Exzellenzcluster
Cognitive Interaction Technology
Kognitronik und Sensorik
Prof. Dr.-Ing. U. Rückert

FPGA-Cluster – Anwendungsgebiete und Kommunikationsstrukturen

zur Erlangung des akademischen Grades eines

DOKTOR-INGENIEUR (Dr.-Ing.)

der Technischen Fakultät
der Universität Bielefeld

genehmigte Dissertation

von

Dipl.-Ing. Johannes Romoth

Referent: Prof. Dr.-Ing. Ulrich Rückert
Korreferent: Prof. Dr. rer. nat. Christian Schröder

Tag der mündlichen Prüfung: 12.10.2017

Bielefeld / März 2018
DISS KS / 14

Danksagung

Im Lauf der Entstehung der hier vorliegenden Arbeit hatten viele Menschen einen kleineren oder größeren Anteil daran, dass es mir möglich war, die Arbeit abzuschließen. Bei all jenen möchte ich mich an dieser Stelle bedanken.

Mein besonderer Dank gilt Herrn Prof. Dr.-Ing. Ulrich Rückert dafür, dass ich ein Teil seiner Fachgruppe werden konnte und so die Möglichkeit zur Promotion erhielt.

Ebenfalls besonders danken möchte ich Dr.-Ing. Mario Porrmann für seine fachlichen Beiträge und Anmerkungen, ohne die die Arbeit nie diesen Stand erreicht hätte.

Jens Hagemeyer danke ich für die tolle Zusammenarbeit bei der Entstehung des RAPTOR-XPress. Sein tiefes fachliches Wissen und die Bereitschaft, jederzeit eine weitere Überstunde zu machen, haben das Projekt überhaupt erst ermöglicht.

Arne Hilgenstein, Dirk Jungewelter und Holger Urban möchte ich für ihre Einsatzbereitschaft und die gute Zusammenarbeit im Rahmen der Entstehung des RAPTOR-XPress-Clusters danken. Für ihre Korrekturen meiner englischsprachigen Veröffentlichungen gilt mein Dank Sandra Brünger. Kai Brackmann hat mit seinen Hinweisen und Anmerkungen zur Darstellung der graphentheoretischen Teile der Arbeit ebenfalls meinen Dank verdient. Für die zur Verfügung gestellten Rechenressourcen möchte ich Michael Klockenhoff und Magnus Asplund danken.

Selbstverständlich gilt mein Dank auch meinem Vater für seine Unterstützung und den Rückhalt, den er mir all die Jahre gegeben hat. Mein persönlichstes DANKE widme ich meiner Frau Katrin. Nur dank ihrer Motivation, ihres offenen Ohres, ihrer Geduld und nicht zuletzt ihrer Bereitschaft zum Korrekturlesen habe ich das Ziel nie aus den Augen verloren.

Zusammenfassung

Fortschritte in der Fertigungstechnik von Halbleitern in Silizium ermöglichen hohe Integrationsdichten und somit den Entwurf von leistungsstarken digitalen logikverarbeitenden Elementen. Mit Hilfe hochparalleler anpassbarer flexibler Architekturen wie im Feld programmierbare Logik-Gatter-Anordnungen (engl.: Field Programmable Gate Array, FPGA) kann eine Vielzahl an Problemstellungen gelöst werden. Aufgrund der gebotenen Parallelität ist es selbst bei den verhältnismäßig geringen Taktraten des FPGAs, die den hochspezialisierten dedizierten Schaltungen anderer Systeme gegenüberstehen, möglich, harte Echtzeitschranken bei der Lösungsberechnung einzuhalten. Darüber hinaus ist die Energieeffizienz aufgrund des proportionalen Einflusses der Taktrate auf die dynamische Verlustleistung von Schaltungen wesentlich höher. Dennoch erfordern unterschiedliche Anwendungsszenarien von FPGAs eine derart hohe Anzahl an Logikressourcen, dass nur die Bündelung mehrerer FPGAs zu einem vernetzten Cluster eine effiziente Verarbeitung gewährleistet.

Im Verlauf dieser Arbeit werden die Anforderungen an eine FPGA-Cluster-Lösung herausgestellt. Anhand eines Überblicks über die typischen Anwendungsfelder rekonfigurierbarer Logiksysteme können die grundlegenden Voraussetzungen identifiziert werden, die eine universell einsetzbare FPGA-Cluster-Architektur erfüllen muss. Insbesondere an die Kommunikationsinfrastruktur zwischen den einzelnen FPGAs im Cluster werden hohe Anforderungen in Bezug auf die Flexibilität gestellt. Die Anpassbarkeit an die individuellen Anforderungen der eingesetzten Algorithmen ist somit neben der Datenrate und der Latenz ein Kernelement bei der Entwicklung des FPGA-Clusters. Zur Evaluation von Systementwürfen wird eine Modellierung erarbeitet, die einen Vergleich auf Basis der Kommunikationsstrukturen ermöglicht. Eine darüber hinausgehende Optimierung des die Verbindungen im Cluster beschreibenden Graphen führt zu einer Minimierung der Latenz von Datenübertragungen und somit zu einer Leistungssteigerung des Gesamtsystems.

Die identifizierten Anforderungen an ein flexibles, modulares und skalierbares FPGA-Cluster-System werden im Rahmen der Arbeit umgesetzt, so dass der RAPTOR-XPress-FPGA-Cluster entsteht, der zudem zur Steigerung der Ressourceneffizienz auf den Mehranwenderbetrieb ausgelegt ist. Auf diese Weise lassen sich in einer Anwendung ungenutzte FPGAs parallel für andere Aufgaben verwenden. Im Zusammenspiel mehrerer Arbeiten des Fachgebiets *Kognitronik und Sensorik* der Universität Bielefeld ist ein Beispielaufbau mit 16 RAPTOR-XPress-Trägersystemen und 64 FPGAs mit insgesamt 44 359 680 Logikzellen-Äquivalenten und 256 GB an lokalem Arbeitsspeicher realisiert worden. Durch die Umsetzung topologieoptimierter Verbindungsstrukturen kann eine gegenüber vergleichbaren Systemen um 28 % gesteigerte Logikdichte erreicht werden, die zusammen mit der erzielbaren Datenrate von $16 \times 11{,}5\,\mathrm{Gbit/s}$ die Leistungsfähigkeit der Kommunikationsinfrastruktur des FPGA-Clusters verdeutlicht.

Inhaltsverzeichnis

Symbolverzeichnis

A^M	Adjazenzmatrix
B	maximale Datenübertragungsrate eines Kommunikationskanals
C	elektrische Kapazität
G	elektrischer Leitwert
$c(e)$	Kosten- oder Gewichtsfunktion einer Kante
$c(p)$	Kosten- oder Gewichtsfunktion eines Pfades
D^M	Distanzmatrix
$D(G)$	Durchmesser des Graphen G
E	Menge der Kanten
e	eine Kante $e \in E$
$\epsilon(v)$	Exzentrizität des Knotens v
ϵ_0	Permittivität des Vakuums
ϵ_r	relative Permittivität eines Leiters
G	Graphenbeschreibung mit $G = (V, E)$
Γ	Menge isomorpher k-regulärer Graphen für ein gegebenes V
γ	ein Graph $\gamma \in \Gamma$
k	regelmäßige Valenz aller Knoten $e \in E$
L	elektrische Induktivität
μ_0	Vakuumpermeabilität
μ_r	relative Permeabilität
N	Ausführungszeit
$\mathbb{N}$	Menge der natürlichen Zahlen

Ω	Menge der Graphen mit minimalem ψ, $\Omega \subseteq \Theta$
ω	ein Graph $\omega \in \Omega$
p	ein Pfad zwischen zwei Knoten eines Graphen als unterbrechungsfreie Verbindung von Kanten
Φ	Menge der Ports
ϕ	ein Port $\phi \in \Phi$
$\psi(G)$	Exzentrizitätssumme aller Knoten des Graphen G
R	elektrischer Widerstand
r	Reflexionsfaktor
ρ	Logikdichte
M	Menge der Module
S	Menge der Schnittstellen
m	ein Modul $m \in M$
s	eine Schnittstelle $s \in S$
T	Latenz einer Verbindung
Θ	$\Theta \subseteq \Gamma$ Menge der Graphen mit minimalem Durchmesser aus Γ
ϑ	ein Graph $\vartheta \in \Theta$
T_R	Rekonfigurationszeit
V	Menge der Knoten
$d(v)$	Valenz oder Grad des Knotens v
W	Wiederverwendbarkeit einer Logikbeschreibung
Ξ	Menge der Verbindungsvarianten
y	Admittanz-Leitungselement
Z	Impedanz
z	Impedanz-Leitungselement
$\zeta(G)$	Exzentrizitätsindex des Graphen G

Glossar und Abkürzungsverzeichnis

ADC engl.: **A**nalog **D**igital **C**onverter – Überführt ein analoges Eingangssignal in ein digitales (zeit- und wertediskretes) Resultat; entscheidende Größen sind die Abtastrate und die Bitbreite des Ausgangs

AES engl.: **A**dvanced **E**ncryption **S**tandard – Symmetrischer, weit verbreiteter Verschlüsselungsstandard

ANN engl.: **A**rtificial **N**eural **N**etwork – Künstliche Neuronale Netze bilden strukturell und funktionell Teile des Gehirns nach

ASIC engl.: **A**pplication-**S**pecific **I**ntegrated **C**ircuit – Integrierter Schaltkreis, der eine nicht mehr veränderbare Schaltung implementiert

ASIF engl.: **A**pplication **S**pecific **I**nflexible **F**PGA – Hardware-Lösung, die durch die Verwendung von grobgranularer Strukturen teilweise auf die Teile der Flexibilität von FPGAs verzichtet zu Gunsten höherer Verarbeitungsgeschwindigkeiten in dedizierten Anwendungen

BCB engl.: **B**road**c**ast **B**us – Geswitchtes Bussystem des RAPTOR-XPress, dass alle Module gleichberechtigt miteinander verbindet

BEE engl.: **B**erkeley **E**mulation **E**ngine – FPGA-Cluster-Systeme mit dem Entwicklungsursprung an der University of California, Berkeley

BLAST engl.: **B**asic **L**ocal **A**lignment **S**earch **T**ool – Heuristische Algorithmen zur Sequenzanalyse in der Bioinformatik

BRAM engl.: **B**lock **RAM** – In Xilinx FPGAs integrierter SRAM-Speicher

CART engl.: **C**lassification **A**nd **R**egression **T**ree – Auf Entscheidungsbäumen aufbauendes Verfahren zur Extraktion bestimmter Merkmale aus großen Datenmengen

CBIR engl.: **C**ontent **B**ased **I**mage **R**etrieval – Bildersuche in einer Datenbank auf Ähnlichkeitsbasis eines oder mehrerer bestimmter Merkmale des Ausgangsbildes

CCM engl.: **C**ustom **C**omputing **M**achines – Auf die Beschleunigung einer speziellen Anwendung hin optimierte Architektur

CLB engl.: **C**onfigurable **L**ogic **B**lock – Grundeinheit eines Xilinx FPGA, welche sich je nach FPGA-Generation aus unterschiedlich vielen und verschieden großen Funktionsblöcken wie z.B. LUTs und MUXen zusammensetzt

COTS engl.: **C**ommodity (auch Commercial oder Components) **O**ff-**T**he-**S**helf– Regulär im Handel erhältliche Software- oder Hardware-Produkte

CPU engl.: **C**entral **P**rocessing **U**nit – Algorithmen in Form von Software verarbeitende zentrale Recheneinheit eines Computers

DAC engl.: **D**igital **A**nalog **C**onverter – Überführt ein digitales Eingangssignal in eine analoge Spannung; entscheidende Größen sind die Abtastrate und die Bitbreite des Ausgangs

DB engl.: **D**aughter **B**oard – Aufsteckmodul für das RAPTOR Rapid-Protoyping-System, welches verschiedene Funktionalitäten wie FPGAs, unterschiedliche IO-Standards oder ASIC-Testumgebungen implementiert

DDC engl.: **D**igital **D**own **C**onverter – Transformiert ein empfangenes Signal mit Zwischenfrequenz in ein digitales komplexes Basisbandsignal

DMA engl.: **D**irect **M**emory **A**ccess – Mit DMA-Transfers ist es möglich, große Datenmengen ohne Belastung der CPU zu übertragen; lediglich die Informationen zu den zu übermittelnden Daten müssen von der CPU an den eigenständigen DMA-Controller übergeben werden

DSP engl.: **D**igital **S**ignal **P**rocessor – Digitaler Signalprozessor

DTC engl.: **D**ecision **T**ree **C**lassification – Ein Verfahren, um aus großen Datenmengen bestimmte Merkmale zu extrahieren

DUC engl.: **D**igital **U**p **C**onverter – Wandelt ein digitales Basisbandsignal mit niedriger Abtastrate in ein Bandpasssignal mit hoher Abtastrate um

ETA engl.: **E**ight-meter-wavelength **T**ransient **A**rray – Ein Radioteleskopaufbau der Virginia Polytechnic Institute and State University

FEM **F**inite-**E**lemente-**M**ethode – Simulationsmethode, die ein Problem in viele kleine Teilprobleme zerlegt, welche durch einfache Verhaltensbeschreibungen erfasst werden können

FEXT engl.: **F**ar **E**nd Cross**t**alk; Elektrische Einkopplung eines signalführenden Leiters auf eine benachbarte Leitung, welche auf Seite der entfernten Signalsenke beobachtet wird

FFT engl.: **F**ast **F**ourier **T**ransform – Algorithmus zur schnellen Berechnung der Werte einer diskreten Fouriertransformation

FHPCA engl.: **F**PGA **H**igh **P**erformance **C**omputing **A**lliance – Am Edinburgh Parallel Computing Centre gegründetes Konsortium aus Wissenschaft und Industrie zur Evaluation der Möglichkeiten von FPGAs als HPC-Plattform

FIFO engl.: **F**irst **I**n **F**irst **O**ut – Beschreibt eine unidirektionale Datenstruktur, in der zuerst eingetragene Daten auch als Erstes wieder gelesen werden müssen

FIR engl.: **F**inite **I**mpulse **R**esponse – Ein meist digitales Filter, welches unabhängig von den gewählten Regelparametern stabil bleibt

FPGA engl.: **F**ield **P**rogrammable **G**ate **A**rray – Programmierbarer Logikbaustein, in dem beliebige digitale Schaltungen abgebildet werden können

FSB engl.: **F**ront **S**ide **B**us – Bussystem, welches einen Prozessor oder mehrere Prozessoren mit der Northbridge verbindet, die wiederum weitere Komponenten wie den Arbeitsspeicher anbindet

FSC engl.: **F**PGA **S**ession **C**ontrol – Client-Server-Architektur zur Verwaltung des Spirit-FPGA-Clusters

GALS engl.: **G**lobally **A**synchronous **L**ocally **S**ynchronous – Architekturansatz, bei dem lokal Logikcluster synchron betrieben werden; das Gesamtsystem besteht aus einer Vielzahl solcher synchronen Einheiten, welche nicht notwendigerweise synchron zueinander betrieben werden; durch die lokal begrenzte Taktverteilung können höhere Verarbeitungsgeschwindigkeiten erreicht werden

GPGPU engl.: **G**eneral **P**urpose **G**raphic **P**rocessing **U**nits – Grafikprozessoren, die sich aufgrund ihrer Architektur und Softwareunterstützung auch für allgemeine Berechnungen verwenden lassen

GPP engl.: **G**eneral **P**urpose **P**rocessors – Mehrzweckprozessoren, die anders als ASICs oder auf bestimmte Aufgabenfelder hin optimierte Prozessoren für eine Vielzahl an Anwendungen geeignet sind

GTH MGT in Xilinx Virtex-6- und Virtex-7-FPGAs, welcher Übertragungsraten von bis zu 11,18 Gbit/s im Fall des Virtex-6 und 13,1 Gbit/s beim Virtex-7 ermöglicht

GTP MGT in Xilinx Virtex-5- und Artix-7-FPGAs, welcher Übertragungsraten von bis zu 3,75 Gbit/s im Fall des Virtex-5 und 6,6 Gbit/s beim Artix-7 ermöglicht

GTX MGT in Xilinx Virtex-5-, Virtex-6-, Kintex-7- und Virtex-7-FPGAs, welcher Übertragungsraten von bis zu 6,5 Gbit/s im Fall des Virtex-5, 6,6 Gbit/s beim Virtex-6 und 12,5 Gbit/s für den Kintex-7 und Virtex-7 ermöglicht

HIL engl.: **H**ardware **I**n The **L**oop – Umgebungssimulatoren, bei denen das zu testende Gerät über seine IO-Leitungen mit den Stimuli des Simulators versorgt wird

HMM engl.: **H**idden **M**arkov **M**odel – Stochastische Systemmodellierung, bei der die internen Zustände nicht von außen beobachtet werden können

HP engl.: **H**igh **P**erformance – IO-Treiber-Typ von Xilinx 7-Series-FPGAs, welcher die IO-Standard-Kompatibilität zu Gunsten höherer Übertragungsgeschwindigkeiten eintauscht

HPC engl.: **H**igh **P**erformance **C**omputing – Aufwendige Berechnungen, für die Standard-Systeme wie PCs nicht ausreichen und die deshalb auf Spezialhardware durchgeführt werden müssen

HPRC engl.: **H**igh **P**erformance **R**econfigurable **C**omputing – Im Gegensatz zum prozessorbasierten HPC werden rekonfigurierbare Architekturkomponenten wie FPGAs ergänzend oder ausschließlich für die Berechnungen eingesetzt

HR engl.: **H**igh **R**ange – IO -Treiber-Typ von Xilinx 7-Series-FPGAs mit IO-Standard-Kompatibilität zu älteren Bausteinen im Rahmen von 1,2 V bis 3,3 V

I2C engl.: **I**nter-**I**ntegrated **C**ircuit – Auch als I^2C bekanntes Bussystem, welches mit lediglich zwei Übertragungsleitungen realisiert wird; Anwendung findet der von Philips entwickelte Bus bei der Einbindung von Komponenten mit relativ niedrigem Datenaufkommen

IO engl.: **I**nput/**O**utput – Als Ein- und Ausgang beschaltbarer Anschluss innerhalb einer oder zwischen verschiedenen elektrischen Baugruppen

IP-Core engl.: **I**ntellectual **P**roperty **Core** – Integrationsfertiger Hardware- oder HDL-Funktionsblock, welcher zumeist auf Lizenz-Basis vertrieben wird

JTAG engl.: **J**oint **T**est **A**ction **G**roup – Umsetzung des IEEE 1149.1 Standards, welcher ursprünglich zum Auslesen und Debuggen von Registern in ICs eingesetzt wurde, inzwischen aber auch für die Konfiguration von FPGAs und anderen programmierbaren Logikbausteinen verwendet wird

KI **K**ünstliche **I**ntelligenz – Programmabläufe, die menschenähnliches Entscheidungsverhalten simulieren

LUT engl.: **L**ook **U**p **T**able – Speicherblock, der als Tabelle dient, in der häufig verwendete Ergebnisse abgefragt werden können, ohne jeweils neu berechnet werden zu müssen

LWL **L**icht**w**ellen**l**eiter – Dielektrische optische Faserleiter, welche zur Übermittlung von Informationen per Lichtsignal benötigt werden

MAC engl.: **M**ultiply **Ac**cumulated – Typischerweise zweischrittige Berechnung auf DSPs, bei der zunächst multipliziert und anschließend addiert wird

MFS engl.: **M**ultiple **F**PGA **S**ystem – Aus mehreren FPGAs bestehendes System unabhängig davon, welche Kommunikationsinfrastruktur zwischen den einzelnen FPGAs vorliegt

MGT engl.: **M**ulti-**G**igabit **T**ransceiver – Serielle Hochgeschwindigkeitsverbinder

MPI engl.: **M**essage **P**assing **I**nterface – Ein Standard des Argonne National Labratory, welcher Kommunikationsabläufe zwischen zwei parallel laufenden Prozessen festlegt; die Anzahl der tatsächlich parallel laufenden Prozesse ist nicht beschränkt; bei der Umsetzung wird kein explizites Protokoll bzw. keine Technologie auf physikalischer Ebene vorgeschrieben

MPSoC engl.: **M**ulti **P**rocessor **S**ystem **o**n **C**hip – Erweitert das Konzept von SoCs auf Prozessoren mit mehreren Ausführungseinheiten, die zusammen mit anderen Komponenten auf einem Die gefertigt werden

MUX engl.: **Mu**ltiple**x**er – Ermöglicht das Schalten verschiedener Eingangssignale auf unterschiedlich viele Ausgänge

NAND engl.: **N**ot-**And** – Boolsche negierte *Und*-Funktion

NEXT engl.: **N**ear **E**nd Cross**t**alk; Elektrische Einkopplung eines signalführende Leiters auf eine benachbarte Leitung, welche auf Seite der Signaleinspeisung beobachtet wird

NIDS engl.: **N**etwork **I**ntrusion **D**etection **S**ystem – Mechanismen in Netzwerken, die den Datenverkehr auf das Verhalten abseits normaler Abläufe untersuchen

NIPS engl.: **N**etwork **I**ntrusion **P**revention **S**ystem – Mechanismen, die das unberechtigte Eindringen in fremde Netzwerke unterbinden

NNUS engl.: **N**on Uniform **N**odes **U**niform **S**ystems – Cluster-System mit verschiedenartigen Rechenelementen auf Knotenebene

NoC engl.: **N**etwork **o**n **C**hip – Netzwerktopologie zur Kopplung verschiedener Strukturen auf einem Die

NOR engl.: **N**ot-**Or** – Boolsche negierte *Oder*-Funktion

NXOR engl.: **N**ot-E**x**clusive-**Or** – Boolsche negierte Exklusiv-*Oder*-Funktion

OS engl.: **O**perating **S**ystem – Betriebssystem

PCB engl.: **P**rinted **C**ircuit **B**oard – Ein- und mehrlagige Leiterkarten als Träger- und Verbindungssystem für elektronische Bauteile

PCI engl.: **P**eripheral **C**omponent **I**nterconnect – Bit-paralleles Bus-System zur Anbindung von Peripheriekomponenten wie Einsteckkarten an den Prozessor eines Computersystems

PCIe engl.: **P**eripheral **C**omponent **I**nterconnect **E**xpress – Serielle Punkt-zu-Punkt-Verbindung; Nachfolger von PCI mit wesentlich höheren Datenraten

PE **P**rozessor**e**inheit

PFC engl.: **P**ower **F**actor **C**orrection – Die Leistungsfaktorkorrektur ist eine Regelung, welche unerwünscht auftretende Oberschwingungen filtert, indem sie den Leistungsfaktor der Schaltung möglichst weit dem Betrag von 1 annähert

PHY engl.: **Phy**sical layer – Bauelement zur Umsetzung der ersten Schicht des ISO/OSI-Modells

PLD engl.: **P**rogrammable **L**ogic **D**evice – Programmierbarer Logikbaustein

PRBS engl.: **P**seudo**r**andom **B**inary **S**equence – Pseudozufällige Bit-Sequenz, welche mit Hilfe von Schieberegistern erzeugt werden kann

PVT engl.: **P**rocess **V**oltage **T**emperature – Einflüsse auf eine Übertragung, die zum einen durch herstellungsbedingte Variationen hervorgerufen werden, zum anderen aufgrund nicht idealer Versorgung oder wechselnder Temperaturen im laufenden Betrieb auftreten

QoS engl.: **Q**uality **o**f **S**ervice – Übertragungsqualität einer Verbindung; in Netzwerkapplikationen werden hiermit vorwiegend eine weiche Echtzeitübertragung bzw. die Mechanismen um diese zu ermöglichen beschrieben

RAMP engl.: **R**esearch **A**ccelerator for **M**ultiple **P**rocessors – Konsortium zur Entwicklung einer Multiprozessoremulationsumgebung auf FPGA-Basis

SAT engl.: Boolean **Sat**isfiability – Erfüllbarkeitsproblem der Aussagenlogik

SoC engl.: **S**ystem **o**n **C**hip – Vollständiges Informationsverarbeitungssystem auf einem einzelnen Die

SURF engl.: **S**peeded **U**p **R**obust **F**eatures – Algorithmus zur Erkennung von Bildmerkmalen

SVM engl.: **S**upport **V**ector **M**achine – Verfahren, um große Datenmengen zu klassifizieren und Muster zu erkennen

TDP engl.: **T**hermal **D**esign **P**ower – Herstellerangabe zur thermischen Verlustleistung eines elektronischen Bauteils, auf die das Kühlkonzept ausgelegt werden muss

UNNS engl.: **U**niform **N**odes **N**on Uniform **S**ystems – Cluster-System, das auf einem Knoten jeweils gleichartige Verarbeitungseinheiten, aber verschiedene Knotentypen im System verwendet

VIA engl.: **V**ertical **I**nterconnect **A**ccess; Eine elektrische Durchkontaktierung zwischen zwei oder mehr Lagen einer Leiterkarte

XOR engl.: E**x**clusive-**Or** – Boolsche Exklusiv-*Oder*-Funktion

1 Einleitung

Die fortschreitenden Entwicklungen in der Digitaltechnologie ermöglichen immer komplexere Systeme. Gleichzeitig sind die Anforderungen der modernen Gesellschaft an die Rechenkapazität selbst kleiner mobiler Geräte in den vergangenen Jahren kontinuierlich gestiegen. Während Mikroprozessoren als zentrale Verarbeitungseinheiten bereits in den 1960er Jahren Erwähnung finden und spätestens mit Intels 4004 in der Öffentlichkeit wahrgenommen werden [1], wird rekonfigurierbare Logik als datenverarbeitendes Element erst in den 1980er Jahren relevant, namentlich durch die Firmen Xilinx und Altera. Der konzeptionelle Unterschied zwischen Prozessorarchitekturen und Bausteinen mit rekonfigurierbarer Logik (engl.: Programmable Logic Device, PLD) besteht darin, dass die Flexibilität bei Prozessoren über den Einsatz von Software erreicht wird, während PLDs auf Hardware-Ebene logische Funktionen abbilden. Aufgebaut sind PLDs aus einer großen Matrixstruktur bestehend aus miteinander flexibel verschaltbaren Logikblöcken. Auf diese Weise ist es möglich, die Parallelität der Struktur Bit-genau auszunutzen. Während Prozessoren als Zustandsautomaten angesehen werden können, die sequentiell Befehle abarbeiten, erhalten PLDs in ihren verschiedenen Ausprägungen wie FPGAs, CPLDs etc. erst eine Funktion durch die Programmierung und Verschaltung ihrer Logikblöcke.

Neben dem gebotenen hohen Grad an Parallelität in der Verarbeitung von Eingangssignalen erweisen sich FPGAs in vielen Anwendungsfällen als energieeffizienter als Prozessoren oder die auf Vektorberechnungen spezialisierten allgemein nutzbaren Grafikprozessoren (engl.: General Purpose Graphic Processing Units, GPGPU) [2]. Insbesondere seit eine kontinuierliche Steigerung der Taktraten bei Prozessoren nicht mehr möglich ist, sind parallel arbeitende alternative Architekturen notwendig. Kombiniert mit der Möglichkeit der Bit-genauen Abbildung einer Problemstellung können je nach Anwendungsfeld wesentlich höhere Verarbeitungsgeschwindigkeiten sowie energieeffizientere Lösungen, bezogen auf die Anzahl der Verarbeitungsschritte pro Watt, realisiert werden [3]. Demgegenüber stehen anwendungsspezifische integrierte Schaltungen (engl.: Application Specific Integrated Circuit, ASIC), die in Bezug auf die Energieeffizienz und Verarbeitungsgeschwindigkeit die programmierbaren Architekturen übertreffen. Dies geht zu Lasten der Übertragbarkeit auf andere Problemstellungen als solche, für die die Schaltung ursprünglich ausgelegt wurde. Hinzu kommt, dass die Initialkosten für eine ASIC-Fertigung sehr hoch sind und diese somit nur bei hohen Stückzahlen ökonomisch sinnvoll ist. Eine Gemeinsamkeit von FPGAs und ASICs besteht darin,

dass für die Verarbeitung einer Aufgabenstellung eine feste Zeitschranke ermittelt werden kann. Auf diese Weise lassen sich harte Echtzeitanforderungen an die Berechnungsdauer umsetzen. Dem gegenüber benötigen komplexe Prozessorsysteme Softwarekonstrukte wie beispielsweise Betriebssysteme, die nicht notwendigerweise ein zeitlich deterministisches Verhalten aufweisen. Insgesamt betrachtet stellen FPGAs einen Kompromiss dar zwischen den beiden Extremen – zwischen den durch die Verwendung von Software hochflexiblen Prozessoren auf der einen Seite und den hocheffizienten, jedoch stark spezialisierten ASIC-Umsetzungen andererseits.

Trotz der hohen Anzahl an Logikzellen moderner FPGAs ist ab einer gewissen Größe der Problemstellung eine Umsetzung auf einem einzelnen FPGA nicht mehr möglich. Die Notwendigkeit, die Aufgaben auf mehrere untereinander verbundene FPGAs aufzuteilen, führt zum Einsatz von FPGA-Cluster-Strukturen, deren Betrachtung und Entwicklung das Zielanliegen der vorliegenden Arbeit ist.

Im folgenden Abschnitt wird ein inhaltlicher Überblick über die weiteren Kapitel der Arbeit gegeben. Die Motivation für die notwendige Optimierung der Topologie der Kommunikationsinfrastruktur eines FPGA-Clusters hin zu einer kompakten und leistungsfähigen Anordnung wird im darauf folgenden Abschnitt erläutert, einschließlich der sich daraus ergebenden Anforderungen an ein neuartiges FPGA-Cluster-System. Den Abschluss des Kapitels bilden Hinweise zur Struktur der Arbeit.

1.1 Überblick

In dieser Arbeit wird eine Auswahl häufig auf FPGAs und FPGA-Clustern implementierter Algorithmen verschiedener Anwendungsbereiche untersucht und ihre Abbildbarkeit sowie die sich daraus ergebenden Anforderungen an eine Kommunikationsinfrastruktur herausgestellt. Für die Unterstützung mehrerer unterschiedlicher Anwendungen, welche parallel auf verschiedenen FPGAs eines Clusters implementiert sind, wird eine Optimierung der Cluster-Topologie erarbeitet. Auf Basis dieser Erkenntnisse erfolgt der Entwurf und die Realisierung eines neuen FPGA-Clusters, welcher die gewonnenen Erkenntnisse umsetzt und auf diese Weise einen allgemein verwendbaren FPGA-Cluster darstellt. Die entstandene Systemarchitektur kann aufgrund ihrer flexibel verschaltbaren Kommunikationsinfrastruktur dahingehend optimiert werden, dass die Kommunikationswege innerhalb des entstandenen Clusters minimal sind. Zur Einordnung der Leistungsfähigkeit wird ein Vergleich mit aktuellen FPGA-Cluster-Lösungen angestellt.

1.2 Motivation

Auch nach Entschleunigung der durch Gordon Moore bereits 1965 beschriebenen Steigerung der Integrationsdichte von in Silizium implementierten integrierten

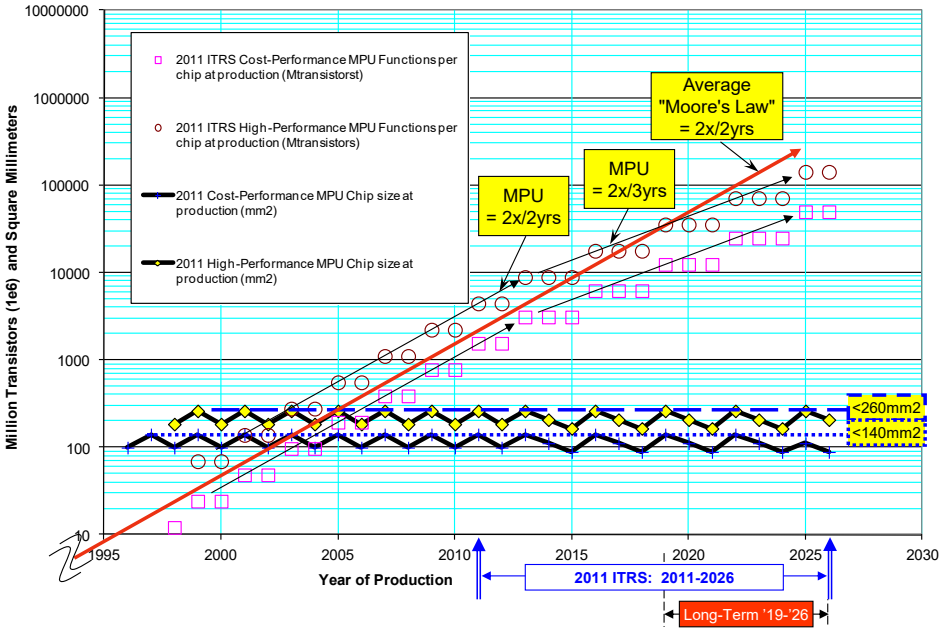


Abbildung 1.1: ITRS-Prognose zur Integrationsdichte von Transistoren [5]

Schaltungen steigt die Komplexität moderner Schaltungsentwürfe weiter an. Der ursprüngliche Logikzuwachs und die Steigerung der Taktfrequenz digitaler Schaltungen alle 18 bis 24 Monate [4] haben sich aufgrund physikalischer Grenzen in der Reduktion der Strukturgrößen verlangsamt. Jedoch können auch mit geringeren Steigerungsraten immer komplexere integrierte Systeme entworfen werden. Die Daten der *International Technology Roadmap for Semiconductors (ITRS)* zeigen, dass die Integrationsdichte in den nächsten zehn Jahren trotz geringerer Steigerungsraten weiterhin ansteigen wird (Abbildung 1.1).

Der Erhöhung der Logikdichte nach Moore folgend werden die Strukturen von FPGAs darüber hinaus um funktionale aufgabenspezifische Blöcke wie etwa digitale Signalprozessoreinheiten (engl.: Digital Signal Processor, DSP), RAM oder sogar vollständige Prozessorkerne erweitert [6], was die Anzahl der potentiellen Einsatzgebiete erhöht und gleichzeitig die benötigten ergänzenden Bausteine auf Leiterkarten- (engl.: Printed Circuit Board, PCB) und Systemebene verringert.

FPGAs bieten die Möglichkeit logische Funktionen abzubilden und können somit die funktionale Repräsentation eines künftigen ASIC übernehmen. Neben dem Einsatz im Rapid-Prototyping finden FPGAs Anwendung in Bereichen, in denen der Schritt zu einer ASIC-Umsetzung wirtschaftlich aufgrund zu geringer Stückzahlen

oder möglicher Änderungen im späteren Betrieb nicht sinnvoll erscheint, eine dedizierte Hardware-Lösung aber dennoch gefordert ist.

Abseits harter Echtzeitanforderungen als Motivation für die Umsetzung von Algorithmen in Hardware ist vor allem die beschleunigte Verarbeitung gegenüber Software-Lösungen eine Erklärung für die durchaus aufwendige Portierung. Aus diesem Grund treten FPGAs immer häufiger in direkte Konkurrenz zu bestehenden Hochleistungsrechensystemen (engl.: High Performance Computing, HPC) und Beschleunigern mit GPGPUs. Aufgrund ihrer Energieeffizienz und der Bit-genauen Anpassbarkeit an die jeweilige Problemstellung finden auch FPGA-Cluster Einzug in moderne Rechenzentren – sei es als Erweiterung bestehender Serversysteme oder sogar als eigenständige Lösungen.

Trotz der stark angestiegenen Anzahl an Logikressourcen, die sich neben der reinen Steigerung der Integrationsdichte von Transistoren in Silizium durch das Platzieren mehrerer untereinander verbundener Chips innerhalb eines Gehäuses ergibt [7], müssen für verschiedene Anwendungsfälle mehrere FPGAs zu einem Cluster verbunden werden. Anhand der Betrachtung der Einsatzgebiete von FPGAs und FPGA-Clustern erfolgt eine Anforderungsanalyse an die Kommunikationsinfrastruktur, die der Grundstein für eine Cluster-Bildung ist.

Zur Beurteilung aktueller Systeme und als Basis für die Entwicklung künftiger Cluster wird eine Metrik benötigt, die Aufschluss über die vom Cluster zur Verfügung gestellten Logikressourcen gibt. Je loser gekoppelt die einzelnen FPGAs im Cluster in Bezug auf die Bandbreite und Latenz der Kommunikationsinfrastruktur sind, desto geringer fällt die Logikdichte insgesamt aus. Relevant ist das Ergebnis nicht nur für große Systementwürfe, die auf dem entsprechenden Cluster realisiert werden sollen, sondern auch für eine Plattform, die von mehreren Anwendern parallel genutzt werden kann.

Betrachtet man die FPGAs des Clusters als eine zusammenhängende Logikfläche, setzt eine Fragmentierung der Ressourcen bei der Ausführung verschiedener Aufgaben mit unterschiedlichem Ressourcen- und zeitlichem Rechenbedarf ein. Die Folge daraus ist, dass, eine flexibel konfigurierbare Kommunikationsinfrastruktur vorausgesetzt, zwar im Verlauf Ressourcen wieder frei werden und neu belegt werden können, diese aber nicht zwangsläufig in direkter Nachbarschaft zueinander liegen. Die zugrundeliegende Topologie der Verbindungen zwischen den einzelnen Elementen des Clusters sollte deshalb so gewählt werden, dass die zeitliche Obergrenze für die Übertragung von Daten zwischen zwei beliebigen Elementen minimal ist. Beschreiben lässt sich die Obergrenze über die Anzahl der benötigten Schritte innerhalb des Verbindungspfades. Regelmäßige Strukturen müssen hierfür nicht zwangsläufig die besten Ergebnisse liefern. So wird in dieser Arbeit eine Übersicht über optimale Strukturen für verschiedene durch die zugrunde liegende Hardware festgelegte Parameter gegeben.

Auf Basis dieser Ergebnisse wird ein neues modulares, nicht auf ein Anwendungsgebiet spezialisiertes FPGA-Cluster-System entworfen und mit vorhandenen Lösungen verglichen.

1.3 Struktur der Arbeit

In Kapitel 2 werden zunächst die Grundlagen der FPGA-Technologie erläutert. Im weiteren Verlauf werden die vielfältigen typischen Anwendungsfelder von FPGAs und FPGA-Clustern aufgezeigt. Eine Evaluierung ausgewählter IEEE-gelisteter Veröffentlichungen ermöglicht eine Bewertung der Anforderungen, welche für eine Umsetzung einer Anwendung auf FPGAs bzw. FPGA-Clustern ausschlaggebend ist.

Methoden zur abstrahierten Darstellung der Kommunikationsinfrastruktur von FPGA-Clustern werden in Kapitel 3 erläutert. Im weiteren Verlauf des Kapitels wird ein Optimierungsverfahren vorgestellt, welches auf die Minimierung der maximalen Pfadlänge zwischen zwei Komponenten des zugrunde liegenden Clusters abzielt. Ein Vergleich mit regelmäßigen Strukturen stellt deren Pfadlängen gleichzeitig als Obergrenze für eine Bewertung des Verbindungsgraphen eines Clusters vor und schließt den Abschnitt ab.

In Kapitel 4 werden aktuell in Veröffentlichungen von Forschungsergebnissen aufgeführte FPGA-Cluster vorgestellt. Dabei wird eine Unterscheidung zwischen Architekturen vorgenommen, die für ein dediziertes Anwendungsfeld aufgebaut wurden, und solchen Systemen, die für unterschiedliche Applikationen eingesetzt werden. Der Aufbau der Kommunikationsinfrastruktur innerhalb des Clusters wird für jedes System gesondert herausgestellt.

Die im Rahmen dieser Arbeit entstandenen Komponenten des universellen RAPTOR-XPress-FPGA-Clusters werden in Kapitel 5 vorstellt. Darüber hinaus wird ein Funktionsblock zur Integration in spätere Anwendungen (engl.: Intellectual Property Core, IP-Core) eingeführt, der eine Abstraktionsebene für implementierte Anwendungen über die Kommunikationsschnittstellen legt und dem Anwender die Partitionierung seiner Applikation vereinfacht. Daraufhin erfolgt eine kurze Einführung der zugehörigen Software-Umgebung. Abschließend wird die Leistungsfähigkeit der Kommunikationsinfrastruktur aufgezeigt und das aus Kapitel 3 bekannte Optimierungsverfahren auf einen Cluster-Aufbau mit 64 FPGAs angewendet.

Ein Vergleich der Eigenschaften der in Kapitel 4 und 5 vorgestellten FPGA-Cluster wird in Kapitel 6 vorgenommen. Neben der Gegenüberstellung genereller Parameter wie der Anzahl der Logikzellenäquivalente und des lokal vorhandenen Speichers erfolgt eine Beurteilung der Kommunikationsinfrastruktur der Systeme und der damit realisierbaren Topologien. Der sich daraus ableitende Begriff der Logikdichte wird eingeführt und für einen Vergleich der Systeme herangezogen.

Im siebten und letzten Kapitel werden die Ergebnisse der vorherigen Abschnitte diskutiert.

1.4 Anmerkungen und Schreibweisen

Im Folgenden werden die für diese Arbeit maßgeblichen Darstellungsstandards aufgeführt.

- Zahlenangaben erfolgen nach DIN 1333 [8].
- Grafische Darstellungen in einem Koordinatensystem folgen DIN 461 [9].
- Formelschreibweise und Formelsatz erfolgt nach DIN 1338 [10] mit Ausnahme der *Größer-gleich-* und *Kleiner-gleich*-Darstellung, welche nach DIN 1302 4.5 [11] erfolgen.
- Bit-Angaben in dieser Arbeit werden gemäß DIN EN 80000-13 [12] ausgewiesen.
- Mengen werden gemäß [13] dargestellt und orientieren sich an [14].

2 Grundlagen und Anwendungsgebiete von FPGAs und FPGA-Clustern

Während FPGAs in ihren Anfängen vorrangig die Funktion von Verbindungslogik und die Integration von einfachen logischen Schaltungen einnahmen [15], hat sich das Einsatzgebiet inzwischen auf sämtliche Bereiche der algorithmischen Verarbeitung von Aufgabenstellungen ausgedehnt. So bieten moderne FPGAs die Möglichkeit, vormals nur durch das Zusammenspiel verschiedener ICs auf Systemebene lösbare Problemstellungen integriert auf einem Baustein zu verarbeiten. Entsprechend ist die Aufmerksamkeit, die FPGAs als Forschungsschwerpunkt erhalten, im Lauf der Jahre kontinuierlich angestiegen, wie in Abbildung 2.1 anhand der Anzahl der in der *IEEE Xplore Digital Library* aufgeführten Veröffentlichungen zu sehen ist [276]. Das in den Jahren 2002 bis 2009 sprunghaft angestiegene Interesse stabilisierte sich in den darauf folgenden Jahren auf ein gleichbleibend hohes Maß. Zu erklären ist dieser Umstand damit, dass FPGAs inzwischen auch stark in industriellen und sogar in kommerziellen Produkten eingesetzt werden und damit bereits ein Transfer aus der Forschung stattgefunden hat und weiterhin stattfindet.

Neben dem Beschleunigungsvorteil durch die Reduktion des Kommunikationsaufwands ergibt sich ein Energievorteil aufgrund der Fähigkeit, die benötigten Strukturen Bit-genau auszulegen. Die genaue Anpassung erlaubt einen höheren Datendurchsatz bezogen auf die eingesetzte Chipfläche bei gleichzeitig reduziertem Energieverbrauch [16], als dies mit Mehrzweckprozessoren (engl.: General Purpose Processors, GPP) zu erzielen ist [17].

Ziel dieses Kapitels ist das Gewinnen eines Überblicks über den grundlegenden Aufbau eines FPGAs. Darauf aufbauend werden die verschiedenen Anwendungsgebiete im Bereich der Forschung der letzten 15 Jahre vorgestellt. Im weiteren Verlauf erfolgt darüber hinaus eine Erweiterung der Betrachtung auf FPGA-Cluster. Eine teilweise Überlappung von Aufgabenfeldern entsteht bei den eingesetzten Algorithmen, die häufig nicht eindeutig einem Anwendungsgebiet zugeordnet werden können, sondern Problemstellungen in verschiedenen Bereichen abdecken. Eine Unterscheidung ergibt sich hinsichtlich der zugrunde liegenden Zielanwendung, sofern diese bekannt ist. Andernfalls erfolgt die Gruppierung auf Basis der Herkunft

der zu verarbeitenden Daten. Häufig ergibt sich aus der Kombination von zwei oder mehreren Algorithmen verschiedener Aufgabenfelder eine Zielimplementierung für einen anderen Anwendungszweck.

Das Kapitel wird in drei Abschnitte unterteilt. Zunächst wird der grundlegende technische Aufbau von FPGAs erläutert. Eine kurze Definition des für diese Arbeit relevanten Begriffs FPGA-Cluster schafft eine Überleitung zu der Auswertung von Veröffentlichungen zu Anwendungsgebieten von FPGAs. Im dritten Abschnitt werden die Anwendungsgebiete von FPGA-Clustern aufgeführt und die spezifischen Anforderungen der jeweiligen Implementierung an die Kommunikationsinfrastruktur des eingesetzten Cluster-Systems untersucht.

2.1 Grundlagen der FPGA-Technologie

Kernelement aller heutzutage verwendeten datenverarbeitenden Schaltungen ist die zweielementige Boolesche Algebra bestehend aus den drei Grundoperationen *und* (engl.: AND), *oder* (engl.: OR) und *nicht* (engl.: NOT). Die beiden Grundelemente sind *1* und *0* bzw. aus Sicht der Aussagenlogik *wahr* (engl.: True) und *falsch* (engl.: False). Mit Hilfe von Transistorschaltungen und unter Interpretation verschiedener Spannungspegel als *1* und *0* lassen sich die Grundgatter *negierte Und-Verknüpfung* (engl.: NAND), negierte *Oder-Verknüpfung* (engl.: NOR) und *Inverter* (engl.: NOT) bilden. Die nichtinvertierten Funktionen *und* und *oder* können durch

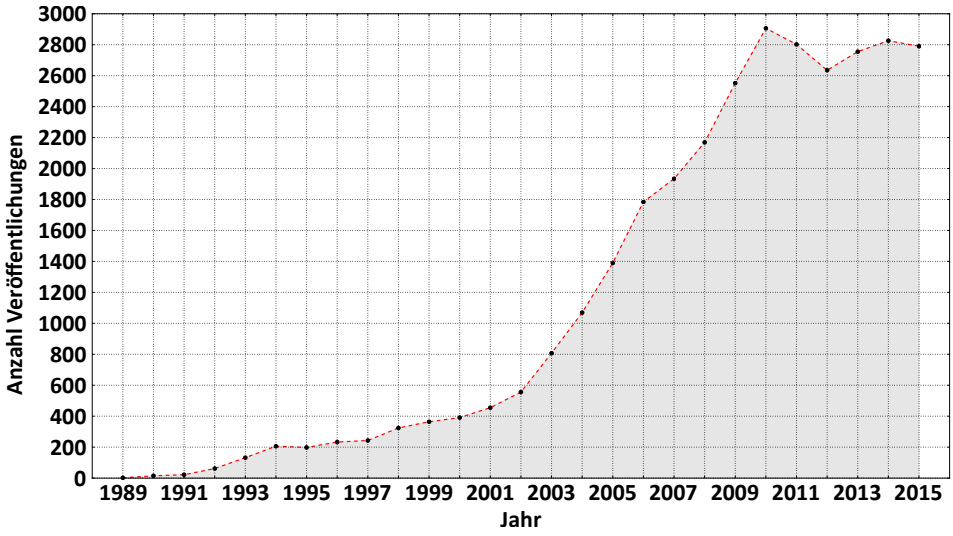


Abbildung 2.1: Anzahl der IEEE-gelisteten FPGA-Veröffentlichungen

Verschaltung der Grundgatter umgesetzt werden, ebenso die häufig gebräuchliche *exklusive Oder-Verknüpfung* (engl.: XOR) bzw. ihre negierte Form (engl.: NXOR). Die Verschaltung dieser Gatter zusammen mit getakteten Speicherelementen wie Registern ermöglicht den Aufbau komplexer logikverarbeitender Strukturen in Silizium.

Werden die Speicherelemente so verschaltet, dass sie ihrerseits in ihrer Funktion programmierbare Logiktabellen (engl.: Look Up Table, LUT) bilden, können programmierbare Logikbausteine aufgebaut werden. Hinzu kommt bei FPGA-Bausteinen eine ebenfalls programmierbare Verbindungsmatrix zwischen den einzelnen LUTs. Kombiniert man eine oder mehrere LUTs mit synchron getakteten Speicherelementen und verschaltet diese Einheiten, so ergibt sich eine Architektur, die das Ergebnis binärer Funktionsbeschreibungen bestimmen kann. Abbildung 2.2 zeigt diese Kombination von farblich hervorgehobenen LUTs und Registern – bei der Firma Xilinx als Slice bezeichnet – zusammen mit verschiedenen Multiplexern (MUX) eines aktuellen Xilinx Virtex-7-FPGAs [18]. Je nach Ausrichtung der Zelle auf Logik- oder Speicherelemente wird zwischen den Typen SLICEL und SLICEM unterschieden; verschaltet werden diese Einheiten zu konfigurierbaren Logikblöcken (engl.: Configurable Logic Block, CLB).

Dieser Aufbau weist eine hohe Flexibilität auf mit dem Nachteil der daraus resultierenden niedrigeren maximalen Systemtaktrate im Vergleich zu starren Strukturen, wie sie beispielsweise bei Prozessoren vorliegen. Durch die Spezialisierung auf eine hochoptimierte Verarbeitungseinheit können deutlich kürzere Wege in der Verschaltungsstruktur gebildet werden, womit die mögliche maximale Taktfrequenz höher ausfallen kann. Gleiches gilt für die noch weiter optimierten Grafikprozessoren, die in ihren Verarbeitungseinheiten, Shader genannt, auf die parallele Berechnung von Vektor- und Matrixoperationen spezialisiert sind. Die Flexibilität von Prozessoren und Grafikprozessoren wird erst auf Software-Ebene erreicht, nicht bereits auf Register-Transfer-Ebene, welche wesentlich näher an der tatsächlichen Hardware liegt. Da die einzelnen Berechnungsschritte auf FPGAs unabhängig voneinander ablaufen können und eine Abhängigkeit der Ergebnisse sich aus der Beschreibung der Verschaltung ergibt, kann die Parallelität der Abläufe maximal ausgeschöpft werden. Da im FPGA parallel berechenbare Teile des Programmablaufs auf einem Prozessor nur linear und möglicherweise in mehreren Schritten umgesetzt werden können, reduziert sich der Vorteil von hohen Taktraten auf die tatsächliche Berechnungsdauer der kleinschrittigsten Operationen. Kombiniert mit der Möglichkeit zur Bit-genauen Umsetzung und zur Implementierung von Pipelinestufen, um die Länge der Verkettung mehrerer Berechnungsschritte gering zu halten, können FPGAs in vielen Bereichen effektiv als Verarbeitungseinheit eingesetzt werden, wie in dieser Arbeit gezeigt wird. Die auf den FPGA abgebildete Funktionalität kann jederzeit neu konfiguriert werden, d. h., dass die FPGA-Matrix zunächst keine Grundfunktionalität bereithält, sondern erst mit Programmierung mit einem sogenannten Bitstrom (engl.: Bitstream) verwendbar wird. Der Bitstrom enthält sämtliche Infor-

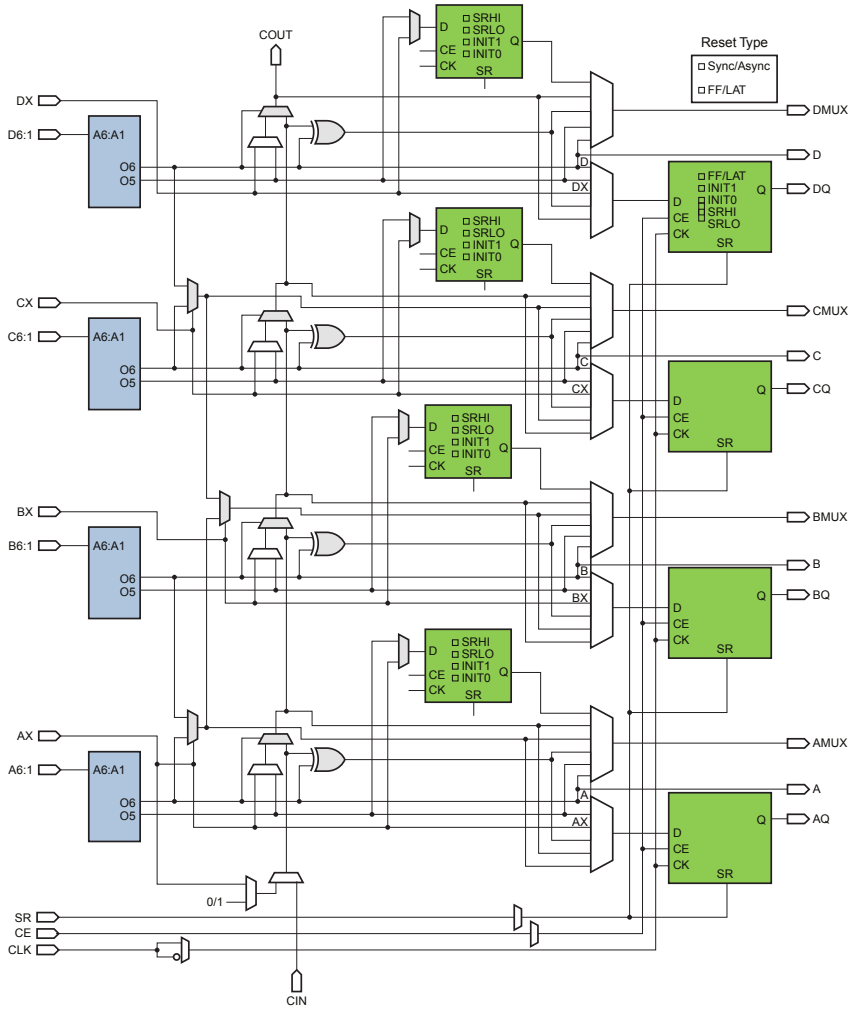


Abbildung 2.2: Übersicht über ein Xilinx Virtex-7-SLICEL [19]

mationen über die in den LUTs abgebildeten Funktionen und der Verschaltung der Verbindungsmatrix. Diese Rekonfiguration kann im laufenden Betrieb und auch für Teilbereiche des FPGAs erfolgen. Um den Nachteil zu umgehen, dass vor Benutzung des FPGAs von einem Hostsystem oder einem parallel angeordneten nichtflüchtigen Speicher ein Bitstrom geladen werden muss, sind FPGA-Technologien entwickelt worden, welche ihrerseits auf nichtflüchtige Speicherelemente wie beispielswei-

se Flash-Speicher aufsetzen. Der Großteil der verwendeten FPGAs nutzt jedoch SRAM-Speicher als Basiselement ihrer LUTs, weshalb eine Konfiguration nach dem Einschalten notwendig ist.

