauf der Simulation ist auch in der Abbildung 10.1 zu sehen.

10.2. Gatterebenenimulation mit Fehlerdetektierung, Fehlerlokalisierung und redundanter Anbindung

In der Abbildung 10.2 ist das Ergebnis der Simulation des implementierten Systems zu sehen. Bei dieser Simulation ist eine Fehlerdetektierung, eine Fehlerlokalisierung und eine redundante Anbindung vorhanden.

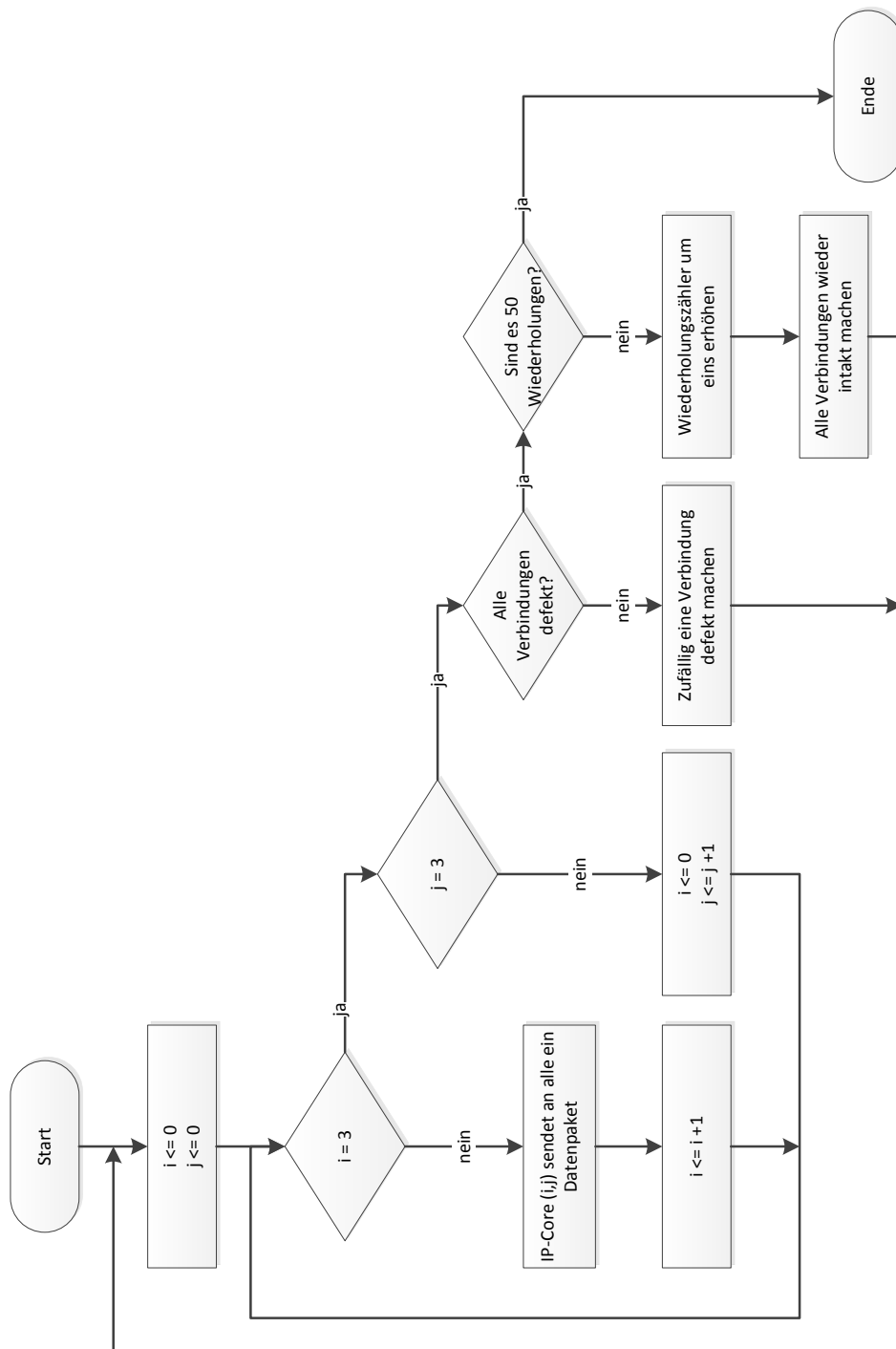


Abbildung 10.1.: Algorithmus zum Bestimmen der globalen Erreichbarkeit (Gatterebensimulation)

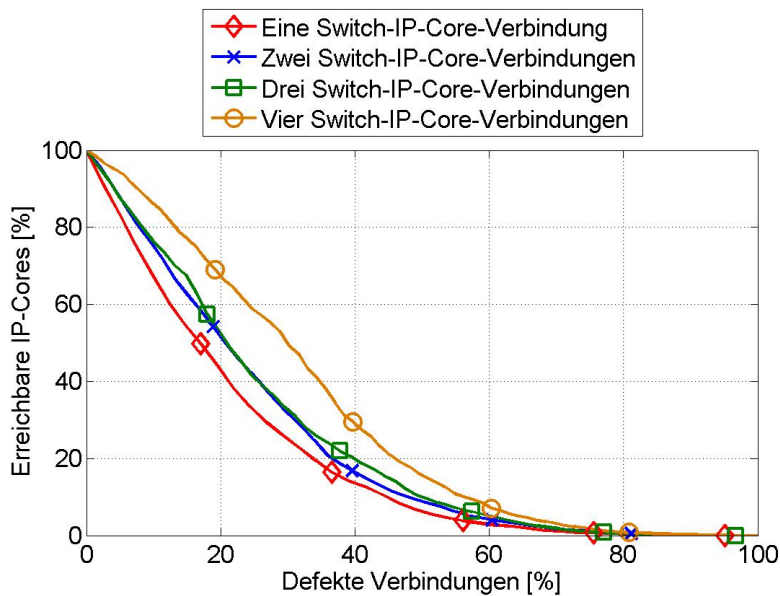


Abbildung 10.2.: Gatterebensimulation eines 3×3 -Gitters mit redundanter Anbindung bei dem alle Verbindungen ausfallen

Im Vergleich zu der äquivalenten numerischen Simulation (Abbildung 4.9) ist der Kurvenverlauf dieser Simulation vergleichbar. Dadurch, dass es kein idealer Routingalgorithmus ist, sind die Kurven des implementierten Systems entsprechend schlechter. Außerdem ist noch zu beachten, dass bei einem unidirektionalen Defekt relativ einfach ein bidirektionaler Fehler markiert wird (Abschnitt 6.2).

Bei Ausfall von 20% der Verbindungen gibt es zwischen der einfachen und der vierfachen Switch-IP-Core-Anbindung einen Unterschied von 24 Prozentpunkten. Die vergleichbare numerische Simulation besitzt bei diesem Punkt eine Differenz von 29 Prozentpunkten.

Die $MTTF_{Sim}$ -Werte für die äquivalente numerische Simulation und der Gatterebensimulation sind in der Tabelle 10.1 zu finden. Hierbei ist zu sehen, dass bei der einfachen Anbindung ein Unterschied von 16,18% und bei der vierfachen Anbindung von 32,67% besteht.

10.2.1. Mit Weiterleitung zu anderen IP-Cores

Bei dieser Gatterebensimulation wird das Datenpaket, falls es den IP-Core nicht mehr erreichen kann, an einen anderen IP-Core weitergeleitet. Das Resultat ist in der Abbildung 10.3 zu sehen. Die Weiterleitung ist so jedoch nur möglich, wenn alle IP-Cores homogen zueinander sind.

Tabelle 10.1.: Vergleich der $MTTF_{Sim}$ (in Jahren) beim Ausfall aller Verbindungen (unidirektional, 3x3-Gitter)

Art der Simulation	Anzahl der Switch-IP-Core-Verbindungen			
	1	2	3	4
numerisch	6,08	10,93	14,01	19,47
Gatterebene	5,09	7,23	8,66	13,11

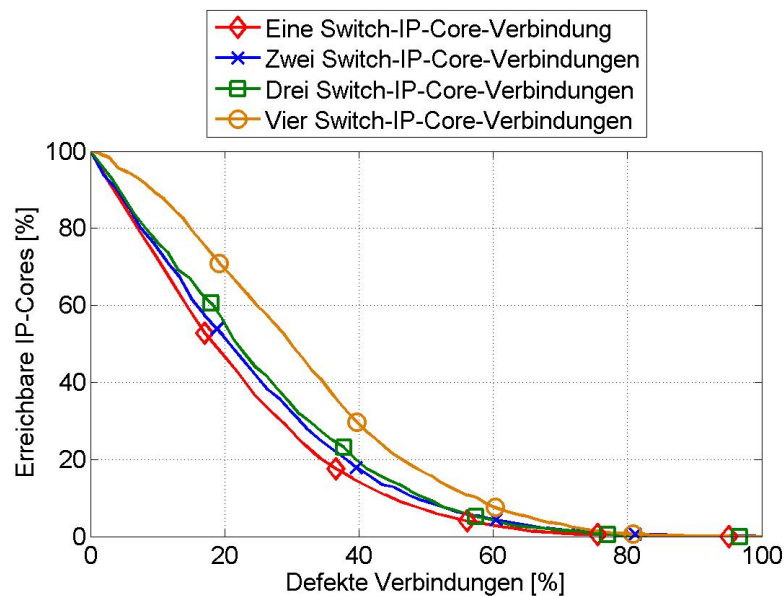


Abbildung 10.3.: Gatterebenen simulation eines 3x3-Gitters mit redundanter Anbindung und Weiterleitung zu anderen IP-Cores bei dem alle Verbindungen ausfallen

Vergleich zwischen mit und ohne Weiterleitung zu anderen IP-Cores

In der Abbildung 10.4 sind die Kurven für eine einfache Anbindung mit und ohne Weiterleitung zu anderen IP-Cores zu sehen. In der Abbildung 10.5 sind die Kurven für eine vierfache Anbindung aufgetragen. Bei beiden Abbildungen ist zu sehen, dass die Ergebnisse der Simulationen nicht stark voneinander abweichen.

Die Differenzen der Erreichbarkeiten zwischen der einfachen und der vierfachen Anbindung sind bei beiden Simulationen 24 Prozentpunkte.