Analog zu den unterschiedlichen Architekturkonzepten von FPGAs und Prozessoren gestaltet sich der Entwurfsablauf für die Implementierung von Aufgaben auf der Zielhardware. Während die für Prozessoren und ähnliche Verarbeitungseinheiten übliche Vorgehensweise zur Umsetzung einer Problemstellung eine Beschreibung der Abarbeitung der linear aufeinander folgenden Lösungsschritte in Form von Software ist, müssen für die Nutzung von FPGAs eine Beschreibung der Verschaltung der LUTs erstellt und die umzusetzende logische Funktion jedes einzelnen LUT generiert werden. Dies geschieht im Allgemeinen über eine abstrahierte Hardwarebeschreibungssprache und einen Übersetzungsschritt (Synthese). Das Ergebnis der Synthese kann im nächsten Schritt auf der eigentlichen Zielhardware umgesetzt werden (engl.: Place & Route). Dieser zweite Schritt ist bei einer Softwareübersetzung durch einen Compiler nicht erforderlich, da der Auswahlraum sich auf den Befehlssatz des Prozessors beschränkt und die Struktur der Hardware fest vorgegeben ist.

FPGA-Cluster

Äquivalent zu der Bündelung von Prozessoren und darauf aufsetzenden Serversystemen zu Funktionsgruppen, sogenannten Clustern, können mehrere FPGAs in einer Cluster-Struktur kombiniert werden. Dies geschieht meist aufgrund der Tatsache, dass sich große Logikbeschreibungen nicht mehr vollständig auf einem einzelnen FPGA-Baustein abbilden lassen und deshalb auf mehreren FPGAs partitioniert umgesetzt werden müssen. Wie die Verbindungen der einzelnen FPGAs zueinander aussehen, entscheidet maßgeblich darüber, ob im Rahmen dieser Arbeit von einem FPGA-Cluster gesprochen wird oder ob das System als Mehr-FPGA-System (MFS) bezeichnet wird. Der Begriff Kommunikationsinfrastruktur beschreibt diese Verbindungen. Aufbauten, die über eine hohe Anzahl an FPGAs verfügen, welche wiederum nicht oder nur sehr gering miteinander vernetzt sind, werden im Folgenden als MFS, nicht als FPGA-Cluster bezeichnet. Die Skalierbarkeit, also die Möglichkeit weitere FPGAs dem Cluster hinzuzufügen, ist ein für diese Arbeit relevanter Aspekt zur Einordnung des Systems in die Kategorie FPGA-Cluster. In Kapitel 4.1 werden sowohl der im Kontext dieser Arbeit verwendete Begriff FPGA-Cluster als auch die Abgrenzung zu MFS definiert.

2.2 Anwendungsgebiete von FPGAs

Wie bereits Moores Gesetz [4] und Amdahls Gesetz [20] aufzeigen, sind die Verkleinerung der zugrunde liegenden Transistorstrukturen und die damit mögliche

Steigerung der Taktfrequenz vorrangig ausschlaggebend für die Entwicklung von neuen Computer-Architekturen. Seit dem Jahr 2000 findet dazu ergänzend eine Steigerung der Parallelität statt, was auf Chip-Ebene hauptsächlich durch Vervielfachung der vorhandenen Prozessoren (engl.: Central Processor Unit, CPU) geschieht. Jedoch zeigt Amdahl auch, dass Anwendungen nur bis zu einem gewissen Grad, nämlich dem parallelisierbaren Anteil entsprechend, beschleunigt werden können [21]. Hinzu kommen Schwierigkeiten beim Zugriff auf notwendigerweise geteilte Ressourcen wie Speicher oder Peripherie. Aus diesem Grund ist abzusehen, dass eine unbegrenzte Parallelität auf Prozessorebene trotz der weiterhin steigenden Transistorzahl pro Chipfläche [22] nicht zu erzielen ist. Gleiches gilt aufgrund physikalischer Beschränkungen für die Steigerung der Taktraten. Diese Einschränkung führt laut [23] dazu, dass konfigurierbare Architekturen in der nahen Zukunft eine immer größere Rolle im HPC-Bereich einnehmen werden.

In verschiedenen Veröffentlichungen wird die Leistungsfähigkeit von Umsetzungen auf GPPs, GPGPUs und FPGAs verglichen, so zum Beispiel in [24] und [25]. Bei den Ergebnissen wird jedoch häufig nicht der Optimierungsaufwand der Implementierung berücksichtigt. Zwar weist [25] auf die vierfache Entwicklungszeit der FPGA-Umsetzung hin, bietet jedoch keinen Vergleich mit einer mit ähnlichem Aufwand optimierten Umsetzung auf GPGPUs. Auf dem thermischen Energiebudget (engl.: Thermal Design Power, TDP) des jeweiligen Bausteins basierend wird in [26] angeführt, dass FPGAs eine um eine Größenordnung energiesparendere Umsetzung erlauben als GPGPUs. Zum einen wird auch hier keine Angabe bezüglich des Optimierungsaufwands der Anpassung des Algorithmus an die jeweilige Zielarchitektur gemacht, zum anderen wird vernachlässigt, dass die FPGA-Implementierung 3,5-mal langsamer ist und nur in Festkommadarstellung rechnet, im Gegensatz zur Fließkommazahlenberechnung auf CPUs und GPGPUs.

Im Folgenden wird ein Überblick über verschiedene Anwendungsfelder von FPGAs im Bereich der Forschung gegeben. Darüber hinaus finden FPGAs Anwendung im industriellen Umfeld, beispielsweise im Bereich der industriellen Kommunikation [27], und übernehmen zunehmend auch Aufgaben in Endverbraucherprodukten wie Fernseher [28]. Der Fokus der weiteren Betrachtung liegt auf den in der Forschung relevanten Anwendungsfeldern.

Die in Abbildung 2.3 gegebene Übersicht über die Anwendungsgebiete wurde unter Berücksichtigung der aktuellen Forschungsschwerpunkte des jeweiligen Jahres im Bereich der Implementierung oder Portierung von Problemstellungen auf FPGAs erstellt. Eine zufällige Auswahl von jeweils 50 Veröffentlichungen pro Jahr aus insgesamt weit über 20.000 in der elektronischen IEEE-Datenbank gelisteten Resultaten zum Stichwort FPGA erlaubt einen Überblick über die Schwerpunkte innerhalb der letzten 15 Jahre. Als eigenständiges Anwendungsgebiet werden solche Veröffentlichungen aufgeführt, die in mindestens zwei unterschiedlichen Jahren erschienen sind und mit denen sich zwei oder mehrere Autoren befasst haben. Zwar lassen sich einige Anwendungsfelder als Untergruppe von anderen Be-

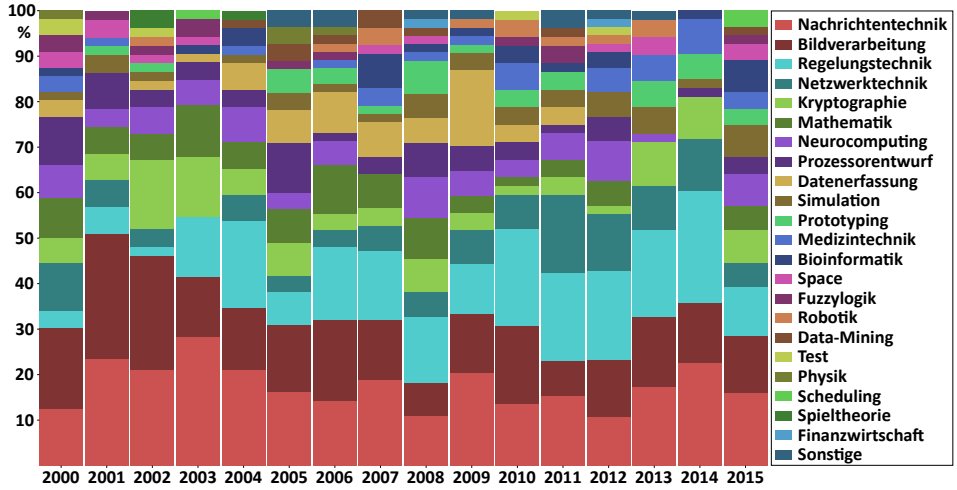


Abbildung 2.3: Jahresübersicht über Anwendungsgebiete von FPGAs

reichen kategorisieren, beispielsweise stellen Teile der Implementierungen aus der Kategorie Physik mehrheitlich Simulationsumgebungen dar, da sich jedoch weitere Verwendungen finden lassen, erfolgt eine Aufstellung als jeweils eigenes Themengebiet. Dennoch führen mögliche Überschneidungen von Anwendungsgebieten zu einer Nennung in beiden Bereichen. Durch die abschließende Normierung mit 50 ergibt sich die dargestellte prozentuale Verteilung. Zur Verdeutlichung der Relevanz der einzelnen Themenfelder stellt Abbildung 2.4 den prozentualen Anteil an den untersuchten Veröffentlichungen dar. Die Sortierung der einzelnen Anwendungen in beiden Abbildungen geschieht auf Basis der Relevanz in den Jahren 2000 bis 2015. Zur besseren Übersichtlichkeit wird die Farbwahl in beiden Darstellungen gleich gehalten. Bei der folgenden Einzelvorstellung der Anwendungsgebiete wird deren Relevanz anhand ausgewählter Veröffentlichungen aufgezeigt, welche nicht notwendigerweise Teil der vorherigen zufälligen Auswahl der Übersichtsgrafiken sind.

Deutlich ersichtlich ist, dass die drei wichtigsten Anwendungsfelder für FPGAs Algorithmen aus den Bereichen Nachrichtentechnik (17 %), Bildverarbeitung (15 %) und Regelungstechnik (14 %) stammen. Allen drei Aufgabenfeldern ist gemein, dass mit wenigen Ausnahmen der Beschleunigungsansatz eine wesentlich größere Rolle für die Umsetzung auf einem FPGA einnimmt als die Verbesserung der Energieeffizienz der Implementierung. Harte Echtzeitanforderungen speziell im Bereich Nachrichtentechnik und Regelungstechnik begründen den Aufwand einer Portierung von DSPs oder Mikrocontroller zu FPGAs.

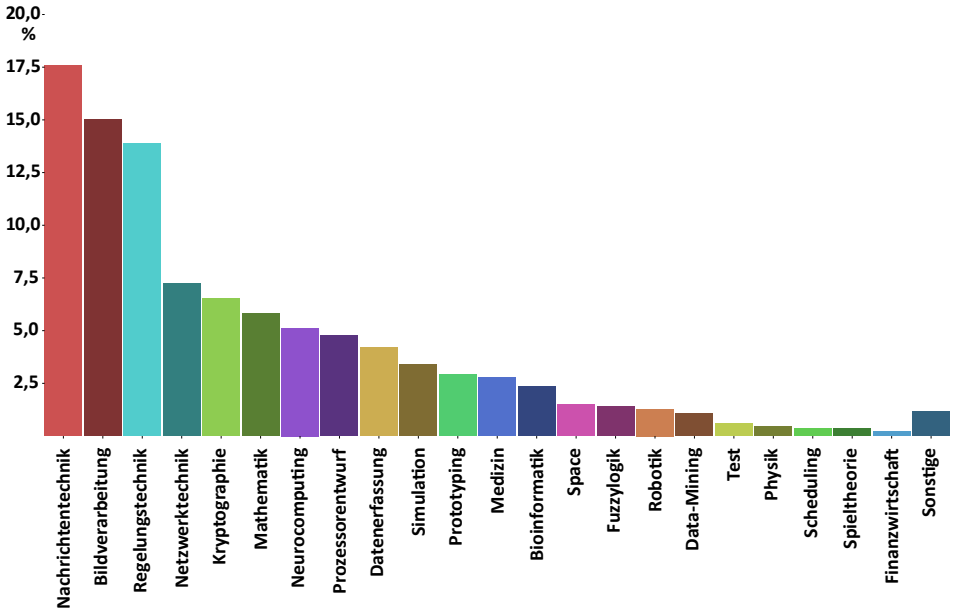


Abbildung 2.4: Themenschwerpunkte für Anwendungsgebiete von FPGAs

2.2.1 Bildverarbeitung

Im Rahmen FPGA-basierter Bildverarbeitung wird ein Eingangsbild oder ein Videostrom manipuliert oder ausgewertet, so dass ausgangsseitig entweder ein Bild bzw. Videostrom oder ein Datensatz mit spezifischen extrahierten Charakteristiken wie detektierten Kanten oder Entfernungen und Positionen von Objekten bereitgestellt wird. Verschiedene Applikationen werden in [29] beleuchtet. Der aufgeführte Vorteil von FPGAs liegt in der hohen Parallelität der implementierten Verarbeitungseinheiten, was die Durchführung von Filteroperation in Echtzeit ermöglicht. Typische Operationen bei der Vorverarbeitung von Eingangsdaten sind Farbraumkonvertierung [30], Kontrastanreicherung [31] und Weißabgleich für unterschiedliche Umgebungsbeleuchtungen [32], während Beispiele für die Weiterverarbeitung und Merkmalsextraktion Kantendetektion [33], Objekterkennung [34], Objektverfolgung [35] oder Stereoskopie [36] sind. Programmiermodelle wie in [26] abstrahieren dabei den FPGA für die Streaming-Anwendung mit dem Ziel, eine einheitliche Plattform für die verschiedenen Anwendungen zu schaffen. Die gewonnenen Informationen können zum Beispiel zur Navigation genutzt werden, wodurch sich eine Überschneidung mit dem im Folgenden aufgeführten Anwendungsgebiet der Robotik ergibt [37].

2.2.2 Bioinformatik

Die Berechnungen von DNA- und Protein-Sequenzanalysen lassen sich stark parallelisieren, weshalb sie auf skalaren Prozessorarchitekturen, welche auf die sequenzielle Abarbeitung von Aufgaben hin optimiert sind, geringere Verarbeitungsgeschwindigkeiten aufweisen als auf langsamer getakteten parallelen Systemen. Der Vergleich von Gen- oder Proteinsequenzen mit einer anderen Sequenz zur Verdeutlichung von Ähnlichkeiten ist ein Weg, mögliche gleiche Funktionalität innerhalb unterschiedlicher Sequenzen zu detektieren. Bei den beiden am häufigsten verwendeten Algorithmen handelt es sich um Needleman-Wunsch [38] bzw. Waterman-Smith-Beyer [39] und Smith-Waterman [40]. Aufgrund der Komplexität der Verfahren wird vielfach der heuristische Basic-Local-Alignment-Search-Tool-Algorithmus (BLAST) [41] verwendet, welcher jedoch eine geringere Genauigkeit gegenüber Smith-Waterman aufweist. Entsprechend lassen sich Veröffentlichungen zu allen genannten Verfahren finden, in denen speziell angepasste Rechensysteme (engl.: Custom Computing Machine, CCM) zur Beschleunigung auf FPGAs implementiert werden [42].

Während Sequenzanalysen stark datenlastig sind, sind das Aufspannen phylogenetischer Bäume und die Bestimmung der Verwandtschaft von Organismen eine rechenintensive Aufgabe [43]. In [44] wird eine Implementierung der phylogenetic-likelihood-Funktion aufgezeigt, wobei die Beschleunigung durch die notwendigen Fließkommaberechnungen begrenzt ist.

Für die Simulation von metabolischen Prozessen wird in [45] ein im FPGA implementierter aufgabenspezifischer Prozessor vorgestellt und dessen Funktionalität am metabolischen Netzwerk des E. Coli Bakteriums nachgewiesen.

2.2.3 Data-Mining

Im Bereich Data-Mining gilt es, aus einer großen Menge an Daten für eine bestimmte Fragestellung relevante Informationen zu extrahieren [46]. Die eigentlichen Anwendungsbereiche, aus denen die Daten stammen, sind dabei vielfältig, ebenso das gesuchte Ereignis. So kann es notwendig werden, Werte zu klassifizieren [47], Anomalien aufzuzeigen [48] oder eine Reduktion hochdimensionaler Daten auf bestimmte Merkmale [49] durchzuführen. Mit Hilfe der in diesem Abschnitt vorgestellten Verfahren können große Datenmengen zielgerichtet untersucht werden. Die drei am häufigsten erwähnten Implementierungen sind Classification-And-Regression-Tree- (CART) bzw. Entscheidungsbaum- (engl.: Decision Tree Classification, DTC)[50], Mustererkennungs- (engl.: Support Vector Machine, SVM) [51] und k-Means-Algorithmen.

Künstliche neuronale Netze (engl.: Artificial Neural Network, ANN) werden ebenfalls im Bereich des Clustering von Informationen eingesetzt und zeigen bessere

Resultate als k-mean-Algorithmen [52]. Aufgrund der häufigen Nennung werden ANNs in Abschnitt 2.2.12 gesondert vorgestellt.

Als verteilter Ansatz für die Suche nach vorgegebenen Mustern wie Text, Gensequenzen oder Bilddaten platzieren die Autoren aus [53] einen FPGA direkt am Massenspeicher, welcher die zu überprüfenden Daten enthält. Der FPGA hat somit einen wesentlich direkteren Zugriff auf die zu untersuchenden Informationen als der Systemprozessor, wodurch bandbreitenlimitierende kaskadierte Bussysteme bei der Anforderung von Daten ausgespart werden. Somit ergibt sich neben der durch den FPGA beschleunigten Verarbeitung des Suchalgorithmus ein weiterer Performanzvorteil gegenüber Prozessoren, welche die Daten zunächst in ihren lokalen Cache laden müssen.

2.2.4 Datenerfassung

Zum Themenbereich Datenerfassung gehören Aufgaben wie die Ansteuerung und Abfrage verschiedener Sensoren und die anschließende Übermittlung der teilweise vorverarbeiteten Daten an andere Stellen. Durch verschiedene Formen der Aufbereitung wie beispielsweise Filterung, Fusion verschiedener Sensoren, Protokollanpassung oder Kompression der aufgenommenen Werte geht die Verwendung der FPGAs häufig weit über den als einfache Verbindungslogik angesehenen Bereich der Datenerfassung hinaus.

Die Nutzung der Strukturen des FPGAs selbst als Sensor zur Erfassung verschiedener Messgrößen ist ebenfalls in diesem Themenfeld erfasst. So zeigt beispielsweise [54] ein Verfahren auf, bei dem mit Hilfe der regelmäßigen Strukturen ausgewählter Ein- und Ausgangsblöcke (engl.: In/Out, IO) des FPGAs eine präzise Zeiterfassung möglich ist. Ein ähnliches Ziel hat die Implementierung von [55], welche die Daten von schnellen Analog-Digital-Konverter-Kanälen (engl.: Analog Digital Converter, ADC) abfragt, auftretende Signalspitzen aufnimmt und für eine präzise zeitliche Zuordnung sorgt.

2.2.5 Finanzwirtschaftliche Anwendungen

Bei der Platzierung von Finanzprodukten an den stark schwankenden internationalen Börsen müssen Startpreise und Renditeerwartungen im Voraus bestimmt werden, um Gewinnerwartungen und Risikominimierung für das emittierende Unternehmen kalkulierbar zu halten. Durch Simulation verschiedener Marktszenarien lassen sich realistische Prognosen über die Produktentwicklung treffen [56]. In [57] wird neben der durch die Parallelität erzielten Beschleunigung der Monte-Carlo-Simulation die Anpassbarkeit der Genauigkeit einzelner Simulationszweige hervorgehoben, wodurch die Ergebnisse schneller vorliegen. Darüber hinaus bietet die Energieeffizienz von FPGAs gegenüber CPU- und GPU-Systemen einen Vorteil in Bezug auf die Betriebskosten der HPC-Systeme.

Zeitnahe Reaktionen auf Kursveränderungen im automatisierten Handel sind maßgebend für erfolgreiche Handelsentscheidungen. Die in [58] aufgezeigte FPGA-Implementierung erlaubt dank ihrer harten Echtzeitfähigkeit eine parallele Beobachtung und Auswertung verschiedener Kursentwicklungen und bietet somit schneller und mit gleichbleibender Latenz Informationen für Transaktionen, als dies CPU-basierte Lösungen vermögen. Durch die parallele Verarbeitung der Daten aus verschiedenen Quellen können darüber hinaus Unsicherheiten, wie sie durch ausbleibende Pakete im Datenstrom entstehen, kompensiert werden [59].

2.2.6 Fuzzylogik

Als ein besonderer Unterpunkt der Regelungstechnik, welche in Abschnitt 2.2.16 vorgestellt wird, sind Regler auf Basis der Fuzzylogik-Theorie anzusehen. Die Komplexität der Regelung wird durch die Einführung von Unsicherheiten reduziert, etwa durch die beiden von Klir und Folger formulierten Unsicherheiten *Unklarheit* und *Mehrdeutigkeit* [60]. FPGAs erlauben eine Umsetzung der Reglerstrukturen unter den an sie gestellten Echtzeitbedingungen [61].

Die Anwendungsgebiete sind sehr breit gestreut und überschneiden sich entsprechend mit den Feldern der klassischen Regelungstechnik, beispielsweise bei der Substitution von PID-Reglern durch eine entsprechende Fuzzylogik-Regelung [62]. Häufig aufgeführt werden Regelungen im Automobilsektor etwa von DC-DC-Spannungswandlern [63] oder von Elektromotorsteuerungen [64].

2.2.7 Kryptologie

In [65] wird aufgezeigt, dass Algorithmen aus dem Bereich des Verschlüsselungsbrechens nicht nur durch die Implementierung von Teilaspekten innerhalb eines rekonfigurierbaren Bausteins gelöst werden können, sondern massiv-parallele Systeme benötigt werden, welche speziell auf den abzubildenden Algorithmus abgestimmt sind. Die massive Parallelität der FPGA-Strukturen wird in der Mehrheit der Veröffentlichungen zum Thema Kryptologie für das Brechen einer bestehenden Verschlüsselung verwendet. Verschiedene Methodiken kommen dabei zur Anwendung. Die häufigste Vorgehensweise ist die Beschleunigung von einfachen Wörterbuchangriffen, beispielsweise auf die verbreitete Festplattenverschlüsselung Truecrypt in [66] oder auf symmetrische Verschlüsselungen wie in [67]. Asymmetrische Verfahren wie die Kryptographie mit elliptischen Kurven werden ebenfalls betrachtet [68].

Auch zur Verschlüsselung von Daten dienen FPGAs bei der beschleunigten Berechnung der rechenintensiven Algorithmen, so zum Beispiel bei der Kryptographie mit elliptischen Kurven [69] oder der Berechnung der Algorithmen des Advanced-Encryption-Standard (AES) [24].

2.2.8 Mathematik

Neben den eigentlichen logikverarbeitenden CLBs verfügen moderne FPGA-Architekturen über dedizierte DSP-Blöcke [70], welche komplexe Berechnungen mit wesentlich höheren Geschwindigkeiten erlauben, als es durch eine Abbildung derselben Funktionalität in CLBs möglich wäre. Darüber hinaus können durch die Verlagerung der Berechnung Logikressourcen für andere Teile der Implementierung freigehalten werden. Die flexible Verschaltung der Blöcke ermöglicht die Portierung von mathematischen Verfahren, welche alternativ auf DSPs berechnet werden, wodurch zusätzliche externe DSPs substituiert werden können, wie dies beispielsweise in [71] sowie für Fließkommaberechnungen in [72] gezeigt wird.

2.2.9 Medizintechnik

Im medizinischen Umfeld finden FPGAs häufig Anwendung im Grenzbereich zu anderen hier aufgeführten Anwendungsgebieten. FPGAs übernehmen beispielsweise Aufgaben der Bildverarbeitung bei der Aufbereitung von 2D- und 3D-Sonographieaufnahmen [73, 74] oder erfassen Daten im Bereich der Schlafapnoe-Syndrom-Überwachung [75]. Das Detektieren eines Atemaussetzers geschieht, wie in [76] beschrieben, über einen permanent getragenen Ultraschallsensor mit FPGA-basierter Auswertung. Des Weiteren können Aussetzer anhand der Messung des Luftflusses durch die Nase bzw. der Ausdehnung des Torax' beim Atmen und der anschließenden Sensorfusion der gewonnenen Daten erkannt werden [75]. Für die Diagnose von Angina Pectoris anhand von EKG-Daten stellt [77] eine FPGA-Lösung vor, welche die vollständige Signalverarbeitung bis hin zur Auswertung und dem Aufzeigen von Unregelmäßigkeiten im EKG-Verlauf implementiert. Die in [78] vorgestellte FPGA-basierte Datenauswertung von direkt im Gehirn aufgenommenen neuronalen Aktivierungspotentialen dient beispielsweise dem Ziel einer Ansteuerung von einem Roboterarm.

2.2.10 Nachrichtentechnik

Ein spezielles Feld der digitalen Signalverarbeitung stellt der Bereich software-defined-radio dar. Ziel ist es, einen Großteil der fest verdrahteten Analogkomponenten durch flexible programmierbare digitale Strukturen zu ersetzen. Dies erlaubt die Unterstützung verschiedener Zwischenfrequenzen sowie eine Bandbreitenanpassung und ermöglicht es, unterschiedliche Modulations- sowie Kodierungsverfahren austauschbar und an sich ändernde Umgebungsbedingungen anpassbar zu gestalten. In [79] wird aufgezeigt, dass FPGAs multiplizier- und additionsschrittlastige Berechnungen (engl.: Multiply Accumulate, MAC) durchführen, welche typischerweise mit Hilfe von DSPs gelöst werden. Die schnelle Fourier-Transformation (engl.: Fast Fourier Transform, FFT), Filter mit endlicher Impulsantwort (engl.: Finite

Impulse Response, FIR), digitale Übertragungsbandwandler (engl.: Digital Down Converter, DDC) sowie Bandpasssignalwandler (engl.: Digital Up Converter, DUC) sind Beispiele hierfür. Der Vorteil liegt in der Integration der Signal- und Datenverarbeitung in einem einzelnen Baustein ohne zusätzliche dedizierte Hardware und mit entsprechend performantem Datentransfer von einer in die andere Verarbeitungseinheit. Jedoch wird ebenfalls auf die Schwierigkeit einer ressourceneffizienten Umsetzung von Fließkommaoperationen im FPGA hingewiesen. Zu erklären ist dies durch den Umstand, dass zur Zeit der Veröffentlichung Virtex-2-FPGA-Architekturen vorhanden waren, die zwar über 18×18-Multiplizierer verfügen [80], in Bezug auf den Leistungsumfang jedoch nicht an die Möglichkeiten der DSP48-Blöcke, auch als XtremeDSP [81] bezeichnet, in den FPGAs der Virtex-4-Generation heranreichen. Im Bereich der optischen Nachrichtentechnik werden FPGAs eingesetzt um beispielsweise digitale Polarisationsfolger zu implementieren [277].

2.2.11 Netzwerktechnik

Ähnlich wie in Abschnitt 2.2.1 ausgeführt, profitiert die Verarbeitung kontinuierlicher Datenströme stark von der Architektur von FPGAs. So können zum einen die Pipelining-Stufen an den durch die Applikation geforderten Durchsatz angepasst werden, zum anderen ist es möglich relevante Informationen aus dem entgegengenommenen Datenpaket zu extrahieren und parallel zu verarbeiten. Im industriellen Bereich lassen sich für spezielle Sonderfunktionen adaptierbare Ethernet-Switch-IP-Cores finden, welche auf den Einsatzzweck individuell angepasst werden können, um zum Beispiel bestimmte Pakete zu filtern, den Netzwerkverkehr zu überwachen oder die Priorisierung von bestimmten Daten (engl.: Quality of Service, QoS) zu implementieren [27]. In [82] wird ein Überblick über die Verlagerung von angriffentdeckenden (engl.: Network Intrusion Detection System, NIDS) und angriffvereitelnden Sicherheitsfunktionen (engl.: Network Intrusion Prevention System, NIPS) von reinen Softwareaufgaben auf FPGAs gegeben. Aufgrund der steigenden Anforderungen an NIDS stößt eine klassische Software-basierte sequentielle Verarbeitung an ihre Grenzen, wenn es darum geht, immer höhere Datendurchsätze zu verarbeiten. Die Flexibilität, Software an neue Anforderungen anzupassen, kann darüber hinaus durch dedizierte Hardwarelösungen (ASICs) nur eingeschränkt gewährleistet werden. Durch den Einsatz von FPGAs wird nicht nur eine Beschleunigung der Verarbeitung erreicht. Die Möglichkeit zur Rekonfiguration erlaubt zudem eine kontinuierliche Anpassung an neue Bedrohungslagen. Eine im wissenschaftlichen Kontext häufig erwähnte Plattform in diesem Zusammenhang ist NetFPGA [83, 84].

Die im Jahr 2003 verfügbaren Virtex-II-Pro-X-FPGAs [85] besitzen bereits serielle Hochgeschwindigkeitstransceiver (engl.: Multi-Gigabit Transceiver, MGT), mit deren Hilfe hohe Datenaufkommen übertragen werden können, welche allerdings nicht an ein spezielles Protokoll gebunden sind. Entsprechend lassen sich verschiedene

Applikationen und Systembrücken zwischen unterschiedlichen Hochgeschwindigkeitskommunikationsstandards abbilden. Anhand eines datenflusskontrollierten Paket-Switch für optische Verbindungen innerhalb eines Datenzentrums wird in [86] ein Beispiel für eine niederlatente Kommunikationsstruktur mit hoher Datenrate gegeben.

Im Bereich der Echtzeitkommunikation, also der Datenübertragung mit festgelegten einzuhaltenden Latenzen, lassen sich ebenfalls für die Protokollverarbeitung verantwortliche Umsetzungen auf Basis von FPGAs finden. So beschreibt [87] eine Implementierung eines FlexRay-Feldbus-Controllers, welcher unabhängig von der ihm zugeordneten Host-CPU das Senden und Empfangen von Nachrichten übernimmt.

2.2.12 Neurocomputing

Die nach [88] grundlegenden Eigenschaften von ANNs sind Parallelität, Modularität und dynamische Adaption. Parallele Strukturen lassen sich gut auf die Logikressourcen von FPGAs abbilden. Dennoch gestaltet sich die Umsetzung mit einer hohen Anzahl an Neuronen schwierig, da die in der Berechnung innerhalb der Neuronen notwendigen Multiplikationen flächenintensiv in Bezug auf die Anzahl der benötigten Logikzellen sind. Neben einer Reduktion der Neuronenzahl erlauben FPGAs durch ihre Fähigkeit zur Rekonfiguration das zeitlich versetzte Abbilden und Berechnen von bestimmten Teilen des Netzes [89]. Die Anpassung der Genauigkeit der Berechnungen an die zu erzielende Ergebnisgüte ist ein weiterer Ansatz, um Logikressourcen zu schonen und dennoch aussagekräftige Ergebnisse zu erzielen [90].

Typische Umsetzungen künstlicher neuronaler Netze sind im Folgenden aufgelistet:

- Deep Belief Networks [91]
- Factored Restricted Boltzmann Machine [92]
- Kohonen Self-Organizing Map [93]
- Convolutional Neural Network [94]

Besondere Beachtung finden Umsetzungen von Deep-Learning-Verfahren getrieben durch die Forschungsbestrebungen [95] internationaler Unternehmen wie Microsoft, deren Ziel es ist, Suchmaschinen (energie)effizienter zu gestalten und funktionell zu erweitern. Weitere Optimierungsziele liegen in der Spracherkennung und der Umsetzung von natürlicher Sprache [96].

2.2.13 Physikalische Simulationen und Datenerfassung physikalischer Vorgänge

Im Bereich physikalischer Simulationen von beispielsweise strömungsmechanischen Modellen [97] können FPGAs zur Verkürzung der Laufzeit beitragen. Ebenfalls zu Überschneidungen mit anderen hier aufgeführten Anwendungsfeldern kommt es im Bereich der Datenerfassung von Experimentverläufen. Eine gesonderte Erfassung des Anwendungsgebiets erfolgt, sofern der eingesetzte FPGA weitere Aufgaben über die reine Datenerfassung hinaus erfüllt, etwa Teile der Auswertung übernimmt wie beispielsweise beim Verfolgen des Bewegungsverhaltens einzelner Partikel in einem Magnetfeld [98]. In dem in [99] vorgestellten Aufbau ist bereits die Datenerfassung selbst an eine komplexe Mustererkennung gekoppelt worden, die im FPGA ausgeführt wird.

2.2.14 Prototyping mikroelektronischer Schaltungen

Eines der ursprünglichen Haupteinsatzgebiete von FPGAs ist das Rapid Prototyping von Schaltungsentwürfen hinsichtlich ihres Verhaltens und zur Überprüfung der Leistungsfähigkeit. Prozessorentwürfe und die Evaluation von Erweiterungen und Koprozessoren sind häufige Themen im Prototyping. Ebenso lassen sich komplette Ein-Chip-Systementwürfe (engl.: System on Chip, SoC), global funktional überprüfen, wie beispielsweise in [278] dargestellt, aber auch einzelne Teilaspekte wie der Einfluss von Kommunikationsparametern der Netzwerkstruktur eines Network-on-Chip (NoC) können gezielt evaluiert werden [100]. Die prototypische Umsetzung eines DVB-2-Empfängers zur Abschätzung der benötigten Ressourcen einer zukünftigen ASIC-Umsetzung wird in [101] gezeigt. Die lokal erzielbaren Taktraten liegen selbst auf aktuellen FPGAs wie dem Kintex 7 Ultra Scale ca. eine Größenordnung unter den in ASICs möglichen Frequenzen. So erreicht der FPGA-interne Block-Speicher (engl.: Block RAM, BRAM), der ein zentrales Element der meisten FPGA-Umsetzungen ist, eine Taktrate von 660 MHz [102], welche je nach Routing der Verbindungsstrukturen nur in einem lokal begrenzten Bereich zu erzielen sind. Dem gegenüber werden bei ASICs Frequenzen von 5,5 GHz erreicht [103].

2.2.15 Prozessorentwurf

Abseits der Möglichkeit Prozessorsysteme prototypisch auf FPGAs zu realisieren, wie dies in [104] mit einem VLIW-Prozessor aufgezeigt wird, werden FPGAs immer häufiger selbst als Erweiterung oder Ersatz von Prozessoren eingesetzt. Bereits der Virtex-II Pro [105] kann mit seinen beiden PowerPC-Prozessorkernen als SoC angesehen werden und mit Betriebssystemen wie Linux genutzt werden.

Verschiedenartige spezialisierte Prozessoren, bei denen eine Umsetzung als ASIC nicht interessant ist, was häufig auf wirtschaftliche Gründe zurückzuführen ist,

können mit Hilfe von FPGAs dennoch einen beschleunigten Programmablauf gewährleisten, im Gegensatz zu einer Emulation auf Standard-Prozessoren. Ein Beispiel hierfür ist der Radix-2-FFT-Prozessor mit einer Auflösung von 1024 Punkten [106]. Im Fall des in [107] vorgestellten auf die Ausführung von Java Card Applets hin optimierten Koprozessors steht neben der beschleunigten Ausführung der Aspekt der Absicherung gegen unbefugtes Auslesen im Vordergrund der FPGA-Implementierung. Ein Vergleich einer FPGA-Lösung für die Berechnung der Lattice Quantenchromodynamik von Quarks und Gluonen mit einem spezialisierten ASIC- und PC-Cluster wird in [108] gegeben, wobei verschiedene Fließkommadarstellungen zur Anwendung kommen, um deren Güte bei der Berechnung mit den anderen Plattformen vergleichen zu können.

2.2.16 Regelungstechnik

Im Bereich der Regelung industrieller Anwendungen finden sich, wie in [109] aufgezeigt, vermehrt FPGAs, die die Echtzeitfähigkeit der Regelung sicherstellen. Beispiele für die Anwendung von FPGAs in regelungstechnischen Applikationen sind die Regelung von verschiedenen Typen von Leistungswandlern [110, 111], Leistungsfaktorkorrekturen (engl.: Power Factor Correction, PFC) [112] sowie Antriebs- und Bewegungssteuerungen [113, 114].

Die besondere Eigenschaft der partiellen Rekonfiguration, nämlich die Möglichkeit Logik-Ressourcen zeitlich unterschiedlich zu nutzen, wird in [115] am Beispiel eines inversen Pendels aufgezeigt. Zwei unterschiedliche Regler, die im Zeitmultiplex-Verfahren denselben Bereich im FPGA belegen, werden abhängig vom Zustand des Pendels zur Laufzeit geladen und setzen ihre jeweilige Regelstrategie um, ohne dass die Verwendung eines größeren FPGA-Bausteins notwendig ist.

2.2.17 Robotik und mechatronische Systeme

Obgleich im Bereich der Robotik Mikrocontroller-basierte Plattformen dominieren, wie beispielsweise in [283] ausgeführt, existieren dennoch mehrere Veröffentlichungen zu einer Kombination aus den zuvor genannten Anwendungsfeldern auf eingebetteten FPGA-Systemen. Im Fall mobiler Systeme, die autonom agieren können, stellen Faktoren wie die Leistungsaufnahme [116] oder die mögliche Integrationsdichte, aufgrund des Formfaktors auch in Bezug auf die thermischen Bedingungen [117], neben der notwendigen Rechenleistung Anforderungen an eine zentrale Ausführungseinheit dar.

Im Fall (semi-)autonomer Systeme wie den Marserkundungsrobotern (Mars Pathfinder, Mars Surveyor ’98 und Mars Surveyor ’01) liegen die Aufgabenfelder im Bereich der Bild- bzw. Videoverarbeitung, der Steuerung der Aktoren sowie der Regelung des Antriebs [118]. FPGAs ermöglichen, wie in [119] gezeigt wird, bereits

beim Platzieren des Designs Optimierungsschritte, um die Abwärme zu reduzieren oder zumindest gleichmäßig auf der Chipfläche zu verteilen. Da keine lokalen Hotspots auftreten, kann das eingesetzte Kühlkonzept minimal ausgelegt werden, wodurch kompaktere Roboter möglich werden. Weitere Anwendungsbeispiele für FPGAs in der Robotik und in mechatronischen Systemen sind mobile dynamisch rekonfigurierbare Plattformen zur Bildverarbeitung [284], die Bewegungssteuerung [120] oder die Berechnung der Trajektorie von Umgebungsobjekten [121].

2.2.18 Scheduling

Die zeitlich exklusive Zuordnung verschiedener Teilnehmer zu einer oder mehreren begrenzten Ressourcen lässt sich häufig in netzwerktechnischen Bereichen, wie Switch-Implementierungen finden. Diese können als diskretes Element in einem Netzwerk auftauchen oder als ein Teil eines NoC auf einem einzelnen Chip vorhanden sein. In [122] werden verschiedene Algorithmen für eine Crossbar-Verbindung eines NoC vorgestellt, die eine Round-Robin-Arbitrierung umsetzen.

Klassisches Aufgaben-Scheduling, eine Hauptaufgabe von Betriebssystemen, kann in Hardware in Form einer Instruktionssatzerweiterung oder eines exklusiven Koprozessors ausgelagert werden. Dies zeigt die Implementierung in [123].

2.2.19 Simulationen

Eine Vielzahl unterschiedlicher Anwendungsfälle zur Berechnung physikalischer Effekte, die durch ihre komplexen Berechnungsschritte oder aufgrund harter Echtzeitforderungen mit Hilfe einzelner FPGAs oder FPGA-Clustern gelöst werden, befindet sich im Bereich der Simulationen.

Simulationen mit Hilfe der Finite-Elemente-Analyse (engl.: Finite Element Methode, FEM) bilden verschiedene Bereiche nach, so zum Beispiel die lineare elastische Deformation von natürlichem Gewebe, wie sie in [124] im vorgestellten Simulator für die Virtual-Reality-Ausbildung von medizinischem Personal Verwendung findet. Mit Hilfe dieses Simulators wird das haptische Feedback berechnet und anschließend über Force-Feedback-Instrumente an den Anwender ausgegeben. Die für den Anwender benötigte realistische Wiedergabe des Verhaltens des Gewebes stellt hohe Anforderungen an die Genauigkeit und die Wiederholungsrate der Berechnung.

Zur Darstellung des thermischen und elektrischen Leistungsverhaltens einer realen Mehrprozessorumgebung bildet der in [125] vorgestellte Aufbau ein Multicore-System in einem FPGA ab. Jeder der Prozessorkerne verfügt über einen eigenen ebenfalls in Logik aufgebauten Temperatursensor, der dem Aufbau nach [126] entspricht. Über Ethernet lassen sich mehrere dieser FPGA-Multicore-Systeme untereinander verbinden, so dass sich ein großes Mehrprozessorsystem (engl.: Multi Processor System on Chip, MPSoC) emulieren lässt und das Verhalten anhand

der Abarbeitung tatsächlicher Aufgaben innerhalb eines laufenden Betriebssystems (engl.: Operating System, OS) untersucht werden kann. Der Einfluss von Spar- und Optimierungsmechanismen wie die Steuerung der Inter-Core- und Inter-FPGA-Kommunikation und das dynamische Anpassen der Taktfrequenz an die aktuelle Lastsituation kann auf diese Weise wesentlich zeitnaher beobachtet werden, als dies in einer Software-basierten Simulation möglich wäre, wie es die 35-fache Beschleunigung der Simulation in [127] zeigt.

Zur Überprüfung der Funktionalität von eingebetteten Prozessorelementen in Fahrzeugumgebungen werden, wie in [128] gezeigt, FPGA-Systeme als Hardware-In-The-Lopp-Simulatoren (HIL) verwendet, die die physikalischen Umgebungsparameter simulieren und die berechneten Stimuli für das Steuergerät darstellen. Mit diesen berechneten Daten verhält sich die virtuelle Umgebung für das Steuerelement wie in einem Fahrzeug. Für die Lösung der linearen Gleichungssysteme, die die Umweltumgebung für das HIL-System beschreiben, stellt [129] ein Rahmenprogramm für die Portierung auf FPGAs vor.

2.2.20 Anwendungen im Weltraum

Außerhalb einer geschützten Umwelt gelten spezielle Anforderungen an die Robustheit und Fehlersicherheit elektronischer Schaltungen. Datenverarbeitende Komponenten im Weltraum sind wesentlich stärker Strahlung ausgesetzt als Systeme auf der Erde, welche durch ihre Atmosphäre den größten Teil der Ionenstrahlung absorbiert. Insbesondere Speicherzellen, die die Grundlage für SRAM-basierte FPGAs darstellen, können unter Strahlungseinfluss ihren Wert verändern. Dies hat für die Schaltung zur Folge, dass beispielsweise nicht nur ein Wort im BRAM oder externen Speicher nicht mehr korrekt ist, sondern die Funktion der Schaltung eines CLBs selbst sich durch das Kippen eines Bits ändern und künftig bei jeder Berechnung fehlerhafte Ergebnisse liefern kann. Zwar gibt es spezielle geschirmte sogenannte radiation hardend FPGAs wie beispielsweise den Xilinx Virtex-5QV [130], diese liegen jedoch technologisch mindestens zwei Generationen hinter den aktuellen FPGA-Familien.

Um diesen Nachteil zu umgehen, zeigt [131] ein Verfahren auf, dessen zugrunde liegende Applikation zwar ebenfalls die Vorverarbeitung von Bilddaten darstellt, das jedoch die Möglichkeit zur partiellen Rekonfiguration einsetzt, um auf single-particle-upset-Fehler zu reagieren und somit eine selbstheilende Plattform für den Einsatz im Orbit zu schaffen. Einen ähnlichen Ansatz verfolgen [132] mit einer integrierten Plattform für die Verarbeitung von Daten an Bord eines Satelliten.

Weitere Anwendungsgebiete von FPGAs sind die Vorverarbeitung und Kompression von Daten, da die Kommunikation zwischen Satelliten bzw. Sonden zur Basisstation und der störanfällig und stark bandbreitenbeschränkt ist; siehe hierzu [133].

2.2.21 Spieltheoretische Anwendungen

Implementierungen mit dem Ziel einer Entscheidungsfindung und zur Steuerung der Interaktion verschiedener kooperativer oder nichtkooperativer Parteien finden ebenfalls Beachtung in der untersuchten Literatur. Mögliche Ziele sind, wie in Abschnitt 2.2.5 aufgezeigt, Prognosen über das Verhalten von Käufern und somit über die Marktentwicklung eines Finanzprodukts, aber auch die Simulation von Spielabläufen. Eine FPGA-basierte künstliche Intelligenz (KI) als Gegner für das Brettspiel Go, basierend auf dem Monte-Carlo-Go-Algorithmus, wird in [134] vorgestellt. Im Vergleich zu Schach verfügt das Spiel Go über einen weit größeren Suchraum und eine sich schneller verändernde Spielsituation, weshalb eine parallele Beschleunigung der Simulation notwendig wird. Ein Beispiel für verschiedene Umsetzungen zur Lösung von Sudoku-Zahlenrätseln wird in [135] gegeben. Die Behandlung als nichtkooperative Interaktion von mehreren Teilnehmern während der Zuteilung von Trägerfrequenzen bei der OFDM-Übertragung ermöglicht eine Reduktion der benötigten Sendeenergie pro Bit [136].

2.2.22 FPGAs als Testmustergeneratoren

Das Anwendungsfeld der Testmustererzeugung unterscheidet sich vom Anwendungsfeld Simulation aus Abschnitt 2.2.19 hauptsächlich in der Art der zu erzeugenden Testmuster. Im Gegensatz zur Umweltsimulation dienen die auszugebenden Stimuli der rein funktionellen Überprüfung von nichtintelligenten Systemen wie beispielsweise Speicherbausteinen [137]. Um die Funktion eines Prozessorentwurfs im Zusammenspiel mit anderen Komponenten, also auf der Ebene des Systementwurfs, zu überprüfen, wird in [138] ein Testsystem vorgestellt, welches einen Multimediaprozessor im FPGA abbildet und Peripheriebausteine wie einen Baustein für die PCI-Schnittstelle sowie verschiedene Speicher enthält.

2.3 Anwendungsgebiete von FPGA-Clustern

Die Anwendungsgebiete von FPGA-Clustern unterscheiden sich in vielen Bereichen nur in ihrer Skalierung von den Anwendungsgebieten einzelner FPGAs. So sind viele der in Kapitel 4.2 vorgestellten Systeme mit konkreten Anwendungsfeldern im Hintergrund entstanden, die nicht vollständig oder aufgrund der benötigten Bausteingröße nicht wirtschaftlich auf einem einzelnen FPGA realisiert werden können. Selbst Mechanismen, die die Fähigkeit zur Rekonfiguration zur Laufzeit haben, um die Logik-Ressourcen im Zeitmultiplex-Verfahren für verschiedene Aufgaben innerhalb der Ausführung eines Algorithmus zu nutzen, unterliegen zum einen der Einschränkung, dass nicht alle Anwendungen einen solchen mit Unterbrechungen verbundenen Ansatz unterstützen, zum anderen muss die Zeit für

die Rekonfiguration in einem geringen Verhältnis zur Ausführungszeit des gerade aktuellen Abschnitts stehen. In Anlehnung an die immer kleiner werdenden Strukturen und damit verbundenen höheren Packungsdichten im ASIC-Bereich wird das Verfahren als (Logik-)Dichtensteigerung (engl.: density enhancement) bezeichnet [139]. Formel 2.1 nach [140] beschreibt die Mindestanzahl der Verwendungen einer zu rekonfigurierenden Operation W, ab der bei einer Rekonfigurationszeit T_R und einer Ausführungszeit N der logischen Operation – jeweils in Takten – eine beschleunigte Ausführung ermöglicht wird.

$$W = \frac{T_R}{N - 1} \tag{2.1}$$

Aufgrund der Bitstream-Größe und der Rekonfigurationsgeschwindigkeit bedeutet jede (partielle) Rekonfiguration des Bausteins einen zeitlichen Aufwand, welcher die Effizienz, mit der Daten verarbeitet werden können, einschränkt. Beispielsweise ist der Bitstream für einen modernen Baustein der Virtex-7-Generation (XC7K480T) bereits 149 880 032 bit (17,86 MiB) groß, so dass nicht mehr nur die Verarbeitungsschnittstelle des Konfigurations-Controllers - im Allgemeinen ein als Hard- oder Softmacro eingebundener eingebetteter Prozessor - einen Einfluss auf die Dauer des Vorgangs hat, sondern auch die verschiedenen am Rekonfigurationsvorgang beteiligten externen und internen Speicher signifikant die Performanz beeinflussen. Werden die Konfigurationsdaten extern von einem Hostsystem zugeführt, so addiert sich der zeitliche Aufwand der Kommunikation ebenfalls zur Laufzeit. Dies geschieht beispielsweise über den PCI-Bus, an welchen der FPGA direkt oder über Kommunikationsbrücken angeschlossen ist.

Zur Beurteilung der zeitlichen Kosten für eine partielle Rekonfiguration hat [141] eine Bewertungsmetrik aufgestellt, welche als Online-Ressource unter [142] genutzt werden kann. Abgefragt werden die Parameter des Speichers mit den Bitstream-Daten sowie der FPGA-Baustein und der darauf implementierte Prozessor, der die Rekonfiguration steuert. Das berechnete Ergebnis in ms kann in Bezug auf die Frage herangezogen werden, ob eine partielle Rekonfiguration für das Anwendungsszenario Vorteile bringt.

Die Anzahl an Veröffentlichungen zu Implementierungen auf FPGA-Clustern ist im Vergleich zu den Anwendungsgebieten von FPGAs relativ gering, obgleich erste FPGA-Cluster bereits 1985 als CCM aufgebaut wurden. In aktuellen Veröffentlichungen bildet vielfach der jeweilige FPGA-Cluster den Schwerpunkt, während die Betrachtung des Anwendungsfeldes in den Hintergrund rückt. Aus diesem Grund wird auf eine gesonderte grafische Aufstellung der Veröffentlichungen im Rahmen dieser Arbeit verzichtet.

Das Hauptaugenmerk bei Cluster-Systemen liegt auf der Kommunikationsinfrastruktur, weshalb die durch die jeweilige Anwendung gestellten Anforderungen bereits an dieser Stelle herausgearbeitet werden. Eine detaillierte Betrachtung und eine im Rahmen dieser Arbeit getroffene Definition eines FPGA-Clusters erfolgt

im nachstehenden Kapitel 4. Die in den folgenden Abschnitten eingesetzten FPGA-Cluster werden im weiteren Verlauf der Arbeit in Kapitel 4.2 vorgestellt.

2.3.1 Bildverarbeitung

Das in [143] vorgestellte Verfahren aus dem Bereich Bildverarbeitung beschreibt den Einsatz eines FPGA-Clusters zur beschleunigten Suche eines Bildes anhand von extrahierten Merkmalen in einer Datenbank mit 150 Millionen Einträgen. Die Umsetzung des eigentlichen Speeded-Up-Robust-Features-Algorithmus (SURF) zur Bestimmung der Feature-Punkte des gesuchten Bildes erfolgt dabei auf einem Host-Server, der die Informationen des Bildes an die einzelnen FPGA-Knoten via 10G-Ethernet überträgt. Den FPGAs steht ein Teil der Datenbank im lokalen Speicher zur Verfügung, auf dem sie die Distanzberechnung durchführen und das Ergebnis an den Host zurückmelden. Eine Kommunikation der FPGAs untereinander findet nicht statt. Aus diesem Grund sowie wegen des eingesetzten Kommunikationsmediums Ethernet lässt sich das System nahezu beliebig skalieren.

2.3.2 Bioinformatik

Wie bereits in [42] gezeigt, profitiert eine Portierung des heuristischen BLAST-Algorithmus von der gebotenen Parallelität. Die ebenfalls aufgezeigte Grenze wird durch die limitierte Kommunikationsinfrastruktur zum einen zwischen den einzelnen CCMs und zum anderen zum Hostsystem gebildet. Neben einer möglichst breitbandigen, niederlatenten Kopplung einzelner FPGAs wird die Flexibilität der Kommunikationsinfrastruktur als wichtig hervorgehoben.

Um die Suche nach übereinstimmenden Parametern in einer großen Datenbank mit Genomdaten wie DNA- oder Proteinsequenzen zu beschleunigen, wird in [144] eine parallele Suche auf unabhängigen Teilbereichen der Datenbank implementiert. Der auf 48 FPGA-Knoten aufgeteilte Suchraum kann parallel analysiert werden; entsprechend ist ein Datenaustausch über Ethernet ausreichend, um die Resultate der Suche an den Host zu übermitteln. Eine möglichst niederlatente Anbindung an die Datenbankausschnitte ist dabei wesentlich relevanter für die Ausführungszeit der Suche als eine breitbandige Kommunikationsmöglichkeit zwischen den FPGA-Knoten. Aus diesem Grund wird der in Abschnitt 2.2.3 vorgestellte Ansatz verfolgt, bei dem der FPGA direkt am Massenspeicher positioniert wird. Eine Kommunikation zwischen den einzelnen FPGAs findet nicht statt.

2.3.3 Datenerfassung

Im Bereich der Radioteleskopie wird in [145] ein FPGA-Cluster für die Datenerfassung und Vorverarbeitung beschrieben, der aus verschiedenen industriefertigen

FPGA-Komponenten (engl.: Commercial Off The Shelf, COTS) besteht. Der aus 16 Xilinx ML310 Evaluation Boards bestehende FPGA-Cluster hat die Aufgabe, die von den einzelnen Eight-meter-wavelength-Transient-Array-Sensorknoten (ETA) übermittelten Daten miteinander zu kombinieren und für die Speicherung im Hostsystem vorzubereiten. Aufgrund der Aufteilung in zwölf äußere und vier innere Knoten findet die Kommunikation ausschließlich in Richtung der inneren Knoten und des daran angeschlossenen Serversystems statt. Das Beamforming der aufgenommenen Signale erfolgt auf den äußeren FPGAs, während die inneren Knoten weitergeleitete Daten synchronisieren und anschließend untereinander kombinieren.

Für die automatische Transkription von Sprachdaten in Echtzeit stellt [146] eine Multi-FPGA-Implementierung von Hidden-Markov-Modellen (HMM) auf Basis des Berkeley-Emulation-Engine2-FPGA-Clusters (BEE) vor. Mit dem Ziel der Minimierung der Inter-FPGA-Kommunikation ist ein Master-Slave-Ansatz entstanden, welcher zum einen den Suchraum anhand der Anfangsbuchstaben der Wörter aufteilt und die entsprechenden HMMs lokal pflegt. Zum anderen werden durch den Master-FPGA die Ablaufsteuerung der Berechnungen und die Zuordnung der Wörter entsprechend des Anfangsbuchstabens vorgegeben.

2.3.4 Data-Mining

Eine Implementierung des CART-Algorithmus auf einem 4-FPGA-Cluster wird in [147] gezeigt. Der Datenaustausch zwischen den einzelnen FPGAs erfolgt aufgrund der Struktur des Convey-HC-1-Systems über gemeinsam adressierbaren Speicher. Da ein breitbandiger Zugriff auf die zu klassifizierenden Daten notwendig ist – die Kommunikation mit den benachbarten Knoten ist hingegen weniger bedeutsam – bietet die speicherbasierte Kommunikationsinfrastruktur Vorteile für die Bearbeitung von datenintensiven Anwendungen. Ein Austausch mit dem im System befindlichen Server-Prozessor ist über die Speicherkanäle entsprechend ebenfalls breitbandig möglich.

2.3.5 Finanzwirtschaftliche Anwendungen

Das bereits in Abschnitt 2.2.5 aufgeführte Anwendungsfeld der Initialpreisfindung im Bereich gehandelter Derivatprodukte findet auch in Veröffentlichungen zum Thema FPGA-Cluster Erwähnung. In [148] wird die Implementierung einer Quasi-Monte-Carlo-Simulation beschrieben. Als zugrunde liegende Zufallszahlenquelle dient dabei nicht eine Schieberegister-basierte pseudozufällige Sequenz (engl.: Pseudorandom Binary Sequence, PRBS), sondern ein Quasi-Zufallszahlen-Generator. Die Zielplattform der Umsetzung ist der Maxwell-FPGA-Cluster, welcher in Kapitel 4.2.2 vorgestellt wird. Da die parallel laufenden Berechnungen der einzelnen Simulationspfade keinen Datenaustausch untereinander benötigen, skaliert

die Anwendung ausschließlich über die Anzahl der FPGAs und nicht anhand der Kommunikationsinfrastruktur.

Die in [149] aufgezeigte Umsetzung von Monte-Carlo-Simulationen setzt das Message-Passing-Interface-Protokoll (MPI) ein, um Daten zwischen den einzelnen Knoten auszutauschen, die ihrerseits ein lokales Linux auf dem eingebetteten Prozessorkern implementieren und mit dedizierten Koprozessoren für die Simulationsbeschleunigung erweitert wurden. Entsprechend wird die vorhandene Ethernet-Verbindung als nichtausreichend beschrieben und die Kommunikation über die MGT-Verbindungen abgehandelt.

2.3.6 Kryptologie

FPGA-Cluster finden, wie beispielsweise in [67] beschrieben, im Bereich des Verschlüsselungsbrechens Anwendung, wenn es gilt, parallel einen Wörterbuchangriff durchzuführen, bei dem die einzelnen FPGAs jeweils eine Untermenge des Symbolraums auf die verschlüsselten Daten anwenden. Da die einzelnen Aufgaben lokal ablaufen und nur von der Partitionierung der Suchschlüssel auf die einzelnen Knoten abhängen, welche sich nicht untereinander austauschen müssen, lässt sich das System nahezu beliebig skalieren und durch die damit steigende Parallelität weiter beschleunigen.

2.3.7 Medizintechnik

Gleichermaßen wie bereits in Abschnitt 2.2.9 zum Einsatz einzelner FPGAs im medizinischen Aufgabenbereich beschrieben, wird in [150] ein Schnittbereich zur Bildverarbeitung berührt. Zum Vergleich von in zeitlichem Abstand gewonnenen Untersuchungsaufnahmen müssen die Bilder möglichst deckungsgleich übereinander gelegt werden, um Abweichungen wie etwa Tumorwachstum detektieren zu können. Der vorgestellte Fluid-Registration-Algorithmus wurde auf den FPGAs des Convey HC-1 Hybrid Computer implementiert. Aufgrund der datenintensiven Berechnungen ist eine breitbandige Anbindung an den Systemspeicher notwendig. Der Datenaustausch unter den FPGAs erfolgt über geteilte Speicherbereiche.

2.3.8 Neurocomputing

Um die in Abschnitt 2.2.12 erwähnte Obergrenze der Anzahl künstlicher Neuronen zu erhöhen, können, wie in [279] aufgeführt, verschiedene parallele Verarbeitungsansätze genutzt werden. Je nach Ansatz unterscheiden sich die Anforderung an die Kommunikationsinfrastruktur zwischen den FPGAs. Gemein ist den vorgestellten Varianten jedoch, dass ein Datenaustausch zwischen den Neuronen und Komponenten innerhalb des Clusters notwendig ist, damit die parallel berechneten Ergebnisse

der einzelnen Prozessorelemente ausgewertet werden und die Adaption des Netzes umsetzbar ist.

Im Rahmen des Catapult-Forschungsprojekts der Firma Microsoft werden Convolutional-Neural-Network-Implementierungen auf dem Catapult-FPGA-Cluster eingesetzt, um Bilddaten auszuwerten und Suchen innerhalb von Datenzentren zu beschleunigen. Unter anderem sollen auf diese Weise Suchanfragen über die Internetsuche *Bing* verbessert werden [95].

2.3.9 Physikalische Simulationen und Datenerfassung physikalischer Vorgänge

Im Bereich der Quantum-Monte-Carlo-Simulationen zur Beobachtung der Energie und der Wellenfunktion von Atomen innerhalb einer inerten Gaswolke beschreibt [151] ein Rahmenprogramm für den Einsatz auf FPGA-Clustern und rekonfigurierbaren Hochleistungsrechensystemen (engl.: High Performance Reconfigurable Computing, HPRC). Skalierbare Anwendungen aus dem Bereich der Strömungsmechanik lassen, wie in [152] gezeigt, detailliertere Beobachtungen zu, jedoch limitiert die Speicherbandbreite die Ausführungszeit. Aufgrund des für FPGA-Cluster üblichen verteilten Speichers, welcher breitbandig an die einzelnen FPGAs angebunden ist, lässt sich dieser Faktor minimieren. Durch das lokale Vorhalten der für die Berechnung erforderlichen Daten wird die Notwendigkeit für einen unter Umständen über mehrere verschiedene Busse laufenden Zugriff auf einen zentralen Speicher, auf den im Fall eines Clusters mehrere Knoten zugreifen müssen, gering gehalten.

Bei der experimentellen Untersuchung subatomarer Partikel wie beispielsweise im HADES-Spektrometer [153] werden Systeme für die Datenerfassung benötigt, die in der Lage sind, aus den kontinuierlich anfallenden riesigen Datenmenge der einzelnen Messpunkte die relevanten Merkmale zu extrahieren und die Informationen zur Speicherung aufzubereiten. Der in [154] vorgestellte FPGA-Cluster bietet aufgrund seiner vollvermaschten Kommunikationsinfrastruktur die Möglichkeit, die auf den einzelnen Computing-Nodes (CN) parallel eintreffenden Signale nicht nur lokal, sondern auch untereinander in Relation zu setzen und so die wichtigen Daten aufzuzeichnen und an Massenspeicher oder PC-Systeme weiterzureichen.

2.3.10 Simulationen

Das bereits in Kapitel 2.2.19 vorgestellte Anwendungsfeld der Simulation des Verhaltens von organischem Gewebe unter Krafteinwirkung zur Wiedergabe auf haptischen Eingabeinstrumenten ist auch das dedizierte Ziel des in [155] vorgestellten FPGA-Clusters, da die Verarbeitungsleistung eines einzelnen FPGAs nicht ausreichend ist, um in Echtzeit die Simulation von nichtlinearen Deformationsmo-

dellen mit einer zufriedenstellenden Genauigkeit zu betreiben. Die zeitkritische Matrixmultiplikation lässt sich durch das Zerlegen in Submatritzen auf die einzelnen Knoten des Systems verteilen und in mehreren Stufen parallel berechnen. Ergebnisvektoren werden in der ringförmig angeordneten Topologie mit jedem Berechnungsschritt ausgetauscht. Durch eine optimierte Anordnung der Matrixelemente lässt sich der Kommunikationsaufwand auf einzelne Vektoren begrenzen. Am Ende kann das Ergebnis über den ersten Knoten ausgelesen werden. Die Skalierbarkeit wird durch die Bandbreite der Kommunikationsinfrastruktur begrenzt, da der Kommunikationsaufwand linear mit der Anzahl der eingesetzten FPGAs steigt. Die Verarbeitungsgeschwindigkeit des FPGA-Clusters sinkt dementsprechend aufgrund der zur Berechnungszeit hinzukommenden Kommunikationszeit im Vergleich zu einer Implementierung auf einem einzelnen großen FPGA mit vergleichbaren Logikressourcen, sofern dieser technologisch und wirtschaftlich verfügbar ist.

Die Echtzeitdarstellung des Einflusses von elektromagnetischen Transienten aufgrund von Lastwechseln innerhalb eines breit angelegten Stromnetzes mit vielen unterschiedlichen Lastteilnehmern erlaubt den Einsatz optimierter Schutzschaltungen. Zur Simulation des Verhaltens wird in [156] eine skalierbare Abbildung der verschiedenen Netzteilnehmer wie Generatoren, Transformatoren, Lasten und Modelle der Übertragungsleitungen des Stromnetzes auf einem FPGA-Cluster beschrieben. Die für drei und für zehn FPGAs umgesetzte Implementierung erfordert trotz der Platzierung von stark kommunikationsabhängigen Komponenten auf benachbarten FPGAs einen Austausch von Informationen zwischen den meisten Elementen. Dieser Datenverkehr erfolgt über benachbarte FPGAs in zum Teil mehreren Schritten, da die vorhandene systemweite Busschnittstelle die benötigte Datenrate nicht zur Verfügung stellt.

2.3.11 Prototyping mikroelektronischer Schaltungen

Systementwürfe, die sich aufgrund ihrer Größe nicht auf einem einzelnen FPGA (wirtschaftlich) abbilden lassen, können auf einem FPGA-Cluster partitioniert untersucht werden, wie dies in [104] für einen VLIW-Prozessor dargestellt wird. Zur Demonstration der Funktionalität ist es möglich, für den Prozessor entworfene Programme auf dem im FPGA-Cluster abgebildeten Prozessor auszuführen, wodurch eine Fehlerbereinigung im Hardwareentwurf parallel zur Bereinigung im Softwareentwurf erfolgen kann. Die Leistungsfähigkeit der zukünftigen ASIC-Implementierung kann somit auf funktionaler Ebene sichergestellt werden.

Für die auf mehrere FPGAs verteilte prototypische Implementierung von SoC-Designs stellt die physikalisch begrenzte Anzahl an Inter-FPGA-Verbindungen eine große Einschränkung bezüglich der funktionalen Verifikation dar, da im SoC vorhandene Verbindungen nicht auf der eigentlichen FPGA-Cluster-Hardware abgebildet werden können, weil sie entweder in zu geringer Anzahl oder gar nicht

vorhanden sind. Wie in [157] aufgezeigt, kann es deshalb notwendig sein, die Kommunikationsinfrastruktur im Zeitmultiplex-Verfahren geteilt zu verwenden.

2.3.12 Prozessorentwurf

Der Datenaustausch von auf MPSoCs ausgeführten C-Programmen erfolgt häufig über das MPI-Protokoll, welches eine Abstraktion der zugrunde liegenden Kommunikationshardware darstellt. Ein in [158] vorgestelltes Rahmenprogramm implementiert Teile des MPI-Standards, so dass sich sowohl die Intra- als auch die Inter-FPGA-Kommunikation für das laufende Programm transparent vollzieht und sich das über mehrere FPGAs erstreckende NoC wie ein geschlossen zusammenhängendes Netz darstellt. Dadurch können die Prozessoren entsprechend ihrer Größe auf die einzelnen FPGAs verteilt werden und dennoch als MPSoC agieren.

Wie bereits im vorherigen Abschnitt 2.3.11 aufgezeigt, wird der in [104] entwickelte CoreVA-VLIW-Prozessor ebenfalls auf einem FPGA-Cluster abgebildet. Auf diese Weise ist es möglich neben der prototypischen Überprüfung des Prozessorkerns eine Entwurfsraumexploration von dedizierten Hardware-Beschleunigern durchzuführen.

Für die Verifikation des Entwurfs des Bluegene/Q-SoC-Vielkern-Prozessors wurde ein eigens für diesen Zweck entworfener FPGA-Cluster mit 28 FPGAs eingesetzt. Die Kommunikationsinfrastruktur über eine aktive Backplane erlaubt es, die Modulsteckplätze für FPGAs und Speicherboards in verschiedenen Topologien miteinander zu verschalten. Neben Sterntopologien mit dedizierten Routing-FPGAs auf der Backplane sind Intra- und Inter-Quadrant-Leitungen vorgesehen. Ein Quadrant umfasst sieben Module. Zur flexiblen Ergänzung der Backplane ist es möglich, kabelgebundene Punkt-zu-Punkt-Verbindungen zwischen den einzelnen FPGA-Knoten zu spannen. Durch diesen flexiblen Aufbau kann den Anforderungen der SoC-Architektur entsprochen werden und eine taktsynchrone Simulation des mit einer Chipfläche von 360 mm^2 in 45-nm-Technologie gefertigten umfangreichen Vielkern-Prozessors umgesetzt werden [159].

2.4 Zusammenfassung

Die Anforderungen an die eingesetzten FPGAs sind trotz der vielfältigen Anwendungsgebiete relativ ähnlich. Der Hauptgrund für einen Wechsel von Prozessor- oder Mikrocontrollerarchitekturen zu FPGAs liegt darin, dass die Anwendung beschleunigt werden muss. Diese Beschleunigung ist zum einen aufgrund harter Echtzeitanforderungen wie beispielsweise in der Regelungstechnik erforderlich, zum anderen aufgrund zu langer Rechenzeiten wie etwa bei Simulationen.

Echtzeitanforderungen sind hauptsächlich im Bereich der Anwendungen zu finden, die auf einem einzelnen FPGA implementiert sind, während rechenintensive

Aufgaben immer mehr auf FPGA-Clustern umgesetzt werden und die gebotene massive Parallelität ausnutzen. Vielfach werden mit hohem Aufwand Mechanismen umgesetzt, um den Kommunikationsaufwand zwischen den einzelnen FPGAs des Clusters so gering wie möglich zu gestalten, da die Kommunikationsinfrastruktur des Systems den Anforderungen an die Flexibilität und Latenz nicht genügt. Gerade diese Unzulänglichkeiten erschweren die Portierung großer Anwendungen mit harten Echtzeitanforderungen auf FPGA-Cluster.

Darüber hinaus konnte in verschiedenen Anwendungsfeldern die Energieeffizienz als ein Grund für die Nutzung von FPGAs identifiziert werden. Aus zwei verschiedenen Gründen ist eine Portierung sinnvoll. Zum einen muss insbesondere im Bereich der Robotik mit einem sehr eng begrenzten Energiebudget das vollständige System mit Sensoren und Aktoren betrieben werden können. Die effiziente Implementierung kontinuierlicher Berechnungen beispielsweise für die Navigation darf somit nur einen geringen Anteil am Gesamtverbrauch haben. Ebenfalls müssen harte Echtzeitschranken bei Berechnungen eingehalten werden. Zum anderen können in großen Rechenzentren durch Senkung des Energieverbrauchs die Betriebskosten optimiert und die Kühlung der Systeme kleiner ausgelegt werden.

Wie gezeigt wurde, entscheidet insbesondere die Kommunikationsinfrastruktur über die Einsatzmöglichkeiten des Systems in verschiedenen Anwendungsbereichen. Zur Beschreibung der unterschiedlichen Kommunikationswege wird im nachfolgenden Kapitel eine Modellierung erarbeitet, mit deren Hilfe eine Beschreibung der lokalen und globalen Verbindungen möglich ist. Ziel dieser Modellierung ist eine Optimierung der Topologie, um die aufgrund technischer Grenzen in der Anzahl der möglichen Verbindungen eingeschränkte Kommunikationsinfrastruktur effizient zu realisieren.

3 Modellierung der Cluster-Kommunikationsinfrastruktur

Um die Kommunikationsstrukturen der FPGA-Cluster beschreiben zu können, ist es erforderlich, eine Nomenklatur zu erarbeiten, die einerseits die für die einzelnen FPGAs möglichen Verbindungsvarianten beschreibt und zum anderen die Topologie des Clusters darstellt. Diese beiden Darstellungsformen können vereinfacht als Fein- und Grobstruktur beschrieben werden. Während durch die Bestimmung der Verbindungsvarianten feingranular aufgeschlüsselt werden kann, welche Schnittstellen miteinander kommunizieren können, wird durch die Topologiebeschreibung die grobe Struktur des Clusters dargestellt. Beiden Angaben ist gemein, dass sie als Maß für die Flexibilität der Kommunikationsinfrastruktur fungieren.

Optimierungen, wie sie aus der Netzwerktechnik bekannt sind, legen ein bereits vorhandenes Netz zugrunde, in dem Daten zwischen verschiedenen Knoten transferiert werden müssen. Routing-Algorithmen bestimmen dabei dynamisch mögliche Pfade für die Verbindung. Das eigentliche Netz umfasst eine große Anzahl an Knoten, beispielsweise mehrere Millionen für das World Wide Web, und muss darüber hinaus flexibel auf eine Änderung sowohl der Knotenmenge V als auch der Kantenmenge E reagieren. Beschreibungen dieser komplexen Netzwerke, wie sie beispielsweise in [160] gegeben werden, sehen keine Optimierung der vollständigen Topologie vor, da dies selbst rein physikalisch nicht zu realisieren wäre. Im Rahmen dieser Arbeit wird jedoch ein optimierter Aufbau einer Netzwerktopologie mit einer vorher festgelegten Anzahl an Knoten und Verbindungsmöglichkeiten gesucht. Zwar untersuchen die Autoren aus [161] anhand eines gegebenen Suchalgorithmus, der auf eine fest begrenzte Suchtiefe ausgelegt ist, wie ein optimales (lokales) Netz aufgebaut sein muss. Das Resultat, dass Graphen mit Sterntopologien optimal für eine geringe Anzahl an parallelen Suchvorgängen sind, lässt sich jedoch nicht zwangsläufig auf realisierbare Cluster-Topologien übertragen. Die im Fachgebiet *Schaltungstechnik* der Universität Paderborn entstandene Modellierung leitungsgebundener Kommunikationsstrukturen aus [162] beschreibt die notwendige Energie für die Übertragung einzelner Bits unterschiedlicher Übertragungsstandards. In der hier vorliegenden Arbeit werden hingegen die darstellbaren Topologien und ihr Einfluss auf die Übertragungszeiten untersucht.

Zunächst wird in Abschnitt 3.2 die Feinstruktur der Verbindungsvarianten beleuchtet, was die Definitionen grundlegender Begriffe notwendig macht. Im weiteren Verlauf des Kapitels erfolgt unter 3.3 die Erfassung der Topologieeigenschaften

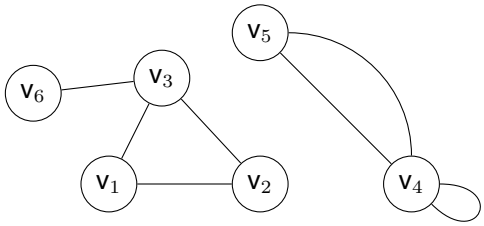


Abbildung 3.1: Nichtzusammenhängender Graph mit einer Schleife und parallelen Kanten

eines FPGA-Clusters mit Hilfe von Graphen. Im Anschluss daran wird die Optimierung der möglichen Cluster-Topologien mit dem Ziel einer minimalen Latenz für den Datenaustausch in Abschnitt 3.4 aufgezeigt und anhand eines Beispiels erläutert. Zum Vergleich werden schließlich unter 3.4.1 typische regelmäßige Strukturen aufgezeigt und mit Hilfe der vorausgegangenen Modellierung bewertet. Eine kurze Zusammenfassung schließt das Kapitel ab.

3.1 Grundlegende Begriffe der Graphentheorie

Die in den folgenden Abschnitten zusammengeführten Grundbegriffe für ungerichtete Graphen entstammen [163] und [164], sofern nicht anders ausgewiesen.

Ein *ungerichteter* Graph $G = \{V, E\}$ besteht aus der Menge der *Knoten* V (engl.: Vertices) und der Menge der *Kanten* E (engl.: Edges). Kanten werden als Tupel der zwei Knoten dargestellt, die sie verbinden, beispielsweise $e_1 = \{v1, v2\}$. Eine Kante $e \in E$, die *inzident* zu einem Knoten $v \in V$ ist, erfüllt die Bedingung $v \in e$. Genau zwei Knoten werden jeweils durch eine Kante miteinander verbunden, wie in Abbildung 3.1 für die Knoten $V = \{v_1, v_2, v_3, v_4, v_5\}$ und Kanten $E = \{ \{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_1\}, \{v_3, v_6\}, \{v_4, v_4\}, \{v_4, v_5\}, \{v_4, v_5\} \}$ gezeigt wird. Verbindungen eines Knotens auf sich selbst $(e = \{v_4, v_4\})$, sogenannte *Schleifen*, sind dabei ebenso möglich wie mehrere Parallelverbindungen zwischen zwei Knoten. Handelt es sich um einen *gerichteten* Graph, so sind die Beschreibungen der Kanten geordnet.

Zwei Knoten sind *adjazent*, wenn gilt $\{v_x, v_y\} \in E, v_x \neq v_y$. Ist jeder Knoten im Graphen mit jedem anderen Knoten adjazent, so ist der Graph *vollständig* und besitzt $|E| = \binom{|V|}{2}$ ungerichtete Kanten. Der Adjazenzbegriff bei gerichteten Graphen gilt für jede Kante immer nur in eine Richtung entsprechend ihrer geordneten Darstellung der Kanten. Die *Valenz* oder der *Grad* $d(v)$ eines Knoten $v \in V$ entspricht der Anzahl der mit ihm inzidenten Kanten $|E(v)|$ und somit der Anzahl der

adjazenten Knoten. So beträgt der Grad der Knoten v_1, v_2 und v_5 aus Beispiel 3.1 $d(v_1) = d(v_2) = d(v_5) = 2$, während die Valenz für die beiden Knoten v_3, v_4 $d(v_3) = d(v_4) = 3$ und für den verbleibenden Knoten $d(v_6) = 1$ ist. Ist die Valenz für alle Knoten im Graphen gleich, wird der Graph als *k-regulär* bezeichnet.

Ein *Pfad* p innerhalb eines Graphen beschreibt eine Abfolge von Kanten, die einen zusammenhängenden Weg adjazenter Knoten von einem Knoten zu einem anderen Knoten im Graph bilden. Wie an dem Beispiel aus Abbildung 3.1 zu erkennen ist, existieren zwei Pfade mit unterschiedlicher Länge von v_1 zu v_3 – zum einen der Pfad $p_1 = \{\{v_1, v_3\}\}$ mit der Länge 1 und zum anderen der Pfad $p_2 = \{\{v_1, v_2\}, \{v2, v3\}\}$ der Länge 2 über den Knoten v_2. Ein Graph, in dem von jedem Knoten ausgehend zu jedem anderen Knoten ein Pfad gefunden werden kann ($p(v_x, v_y) \neq \infty, \forall v_x, v_y \in V$ mit $v_x \neq v_y$), wird als *zusammenhängend* bezeichnet. Das Beispiel aus Abbildung 3.1 ist somit nichtzusammenhängend. Besitzen die Kanten des Graphen keine gesonderten Kantengewichte ($c(e) = 1, \forall e \in E$), so entspricht die Länge eines Pfades genau der Anzahl der enthaltenen Kanten; folglich wird die Pfadlänge zu einem *isolierten* Knoten als unendlich angegeben. Der kürzeste Pfad $p(v_x, v_y)$ zwischen zwei Knoten wird als *Abstand* bezeichnet. Der *Durchmesser* $D(G)$ eines zusammenhängenden Graphen entspricht der Anzahl der Kanten des längsten der kürzesten Pfade zwischen allen enthaltenen Knoten.

Zwei Graphen G und G' werden als *isomorph* zueinander bezeichnet, wenn eine bijektive Abbildung $\delta : V \rightarrow V'$ existiert, für die gilt $\{v_x, v_y\} \in E \Leftrightarrow \{\delta(v_x), \delta(v_y)\} \in E', \forall v_x, v_y \in V$. Ein Beispiel für zwei isomorphe Graphen wird in Abbildung 3.2 gegeben.

Darstellungsmöglichkeiten von Graphen

Neben der grafischen Darstellung sind verschiedene textuelle Beschreibungsformen für Graphen üblich. Im Allgemeinen erfolgt die Darstellung über die Adjazenzbeziehungen der Knoten. Die einfachste Form ist die Adjazenzmatrix, welche die Anzahl vorhandener Kanten adjazenter Knoten als einen Eintrag aufweist. Die Adjazenzmatrix für das Beispiel aus Abbildung 3.1 gestaltet sich wie folgt:

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{array} \begin{array}{c} \begin{array}{cccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{3.1}$$

Eine Auflistung der Nachbarschaftsbeziehungen der Knoten wird als Adjazenzliste bezeichnet und erfasst nur die tatsächlich vorhandenen Verbindungen zwischen den Knoten.

$$\begin{array}{ll} v_1: & v_2, v_3 \\ v_2: & v_1, v_3 \\ v_3: & v_1, v_2 \\ v_4: & v_4, v_5, v_5 \\ v_5: & v_4, v_4 \end{array} \tag{3.2}$$

Eine weitere, im Rahmen dieser Arbeit verwendete Darstellungsform erfasst die Distanz, die jeder Knoten zu jedem anderen Knoten im Graphen besitzt. Entsprechend wird diese Art der Darstellung als Distanzmatrix bezeichnet.

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{array} \begin{array}{c} \begin{array}{cccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & \infty & \infty & 2 \\ 1 & 0 & 1 & \infty & \infty & 2 \\ 1 & 1 & 0 & \infty & \infty & 1 \\ \infty & \infty & \infty & 0 & 1 & \infty \\ \infty & \infty & \infty & 1 & 0 & \infty \\ 2 & 2 & 1 & \infty & \infty & 0 \end{pmatrix} \end{array} \tag{3.3}$$

Im weiteren Verlauf des Kapitels werden die hier aufgeführten Grundbegriffe zur Beschreibung von Clusterstrukturen verwendet und um spezifische für die Darstellung der Besonderheiten von Kommunikationsinfrastruktur notwendige Begriffe erweitert.

3.2 Allgemeine Bestimmung der Verbindungsvarianten

Zur allgemeinen Bestimmung der darstellbaren Verbindungsvarianten für gegebene Systeme muss zunächst erfasst werden, was eine Verbindungsvariante einzigartig und somit als Element der Menge aller Varianten quantifizierbar macht. Zur Einordnung von Verbindungsvarianten, die über die leere Menge $\emptyset$ hinausgehen, bedarf es der Festlegung, in welcher Form Teile eines Systems Daten austauschen können. Definition 1 beschreibt hierfür die *Schnittstelle*, die benötigt wird, um Verbindungen zwischen einzelnen Komponenten herzustellen.

Definition 1 *Eine Schnittstelle $s \in S$ stellt eine unidirektionale Verbindungsmöglichkeit dar.*

Gemäß Definition 1 existieren eingehende und ausgehende Schnittstellen. Der Datenverkehr erfolgt je Schnittstelle in eine der beiden möglichen Richtungen.

Definition 2 *Ein Modul $m \in M$ verfügt über $s_1, s_2..s_n \in S, n \in \mathbb{N}$ Schnittstellen.*

Definition 2 besagt nicht, dass alle n Schnittstellen tatsächlich genutzt werden; offene Verbindungen sind ebenfalls möglich. Eine gleichmäßige Verteilung der Schnittstellen auf die Module kann, muss jedoch nicht gegeben sein. Module mit einer unterschiedlichen Anzahl an Schnittstellen können in beliebigen Kombinationen innerhalb eines Systems verfügbar sein.

Um einen Datenaustausch zwischen zwei Komponenten zu ermöglichen, müssen auf einer Seite mindestens eine ausgehende Schnittstelle und auf der anderen gleich viele oder mehr eingehende Schnittstellen vorhanden sein. Für die meisten Anwendungen ist eine Bündelung von zwei Schnittstellen s_I und s_O sinnvoll. Die Definition eines *Ports*, welcher aus einer ausgehenden und einer eingehenden Schnittstelle besteht und somit einen Hin- und Rückkanal ermöglicht, vereinfacht die Formulierung weiterer Bedingungen, siehe Definition 3.

Definition 3 *Ein Port $\phi \in \Phi$ besteht aus einer eingehenden Schnittstelle s_I und einer ausgehenden Schnittstelle s_O. Ein Port $\phi \in \Phi$ eines Moduls $m \in M$ verbindet $n \in \mathbb{N}$ Module, wobei auch eine Verbindung auf sich selbst möglich ist.*

Verbindungen auf sich selbst können sowohl zu Testzwecken als auch aus Entwurfsgründen erforderlich sein. Dies kann im Fall von Xilinx Aurora sogar ohne externe Verbindung erfolgen, wie in [165, 166] ausgeführt wird.

Definition 4 *Eine Verbindung zwischen zwei Modulen $m, n \in M$ mit den Schnittstellen $s, t \in S$ wird beschrieben durch $\{m, s\} \rightarrow \{n, t\}$. Für eine Verbindung $e \in E$ gilt, es existieren genau ein sendendes Modul $m_t(s_O) \in M$ und $n \in \mathbb{N}$ empfangende Module $m_r(s_I) \in M$.*

Zur Vereinfachung werden bidirektionale Punkt-zu-Punkt-Verbindungen, also solche Verbindungen, die die Bedingung erfüllen,

$$\{m_k, s_{kO}\} \rightarrow \{m_j, t_{jI}\} \wedge \{m_j, t_{jO}\} \rightarrow \{m_i, s_{kI}\}$$

in Zukunft mit Hilfe von Ports wie folgt dargestellt:

$$\{m_k, \phi_k\} \leftrightarrow \{m_j, \phi_j\}$$

Offene Verbindungen, also Verbindungen, für die gilt $\{m_k, s_{kO}\} \rightarrow \emptyset$ oder/und $\emptyset \rightarrow \{m_k, s_{kI}\}$, werden aus Gründen der Übersichtlichkeit nicht aufgeführt.

Die durch Definition 2 ermöglichten offenen Verbindungen können sich neben der offensichtlichen Absicht im Entwurf auch dadurch ergeben, dass durch Verbindungen von einem Modul auf $n \in \mathbb{N}$ andere Module eine Rückantwort nicht möglich ist, was implizit durch Definition 4 begründet wird. Durch Definition 4 ist festgelegt, dass die Abbildung von empfangenden Schnittstellen auf sendende Schnittstellen

eindeutig ist. Der umgekehrte Fall, also die Zuordnung von sendenden zu empfangenden Schnittstellen, kann mehrdeutig sein. Das heißt, dass eine ausgehende Schnittstelle beliebig viele eingehende Schnittstellen erreichen kann, während jeder eingehenden Schnittstelle genau eine ausgehende Schnittstelle zugeordnet ist. Eine ungleiche Aufteilung von eingehenden auf ausgehende Schnittstellen innerhalb eines Systems ergibt sich auf diese Weise ebenfalls.

Beispiel Zwei Module $m_1, m_2 \in M$ mit je einem Port $\phi \in \Phi$ können somit die folgenden $|\,\Xi(V)\,| = 7$ verschiedenen Verbindungsvarianten bilden.

$$\begin{aligned}\Xi\,(V) = \{\,&\emptyset,\\ &(m_1 s_O \rightarrow m_1 s_I)\,,\\ &(m_2 s_O \rightarrow m_2 s_I)\,,\\ &\{(m_1 s_O \rightarrow m_1 s_I)\,, (m_2 s_O \rightarrow m_2 s_I)\}\,,\\ &(m_1 s_O \rightarrow m_2 s_I)\,,\\ &(m_2 s_O \rightarrow m_1 s_I)\,,\\ &(m_1 \phi \leftrightarrow m_2 \phi)\ \}\end{aligned} \tag{3.4}$$

Definition 4 ermöglicht eine eineindeutige Zuordnung eines sendenden Moduls zu einem Empfänger. Da durch die in Definition 4 festgelegte Schreibweise die Abbildung von s_I und s_O auf die Module eindeutig ist, werden die Indizes im Folgenden für eine bessere Übersichtlichkeit ausgelassen.

Beispiel Für die Bedingungen $M = \{m_1, m_2\}$, $\Phi = \{\phi_1, \phi_2\}$ wird beispielhaft die Bestimmung der Anzahl möglicher Varianten aufgezeigt. Die Verschaltung unterliegt keiner Einschränkung, so dass jedes Modul $m_i \in M$ mit jedem seiner beiden Ports ϕ_1 und ϕ_2 jedes andere Modul auf jedem Port erreichen kann. Als Ausgangsmenge lässt sich somit das kartesische Produkt $V = M \times \Phi$ formulieren. Aus der Definition von Ports ergibt sich, dass jedes Modul über zwei eingehende und zwei ausgehende Schnittstellen verfügt. Somit lässt sich die Menge der möglichen ausgehenden Verbindungen für das Tupel (m_1, s_1) gemäß Gleichung 3.5 beschreiben.

$$\begin{aligned}E_{O_{Einzelverbindung}}\,((m_1, s_1)) &= (m_1, s_1) \rightarrow \{\{\emptyset\} \cup V\}\\ &= \{\ \{\emptyset\}\,,\\ &\quad \{(m_1, s_1) \rightarrow (m_1, s_1)\}\,,\\ &\quad \{(m_1, s_1) \rightarrow (m_1, s_2)\}\,,\\ &\quad \{(m_1, s_1) \rightarrow (m_2, s_1)\}\,,\\ &\quad \{(m_1, s_1) \rightarrow (m_2, s_2)\}\ \}\end{aligned} \tag{3.5}$$

Zunächst sind nur die Zuordnungen zu einer möglichen Empfangsschnittstelle formuliert worden. Die Abbildung auf die leere Menge beschreibt eine offene Verbindung und ist somit die leere Menge. Analog zu $E_{O_{Einzelverbindung}}((m_1, s_1))$ gestalten sich die Mengen der möglichen ausgehenden Verbindungen für (m_1, s_2), (m_2, s_1) und (m_2, s_2).

Gestattet man gerichtete Mehrfachverbindungen, wie in Definition 4 formuliert, so erhöht sich abermals die Anzahl der Elemente der ausgehenden Verbindungen (Gleichung 3.5), wobei die Menge der eingehenden Verbindungen gleich bleibt, ebenfalls aufgrund von Definition 4. Gleichung 3.6 zeigt die Bestimmung der erweiterten Menge als Abbildung von (m_1, s_1) auf die Potenzmenge der Ausgangsmenge. Die Mächtigkeit der entstandenen Menge ist somit $|E_O((m_1, s_1))| = 16$.

$$
\begin{aligned}
E_O\left((m_1, s_1)\right) &= (m_1, s_1) \to \mathcal{P}\left(V\right) \\
&= \{\ \emptyset, \\
&\quad \{(m_1, s_1) \to (m_1, s_1)\}, \\
&\quad \{(m_1, s_1) \to (m_1, s_2)\}, \\
&\quad \{(m_1, s_1) \to (m_2, s_1)\}, \\
&\quad \{(m_1, s_1) \to (m_2, s_2)\}, \\
&\quad \{(m_1, s_1) \to \{(m_1, s_1), (m_1, s_2)\}\}, \\
&\quad \{(m_1, s_1) \to \{(m_1, s_1), (m_2, s_1)\}\}, \\
&\quad \{(m_1, s_1) \to \{(m_1, s_1), (m_2, s_2)\}\}, \\
&\quad \{(m_1, s_1) \to \{(m_1, s_2), (m_2, s_1)\}\}, \\
&\quad \{(m_1, s_1) \to \{(m_1, s_2), (m_2, s_2)\}\}, \\
&\quad \{(m_1, s_1) \to \{(m_2, s_1), (m_2, s_2)\}\}, \\
&\quad \{(m_1, s_1) \to \{(m_1, s_1), (m_1, s_2), (m_2, s_1)\}\}, \\
&\quad \{(m_1, s_1) \to \{(m_1, s_1), (m_1, s_2), (m_2, s_2)\}\}, \\
&\quad \{(m_1, s_1) \to \{(m_1, s_1), (m_2, s_1), (m_2, s_2)\}\}, \\
&\quad \{(m_1, s_1) \to \{(m_1, s_2), (m_2, s_1), (m_2, s_2)\}\}, \\
&\quad \{(m_1, s_1) \to \{(m_1, s_1), (m_1, s_2), (m_2, s_1), (m_2, s_2)\}\}\ \}
\end{aligned}
\tag{3.6}
$$

Die Menge der eingehenden Verbindungen bildet sich aus der Vereinigungsmenge der einelementigen Menge mit dem Element der leeren Menge und der Abbildung des kartesischen Produkts der Ausgangsmengen auf das Modul-Verbinder-Tupel (Gleichung 3.7).

$$
\begin{aligned}
E_I\left((m_1, s_1)\right) &= \{\{\emptyset\} \cup V\} \rightarrow (m_1, s_1) \\
&= \{\ \{\{\emptyset\} \rightarrow (m_1, s_1)\}, \\
&\quad \{(m_1, s_1) \rightarrow (m_1, s_1)\}, \\
&\quad \{(m_1, s_2) \rightarrow (m_1, s_1)\}, \\
&\quad \{(m_2, s_1) \rightarrow (m_1, s_1)\}, \\
&\quad \{(m_2, s_2) \rightarrow (m_1, s_1)\}\ \}
\end{aligned}
\tag{3.7}
$$

Durch Bilden der Vereinigung der Mengen $E_O((m_1, s_1))$ und $E_I((m_1, s_1))$ erhält man die Menge aller möglichen gerichteten Kanten im Verbindungsgraph für (m_1, s_1), dargestellt in Gleichung 3.8.

$$
\begin{aligned}
E\left((m_1, s_1)\right) &= E_I \cup E_O \\
&= \{\ \emptyset, \\
&\quad \{(m_1, s_1) \rightarrow (m_1, s_1)\}, \\
&\quad \{(m_1, s_2) \rightarrow (m_1, s_1)\}, \\
&\quad \{(m_2, s_1) \rightarrow (m_1, s_1)\}, \\
&\quad \{(m_2, s_2) \rightarrow (m_1, s_1)\}, \\
&\quad \{(m_1, s_1) \rightarrow (m_1, s_2)\}, \\
&\quad \{(m_1, s_1) \rightarrow (m_2, s_1)\}, \\
&\quad \{(m_1, s_1) \rightarrow (m_2, s_2)\}, \\
&\quad \{(m_1, s_1) \rightarrow \{(m_1, s_1), (m_1, s_2)\}\}, \\
&\quad \{(m_1, s_1) \rightarrow \{(m_1, s_1), (m_2, s_1)\}\}, \\
&\quad \{(m_1, s_1) \rightarrow \{(m_1, s_1), (m_2, s_2)\}\}, \\
&\quad \{(m_1, s_1) \rightarrow \{(m_1, s_2), (m_2, s_1)\}\}, \\
&\quad \{(m_1, s_1) \rightarrow \{(m_1, s_2), (m_2, s_2)\}\}, \\
&\quad \{(m_1, s_1) \rightarrow \{(m_2, s_1), (m_2, s_2)\}\}, \\
&\quad \{(m_1, s_1) \rightarrow \{(m_1, s_1), (m_1, s_2), (m_2, s_1)\}\}, \\
&\quad \{(m_1, s_1) \rightarrow \{(m_1, s_1), (m_1, s_2), (m_2, s_2)\}\}, \\
&\quad \{(m_1, s_1) \rightarrow \{(m_1, s_1), (m_2, s_1), (m_2, s_2)\}\}, \\
&\quad \{(m_1, s_1) \rightarrow \{(m_1, s_2), (m_2, s_1), (m_2, s_2)\}\}, \\
&\quad \{(m_1, s_1) \rightarrow \{(m_1, s_1), (m_1, s_2), (m_2, s_1), (m_2, s_2)\}\}\ \}
\end{aligned}
\tag{3.8}
$$

Eine Abbildung der leeren Menge auf (m_1, s_1), wie sie in E_I vorkommt, sagt aus, dass es keine eingehenden Verbindungen zu (m_1, s_1) gibt. Dies ist für die Anzahl

der vorhandenen Verbindungen mit der leeren Menge gleichzusetzen. Zur Bestimmung der Gesamtmenge Ξ der gerichteten Kanten für M, S wird die Vereinigung aus den einzelnen Kantenmengen über V gebildet. Die vollständige Menge aus Gleichung 3.9 ist im Anhang C dargestellt.

$$\Xi(V) = E((m_1, s_1)) \cup E((m_1, s_2)) \cup E((m_2, s_1)) \cup E((m_2, s_2)) \tag{3.9}$$

Benötigt wird diese Form der Darstellung auf Schnittstellenebene bei der Realisierung eines Clusters mit Hardware-Komponenten. Für die auf dem Cluster abzubildende Anwendung ist die exakte Information darüber erforderlich, welche Schnittstellen miteinander verbunden sind. Dies gilt auch, wenn die Verbindungen zwischen zwei Komponenten durch geschaltete Verbindungsbrücken (Switche) hergestellt werden.

Im folgenden Abschnitt wird die hier beschriebene feingranulare Darstellung auf die Ebene der globalen Verbindungen des Clusters abstrahiert, um einen Überblick über die Verschaltungsmöglichkeiten der einzelnen Systeme im Cluster zu gewinnen. Darauf aufbauend können Optimierungen der Struktur vorgenommen werden.

3.3 Allgemeine Bestimmung der Cluster-Topologien

Neben der Bestimmung der Verbindungsvarianten ist die Evaluation der eigentlichen Cluster-Topologie für eine effiziente Implementierung der Kommunikationsinfrastruktur essentiell. Aufgrund von Hardware-bedingten Einschränkungen bezüglich der realisierbaren Verbindungen können in der Praxis zwangsläufig nicht alle theoretischen Verbindungsvarianten umgesetzt werden. Kabel und Steckverbinder bündeln die Ausgangsschnittstellen der Cluster-Komponenten und beschränken in der Regel eine wahlfreie Zuführung auf alle Eingangsschnittstellen der zu verbindenden Systeme. Aus diesem Grund ist aus dem eingeschränkten Entwurfsraum eine Topologie auszuwählen, die entweder die geplante Anwendung optimal unterstützt oder für eine Vielzahl an Umsetzungen geeignet ist, ohne dass ein Umstecken oder Neuverkabeln notwendig ist.

Im Rahmen dieser Arbeit werden die Bündelung von FPGAs zu einem Cluster betrachtet und die für diese Architektur relevanten speziellen Anforderungen herausgestellt. Soll etwa der FPGA-Cluster parallel von mehreren Anwendern genutzt werden, so kann unter Umständen eine Veränderung der Topologie sogar zu Unterbrechungen in der Kommunikation zwischen verschiedenen Bereichen des Clusters führen. Im Hinblick auf die parallele Verwendung durch mehrere Benutzer ist außerdem eine Anordnung der Cluster-Komponenten zu wählen, welche verschiedenen Ressourcenanforderungen gerecht wird. Themen wie Verschnitt bei der Partitionierung und Fragmentierung durch das Freiwerden und das Neubelegen von FPGAs müssen ebenfalls bei der Betrachtung erfasst werden.

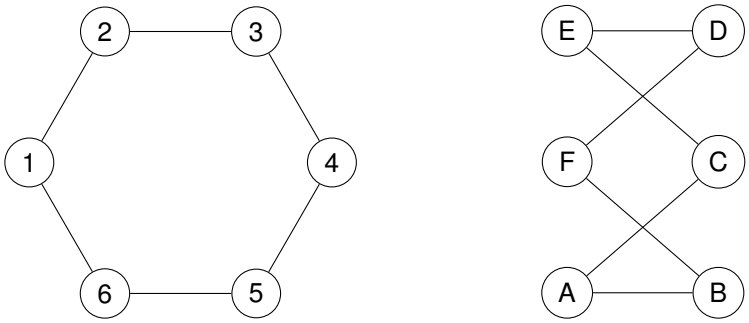


Abbildung 3.2: Isomorphe Graphen

Durch eine Modellierung der realisierbaren Cluster-Topologien wird eine Abstraktion gegenüber der Darstellung der Verbindungsvarianten mit Hilfe einer Graphenbeschreibung erreicht. Die Knotenmenge V entspricht den Cluster-Komponenten, die Verbindungen zu anderen Komponenten aufbauen können. Während die Ausgangsvalenz eines Knotens $d(v)$ bei der Beschreibung der Verbindungsvarianten über die Anzahl der genutzten Ausgangsschnittstellen $s_1..s_k \in S_o, k = |S_o|$ definiert ist und somit Werte zwischen 0 und $|S_o|$ annimmt, kann für die Abbildung der Topologie auf eine gerichtete Darstellung verzichtet werden. Darüber hinaus bündeln, wie im vorhergehenden Abschnitt ausgeführt, Kabel- und Steckverbinder mehrere Kommunikationsleitungen, so dass eine Betrachtung, wie viele Schnittstellen je Verbindungskomponente genutzt werden, für die Topologiedarstellung nicht relevant ist. Durch diese Abstraktionen ergibt sich, dass der Graph k-regulär ist. Darüber hinaus muss der Graph zusammenhängend sein, da andernfalls zu den isolierten Teilen keine Daten gelangen können und sie somit nicht Bestandteil des Clusters sind. Eine optimale Ausnutzung der begrenzten Kommunikationswege setzt voraus, dass alle gebotenen Ressourcen durch die darstellende Struktur genutzt werden, der Graph somit k-regulär ist und keine Mehrfachkanten beinhaltet. Erfasst werden dürfen nur unterschiedliche Topologien, also solche, deren beschreibende Graphen nicht isomorph sind. Die beiden Graphen in Abbildung 3.2 sind trotz unterschiedlicher Darstellung und Benennung ihrer Knoten isomorph und werden entsprechend bei der Topologiebetrachtung als eine Beschreibung gezählt.

In Tabelle 3.1 und Abbildung 3.3 ist für $k \geq 3$ und eine ansteigende Mächtigkeit von V die Anzahl der möglichen Topologien $\mid \Gamma_{k\_|V|} \mid$ aufgeführt worden, welche unter der Voraussetzung einer wahlfreien Verbindung der Cluster-Komponenten realisierbar sind. Sollte die Wahlfreiheit in der Praxis nicht gegeben sein, weil etwa starre Steckverbindungen statt flexibel konnektierbarer Kabelverbindungen durch die Hardware vorgegeben sind, fällt das System aus der Betrachtung aufgrund der Vorgabe, dass alle vorhandenen Verbindungsmöglichkeiten genutzt werden sollen.

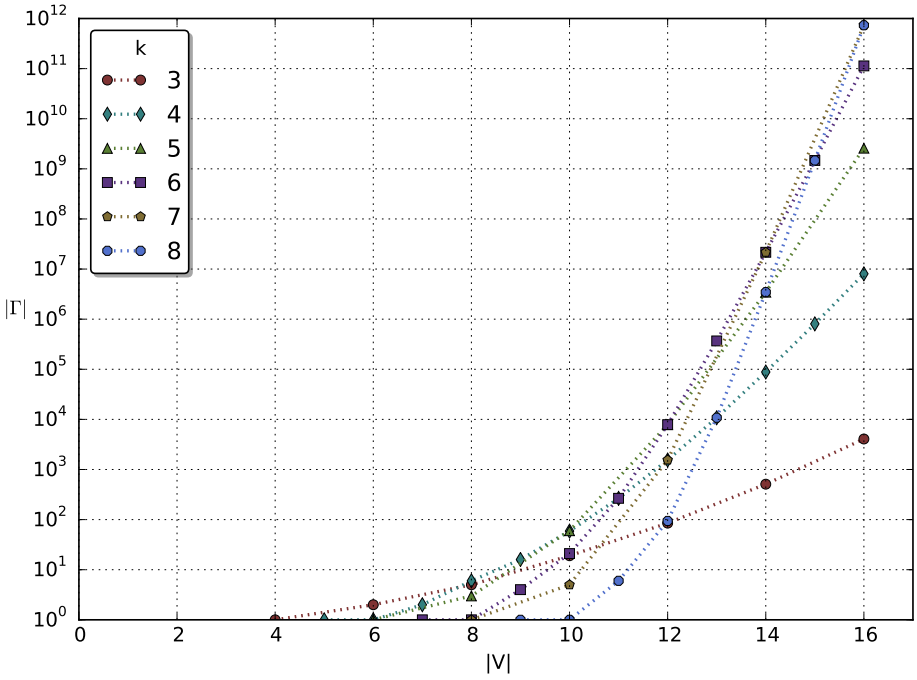


Abbildung 3.3: Mächtigkeit $\Gamma(V,k)$ in Abhängigkeit von $|V|$ für ansteigende k

Aus diesem Grund beginnen die aufgeführten Topologievarianten immer mit dem vollständigen Graphen. Ein vollständiger Graph erfüllt die Bedingung $k = |V| - 1$ und hat $|E| = \frac{|V|k}{2}$ Kanten. Für $k = 2$ ergibt sich eine Ringtopologie für $|V| \geq 3$, weshalb dieser Sonderfall nicht explizit aufgeführt worden ist. Erzeugt wurden die Daten mit Hilfe des in [167] vorgestellten Verfahrens zur schnellen Generierung von Graphen, welches auf einer Intel Core i7-3770 CPU mit 16 GB Arbeitsspeicher die Berechnung von ca. $50\,000$ Graphen pro Sekunde ermöglicht. Dabei wurden die Werte für $\left|\Gamma_{4\_20}\right|$, $\left|\Gamma_{5\_18}\right|$ sowie sämtliche Topologievarianten $\left|\Gamma_{8\_|V|}\right|$ erstmalig im Rahmen dieser Arbeit ermittelt, alle anderen auf [168] veröffentlichten Ergebnisse konnten bestätigt werden.