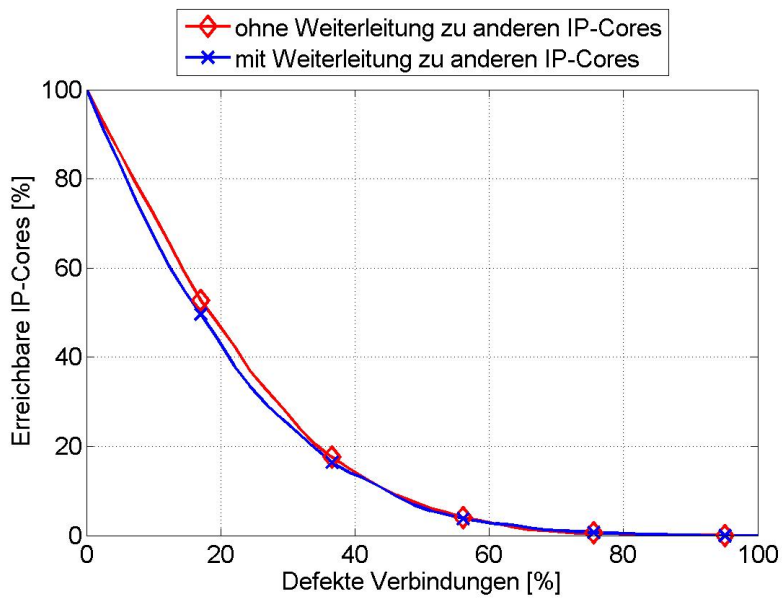


Abbildung 10.4.: Mit und ohne Weiterleitung zu anderen IP-Cores bei einer einfachen Switch-IP-Core-Anbindung

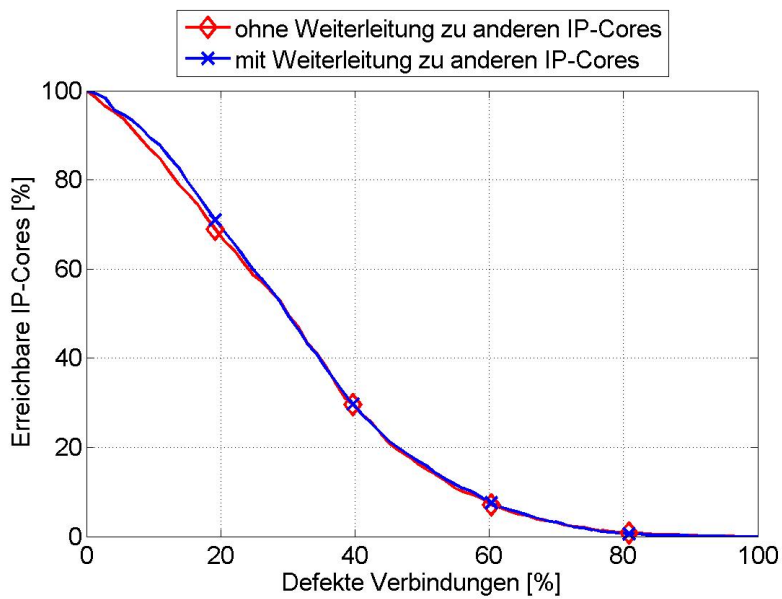


Abbildung 10.5.: Mit und ohne Weiterleitung zu anderen IP-Cores bei einer vierfachen Switch-IP-Core-Anbindung

10.3. Gatterebensimulation mit transienten Fehlern

In den Abbildungen 10.6 bis 10.9 sind die Ergebnisse der Gatterebensimulationen mit transienten Fehlern zu sehen. Die transienten Fehler kommen immer in entsprechendem Abstand zu den permanenten Fehlern, vergleichbar mit den numerischen Simulationen. Zwischen den numerischen Simulationen und den Simulationen des implementierten Systems gibt es einen Unterschied, denn bei den Gatterebensimulationen werden die Fehler nur für eine kurze Zeit eingepreßt, und nicht wie bei den numerischen Simulation für einen gesamten Durchlauf. Hierdurch ist zu erklären, warum die Knicke bei den Gatterebensimulationen nicht so extrem sind. Durch die Gatterebensimulationen werden die numerischen Simulationen bestätigt, wenn der implementierte Routingalgorithmus entsprechend beachtet wird.

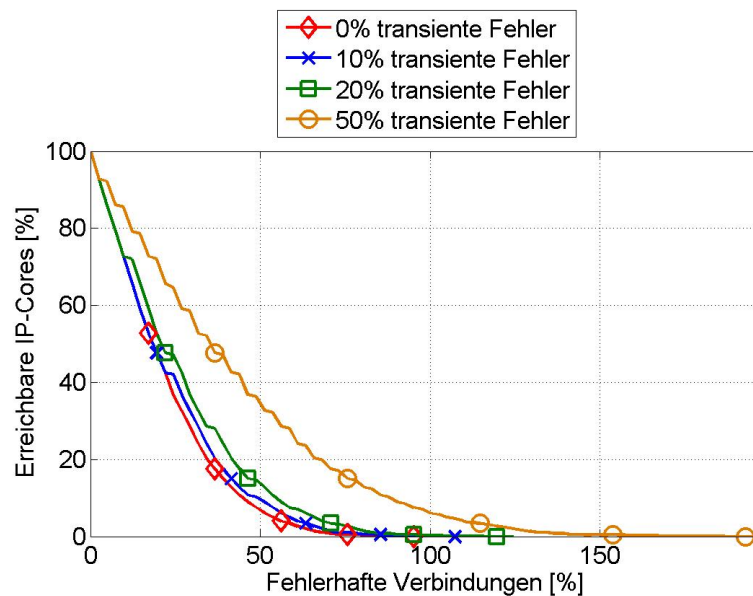


Abbildung 10.6.: Gatterebensimulation eines 3x3-Gitters mit einer einfachen Switch-IP-Core-Anbindung und transienten Fehlern

10.4. Gatterebensimulation mit dynamischer dreifach modularer Redundanz

Bei diesen Simulationen wird die dynamische dreifach modulare Redundanz untersucht. Das Resultat bei einer dreifachen Switch-IP-Core-Anbindung ist in der Abbildung 10.10 zu sehen. Das Ergebnis bei einer vierfachen Switch-IP-Core-Anbindung ist in der Abbildung 10.11 zu sehen.

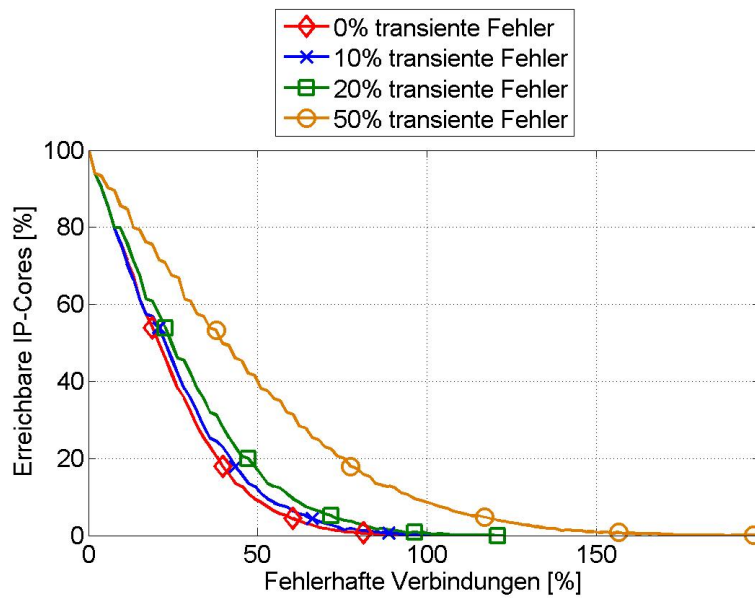


Abbildung 10.7.: Gatterebensimulation eines 3x3-Gitters mit einer zweifachen Switch-IP-Core-Anbindung und transienten Fehlern

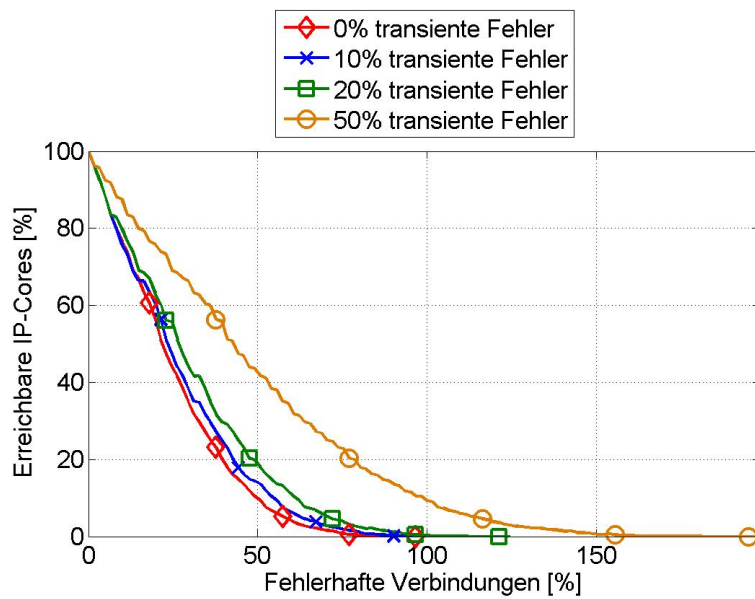


Abbildung 10.8.: Gatterebensimulation eines 3x3-Gitters mit einer dreifachen Switch-IP-Core-Anbindung und transienten Fehlern

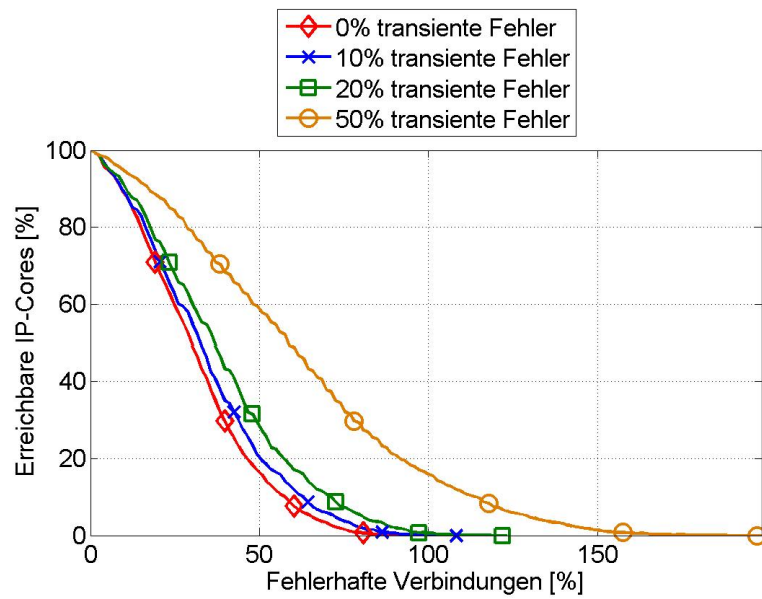


Abbildung 10.9.: Gatterebensimulation eines 3x3-Gitters mit einer vierfachen Switch-IP-Core-Anbindung und transienten Fehlern