Aus den aufgezeigten Topologievarianten gilt es, die Struktur auszuwählen, die entweder das Anwendungsszenario des FPGA-Clusters unterstützt, wie etwa systolische Strukturen für die Genom- bzw. Protein-Sequenzierung, oder eine einfache Partitionierung großer Entwürfe ermöglicht. Regelmäßige Strukturen erleichtern die Platzierung, da im Cluster unterschiedliche FPGAs sich im Gesamtkontext gleich

Tabelle 3.1: k-reguläre, zusammenhängende, nichtisomorphe Graphen

$\|V\|$	k	Topologie-varianten $\|\Gamma\|$	$\|V\|$	k	Topologie-varianten $\|\Gamma\|$
4	3	1	5	4	1
6	3	2	6	4	1
8	3	5	7	4	2
10	3	19	8	4	6
12	3	85	9	4	16
14	3	509	10	4	59
16	3	4 060	11	4	256
18	3	41 301	12	4	1 544
20	3	510 489	13	4	10 778
22	3	7 319 447	14	4	88 168
24	3	117 940 535	15	4	805 491
26	3	2 094 480 864	16	4	8 037 418
			17	4	86 221 634
			18	4	985 870 522
			19	4	11 946 487 647
			20	4	152 808 063 181

$\|V\|$	k	Topologie-varianten $\|\Gamma\|$	$\|V\|$	k	Topologie-varianten $\|\Gamma\|$
6	5	1	7	6	1
8	5	3	8	6	1
10	5	60	9	6	4
12	5	7 848	10	6	21
14	5	3 459 383	11	6	266
16	5	2 585 136 675	12	6	7 849
18	5	2 807 105 250 897	13	6	367 860
			14	6	21 609 300
			15	6	1 470 293 675
			16	6	113 314 233 808

$\|V\|$	k	Topologie-varianten $\|\Gamma\|$	$\|V\|$	k	Topologie-varianten $\|\Gamma\|$
8	7	1	9	8	1
10	7	5	10	8	1
12	7	1 547	11	8	6
14	7	21 609 301	12	8	94
16	7	733 351 105 934	13	8	10 786
			14	8	3 459 386
			15	8	1 470 293 676
			16	8	733 351 105 935

verhalten. In Bezug auf die Latenz beim Nachrichtenaustausch zwischen zwei oder mehreren Knoten lässt sich die Struktur jedoch weiter optimieren.

Zur Darstellung der Topologie wird in dieser Arbeit nicht die Adjazenzmatrix verwendet, sondern die Distanzmatrix, da diese neben den reinen Verbindungsinformationen die Pfadlänge von allen Knoten zu allen anderen Knoten enthält. Sollte dennoch die Adjazenzmatrix benötigt werden, so kann diese im Fall ungewichteter Graphen mit der Beschreibung 3.10 aus der Distanzmatrix $D^M = [p_{i,j}]$ gebildet werden, indem alle Pfade der Länge 1 zwischen zwei Knoten eine direkte Nachbarschaftsbeziehung beschreiben; somit wird ein Eintrag in der Adjazenzmatrix $A^M = [a_{i,j}]$ erforderlich.

$$a(i,j) = \begin{cases} 1, & \text{falls } p(i,j) = 1 \\ 0, & \text{sonst} \end{cases} \tag{3.10}$$

3.4 Topologieoptimierung

Zur Minimierung der Latenz bei der Übertragung von Daten muss die Anzahl an Schritten innerhalb des Netzes minimiert werden. Der kürzeste Weg innerhalb eines Graphen $G = (V, E)$ ist definiert als derjenige Pfad p zwischen zwei Knoten $s, t \in V$, dessen Summe der Kantengewichte minimal gegenüber allen anderen Pfaden von s nach t ist [164]. Innerhalb der Topologiebetrachtung sind alle Pfade gleichwertig, weshalb alle Kanten ein Kantengewicht von $c\,(e) = 1$ haben. Demzufolge lässt sich die Länge eines Pfades p wie in Formel 3.11 angeben und entspricht somit der Anzahl an im Pfad enthaltenen Kanten.

Der Ansatz, alle Kanten als gleichwertig hinsichtlich ihres Kantengewichts $c\,(e)$ anzusehen, schränkt das Ergebnis der Analyse insofern nicht ein, als dass im Hinblick auf die später tatsächlich eingesetzte Hardware des Clusters eine homogene Verbindung der einzelnen Knoten vorausgesetzt werden kann. Betrachtet werden nur solche Verbindungen, die eine Cluster-Bildung ermöglichen, was in den meisten Fällen serielle Hochgeschwindigkeitsverbindungen sind (siehe Kapitel 4.2 und 5.6). Eventuell zusätzlich vorhandene Verbindungen auf den einzelnen Knoten des Clusters haben keinen Einfluss auf die globale Verbindungsstruktur der Knoten untereinander und werden deshalb in die Optimierung nicht mit einbezogen.

$$c\,(p) = \sum_{e \in p} 1 \tag{3.11}$$

Die Exzentrizität $\epsilon\,(v)$ eines Knotens $v \in V$ ist der maximale Pfad p zu allen anderen Knoten von G [169].

$$\epsilon\,(v) = \max_{t \in V} p\,(v,t) \tag{3.12}$$

Für ein gegebenes k und eine Anzahl an Knoten $n = |V|$ bedeutet dies, dass die Teilmenge an Graphen Θ_{k_n} aus der Menge der k-regulären, zusammenhängenden, nichtisomorphen Graphen Γ_{k_n} mit $\gamma \in \Gamma$ gefunden werden muss, für die gilt, dass der Durchmesser $D(G)$ der enthaltenen Graphen $\vartheta \in \Theta$ minimal ist.

$$D(G) = \max_{v \in V} \epsilon(v) \tag{3.13}$$

$$\Theta_{k\_n} := \min_{\gamma \in \Gamma_{k\_n}} D(\gamma) \tag{3.14}$$

Aus der Teilmenge $\Theta_{k\_n} \subseteq \Gamma_{k\_n}$ können jetzt die Graphen ausgewählt werden, die die geringste Exzentrizitätssumme $\psi(G)$ über alle Knoten aufweisen.

$$\psi(G) = \sum_{v \in V} \epsilon(v) \tag{3.15}$$

$$\Omega_{k\_n} := \min_{\vartheta \in \Theta_{k\_n}} \psi(\vartheta) \tag{3.16}$$

Dadurch ergibt sich das in Formel 3.16 definierte Optimierungsziel Ω_{k_n}, das nur noch Graphen $\omega \in \Omega_{k\_n}$ enthält, für die für alle Knoten die Pfadlängen zu allen anderen Knoten minimiert sind. Somit ist die topologiebedingte Latenz bei Datenübertragungen ebenfalls minimal. Für den Vergleich unterschiedlicher Topologien kann der auf die Anzahl der Knoten normierte Wert aus 3.17 als Exzentrizitätsindex $\zeta(G)$ herangezogen werden.

$$\zeta(\omega) = \frac{\psi(\omega)}{|V|}, \omega \in \Omega_{k\_n} \tag{3.17}$$

Der Durchmesser beschreibt den längsten Pfad des Graphen und bildet damit die Obergrenze, welche für einen Vergleich der Topologien herangezogen werden kann. Zusammen mit dem Exzentrizitätsindex lassen sich somit zwei Schranken beschreiben, die für die Beurteilung der Topologie hinsichtlich ihrer Kompaktheit herangezogen werden können.

Beispiel Gegeben sei die Menge Γ_{3_22} aller nichtisomorphen, zusammenhängenden, 3-regulären Graphen mit $|V| = 22$. Die Mächtigkeit dieser Menge ist $\left|\Gamma_{3\_22}\right| = 7\,319\,447$. Die Menge Θ_{3_22} beinhaltet alle Graphen, deren Durchmesser $D(G) = 4$ beträgt und umfasst insgesamt $\left|\Theta_{3\_22}\right| = 185\,836$ Graphen. Aufgrund der Auswahl der Graphen mit minimalem $\psi\left(\Theta_{3\_22}\right)$ beträgt der Exzentritzitätsindex lediglich $\zeta\left(\omega_{3\_22}\right) = 3,\overline{63}$ und die Ergebnismenge Ω_{3_22} ist einelementig. Der Exzentrizitätsvektor des Graphen stellt sich wie folgt dar:

$$\vec{\epsilon}(V) = (4\ 4\ 4\ 4\ 4\ 4\ 3\ 4\ 3\ 4\ 3\ 3\ 4\ 3\ 3\ 4\ 3\ 3\ 4\ 4\ 4\ 4\ 4)^{\intercal}$$

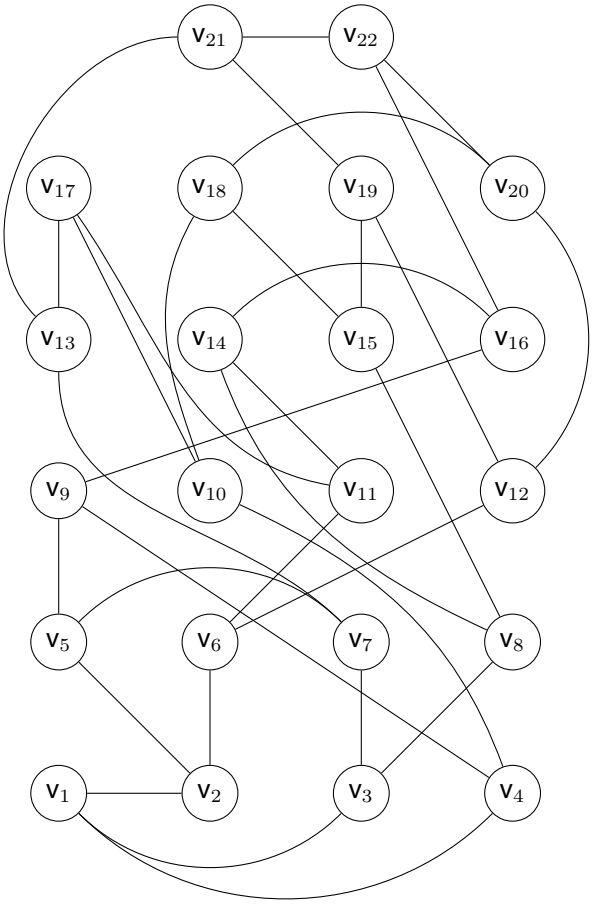


Abbildung 3.4: Graph ω_{3_22}

In Abbildung 3.4 wird der zugehörige Graph gezeigt. Die Adjazenzmatrix von Ω_{3_22} wird im Anhang unter C.1 aufgeführt.

3.4.1 Ausgewählte regelmäßige Topologien

Im Folgenden wird ein Überblick über regelmäßig strukturierte Graphen gegeben. In den Abbildungen 3.5a bis 3.5f werden typische Graphen dargestellt, die in Netzwerken und Cluster-Systemen verwendet werden. Der Vorteil von regelmäßigen Strukturen liegt in einem geringen Routing-Aufwand und der Möglichkeit, bei

Tabelle 3.2: Topologievergleich

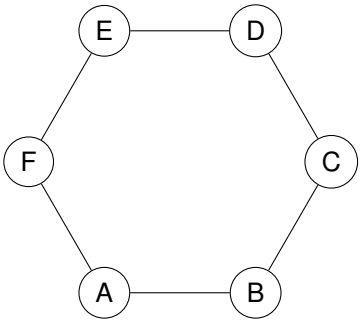
Topologie	**Valenz** $\delta(G)$ | $\Delta(G)$	**Durchmesser** $D(G)$	**Exzentrizitätssumme** $\psi(G)$
Ring	2-regulär	$\lfloor \frac{\lvert V\rvert}{2} \rfloor$	$\lvert V\rvert \cdot D(G)$
Gitter	2 | 4	$2\left(\sqrt{\lvert V\rvert} - 1\right)$	$\lvert V\rvert \cdot D(G)$
2D-Torus	4-regulär	$\sqrt{\lvert V\rvert}$	$\lvert V\rvert \cdot D(G)$
3D-Torus	6-regulär	$\frac{3}{2} \cdot \sqrt[3]{\lvert V\rvert}$	$\lvert V\rvert \cdot D(G)$
nD-Torus	$2n$-regulär	$\frac{n}{2} \cdot \sqrt[n]{\lvert V\rvert}$	$\lvert V\rvert \cdot D(G)$
Würfel	3-regulär	3	24
Tesserakt	4-regulär	4	64
nD-Hyperwürfel	n-regulär	n	$n2^n$
Vollständiger Graph	$(\lvert V\rvert - 1)$-regulär	1	$\lvert V\rvert$

höher dimensionalen Strukturen gleichlange alternative Pfade zu verwenden. Der Umstand, auf alternative Routing-Wege zurückgreifen zu können, ist in solchen dynamischen Systemen relevant, in denen die Knoten scheinbar zufällig miteinander kommunizieren. Dabei können Übertragungen zwischen zwei Knoten bestimmte Pfade blockieren, so dass ein Ausweichen über andere Wege das Versenden bzw. Empfangen der Daten bei gleicher Übertragungszeit ermöglicht. Da für die Kommunikation innerhalb eines FPGA-Clusters bei den meisten Applikationen der Kommunikationsaufwand bereits im Vorhinein durch die Partitionierung der Anwendung zwischen den Knoten bekannt ist, wird kein lastabhängiges Routingkonzept benötigt.

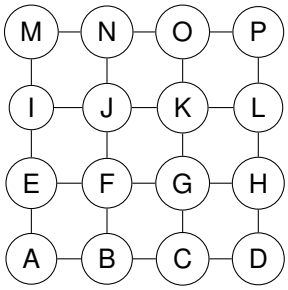
Für die gegebenen und weiteren Topologien sind die jeweiligen Durchmesser in Tabelle 3.2 aufgeführt. Gemein ist allen vollständigen k-regulären Graphen, dass die Bedingung aus Gleichung 3.18 erfüllt ist.

$$|E| = \frac{1}{2} |V| \, k \tag{3.18}$$

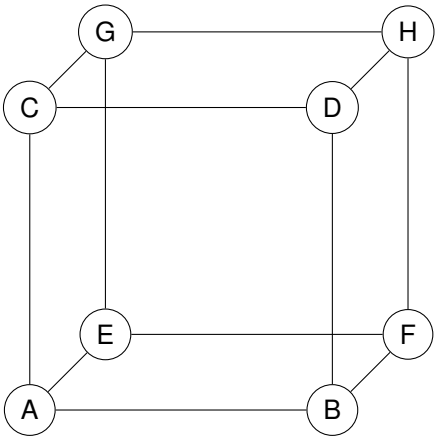
Je nach Anwendungsgebiet und Partitionierung der Implementierung können unterschiedliche Topologien für eine Umsetzung besser geeignet sein. Die Flexibilität andere Strukturen abbilden zu können steigt mit der Valenz der Knoten. So kann eine alle Knoten erfassende Ringtopologie in allen anderen aufgezeigten Topologien abgebildet werden; beispielsweise sieht der alle Knoten durchlaufende Ringpfad im Tesserakt wie folgt aus:



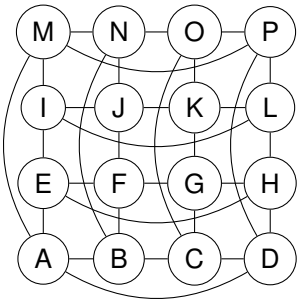
(a) Ringtopologie



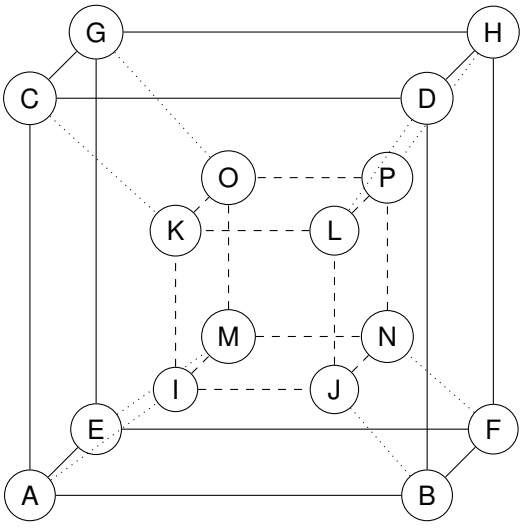
(b) Gitter



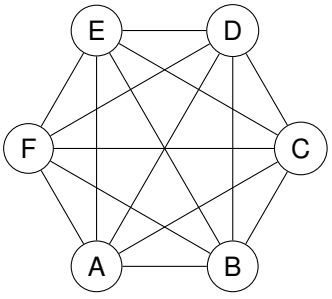
(c) Würfel



(d) 2D-Torus



(e) Tesserakt (4D-Hyperwürfel)



(f) Vollständiger Graph

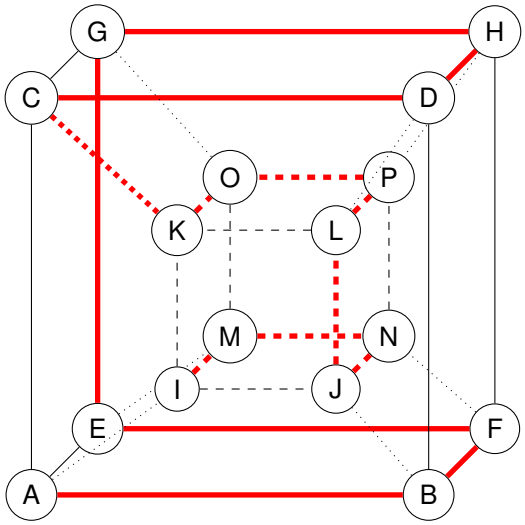


Abbildung 3.6: Ringpfad im Tesserakt

$$
\begin{aligned}
p_{Ring}\,(Tesserakt) = \{\ & (A, s_1) \rightarrow (B, s2)\,, (B, s_1) \rightarrow (F, s2)\,, \\
& (F, s_1) \rightarrow (E, s2)\,, (E, s_1) \rightarrow (G, s_2)\,, \\
& (G, s_1) \rightarrow (H, s_2)\,, (H, s_1) \rightarrow (D, s_2) \\
& (D, s_1) \rightarrow (C, s_2)\,, (C, s_1) \rightarrow (K, s_2) \\
& (K, s_1) \rightarrow (O, s_2)\,, (O, s_1) \rightarrow (P, s_2) \\
& (P, s_1) \rightarrow (L, s_2)\,, (L, s_1) \rightarrow (J, s_2) \\
& (J, s_1) \rightarrow (N, s_2)\,, (N, s_1) \rightarrow (M, s_2) \\
& (M, s_1) \rightarrow (I, s_2)\ \}
\end{aligned}
$$

Durch einen letzten Schritt in der Abbildung 3.6 von $p = \{(I, s_1) \rightarrow (A, s_2)\}$ könnte der Kreis geschlossen werden. Ringpfade, die beliebige Knoten durchlaufen, können mit Ausnahme vollständiger Graphen nicht zwangsläufig gebildet werden. Entsprechend kann ein Cluster-Aufbau, der mit Hilfe eines vollständigen Graphen beschrieben werden kann, als am flexibelsten angesehen werden. Da jedoch diese Form der Umsetzung aufgrund von eingeschränkten Platz- und Routingressourcen auf der realen Hardware nicht zwangsläufig zu erzielen ist, muss auf andere Darstellungen zurückgegriffen werden. Für die Anforderung, den Zielknoten in möglichst wenigen Schritten zu erreichen, lassen sich optimierte Topologien mit Hilfe des im vorangegangenen Abschnitt 3.4 vorgestellten Verfahrens finden.

Tabelle 3.3: Regelmäßige und ψ-optimale Strukturen

Topologie	**Durchmesser**		**Exzentrizitätssumme**	
	$D(G)$	$D\left(\omega_{k\_\lvert V\rvert}\right)$	$\psi(G)$	$\psi\left(\omega_{k\_\lvert V\rvert}\right)$
Ring	8	3	128	36
2D-Torus	4	3	64	36
Würfel	3	2	24	16
Tesserakt	4	3	64	36

3.4.2 Gegenüberstellung von regelmäßigen und optimierten Topologien

Vergleicht man die im vorherigen Abschnitt vorgestellten regelmäßigen Strukturen mit den auf eine minimale Exzentrizitätssumme hin optimierten Topologien, so zeigt sich, dass es möglich ist, zu jeder regelmäßigen Struktur mit Ausnahme der Ringtopologie, eine kompaktere Darstellung mit Hilfe des in Abschnitt 3.4 vorgestellten Verfahrens zu finden. Tabelle 3.3 fasst die Ergebnisse für eine angenommene Obergrenze von $|V| \leq 16$ zusammen. Die Unterschiede in den Exzentrizitätssummen treten sogar bei einfachen Strukturen wie dem Würfel deutlich hervor. Verfahrensbezogen fällt auch der Durchmesser für die optimalen Strukturen geringer aus.

In Abbildung 3.7 wird die Exzentrizitätssumme $\psi(\omega), \omega \in \Omega_{k\_|V|}$ über ansteigende Knotenzahlen $|V|$ und ansteigende Valenz der Knoten k aufgezeigt. Der Einfluss der Valenz wird insbesondere für Werte von $k \geq 5$ deutlich. Für ungerade k lassen sich nur k-reguläre Graphen mit gerader Anzahl an Knoten bilden, so dass die dargestellten Zwischenwerte eine rein theoretische Interpolation sind. Änderungen der Steigung in der Abbildung gehen mit einer Erhöhung des Durchmessers $D(\omega), \omega \in \Omega_{k\_|V|}$ einher.

3.5 Zusammenfassung

In den vorangegangenen Abschnitten des Kapitels wurden zwei Verfahren zur Modellierung der möglichen Verbindungsvarianten eines FPGA-Clusters aufgezeigt. Die beiden Beschreibungen unterscheiden sich hinsichtlich ihres Abstraktionsgrades und dienen somit der Extraktion zweier sich ergänzender Informationen. Während die feingranulare Darstellung die Verbindungsvarianten auf Schnittstellenebene eines einzelnen FPGAs innerhalb der vollständigen FPGA-Cluster-Struktur aufzeigt und somit eine Beschreibung des Vernetzungsgrades innerhalb der einzelnen Systemkomponenten ermöglicht, erfolgt durch die Aufbereitung der höher abstrahierten

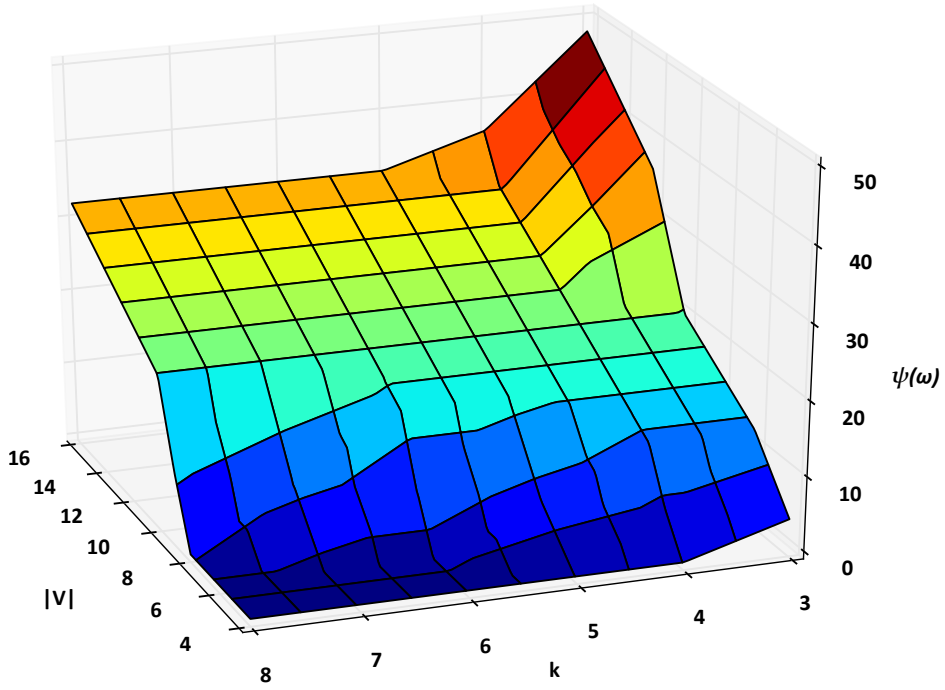


Abbildung 3.7: Exzentrizitätssumme $\psi(\omega)$ für ansteigende $|V|$ und k

Cluster-Topologie eine Darstellung, die eine Beurteilung über die Abbildbarkeit unterschiedlicher Anwendungsfälle auf der globalen Systemebene abgibt.

Auf Basis der Darstellung der Cluster-Topologie wird ein Verfahren zur Optimierung des den FPGA-Cluster beschreibenden Graphen aufgezeigt, mit dessen Hilfe eine Struktur bestimmt werden kann, die in Bezug auf die maximal notwendigen Schritte minimal ist, um Nachrichten zwischen zwei Knoten entlang aller Pfade zu übertragen. Da die Pfadlänge direkt mit der benötigten Übertagungszeit korreliert, lässt sich durch Umsetzung des ermittelten kompakten Graphen die Latenz für die Kommunikation im FPGA-Cluster minimieren.

Abschließend werden häufig in der Literatur anzutreffende reguläre Topologien sowie ihre jeweiligen Durchmesser und Exzentrizitätssummen dargestellt. Eine Gegenüberstellung mit ihren ψ-optimalen Strukturen zeigt deutlich, dass ein Abweichen von einer regelmäßigen Struktur zu wesentlich kürzeren Wegen innerhalb des Graphen führt. Da nicht alle in Kapitel 4 vorgestellten Systeme die Möglichkeit bieten, eine optimierte Cluster-Struktur aufzubauen, kann mit Hilfe dieser Angaben die erzielbare Leistungsfähigkeit der Kommunikationsstruktur beurteilt werden.

Ausgehend von der optimierten Graphenbeschreibung einer Cluster-Topologie werden im folgenden Kapitel derzeitig in der Forschung eingesetzte FPGA-Cluster-Systeme vorgestellt. Dabei wird insbesondere die Verbindung der einzelnen FPGAs innerhalb des Clusters betrachtet.

4 Stand der Technik von FPGAs und FPGA-Clustern

Die in Kapitel 2.3 vorgestellten Anwendungsfelder für FPGA-Cluster erfordern für eine effiziente Umsetzung eine Plattform, welche auf die spezifischen Anforderungen hin ausgelegt ist. Wie in den nachfolgenden Abschnitten gezeigt wird, existieren viele FPGA-Cluster-Systeme, die im Hinblick auf eine spezifische Anwendung entstanden sind. Darüber hinaus gibt es Cluster, die aufgrund ihrer Anpassbarkeit universell einsetzbar sind und auf denen die Anwendungen dedizierter Systeme umgesetzt werden können. Möglich wird die Portierung auf ein anderes als das ursprüngliche System mit vertretbarem Aufwand nur, wenn insbesondere die Kommunikationsstruktur flexibel genug ist. Dabei ist ein Umzug von FPGAs älterer Generation auf FPGAs neueren Entwicklungsstandes meist einfacher zu vollziehen als in die entgegensetzte Richtung. Auch unter Auslassung dedizierter Hardmacros führt die fortschreitende Fertigungstechnologie zu einer höheren Logikdichte, so dass bei gleichbleibender Systemarchitektur des Clusters eine neue Partitionierung zwar angeraten, jedoch nicht zwingend erforderlich ist.

Im Folgenden wird der aktuelle Entwicklungsstand im Bereich rekonfigurierbarer Hardware aufgezeigt. Dabei findet sowohl der für den Benutzer wenig beeinflussbare Hardware-Aufbau als auch der Entwurfsablauf zur Beschreibung von Funktionen Beachtung, welche auf der Hardware abgebildet werden. In der grafischen Aufbereitung der einzelnen Systeme wird der Fokus auf den Teil der Kommunikationsinfrastruktur gelegt, der die Verbindung mehrerer Systemkomponenten und somit eine Cluster-Bildung ermöglicht.

Nach einer Einordnung des FPGA-Cluster-Begriffs im Rahmen dieser Arbeit folgt in Abschnitt 4.1 eine Abgrenzung gegenüber MFS. Eingegrenzt durch diese Definition werden die Entstehung von FPGA-Clustern beleuchtet und anschließend aktuelle in der Forschung eingesetzte FPGA-Cluster vorgestellt. Die Architekturübersicht über die einzelnen Systeme erfolgt in einer abstrahierten Form, wodurch Merkmale der Verbindungstopologie und der Speicheranbindung hervorgehoben werden. Für eine ausführlichere Darstellung sei auf die herstellereigenen Blockschaltbilder in Anhang B verwiesen. Eine kurze Zusammenfassung in Abschnitt 4.3 schließt das Kapitel ab.

4.1 Definition von FPGA-Clustern

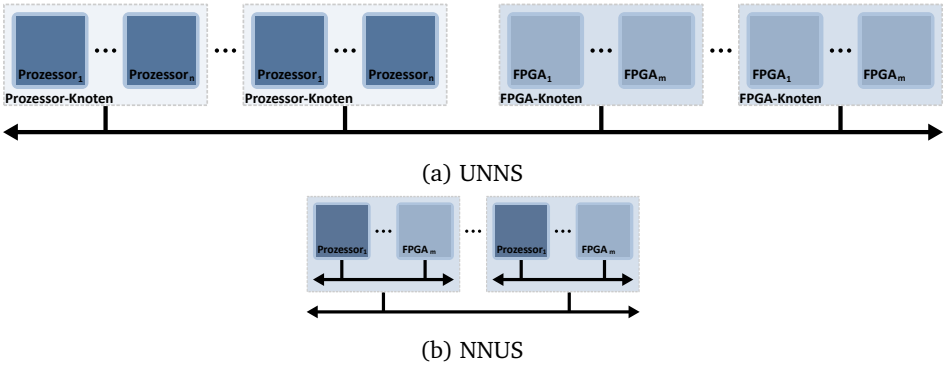
Dieses Kapitel gibt eine Definition von FPGA-Clustern im Rahmen dieser Arbeit. Hierzu wird eine Unterteilung in zwei verschiedene Varianten der Realisierung von Cluster-Systemen vorgestellt, sowie eine Abgrenzung gegenüber MFS aufgezeigt.

In dieser Arbeit untersuchte FPGA-Cluster implementieren sämtliche Benutzeralgorithmen in FPGAs. Erweiterungen in Form von Prozessoren, eingebettet oder als diskrete Elemente, können Sonder- oder Steuerungsaufgaben übernehmen. Viele Systeme aus dem HPRC-Segment kombinieren klassische Serverprozessorarchitekturen mit FPGAs als Koprozessorelement, beispielsweise der Cray XT5h und der Convey HC-1. Eine Gemeinsamkeit besteht darin, dass der Haupt-Thread auf dem Betriebssystem des Prozessors läuft und berechnungsintensive Abläufe auf den FPGA bzw. die FPGAs ausgelagert sind.

Sofern nicht anders angegeben, beziehen sich die im weiteren Verlauf der Arbeit gemachten Angaben auf SRAM-basierte feingranulare FPGA-Architekturen. Grobgranulare Strukturen wie Application Specific Inflexible FPGAs (ASIF) [170] tauschen Teile ihrer Flexibilität zugunsten höherer Verarbeitungsraten in speziellen Anwendungsgebieten ein. Aus diesem Grund sind sie nicht Bestandteil der weitergehenden Betrachtung in dieser Arbeit.

Unterteilt werden die Systeme gemäß [171] in Systeme mit gleichartigen Komponenten auf Knotenebene (engl.: Uniform Nodes / Non-Uniform System, UNNS) und Systeme verschiedenartiger Komponenten auf Knotenebene (engl.: Non-Uniform Nodes / Uniform System, NNUS). UNNS beherbergen verschiedenartige Knoten, wobei jeder Knoten allerdings nur eine Funktionalität beispielsweise in Form eines Prozessors oder eines FPGAs zur Verfügung stellen. Durch die Kopplung dieser in sich homogenen Knoten lässt sich ein System aufbauen, das nur aus Verarbeitungseinheiten besteht, die für die Anwendung benötigt werden. Nachteilig bei dieser Struktur ist, dass die Latenz beim Austausch von Daten zwischen CPUs und FPGAs steigt. NNUS wie beispielsweise der Cray XD1 [172] bestehen demgegenüber aus mehreren baugleichen Knoten, die ihrerseits gemischte Verarbeitungseinheiten beinhalten, beispielsweise Prozessoren und FPGAs gebündelt auf einem Knoten. Durch das vorgegebene Verhältnis von FPGAs zu CPUs bleiben Verarbeitungseinheiten ungenutzt, sofern die Partitionierung des abzubildenden Algorithmus nicht exakt der vorhandenen Hardware entspricht. Beiden Arten von Systemen ist gemein, dass die Knoten Daten entweder über eine Verbindungsstruktur oder einen globalen geteilten Speicher austauschen. Abbildung 4.1 verdeutlicht die beiden Architekturkonzepte [173].

Trotz der Größe moderner FPGAs im Hinblick auf die zur Verfügung gestellte Anzahl an Logik-, Speicher- und Hard-IP-Core-Elementen existieren Anwendungsfelder (siehe Kapitel 2.3), deren Ressourcenbedarf aufgrund der Größe der Einzelkomponenten oder der Anzahl benötigter paralleler gleicher Komponenten höher ist, als die von einem einzelnen FPGA gebotenen Ressourcen. Die Wirtschaftlichkeit einer



(a) UNNS

(b) NNUS

Abbildung 4.1: Unterscheidung UNNS und NNUS

Umsetzung auf einem großen FPGA oder die Forderung nach einem erweiterbaren System sind ebenfalls Gründe für eine Verschaltung von mehreren FPGAs zu einer starren oder skalierbaren Cluster-Struktur.

Unter der Vorgabe, dass ausreichend Logik-Ressourcen vorhanden sind, ist die Kommunikationsinfrastruktur zwischen den einzelnen Knoten des Systems der in Bezug auf die Verwendbarkeit für einen bestimmten Algorithmus entscheidende Faktor beim Aufbau eines FPGA-Clusters. Im Fall von NNUS ist es die Verbindungsstruktur zwischen den einzelnen Verarbeitungseinheiten eines Knotens. Lose gekoppelte FPGA-Systeme, die sich zum Beispiel einer Ethernet-basierten Infrastruktur bedienen, bieten einerseits in Bezug auf die Skalierbarkeit oftmals eine hohe Flexibilität, da weitere Knoten ohne Veränderung des Verbindungsnetzes mit eingebracht werden können, leiden andererseits aber unter den beiden folgenden wesentlichen Faktoren:

1. Ein für die Kommunikation notwendiges ungünstiges Verhältnis von Nutzdaten zu übertragenden Daten
2. Nichtdefinierte oder nur durch einen gesteigerten Overhead ermöglichte feste Latenz bei der Übertragung von Informationen

Abgrenzung eines FPGA-Clusters von einem MFS

Im Gegensatz zu einem MFS bündelt ein FPGA-Cluster nicht bloß mehrere FPGAs auf einer Leiterkarte, sondern ermöglicht darüber hinaus die flexible Erweiterung des Systems um weitere FPGA-Knoten. Ein Beispiel für ein typisches MFS ist das in Abbildung 4.2 dargestellte DN9000K10 MFS der Firma DINI Group, welches 16 Xilinx Virtex-5-LX330-FPGAs auf einem Basisboard bündelt, siehe Blockschaltbild B.14, und zur Abbildung großer Logiksysteme wie etwa der prototypischen

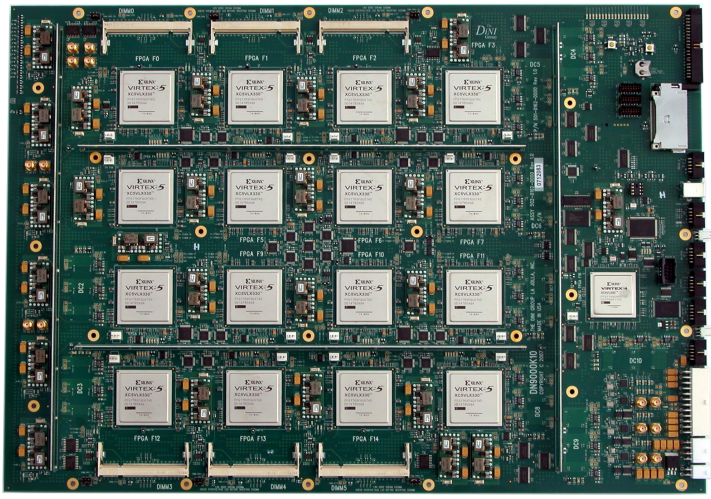


Abbildung 4.2: DN9000k10 MFS [174]

Implementierung von Prozessoren entworfen wurde [174]. Die Kommunikationsinfrastruktur ist fest verdrahtet und unterscheidet sich darüber hinaus individuell in der Anzahl der Verbindungen zwischen den einzelnen FPGAs. Zwar können einzelne FPGAs mit Hilfe von Konnektoren Erweiterungskarten, Daughter-Cards genannt, ansteuern, eine Cluster-Bildung von mehreren DN9000K10 und somit eine Skalierbarkeit des Gesamtsystems ist jedoch nicht vorgesehen.

FPGA-Cluster hingegen zeichnen sich durch die Flexibilität ihrer Kommunikationsinfrastruktur aus, die es einerseits erlaubt, weitere Komponenten in das System einzubringen, und zum anderen eine Ergänzung der auf den Leiterkarten vorhandenen starren Infrastruktur ermöglicht. Bei weiteren Komponenten kann es sich neben zusätzlichen FPGA-Systemen beispielsweise um spezialisierte und standardisierte Schnittstellen handeln.

4.2 FPGA-Cluster-Systeme

In [152] wird aufgezeigt, dass bereits 1985 FPGAs zu Clustern verschaltet wurden um Custom-Computing-Machines zu ermöglichen. Diese Cluster folgten einem dedizierten Anwendungszweck. Da einzelne FPGAs für die Implementierung häufig zu wenig Ressourcen vorhalten, wird das Konzept der Bündelung mehrerer FPGAs auch heutzutage angewandt, wie in dieser Arbeit gezeigt wird. Bei zwei der frühen Cluster-Entwürfe handelt es sich um Splash-2, vorgestellt in [175], und Virtual Computer, bestehend aus 40 Xilinx 4010 als FPGA-Elemente [176]. Hervorzuheben

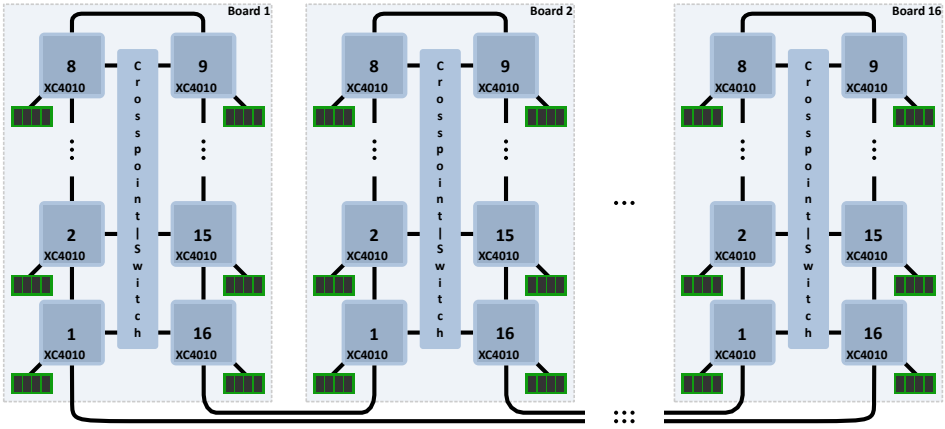


Abbildung 4.3: Architekturübersicht des Splash-2-FPGA-Clusters

ist das Konzept des Splash-2, das bereits verschiedene Architekturkonzepte sowie unterstützende Ansätze für die Anwendungsentwicklung späterer FPGA-Cluster umsetzt.

Splash-2 Splash-2 ist als paralleler Koprozessor innerhalb einer Sun Sparc-Station ausgelegt. Auf einem Interface-Board können bis zu 16 Splash-Array-Boards mit jeweils 16 Processing-Elements (PEs), bestehend aus einem Xilinx XC4010 FPGA und 512 KiB Speicher, verwendet werden. Zusätzlich zu den Anwendungs-FPGAs ist ein weiterer XC4010 FPGA für Steuerungsaufgaben vorhanden. Die 16 FPGAs eines Splash-Array-Boards sind in einer offenen 36-bit-Ring-Topologie sowie über eine ebenfalls 36 bit breite Verbindung zu einem 16×16-Crossbar-Switch untereinander verbunden. Über die beiden Enden des offenen Rings können, wie in Abbildung 4.3 dargestellt, weitere Splash-Array-Boards eingebunden werden, wodurch sich eine lineare systolische Struktur von bis zu 256 FPGAs realisieren lässt [177]. Dem Benutzer steht somit innerhalb eines Boards eine vollvermaschte Struktur zur Verfügung und im FPGA-Cluster eine Ring-Topologie. Die vollvermaschte Struktur erweitert das Konzept des Splash-Attached-Processor-Board, welches aufgrund der Untersuchungen von [178] als systolisches Array eine Ring-Topologie darstellt [175].

Unterstützt wird der Anwender in der Entwurfsphase durch vorhandene Simulationsmodelle sowie eine Schnittstelle für die Programmiersprache *C*. Dies ermöglicht die transparente Interaktion der Applikationen des Hostsystems mit der Simulation, einschließlich der Konfiguration des Crossbar-Switch und initialer Speicherinhalte. Das Modell überprüft außerdem, ob Zeitschrankenvorgaben und Speicherzugriffs-

reihenfolgen korrekt implementiert worden sind. Nach Abbildung auf das Splash-2-System kann ein Laufzeit-Debugger mit Hilfe von Tcl-Skripten die Zustände der FPGAs nach jedem oder mehreren Taktzyklen auslesen. Direkte Speicherzugriffe (engl.: Direct Memory Access, DMA) durch Anwenderprogramme auf die verteilten Speicher des FPGA-Clusters werden ebenso ermöglicht wie die Kontrolle über den Systemtakt durch die C-Bibliothek [179].

Anwendungsfelder des FPGA-Clusters lagen hauptsächlich im Bereich der Bildverarbeitung mit Kantendetektion und Mustererkennung [180, 181, 177], die Schlüsselwortsuche in Texten [182] sowie die Genom- bzw. Protein-Sequenzierung [183] waren jedoch ebenfalls im Fokus.

Aktuelle FPGA-Cluster-Systeme

Im Folgenden werden bestehende FPGA-Cluster-Systeme vorgestellt und anschließend einander gegenübergestellt. Unterschieden werden können die einzelnen Plattformen bereits anhand der bei der Entwicklung maßgeblichen Zielanwendung. So unterscheiden sich Architekturen, die auf ein bestimmtes Anwendungsszenario hin entwickelt wurden, von solchen, die als allgemeine Beschleunigungs- und Evaluationsplattformen entworfen wurden. Letztere Systeme sind häufiger im industriellen Angebot als kommerzielle Einzelkomponenten erhältlich, die sich auch zu Clustern kombinieren lassen. Betrachtet werden dabei nur solche Systeme, die als FPGA-Cluster ausgelegt sind oder, sofern sie aus Einzelkomponenten bestehen, breitbandig miteinander verbunden werden können und somit eine ähnliche enge Kopplung erlauben wie integrierte Komponenten.

Da hohe Datenraten entweder eine hohe Anzahl an relativ langsam getakteten parallelen Übertragungsstrecken voraussetzen, welche nur schwer über Boardgrenzen hinweg realisiert werden können, oder mit Hilfe von seriellen Hochgeschwindigkeitstransmittern darstellbar sind, handelt es sich bei den aufgeführten Systemen aus kombinierbaren Einzelkomponenten um solche, die MGTs zur externen Anbindung zur Verfügung stellen.

Es existieren viele Systemvarianten, die verteilte FPGAs einsetzen, um beispielsweise die Datenerfassung von mehreren Sensorknoten lokal vorzunehmen und anschließend an einen Host zu übertragen, beispielsweise am ATLAS-Teilchendetektor am Large Hadron Collider (LHC) [184]. Obgleich die Anzahl an FPGAs im Gesamtsystem auf diese Weise sehr hoch sein kann, ist es nicht möglich, andere Anwendungsfelder sinnvoll darauf abzubilden, weshalb keine Berücksichtigung innerhalb dieses Kapitels stattfindet.

Die spezifischen Größen wie Speicher und Logikzellen-Äquivalente der einzeln FPGA-Cluster werden im Anschluss an das Kapitel in Tabelle 4.1 aufgeführt. Die Angaben zur Gesamtspeichergröße im System erfolgt hierfür in der in den jeweiligen Veröffentlichungen angegebenen Einheit.

4.2.1 FPGA-Cluster mit dediziertem Anwendungsgebiet

Die im Folgenden vorgestellten FPGA-Cluster-Systeme sind mit einer konkreten Aufgabenstellung im Hintergrund entstanden. Darum ist ihre Struktur auf diesen Einsatzzweck hin optimiert worden. Dennoch werden auch andere Felder genannt, für die der jeweilige FPGA-Cluster eingesetzt werden kann.

CUBE-FPGA-Cluster Der in [6] vorgestellte FPGA-Cluster besteht aus insgesamt 512 Xilinx Spartan-3 (SC3S4000) FPGAs, die auf acht untereinander verbundenen Trägersystemen in einer 8×8-Matrix angeordnet sind. Jeder FPGA wird als eine Prozessoreinheit (PE) behandelt, wobei jeweils 16 PEs zu gerichteten offenen Ringen verbunden sind, wie in Abbildung 4.4 dargestellt.

Jeder der 64 auf einem Trägersystem vorhandenen FPGAs erhält eine eigene ID bestehend aus der x- und y-Koordinate in der Platzierungsmatrix. Zwei unabhängige Bussysteme verbinden die FPGAs eines Systems sowie verschiedene Trägersysteme untereinander. An die Busse sind die einzelnen FPGAs jedoch nicht parallel angeschlossen. Aus diesem Grund ist die in der Veröffentlichung gewählte Bezeichnung Bus irreführend, schließlich können die FPGAs jeweils nur Verbindungen zu ihren horizontalen Nachbarn aufbauen. Auf diesem Weg erfolgt eine systemweite Taktverteilung, welche parallel zu den Datenleitungen geführt ist und aufgrund des Aufbaus eine für den angestrebten Betrieb mit 100 MHz ausreichend geringe Leitungsverzögerung aufweist. Durch die jeweilige Aufbereitung des Taktes über eine DCM erhöhen sich der Jitter und die Verzögerung sukzessive. Nach Einschalten des Systems benötigen die DLLs der DCMs ein für mehrere Taktzyklen stabil anliegendes Taktsignal. Um dieses Verhalten sicherzustellen, wird das LOCKED-Signal einer DCM als Reset-Signal der im benachbarten FPGA befindlichen DCM verwendet. Dieser Aufbau bedeutet eine zusätzliche Verzögerung, bis alle 64 FPGAs betriebsbereit sind. Zusätzlich zu der Taktzuführung über den benachbarten FPGA erhält jeder Baustein einer Zeile den gleichen 25-MHz-Takt, so dass sich ein lokal synchrones jedoch global asynchrone Systeme (engl.: Globally Asynchronous / Locally Synchronous, GALS) ergibt. Die Konfiguration der FPGAs erfolgt zeilenweise parallel über dedizierte CPLDs. Die FPGAs einer Zeile sind in zwei Gruppen (gerade/ungerade Position) unterteilt. Jedes FPGA einer Gruppe kann parallel mit dem gleichen Bitstrom konfiguriert werden. Ein individueller Bitstrom für jedes FPGA kann über die industrieübliche Schnittstelle der Joint-Test-Action-Group (JTAG) geschrieben werden. Für die Kommunikation mit externen Komponenten sowie die Kopplung mehrerer Trägersysteme teilen sich zum einen je zwei benachbarte FPGAs einen 32 bit breiten Konnektor, zum anderen sind die beiden Bussysteme auf insgesamt drei IDE-Anschlüsse geführt. Über die Xilinx Herstellerwerkzeuge für den Entwurf hinaus gibt es keine weitere Software-Unterstützung. Es werden lediglich Templates für die Buslogik bereitgestellt. In der Veröffentlichung angeführter Verwendungszweck des Clusters ist die Suche nach RC4-Schlüsseln.

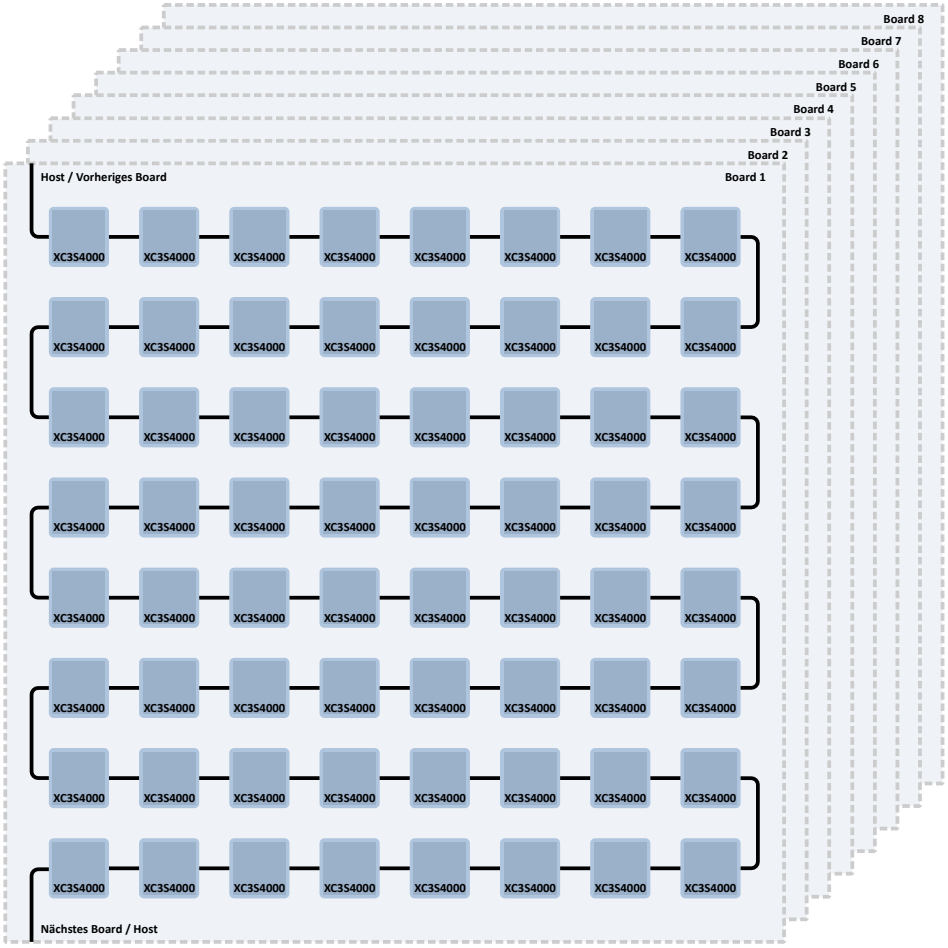


Abbildung 4.4: Architekturübersicht des CUBE-FPGA-Clusters

COPACOBANA – Cost Optimized PArallel COde Breaker Der COPACOBANA-FPGA-Cluster ist bereits in der zweiten Entwicklungsgeneration verfügbar. Im Folgenden werden das Architekturkonzept anhand des ersten Aufbaus und im weiteren Verlauf die Änderungen des Generationswechsels erläutert.

Die in [185] vorgestellte Architektur besteht aus insgesamt 120 Xilinx Spartan3-1000, die sich auf 20 FPGA-Module verteilen. Untereinander sind die FPGA-Module über einen 64 bit breiten Datenbus mit zugehörigem 16-bit-Adressbus verbunden. Die beiden Bussysteme werden auf der Backplane umgesetzt, wie Abbildung 4.5

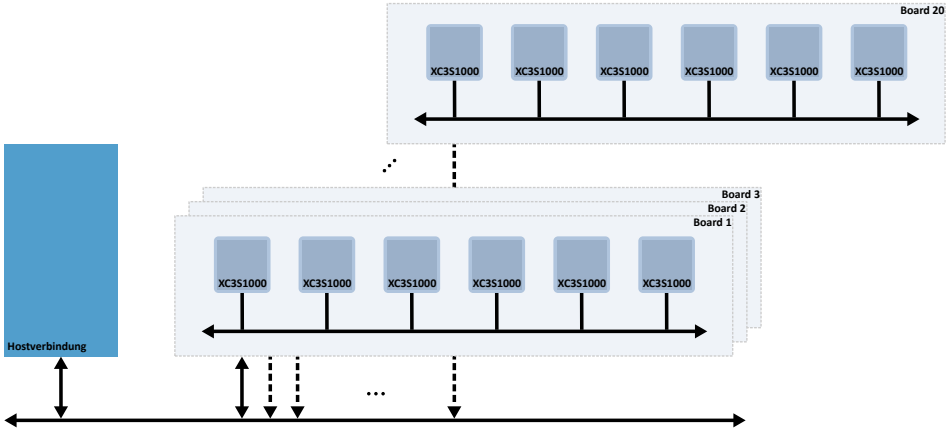


Abbildung 4.5: Architekturübersicht des COPACOBANA-FPGA-Clusters

schematisch aufzeigt. Der Host-PC ist über eine Controller-Card per USB oder Gigabit-Ethernet ebenfalls an die Backplane angeschlossen und übernimmt die Ablaufsteuerung und Ergebnisauswertung. Das Bussystem ist darauf ausgelegt, dass ein einzelner Master die FPGAs abfragt oder ein statisches Schedulingverfahren die Zugriffsreihenfolge auf das Übertragungsmedium regelt. Die Übertragungsfrequenz für das vollbestückte System ist auf 33 MHz beschränkt. Eine Interrupt-Behandlung ist nicht vorgesehen. Ebenfalls über die Backplane werden zwei Taktsignale systemsynchron verteilt. Jedes der FPGA-Module verfügt über einen systemweit einzigartigen Identifier. Die Kommunikationsarchitektur des FPGA-Clusters entspricht der Annahme, dass implementierte Applikationen sich stark parallelisieren lassen und der Datenaustausch zwischen den parallel ablaufenden Berechnungen minimal ist. Eine weitere Prämisse betrifft den lokalen Speicherbedarf der Implementierungen. Dieser muss so gering sein, dass er aus den in den FPGAs vorhandenen BRAM gedeckt wird [186].

Die Konfiguration der FPGAs erfolgt durch die Controller-Card in slave-parallel-mode, wobei sämtliche FPGAs mit der gleichen Konfiguration beschrieben werden.

Der vorgestellte FPGA-Cluster wurde für das Anwendungsszenario des Brechens von Verschlüsselungen entworfen; entsprechend sind die veröffentlichten Implementierungen größtenteils in diesem Bereich angesiedelt [67, 187, 188, 189]. Das System wurde außerdem für die Beschleunigung von Primfaktorzerlegungen eingesetzt [190], da diese ein essentieller Bestandteil vieler Verfahren zur Entschlüsselung sind, so auch im Bereich der Verschlüsselung mit Hilfe von elliptischen Kurven. Zur weiteren Beschleunigung des Verfahrens wurden die Spartan3-Module durch 16 Virtex-4-SX35-FPGA-Module ersetzt. Dies erlaubt die Nutzung dedizierter

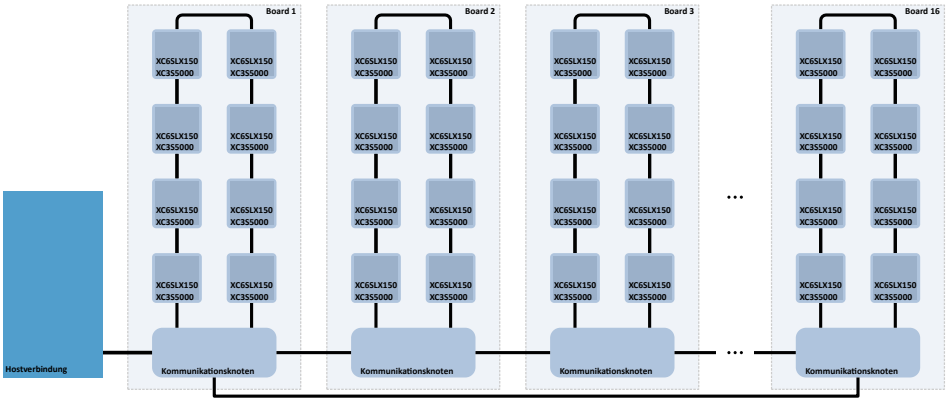


Abbildung 4.6: Architekturübersicht des RIVYERA-S3-/-S6-FPGA-Clusters

DSP-Blöcke für die Berechnung, welche im Spartan3 nicht vorhanden sind. Es existiert ebenfalls eine Implementierung des COPACOBANA-FPGA-Clusters, die vollständig auf Spartan3-Module verzichtet und stattdessen 128 Virtex-4-FPGAs verwendet [191].

RIVYERA – Redesign of the Increadibly Versatile Yet Energy-efficient, Reconfigurable Architecture Als Nachfolgesystem zum COPACOBANA-FPGA-Cluster wurde beim Entwurf des RIVYERA-Cluster hoher Wert auf Modularität und somit Zukunftssicherheit durch den einfachen Austausch von Komponenten gelegt. Dank dieses Aufbaus ist es möglich, die einzelnen FPGA-Boards als Kernkomponenten auszutauschen oder sogar kombiniert einzusetzen. Zur Verfügung stehen dabei Module mit 8 Spartan-3 5000, 8 Spartan-6 LX150, 4 Virtex-6 LX550T und 1 Virtex-7 2000T [192, 193, 194, 195]. Der Aufbau der FPGA-Module gestaltet sich wie in Abbildung 4.6 dargestellt.

Die Kommunikationsstruktur der Systeme mit Spartan-3-, Spartan-6- und Virtex-6-FPGAs entspricht dem Ansatz des CUBE-FPGA-Clusters aus Abschnitt 4.2.1. Die einzelnen FPGAs eines Moduls sind in einer Ringstruktur mit einem Kommunikationsknoten in Form eines weiteren FPGA verbunden. Die Kommunikationsknoten der FPGA-Module können wiederum in einer Ringstruktur über eine Backplane Daten miteinander austauschen, wobei insgesamt 16 Modulplätze vorhanden sind. Ein FPGA-Modul verfügt außerdem über eine PCIe-Verbindung zum Hostsystem, welches auf Standardhardware Linux als Betriebssystem nutzt und mit im Servergehäuse untergebracht ist. Der Aufbau des FPGA-Clusters mit Virtex-7-Modulen, welche im Anhang unter B.6 abgebildet sind, unterscheidet sich von den Modulen mit FPGAs älterer Generationen insofern, als dass nur ein FPGA pro Modul vorhan-

den ist und insgesamt maximal 4 Module pro Backplane genutzt werden können. Darüber hinaus verfügen die Virtex-7-Module über vier Speicherslots am FPGA, die bis zu 1024 MB DDR3-Speicher je Steckplatz verwalten können.

Der Anwender erhält für den Entwurf Unterstützung in Form von APIs, welche die Kommunikation mit der auf dem Host laufenden Software kapseln und neben Zugriffen auf einzelne FPGAs auch Broadcasts erlauben. Die Konfiguration des FPGAs zur Anbindung an die Backplane wird ebenfalls durch eine API abstrahiert [196].

Anwendungsgebiete des FPGA-Clusters sind Untersuchungen zur linearen Verschlüsselungsanalyse. Die Implementierung der aufgezeigten Methoden erfolgt bewusst ohne Unterstützung durch vorgefertigte IP-Cores des Xilinx CORE-Generators, da die Integration in das Gesamtsystem unzureichende Ergebnisse in Bezug auf den Ressourcenverbrauch und die erzielbare maximale Taktrate hervorbringt [197]. Des Weiteren können Wörterbuchangriffe auf die weit verbreitete Verschlüsselungssoftware *TrueCrypt* durchgeführt werden [66]. Im Bereich der Bioinformatik werden durch den Einsatz des RIVYERA-FPGA-Cluster mit Spartan-6-Modulen Berechnungen zur Sequenzanalyse mit Hilfe des BLASTp-Algorithmus durchgeführt [196]. Weitere Beispiele für den Beweis der Leistungsfähigkeit im Bereich der Bioinformatik werden in [198] aufgezeigt. Implementierungen von Needleman-Wunsch- und Smith-Waterman-Algorithmen weisen einen bis 73-fachen Durchsatz gegenüber Berechnungen auf einem Intel Xeon W3530 System auf. Eine Beschleunigung um den Faktor 12 konnte gegenüber GPGPU-Hardware in Form eines nVidia Tesla M2090 erzielt werden. Hinzu kommt eine Energieersparnis von über 90 % gegenüber Berechnungen auf PC-Clustern [198].

HAPS-64 Der Synopsys HAPS-64-FPGA-Cluster nutzt vier Virtex-6 (XC6VLX760), welche untereinander vollvermascht differentiell verbunden sind. Der Aufbau der Kommunikationsinfrastruktur ist jedoch nicht homogen ausgelegt und variiert in der Anzahl der vorhandenen Verbindungen, wobei von einem FPGA ausgehend immer mindestens 29 differentielle Leiterpaare zur Verfügung stehen. Geteilte Verbinder, die entweder zur Kommunikation von zwei FPGAs eines Basisboards oder zur Anbindung externer Komponenten auf Steckverbinder geführt werden, erweitern die Struktur, wie in Abbildung 4.7 als Verbindung A1..A6,..,D1..D6 dargestellt. Um das Basisboard mit einem Hostsystem zu koppeln, können über optionale Aufsteckmodule die Kommunikationsschnittstellen 4x PCIe, USB 2.0 und Gigabit-Ethernet realisiert werden. Insgesamt ist das System auf eine hohe Modularität hin ausgelegt worden. Es können verschiedene Komponenten wie DDR-2-Speicher und Analog- bzw. Digitalschnittstellen hinzugefügt werden. Zudem erlaubt der symmetrische Aufbau der externen Verbinder eine Kopplung mehrerer Basisboards in jede Raumrichtung.

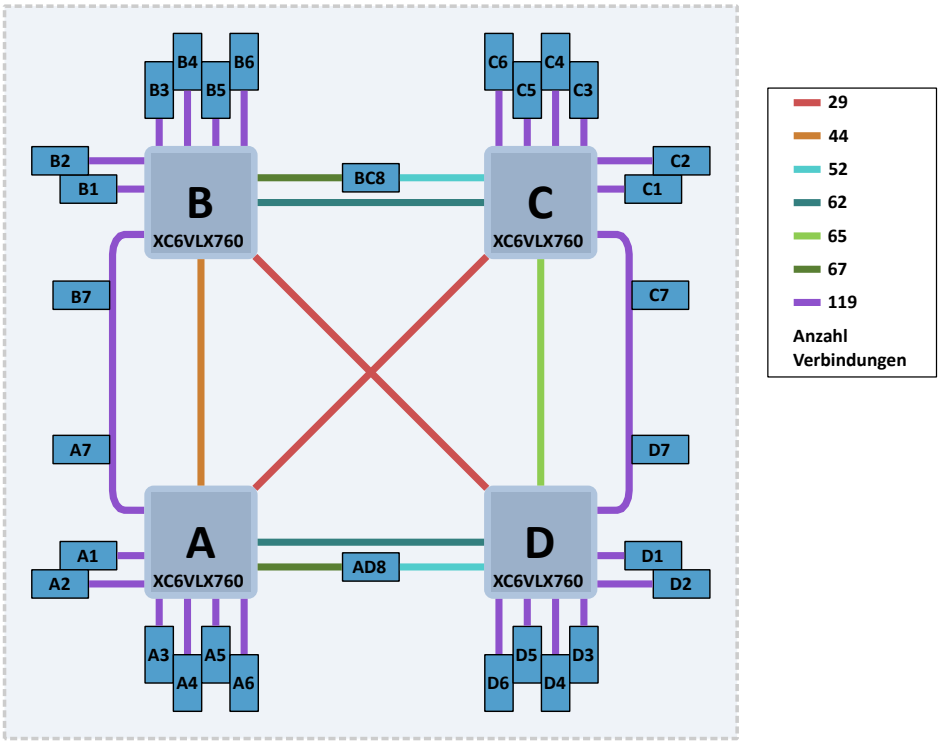


Abbildung 4.7: Architekturübersicht des HAPS-64-FPGA-Clusters

Die Konfiguration der FPGAs kann individuell erfolgen. Zur Verfügung stehen JTAG, MicroSD-Card und ein proprietärer Universal-Multi-Resource-Bus (UMR-Bus) über externe Kommunikationsmodule. Einschränkungen erfährt das modulare Konzept bei der Kopplung mehrerer Basisboards. So können externe DDR-2-Aufsteckmodule und eine voll verbundene Struktur nicht gleichzeitig realisiert werden.

Unterstützung erhält der Anwender durch die ebenfalls von Synopsys zur Verfügung gestellten Software-Lösungen und Virtualisierungsprodukte sowie durch Design-Ware-IP-Blöcke speziell für das prototypische Realisieren von ASICs [199].

FPGA-based Custom Computing Board (FCCB) Bei dem in [200] vorgestellten FCCB handelt es sich um einen skalierbaren Altera APEX-20KE-Cluster, der vor dem Hintergrund der medizinischen Bildverarbeitung entworfen wurde. Der darauf abgebildete Concentration-Index-Bildfilter wird eingesetzt, um anhand von Röntgenbildern Magenkrebs zu diagnostizieren. Wie in Abbildung 4.8 dargestellt,

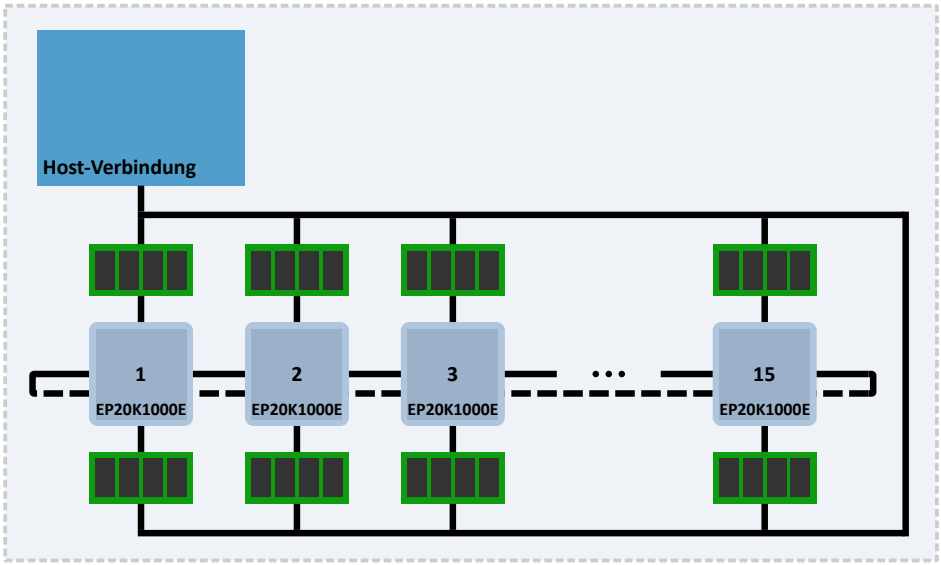


Abbildung 4.8: Architekturübersicht des FCCB-FPGA-Clusters

werden jedem der FPGAs zwei dual-port-SRAM-Bausteine zugeordnet. Die so aufgebauten Processing-Units (PU) sind zum einen untereinander als geschlossener Ring verbunden, zum anderen konnektiert ein Datenbus alle Speicher mit einem Schnittstellen-FPGA für die Hostkommunikation. Sämtliche Kommunikationspfade sind homogen 96 bit breit ausgelegt.

Die skalierbare Verarbeitung der Bilddaten wird durch die horizontale Aufteilung der Bilddaten auf die einzelnen PUs erreicht. Das realisierbare Maximum an PUs beträgt 15, ohne dass ein konkreter Grund für das Maximum angegeben wird. Es ist jedoch davon auszugehen, dass die Leitungslänge des Datenbusses den limitierenden Faktor darstellt.

Catapult Mit dem Ziel ein Machine-Learning-Netzwerk aus Servern aufzubauen, mit dessen Hilfe Machine-Learning- und Deep-Learning-Algorithmen umgesetzt werden können, ist der Catapult-Cluster entstanden. Die Firma Microsoft implementiert dabei ein NNUS aus 48 Intel Xenon-Servern, die um jeweils einen Intel (ehemals Altera) Stratix V GS D5 ergänzt werden. Verbunden sind die FPGAs mit dem Server über eine PCIe 3.0 8x Schnittstelle. Untereinander bilden jeweils 48 FPGAs einen in Abbildung 4.9 dargestellten 6×8-Torus mit jeweils zwei MGT-Verbindungen pro Senderichtung [201]. Die auf diese Weise erzielbare Datenrate liegt bei 20 Gbit/s im Vollduplex-Verfahren [202]. Jedem FPGA stehen insgesamt 8 Gbit DDR3-1333

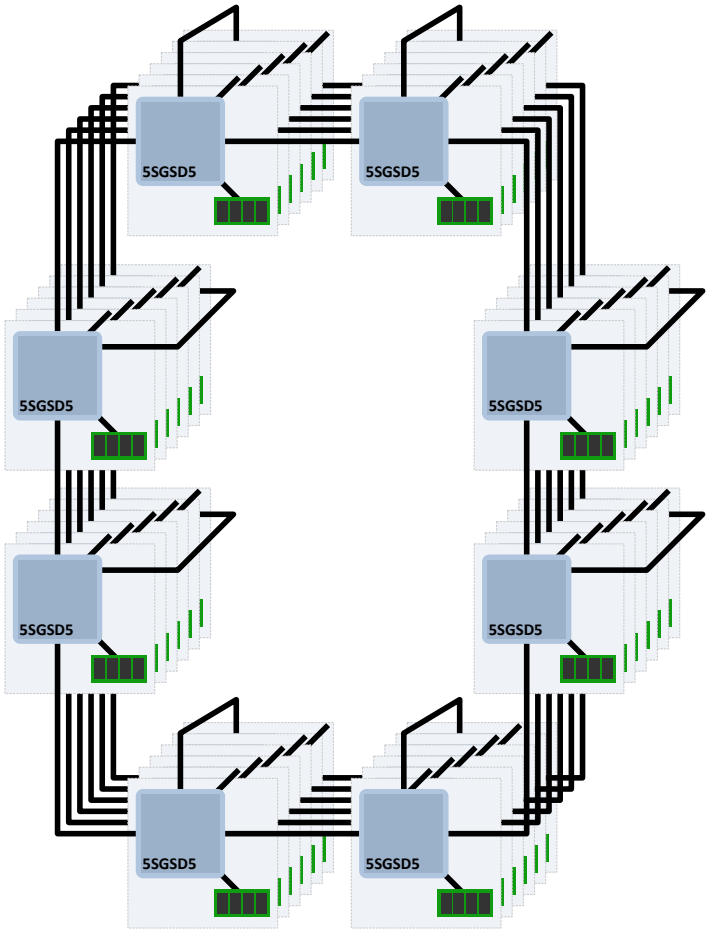


Abbildung 4.9: Architekturübersicht des Catapult-FPGA-Clusters

Speicher verteilt auf zwei SODIMM-Sockeln zur Verfügung. Derzeit findet eine Umstellung des eingesetzten FPGA-Typs von Intel Stratix V GS D5 auf Intel Arria 10 statt [95]. Auf dem Prozessorsystem wird das Betriebssystem Microsoft Windows Server eingesetzt.

Zwar sind die Systeme theoretisch beliebig skalierbar, jedoch füllt ein Aufbau mit 48 Servern ein halbes Standard-Server-Rack und bildet somit eine verbindungstechnisch abgeschlossene Einheit. Auf Protokollebene der Inter-FPGA-Kommunikation wird ein an UDP angelehntes Lightweight-Transport-Layer-Protokoll eingesetzt.

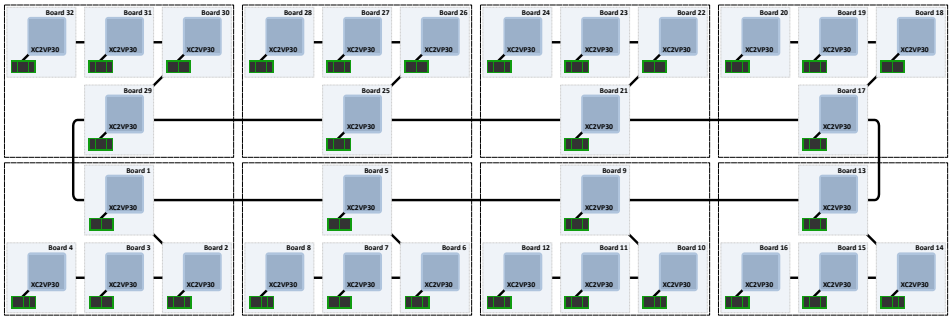


Abbildung 4.10: Architekturübersicht des SMILE-FPGA-Clusters

In den Microsoft Rechenzentren werden die Catapult-FPGA-Cluster unter anderem für eine beschleunigte Suche der Suchmaschine Bing verwendet. Darüber hinaus wird in [201] die AES-Ver- und -Entschlüsselung von 40 Gbit/s Netzwerkströmen aufgezeigt.

4.2.2 Allgemeine FPGA-Cluster

Im Gegensatz zu den im vorangegangenen Abschnitt aufgeführten anwendungsspezifischen FPGA-Clustern sind allgemeine FPGA-Cluster unter dem Vorsatz entwickelt worden, dass sie verschiedene Anwendungsgebiete durch ihre Struktur unterstützen.

SMILE – Scientific Parallel Multiprocessing based on Low Cost Reconfigurable Hardware Das in [203] vorgestellte System verfolgt das Ziel, mit Hilfe kostengünstiger COTS-FPGA-Boards einen für eine Vielzahl an Anwendungen verwendbaren FPGA-Cluster aufzubauen. Entsprechend besteht der Verbund aus 32 Digilent XUP-V2Pro-Boards [204] mit jeweils einem Virtex-2 Pro XC2VP30. Verbunden werden die einzelnen Plattformen mit Hilfe der vorhandenen SATA-Anschlüsse, welche ihrerseits auf die MGTs des Virtex-2 Pro geführt werden. Die Topologie gestaltet sich wie in Abbildung 4.10 dargestellt. Auf Protokollebene abstrahiert Xilinx Aurora die zugrunde liegende Hardware. Mit Hilfe einer Schnittstellenbrücke erscheinen die Verbindungen als Ethernet-Schnittstelle im Linux-Betriebssystem, welches auf den integrierten PowerPC-Kernen umgesetzt worden ist. Zusätzlich verfügen die drei Schnittstellen über eine in der FPGA-Logik implementierte Routing-Architektur, um netzwerkartige Topologien abzubilden. Für den Datenaustausch wird MPI verwendet, wodurch die Portierbarkeit der Applikationen gesichert ist.

Bei der Applikationsentwicklung wird der Anwender durch ein System-C-Modell unterstützt [205]. Ziel ist die Beschleunigung einer MPI-Anwendung durch

Auslagerung kritischer Bereiche des Algorithmus in die FPGA-Ressourcen. Die ausgelagerten Module werden über den PLB an den PowerPC angebunden und sind vom Benutzer mit Hilfe der Xilinx Tool-Unterstützung zu entwickeln.

Die demonstrierte Anwendung ist dem Bereich der Bildanalyse und dem Auffinden extrahierter Merkmale innerhalb einer Datenbank (engl.: Content Based Image Retrieval, CBIR) entnommen. Zwei verschieden implementierte Umsetzungen für die Monte-Carlo-Simulation von Derivatprodukten der Finanzwirtschaft werden in [149] aufgezeigt.

ScalableCore Der aus Spartan-6-FPGAs (XC6SLX16-2) aufgebaute Cluster in [206] zielt darauf ab, dass jeder FPGA in einer zweidimensionalen Ebene mit seinen beiden horizontalen und vertikalen Nachbarn kommunizieren kann. Im System werden dafür einzelne Boards mit je einem Spartan-6 als Logikeinheit über Steckverbinder miteinander verknüpft. Das vorgestellte System stellt somit eine Weiterentwicklung des Spartan-3-FPGA-Clusters aus [207] dar. Die Architektur der zugrundeliegenden Kommunikationsinfrastruktur ist erhalten geblieben. Ein schematischer Aufbau wird in Abbildung 4.11 dargestellt. Neben den ScalableCore-Units können Speichererweiterungsmodule an den Randbereichen des FPGA-Clusters eingesetzt werden, welche den jedem FPGA zur Verfügung stehenden 512 kB SRAM durch DRAM ersetzen.

Eine Besonderheit des Systems ist das Fehlen globaler Verbindungen jeglicher Art. Die Taktverteilung erfolgt lediglich lokal je ScalableCore-Unit, wodurch der FPGA-Cluster beliebig skalieren lässt. Der Nachteil dieser Architektur besteht darin, dass es nicht möglich ist, zur Vermeidung aufwendiger Synchronisationsmechanismen kleinere benachbarte Gruppen von FPGAs synchron zueinander zu betreiben.

Die Anbindung an ein Hostsystem erfolgt über einen beliebigen Randknoten des Clusters und nicht individuell für jede ScalableCore-Unit. Da es sich bei dem Anwendungsszenario um die Simulation eines NoC-basierten Many-Core-Prozessorsystems handelt, erfolgt die Beschreibung der FPGAs über ein auf jeder Einheit vorhandenes ROM, welches vor Inbetriebnahme der Cluster-Architektur per JTAG beschrieben wird.

BEE3 Der BEE3-FPGA-Cluster, der in [208] vorgestellt wird, besteht aus einem Basissystem mit vier Virtex-5-FPGAs (XC5VFX100T), wobei auch andere vom Leiterplattenanschlussbild (engl.: footprint) kompatible Varianten des Virtex-5 bestückt werden können. Der eigentlichen FPGA-Trägerleiterkarte steht eine zweite Leiterkarte zur Seite, die für Kontrollaufgaben wie die Konfiguration des Systems oder die persistente Speicherung von Konfigurationsdaten zuständig ist. Für eine dauerhafte Speicherung von Daten verfügt das Kontroll-PCB über eine Compact-Flash-Karte, die von einem Xilinx System-ACE angesteuert wird. Darüber hinaus existieren vier SD-Card-Einschübe, die von den FPGAs direkt über SPI angesprochen werden

Abbildung 4.11: Architekturübersicht eines ScalableCore-FPGA-Clusters

können. Jedes der vier FPGAs kann bis zu vier DDR2 DRAM DIMMS mit jeweils bis zu 4 GB Speicher ansteuern. Untereinander sind die FPGAs, wie in Abbildung 4.12 dargestellt, in einer Ringstruktur mit einem 72 bit breiten Interface verbunden, für welches ein DDR2-500-IP-Core bereitgestellt wird. 40 differentielle Signale je FPGA werden auf jeweils einen Samtec QSH-Verbinder geführt, so dass es möglich ist, über eine zusätzliche externe Verbindung eine vollvermaschte Topologie aufzubauen oder andere Komponenten einzubinden. Die MGTs werden jeweils auf zwei Konnektoren und einen PCIe-8x-Verbinder geführt. Darüber hinaus verfügt jeder

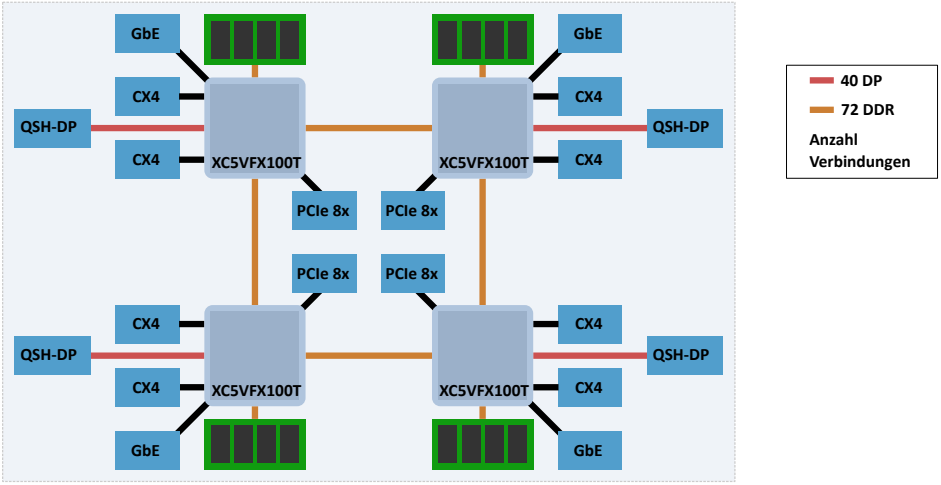


Abbildung 4.12: Architekturübersicht eines BEE3-FPGA-Clusters

FPGA über eine Hardmacro-GbE-Schnittstelle, die mit einem externen PHY-Baustein verbunden ist.

Neben den von Xilinx zur Verfügung gestellten Werkzeugen zur Entwicklung wie Macros für PCIe, 10-Gb-XAUI, 1-Gb-Ethernet-MAC kann zum Beispiel der IP-Core für die DDR2-Speicher und die Kommunikationsschnittstellen über Microsoft bezogen werden. Zusätzlich gibt es eine enge Einbindung von Matlab Simulink bzw. System-Generator-Elementen um den Entwurfsablauf auf eine abstrakte Ebene abseits von Hardware-Beschreibungssprachen zu heben und somit zu vereinfachen und zu beschleunigen. Bereits in den Veröffentlichungen [209] und [210] wird darüber hinaus die Möglichkeit beschrieben, parallel dazu Mapping-Informationen für eine ASIC- und FPGA-Implementierung zu erzeugen. Dies stellt sicher, dass das auf den FPGAs prototypisch abgebildete System sowohl im Takt- als auch im bit-level-Verhalten äquivalent ist. Darüber hinaus lassen sich durch die Nutzung vorgefertigter Logikblöcke bereits sehr früh genaue Aussagen über den Ressourcenbedarf der endgültigen Implementierung treffen.

Anwendungsfelder für das BEE3-System sind neben der Lösungsfindung bei Erfüllbarkeitsproblemen (engl.: Satisfiability, SAT), Öl- und Gas-Explorationsdatenverarbeitung sowie Finanzkalkulationen. Im Bereich des Prototyping werden Felder von neuartigen Netzwerkdesigns genannt [208]. Innerhalb des Research-Accelerator-for-Multiple-Processors-Konsortiums (RAMP) ist der BEE3-FPGA-Cluster ebenfalls als Forschungs- und Entwicklungssystem zusammen mit der Vorgängerarchitektur BEE2 aufgestellt worden [211], wobei die abgebildeten Applikationen MPI-basierte Benchmarks wie der NAS-Parallel-Benchmark oder die Simulation

von TCP/IP-Strukturen für Suchanwendungen in großen Netzwerken sind (*RAMP blue*). Eine weitere Säule innerhalb von RAMP (*RAMP red* und *RAMP white*) bildet die Simulation verschiedener Modelle von Cache-Kohärenz in Speichersystemen [212]. In [213] wird ein BEE3 zur Emulation eines Datenzentrum-Netzwerks mit 10 000 Knoten eingesetzt, auf denen tatsächliche Applikationen gestartet werden können. Simulationen für Interaction-Networks, wie sie zum Beispiel im Bereich des DNA-Sequencing benötigt werden, in [214] am Beispiel des Signalnetzes der menschlichen T-Zelle aufgezeigt, bilden ebenfalls ein Anwendungsgebiet. Bei einer weiteren Beispielimplementierung aus dem Bereich der Bioinformatik handelt es sich um den Sequenzvergleich mit Hilfe des CAST-Algorithmus, wie in [215] aufgezeigt. Bereits auf dem Vorgängersystem BEE2 begonnene Implementierungen im Bereich der Sequenzanalyse sind auf dem BEE3 weitergeführt worden [216]. Darüber hinaus ist Fourier-Domain-Optical-Coherence-Tomography ein Anwendungsfeld der Bioinformatik, welches in [25] sowohl auf GPGPUs als auch auf der BEE3-Plattform implementiert wird. Im Vergleich zeigt die FPGA-Umsetzung einen mehr als doppelt so hohen Datendurchsatz gegenüber der Berechnung auf GPGPUs. Als Nachweis für die praktikable Realisierung einer verteilten Netzwerkanwendung wird das System in [217] eingesetzt. In [218] dient der BEE3-FPGA-Cluster als Emulator für skalierbare Multicore-Systeme mit dem Open-Source Softcore-Prozessor Plasma eingesetzt. Mit Hilfe von Software Transactional Memory Benchmarks wird die Beschleunigung gegenüber Software-basierten Emulatoren nachgewiesen.

BEE4 Der in [219] vorgestellte BEE4-FPGA-Cluster der Firma BEEcube und Nachfolger des BEE3 erweitert das Konzept des Vorgängers. Die vier Virtex-6-FPGAs sind untereinander vollvermascht verbunden, wie in Abbildung 4.13 dargestellt. Die Ringstruktur verfügt dabei über 84 Verbindungen, während die diagonalen Verbindungen 64 bit breit realisiert worden sind. Zusätzlich werden FPGAs über je 4 GTX-Transceiver-Paare miteinander verbunden. Jedem FPGA steht ein PCIe 2.0 8x-Steckplatz zur Verfügung, über den Module über die MGTs eingebunden werden können. Zusätzlich können über zwei QSFP+ mehrere Systeme gekoppelt oder die bereits vorhandene Kommunikationsinfrastruktur erweitert werden.

Zur Vereinfachung des Entwurfsablaufs kann mit Hilfe des BEEcube Platform Studio (BPS) auf eine Vielzahl von IP-Cores für Matlab Simulink bzw. System-Generator zurückgegriffen werden. Im Wesentlichen kapseln die IP-Cores Hardware-Schnittstellen wie MGTs, Speichercontroller oder Systemkonfiguration und erlauben im Zusammenspiel mit selbst entworfenen Komponenten einen schnellen Aufbau eines Systems. Neben den FPGA-Systemen von Beecube werden die Xilinx Evaluationboards ML605, ML505, ML507 und XUP-V5 unterstützt. Das Linux-basierte NectarOS virtualisiert Hardware-Ressourcen und gewährleistet so einen vereinfachten Zugriff aus Software-Anwendungen [220].

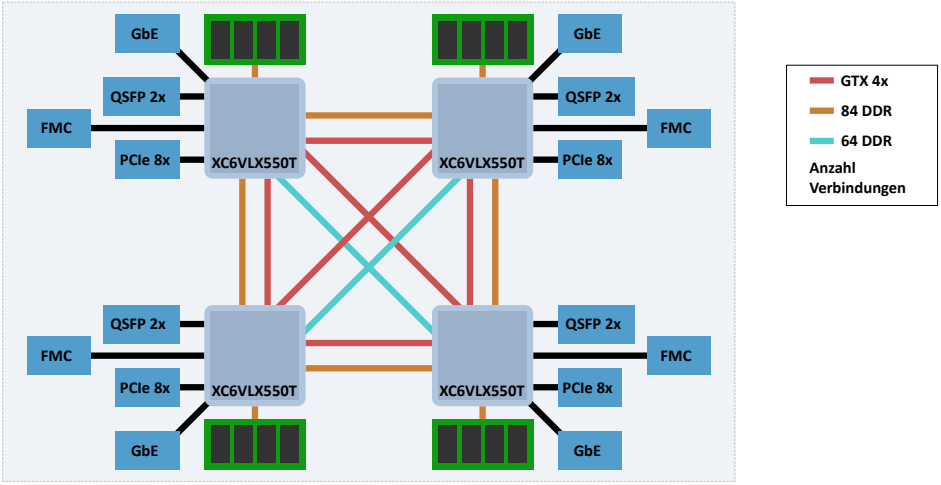


Abbildung 4.13: Architekturübersicht eines BEE4-FPGA-Clusters

Neben Software-Defined-Radio und der Vorverarbeitung für drahtlose Übertragungsstandards werden in [221] Video Processing, Computer Architecture Research und die Beschleunigung von Simulationen verschiedener Anwendungsgebiete durch Auslagerung berechnungsintensiver Aufgaben in Hardware aufgeführt.

Maxwell Die in [222] vorgestellte FPGA-Cluster-Lösung Maxwell der FPGA High Performance Computing Alliance (FHPCA) bestehend aus den Firmen Algotronix, Alpha Data, EPCC, Institute for System Level Integration, Nallatech und Xilinx versucht, FPGA-basierte Rechenknoten in einer Mehrzweck-HPC-Umgebung zu platzieren. Aus diesem Grund wird der Aufwand, den eine Beschleunigungslösung mit FPGAs für den Anwender bedeutet, mit dem Aufwand verglichen, Softwareoptimierungen in Assembler vorzunehmen. Entsprechend wird dazu geraten, die geplanten Anwendungsfälle in kleine Kernel tasks aufzuteilen, die durch einen FPGA beschleunigt werden können. Dabei ist der Kommunikationsbedarf zwischen dem Hauptspeicher des Kernel-Systems und den einzelnen lokalen Speichern der FPGAs sowie die Kommunikation der FPGAs untereinander zu minimieren.

Die Kommunikationsinfrastruktur des Clusters wird als 2D-Torus mit Hilfe der MGT-Transceiver realisiert. Entsprechend sieht die Topologie wie in Abbildung 4.14 dargestellt aus. Die Datenrate pro Lane wird mit über 2,5 Gbit/s angegeben. Kernelement sind die Karten der beiden Hersteller Nallatech (H101-PCIXM mit Virtex-4 LX100) und Alpha Data (ADM-XRC-4FX mit Virtex-4 FX100), um herstellerneutrale Programmiermodelle zu ermöglichen. Zusammengefasst wird das System in meh-

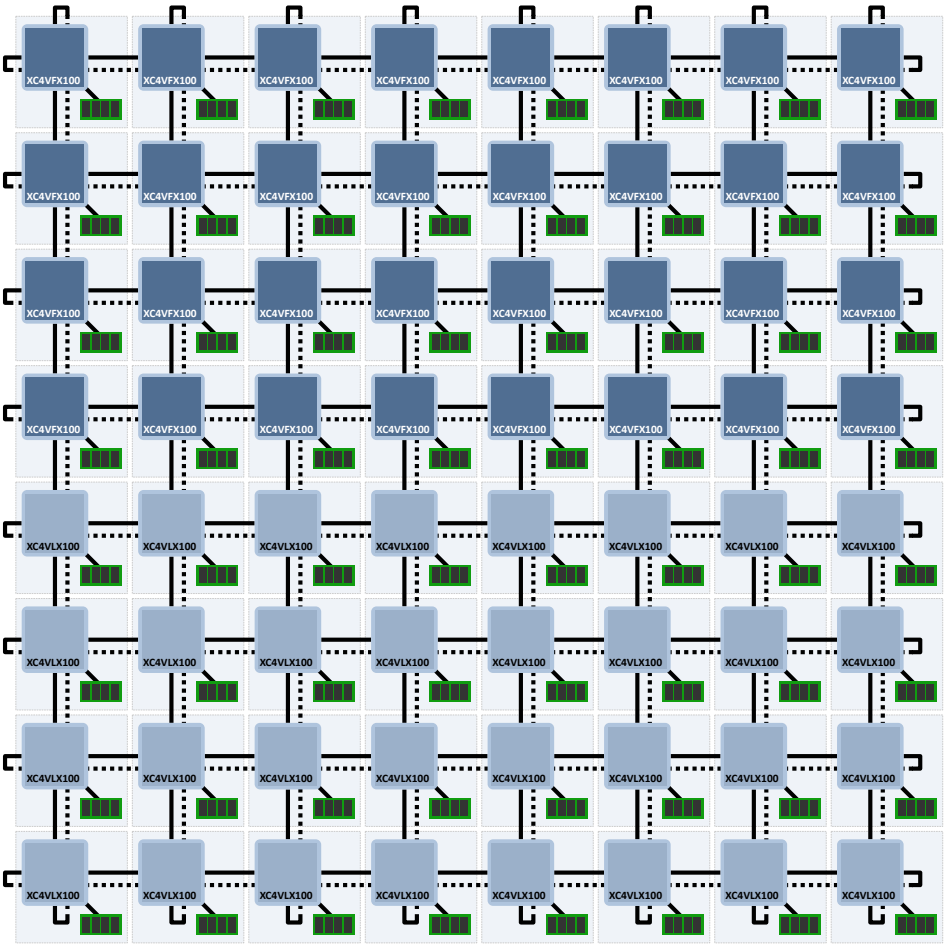


Abbildung 4.14: Architekturübersicht des Maxwell-FPGA-Clusters

reren IBM Blade Centern, wobei jedes Blade neben einem 2,8 GHz Intel Xeon mit 1 GB Hauptspeicher zwei der aufgeführten Karten beherbergt. Insgesamt besteht das System aus jeweils 32 FPGAs auf den PMC-Modulen der beiden Hersteller [223].

Mit dem Ziel, auf Anwendungsebene transparent zu Standard-HPC-Architekturen zu erscheinen, werden CentOS als Linux-Distribution und MPI zur Inter-Prozesskommunikation eingesetzt.

Anwendungsgebiete sind Monte-Carlo-Simulationen von Finanzprodukten [148], 3D-Modell-Berechnungen für Gesichtsbilder und -videos sowie Explorationsuntersuchungen für Öl- und Gasfelderkundungen [222]. In [224] werden darüber hinaus Berechnungen im Bereich der Bioinformatik durchgeführt.

Spirit Der im Jahr 2006 durch die University of North Carolina im Rahmen des Reconfigurable-Computing-Cluster-Projekts vorgestellte FPGA-Cluster Spirit besteht aus 64 FPGA-Knoten basierend auf Xilinx ML410 Development-Boards, die über eine konfigurierbare Kommunikationsinfrastruktur miteinander verbunden sind. Die auf dem Board vorhandenen Virtex-4-FPGAs XC4VFX60 verfügen neben den 8-Lane-MGT-Verbindungen über Standard-Schnittstellen wie Gigabit-Ethernet, Serial ATA sowie zwei PCIe-Karten-Slots. Das für die MGT-Verbindungen eingesetzte AIREN-Netzwerkprotokoll verknüpft auf Basis des Xilinx Aurora-Protokolls die Boards in einem 4-Ebenen-3D-Würfel [225]. Die mit 2 m Kabellänge erzielte Übertragungsrate beträgt 3,2 Gbit/s [226]. Die Konfiguration des FPGAs erfolgt im regulären Betrieb über die durch den eingesetzten Xilinx System-ACE eingebundene Compact-Flash-Karte. Der schematische Aufbau des Clusters ist in Abbildung 4.15 dargestellt.

Von den hier vorgestellten Systemen ist Spirit der einzige FPGA-Cluster, bei dem bereits beim Entwurfsprozess die Möglichkeit geschaffen wurde, nicht nur die Anwendung eines Benutzers auf die vorhandenen FPGAs zu verteilen, sondern mehreren Benutzern parallel die Nutzung freier Ressourcen zu gewährleisten. Der FPGA-Cluster ist als Ethernet-Komponente für mehrere Anwender über das lokale Netzwerk verfügbar. Durch die als zweistufige Client-Server-Anwendung entwickelte FPGA Session Control (FSC) Software wird das Hochladen eines Bitstroms auf die Compact-Flash-Karte vereinfacht [227]. Des Weiteren ist es möglich, ein Aufgaben-Scheduling auf Basis verschiedener Bitströme zu erstellen und die Konfigurationen zu verwalten. Die JTAG-Kette kann ebenfalls über die FSC im Fernzugriff verwendet werden, ohne dass eine tatsächliche physikalische Verbindung zu allen vorhandenen FPGAs aufgebaut werden muss.

Ausgelegt ist das System als Ersatz für herkömmliche prozessorbasierte Rechencluster. Entsprechend bildet jeder der FPGAs einen oder mehrere SoC-Knoten und einen Router zur Anbindung an die Kommunikationsinfrastruktur sowie zum Einbinden von anwendungsspezifischen Beschleunigern ab [228]. Als konkrete Anwendungen werden Applikationen aus dem Bereich der Strömungsmechanik und der Bioinformatik, speziell aus dem Bereich Sequenzsuchen, genannt [152].

DNBFC_S12_12_Cluster Die Firma Dini Group [229] bietet als industriefertige Lösung mit dem DNBFC_S12_12_Cluster ein System an, das aus zwölf auch als Einzelkomponente verwendbaren DNBFC_S12_PCIe-Karten derselben Firma aufgebaut ist. Jede dieser Karten verfügt über zwölf für die Anwenderlogik vorgesehene Spartan-6-FPGAs, so dass sich die Gesamtzahl der im System vorhandenen FPGAs

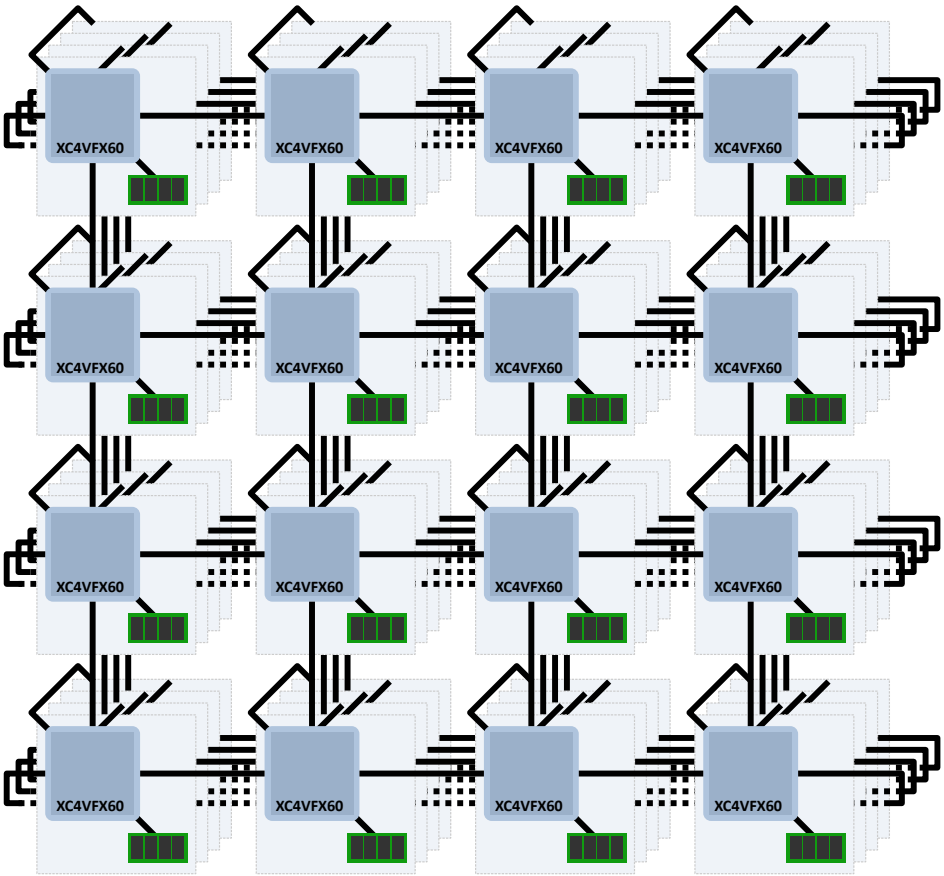


Abbildung 4.15: Architekturübersicht des Spirit-FPGA-Clusters

auf 144 multipliziert. Die Anwender-FPGAs sind jeweils mit ihren horizontalen Nachbarn über ein 77-bit- und mit ihren vertikalen über ein 64-bit-Interface verbunden. Aufgrund der zweizeiligen Anordnung bedeutet dies, dass die letzten beiden FPGAs über beide Interfaces miteinander verbunden sind. Zwar ist über JTAG ein Zugriff auf die einzelnen FPGAs möglich, jedoch erfolgt der Austausch mit der auf dem Server CPU laufenden Anwendung ausschließlich über den ersten und den letzten FPGA der Anordnung gebündelt über einen weiteren Spartan-6-FPGA, nicht individuell von jedem FPGA aus. Der Kommunikations-FPGA verbindet nicht nur die FPGAs einer Leiterkarte mit der PCIe-Gen1-4x-Brücke, er verknüpft zusätzlich mit Hilfe der GTP-Transceiver die zwölf Boards des FPGA-Clusters jeweils mit

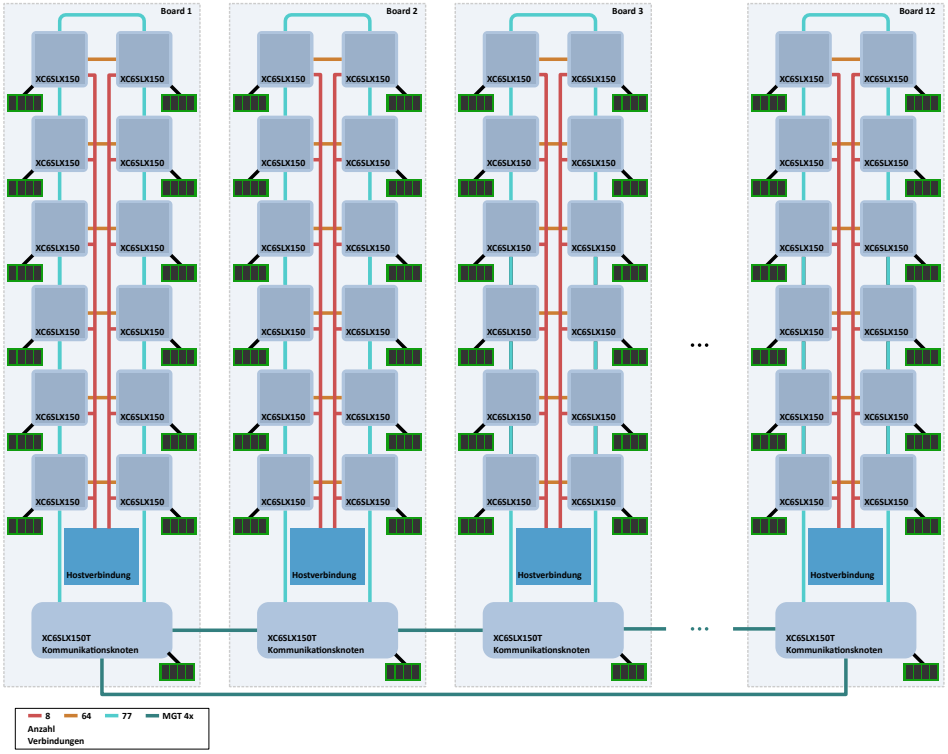


Abbildung 4.16: Architekturübersicht des DNBFC_S12_12-FPGA-Clusters

den physikalischen Nachbarn. Aus diesen Verbindungen baut sich eine Vollduplex-Ringstruktur mit vier Transceivern je Richtung auf, dargestellt in Abbildung 4.16.

Der FPGA-Cluster lässt sich in ähnlicher Form auch mit verschiedenen anderen DINI Group Boards realisieren, wie z.B. dem DNV6F6PCIe mit sechs Virtex-6-FPGAs [230]. Darüber hinaus gibt es ein MFS mit Xilinx Virtex-7-FPGAs. Der Aufbau des DNK7_F5_8_MFS [231] entspricht der oben beschriebenen Architektur des DNBFC_S12_12 mit dem Unterschied, dass die Kommunikation der Boards untereinander ausschließlich über das PCIe-Hostsystem erfolgt, da keine weiteren Kommunikationsstrukturen die Module untereinander koppeln, weshalb eine Einordnung zu den MFS erfolgt.

Tabelle 4.1: FPGA-Cluster-Übersicht

System	Struktur		Funktionalität		
	Anzahl Knoten	**Anzahl FPGAs**	**Logikzellen-Äquivalente**	**Speicher**	**Host-Anbindung**
CUBE	8	512	31 850 496	-	IDE / Global
COPACOBANA	20	120	2 073 600	-	USB / Global
RIVYERA S3	16	128	9 584 640	-	PCIe / Global
RIVYERA S6	16	128	18 872 704	65 536 MB	PCIe / Global
HAPS-64	1	4	3 035 136	-	PCIe / FPGA
FCCB	1	15	648 000	keine Angabe	System Bus / Global
Catapult	48	48	21 936 000	384 GB	PCIe / FPGA
SMILE	32	32	986 112	64 GB	Ethernet / Global
ScalableCore	64	64	933 056	32 768 kB	USB / Global
BEE3	1	4	409 600	64 GB	OS / Global
BEE4	1	4	2 199 552	128 GB	OS / Global
Maxwell	64	64	6 575 616	48 896 MB	PCIe / FPGA
Spirit	64	64	3 640 320	20 480 MB	Ethernet / Global
DNBFC_S12_12	12	144	21 231 792	288 GB	PCIe / Trägersystem

4.3 Zusammenfassung

Die vorgestellten FPGA-Cluster-Systeme verwenden unterschiedliche Architekturkonzepte. Die Cluster sind so angelegt, dass sie in ihrer Größe skalierbar sind. Verschiedene Systeme weisen jedoch eine Obergrenze bezüglich der Eingliederung weiterer FPGAs auf. Aus zwei Gründen ist die Skalierbarkeit limitiert. Zum einen ist in modularen Systemen mit einer Backplane eine feste Anzahl an Modulplätzen vorgesehen, wobei eine Kopplung mehrerer Backplanes nicht möglich ist. Zum anderen schränken Signalintegritätsprobleme bei der Übertragung von Daten über zu lange Datenleitungen die Erweiterbarkeit ein.

Mit Ausnahme des Spirit-FPGA-Clusters wird bei keinem System eine parallele Nutzung der vorhandenen Ressourcen von mehreren unabhängigen Anwendern vorgesehen. Die Möglichkeit besteht jedoch auch ohne expliziten Hinweis in den entsprechenden Veröffentlichungen beispielsweise beim Catapult-FPGA-Cluster aufgrund der Architektur aus prozessorbasierten Servern, die um jeweils einen FPGA erweitert werden. Die Hauptursache, die verhindert, dass die FPGAs des Clusters unabhängig voneinander verwendet werden können, ist, dass die eingesetzte Kommunikationsstruktur ein individuelles Ansprechen der einzelnen FPGAs nicht gestattet; dies gilt insbesondere für FPGA-Cluster, die für ein dediziertes Aufgabengebiet entworfen worden sind.

Die Übersicht über die individuellen Stärken und Schwächen der einzelnen Systeme ermöglicht eine Eingrenzung des Entwurfsraums für das im folgenden Kapitel vorgestellte Konzept eines neuartigen FPGA-Clusters sowie insbesondere eine Optimierung der Kommunikationsinfrastruktur auf Basis des im vorhergehenden Kapitel vorgestellten Verfahrens für eine Vielzahl von Anwendungen und für den Mehrnutzerbetrieb.