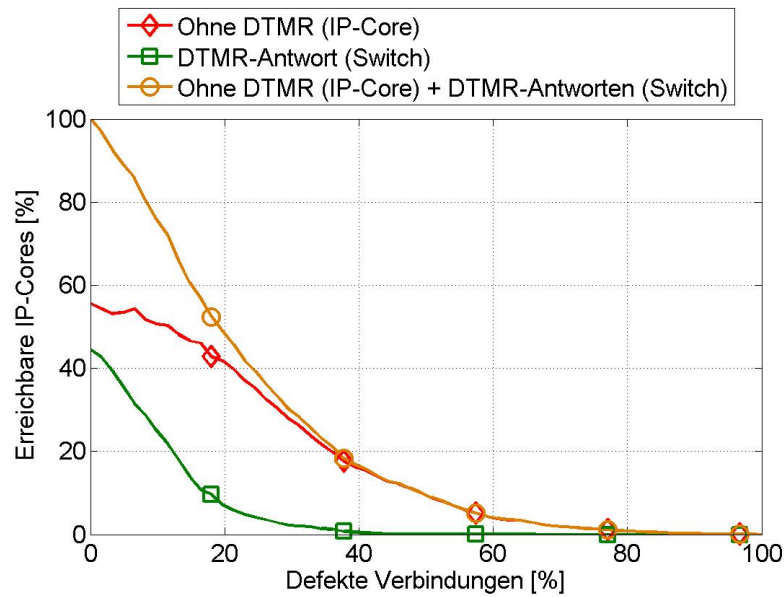


Abbildung 10.10.: Gatterebensimulation eines 3x3-Gitters mit einer dreifachen Switch-IP-Core-Anbindung und der dynamischen dreifach modularen Redundanz

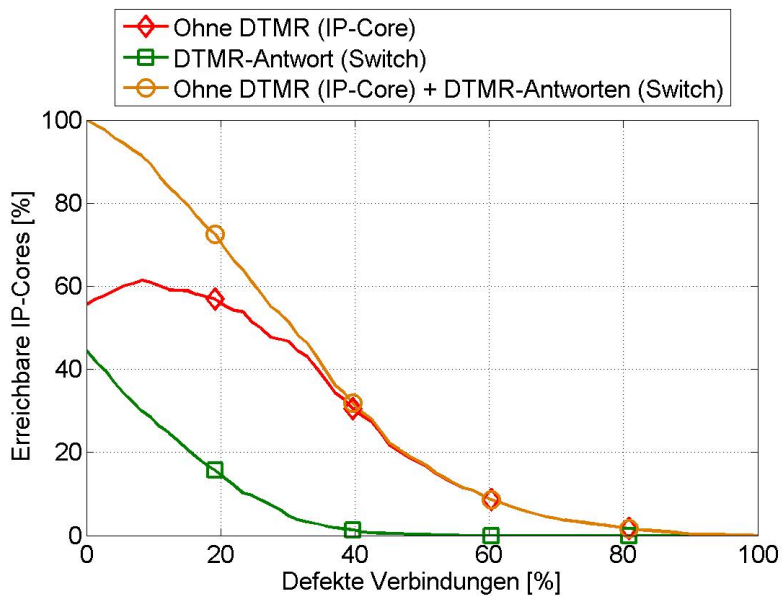


Abbildung 10.11.: Gatterebensimulation eines 3×3 -Gitters mit einer vierfachen Switch-IP-Core-Anbindung und der dynamischen dreifach modularen Redundanz

Da nicht jeder Switch mit mindestens drei IP-Cores verbunden ist, können nicht alle Switches ein DTMR ausführen. Diese Switches versenden dann an einen IP-Core ein Paket ohne DTMR. Diese Pakete werden dann in den IP-Cores separat gezählt. Wenn die Antworten von den IP-Cores auf ein DTMR-Paket in dem Switch ankommen sind, dann wird eine Mehrheitsentscheidung gemacht. Ergibt dieser ein richtiges Ergebnis dann wird dies im Switch gezählt. Da nur vier der neun Switches genügend Switch-IP-Core-Verbindungen haben um DTMR-Pakete an die IP-Cores versenden können, kann erklärt werden, warum die Kurve der DTMR-Antwort bei 44 % ($4/9$) und die Kurve ohne DTMR bei 56 % ($5/9$) beginnt. Die dritte Kurve ist die Addition der beiden anderen Kurven. Der Anstieg der Kurve ohne DTMR, bei der vierfachen Switch-IP-Core-Anbindung, ist dadurch zu erklären, dass die Switches nicht mehr genügend IP-Cores für das DTMR zur Verfügung haben und hierdurch alternativ an einen IP-Core ein Paket ohne DTMR versenden. Dieser Anstieg ist bei der dreifachen Switch-IP-Core-Anbindung nur durch einen flacheren Abfall zu sehen.

Durch eine Torustopologie (Abschnitt 2.2.1) könnte verhindert werden, dass nicht jeder Switch genügend Switch-IP-Core-Verbindungen zur Verfügung hat. Dadurch könnten alle Switches ein DTMR ausführen.

Zusammenfassung und Ausblick

Bei dem untersuchten Network-on-Chip spannen die Switche ein regelmäßiges Gitter auf. An jedem dieser Switche ist ein IP-Core angebunden. Durch die zuvor noch nicht umgesetzte redundante Anbindung der IP-Cores an die Switche kann die Zuverlässigkeit erhöht werden.

Was mittels der redundanten Anbindung der IP-Cores maximal an Erreichbarkeit machbar ist, wurde in den numerischen Simulationen untersucht. Bei den Simulationen, die betrachtet werden, fallen immer die Switch-Switch-Verbindungen und die Switch-IP-Core-Verbindungen aus. Die Simulationen mit der Erweiterung der redundanten Anbindung der IP-Cores hat bei einem Ausfall von 20 % der Verbindungen (bidirektional) und einer Gittergröße von 3×3 eine Verbesserung zwischen der einfachen und der vierfachen Anbindung eine Differenz von 34 Prozentpunkten. Bei einer Gittergröße von 20×20 sind es 35 Prozentpunkte. Diese Simulationen werden auch für unidirektionale Verbindungen durchgeführt bei einem 3×3 -Gitter erhält man eine Differenz von 29 Prozentpunkten und bei einem 20×20 -Gitter sind es 35 Prozentpunkte. Die numerischen Simulationen zeigen, für die redundante Anbindung der IP-Cores, ein hohes potential um die Zuverlässigkeit des Systems zu steigern.

Für diese Simulationen werden exemplarisch die MTTF bestimmt. Dabei ist zu sehen, dass die MTTF der vierfachen Anbindung im Vergleich zur einfachen Anbindung beim Ausfall aller Verbindungen und bei allen Gittergrößen um den Faktor 3,2 größer ist.

Motiviert durch diese Simulationen wurde ein vergleichbares System auf Gatterebene implementiert. Die Gittergröße des Systems beträgt 3×3 . Um die redundante Anbindung umzusetzen, werden zusätzliche Ports hinzugefügt. In diesem Zuge wird auch der Routingalgorithmus erweitert. Außerdem wird ein zweistufiger Algorithmus zur Fehlererkennung implementiert.

Im ersten Schritt werden die Datenpakete im Ziel immer auf Richtigkeit überprüft. Wenn ein Fehler aufgetreten ist, wird ein Paket an die Quelle zurückgesendet. Hier wird eine abfragebasierte Überprüfung gestartet. Diese Überprüfung findet von Komponente zu Komponente statt. Für die Überprüfung der Pakete wird eine CRC-Einheit verwendet. Wenn ein Fehler lokalisiert wurde, dann wird er in einer lokalen Defekttabelle markiert.

Hiermit ist der entsprechende Port als defekt markiert. Die Simulationen auf Gatter ergibt beim Ausfall von 20 % der Verbindungen eine Verbesserung zwischen der einfachen und der vierfachen Anbindung von 24 Prozentpunkten.

Auf diesem System aufbauend wurden zwei zusätzliche Erweiterungen hinzugefügt. Bei der ersten Erweiterung wird zwischen permanenten und transienten Fehlern unterschieden. Neben dem implementierten System wurde dies auch in einer vergleichbaren numerischen Simulation umgesetzt. Die zweite zusätzliche Erweiterung ist ein neues erweitertes Konzept des TMR. Dieses Konzept kann erst durch die redundante Anbindung der IP-Cores an die Switch umgesetzt werden. Bei diesem Konzept verwaltet der Switch das TMR. Der Switch sendet das entsprechende Paket an drei homogene direkt angebundene IP-Cores weiter. Die IP-Cores senden eine Antwort an den Switch zurück. Daraufhin folgt in dem Switch ein Vergleich der Pakete. Sollte ein IP-Core ausfallen, dann ist das TMR bei einer vierfachen Anbindung immer noch voll funktionsfähig. Durch diese Erweiterung können neben den Verbindungen auch die IP-Cores selbst auf Fehler untersucht werden.

In weiteren Arbeiten kann das implementierte System zum Beispiel durch einen Code zur Fehlerkorrektur erweitert werden. Hierdurch kann sichergestellt werden, dass möglichst alle empfangenen Pakete verwendet werden. Außerdem kann der erweiterte Routingalgorithmus weiter verbessert werden. Zum Beispiel, dass immer der nächstgelegene Port eines IP-Cores verwendet wird oder, wenn bei der Überprüfung einer Verbindung ein Port als defekt markiert ist, ein alternativer Port verwendet wird.

Das TMR kann dahingehend erweitert werden, dass es nicht nur auf die IP-Cores beschränkt wird, die direkt an dem jeweiligen Switch direkt angebunden sind. Sondern, dass die Berechnungen auf allen homogenen IP-Cores des Systems durchgeführt werden können.