5 RAPTOR-XPress und RAPTOR-XPress-Cluster

Im Rahmen dieser Arbeit ist eine Neuentwicklung der Rapid-Prototyping-Plattform RAPTOR entstanden. Das ursprüngliche RAPTOR2000-System wird in [232] vorgestellt. Die FPGA-Plattformen der RAPTOR-Familie sind modulare Prototyping-Systeme, die mit einer Vielzahl unterschiedlicher Module bestückt werden können. Die Erweiterungsmodule (engl.: Daughter Board, DB) ihrerseits stellen dem Anwender verschiedenartige Funktionalitäten zur Verfügung [280]. Sämtliche Entwicklungen sind in der Arbeitsgruppe *Kognitronik und Sensorik* der Universität Bielefeld entstanden. Der modulare Aufbau ermöglicht die Kopplung verschiedener FPGA-Kombinationen mit unterschiedlichen IO-Boards, welche zum Beispiel über ADCs, Digital-analog-Wandler (engl.: Digital Analog Converter, DAC) oder proprietäre und industrieübliche Schnittstellen verfügen, um damit das benötigte Zielsystem aufzubauen. Trotz der Abwärtskompatibilität mit DBs der RAPTOR2000- bzw. RAPTOR-X64-Generation unterscheiden sich die Architekturen des RAPTOR-XPress und der aktuellen DBs wie DB-V5 [285], DB-V7 [233], DB-AD3 etc. erheblich von den vorher genannten Plattformen.

Neben der zugrunde liegenden Systemarchitektur des RAPTOR-XPress ist im Rahmen dieser Arbeit die Hardware des Trägersystems mit ihrer modularen Stromversorgung und ihrer ebenfalls modularen seriellen Hochgeschwindigkeits-Kommunikationsschnittstelle entstanden. Darüber hinaus wurden parallel in der Fachgruppe *Kognitronik und Sensorik* der Universität Bielefeld die Front-Side-Bus-MGT-Bridge (FSB-MGT-Bridge; siehe Abschnitt 5.6) sowie das Erweiterungsmodul DB-V5 aus Unterkapitel 5.3 realisiert. Das DB-V5 dient als Grundlage für das ebenfalls in der Fachgruppe *Kognitronik und Sensorik* der Universität Bielefeld entstandene DB-V7, welches in Abschnitt 5.4 vorgestellt wird.

Wie bereits bei den Vorgängergenerationen stellt die prototypische Implementierung von FPGA-Entwürfen ebenfalls ein Anwendungsgebiet des RAPTOR-XPress dar. Entwurfsraumexplorationen sowie die Validierung von ASICs auf Basis von FPGAs lassen sich in einem Echtzeit-Umfeld durchführen [104]. Darüber hinaus können neben dem allgemeinen Rapid-Prototyping flexibel rekonfigurierbare Hardware-Beschleuniger und Koprozessoren realisiert werden. Neu im Fokus stehen die Möglichkeiten eines Einsatzes im Bereich des HPRC.

Aufbauend auf den vorangegangenen Überlegungen ist der Entwurf einer möglichst universell einsetzbaren FPGA-Plattform das Ziel, auf deren Grundlage ein

skalierbarer FPGA-Cluster umgesetzt werden kann, welcher in der Lage ist, die in Kapitel 2 vorgestellten Anwendungen zu implementieren. Die parallele Verwendung der einzelnen Cluster-Ressourcen von unterschiedlichen Benutzern steht beim Entwurf des RAPTOR-XPress ebenfalls im Vordergrund. Eine Grundvoraussetzung für dieses Vorhaben ist neben der Modularität, das heißt der Möglichkeit, verschiedene Funktionalitäten und Verarbeitungseinheiten in das System zu integrieren, die Flexibilität der Kommunikationsinfrastruktur. Nur wenn die unterschiedlichen Topologieanforderungen der Anwendungen auf dem Cluster abbildbar sind, ist eine einfache und schnelle Realisierung möglich.

Im Folgenden werden zunächst die physikalischen Effekte betrachtet, die die leitungsgebundene Übertragung von Daten beeinflussen und Grundlage für den erfolgreichen Entwurf eines FPGA-Trägersystems sind, welches Datenbandbreiten im Multigigabit-Bereich zwischen den einzelnen Komponenten ermöglicht. Anschließend werden das RAPTOR-XPress sowie der mit diesem Basissystem als Grundlage realisierte FPGA-Cluster vorgestellt.

5.1 Grundlagen leitungsgebundener Datenübertragung

Während für die Beschreibung der Verbindungstopologien die physikalische Ebene der Datenübertragung außer Acht gelassen werden kann, müssen für die praktische Realisierung eines FPGA-Cluster-Systems die Kommunikation betreffende physikalische Effekte berücksichtigt werden. Wie bereits in den vorangegangenen Kapiteln gezeigt, bedingt eine Datenübertragung das Zusammenspiel von drei Teilkomponenten:

1. Eine oder mehrere Sendeschnittstellen
2. Ein einzelner Übertragungskanal oder die Bündelung mehrerer Übertragungskanäle
3. Eine zu der Anzahl der Sendeschnittstellen mindestens gleichwertige Anzahl an Empfangsschnittstellen

Die sende- und empfangsseitigen Schnittstellen werden durch die eingesetzten ICs bereitgestellt und unterliegen damit nicht dem Einfluss des Entwicklers eines Cluster-Systems. Zwar bieten moderne Treiberstufen Möglichkeiten, das zu übertragende Signal an den Übertragungskanal anzupassen, um einen fehlerfreien Datentransfer zu gewährleisten. Der leitungsgebundene Übertragungskanal muss jedoch darauf ausgelegt werden, die theoretisch gebotenen Transferraten möglichst störungsfrei übermitteln zu können. Mit Hilfe gezielter Vorverzerrungen auf Senderseite und Filtern, Verstärkern und Entzerren auf Empfangsseite kann der

Einfluss des Kanals zu einem gewissen Teil kompensiert werden. Je geringer die Einflüsse auf das Übertragungssignal jedoch sind, desto höhere Datenraten und Signalstreckenlängen lassen sich realisieren. Unterschiedliche Übertragungsstandards stellen dabei entsprechend unterschiedliche Anforderungen an die Signalqualität, also an den Anteil des Signalbildes, das am Empfänger aufgenommen wird. In diesem Abschnitt werden die für die Entwicklung des komplexen RAPTOR-XPress-Trägersystems relevanten Grundlagen der kupfergebundenen Datenübertragung erläutert. Die vorgestellten Informationen entstammen[234], [235] und [236], sofern nicht anders ausgewiesen.

Für die Umsetzung von Transferraten im Gigabit-Bereich muss beim Layout von Leiterkarten und beim Einsatz von Verbindungskabeln besondere Sorgfalt im Hinblick auf den Verlauf der Signalleitung gelegt werden. Genauso wichtig ist die zweite Komponente eines Übertragungskanals: der Rückstrompfad. Meist wird dieser über eine Massereferenzlage bzw. im Fall einer kabelgebundenen Übertragung über die Schirmung realisiert. Die Ortsnähe zum Signalpfad und eine unterbrechungsfreie Umsetzung sind für eine hohe Signalqualität wichtig. In Abbildung 5.1 werden typischerweise verwendete Aufbauten von Leiterkarten und Kabeln im Querschnitt gezeigt.

Die nachfolgend aufgeführten Angaben beziehen sich auf die Verhältnisse für Leiterkarten. Dennoch finden die Grundlagen auch auf kabelgebundene Übertragungsstrecken Anwendung. Diese müssen sogar explizit beachtet werden, wenn verschiedene Leiterkarten über Kabel miteinander verbunden sind. Im Folgenden werden die grundlegenden Größen und Verfahren beschrieben, die für den erfolgreichen Entwurf einer Leiterkarte zu berücksichtigen sind.

5.1.1 Impedanz

Als komplexe Größe beschreibt die Wellenimpedanz den Einfluss der elektrischen Eigenschaften der für die Formung eines Übertragungskanals verwendeten Materialien und der Leitungsgeometrie auf die Signalausbreitung. Der Kanal kann durch infinit viele Elemente bestehend aus der Impedanz z und der Admittanz y modelliert werden. Dabei bilden der Reihenwiderstand R und die Induktivität L des Signalpfades die Impedanz, während sich die Admittanz aus der kapazitiven Beziehung C und dem Leitwert G zwischen dem Signalpfad und dem Rückstrompfad zusammensetzt. In Abbildung 5.2 wird die Aneinanderreihung der einzelnen Elemente aufgezeigt.

Die charakteristische Impedanz Z_c als konstantes Verhältnis der ortsabhängigen Größen Spannung u und Strom i in einer verlustbehafteten Kupferleitung gestaltet sich wie folgt:

$$Z_c = \frac{u}{i} \tag{5.1}$$

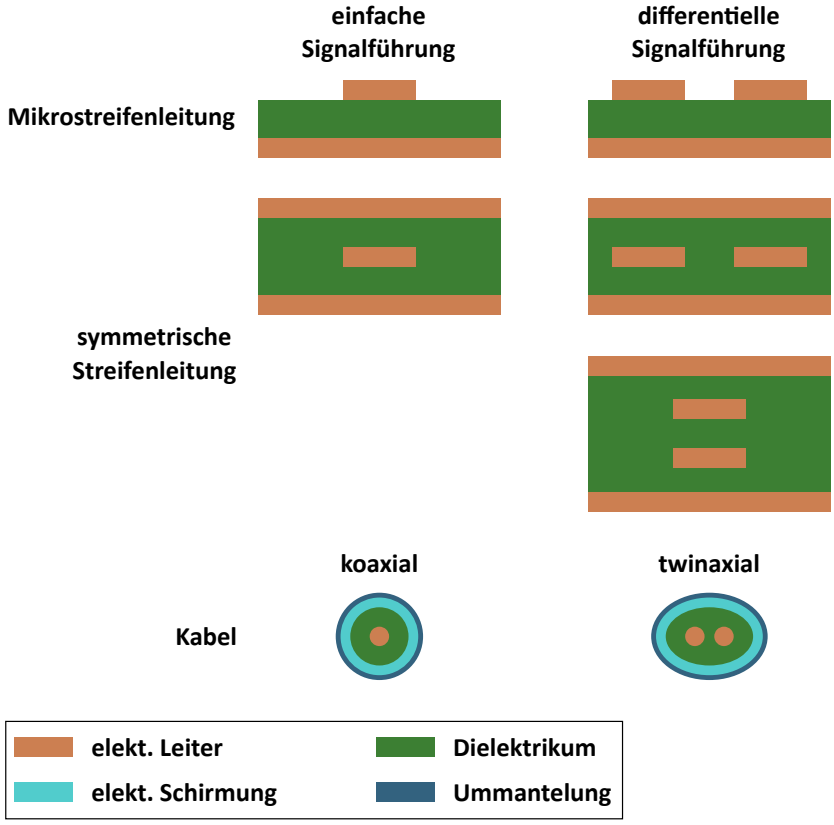


Abbildung 5.1: Querschnitte typischer Leiterbahnanordnungen in Leiterkarten und Kabeln

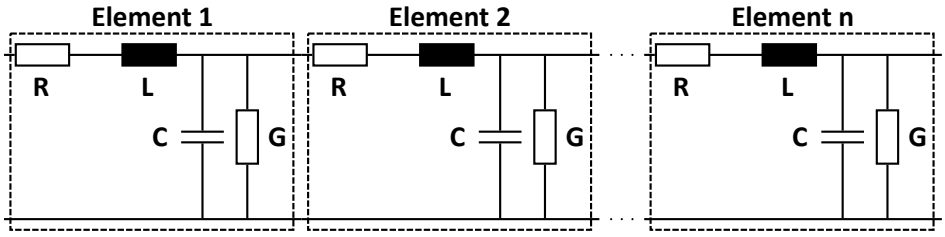


Abbildung 5.2: Leitungsmodellierung mit n RLGC-Elementen [234]

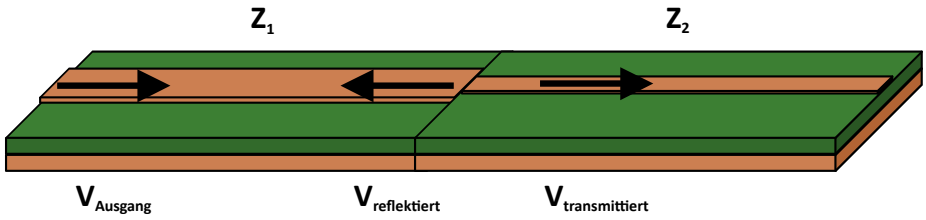


Abbildung 5.3: Signalreflexion am Impedanzsprung [234]

Aufgrund der Abhängigkeit der charakteristischen Impedanz Z_c von der Frequenz ω lässt sich Z_0 für ein gegebenes ω anhand des Leitermodells aus Abbildung 5.2 wie folgt bestimmen:

$$Z_0 = \sqrt{\frac{R_L + j\omega L_L}{G_L + j\omega C_L}} \tag{5.2}$$

Ab Frequenzen oberhalb von ca. 10 MHz kann die charakteristische Impedanz vereinfacht durch

$$\mathrm{Re}\,(Z_0) = \sqrt{\frac{L_L}{C_L}} \tag{5.3}$$

dargestellt werden, da $\mathrm{Im}\,(Z_0)$ gegen 0 strebt.

Signalreflexionen

Ändert sich die Impedanz entlang des Signalweges, beispielsweise aufgrund einer Geometrieänderung wie in Abbildung 5.3 oder an Steckverbindungen, so werden Teile des Signals reflektiert. Entsprechend gestaltet sich der Reflexionsfaktor r als Verhältnis zwischen reflektierter Spannung und Ausgangssignalspannung:

$$r = \frac{V_{reflektiert}}{V_{Ausgang}} = \frac{Z_2 - Z_1}{Z_2 + Z_1} \tag{5.4}$$

Entsprechend können drei Spezialfälle für die Reflexion identifiziert werden:

1. Offenes Leitungsende: Das eintreffende Signal wird vollständig reflektiert.
2. Kurzgeschlossenes Leitungsende: Das eintreffende Signal wird mit negativem Vorzeichen vollständig reflektiert.
3. Terminiertes Leitungsende: Bei einer auf Z_0 abgestimmten Terminierung ist der Reflexionsfaktor $r = 0$.

Alle anderen Verhältnisse führen zu Überlagerungen des reflektierten und des ausgesendeten Signals, was eine Verschlechterung des Signalbildes zur Folge hat. Impedanzkritische Komponenten im Leitungsverlauf auf einer Leiterkarte sind insbesondere Pads von Bauteilen sowie Lagenwechsel mit Hilfe von VIAs (engl.: Vertical Interconnect Access). Für den eigentlichen Lagenwechsel nicht benötigte überhängende Teile eines VIAs verhalten sich wie offene, nichtterminierte Leitungsenden und führen entsprechend zur Reflexion und somit zu Überlagerungen mit dem eigentlichen Signal.

5.1.2 Ausbreitungsgeschwindigkeit

Die Signalgeschwindigkeit in einem Leiter wird bestimmt durch die Eigenschaften des den Leiter umgebenden Materials und ist nach oben durch die Lichtgeschwindigkeit c begrenzt. Mit Hilfe der Permitivität ϵ_0 im Vakuum, der Dielektrizitätskonstante des verwendeten Materials ϵ_r, der Permeabilität im Vakuum μ_0 und der relativen Permeabilität des Materials μ_r ergibt sich die Ausbreitungsgeschwindigkeit:

$$v_S = \frac{1}{\sqrt{\epsilon_0 \epsilon_r \mu_0 \mu_r}} \tag{5.5}$$

Entsprechend kann die Leitungsverzögerung t_d mit der Leitungslänge s_L gebildet werden:

$$t_d = \frac{s_L}{v_S} \tag{5.6}$$

5.1.3 Dämpfung

Die im Ersatzschaltbild des Leiters aufgeführten Komponenten (Abbildung 5.2) beeinflussen in Abhängigkeit von der Leitungslänge und der Übertragungsfrequenz die Dämpfung des Signals unterschiedlich stark. In Abbildung 5.4 wird für eine gegebene Leitungslänge und die Frequenz der Übertragung der für die Dämpfung vorrangig verantwortliche physikalische Effekt veranschaulicht. Wie zu erkennen ist, verschiebt sich mit ansteigender Frequenz das für die Dämpfung ausschlaggebende dominante Element.

Signale auf sehr kurzen Übertragungsstrecken mit niedrigen Frequenzen werden vorwiegend durch die ohmschen Verluste entlang des Leiters sowie durch die kapazitive Kopplung zum Rückstrompfad beeinflusst. Im unteren Megahertzbereich ist der Einfluss der Induktivität des Leiters im Zusammenspiel mit der Kapazität der ausschlaggebende Faktor. Ab Frequenzen oberhalb von ca. 100 MHz setzt der Skin-Effekt ein, bei welchem die Ladungsträger in den Außenbereich des Leiters

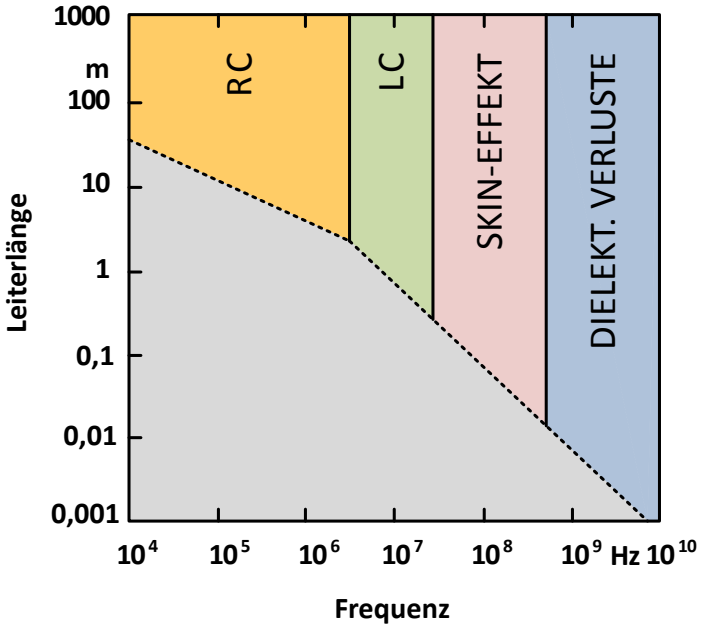


Abbildung 5.4: Hauptdämpfungseffekt auf einer 150 μm breiten 50-Ω-Streifenleitung [235]

gedrängt werden, was eine Verringerung des für die Übertragung nutzbaren Leiterquerschnitts zur Folge hat. Mit ansteigender Frequenz findet ein Stromfluss schließlich nur noch auf der Leiteroberfläche statt, so dass sogar die Oberflächenstruktur des Kupfers Einfluss auf die Signalausbreitung hat. Bei noch weiter ansteigenden Frequenzen nehmen die dielektrischen Verluste über das Substratmaterial immer mehr zu und führen letztendlich zu einer Verschlechterung des Signalbildes beim Empfänger.

5.1.4 Übersprechen

Die gegenseitige Beeinflussung von zwei oder mehr parallel verlaufenden Leitungen wird als Übersprechen oder englisch Cross-Talk bezeichnet. Dabei muss nicht nur der eigentliche signalführende Leiter betrachtet werden, sondern auch der Rückstrompfad. Bei der Übertragung eines elektrischen Signals entlang eines Leiters wird der Leiter konzentrisch um den Signal- und Rückstrompfad von magnetischen Feldern umgeben, während sich zwischen Signal- und Rückstrompfad die zum elektrischen Feld gehörigen Feldlinien ausbilden. Befinden sich andere Leitungen

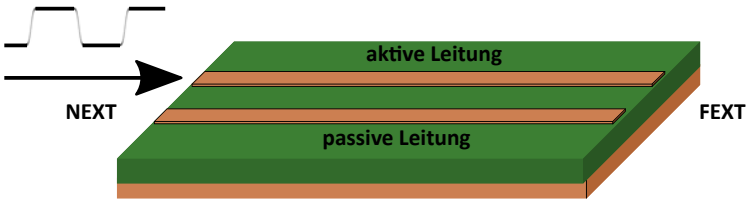


Abbildung 5.5: NEXT und FEXT anhand einer aktiven und einer passiven Streifenleitung [234]

innerhalb des Wirkungsbereichs dieser Feldlinien, findet eine kapazitive und induktive Einkopplung in Form von Rauschen auf der elektrisch inaktiven Leitung mit jeder Spannungs- und Stromänderung der betriebenen Übertragungsstrecke statt. In Abbildung 5.5 wird eine Beispielkonfiguration aus zwei parallelen Leiterbahnen mit gemeinsamer Massereferenz aufgezeigt. Es wird davon ausgegangen, dass die Enden der Leiter mit der Quellen- und Leitungsimpedanz terminiert sind, so dass keine Reflexionen auftreten. Die Einstreuungen, die auf der Seite des eingespeisten Signals auf der passiven Leitung beobachtet werden können, werden als near-end-crosstalk (NEXT) bezeichnet. Entsprechend werden die Einflüsse, die auf der Seite der entfernten Signalsenke auf der passiven Leitung erscheinen, far-end-crosstalk (FEXT) genannt.

Minimiert werden kann der NEXT-Einfluss lediglich durch weitere Separation der beiden Leiter. Das Übersprechen an der Senke, also der FEXT-Einfluss, kann durch die folgenden drei Maßnahmen beeinflusst werden:

- Reduzierung der Strecke auf der das Übersprechen stattfindet
- Erhöhung der Anstiegszeit des Signals
- Erhöhung der Abstände zwischen den einzelnen Leiterbahnen

5.1.5 Differentielle Übertragung

Im Gegenteil zum Cross-Talk, also zur unerwünschten gegenseitigen Beeinflussung paralleler Leitungssegmente, wird die Beeinflussung bei der differentiellen Signalübertragung gezielt eingesetzt, um auf zwei parallelen Leiterbahnen Daten zu übertragen. Die Übertragung auf den beiden Leitern erfolgt mittels komplementärer Signalpegel. Zur Unterscheidung wird der Leiter, der das positive Datensignal transportiert, als P-Leiter bezeichnet, entsprechend führt der N-Leiter das inverse Signalbild. Wie aus der Bezeichnung der Übertragungsmethode abgeleitet werden kann, wird das Zielsignal aus der Differenz der beiden aufgenommenen Signalpegel entsprechend Gleichung 5.7 gebildet.

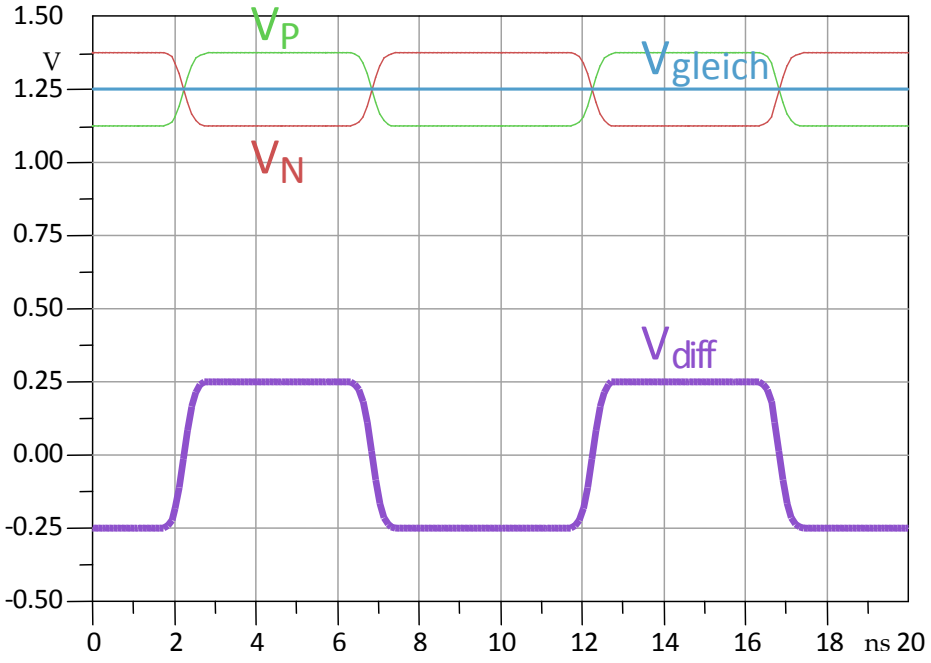


Abbildung 5.6: Spannungskomponenten der differentiellen Signalübertragung [234]

$$V_{diff}(t) = V_P(t) - V_N(t) \tag{5.7}$$

Neben dem erwünschten differentiellen Signal existiert noch ein Gleichtaktsignal als Durchschnitt der beiden einzelnen Spannungen, welches bei idealer Leitungsführung, wie in Abbildung 5.6 dargestellt, über die Zeit konstant ist.

$$V_{gleich}(t) = \frac{V_P(t) + V_N(t)}{2} = \text{konst.} \tag{5.8}$$

Variationen im Leitungsverlauf beispielsweise an Steckverbindungen bewirken jedoch, dass der Gleichtaktanteil selbst zu einem Signal wird. Dies kann zum einen empfangsseitig ein sicheres Detektieren des Signals in der Eingangsstufe verhindern und zum anderen zur Ausstrahlung von elektromagnetischen Störungen führen.

Für eine erfolgreiche Übertragung mit Hilfe eines differentiellen Signals müssen in Bezug auf die Leitungsführung die folgenden Voraussetzungen geschaffen werden:

1. Konstante Impedanz über die Leitungsstrecke zur Vermeidung von Reflexionen

2. Minimierung des zeitlichen Unterschieds (engl. Skew) der beiden Signalanteile innerhalb eines Übertragungbits

3. Symmetrische Leitungsführung im Hinblick auf einen konstanten Leiterbahnabstand zwischen den beiden Leitungen und eine konstante Leitungsbreite

4. Minimierung des Längenunterschieds zwischen den beiden Leitern zur Minimierung von Laufzeitunterschieden

5. Bewusste Signalkopplung der beiden Leiter, beispielsweise über die gemeinsame Referenzlage, zur Minimierung der externen Störfaktoren

Die äquivalente differentielle Impedanz von Leitern, die aufgrund ihres Abstands sich nicht gegenseitig beeinflussen, bestimmt sich aus der Summe der einfachen Impedanz Z_0 der beiden Leiter durch:

$$Z_{diff} = 2Z_0 \tag{5.9}$$

Bei einer typischen Leitungsimpedanz von $Z_0 = 50\,\Omega$ beträgt die differentielle Impedanz entsprechend $Z_{diff} = 100\,\Omega$. Wird der Abstand verringert, so findet eine gegenseitige Einkopplung innerhalb des Leitungspaares statt. Die Folge ist, dass die Impedanz von den jeweils eingeprägten Signalen abhängig wird. Zwei ideale Zustände können auftreten:

1. Der gleiche Signalpegel wird auf beiden Leitern übertragen, auch als Even-Mode bezeichnet.

2. Der komplementäre Signalpegel wird auf beiden Leitern übertragen, auch als Odd-Mode bezeichnet.

In beiden Fällen wird das Signal unverändert übertragen. Die differentielle Impedanz für den Odd-Mode ergibt sich aus der Summe der Odd-Mode-Impedanz der beiden Leiter:

$$Z_{diff} = 2Z_{odd} \tag{5.10}$$

Im Fall gleicher Signalpegel ergibt sich die Impedanz aus der Parallelschaltung der beiden Einzelimpedanzen:

$$Z_{common} = \frac{Z_{even}Z_{even}}{Z_{even} + Z_{even}} = \frac{1}{2}Z_{even} \tag{5.11}$$

Jede Unregelmäßigkeit in der Leitungsführung des differentiellen Paares hat eine Überführung von Signalanteilen des differentiellen Signals in ein unerwünschtes

Abbildung 5.7: Übertragungskanal als Zweitor-Black-Box

Common-Signal zur Folge. Neben der Verschlechterung des differentiellen Signalbildes am Empfänger steigt die elektromagnetische Abstrahlung entlang der Leiterbahn oder des Kabels.

Sowohl die Terminierung des differentiellen Signals zwischen den Leitern als auch des Common-Signals der einzelnen Signalleitungen muss berücksichtigt werden. Zusätzlich kann eine Kapazität zur Terminierung hinzugefügt werden um Gleichströme zu blocken.

5.1.6 S-Parameter

Zur Beschreibung der verschiedenen vorgestellten Einflüsse also des Einwirkens des Kanals auf ein Eingangssignal werden Streuparameter (engl. scattering parameter), kurz S-Parameter verwendet. Der eigentliche Übertragungskanal wird dabei als Black-Box aufgefasst, welche über ein oder mehrere Ein- und Ausgangstore verfügt. Die S-Parameter beschreiben dabei immer das Verhältnis zwischen dem Signal am Ausgangstor und dem angelegten Signal am Eingangstor. In der Frequenzdomäne ergibt sich somit die Beschreibung anhand des Verhältnisses zwischen der Amplitude und der Phasenlage des Sinussignals am Ausgang und dem angelegten Eingangssinussignal. Eine einfache Verbindungsstrecke mit einer Signalleitung und einem Rückstrompfad über beispielsweise eine Referenzlage kann demzufolge als vierpolige Beschreibung dargestellt werden. Fasst man die hin- und rücklaufenden Signaleingänge jeweils wie in Abbildung 5.7 dargestellt zusammen, ergibt sich ein Zweitor-Netzwerk.

Durch das Bilden von Toren können komplexe Netzwerke mit mehreren Toren ebenfalls kompakt beschrieben werden. Zur vereinfachten Schreibweise werden die einzelnen S-Parameter zur S-Matrix zusammengefasst, welche für jedes Tor i das Verhältnis des hinlaufenden Signals a_i und des rücklaufenden Signals b_i beschreibt. Im Beispiel aus Abbildung 5.7 ergibt sich die S-Matrix S durch:

$$b = Sa \iff \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \tag{5.12}$$

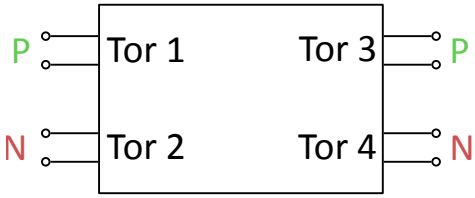


Abbildung 5.8: Differentieller Übertragungskanal als Viertor-Black-Box

Entsprechend beschreibt Vektor b die Antworten des Systems, während Vektor a die Stimulisignale darstellt. Allgemein führt dies zu der folgenden Beschreibung:

$$b_k = S_{kj} \times a_j \tag{5.13}$$

mit

$$S_{kj} = \frac{\text{Signal}_{k\,O}}{\text{Signal}_{j\,I}} \tag{5.14}$$

Zu beachten ist, dass die Indizes I Eingangsstimuli und O Ausgangssignale beschreiben. Die vier einzelnen S-Parameter der Matrix ergeben sich dabei durch:

- den Eingangsreflexionsfaktor S_{11} als Verhältnis zwischen dem einlaufenden und dem rücklaufenden Signal, beobachtet am Eingang,
- den Ausgangsreflexionsfaktor S_{22} als Verhältnis zwischen dem einlaufenden zum dem rücklaufenden Signal, beobachtet am Ausgang,
- den Vorwärts-Transmissionsfaktor S_{21}, welcher das Übertragungsverhalten des Kanals vom Eingang zum Ausgang beschreibt,
- den Rückwärts-Transmissionsfaktor S_{12}, welcher das Übertragungsverhalten des Kanals vom Ausgang zum Eingang beschreibt.

Für lineare, passive Elemente gilt, dass $S_{21} = S_{12}$ ist, wodurch das Verhalten des Übertragungskanals mit Hilfe der drei beschreibenden Elemente der Matrix erfasst werden kann.

S-Matrix für differentielle Signale

Analog zu den vorangegangenen Überlegungen ergibt sich für differentielle Signale eine Beschreibung des Übertragungsverhaltens durch ein Viertor-Netzwerk (Abbildung 5.8).

Die beschreibende S-Matrix aus Gleichung 5.15 besitzt $N = \frac{n(n+1)}{2} = 10$ eindeutige S-Parameter, wobei n der Anzahl der Tore entspricht. Anhand der Indizierung der einzelnen Parameter in der S-Matrix findet man die Transmissionsfaktoren, im Englischen auch als Insertion-Loss bezeichnet, an den Positionen S_{31} und S_{13} für den P-Kanal bzw. S_{42} und S_{24} für den N-Kanal.

$$S = \begin{bmatrix} S_{11} & S_{12} & S_{13} & S_{14} \\ S_{21} & S_{22} & S_{23} & S_{24} \\ S_{31} & S_{32} & S_{33} & S_{34} \\ S_{41} & S_{42} & S_{43} & S_{44} \end{bmatrix} \tag{5.15}$$

Dem gegenüber befinden sich die Reflexionsfaktoren, welche auf Englisch auch als Return-Loss bezeichnet werden, auf der Hauptdiagonalen mit S_{11}, S_{22}, S_{33} und S_{44}. Darüber hinaus kann die gegenseitige Beeinflussung der Leitungen, also das Übersprechen, an den restlichen Eintragungen abgelesen werden. So beschreiben die Parameter S_{21}, S_{12}, S_{43} und S_{34} den NEXT-Anteil, während anhand von S_{41}, S_{14}, S_{32} und S_{23} der FEXT-Einfluss dargestellt wird.

5.2 RAPTOR-XPress

Das RAPTOR-XPress ermöglicht eine enge Kopplung von bis zu vier DBs auf einem Basisboard. Dabei können, wie in Abbildung 5.10 dargestellt, benachbarte Module über Direktverbinder mit geringer Latenz miteinander kommunizieren. Darüber hinaus existiert eine über einen FPGA geführte Punkt-zu-Punkt-Verbindung, welche beliebige Verschaltungen der Module einschließlich Broad- und Multicast erlaubt. Zur Kopplung mit anderen RAPTOR-XPress-Systemen werden die seriellen Hochgeschwindigkeitsverbinder der eingesetzten DBs wie DB-V5 und DB-V7 über ein Crosspoint-Switch-Array genutzt. Das Crosspoint-Switch-Array ermöglicht sowohl eine Verschaltung der DBs eines RAPTOR-XPress untereinander als auch eine Anbindung von bis zu vier weiteren voll bestückten RAPTOR-XPress-Systemen. Der so entstandene FPGA-Cluster zeichnet sich durch seine hohe Kommunikationsbandbreite aus. Die Anbindung zum Host-PC erfolgt über einen PCIe 2.0 8x-Switch, wodurch jedem Modul die volle Bandbreite der PCIe-Schnittstelle zugänglich ist. Abbildung 5.9 zeigt das RAPTOR-XPress-Basissystem mit zwei DB-V5-Modulen und einem für kupfergebundene Kommunikation ausgelegten High-Speed-Piggyback.

In den folgenden Abschnitten wird eine Beschreibung der Funktionalitäten und insbesondere der Kommunikationsstruktur des RAPTOR-XPress gegeben. Dabei werden sowohl die für die Intra-Board-Kommunikation eingesetzten Strukturen als auch die für die Kopplung mehrerer RAPTOR-XPress-Boards relevanten MGT-Verbindungen vorgestellt. Dargestellt werden neben der Organisation der Übertragungsstrukturen die notwendigen Maßnahmen zur Sicherstellung der Signalqualität.

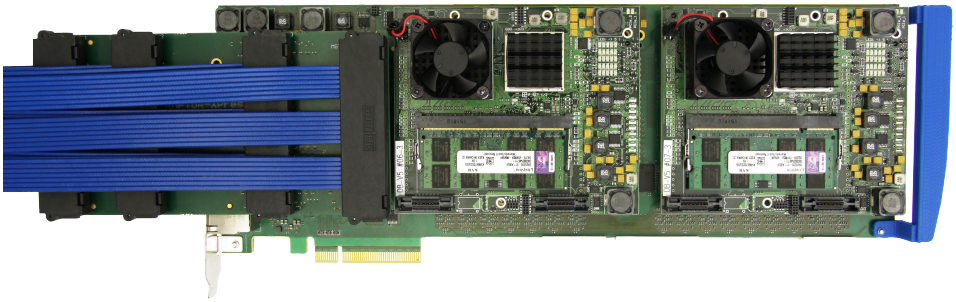


Abbildung 5.9: RAPTOR-XPress mit zwei DB-V5-Modulen und High-Speed-Piggyback für kupfergebundene Kommunikation

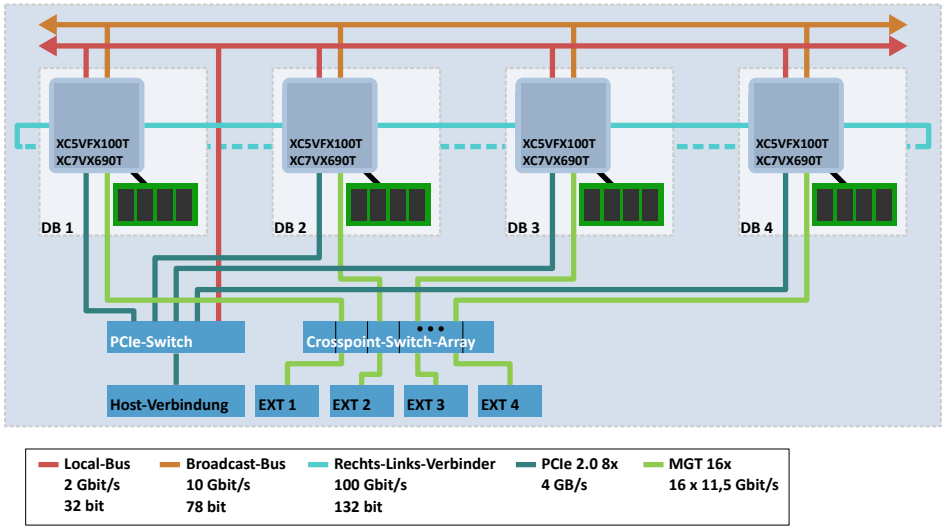


Abbildung 5.10: Vereinfachte Architekturübersicht des RAPTOR-XPress

Eine vollständige Übersicht über die Kommunikationsstrukturen wird im Anhang in Abbildung A.1 gegeben und in den folgenden Abschnitten näher erläutert.

5.2.1 Crosspoint-Switch-Array

Die für die Benutzerlogik verwendbaren 16 GTX-Transceiver der DB-V5-Module ermöglichen Übertragungsraten von bis zu 6,5 Gbit/s [237], die 20 GTH-Transceiver der DB-V7-Module sogar bis zu 13,1 Gbit/s [18]. Aufbauend auf den vorangegan-

Tabelle 5.1: Crosspoint-Switch-Ausgangsbelegung für VSC3008

Gruppe	Ausgang	PCB-Beschaltung
1	Y0	EXT 4
1	Y1	EXT 2
1	Y4	EXT 3
1	Y5	EXT 1
2	Y2	DB 4
2	Y3	DB 2
2	Y6	DB 3
2	Y7	DB 1

genen Untersuchungen zur Entwicklung einer flexiblen Kommunikationsinfrastruktur für die Inter- und Intra-Board-Kommunikation sind die Transceiver über ein Crosspoint-Switch-Array untereinander verbunden. Der Aufbau des Crosspoint-Switch-Array entspricht dem in [238] beschriebenen Prinzip der partiellen Crossbar. Erst diese Verschaltung macht eine breitbandige Kopplung der FPGAs und somit die Bildung eines FPGA-Clusters möglich. Abbildung 5.11 zeigt das aus asynchronen 8×8 VSC3008-Crosspoint-Switchen der Firma Vitesse (inzwischen Teil von Microsemi) implementierte Array. Um eine bessere Signalqualität bei der Übertragung zu erreichen, können die VSC3008 durch VSC3308 ersetzt werden, da letztere erweiterte Signalkonditionierungsoptionen bereitstellen. Mit einer maximalen Datenrate von 11,5 Gbit/s bieten beide Bestückungsoptionen ausreichend Bandbreite, um zusätzlich die GTH-Transceiver der Virtex-6-Familie (maximal 11,18 Gbit/s, je nach Speedgrade des Bausteins) bei voller Übertragungsgeschwindigkeit zu unterstützen; typischerweise wird eine Datenrate von 10,3125 Gbit/s durch die meisten IP-Cores genutzt.

Broadcast-Übertragungen sind beim VSC3008 im Frequenzbereich von 8 Gbit/s bis 11,5 Gbit/s auf eine Verschaltung von einem Eingang auf zwei Ausgänge, die sich in unterschiedlichen Gruppen befinden müssen, limitiert [239, 240]. In Tabelle 5.1 wird die konkrete Verschaltung der Gruppen aufgezeigt. Entsprechend ist es möglich bei Übertragungsraten größer als 8 Gbit/s Verbindungen zu einem DB des gleichen RAPTOR-XPress sowie zu einem DB eines zweiten verbundenen RAPTOR-XPress aufzubauen. Eine Weiterverschaltung über das Crosspoint-Switch-Array des zweiten RAPTOR-XPress ist ebenfalls realisierbar.

Die MGT-Kopplung zwischen den RAPTOR-XPress-Boards kann sowohl kupferbasiert mit Kabeln der Firma Samtec als auch als optische Übertragung via Lichtwellenleiter (LWL) mit Zwölf-Kanal-Transceivern der Firma Broadcom bzw. mit Samtec Firefly Verbindern erfolgen. Realisiert wird die externe Verbindung über ein dediziertes High-Speed-Piggyback, das im Fall der kupfergebundenen Übertragung vier Samtec QSE 42-DP-Stecker zur Verfügung stellt, an denen jeweils

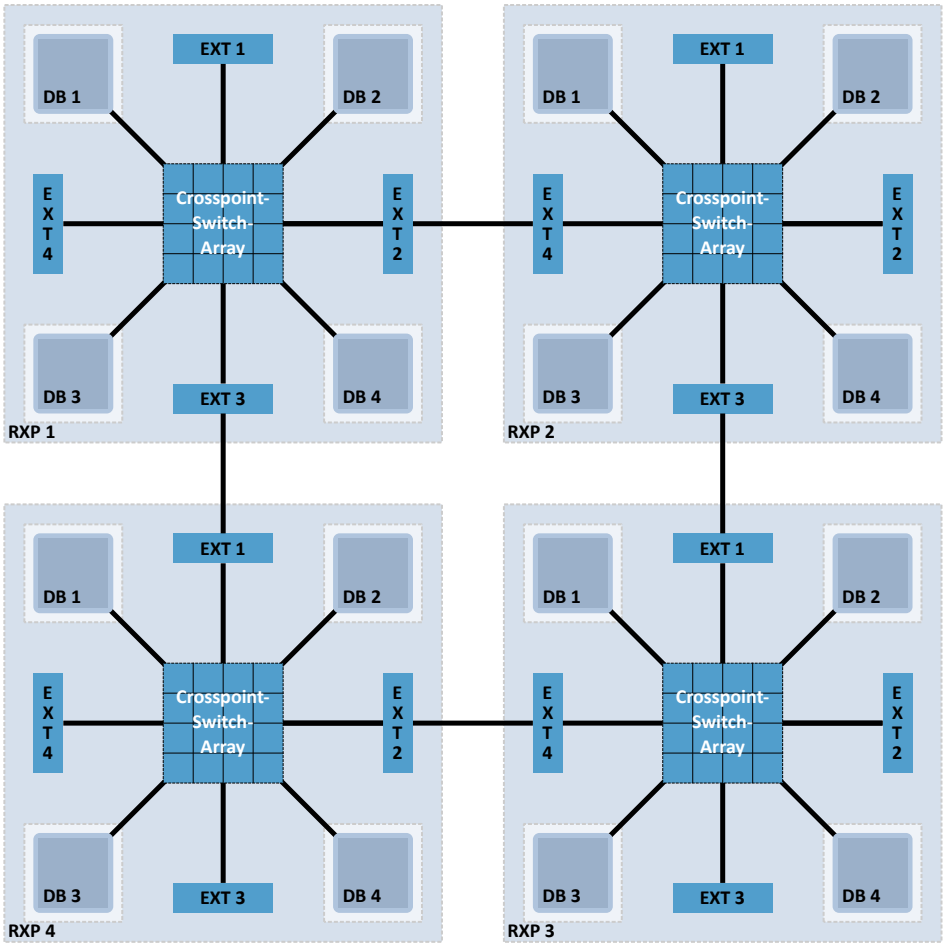


Abbildung 5.11: Beispielkonfiguration der MGT-Kommunikationsinfrastruktur mit 4 RAPTOR-XPress

16 MGT-Lanes anliegen. Mit Samtec EQDP-Kabeln ist eine Übertragung bis zu einer Länge von 2 m möglich. Bei einer Bestückung mit VSC3308, welche voll Broadcast-fähig sind, ist es möglich, von einem FPGA ausgehend an sämtliche andere FPGAs des Clusters mit den vollen 16 MGT-Lanes Daten zu senden, ohne dass ein zwischengeschalteter FPGA als Router fungieren muss. Dies führt zu einer minimalen Latenz bei der Verteilung von Broadcast-Mitteilungen im Cluster, da sich genau wie

bei einfachen Übertragungen lediglich die Leitungslatenzen und die Latenzen der zwischengeschalteten Crosspoint-Switche summieren.

Im Fall der optischen Übertragung befinden sich acht Mini-Pods auf dem Adapterboard. Übertragungsstrecken von bis zu 150 m können mit voller Übertragungsgeschwindigkeit der Virtex-5-MGTs bei Nutzung von OM4-spezifizierten LWL realisiert werden.

Eine transparente Kopplung von Cluster-FPGAs ist nur möglich mit hoher Datenbandbreite und geringen Latenzen. Ideale Voraussetzungen hierfür schaffen die Links-Rechts-Verbinder und der Broadcast-/Config-Bus, also die Verbindung der vier DBs eines RAPTOR-XPress-Basisboards untereinander.

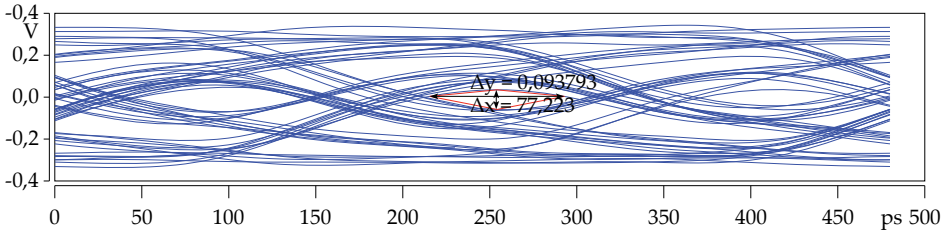
5.2.1.1 Maßnahmen zur Sicherung der Signalqualität

Für den Betrieb der 256 differentiellen MGT-Übertragungsstrecken eines RAPTOR-XPress mit bis zu 11,5 Gbit/s spielt die Signalqualität eine wichtige Rolle, um die gesendeten Daten fehlerfrei am Empfänger umsetzen zu können. Die im Rahmen dieser Arbeit getroffenen Maßnahmen werden im Folgenden kurz erläutert. Darüber hinaus werden Messergebnisse vorgestellt, welche die im Entstehungsprozess ausgeführten Simulationen bestätigen.

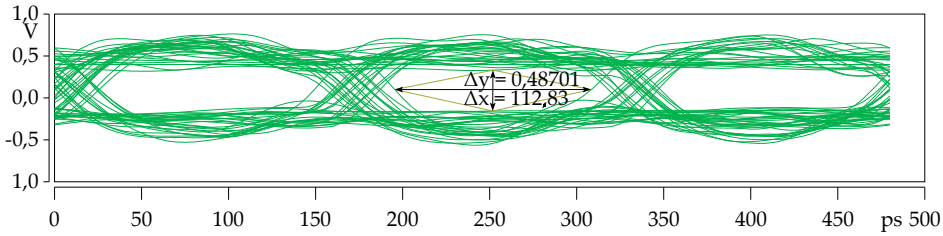
Voraussetzung für eine hohe Signalqualität ist eine saubere Führung der differentiellen Leiterbahnen innerhalb einer impedanzkontrollierten Leiterkarte. Entsprechend gestalten sich der Lagenaufbau und die Wahl der Strukturgrößen der Leiterbahnen. Die für die Führung der als 100 Ω edge-coupled realisierten Signalpaare vorgesehene Lage, das heißt mit konstantem Abstand zueinander in der gleichen Ebene parallel verlaufenden Leitungen, wird mit einer gegen Masse verbundenen Kupferfläche geflutet, welche jedes Paar umschließt. Die darüber und darunter befindlichen Lagen sind ebenfalls als Massereferenzflächen ausgelegt, so dass sich eine Schirmung für die eigentliche Übertragungsstrecke in alle Raumrichtungen ergibt. Eine unerwünschte Beeinflussung durch benachbart verlaufende Leiterbahnen oder sogar EMI-Einkopplungen von außerhalb kann auf diese Weise minimiert werden. Darüber hinaus wird eine unterbrechungsfreie dem Signalpfad folgende Strecke für den Rückstrompfad gebildet.

Simulationen der MGT-Übertragungsstrecke zeigen, dass die sorgfältige Signalführung und die Anpassung des empfangenen Signals mit Hilfe der im VSC3008 und Virtex-5 vorhandenen Equalizer Datenraten von 6,25 Gbit/s ermöglichen. Eine streckenabhängige Optimierung durch Signalvorverzerrung und Empfangsequalizer gewährleistet stabile Augenöffnungen oberhalb der Entscheidungsschwellen der GTX-Transceiver, wie in Abbildung 5.12 dargestellt [285].

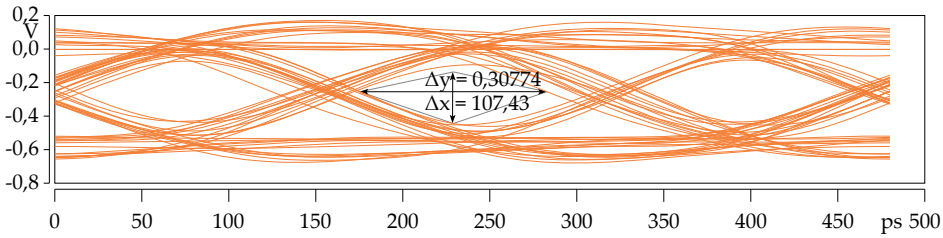
Neben den nicht zu beeinflussenden Impedanzsprüngen an den Lötverbindungen innerhalb der Übertragungsstrecke (Steckverbinder, Crosspoint-Switch und FPGAs) haben überhängende, nicht für den eigentlichen Lagenwechsel benötigte Teile der VIAs Einfluss auf die Signale. Diese für das Signal offenen Enden führen zu Refle-



(a) Empfangsseite VSC3008



(b) Sendeseite VSC3008



(c) Empfangsseite GTX-Virtex-5

Abbildung 5.12: Augendiagramme einer DB-V5-GTX-VXC3008-GTX-Strecke bei einer Übertragungsrate von 6,25 Gbit/s

xionen und verschlechtern somit das Empfangsbild der Zielkomponente. Neben einem angepassten Lagenaufbau und der Auswahl einer geeigneten VIA-Topologie (Micro-VIAs, vergrabene Micro-VIAs und ein mehrere Lagen überbrückendes vergrabenes VIA) ist das teilweise Zurückbohren und somit mechanische Entfernen des elektrisch leitfähigen Teils der VIA-Hülse des vergrabenen VIAs bis auf die entsprechende Innenlage nach der Fertigung eine Maßnahme zur Minimierung dieser Störfaktoren. Mit Hilfe dieses Verfahrens lässt sich die Anzahl an zusätzlich benötigten angepassten VIAs reduzieren, wodurch eine weniger Aufwendige und damit kostenintensivere Fertigung möglich ist. Der Lagenaufbau ist im Anhang dieser Arbeit in Abbildung A.2 dargestellt. Zwar können die Einflüsse der VIA-

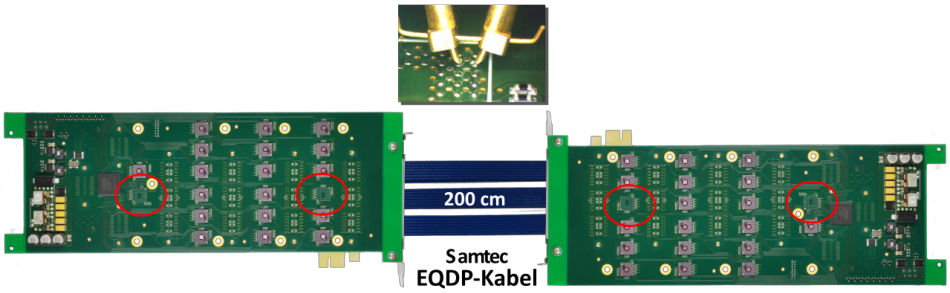


Abbildung 5.13: Messaufbau an der FSB-MGT-Bridge

Hülsen auf den Signalverlauf mit Hilfe von 2,5D- bzw. 3D-Simulationsverfahren ausgewertet werden, eine zusätzliche Verifikation durch Messungen ist jedoch trotzdem angeraten.

Aus den Simulationsergebnissen geht hervor, dass die getroffenen Maßnahmen eine Fertigung mit Standard-FR4-Materialien bei voller Leistungsfähigkeit des Gesamtsystems ermöglichen und ein Einsatz von teuren Speziallaminaten nicht notwendig ist. Messungen an der FSB-MGT-Bridge (siehe Abschnitt 5.6) bestätigen die getroffenen Überlegungen und werden im folgenden Abschnitt kurz vorgestellt.

5.2.1.2 Verifikation der Signalqualität

Anhand von Messungen am Crosspoint-Switch-Array der FSB-MGT-Bridge kann die Wirksamkeit der getroffenen Maßnahmen zur Sicherung der Signalqualität auf den MGT-Übertragungsstrecken verifiziert werden. Die Messreihen erfolgen mit Hilfe eines Netzwerk-Analysators der Firma Agilent (E5071C mit differentieller Prüfspitze N1021B). Für den Signalabgriff werden zwei der insgesamt 20 Crosspoint-Switche entfernt, um direkten Zugang zu den Lötpunkten auf der Leiterkarte zu erhalten. Die Auswahl der Messpunkte erfolgt aufgrund der aus dem Layout bekannten unterschiedlichen Leitungslängen von 55 mm bis 398 mm, um die für alle anderen Pfade repräsentativen Ergebnisse aufzuzeichnen. Hinzu kommt die Übertragung über eine Kabellänge von 2 m, wie in Abbildung 5.13 dargestellt einschließlich der Detailansicht der Messung an den differentiellen Pads.