Weitere Ergebnisse von der Bestimmung der maximalen Metrik des Systems

A.1. Bei Ausfällen von gesamten Komponenten

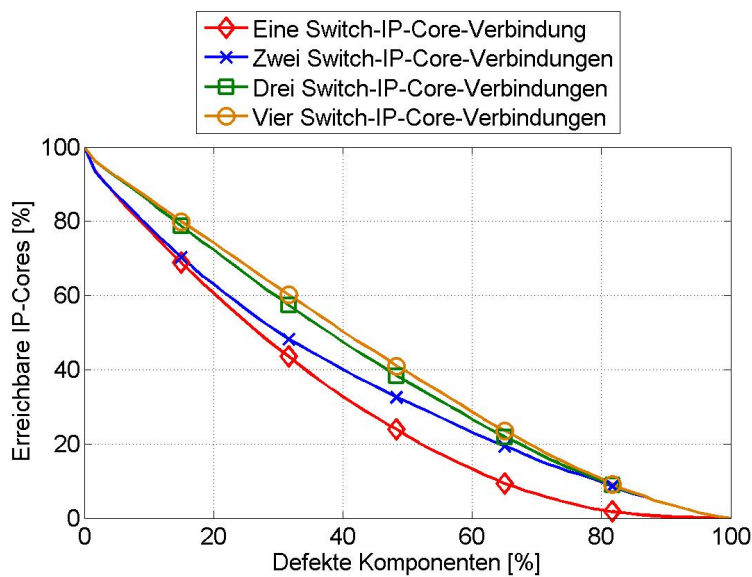


Abbildung A.1.: Numerische Simulation eines 5x5-Gitters mit redundanter Anbindung bei dem gesamte Komponenten ausfallen

A.2. Bei Ausfällen von Switch-Switch-Verbindungen

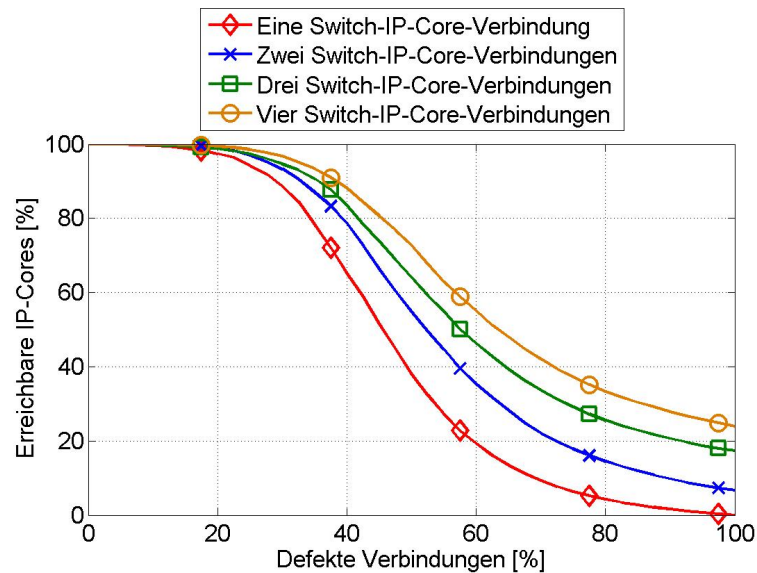


Abbildung A.2.: Numerische Simulation eines 5x5-Gitters mit redundanter Anbindung bei dem bidirektionale Switch-Switch-Verbindungen ausfallen

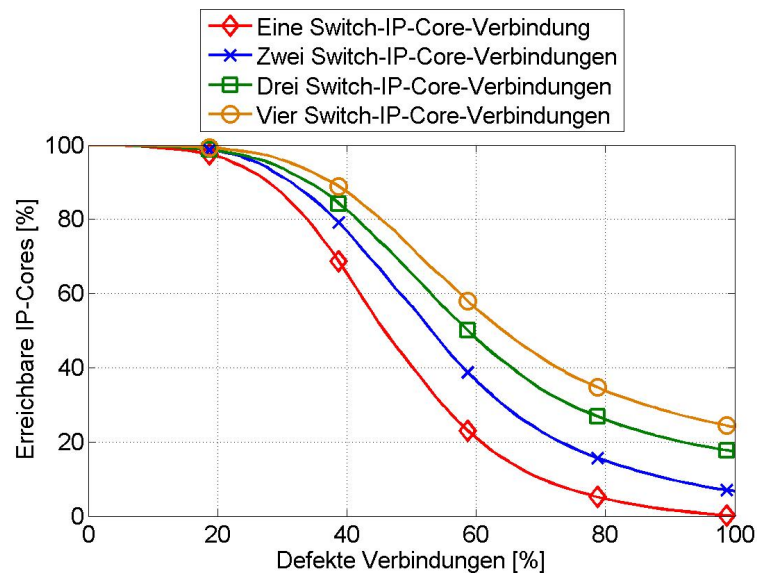


Abbildung A.3.: Numerische Simulation eines 5x5-Gitters mit redundanter Anbindung bei dem unidirektionale Switch-Switch-Verbindungen ausfallen

A.3. Bei Ausfällen von Switch-Switch- oder Switch-IP-Core-Verbindungen

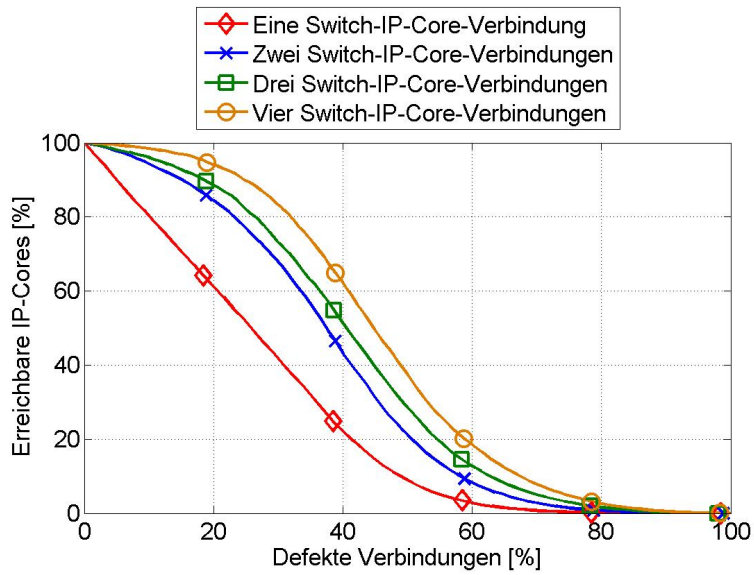


Abbildung A.4.: Numerische Simulation eines 5x5-Gitters mit redundanter Anbindung bei dem alle Verbindungen (bidirektional) ausfallen

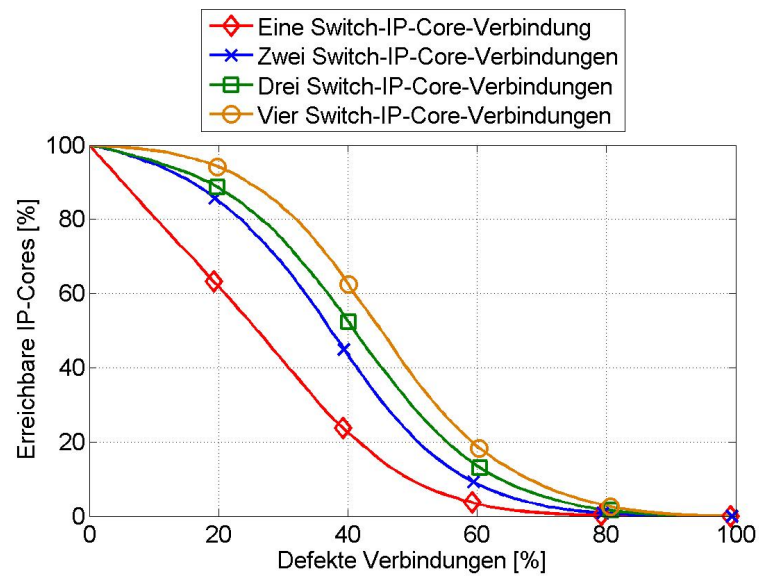


Abbildung A.5.: Numerische Simulation eines 5x5-Gitters mit redundanter Anbindung bei dem alle Verbindungen (unidirektional)

A.4. Bei Ausfällen von Switch-Switch- oder Switch-IP-Core-Verbindungen mit transienten Fehlern

A.4. Bei Ausfällen von Switch-Switch- oder Switch-IP-Core-Verbindungen mit transienten Fehlern

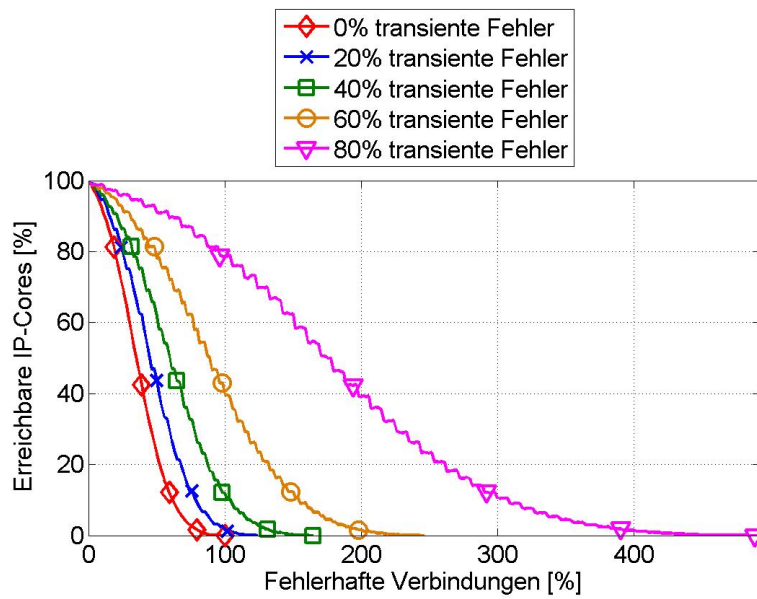


Abbildung A.6.: Numerische Transientensimulation eines 3x3-Gitters mit zwei Switch-IP-Core-Verbindungen

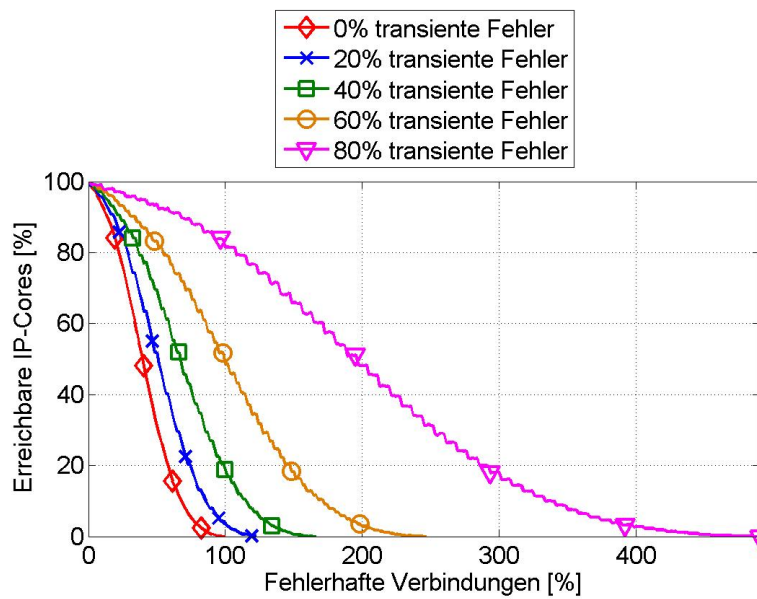


Abbildung A.7.: Numerische Transientensimulation eines 3x3-Gitters mit drei Switch-IP-Core-Verbindungen

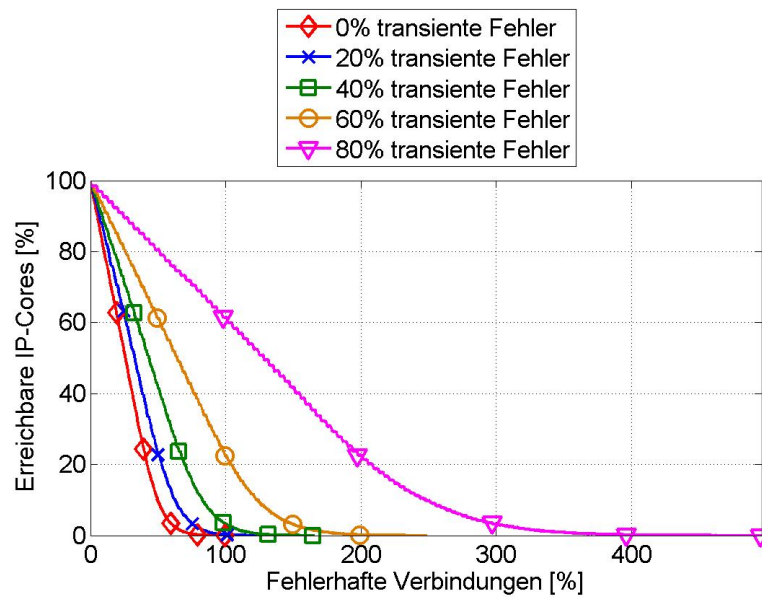


Abbildung A.8.: Numerische Transientensimulation eines 5x5-Gitters mit einer Switch-IP-Core-Verbindung

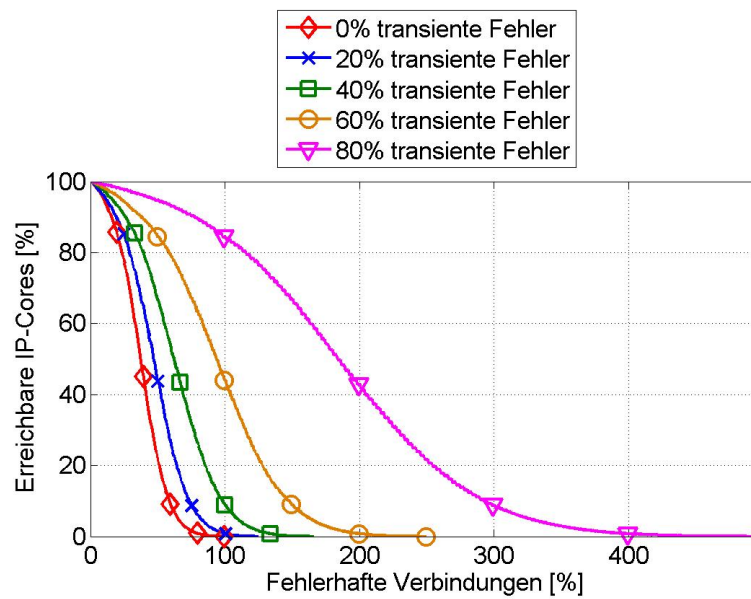


Abbildung A.9.: Numerische Transientensimulation eines 5x5-Gitters mit zwei Switch-IP-Core-Verbindungen

A.4. Bei Ausfällen von Switch-Switch- oder Switch-IP-Core-Verbindungen mit transienten Fehlern

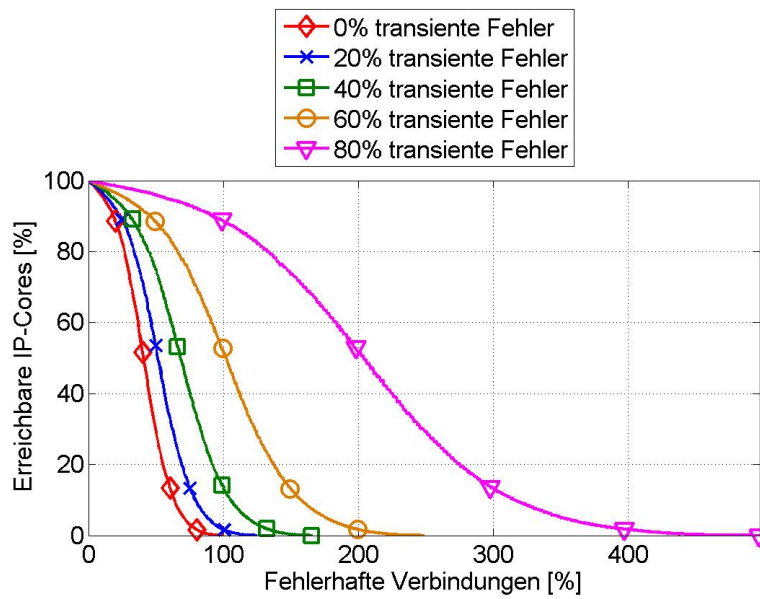


Abbildung A.10.: Numerische Transientensimulation eines 5x5-Gitters mit drei Switch-IP-Core-Verbindungen

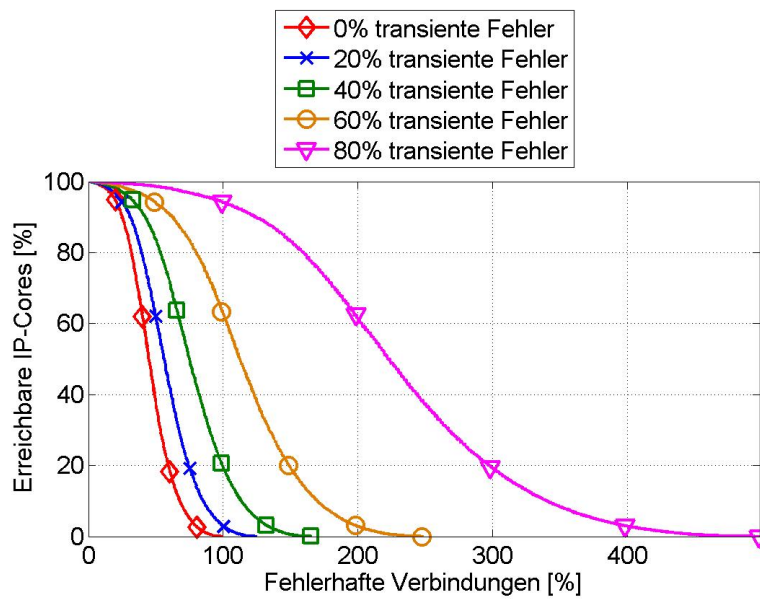


Abbildung A.11.: Numerische Transientensimulation eines 5x5-Gitters mit vier Switch-IP-Core-Verbindungen

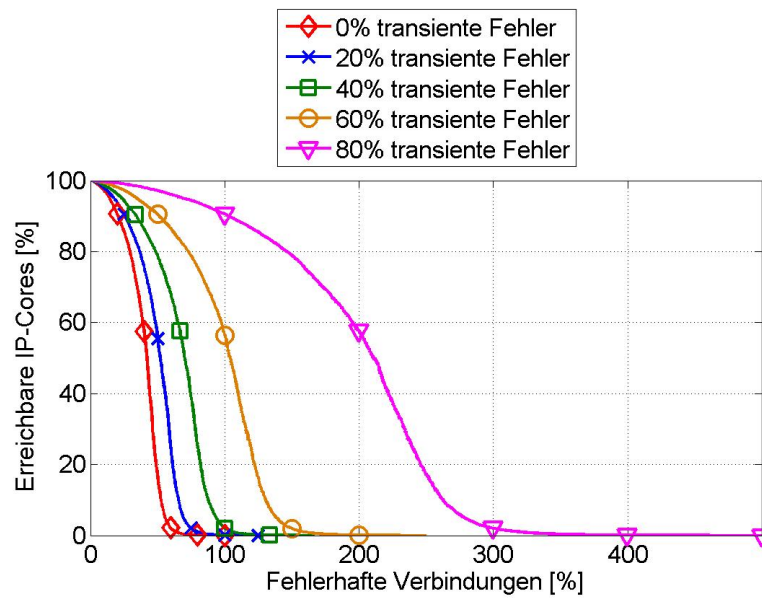


Abbildung A.12.: Numerische Transientensimulation eines 20x20-Gitters mit zwei Switch-IP-Core-Verbindungen

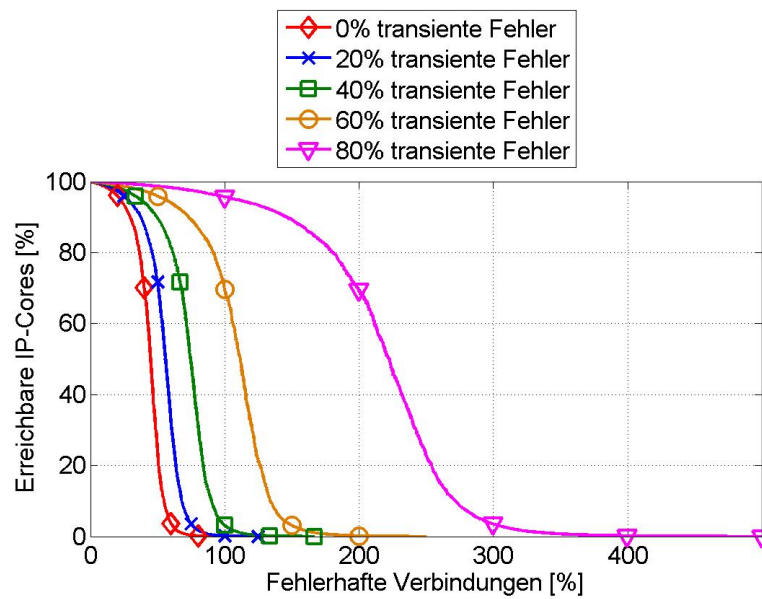


Abbildung A.13.: Numerische Transientensimulation eines 20x20-Gitters mit drei Switch-IP-Core-Verbindungen

Weitere Algorithmen

B.1. Weitere Routingalgorithmen (Komponente zu Komponente)

B.1.1. Routingalgorithmen inklusive der Erweiterung für das yx-Routing

Eine andere Erweiterung des Routingalgorithmus, die untersucht wurde, enthält einen zusätzlichen yx-Routingalgorithmus. Die Erweiterung wurde mittels des Verfahrens aus Abschnitt 6.4.5 auf Deadlockfreiheit untersucht. Dabei hat sich herausgestellt, dass diese Erweiterung nicht deadlockfrei ist.