Messungen der Laufzeitverzögerung für eine 205,1 mm lange Leiterkartenstrecke und für die eingesetzten Samtec EQDP-Kabel bestätigen das spezifizierte ϵ_r der verwendeten Materialien, FR4 für die Leiterkarte und Fluorethylenpropylen (FEP) für die Kabel, wie die in Tabelle 5.2 aufgeführten Ergebnisse zeigen. Anhand der Materialkonstante des Dielektrikums wird die Ausbreitungsgeschwindigkeit des Signals im elektrischen Leiter nach Gleichung 5.16 bestimmt.

Tabelle 5.2: Verifikation der Materialkonstanten

	Leiterkarte	**EQDP-Kabel**
Material Dielektrikum	FR-4	FEP
Signalstrecke	$250{,}1\,\mathrm{mm}$	$2000\,\mathrm{mm}$
Signallaufzeit	$1{,}743\,\mathrm{ns}$	$9{,}62\,\mathrm{ns}$
Spezifiziertes ϵ_r	$4,2$	$2,062$
Ermitteltes ϵ_r	$\approx 4,365$	$\approx 2,079$

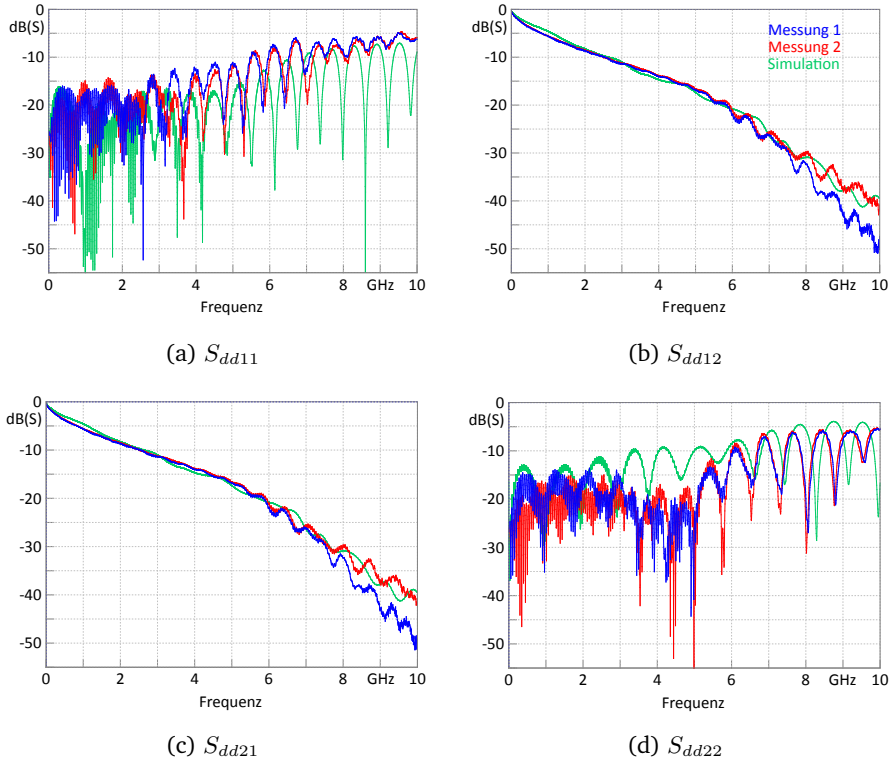
$$v = \frac{1}{\sqrt{\epsilon_0 \epsilon_r \mu_0 \mu_r}} \tag{5.16}$$

Unter Vernachlässigung der relativen magnetischen Permeabilität ($\mu_r = 1$) und gemäß Beschreibung der Lichtgeschwindigkeit nach den Maxwell'schen Gleichungen in Abhängigkeit von der elektrischen Feldkonstante ϵ_0 und der magnetischen Feldkonstante μ_0 ergibt sich für die relative elektrische Permittivität die vereinfachte Gleichung 5.17.

$$\epsilon_r = \left(\frac{c}{v}\right)^2 \tag{5.17}$$

Zusätzlich zur Verifikation der Materialkonstanten und den davon abhängigen berechneten Impedanzen des Lagenaufbaus der Leiterkarte wird das eigentliche Übertragungsverhalten der Kommunikationsstrecke anhand von Messungen mit den Ergebnissen von der Fertigung der Leiterkarte vorangegangener Simulationen mit Hilfe der Software-Werkzeuge Hyperlynx SI von Mentor Graphics und HSPICE der Firma Synopsys verglichen. Dazu wurden verschiedene Messreihen an Übertragungsstrecken mit unterschiedlicher Länge angefertigt und deren differentielle Streuparameter (S-Parameter) aufgenommen und übereinander gelegt. Zur Überprüfung der Messung wird jeweils eine Aufnahme in beide Richtungen vorgenommen, die im Ergebnis wie in Abbildung 5.14 die Symmetrie der inklusive 2-m-EQDP-Kabel 2201,1 mm langen Übertragungsstrecke nachweist.

Auftretende Kontaktierungsschwierigkeiten der differentiellen Messspitzen lassen sich ab einer Frequenz von 7 GHz insbesondere bei den Reflexionsfaktoren S_{dd11} und S_{dd22} beobachten. Zudem ist an den Transmissionsfaktoren S_{dd12} und S_{dd21} erkennbar, dass bei einer Übertragung mit der maximalen Datenrate der Virtex-5-Transceiver von 6,5 Gbit/s, also einer Grundfrequenz von 3,25 GHz eine Dämpfung von 12,5 dB auftritt, was dank Eingangsentzerrung der Vitesse VSC3008 Crosspoint-Switche und einer entsprechend angepassten Vorverzerrung eine fehlerfreie Kommunikation ermöglicht. Für höhere Datenraten bis hin zur maximalen Übertragungsrate der Crosspoint-Switche von 11,5 Gbit empfiehlt sich der Einsatz der VSC3308, da diese bessere Möglichkeiten zur Signalkonditionierung bereitstellen



(a) S_{dd11} (b) S_{dd12}

(c) S_{dd21} (d) S_{dd22}

Abbildung 5.14: differentielle S-Parameter einer MGT-Strecke mittlerer Länge

und einen höheren Aussteuerungsbereich der Empfangsentzerrung von bis zu 20 dB bieten.

Messungen an Übertragungsstrecken mit einer Länge von 71,6 mm und 465,5 mm auf der Leiterkarte plus einem 2-m-EQDP-Kabel bestätigen das Ergebnisbild der mittellangen Strecke. Zusammenfassend zeigt sich, dass die Simulationsergebnisse den Resultaten der Messreihen ausreichend entsprechen und somit qualitative Aussagen über die zu erwartende Signalqualität der MGT-Übertragungsstrecken erlauben [286].

5.2.2 Links-Rechts-Verbinder

Für die Kommunikation benachbarter Module werden insgesamt 66 lose gekoppelte differentielle Signale auf dem RAPTOR-XPress geführt, wodurch sich eine Ringto-

pologie ergibt. Unter Verwendung eines DB-V5 kann so eine maximale Bandbreite von 90 Gbit/s und bis zu 100 Gbit/s mit dem DB-V7 im source-synchronen DDR-Betrieb erreicht werden [241]. Der Vorteil dieser Verbindung ist, dass breitbandige Kommunikationsstrecken mit geringen Latenzen realisiert werden können, da zum einen kein Mehraufwand durch ein vorgegebenes Protokoll entsteht, zum anderen eine Serialisierung der Daten nicht zwingend erforderlich ist. Die Verbinder besitzen keinen Anschluss an weitere Komponenten des Basisboards. Aus diesem Grund ist es möglich, die Spannungspegel für die Kommunikation durch die eingesetzten DBs aushandeln zu lassen. Die für das RAPTOR-XPress entworfenen DBs wie das DB-V5 verfügen zu diesem Zweck über eine konfigurierbare Spannungsversorgung für die angeschlossenen IO-Bänke und unterstützen einen Aussteuerungsbereich von 1,2 V bis 3,3 V. Dies ermöglicht auch das Einbinden ältere Module, die fest mit 3,3 V-Logikpegeln arbeiten oder nur Single-Ended-Verbindungen zulassen[232].

5.2.3 Broadcast-Bus und Config-Bus

Für die schnelle Konfiguration der FPGAs der Erweiterungsboards wird das 32 bit breite Slave SelectMAP Interface genutzt [242]. Auf diese Weise kann beim maximalen Konfigurationstakt von 100 MHz ein Virtex-5 FX100T [241], der auf dem DB-V5 Verwendung findet, innerhalb von ca. 13 ms vollständig rekonfiguriert werden.

Nach Abschluss der Konfiguration des Modul-FPGAs können die Verbindungen der Schnittstelle mit den bereits vorhandenen 46 Leitungen kombiniert werden, um eine insgesamt 78 bit breite Punkt-zu-Punkt-Kommunikationsstrecke zu einem Kontroll-FPGA auf dem RAPTOR-XPress-Basisboard zu realisieren. Neben der Überwachung und Steuerung der Modulkonfiguration ist es Aufgabe des Kontroll-FPGA, einen Switch zu realisieren, der die vier beteiligten Module miteinander verbindet. So ist es möglich, in der vorhandenen Ringtopologie nicht benachbarte Module direkt miteinander zu koppeln oder Broad- und Multicast-Verbindungen mit geringer Latenz aufzubauen. Unidirektional lässt sich bei Verwendung des DB-V5 und differentieller source-synchroner DDR-Übertragung eine Übertragungsrate von 10 Gbit/s erzielen. Wie bereits im vorhergehenden Abschnitt 5.2.2 aufgezeigt, ist es sinnvoll, unterschiedliche Logikpegel bei der Kommunikation zwischen den Modulen zu unterstützen. Da die Verbindung über einen FPGA des Basisboards erfolgt, muss bei der Aushandlung der Spannung die Spannungskontrolle des Trägersystems mit einbezogen werden. Eine Aufteilung der vier Module auf je eine Bank des eingesetzten Xilinx Spartan-3-FPGA, was eine individuell einstellbare IO-Spannung für jedes Modul erlauben würde, scheitert an der bausteinbedingten ungleichen Verteilung der IOs auf die einzelnen Bänke sowie an der Unterteilung des FPGAs in zwei unabhängige Takt-Hemisphären mit nur wenigen globalen Taktein- und -ausgängen.

Gesteuert wird der Bus über einen weiteren Xilinx Spartan-3 1400AN FPGA, dessen Aufgabe es ist, die Konfigurationsleitungen zu verwalten sowie eine Verbin-

dungsstruktur für die Modulkommunikation zu realisieren. Benötigt werden die Konfigurationsleitungen nur, wenn die Informationen nicht über die PCIe-Schnittstelle direkt an das DB-V5 übertragen werden können oder sollen. Außerhalb des Konfigurationsprozesses können die Leitungen ergänzend zu den Datenleitungen für die anwendergesteuerte Kommunikation zwischen den Modulen genutzt werden.

5.2.4 PCIe

Für die Hostkommunikation wird PCIe der zweiten Generation mit einer Datenrate von 5 GT/s eingesetzt. Die Anbindung über den PCIe-Switch PEX8648 des Herstellers Broadcom (ehemals PLX-Technology) ermöglicht eine Verbindung der vier Daughterboards des RAPTOR-XPress als Endpoint über eine 8x-Lane zum Rootcomplex des Hostrechners. Für weiterführende Informationen zur Organisation von PCIe sei auf [243] verwiesen.

Da Zugriffe auf den Hauptspeicher des Hostsystems über den Rootcomplex erfolgen, ist für Anwendungen, die demzufolge große Datenmengen mit dem Prozessor austauschen müssen, eine Anbindung über die FSB-FPGA-Plattform zu bevorzugen. Der Zugriff über den FSB erlaubt eine dem Hauptprozessor der Architektur gleichgestellte Bandbreite und Latenz bei Operationen auf dem Hauptspeicher.

Um abwärtskompatibel zu den DBs zu bleiben, die für die Vorgängerplattformen RAPTOR2000 und RAPTOR-X64 entwickelt wurden, wird die PLX-Technology PEX8311 PCIe-zu-Local-Bus-Bridge verwendet. Angeschlossen als 1x-Lane-Einheit an den PCIe-Switch erfolgt intern eine Protokollumwandlung zu PCI-X und darauf folgend in den Local-Bus-Standard [244]. Diese Art der Umsetzung ermöglicht eine Portierung der Einstellungen des PCI9656 [245], entsprechend seines Einsatzes auf dem RAPTOR-X64. Auf diese Weise ist es möglich, alte Module zu rekonfigurieren und Daten mit dem Host auszutauschen. DMA-Übertragungen sind über diese Schnittstelle ebenfalls möglich.

Eine mechanisch relevante Rahmenbedingung wird durch die Verwendung des PCIe-Standards in Bezug auf die maximale Dicke des Boards vorgegeben. Definiert worden ist die Dicke des Steckerbereichs mit 1,57 mm durch den PCIe-Standard. Bei einer Toleranz von ±0,13 mm ergibt sich eine maximale Dicke von 1,7 mm [246]. Bei der Auslegung des Lagenaufbaus für das RAPTOR-XPress ergeben die Anforderungen an die spezifischen Impedanzen für die Leiterbahnen und die Menge an Signalen eine Lagenanzahl von insgesamt 24 inklusive der stromführenden Vollkupferlagen mit einer Gesamtdicke von 3,075 mm. Da die Abstände der Lagen, insbesondere der Referenzvollkupferlagen zu den Signallagen, maßgebend für die Impedanz der Leitungen sind, lässt sich die geforderte Dicke nicht vollflächig für die gesamte Leiterkarte realisieren. Um dennoch mechanisch kompatibel zu in Servern und PCs üblichen PCIe-Steckplätzen zu sein, ist es notwendig, nach der Fertigung Lagen im Steckerbereich zu entfernen, so dass die ursprünglichen Innenlagen 4 und 9 die Goldsteckkontakte stellen, siehe Abbildung A.2. Mit Hilfe eines Separators,

der das Prepreg-Material im Bereich der Rückfräsung ersetzt, können nach dem letzten Verpressvorgang in der Herstellung die aufbauenden Lagen zurückgefräst und anschließend die Kontaktflächen als Hartgoldoberfläche galvanisiert werden.

5.2.5 Local-Bus

Der Local-Bus-FPGA, ebenfalls ein Xilinx Spartan-3 1400AN, übernimmt mehrere Aufgaben. Zum einen ist er das Bindeglied zwischen der PCIe-Schnittstelle PEX8311 und den einzelnen Modulen. Die Architektur folgt dabei dem bereits auf dem RAPTOR2000 implementierten Aufbau und setzt das Local-Bus-Protokoll nach [247] um. Eine Übersicht über die Belegung und das Verhalten der Local-Bus-Schnittstelle des PEX8311 wird in [248] gegeben. Der FPGA tritt als Master am Bus auf und sorgt für die Arbitrierung. Des Weiteren dient der FPGA zum Einbinden der USB-Slave-Schnittstelle in das Gesamtsystem. Somit ist es möglich, über USB vom Hostsystem aus Befehlssequenzen auf den Local-Bus zu schreiben. Dies kann unter anderem zum Debuggen von FPGA-Designs genutzt werden. Abschließend leistet der FPGA die folgenden Überwachungs- und Steuerungsaufgaben.

- **Port-Statussignale des PCIe-Haupt-Switch PEX8648** – Folgende Informationen über den Link-Status können durch den FPGA detektiert werden [249]:
  - Aus
  - 5,0 GT/s; alle Lanes aktiv
  - 5,0 GT/s; reduzierte Lane-Anzahl aktiv
  - 2,5 GT/s; alle Lanes aktiv
  - 2,5 GT/s; reduzierte Lane-Anzahl aktiv
- **Reset-Kontrolle des PCIe-Switch PLX Technology PEX8648** – Die Schnittstelle ermöglicht eine vom Hostsystem unabhängige Kontrolle des PCIe-Switches.
- **Steuerung der PCIe-zu-Local-Bus-Brücke PLX Technology PEX8311** – Zur Sicherstellung der Local-Bus-Funktionalität kann die Schnittstelle zum Hostsystem direkt von der RAPTOR-XPress-Firmware angesteuert werden.
- **Steuerung der JTAG-Schnittstellen über einen TI SCANSTA112** – Zum gezielten Herausnehmen einzelner Teilnehmer aus der JTAG-Ketter kann über diese Schnittstelle der JTAG-Switch SCANSTA112 gesteuert werden.
- **Interface zum PIC-Mikrocontroller** – Die Schnittstelle dient der Statusabfrage der Spannungsversorgungsüberwachung.

- **Statusinformationen** – Zu Informations- und Debugging-Zwecken werden durch den FPGA verschiedene Leuchtdioden, beispielsweise für Busfehler auf dem RAPTOR-XPress-Trägersystem, sowie für jedes Modul zwei Status-LEDs angesteuert.

5.2.6 Weitere Kommunikationsschnittstellen

Zu den bisher genannten Schnittstellen sowie deren Kontrolllogik kommen weitere Kommunikationsschnittstellen hinzu. Neben der Übermittlung von Steuerbefehlen an verschiedene Bereiche auf dem RAPTOR-XPress sowie an die Modul-Steckplätze existieren Standardschnittstellen, um eine einfache Benutzbarkeit und Fernwartbarkeit durch den Anwender zu ermöglichen.

5.2.6.1 Steuer- und Überwachungsbussysteme

Für Steuer- und Überwachungsaufgaben befinden sich mehrere unabhängige Zweidraht-I2C-Bussysteme auf dem Basisboard. Die einzelnen Aufgabenbereiche werden im Folgenden kurz erläutert.

- **Modulidentifizierung** – Die Identifizierung eines aufgesteckten Moduls erfolgt über den Local-Bus-FPGA, welcher die Daten an den Benutzer weiterreicht.
- **Einstellen der Spannung verschiedener Übertragungsstrecken** – Die Busspannung des Broadcast-Busses und Config-Busses (BCB) sowie die Spannungspegel für die Links-Rechts-Verbinder können über digitale Potentiometer an die jeweiligen Aufgabenbereiche angepasst werden. Die Ansteuerung hierfür erfolgt über den PIC-Mikrocontroller und wird in Abschnitt 5.2.8 erläutert.
- **Konfiguration des PEX8648** – Der PCIe-Switch verfügt über zwei I2C-Interfaces. Zum einen können über das Slave-Interface sämtliche Register des Bausteins beschrieben und ausgelesen werden, zum anderen ist es möglich, die Verbindungsqualität der PCIe-Verbindungen abzufragen. Die zweite I2C-Schnittstelle ist ausschließlich für die im PCIe-Standard vorgesehenen Hot-Plug-Signale reserviert [249].

5.2.6.2 USB

Zur Anbindung von unterschiedlichen USB-Komponenten wie Speichersticks, Eingabegeräte oder Kameras befindet sich eine USB-A-Buchse auf dem Basisboard. Umgesetzt wird die USB-Signalführung durch einen Cypress CY7C67300-Controller, der aufgrund des integrierten Prozessors als USB-Host-Interface dient. Da eine Anbindung von Peripheriegeräten direkt in Hardware nicht übergreifend möglich

ist – speziell, wenn unterschiedliche Geräteklassen wie Eingabegeräte, Speicher, Kamerasysteme etc. unterstützt werden sollen – findet die Einbindung in das System über den MicroBlaze-FPGA statt. Da dieser die Plattform für ein μLinux bildet, können die USB-Schnittstelle und die daran angeschlossenen Geräte durch Treiberunterstützung eingebunden werden.

Neben dem Host-Interface befindet sich eine Micro-USB-Buchse auf dem RAPTOR-XPress, über den eine USB-Slave-Schnittstelle realisiert wird. Der FX3-USB-Controller von Cypress übernimmt dabei die Signal- und Protokollumsetzung. Verbunden ist der Controller mit dem Local-Bus-FPGA.

5.2.6.3 Ethernet

Durch einen VSC8641 PHY-Baustein der Firma Vitesse, der ebenfalls mit dem MicroBlaze-FPGA verbunden und somit durch dessen μLinux in das System eingebunden ist, wird eine Gigabit-Ethernet-Schnittstelle realisiert. Die Einbindung einer Netzwerkschnittstelle ermöglicht die Durchführung von Fernwartungsaufgaben auf dem eingebetteten Linux-System, die Überwachung der Betriebszustände oder eine Nutzung ohne Host-PC innerhalb eines Netzwerks. Darüber hinaus kann Netzwerkperipherie eingebunden werden, wodurch beispielsweise IP-Kameras als Datenquelle genutzt werden können.

5.2.6.4 JTAG

Zur Konfiguration der FPGAs befindet sich eine JTAG-Kette auf dem RAPTOR-XPress. Diese wird über den SCANSTA112-JTAG-Multiplexer von National Semiconductor sowohl an die FPGAs des Trägersystems als auch an die Module weitergereicht. Der Einsatz eines Multiplexers ermöglicht es, einzelne FPGAs aus der Kette auszublenden und somit Unterbrechungen zu maskieren, die entstehen würden, wenn Modulsteckplätze nicht bestückt werden.

5.2.7 Spannungsversorgung

Eine stabile und ausreichende Spannungsversorgung der einzelnen Komponenten ist die Voraussetzung für einen zuverlässigen Betrieb. Besonders im Hinblick auf die Versorgung der MGT-Ausgänge dürfen kurzzeitige Einbrüche in der Betriebsspannung, wie sie typischerweise bei Schaltvorgängen auftreten, nicht vorkommen. Aufgrund der eingesetzten Schaltregler ist außerdem darauf zu achten, dass sich betriebsartbedingt die jeweilige Frequenz der Schaltregler im Ausgangsbild der Spannung wiederfindet und eine entsprechende Filterung zur Glättung benötigt wird. Insbesondere Bausteine, die empfindlich auf verrauschte Versorgungsspannungen reagieren, wie zum Beispiel diskrete und FPGA-interne Taktgeneratoren, ist neben der konstanten Überwachung der erzeugten Spannungen eine genaue

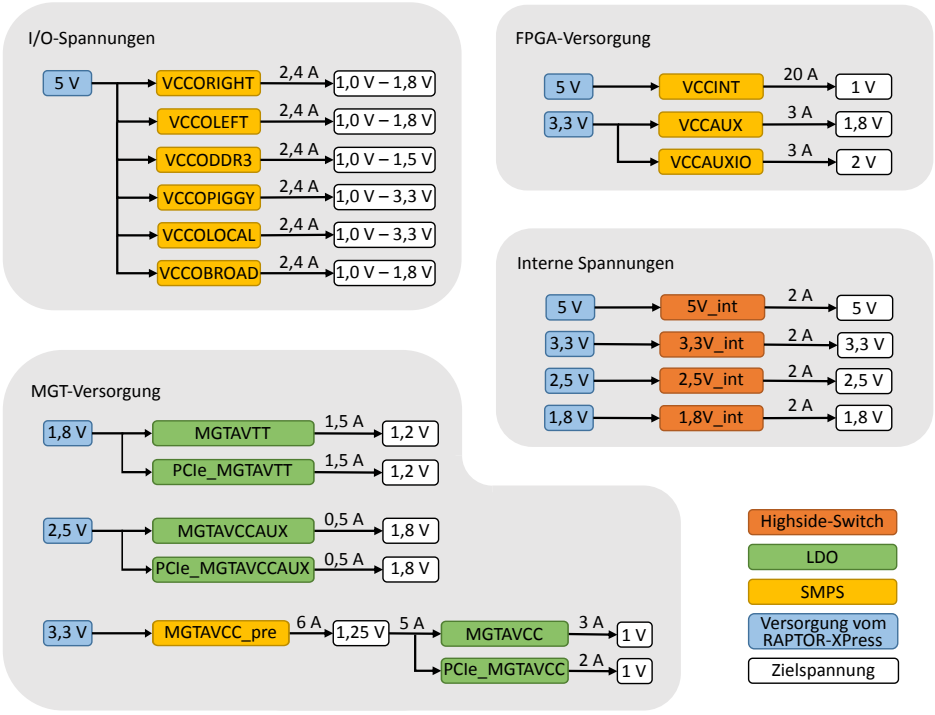


Abbildung 5.15: Spannungshierarchie des DB-V7 [251]

Simulation bei der Auslegung der Schaltregler und eine anschließende Verifikation des Verhaltens anhand der produzierten Baugruppe notwendig.

Die Effizienz der Spannungsversorgung ergibt sich aus dem Produkt der Effizienz der einzelnen kaskadierten Spannungswandler. Dabei spielt neben einer ausreichenden Stromversorgung der angeschlossenen Komponenten der bereits angesprochene Aspekt der Rauscharmut eine wichtige Rolle in Bezug auf die Auswahl der eingesetzten Wandlertechnologie und der damit einhergehenden Effizienz. Besonders rauscharme Linearregler schneiden in ihrer Effizienz schlechter ab als effizientere und höhere Stromstärken erreichende Schaltregler [250].

Das RAPTOR-XPress stellt den vier Modul-Steckplätzen die belastbaren vier Grundspannungen 5 V, 3,3 V, 2,5 V und 1,8 V zur Verfügung, so dass lokal auf den DBs die für den Betrieb erforderlichen Spannungen erzeugt werden können. Diese Aufteilung der Spannungsversorgung ermöglicht eine platz- und verlustleistungsoptimierte DB-Realisierung. Neben der eigentlichen Versorgung wird abhängig von der Bestückung der einzelnen Steckplätze die IO-Spannung für die gemeinsam

Tabelle 5.3: Spannungsübersicht des RAPTOR-XPress

Quelle	Spannung	Funktion
Power-Piggy-Back	5 V	Grundspannung
	5 V	Systemüberwachungs-Mikrocontroller
	3,3 V	Grundspannung
	2,5 V	Grundspannung
	2,5 V	Crosspoint-Switch-Array
	1,8 V	Grundspannung
RAPTOR-XPress	3,3 V	FPGAs: IO und Peripherie
	2,5 V	PCIe-Switch und PCIe-Local-Bus-Brücke
	1,8 V	DDR2
	1,5 V	PCIe-Local-Bus-Brücke
	1,2 V	RAPTOR-XPress-FPGAs
	1,0 V	PCIe-Switch: Digitalteil
	1,0 V	PCIe-Switch: Analogteil
	0,9 V	DDR2-Terminierung
	0,9 V bis 3,5 V	Local-Bus-IO
	0,9 V bis 3,5 V	BCB-IO

genutzten Kommunikationsstrukturen BCB und Links-Rechts-Verbinder festgelegt. Die lokale Spannungswandlung der DB-Betriebsspannungen verhindert zum einen, dass Gleichspannungsverluste aufgrund langer Transportwege über das gesamte RAPTOR-XPress kompensiert werden müssen, und zum anderen, dass einkoppelnde Störungen durch die Wandlung geglättet werden. Durch das modulare Konzept ist außerdem nicht abzusehen, welche Anforderungen an die Versorgungsspannung gestellt werden. Hierarchien beim Einschalten der einzelnen Betriebsspannungen variieren dabei gleichermaßen wie die eigentlichen zur Versorgung erforderlichen Spannungswerte. Abbildung 5.15 gibt eine Übersicht über die Spannungshierarchie der lokal auf dem DB-V7 erzeugten Spannungen. Überwacht werden die einzelnen Spannungen ebenfalls modulintern auf den DBs.

Neben den Modulgrundspannungen und der separaten 2,5-V-Versorgung des Crosspoint-Switch-Array, die zusammen auf dem Power-Piggy-Back-Versorgungsmodul erzeugt werden, müssen lokale Versorgungsspannungen der RAPTOR-XPress-eigenen Komponenten aufbereitet werden. Tabelle 5.3 gibt einen Überblick über die benötigten Spannungspegel [287].

5.2.8 Systemsteuerung und -überwachung

Eine kontinuierliche Überwachung der Betriebsparameter erlaubt mögliche Fehlerzustände abzufangen und die Betriebssicherheit der eingesetzten Komponenten zu

gewährleisten. Zu den aufgenommenen Werten gehört neben den in Tabelle 5.3 aufgeführten Spannungspegeln die Temperatur an sechs Positionen auf dem RAPTOR-XPress einschließlich der Die-Temperatur des PCIe-Switch. Überwacht werden die Parameter mit Hilfe eines PIC-Mikrocontrollers, der auch die Einschaltsequenz der einzelnen Spannungen realisiert.

MicroBlaze-FPGA

Der Kern des intelligenten Trägersystems RAPTOR-XPress ist ein Xilinx Spartan-3 1400AN, der MicroBlaze-FPGA, welcher eine MicroBlaze-Prozessor-Architektur implementiert. Bei dem Xilinx MicroBlaze-Prozessor handelt es sich um eine 32-bit-RISC-Harvard-Prozessorarchitektur, die speziell für den Einsatz auf Xilinx FPGAs optimiert wurde [252]. Ergänzt wird der FPGA durch einen 512-Mibit-NOR-Flash-Speicher der Firma Micron. Als nichtpersistenter Arbeitsspeicher können zwei MT47H64M16 DDR2-SDRAM-Bausteine, ebenfalls von Micron, mit jeweils 1 Gibit genutzt werden. Die beiden Speicher sind datenparallel angeschlossen und implementieren so ein 32 bit breites Interface. Hauptaufgabe des FPGAs ist die Konfiguration des Crosspoint-Switch-Array. Hierüber realisiert der MicroBlaze-FPGA die Gegenstelle zum USB-Host-Interface und sorgt für die Einbindung der Ethernet-Schnittstelle. Für den Datenaustausch mit dem Local-Bus-FPGA ist eine 36 bit breite Übertragungsstrecke vorhanden. Diese wurde mit den für beide Richtungen vorgesehenen Taktverbindungen für differentielle Kommunikation ausgelegt. Zur einfachen Umsetzung der Aufgaben wird ein μLinux-System auf dem implementierten Prozessorsystem betrieben.

5.3 Erweiterungsmodul DB-V5

Parallel zum RAPTOR-XPress wurde das FPGA-Modul DB-V5 im Hinblick auf eine vollständige Unterstützung aller angebotenen Infrastrukturen entwickelt. Kernlogikelement des DB-V5 ist, wie in Abbildung 5.16 zu sehen, ein Virtex-5-FX100T-FPGA [237], welcher die Benutzerlogik implementiert. Verschiedene Bestückungsoptionen durch kompatible FPGAs, angefangen beim XC5VLX50T bis hin zum eingesetzten XC5VFX100T, erlauben kostengünstige und auf ein spezielles Anwendungsfeld angepasste Fertigungsversionen des Erweiterungsmoduls. Die Unterschiede der einzelnen FPGAs werden in Tabelle 5.4 aufgeführt [285].

Unterstützt wird der FPGA durch einen weiteren Virtex-5-LX30T-FPGA, welcher mit seinen GTP-Transceivern den PCIe-Endpoint realisiert und breitbandig an den FX100T gekoppelt ist. Diese Partitionierung ermöglicht dem Benutzer einen abstrahierten Zugriff auf die PCIe- und die Local-Bus-Schnittstelle. Da die Kopplung der beiden FPGAs über reguläre IOs geschieht, stehen die 16 GTX-Verbinder für die

Tabelle 5.4: Übersicht über die DB-V5-Bestückungsoptionen

	LX50T	LX85T	LX110T	LX155T	SX50T	SX95T
Slices	7200	12960	17280	24320	8160	14720
BRAM	2160	3888	5328	7632	4752	8784
Kibit						
DSP48E	48	48	64	128	288	640
MGT	GTP	GTP	GTP	GTP	GTP	GTP
	12	12	16	16	12	16

	FX70T	FX100T
Slices	11200	16000
BRAM	5328	8208
Kibit		
DSP48E	128	256
MGT	GTX	GTX
	16	16

Anbindung an das Crosspoint-Switch-Array des RAPTOR-XPress-Trägersystems zur Verfügung.

2×36 IOs des LX30T FPGA werden auf Steckern der Firma Samtec hinausgeführt, wodurch Piggyback-Erweiterungsmodule aufsteckbar sind. Beispiele hierfür sind:

- Gigabit-Ethernet-Schnittstellen
- ADC-Erweiterungen
- Adapter auf andere Steckerstandards für digitale GPIOs

Für Anwendungen, die nicht mit dem im Benutzer-FPGA vorhandenen BRAM realisierbar sind, steht ein DDR2-SODIMM-Sockel zur Einbindung von externem Speicher zur Verfügung. Status- oder Debug-Informationen können auf der 5×5-LED-Matrix ausgegeben werden.

Die Konfiguration der Kommunikationsspannung im Bereich von 1,2 V bis 3,5 V ermöglicht den Einsatz einer Vielzahl unterschiedlicher FPGA-Typen. Nach dem Einschalten und der damit verbundenen Einschaltverzögerung aufgrund des kontrollierten Hochfahrens der Versorgungsspannungen ist es im Hinblick auf eine erfolgreiche Teilnahme an der Enumeration des Root-Complex erforderlich, dass der PCIe-Endpoint innerhalb von 120 ms betriebsbereit antwortet. Die Konfigurationsschnittstelle muss somit die Bedingung 5.18 erfüllen. Um sicherzustellen, dass die Konfiguration des LX30T innerhalb der durch die PCIe-Spezifikation geforderten Zeit abgeschlossen ist, bezieht der FPGA seine Konfigurationsdaten aus einem Xilinx

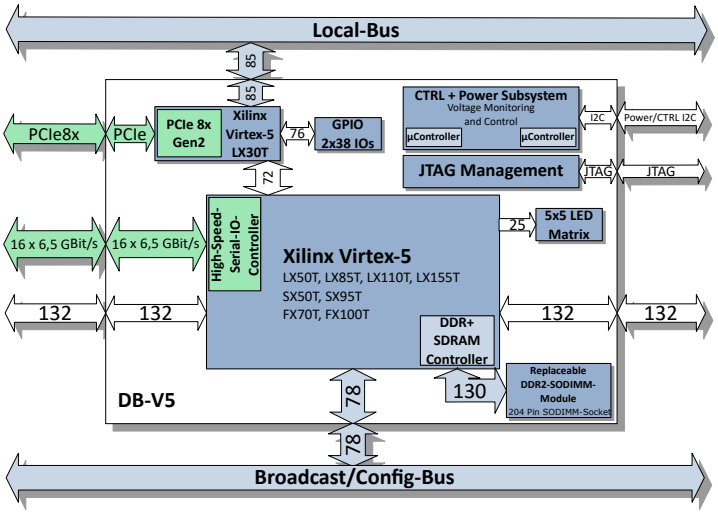


Abbildung 5.16: Architekturübersicht des DB-V5

XCF32P 32-Mibit-Platform-Flash, welcher 8 bit breit angebunden ist. Bei einer Bitstream-Größe von 8,94 Mibit (Virtex-5 FX30T) und einer maximalen Taktrate der Schnittstelle von 33 MHz ist die Konfiguration innerhalb von 36 ms abgeschlossen und der PCIe-Endpoint kann regulär an der Enumeration durch den Rootcomplex teilnehmen.

$$\text{Einschaltverzögerung} + \frac{\text{Bitstreamsize [bit]}}{\text{Interface width [bit]} \cdot \text{Interface speed [Hz]}} < 120\,\text{ms} \tag{5.18}$$

5.4 Erweiterungsmodul DB-V7

Als Weiterentwicklung des DB-V5 ist im Fachgebiet *Kognitronik und Sensorik* das DB-V7 entstanden [251]. Der grundlegende Aufbau des Moduls unterscheidet sich nur geringfügig von seinem Vorgänger, wie in Abbildung 5.17 dargestellt. Der eingesetzte Virtex-7 lässt ebenfalls durch FFG1157-Gehäuse-kompatible Bausteine Bestückungsoptionen für den Benutzer-FPGA vom XC7VX330T bis zum eingesetzten XC7VX690T zu. Die spezifischen Eigenschaften der einsetzbaren Virtex-7-FPGAs werden in Tabelle 5.5 aufgeführt. Ergänzt wird der Benutzer-FPGA um einen Kintex-7 XC7K70T als Schnittstellen-FPGA zur PCIe-Kommunikationsinfrastruktur, zum Local-Bus und zu den Samtec Erweiterungskonnektoren. Aufgrund der gegenüber der

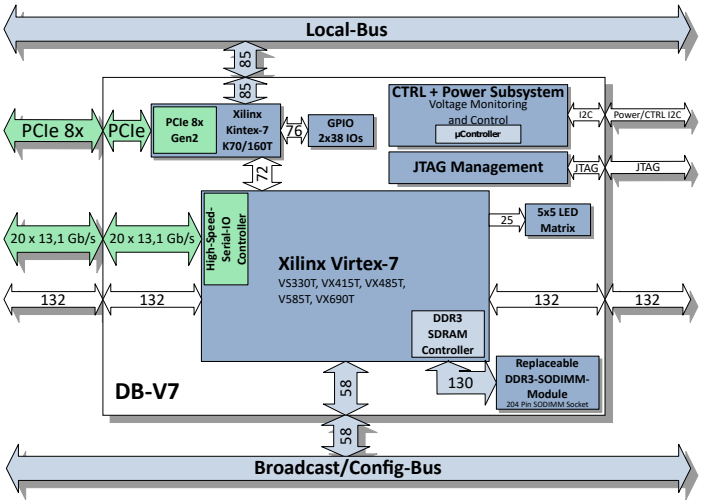


Abbildung 5.17: Architekturübersicht des DB-V7

Virtex-5-Serie erweiterten IO-Bänke des Virtex-7 ist es möglich, 20 GTH-Transceiver für die MGT-Kommunikation vorzusehen und somit auch die Kommunikationsinfrastruktur eines potentiell erweiterten RAPTOR-XPress-Basissystems zu nutzen. Entsprechend wurde die Speichererweiterung von DDR2-SODIMM-Modulen auf die DDR3-Speichertechnologie umgestellt.

Hervorzuheben bei der Auswahl des FFG1157-Gehäuses ist, dass sämtliche Benutzer-IOs dem HP-Standard entsprechen und somit nur IO-Standards im Rahmen von 1,2 V bis 1,8 V unterstützen. Relevant ist dieser Umstand, wenn beispielsweise ein DB-V5 und ein DB-V7 auf benachbarten Modulplätzen Daten miteinander austauschen müssen. Übertragungsstandards mit 2,5 V oder 3,3 V können nur mit

Tabelle 5.5: Übersicht über die DB-V7-Bestückungsoptionen

	VX330T	**VX415T**	**VX485T**	**VX690T**	**V585T**
Slices	51000	64400	75900	108300	91050
BRAM	27000	31680	37080	52920	28620
Kibit					
DSP48E1	1120	2160	2800	3600	1260
MGT	GTH	GTH	GTX	GTH	GTX
	20	20	20	20	20

HR-IOs umgesetzt werden, die allerdings nicht zur Verfügung stehen. Beim Vergleich der FPGA-Slices von Virtex-5- und Virtex-7-FPGAs ist zu beachten, dass sich die Struktur und die vorhandenen Elemente geändert haben. Ein Virtex-5-Slice beinhaltet vier Blöcke mit LUTs und vier Flip-Flops, während Virtex-7-Slices über vier zusätzliche Flip-Flops, also insgesamt acht Flip-Flops pro Slice verfügen. Für eine genauere Übersicht über den Aufbau sei auf [237] und [19] verwiesen.

5.5 Administration und Diagnose

Neben den plattforminternen Hardware-Komponenten zur Administration des Systems wie dem Local-Bus- und dem BCB-FPGA sowie den System-Mikrocontrollern, bietet die Host-basierte Software-Umgebung RaptorSuite die Möglichkeit, sämtliche Funktionen zu steuern und zu überwachen. Die Hauptfunktionen werden im Folgenden aufgelistet:

- DB-Programmierung
- Local-Bus-Verwaltung einschließlich Fehlererkennung und Bereinigung fehlerhafter Zustände
- JTAG-Verwaltung
- Konfiguration der BCB-Kommunikationsinfrastruktur
- Konfiguration des Crosspoint-Switch-Array
- Zugriffe auf die DB-Speichermodule
- Infrastruktur-Tests

Dabei ist nicht nur eine Verwaltung von lokalen, im Host vorhandenen Raptor-Systemen möglich, sondern auch die entfernte Kontrolle mittels einer Server-Client-Architektur auf einem weiteren im Netzwerk verbunden Host oder eine direkte Verbindung eines RAPTOR-XPress über Ethernet.

5.6 RAPTOR-XPress-Cluster

Die Kombination von mehreren flexibel miteinander verschaltbaren FPGAs ermöglicht den Aufbau einer leistungsfähigen Plattform für vielfältige Aufgabenstellungen. Eng gekoppelte Strukturen auf einem Trägersystem, hergestellt durch Rechts-Links-Verbinder und Broadcast-Bus, werden durch breitbandige Verbindungen über das Crosspoint-Switch-Array zu anderen Basisboards erweitert und durch direkt an den

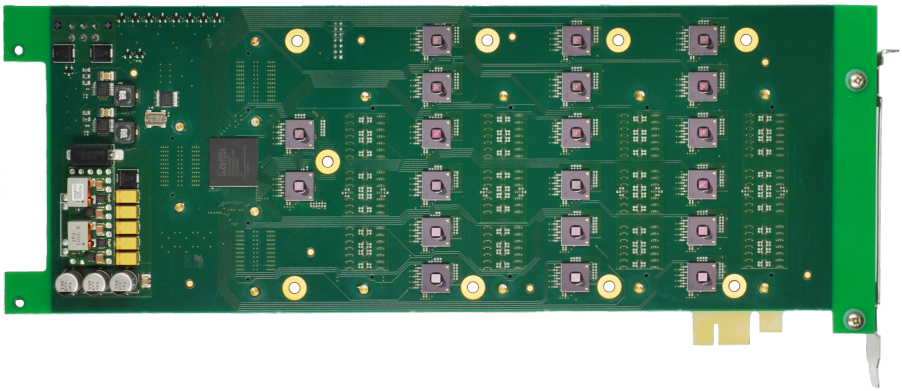


Abbildung 5.18: Crosspoint-Switch-Array der FSB-MGT-Bridge

FSB angebundene FPGAs innerhalb von Intel Xeon Server-Systemen der Firma Nallatech ergänzt. Eine Besonderheit dieser Server ist, dass einer der Prozessorsockel mit einem angepassten FPGA-Basis-Modul belegt ist [253]. Die Einbindung dieser speziellen FPGA-Module erfolgt über die im Rahmen einer parallel angefertigten Arbeit [286] entworfene und in Abbildung 5.18 dargestellte FSB-MGT-Bridge.

Aufgrund der gleichen Crosspoint-Switch-Array-Organisation gliedert sich das System in die Cluster-Topologie wie ein RAPTOR-XPress ein. Abbildung 5.19 zeigt schematisch die Integration in das Gesamtsystem.

Weitergehende Informationen zur Nallatech-FSB-Plattform werden in [253] aufgeführt.

5.7 Anforderungsverifikation

Die bereits für das RAPTOR2000 formulierten Anforderungen an ein skalierbares Trägersystem für FPGAs [232] wurden, wie im Folgenden erläutert, konsequent umgesetzt und erweitert. Anhand dieser Konzepte lässt sich auch das Alleinstellungsmerkmal der Plattform gegenüber auf dem Markt verfügbaren Systemen aufzeigen.

Verkürzter Entwurfszyklus

Die in dieser Arbeit vorgestellten Methoden und die Software-Lösung RaptorSuite tragen zur Verkürzung der Entwurfszeit für auf mehrere FPGAs verteilte Applikationen bei. Die Architektur des RAPTOR-XPress unterstützt mit ihren vielseitigen

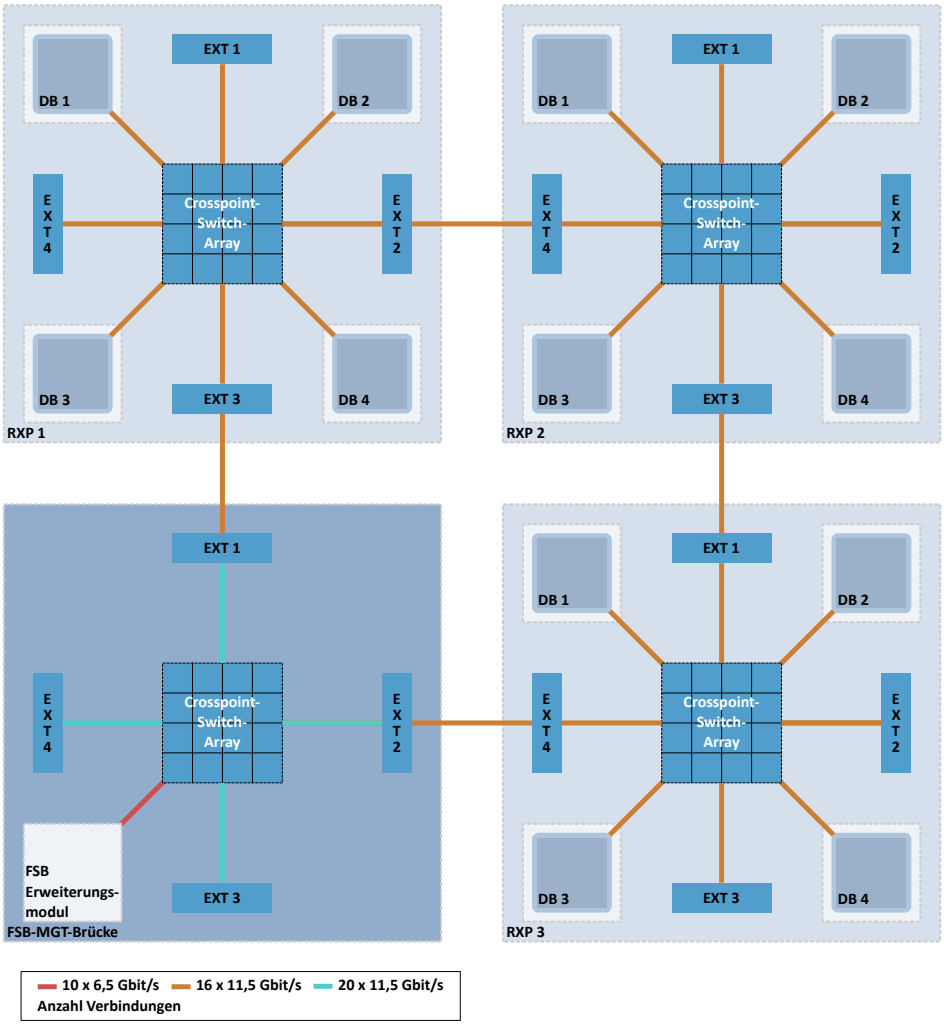


Abbildung 5.19: Beispielkonfiguration der MGT-Kommunikationsinfrastruktur mit drei RAPTOR-XPress und einer FSB-MGT-Bridge

Verbindungsmöglichkeiten die Partitionierung. Durch die Bereitstellung von Schnittstellen wie Inter-FPGA [281] wird der Übergang zwischen zwei FPGAs transparent gehalten. Darüber hinaus können die in [282] vorgestellten Verfahren zur schnellen Entwurfsraumexploration durch die architekturseitige Unterstützung von dynamischer Rekonfiguration angewendet werden.

Modularität

Die Trägersysteme der RAPTOR-Familie zeichnen sich durch ihre Modularität aus. Das Konzept wurde beim RAPTOR-XPress konsequent ausgebaut. Die vormals festgesetzte Spannungsversorgung wurde modular ausgelegt, um veränderte Anforderungen durch neue DBs erfüllen zu können. Die externe Anbindung anderer RAPTOR-XPress ermöglicht eine hohe Flexibilität bei der Erstellung der benötigten Zielarchitektur. Bei der Einbindung weiterer RAPTOR-XPress oder anderer externer Komponenten wie FSB-FPGA-Systeme können klassische kupfergebundene Kabel und Transceiver für optische Übertragungsstrecken eingesetzt oder miteinander kombiniert werden.

Verifizierbarkeit

Die Verifizierbarkeit von Entwürfen kann auf verschiedenen Ebenen erfolgen. Neben einer speziellen Testleiterplatte, die einen Großteil der Signale, die über die Konnektoren zum Trägersystem verlaufen, über Testpunkte bzw. Anschlussstecker für einen Logikanalysator zugänglich macht, bieten die FPGA-Hersteller Xilinx und Altera Hard-/Softwarelösungen an [254, 255], die ein FPAG-internes Hardware-Debugging auch ohne zusätzliche Messaufbauten ermöglichen. Dabei übermitteln in das zu überwachende Design eingebrachte Hardwarekomponenten über JTAG oder USB Statusinformationen bzw. Registerinhalte an die überwachende Software.

Skalierbarkeit

Die bisher gültige Obergrenze der Skalierbarkeit auf Hostsysteme wird mit dem RAPTOR-XPress aufgehoben. Die Kommunikationsarchitektur ist konsequent auf eine Cluster-Bildung aus mehreren RAPTOR-XPress-Systemen ausgelegt. Dabei kann eine bidirektionale Anbindung eines einzelnen DB an ein bis maximal sechzehn weitere auf unterschiedlichen Basisboards befindliche DBs erfolgen. Mögliche Architekturvarianten werden in Kapitel 5.9 diskutiert. Die in [232] angesprochene Skalierbarkeit auf Modul- und Systemebene bleibt dabei uneingeschränkt bestehen.

Prozessor- und Betriebssystemunabhängigkeit

Die Einbindung in Betriebssysteme geschieht über mehrere standardisierte Schnittstellen. Neben PCIe 2.0 zur direkten Anbindung in handelsüblichen PC- und Serversystemen besteht die Möglichkeit eines Aufbaus unabhängig vom Hostsystem. Hier erfolgt die Stromversorgung des RAPTOR-XPress nicht über den PCIe-Konnektor, sondern sie wird separat eingespeist. Zur Anbindung an stationäre oder mobile Geräte werden zwei USB-High-Speed-Schnittstellen implementiert, von denen eine im Host-Modus und die andere im Peripheral-Modus betrieben wird.

Darüber hinaus kann über die ebenfalls vorhandene Gigabit-Ethernet-Anbindung des Trägersystems und der Module konfiguriert werden. Für den Betrieb ohne jegliche Rechnerunterstützung können Konfigurations- und Betriebsdaten über eine SD-Karte in das Basisboard eingebracht und von dort an die entsprechenden Komponenten verteilt werden. Durch die Nutzung von Standardschnittstellen bleibt die Forderung nach Prozessor- und Betriebssystemunabhängigkeit gewahrt, da eine jeweilige Unterstützung mittels Entwicklung entsprechender Treiber erfolgen kann.

Lebenszyklus

Aufgrund des modularen Aufbaus kann auch der Lebenszyklus der Plattform als zukunftssicher erachtet werden. Administrative Aufgaben werden auf dem Basisboard abgehandelt. Die globale Spannungsversorgung wird ebenfalls auf einem separaten Modul implementiert und so ausgelegt, dass eine Änderung der eingesetzten Spannungsregler hin zu leistungsfähigeren oder energieeffizienteren Strukturen möglich ist. Die Kommunikationsstrukturen für den externen Hochgeschwindigkeitsdatenaustausch können ebenfalls unterschiedlich implementiert werden – zum Beispiel kupfergebunden oder über Lichtwellenleiter.

Ein weiterer Faktor, der den effektiven Lebenszyklus des Systems erhöht, ist die Umsetzung einer variabel konfigurierbaren IO-Spannung auf den Modulen. Die Aushandlung der erlaubten bzw. benötigten Spannung erfolgt durch Mikrocontroller, die sich auf dem Basisboard sowie auf den Modulen befinden und selbstständig erkennen, welche Rahmenbedingungen durch die eingesetzten Module zu beachten sind. Durch diese Umsetzung können auch neuere FPGA-Familien zusammen mit DBs aus Vorgängerentwicklungen des RAPTOR-XPress in das System integriert werden.

Dynamische Rekonfiguration

Dynamische Rekonfiguration wird weiterhin durch die Architektur unterstützt. Die Module können ihre Konfigurationsdaten über verschiedene Wege beziehen. Neben der direkten Konfiguration über die PCIe-Schnittstelle ist es möglich, vorbereitete Bitströme im Speicher des Moduls oder des Trägersystems abzulegen und entsprechend die DBs zu konfigurieren. Durch die unabhängige Ansteuerung der Konfigurationsleitungen der einzelnen Module kann parallel für jedes DB die Konfiguration mit gleichen oder unterschiedlichen Daten durchgeführt werden. Unabhängig davon ist die partielle Selbstrekonfiguration durch den FPGA des eingesetzten DB jederzeit möglich.

Universeller Hardwarebeschleuniger

Das Einbinden externer Komponenten ist durch die konsequente Weiterentwicklung der hierfür vorgesehenen DBs weiterhin möglich. Dabei werden sowohl Standardschnittstellen wie Gigabit-Ethernet als auch das Erfassen analoger Signale mit einer hohen Bandbreite und Abtastrate unterstützt. Schnittstellen zum Einbinden von Kameras und die anschließende Auswertung oder Vorverarbeitung der erfassten Bilddaten stellen eine weitergehende Umsetzung der Ansätze dar, die bereits für RAPTOR2000 und RAPTOR-X64 gültig waren.