Bei dieser Variante wird, wenn der Port eins oder drei defekt ist, alternativ an den Port zwei beziehungsweise vier weitergeleitet. Für einen Switch wird dann ein yx-Routing durchgeführt. Dies heißt, es wird erst in die y-Koordinate geroutet und dann in die x-Koordinate. Sollte der gewünschte Port eins beziehungsweise drei auch defekt sein, so wird wieder zum alternativen Port geroutet. Dies wird solange weiter durchgeführt, bis bei einem Switch der Port eins beziehungsweise drei nicht defekt ist oder der alternative Port ebenso defekt ist. Die Ports am Rand sind alle als defekt deklariert. Dieses Szenario ist in der Abbildung B.1 und Abbildung B.2 zu sehen.

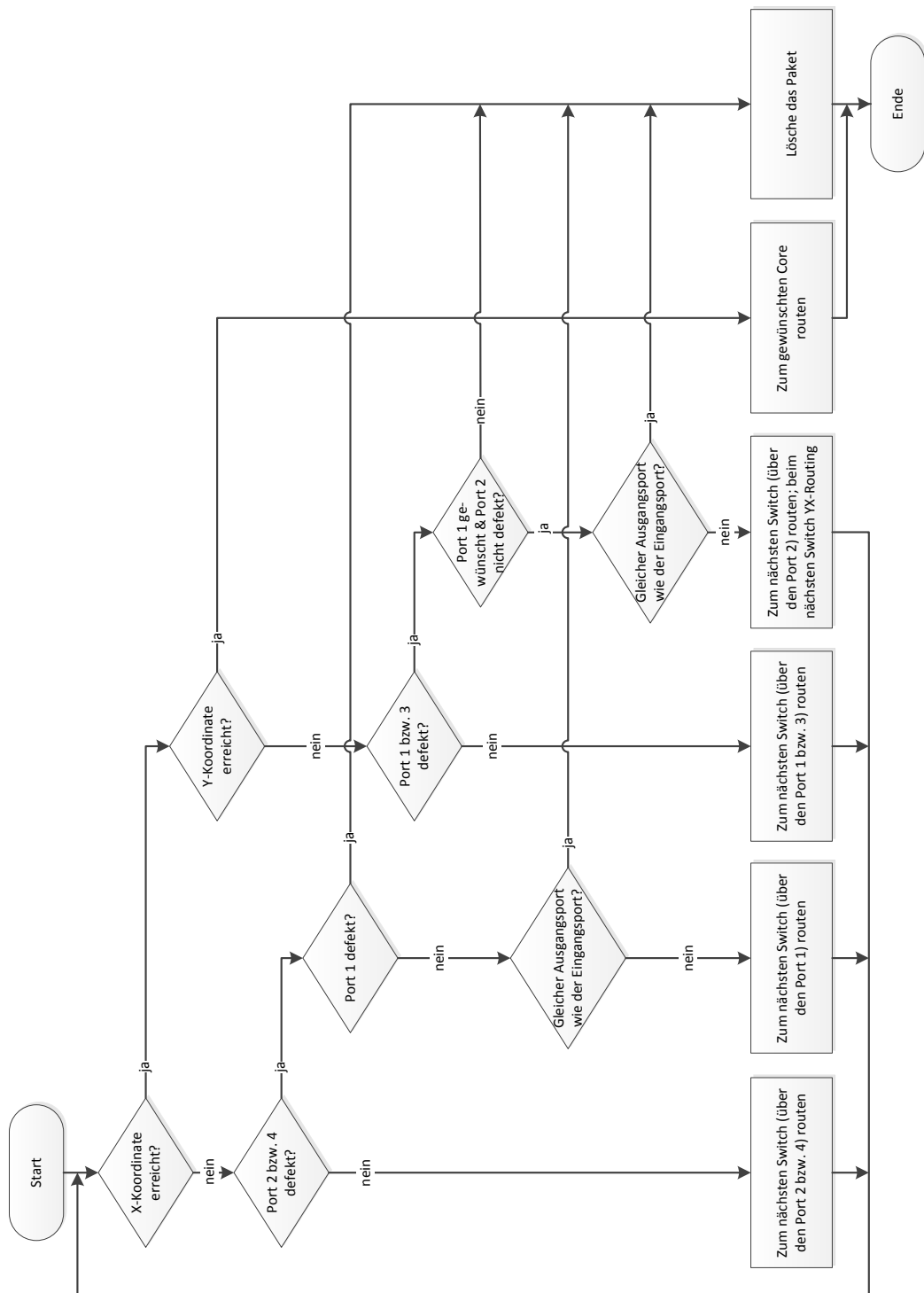


Abbildung B.1.: Xy-Routing mit einer Alternative und der Erweiterung des yx-Routings

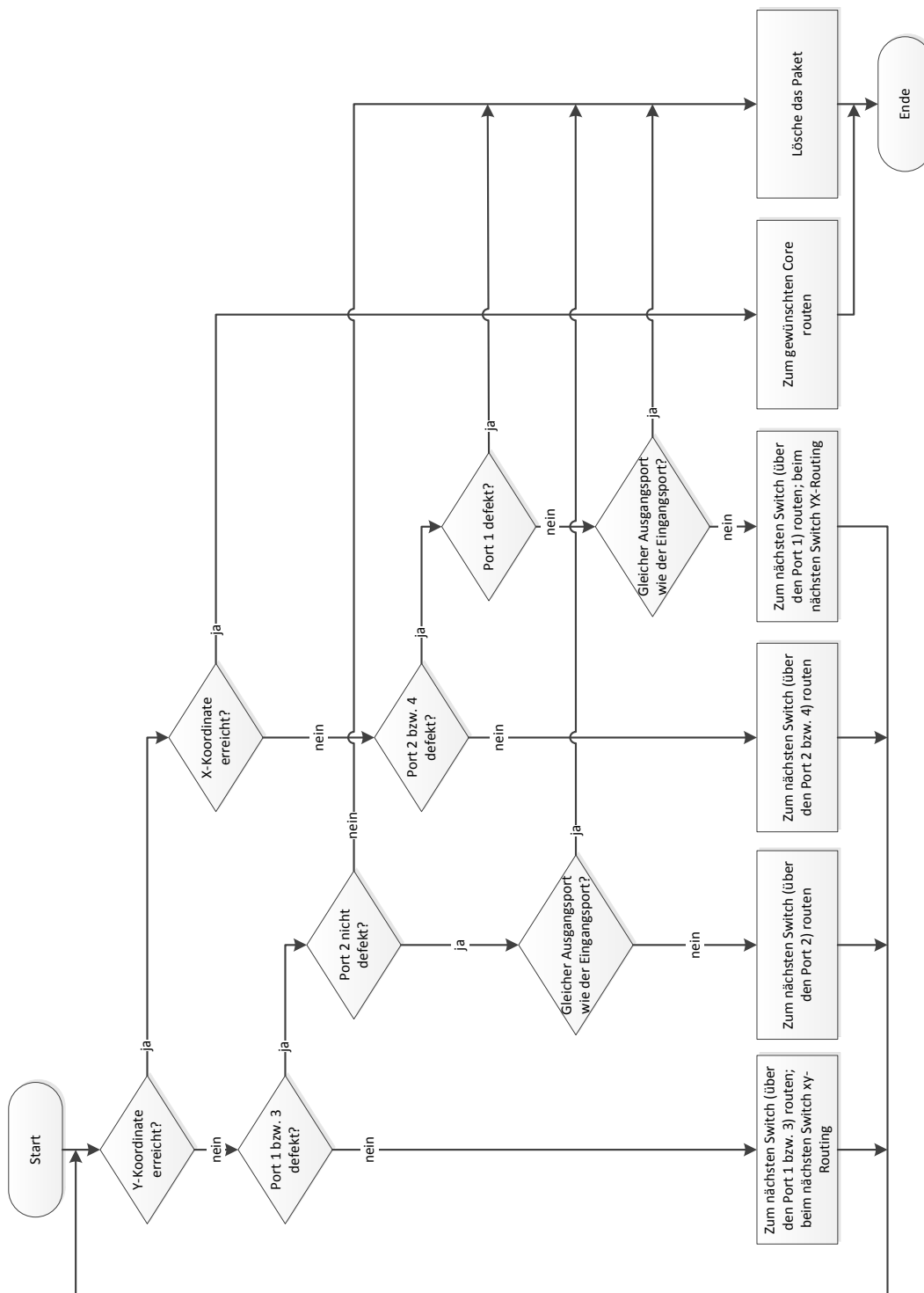


Abbildung B.2.: Yx-Routing mit einer Alternative

Weitere Konfigurationen für die redundante Anbindung

C.1. Zweidimensionales Gitter inkl. der redundanten Anbindung von IP-Cores

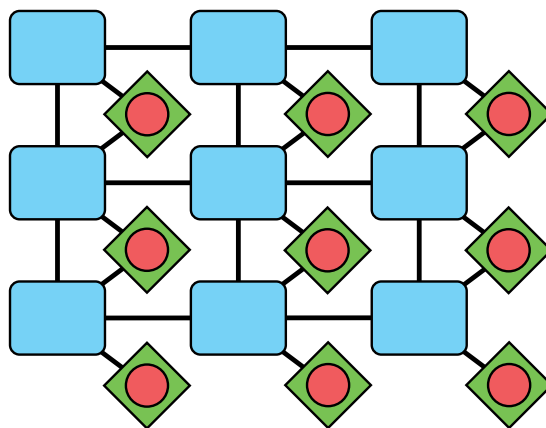


Abbildung C.1.: Zweidimensionales Gitter inkl. der redundanten Anbindung von IP-Cores
(zwei verbindbare Switche an einen IP-Core)

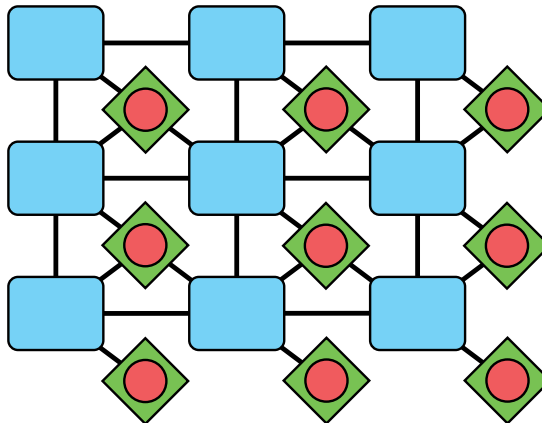


Abbildung C.2.: Zweidimensionales Gitter inkl. der redundanten Anbindung von IP-Cores (drei verbindbare Switches an einen IP-Core)

C.2. Switch

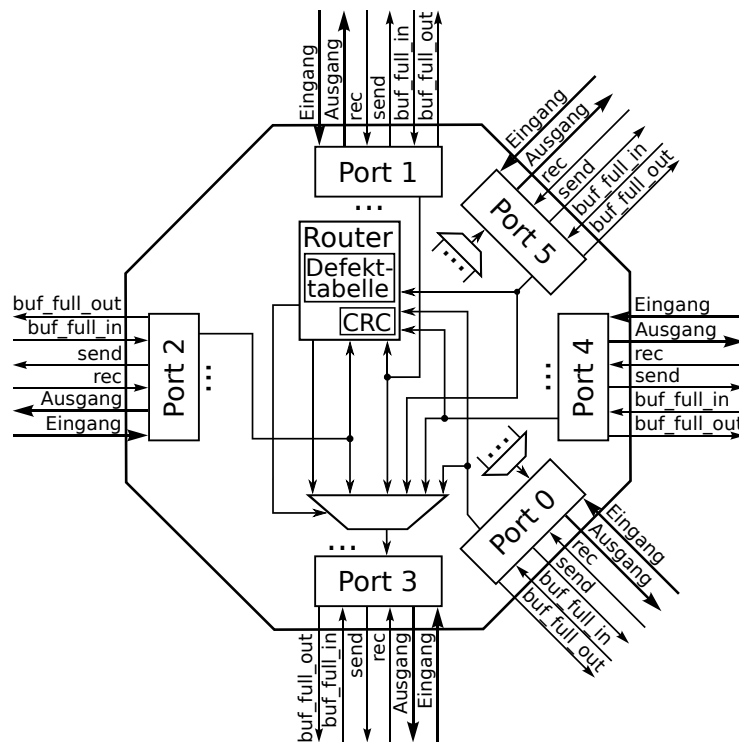


Abbildung C.3.: Architektur des erweiterten Switches inkl. der redundanten Anbindung (zwei verbindbare IP-Cores)

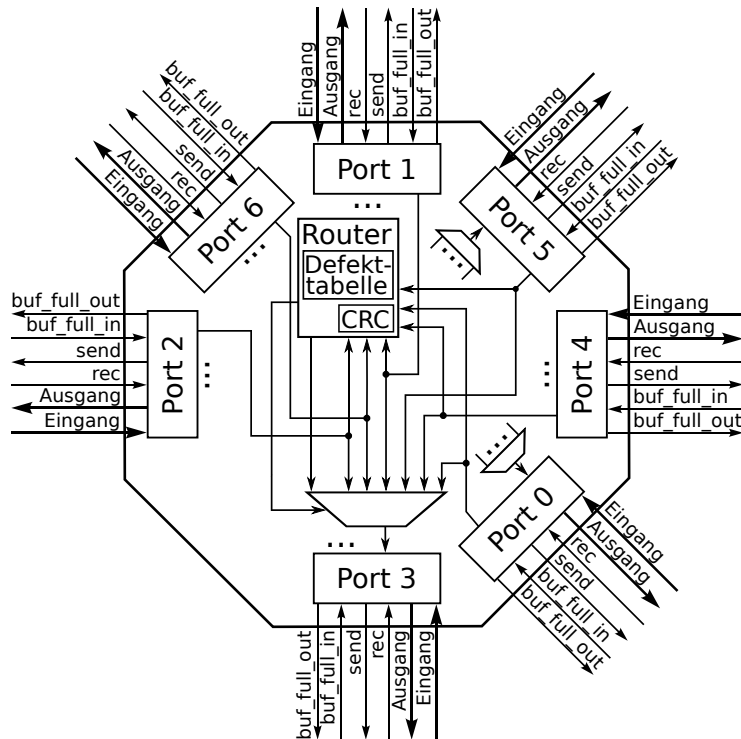


Abbildung C.4.: Architektur des erweiterten Switches inkl. der redundanten Anbindung (drei verbindbare IP-Cores)

C.3. Netzwerkschnittstelle

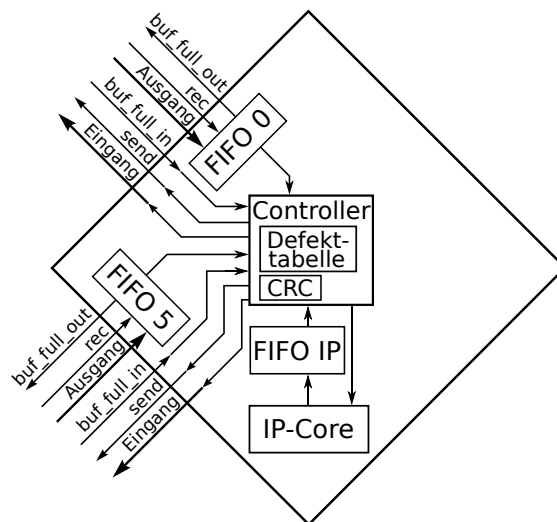


Abbildung C.5.: Architektur der erweiterten Netzwerkschnittstelle inkl. der redundanten Anbindung (zwei verbindbare Switches)

C. Weitere Konfigurationen für die redundante Anbindung

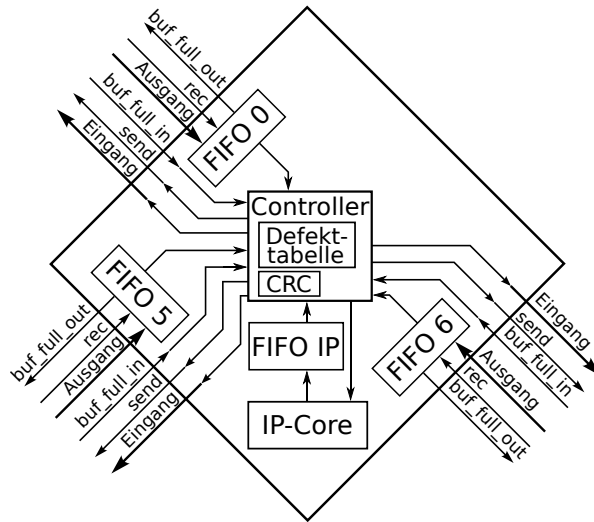


Abbildung C.6.: Architektur der erweiterten Netzwerkschnittstelle inkl. der redundanten Anbindung (drei verbindbare Switches)

Abbildungsverzeichnis

2.1.	Ringtopologie (nach [Chao5])	4
2.2.	Sterntopologie (nach [Scho5])	4
2.3.	Baumtopologie (nach [Scho5])	5
2.4.	Vollständig verbundene Topologie (nach [Scho5])	5
2.5.	Unregelmäßige Gittertopologie (nach [Chao5])	6
2.6.	Bustopologie (nach [Chao5])	6
2.7.	Regelmäßige Gittertopologie (nach [Chao5])	7
2.8.	Torustopologie (nach [Scho5])	8
2.9.	Gesamtkosten eines Systems in Abhängigkeit von seiner Zuverlässigkeit (nach [Hed84])	11
2.10.	Zeitlicher Verlauf der Ausfallrate („Badewannenkurve“) (nach [sim10])	13
2.11.	Skalierbare Leistung im Vergleich zur Systemverfügbarkeit (nach [Hwa00])	14
2.12.	Serienverschaltung der Komponenten R_1 und R_2 (nach [O’C90])	15
2.13.	Parallelverschaltung der Komponenten R_1 und R_2 (nach [O’C90])	16
2.14.	Dreifach modulare Redundanz (TMR) (nach [Hed84])	17
3.1.	Zweidimensionales regelmäßiges Gitter mit angeschlossenen IP-Cores (nach [Chao5])	20
3.2.	Flittypen des Ausgangssystems	20
3.3.	Architektur des Switches (ohne Erweiterungen) (nach [Hos07])	21
3.4.	Xy-Routing ohne Fehlertoleranz oder das Löschen von Paketen (nach [Dal87a])	22
3.5.	Anzahl der eintreffenden Pakete bei einem defektfreien NoC	23
3.6.	Architektur der Netzwerkschnittstelle (ohne Erweiterungen)	23
4.1.	Algorithmus zum Bestimmen der globalen Erreichbarkeit (numerische Simulation)	26
4.2.	Numerische Simulation eines 3×3 -Gitters mit redundanten Anbindungen bei dem gesamte Komponenten ausfallen	28
4.3.	Numerische Simulation eines 20×20 -Gitters mit redundanten Anbindungen bei dem gesamte Komponenten ausfallen	28
4.4.	Numerische Simulation eines 3×3 -Gitters mit redundanten Anbindungen bei dem bidirektionale Switch-Switch-Verbindungen ausfallen	29

4.5.	Numerische Simulation eines 3x3-Gitters mit redundanten Anbindungen bei dem unidirektionale Switch-Switch-Verbindungen ausfallen	29
4.6.	Numerische Simulation eines 20x20-Gitters mit redundanten Anbindungen bei dem bidirektionale Switch-Switch-Verbindungen ausfallen	30
4.7.	Numerische Simulation eines 20x20-Gitters mit redundanten Anbindungen bei dem unidirektionale Switch-Switch-Verbindungen ausfallen	30
4.8.	Numerische Simulation eines 3x3-Gitters mit redundanten Anbindungen bei dem alle bidirektionalen Verbindungen ausfallen	32
4.9.	Numerische Simulation eines 3x3-Gitters mit redundanten Anbindungen bei dem alle unidirektionalen Verbindungen ausfallen	32
4.10.	Numerische Simulation eines 20x20-Gitters mit redundanten Anbindungen bei dem alle bidirektionalen Verbindungen ausfallen	33
4.11.	Numerische Simulation eines 20x20-Gitters mit redundanten Anbindungen bei dem alle unidirektionalen Verbindungen ausfallen	33
4.12.	Numerische Transientensimulation eines 3x3-Gitters mit einer Switch-IP-Core-Verbindung	35
4.13.	Numerische Transientensimulation eines 3x3-Gitters mit vier Switch-IP-Core-Verbindungen	36
4.14.	Numerische Transientensimulation eines 20x20-Gitters mit einer Switch-IP-Core-Verbindung	36
4.15.	Numerische Transientensimulation eines 20x20-Gitters mit vier Switch-IP-Core-Verbindungen	37
5.1.	Erreichbarkeitswahrscheinlichkeit wenn gesamte Komponenten ausfallen bei einem 3x3-Gitter	41
5.2.	Erreichbarkeitswahrscheinlichkeit wenn gesamte Komponenten ausfallen bei einem 20x20-Gitter	41
5.3.	Erreichbarkeitswahrscheinlichkeit wenn unidirektionale Verbindungen ausfallen bei einem 3x3-Gitter	43
5.4.	Erreichbarkeitswahrscheinlichkeit wenn unidirektionale Verbindungen ausfallen bei einem 20x20-Gitter	43
6.1.	CRC-basierte Fehlererkennung (Fehlerdetektion, Ende zu Ende)	46
6.2.	Abfragebasierte Fehlererkennung (Fehlerlokalisierung, Komponente zu Komponente)	47
6.3.	Abfragebasierte Überprüfung der Verbindungen	48
6.4.	Weiterleitung eines Überprüfungspaketes	49
6.5.	Flittypen mit der Erweiterung zur Erkennung von fehlerhaften Verbindungen	51
6.6.	Architektur des erweiterten Switches um fehlerhafte Ports erkennen zu können	52
6.7.	Architektur der erweiterten Netzwerkschnittstelle, um fehlerhafte Ports erkennen zu können	53
6.8.	Deadlockfreier fehlertoleranter Routingalgorithmus	54
6.9.	Livelock beim xy-Routing	55
6.10.	Verbindungsgraph eines 3x2-Gitters mit xy-Routingalgorithmus	55

6.11.	Teilkanalabhängigkeitsgraph des gelben/durchgezogenen Kanals des 3x2-Gitters mit xy-Routingalgorithmus	56
6.12.	Verbindungsgraph zum Überprüfen der Deadlockfreiheit des Routingalgorithmus mit einer Alternative	56
6.13.	Kanalabhängigkeitsgraph (blau/gestrichelt) zum Überprüfen der Deadlockfreiheit des Routingalgorithmus mit einer Alternative	57
6.14.	Kanalabhängigkeitsgraph (grün/durchgezogen) zum Überprüfen der Deadlockfreiheit des Routingalgorithmus mit einer Alternative	58
7.1.	Zweidimensionales Gitter ohne redundante Anbindung von IP-Cores	59
7.2.	Zweidimensionales Gitter inkl. der redundanten Anbindung von IP-Cores (vier verbindbare Switche an einen IP-Core)	60
7.3.	Flittypen mit der Erweiterung der redundanten Anbindung	61
7.4.	Architektur des erweiterten Switches inkl. der redundanten Anbindung (vier verbindbare NI)	62
7.5.	Routingalgorithmus für die Erreichbarkeit von den IP-Cores bei einer redundanten Anbindung	63
7.6.	Weiterleitung eines Paketes, wenn ein IP-Core nicht erreichbar ist	64
7.7.	Architektur der erweiterten Netzwerkschnittstelle inkl. der redundanten Anbindung (vier verbindbare Switche)	64
9.1.	Algorithmus des DTMR eines Switches	71
10.1.	Algorithmus zum Bestimmen der globalen Erreichbarkeit (Gatterebenessimulation)	74
10.2.	Gatterebenessimulation eines 3x3-Gitters mit redundanter Anbindung bei dem alle Verbindungen ausfallen	75
10.3.	Gatterebenessimulation eines 3x3-Gitters mit redundanter Anbindung und Weiterleitung zu anderen IP-Cores bei dem alle Verbindungen ausfallen	76
10.4.	Mit und ohne Weiterleitung zu anderen IP-Cores bei einer einfachen Switch-IP-Core-Anbindung	77
10.5.	Mit und ohne Weiterleitung zu anderen IP-Cores bei einer vierfachen Switch-IP-Core-Anbindung	77
10.6.	Gatterebenessimulation eines 3x3-Gitters mit einer einfachen Switch-IP-Core-Anbindung und transienten Fehlern	78
10.7.	Gatterebenessimulation eines 3x3-Gitters mit einer zweifachen Switch-IP-Core-Anbindung und transienten Fehlern	79
10.8.	Gatterebenessimulation eines 3x3-Gitters mit einer dreifachen Switch-IP-Core-Anbindung und transienten Fehlern	79
10.9.	Gatterebenessimulation eines 3x3-Gitters mit einer vierfachen Switch-IP-Core-Anbindung und transienten Fehlern	80
10.10.	Gatterebenessimulation eines 3x3-Gitters mit einer dreifachen Switch-IP-Core-Anbindung und der dynamischen dreifach modularen Redundanz	80
10.11.	Gatterebenessimulation eines 3x3-Gitters mit einer vierfachen Switch-IP-Core-Anbindung und der dynamischen dreifach modularen Redundanz	81

A.1.	Numerische Simulation eines 5x5-Gitters mit redundanter Anbindung bei dem gesamte Komponenten ausfallen	85
A.2.	Numerische Simulation eines 5x5-Gitters mit redundanter Anbindung bei dem bidirektionale Switch-Switch-Verbindungen ausfallen	86
A.3.	Numerische Simulation eines 5x5-Gitters mit redundanter Anbindung bei dem unidirektionale Switch-Switch-Verbindungen ausfallen	86
A.4.	Numerische Simulation eines 5x5-Gitters mit redundanter Anbindung bei dem alle Verbindungen (bidirektional) ausfallen	87
A.5.	Numerische Simulation eines 5x5-Gitters mit redundanter Anbindung bei dem alle Verbindungen (unidirektional)	88
A.6.	Numerische Transientensimulation eines 3x3-Gitters mit zwei Switch-IP-Core-Verbindungen	89
A.7.	Numerische Transientensimulation eines 3x3-Gitters mit drei Switch-IP-Core-Verbindungen	89
A.8.	Numerische Transientensimulation eines 5x5-Gitters mit einer Switch-IP-Core-Verbindung	90
A.9.	Numerische Transientensimulation eines 5x5-Gitters mit zwei Switch-IP-Core-Verbindungen	90
A.10.	Numerische Transientensimulation eines 5x5-Gitters mit drei Switch-IP-Core-Verbindungen	91
A.11.	Numerische Transientensimulation eines 5x5-Gitters mit vier Switch-IP-Core-Verbindungen	91
A.12.	Numerische Transientensimulation eines 20x20-Gitters mit zwei Switch-IP-Core-Verbindungen	92
A.13.	Numerische Transientensimulation eines 20x20-Gitters mit drei Switch-IP-Core-Verbindungen	92
B.1.	Xy-Routing mit einer Alternative und der Erweiterung des yx-Routings	94
B.2.	Yx-Routing mit einer Alternative	95
C.1.	Zweidimensionales Gitter inkl. der redundanten Anbindung von IP-Cores (zwei verbindbare Switche an einen IP-Core)	97
C.2.	Zweidimensionales Gitter inkl. der redundanten Anbindung von IP-Cores (drei verbindbare Switche an einen IP-Core)	98
C.3.	Architektur des erweiterten Switches inkl. der redundanten Anbindung (zwei verbindbare IP-Cores)	98
C.4.	Architektur des erweiterten Switches inkl. der redundanten Anbindung (drei verbindbare IP-Cores)	99
C.5.	Architektur der erweiterten Netzwerkschnittstelle inkl. der redundanten Anbindung (zwei verbindbare Switche)	99
C.6.	Architektur der erweiterten Netzwerkschnittstelle inkl. der redundanten Anbindung (drei verbindbare Switche)	100

Tabellenverzeichnis

3.1.	Flittypen (Ausgangssystem)	19
4.1.	Erreichbarkeit der IP-Cores nach Ausfall von allen Switch-Switch-Verbindungen bei einem 3x3-Gitter (mit dem Algorithmus aus Abbildung 4.1)	31
5.1.	MTTF _{Sim} (in Jahren) beim Ausfall aller Komponenten	40
5.2.	MTTF _{Sim} (in Jahren) beim Ausfall aller Verbindungen (bidirektional)	42
5.3.	MTTF _{Sim} (in Jahren) beim Ausfall aller Verbindungen (unidirektional)	42
5.4.	MTTF _{Sim} (in Jahren) beim Ausfall aller Verbindungen (unidirektional, 3x3-Gitter) inklusive transienter Fehler	44
5.5.	MTTF _{Sim} (in Jahren) beim Ausfall aller Verbindungen (unidirektional, 20x20-Gitter) inklusive transienter Fehler	44
6.1.	Flittypen (mit Fehlererkennung)	50
6.2.	Ausgangsports für die Paketweiterleitung	53
7.1.	Alternativer IP-Coreport	61
9.1.	Flittypen (mit dynamischer dreifach modularer Redundanz)	72
10.1.	Vergleich der MTTF _{Sim} (in Jahren) beim Ausfall aller Verbindungen (unidirektional, 3x3-Gitter)	76

Literaturverzeichnis

- [ope] OpenCores. Internet, URL <http://opencores.org/>
- [sim10] Mean Time Between Failures (MTBF). Internet (2010), URL http://support.automation.siemens.com/WW/llisapi.dll/csfetch/16818490/mtbf_de.pdf. Hintergrundinformation zur MTBF
- [nvi13] Vergleich von NVIDIA-GeForce-Grafikkarten. Internet (2013), URL http://www.nvidia.de/object/graphics_cards_buy_now_de.html
- [Cha05] Chan, J.: Networks on Chip : a very quick introduction! Internet (2005), URL <http://www.cse.unsw.edu.au/~cs4211/seminars/jeremy.ppt>
- [Cot12] Cota, E.: Reliability, availability and serviceability of networks-on-chip. Springer, New York (2012)
- [Dal87a] Dally, W.J.: Wire-efficient VLSI multiprocessor communication networks. P. Losleben (Hrsg.) Proceedings / Conference on Advanced Research in VLSI, S. 391–415. MIT Pr., Cambridge, Mass. (1987)
- [Dal87b] Dally, W.J.: Deadlock-Free Message Routing in Multiprocessor Interconnection Networks. IEEE Transactions on Computers 36(5), 547–553 (1987)
- [Hed84] Hedtke, R.: Mikroprozessorsysteme: Zuverlässigkeit, Testverfahren, Fehlertoleranz. Springer, Berlin (1984)
- [Hos07] Hosseinabady, M.: Using the Inter- and Intra-Switch Regularity in NoC Switch Testing. Design, Automation & Test in Europe Conference & Exhibition, 2007. DATE '07, S. 361–366 (2007)
- [Hwa00] Hwang, K.: Network-Based Cluster Computing. Internet (2000), URL <http://www-classes.usc.edu/engr/ee-s/657h/clusterbasic.pdf>
- [Kir12] Kirchstädter, A.: Kommunikationsnetze 1 - Netzarchitekturen und Protokolle. Universität Stuttgart, Institut für Kommunikationsnetze und Rechnersysteme (2012)
- [Liu11] Liu, J.: Reliability of microtechnology: interconnects, devices and systems. Springer, New York, NY (2011)

- [O'C90] O'Connor, P.D.T.: Zuverlässigkeitstechnik: Grundlagen und Anwendungen. VCH, Weinheim (1990)
- [Scho5] Schiffmann, W.: Technische Informatik 2 - Grundlagen der Computertechnik, 5., neu bearb. u. erg. Aufl. Ausgabe. Springer, Berlin (2005)
- [Wun10] Wunderlich, H.J.: Grundlagen der Rechnerarchitektur. Universität Stuttgart, Institut für Technische Informatik (2010)

Alle Links wurden zuletzt am 22. Juli 2013 besucht.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

(Ort, Datum, Unterschrift)