5.8 Inter-FPGA-IP-Core

Während der Entwurfsphase von Applikationen für ein RAPTOR-XPress-Board oder den RAPTOR-XPress-Cluster erhält der Anwender Unterstützung durch die gebündelten Funktionen der Inter-FPGA-Schnittstelle, die in einer weiteren im Fachgebiet *Kognitronik und Sensorik* der Universität Bielefeld entstandenen Arbeit [288] entwickelt wurde. Der IP-Core ermöglicht dem Benutzer eine einfache Integration in den Implementierungsablauf und stellt den Austausch von Nachrichten über FPGA-Grenzen hinaus für die Anwendung transparent dar. Aufgrund der Xilinx-typischen Schnittstelle für Speicher, die nach dem Prinzip einer Warteschlange organisiert sind (engl.: First In First Out, FIFO), können die vier FPGAs eines RAPTOR-XPress auf Sende- und Empfangsseite für eine Applikation zu einer zusammenhängenden großen Logikfläche verschaltet werden. Im Folgenden wird die Funktionsweise der Inter-FPGA-Schnittstelle erläutert und mit ähnlichen Kommunikationslösungen verglichen.

5.8.1 Inter-FPGA im Vergleich

Um die verfügbare Fläche eines FPGAs für den Benutzer so transparent wie möglich zu erweitern, bietet der Inter-FPGA-IP-Core eine Schnittstelle, die in Bezug auf Aufbau und Verhalten der Portbelegung von Xilinx FIFOs entspricht. Hervorzuheben ist dabei, dass die FPGAs weder typgleich sein müssen, noch im Bereich der Anwenderlogik mit der gleichen Frequenz betrieben werden müssen. Für das Design stellt sich der Übergang zwischen zwei FPGAs wie eine asynchrone FIFO dar und somit genau wie bei einem typischerweise realisierten Taktdomänenwechsel von mehrere Bit breiten Signalen. Die Art der Verknüpfung mehrerer FPGAs bzw. der eingesetzte Übertragungsstandard ist für die Partitionierung eines Entwurfs demnach nicht von Belang. Beim Aufbau von GALS-Architekturen, wie sie beispielsweise in [256] beschrieben werden, können FIFOs genutzt werden, um Nachrichten zwischen den zueinander asynchron verlaufenden Teilen auszutauschen.

Wie in [257] aufgezeigt, stellt die Partitionierung von Aufgaben auf unterschiedliche FPGAs hohe Anforderungen an die eingesetzte Kommunikationsinfrastruktur. Die vorgestellte MPSoC-Plattform besteht aus Soft- und Hard-IP-Cores, die über Interrupts synchronisiert werden. Zur Abbildung von Pipeline-strukturierten Anwendungen auf dieser Plattform wird eine Abstraktionsebene in Software implementiert. Das Ausnutzen räumlich begrenzter Systeme, wie es im vorgestellten System in [207] mit 64 FPGAs der Fall ist, und somit eine direkte Abbildung eines Prozessors auf einem FPGA erfordert ebenfalls eine leistungsfähige Kommunikationsinfrastruktur für den Datenaustausch sowie die Synchronisation der einzelnen asynchron betriebenen Knoten. Die Synchronisation erfolgt über die örtliche Position im Grid des Systems. Die Knoten sind in einem Gitter miteinander verbunden, was eine einfache Erweiterbarkeit gewährleistet. Jeder Knoten propagiert sein Ergebnis an seine nachfolgenden Nachbarn (local barrier synchronization), und der letzte Knoten der Kette sammelt die Ergebnisse, um sie nach außen weiterzugeben. Durch diesen Mechanismus werden lokale und globale Zyklen für die Bearbeitung eines Auftrags gebildet. Die dargelegte Vorgehensweise wird im Nachfolgesystem [206] mit 100 FPGAs weiter evaluiert. Als zukünftige Entwicklung wird ein Zeitmultiplex-Verfahren vorgeschlagen, das virtuelle Knoten ermöglicht.

Durch die Einführung von Broadcast- und prioritätgesteuerten Nachrichtenaustausch auf Basis eines dedizierten sogenannten Interaccelerator-Netzwerks können in [258] die Eigenschaften der physikalischen Kommunikation für den Anwender abstrahiert werden, um eine verbesserte Modularität und Effizienz zu erreichen. Plattformabhängige Gegebenheiten werden modelliert. Die Ergebnisse für verschiedene Topologien ergeben eine Reduktion der Kommunikationszeit von bis zu 55 % bei Anwendung im CusComNet-Framework. Die Cluster-Größe und die Tiefe der eingesetzten Paketpuffer haben keinen Einfluss auf das Ergebnis. Durch den Einsatz einer kaskadierenden Broadcast-Ausbreitung lässt sich das Ergebnis auf alle Topologien übertragen. In [259] wird mit Hilfe des Zeitmultiplex-Verfahrens ein über geteilte Speicherbereiche kommunizierender Multicore auf einem einzelnen FPGA abgebildet.

Eine als Crossbar-Switch-Plattform ausgelegte Multi-FPGA-Lösung wird in [260] vorgestellt. Relevant für den Vergleich mit dem Inter-FPGA-IP-Core ist das Parallel Handshaking Interface (PHI), welches die FPGAs untereinander verbindet und in Abbildung 5.20 dargestellt wird.

Das PHI bildet wie die Inter-FPGA-Schnittstelle eine sich für die Anwendung als FIFO-Interface darstellende Kommunikationsstruktur, welche zwei asynchron zueinander arbeitende FPGAs miteinander verbindet. Es wird jedoch kein zusätzlicher source-synchroner Takt übertragen, sondern über zwei Signale die Gültigkeit (valid) in Empfängerrichtung und die Bestätigung für die Übernahme der Daten (ack) in Senderrichtung angezeigt. Das parallele Dateninterface kann an die Anzahl der zur Verfügung stehenden Signale angepasst werden, jedoch ohne den Abgleich der parallelen Abtastzeitpunkte zu unterstützen. Echte Burst-Übertragungen sind

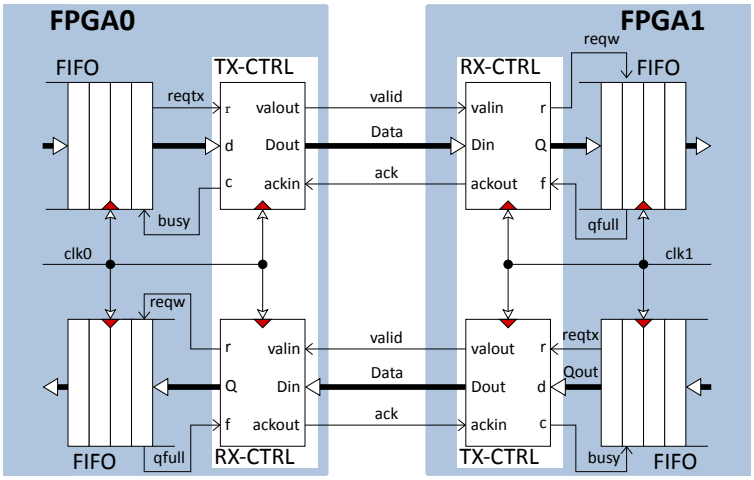


Abbildung 5.20: Übersicht der PHI-Architektur [260]

aufgrund des Handshaking-Verfahrens nicht möglich, da jedes zu übertragende Datenwort zunächst durch den Sender als gültig markiert und anschließend durch den Empfänger als empfangen bestätigt werden muss, woraufhin das Gültigkeitssignal und darauf folgend das Bestätigungssignal zurückgenommen werden müssen, bevor ein neues gültiges Datenwort angezeigt werden kann. Die Flusskontrolle ergibt sich über das ack-Signal, welches vom Empfänger nur gesetzt wird, sofern dieser ein weiteres Datenwort verarbeiten kann.

Die beiden Xilinx Application Notes [261] und [262] zeigen den Entwurf einer 16-Kanal-source-synchronen Verbindung zwischen zwei FPGAs auf. Die Art der Kanaladaptierung erfolgt im Fall von [261] statisch nach dem Einschalten zum Ausgleich herstellungsprozessabhängiger Einflüsse und im Fall von [262] dynamisch während des laufenden Betriebs, um Einflussfaktoren die Versorgungspannung und die Temperatur betreffend (engl.: Process Voltage Temperature, PVT) auszugleichen. In beiden Fällen steht dem Anwender eine 128 bit bzw. 96 bit breite Schnittstelle zur Verfügung, welche zusätzlich mit dem Übertragungstakt versorgt werden muss. Im Gegensatz zum Inter-FPGA-IP-Core ist die Kanalbreite auf 16 Kanäle festgelegt worden und das Ergebnis der Kanaladaptierung nicht auslesbar. Somit können die ermittelten Werte nicht für die Auswahl sich annähernd gleich verhaltender Kanäle und damit zur Maximierung der Datenrate herangezogen werden.

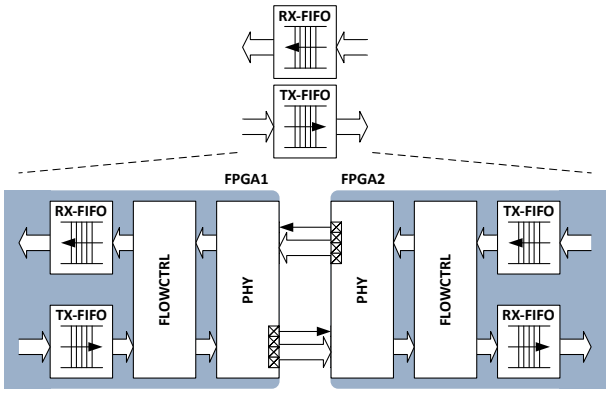


Abbildung 5.21: Architekturübersicht des Inter-FPGA-IP-Core

5.8.2 Inter-FPGA im Detail

Zur Darstellung eines transparenten Übergangs zwischen zwei FPGAs bieten sich Verfahren an, wie sie zum Beispiel in [259] beschrieben werden. Beim Überwinden der FPGA-Grenze kann der Nutzer jedoch nicht intuitiv vorgehen, da die Daten zunächst in einem von beiden Bausteinen aus zugänglichen Speicherbereich abgelegt werden und anschließend für den lesenden FPGA verfügbar sind. Die Synchronisation der Lese-Schreib-Zugriffe muss dabei besonders beachtet werden, zum einen um zu verhindern, dass unvollständige Daten gelesen und verarbeitet werden, zum anderen um im Fall beidseitiger Schreibzugriffe ein unbeabsichtigtes Überschreiben auszuschließen.

Aufgrund ihrer Datenstruktur gewährleisten FIFOs eine Synchronisation, die ein unbeabsichtigtes Überschreiben unterbindet und gleichzeitig die Reihenfolge der eingetragenen Daten sicherstellt. Darüber hinaus bieten sie Flusskontrollmechanismen, die den derzeitigen Füllstand des FIFO anzeigen und die lesende Seite über neue Daten in Kenntnis setzen.

Die in Abbildung 5.21 dargestellte Architektur bildet das oben beschriebene FIFO-Verhalten für die Kommunikation zwischen zwei FPGAs ab. Für die Anwenderlogik verhält sich die Schnittstelle wie eine der eingesetzten FPGA-Technologie entsprechende FIFO-Schnittstelle. Der in [281] vorgestellte Ansatz reduziert den Kommunikationsvorgang auf das Lesen und Beschreiben eines FIFO. Die benötigte Bandbreite ist richtungsunabhängig anpassbar, wodurch sich auch asymmetrische Verbindungen realisieren lassen. Da eine Kommunikation zwischen unterschiedlichen FPGA-Familien notwendig sein kann, um zum Beispiel günstige FPGAs zur Signalvorverarbeitung mit resourcenstarken FPGAs zur Weiterverarbeitung zu koppeln, werden alle aktuellen Xilinx FPGAs (Spartan-3/6, Virtex-2/4/5/6 und

7-Series) unterstützt. Eine Adaption auf andere FPGA-Hersteller wie Altera (inzwischen zugehörig zu Intel) ist ebenfalls möglich. Die Betrachtung der physikalischen Verbindung ist für den Benutzer auf die Auswahl der zur Verfügung stehenden Verbindungen und ihrer Pins an den FPGAs reduziert [288]. Der Ressourcenbedarf wird in Tabelle 5.6 angegeben, wobei M die Anzahl der IOs einer PHY-Schnittstelle und P die Anzahl der Pipeline-Register pro Kanal beschreibt.

Tabelle 5.6: Ressourcenbedarf der Inter-FPGA-Schnittstelle

LUT Architektur	#LUTs	#Registers	#BRAMs / #FIFOs
6-Eingänge	430	$700 + \mathrm{M} \cdot \mathrm{P}$	$\lceil \frac{\mathrm{M}}{32} \rceil$
4-Eingänge	650	$700 + \mathrm{M} \cdot \mathrm{P}$	$\lceil \frac{\mathrm{M}}{32} \rceil$

5.9 Verbindungsvarianten des RAPTOR-XPress

Im Folgenden werden die Ergebnisse aus Kapitel 3 auf die durch das RAPTOR-XPress zur Verfügung gestellten Kommunikationsstrukturen übertragen. Da der Fokus der Betrachtung auf der Bildung der Cluster-Struktur liegt, werden die reinen Intra-Board-Verbindungen wie Broadcast-Bus (Abschnitt 5.2.3) und die Direktverbinder (Abschnitt 5.2.2) nicht in die Untersuchung mit einbezogen, da sie keine globalen Verbindungen realisieren können.

Im Bereich des RAPTOR-XPress ist die wahlfreie Verbindung der Schnittstellen nicht gegeben. Um die Einschränkungen zu formulieren, muss zunächst eine zusätzliche Menge R eingeführt werden, die die einzelnen RAPTOR-XPress beinhaltet. Die Menge der Module beschränkt sich in ihrer Anzahl auf $|M| = 4$, und die Menge der Ports ist mit $|\Phi| = 16$ für das DB-V5 und $|\Phi| = 20$ für das DB-V7 nach oben beschränkt. Durch das Trägersystem erfolgt eine DB-unabhängige Beschränkung auf $|\Phi| = 16$ Ports. Die Ausgangsmenge V bestimmt sich durch $V = R \times M \times \Phi$. Für die Abbildung der Ports gilt dann $\exists r \in R, \exists m \in M, \forall \phi_i \in \Phi | \, (r, m, \phi_i) \rightarrow (r, m, \phi_i) \, ; i \in \{1, 2, 3, .., 16\}$.

Gegeben sei ein RAPTOR-XPress $R = \{r\}$, das mit FPGA-Modulen vollbesetzt ist; somit umfasst M die vier Elemente $\{m_1, m_2, m_3, m_4\}$, wobei jedes Modul $m_i \in M, \{i \in \mathbb{N} | 0 < x \leq 16\}$ alle 16 Ports realisiert ($\Phi = \{\phi_1, \phi_2, .., \phi_{16}\}$). Somit lässt sich die Mächtigkeit der Menge der ausgehenden Verbindungen eines Ports $\phi \in \Phi$ eines Moduls $m \in M$ aufgrund der Verschaltungsmöglichkeit der 8×8-Crosspoint-Switche durch $|E_O(m, \phi)| = \sum_{i=0}^{|8|} 8^{\underline{i}} = 109601$ bestimmen. Durch die eindeutige Zuweisung einer Sendeschnittstelle zu einer Empfangsschnittstelle (sie-

he Kapitel 3.2, Definition 4) kommen nur die Sendeschnittstellen eines lokalen Moduls oder die über einen der EXT-Verbinder angeschlossenen Schnittstellen eines entfernten Moduls in Betracht, wodurch sich insgesamt acht Möglichkeiten ergeben.

$$
\begin{aligned}
\Xi\left(RXP_1, DB_1\right) = \{ & \{\left(\left(RXP_1, DB_1\right), \phi_1\right) \xleftrightarrow{\text{EXT}_1 - \text{EXT}_1} \left(\left(RXP_2, DB_1\right)\right), \phi_1\}, \\
& \{\left(\left(RXP_1, DB_1\right), \phi_2\right) \xleftrightarrow{\text{EXT}_1 - \text{EXT}_1} \left(\left(RXP_2, DB_2\right)\right), \phi_2\}, \\
& \{\left(\left(RXP_1, DB_1\right), \phi_3\right) \xleftrightarrow{\text{EXT}_1 - \text{EXT}_1} \left(\left(RXP_2, DB_3\right)\right), \phi_3\}, \\
& \{\left(\left(RXP_1, DB_1\right), \phi_4\right) \xleftrightarrow{\text{EXT}_1 - \text{EXT}_1} \left(\left(RXP_2, DB_4\right)\right), \phi_4\}, \\
& \{\left(\left(RXP_1, DB_1\right), \phi_5\right) \xleftrightarrow{\text{EXT}_2 - \text{EXT}_1} \left(\left(RXP_3, DB_1\right)\right), \phi_5\}, \\
& \{\left(\left(RXP_1, DB_1\right), \phi_6\right) \xleftrightarrow{\text{EXT}_2 - \text{EXT}_1} \left(\left(RXP_3, DB_2\right)\right), \phi_6\}, \\
& \{\left(\left(RXP_1, DB_1\right), \phi_7\right) \xleftrightarrow{\text{EXT}_2 - \text{EXT}_1} \left(\left(RXP_3, DB_3\right)\right), \phi_7\}, \\
& \{\left(\left(RXP_1, DB_1\right), \phi_8\right) \xleftrightarrow{\text{EXT}_2 - \text{EXT}_1} \left(\left(RXP_3, DB_4\right)\right), \phi_8\}, \\
& \{\left(\left(RXP_1, DB_1\right), \phi_9\right) \xleftrightarrow{\text{EXT}_3 - \text{EXT}_1} \left(\left(RXP_4, DB_1\right)\right), \phi_9\}, \\
& \{\left(\left(RXP_1, DB_1\right), \phi_{10}\right) \xleftrightarrow{\text{EXT}_3 - \text{EXT}_1} \left(\left(RXP_4, DB_2\right)\right), \phi_{10}\}, \\
& \{\left(\left(RXP_1, DB_1\right), \phi_{11}\right) \xleftrightarrow{\text{EXT}_3 - \text{EXT}_1} \left(\left(RXP_4, DB_3\right)\right), \phi_{11}\}, \\
& \{\left(\left(RXP_1, DB_1\right), \phi_{12}\right) \xleftrightarrow{\text{EXT}_3 - \text{EXT}_1} \left(\left(RXP_4, DB_4\right)\right), \phi_{12}\}, \\
& \{\left(\left(RXP_1, DB_1\right), \phi_{13}\right) \xleftrightarrow{\text{EXT}_4 - \text{EXT}_1} \left(\left(RXP_5, DB_1\right)\right), \phi_{13}\}, \\
& \{\left(\left(RXP_1, DB_1\right), \phi_{14}\right) \xleftrightarrow{\text{EXT}_4 - \text{EXT}_1} \left(\left(RXP_5, DB_2\right)\right), \phi_{14}\}, \\
& \{\left(\left(RXP_1, DB_1\right), \phi_{15}\right) \xleftrightarrow{\text{EXT}_4 - \text{EXT}_1} \left(\left(RXP_5, DB_3\right)\right), \phi_{15}\}, \\
& \{\left(\left(RXP_1, DB_1\right), \phi_{16}\right) \xleftrightarrow{\text{EXT}_4 - \text{EXT}_1} \left(\left(RXP_5, DB_4\right)\right), \phi_{16}\} \}
\end{aligned}
\tag{5.19}
$$

Die Verbindung lässt sich als ungerichteter Graph $G = \{V, E\}$ mit einer Menge von Knoten V und einer Menge von ungerichteten Kanten $E \subseteq \{\{s,t\} \,|s,t \in V\}$ modellieren Dieser Ansatz ist gültig für Lane-orientierte Ansätze wie Xilinx Aurora [263, 264], bei denen jede Verbindung bidirektional aufgebaut ist. Trotz der Auslegung Auroras als Streaming-Interface ist eine Modellierung der MGT-Netze und typischer LVDS-Netze mit gerichteten Kanten $E \subseteq V \times V$ möglich. Dieser Ansatz wird bei verschiedenen Applikationen notwendig, so zum Beispiel wenn externe

Tabelle 5.7: TX-Verbindungsmatrix für 17 FPGAs

R	DB	S_o	X	S_i	S_o	R	X	S_i	S_o	DB	S_i
1	1	01	01	01M1	01E1	2	01	01E1	01M1	1	01
1	1	02	02	02M1	02E1	2	02	02E1	02M2	2	02
1	1	03	03	03M1	03E1	2	03	03E1	03M3	3	03
1	1	04	04	04M1	04E1	2	04	04E1	04M4	4	04
1	1	05	05	05M1	05E2	3	05	05E1	05M1	1	05
1	1	06	06	06M1	06E2	3	06	06E1	06M2	2	06
1	1	07	07	07M1	07E2	3	07	07E1	07M3	3	07
1	1	08	08	08M1	08E2	3	08	08E1	08M4	4	08
1	1	09	09	09M1	09E3	4	09	09E1	09M1	1	09
1	1	10	10	10M1	10E3	4	10	10E1	10M2	2	10
1	1	11	11	11M1	11E3	4	11	11E1	11M3	3	11
1	1	12	12	12M1	12E3	4	12	12E1	12M4	4	12
1	1	13	13	13M1	13E4	5	13	13E1	13M1	1	13
1	1	14	14	14M1	14E4	5	14	14E1	14M2	2	14
1	1	15	15	15M1	15E4	5	15	15E1	15M3	3	15
1	1	16	16	16M1	16E4	5	16	16E1	16M4	4	16

Datenquellen wie ADCs oder Kamerainterfaces eingebunden werden, bei denen der Rückkanal fehlt.

Für eine Crosspoint-Switch-Strecke gilt $|M| \geq 1$ und $|C| \geq 1$. $V = \{M, C\}$. Aufgrund der Anzahl möglicher MGT-Verbindungen kann die Valenz $d(V)$ dabei 16 nicht übersteigen. Wie den Tabellen 5.8 und 5.7 zu entnehmen, ist keine Verbindung zwischen unterschiedlichen Crosspoint-Switchen **X** eines Trägersystems **R** möglich.

Mit Hilfe des Crosspoint-Switch-Array lässt sich eine Vielzahl unterschiedlicher Topologien darstellen. In Kombination mit den Crosspoint-Switchen der FSB-MGT-Bridge können bis zu 17 FPGAs vollvermascht über je eine MGT-Lane gekoppelt werden, im Fall des DB-V5 somit mit bis zu 6,5 Gbit/s pro Verbindung. Die Tabellen 5.7 und 5.8 zeigen dabei den Hinweg von den 16 Ausgangsschnittstellen S_o des ersten FPGA zu den jeweiligen Eingangsschnittstellen S_i auf den Ziel-FPGAs bzw. den Rückverlauf sowie die Zuordnung der Schnittstellen zu den DBs und RAPTOR-XPress-Trägersystemen.

Unter Auslassung der beteiligten Crosspoint-Switche ergibt sich die kompakte Beschreibung 5.19 der Verbindung mit $P = \{RXP_1, RXP_2, ..., RXP5\}$, $M = \{DB_1, DB_2, DB_3, DB_4\}$ und $\Phi = \{\phi_1, \phi_2, ..., \phi_{15}\}$ für $m = (RXP_1, DB_1)$ auf die Zielmenge:

$$\Xi = \{\ (P \times M) \setminus \{\ (RXP_1, DB_1), (RXP_1, DB_2), \\ (RXP_1, DB_3), (RXP_1, DB_4)\ \}\ \} \times \Phi$$

Tabelle 5.8: RX-Verbindungsmatrix für 17 FPGAs

R	DB	S_o	X	S_i	S_o	R	X	S_i	S_o	DB	S_i
2	1	01	01	01M1	01E1	1	01	01E1	01M1	1	01
2	2	02	02	02M2	02E1	1	02	02E1	02M1	1	02
2	3	03	03	03M3	03E1	1	03	03E1	03M1	1	03
2	4	04	04	04M4	04E1	1	04	04E1	04M1	1	04
3	1	05	05	05M1	05E1	1	05	05E2	05M1	1	05
3	2	06	06	06M2	06E1	1	06	06E2	06M1	1	06
3	3	07	07	07M3	07E1	1	07	07E2	07M1	1	07
3	4	08	08	08M4	08E1	1	08	08E2	08M1	1	08
4	1	09	09	09M1	09E1	1	09	09E3	09M1	1	09
4	2	10	10	10M2	10E1	1	10	10E3	10M1	1	10
4	3	11	11	11M3	11E1	1	11	11E3	11M1	1	11
4	4	12	12	12M4	12E1	1	12	12E3	12M1	1	12
5	1	13	13	13M1	13E1	1	13	13E4	13M1	1	13
5	2	14	14	14M2	14E1	1	14	14E4	14M1	1	14
5	3	15	15	15M3	15E1	1	15	15E4	15M1	1	15
5	4	16	16	16M4	16E1	1	16	16E4	16M1	1	16

Zu beachten ist, dass die Verteilung der FPGAs einem Minimalbeispiel in Bezug auf die Latenz T der Übertragung entspricht, da jede MGT-Verbindung genau zwei Crosspoint-Switche passiert. Entsprechend gilt für Formel 5.20 $n = 2$. Eine Übertragungsstrecke zwischen zwei FPGAs auf DBs von unterschiedlichen direkt miteinander verbundenen RAPTOR-XPress-Trägersystemen besteht aus den Leiterbahnen auf den DBs und dem RAPTOR-XPress zwischen den FPGAs und den Crosspoint-Switchen, dem Crosspoint-Switch selbst, den Leiterbahnen auf dem RAPTOR-XPress und den High-Speed-Piggybacks sowie aus der Verbindung zwischen den zwei High-Speed-Piggybacks in Form von Samtec Kabeln. Entsprechend erhöhen sich die Signallaufzeiten um die spezifischen Zeiten der beteiligten Komponenten. Hinzu kommt die Verarbeitungszeit in den Sende- und Empfangs-Transceivern, wozu ebenfalls Schritte der Protokollverarbeitung gehören können. Erfolgt eine Weiterleitung des Signals zu einem dritten RAPTOR-XPress, so erhöht sich die Signallaufzeit um den Anteil der Verarbeitungszeit des zusätzlich beteiligten Crosspoint-Switch und der weiteren Verbindung T_{Verb}.

$$T_{\text{InterRXP}} = T_{\text{out}} + n \cdot T_{\text{XPS}} + (n-1) \cdot T_{\text{Verb}} + T_{\text{in}} \text{ , mit } n \geq 2 \tag{5.20}$$

$$T_{\text{out}} = T_{\text{TX}} + T_{\text{DB-XPS}} \tag{5.21}$$

$$T_{\text{in}} = T_{\text{RX}} + T_{\text{XPS-DB}} \tag{5.22}$$

$$T_{\text{Verb}} = 2 \cdot T_{\text{XPS-HSPB}} + T_{\text{Kabel}} \tag{5.23}$$

Wird die MGT-Verbindung für die Intra-RAPTOR-XPress-Kommunikation verwendet, entfällt der Zeitanteil der Kabelverbindung aus Gleichung 5.23. Darüber hinaus befindet sich lediglich ein Crosspoint-Switch im Signalpfad zwischen zwei DBs eines RAPTOR-XPress, wodurch sich Formel 5.20 zur nachfolgenden Gleichung 5.24 vereinfacht.

$$T_{\text{IntraRXP}} = T_{\text{out}} + T_{\text{XPS}} + T_{\text{in}} \tag{5.24}$$

Durch Gleichung 5.24 können die Latenzen für Übertragungen innerhalb des RAPTOR-XPress-Clusters quantitativ bestimmt werden. Für die Ermittlung von T_{TX} und T_{RX} sind sowohl die eingesetzten Protokolle und somit der zeitliche Mehraufwand bei der Protokollverarbeitung als auch die Konfiguration der MGTs maßgeblich. Zusammen mit der Übertragungszeit zu bzw. von den Crosspoint-Switchen auf den DBs und den RAPTOR-XPress-Trägersystemen ergeben sich die Werte für T_{out} und T_{in}.

5.10 RAPTOR-XPress-Cluster-Topologie

Mit Hilfe des High-Speed-Piggyback können vier RAPTOR-XPress-Trägersysteme miteinander verbunden werden. Überträgt man die Ergebnisse der Topologieoptimierung aus Kapitel 3.4 auf dieses Anwendungsszenario, so können unter Ausnutzung aller vier Verbindungen die Topologievarianten aus der Menge Θ_{4_n} 4-regulärer Graphen mit der Anzahl $n \forall \{n \in \mathbb{N} | n \geq 5\}$ der eingesetzten RAPTOR-XPress als Knoten bestimmt werden. In einem ersten Aufbau werden $n = 16$ RAPTOR-XPress zu einem Cluster verbunden.

Als regelmäßige Struktur bietet sich neben dem Gitter der 2D-Torus an. Entsprechend ist der Durchmesser $D\left(G_{\text{Torus 4\_16}}\right) = 4$, die Exzentrizitätssumme beträgt $\psi\left(G_{\text{Torus 4\_16}}\right) = 64$. Gleiches gilt für eine Realisierung als Tesserakt (4D-Hyperwürfel). Dies bedeutet, dass für jeden Knoten $v \in V$ die Exzentrizität $\epsilon\left(v\right) = 4$ ist. Der Exzentrizitätsvektor der Topologie lautet entsprechend:

$$\vec{\epsilon}(V) = (4\ 4\ 4\ 4\ 4\ 4\ 4\ 4\ 4\ 4\ 4\ 4\ 4\ 4\ 4\ 4)^{\mathsf{T}} \tag{5.25}$$

Die maximale Latenz bei der Übertragung von Daten ergibt sich somit durch:

$$T_{\text{InterRXP}} = T_{\text{out}} + 5 \cdot T_{\text{XPS}} + 4 \cdot T_{\text{Verb}} + T_{\text{in}}$$

Eine gemäß dem in Kapitel 3.4 vorgestellten Ansatz ψ-optimierte Topologie hat einen Durchmesser von

$$D\left(\omega_{4\_16}\right) = 3$$

und eine Exzentrizitätssumme von

$$\psi\left(\omega_{4\_16}\right) = 36$$

Entsprechend ist die Exzentrizität der Knoten unterschiedlich und nach oben mit 3 beschränkt. Die Distanzmatrix aus 5.26 beschreibt die Struktur des optimierten Graphen und zeigt gleichzeitig die einzelnen minimalen Pfadlängen zwischen zwei Knoten auf. Der beschreibende Graph wird in Abbildung 5.22 aufgeführt. Aufgrund der Struktur ist eine kreuzungsfreie zweidimensionale Darstellung nicht möglich. Zwar lässt sich die Anzahl der Kreuzungen durch Verschieben der Knoten reduzieren, da sich die Übersichtlichkeit nur unwesentlich erhöht, wird allerdings zu Gunsten fester Positionen der Knoten auf diesen Schritt verzichtet. Für die praktische Realisierung mit Hilfe von Kabeln ist das Merkmal der Kreuzungsfreiheit in der Ebene nicht relevant.

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \\ v_9 \\ v_{10} \\ v_{11} \\ v_{12} \\ v_{16} \\ v_{14} \\ v_{15} \\ v_{16} \end{array} \begin{array}{c} \begin{array}{cccccccccccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 & v_9 & v_{10} & v_{11} & v_{12} & v_{13} & v_{14} & v_{15} & v_{16} \end{array} \\ \left(\begin{array}{cccccccccccccccc} 0 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 3 \\ 1 & 0 & 2 & 2 & 2 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 1 & 2 & 0 & 2 & 2 & 1 & 2 & 2 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 2 \\ 1 & 2 & 2 & 0 & 2 & 2 & 2 & 2 & 2 & 2 & 1 & 1 & 1 & 2 & 2 & 2 \\ 1 & 2 & 2 & 2 & 0 & 2 & 2 & 2 & 2 & 2 & 1 & 2 & 2 & 1 & 1 & 2 \\ 2 & 1 & 1 & 2 & 2 & 0 & 2 & 2 & 2 & 2 & 3 & 1 & 2 & 1 & 2 & 2 \\ 2 & 1 & 2 & 2 & 2 & 2 & 0 & 2 & 1 & 2 & 1 & 2 & 2 & 2 & 2 & 1 \\ 2 & 1 & 2 & 2 & 2 & 2 & 2 & 0 & 2 & 1 & 2 & 2 & 1 & 2 & 1 & 2 \\ 2 & 2 & 1 & 2 & 2 & 2 & 1 & 2 & 0 & 2 & 2 & 2 & 1 & 2 & 1 & 2 \\ 2 & 2 & 1 & 2 & 2 & 2 & 2 & 1 & 2 & 0 & 1 & 2 & 2 & 2 & 2 & 1 \\ 2 & 2 & 2 & 1 & 1 & 3 & 1 & 2 & 2 & 1 & 0 & 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 1 & 2 & 1 & 2 & 2 & 2 & 2 & 2 & 0 & 2 & 2 & 1 & 1 \\ 2 & 2 & 2 & 1 & 2 & 2 & 2 & 1 & 1 & 2 & 2 & 2 & 0 & 1 & 2 & 2 \\ 2 & 2 & 2 & 2 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 1 & 0 & 2 & 1 \\ 2 & 2 & 2 & 2 & 1 & 2 & 2 & 1 & 1 & 2 & 2 & 1 & 2 & 2 & 0 & 2 \\ 3 & 2 & 2 & 2 & 2 & 2 & 1 & 2 & 2 & 1 & 2 & 1 & 2 & 1 & 2 & 0 \end{array} \right) \end{array} \tag{5.26}$$

Zur vereinfachten Darstellung der maximalen Pfadlänge zwischen zwei Knoten wird in Abbildung 5.27 der Exzentrizitätsvektor aufgeführt.

$$\vec{\epsilon}(V) = (3\ 2\ 2\ 2\ 2\ 3\ 2\ 2\ 2\ 2\ 3\ 2\ 2\ 2\ 2\ 3)^{\mathsf{T}} \tag{5.27}$$

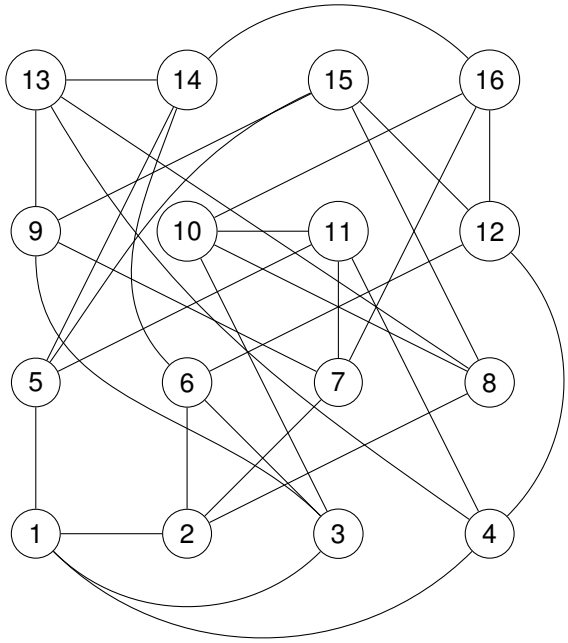


Abbildung 5.22: Graph mit minimalem $\zeta\left(G_{\omega_{4\_16}}\right)$

Betrachtet man den Exzentrizitätsindex beider Strukturen, so zeigt sich, dass die optimierte Topologie mit $\zeta\left(\omega_{4\_16}\right) = 2,25$ deutlich kleiner ist als $\zeta\left(G_{\text{Torus}\,4\_16}\right) = 4$. Entsprechend fällt auch die gemittelte Latenz zwischen den Knoten geringer aus:

$$T_{\text{ø}} = T_{\text{out}} + 3,25 \cdot T_{\text{XPS}} + 2,25 \cdot T_{\text{Verb}} + T_{\text{in}}$$

Sogar die maximale Latenz ist in diesem Fall geringer als die einer Gitter- oder Torustopologie:

$$T_{\max} = T_{\text{out}} + 4 \cdot T_{\text{XPS}} + 3 \cdot T_{\text{Verb}} + T_{\text{in}}$$

Wird die Anzahl an Trägersystemen weiter gesteigert, ergeben sich die in Tabelle 5.9 aufgeführten Werte. Aufgrund des gestiegenen Berechnungsaufwandes gegenüber der Bestimmung der reinen Anzahl an Topologievarianten aus Kapitel 3.3 reduziert sich die Berechnungsgeschwindigkeit von ca. $50\,000$ Graphen pro Sekunde auf etwa 600 Graphen pro Sekunde auf einer Intel Core i7-3770 CPU mit 16 GB Arbeitsspeicher. Aus diesem Grund konnten im Rahmen dieser Arbeit nur praxisrelevante Werte bis $|V| \leq 19$ untersucht werden.

Tabelle 5.9: Betrachtung ψ-optimaler 4-regulärer Graphen mit $|V| > 16$

$\lvert V\rvert$	$D\left(\omega_{4\_\lvert V\rvert}\right)$	$\psi\left(\omega_{4\_\lvert V\rvert}\right)$	$\zeta\left(\omega_{4\_\lvert V\rvert}\right)$
17	3	43	2,5294
18	3	54	3
19	3	57	3,3529

Deutlich wird, dass auch mit steigender Anzahl an Knoten der Durchmesser und der Exzentrizitätsindex unter den entsprechenden Werten regelmäßiger Strukturen liegen.

5.11 Topologieverhalten im Betrieb mit mehreren Anwendern

Zur Betrachtung des Topologieverhaltens während des Betriebs des Clusters mit mehreren Anwendern werden die Ressourcen einzelner Knoten belegt und somit einschließlich ihrer vier Kanten aus dem Graphen entfernt, da sie anderen Anwendern nicht zur Verfügung stehen. Bei jedem Schritt werden der Durchmesser und die Exzentrizitätssumme betrachtet. Wie zu erwarten kann durch Entfernen von vier Knoten der Zusammenhang des Graphen zerstört werden. Der angegebene Maximalwert für den Durchmesser $D\left(\omega_{4\_16}\right)$ gilt für die Menge der zusammenhängenden Graphen. Der Ansatz, vollständige Knoten und die anhänglichen Kanten zu entfernen, geht von einer vollständigen Ausnutzung des Crosspoint-Switch-Array des jeweiligen RAPTOR-XPress aus. Der Vergleich zwischen den beiden Topologien 2D-Torus und ψ-optimaler Graph bei steigender Zahl an nicht mehr für die Verbindung zur Verfügung stehenden Knoten ergibt das in Abbildung 5.23 aufgezeigte Verhalten. Zu erkennen ist, dass sowohl die minimal erzielbare Exzentrizitätssumme als auch die entgegengesetzte Worst-Case-Betrachtung der maximalen Exzentrizitätssumme immer ein mindestens gleichwertiges oder in weiten Bereichen sogar besseres Ergebnis für den ψ-optimalen Ausgangsgraphen liefern. Für die maximalen ψ-Werte ergibt sich aufgrund der Optimierung eine gemittelte Verbesserung um 16,8 %, für die minimalen Werte sogar um 31,0 %.

Der Durchmesser der Topologie ändert sich ebenfalls mit dem Belegen von Knoten, wie im unteren Bereich von Abbildung 5.23 zu sehen ist. Auch hier lässt sich erkennen, dass ausgehend von einem ψ-optimalen Graphen kleinere Durchmesser der sich ergebenden Graphen erzielbar sind.

Betrachtet man darüber hinaus die Möglichkeit, eine beliebige Anzahl von Knoten zu belegen, ohne dass ein nicht zusammenhängender Graph entsteht, so können mehr Möglichkeiten bei einem ψ-optimalen Graphen als Ausgangsmenge gefunden

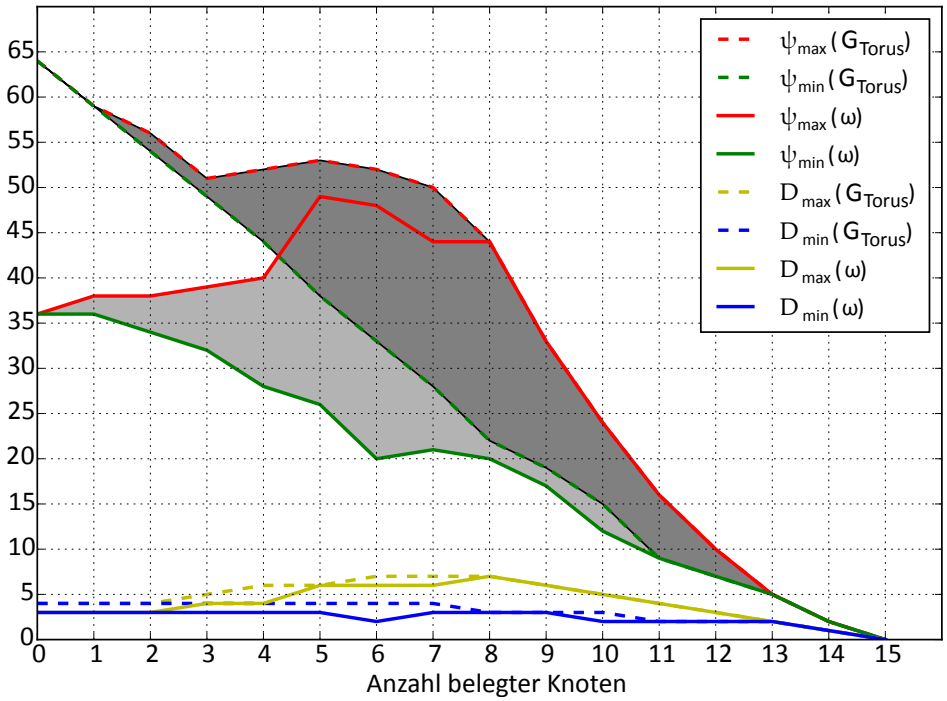


Abbildung 5.23: Verhalten von ψ und des Durchmessers bei steigender Knotenbelegung

werden als bei einem 2D-Torus. Abbildung 5.24 zeigt diesen Zusammenhang auf. Die gepunktete Linie gibt darüber hinaus die maximale Anzahl an Kombinationsmöglichkeiten der zu belegenden Knoten an.

Die erzielte Steigerung in der Anzahl der Platzierungsmöglichkeiten beim Vergleich der beiden Topologien entspricht dem hervorgehobenen Bereich und beträgt ca. 8,5 %. Durch die Anforderung, dass der verbleibende Graph zusammenhängend sein muss, ergibt sich der Unterschied gegenüber der zu Vergleichszwecken aufgeführten Binomialverteilung.

Hier wird deutlich, dass der Vorteil für die Benutzer eines ψ-optimiertenFPGA-Clusters nicht nur in der verringerten Latenz beim Nachrichtenaustausch zwischen verteilten FPGAs innerhalb des Systems liegt, sondern dass auch die Platzierungsmöglichkeiten von weiteren parallelen Anwendungen im Mehrbenutzerbetrieb gesteigert werden.

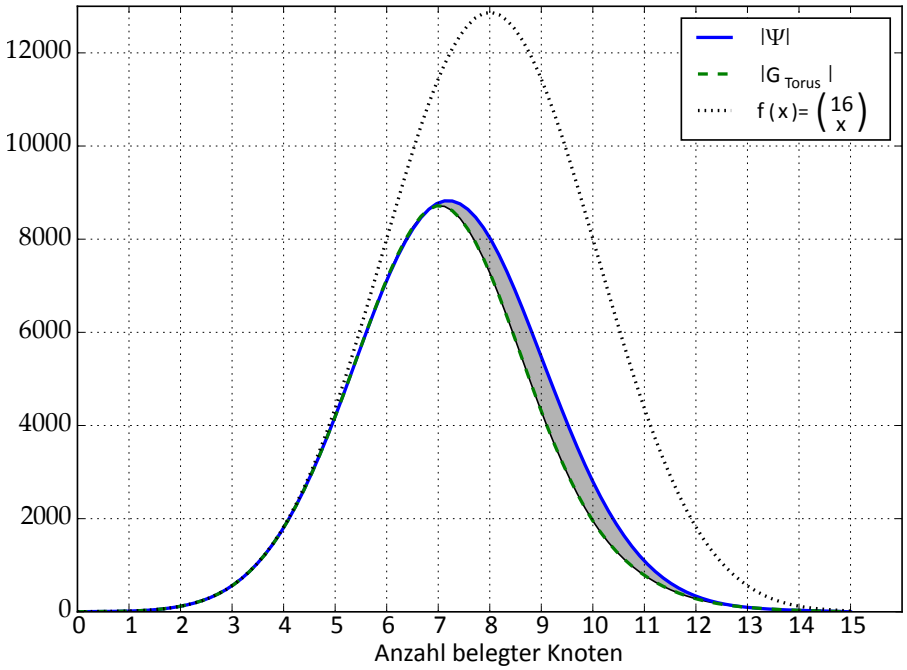


Abbildung 5.24: Platzierungsmöglichkeiten bei steigender Belegung

5.12 Zusammenfassung

Im Verlauf dieses Kapitels wurden der im Rahmen dieser Arbeit entstandene Entwurf sowie die Umsetzung des RAPTOR-XPress-Systems und die Eigenschaften des daraus gebildeten RAPTOR-XPress-Clusters vorgestellt. Die als Kooperation verschiedener Arbeiten im Fachgebiet *Kognitronik und Sensorik* der Universität Bielefeld entstandenen Komponenten DB-V5, DB-V7 und FSB-MGT-Bridge sowie der IP-Core Inter-FPGA wurden dabei detailliert betrachtet, und das Zusammenwirken mit dem RAPTOR-XPress-Trägersystem wurde erläutert. Ein besonderer Fokus wurde dabei auf die Kommunikationsinfrastruktur gelegt, welche den vorgestellten DBs zur Verfügung steht. Die verschiedenen Möglichkeiten zur Steuerung und Überwachung der Systemkomponenten durch den Anwender wurden aufgezeigt. Für das am Fachgebiet *Kognitronik und Sensorik* der Universität Bielefeld entwickelte modulare FPGA-Trägersystem wurde eine Anforderungsverifikation erstellt. Die dem Anwender zur Abstraktion der Kommunikation zwischen zwei FPGAs des Clusters zur Verfügung gestellte Inter-FPGA-Schnittstelle wurde zusammen mit den

darstellbaren Topologievarianten der Kommunikationsinfrastruktur in weiteren Abschnitten erläutert. Unter Anwendung der in dieser Arbeit entwickelten Methoden zur Auffindung optimaler Verbindungsstrukturen wurde abschließend ein Architekturvorschlag für einen RAPTOR-XPress-Cluster mit insgesamt 16 RAPTOR-XPress und 64 FPGAs vorgestellt. Diese Umsetzung erreicht bei der tatsächlichen Nutzung von Kommunikationswegen eine um bis zu 31 % geringere Exzentrizitätssumme gegenüber regelmäßigen Strukturen wie dem 2D-Torus bzw. dem Tesserakt. Die Auswirkungen der Optimierung werden ebenfalls anhand der Auswahl verwendbarer Knoten deutlich. So vergrößern sich der Platzierungsraum der ψ-optimalen Struktur als Ausgangsmenge um 8,5 % und mit ihm die Flexibilität des Systems bei zeitlich unabhängiger paralleler Nutzung. Da sich die Pfadlängen proportional auf die Latenz der Kommunikation auswirken, bedeutet eine Verringerung der Anzahl notwendiger Schritte immer auch eine geringere Latenz. Dies ist nicht nur in Bezug auf die Echtzeitfähigkeit des Systems relevant, sondern hat ebenfalls positiven Einfluss auf Anwendungen mit einem hohen Kommunikationsaufkommen.

Zur Einordnung der Leistungsfähigkeit des entstandenen FPGA-Clusters wird im folgenden Kapitel eine Gegenüberstellung von diesem System sowie der in Kapitel 4 vorgestellten Cluster gegeben. Herausgestellt werden die Möglichkeiten der jeweils vorhandenen Kommunikationsinfrastruktur. Darüber hinaus werden systemspezifische Parameter wie die Anzahl verfügbarer FPGAs, lokaler Speicher etc. herausgearbeitet.

6 Vergleich aktueller FPGA-Cluster

Um einen Vergleich der verschiedenen FPGA-Cluster erstellen zu können, muss eine Aufstellung über die für die Leistungsfähigkeit eines Systems relevanten Komponenten erfolgen. Die ressourcenoptimale Nutzung der in der Architektur vorhandenen Komponenten muss bereits beim Entwurf berücksichtigt werden. Innerhalb des Entwurfraums für eine FPGA-Cluster-Architektur müssen Schnittpunkte zwischen den unterschiedlichen Optimierungszielen ausfindig gemacht werden, die die Abhängigkeiten zwischen den unterschiedlichen Vorgaben berücksichtigen. Anwendungsspezifische FPGA-Cluster-Systeme können dabei Vereinfachungen zulassen, die beim Entwurf von allgemeinen Clustern und auch von solchen, die von mehreren Anwendern parallel nutzbar sind, Nachteile bringen würden. Deutlich zu sehen ist dies bei der Auslegung der Kommunikationsinfrastruktur. Da es bei für eine dedizierte Anwendung ausgelegten Systemen nicht relevant ist, ob sämtliche FPGAs innerhalb der Architektur einzeln von einem Hostsystem angesprochen werden können, kann die Verbindungsstruktur unregelmäßig und asymmetrisch ausgelegt werden. Diese Vereinfachung kann sich in Form von verringerten Latenzen beim Nachrichtenaustausch, einem wesentlich geringeren Aufwand im PCB-Layout oder einer auf die spezifische benötigte Bandbreite der einzelnen Komponenten der partitionierten Anwendung hin optimierten Struktur niederschlagen.

Zunächst erfolgt eine Definition der für den Vergleich notwendigen Ressourcenbegriffe. Anschließend wird in Abschnitt 6.1 eine differenzierte Übersicht über die Kommunikationsinfrastruktur eines FPGA-Clusters erstellt. Auf dieser Basis lassen sich unterschiedliche Modelle identifizieren und auf die verschiedenen Cluster anwenden. Die Gegenüberstellung der Kommunikationsinfrastrukturen der in Kapitel 4.2 vorgestellten Systeme erfolgt in Abschnitt 6.2. Den Abschluss des Kapitels bilden die Auswertung der in den vorherigen Teilen aufgezeigten Ergebnisse sowie eine Zusammenfassung.

6.1 Kommunikationsinfrastruktur

Neben den bereits aufgeführten Bewertungskriterien spielt die Leistungsfähigkeit der Kommunikationsinfrastruktur eine entscheidende Rolle hinsichtlich der Frage, ob ein FPGA-Cluster für bestimmte Aufgaben geeignet ist. In [23] wird herausgestellt, dass klassische Computer-Topologien wie Vektoren, Würfel oder Hypercubes nicht flexibel genug sind, um die Datenflussgraphen typischer Algorithmen sinnvoll

abzubilden. Zwar lassen sich Datenflussgraphen im Hinblick auf die Zielarchitektur partitionieren, jedoch unterliegen beispielsweise die in [265] aufgeführten Methoden starken Einschränkungen, was den Kommunikationsbedarf der Aufgaben untereinander betrifft. Dies zeigt, dass ein nichtspezialisiertes FPGA-Cluster-System eine flexible Kommunikationsinfrastruktur benötigt, die verschiedene Topologien abbilden kann. Die gleichzeitige Nutzung des Systems mit verschiedenen unabhängigen Anwendungen bedarf ebenfalls der Möglichkeit, von verschiedenen Stellen aus auf noch freie Ressourcen zugreifen zu können und diese untereinander zu verbinden.

Die folgenden drei Kriterien beschreiben die Anforderungen:

1. **Inter-FPGA-Kommunikation**: Die Verbindungsstruktur zwischen den einzelnen Anwendungs-FPGAs lässt sich durch die Anzahl an Kanälen und ihre jeweilige Bandbreite erfassen.

2. **Hostkommunikation**: Der Datenaustausch mit dem Anwender erfolgt, abgesehen von direkten Zugriffen über Taster und LEDs, über ein Hostsystem. Auch hierbei ergibt sich die Bewertung über die Bandbreite der Kommunikationskanäle zum Host, die den einzelnen Anwendungs-FPGAs zur Verfügung steht.

3. Die **Topologie** oder hierarchische Struktur spiegelt sich in den beiden oberen Punkten wider. Zu erfassen ist die Anzahl an Anwendungs-FPGAs je Subsystem und die Verteilung der Subsysteme auf einen oder mehrere Hosts.

Betrachtet werden nur leitungsgebundene Kommunikationsstrukturen. Zwar gibt es Ansätze drahtlose Übertragungen ebenfalls in FPGA-Cluster-Systemen einzusetzen, wie in [266] beschrieben, jedoch lassen sich auch diese Verfahren mit den im Folgenden vorgestellten Klassifizierungen erfassen. Eine Integration drahtloser Kommunikation in bestehende Infrastrukturlösungen von FPGA-Clustern ist häufig nur eingeschränkt möglich und kann allenfalls als Ergänzung zur leitungsgebundenen Datenübermittlung gesehen werden. Unterteilen lassen sich die Möglichkeiten für den leitungsgebundenen Datenaustausch in die drei folgenden Kategorien:

- **Punkt-zu-Punkt-Verbindungen** verbinden genau zwei Knoten miteinander.

- **Busverdrahtungen** verbinden mehrere Knoten gleichzeitig und benötigen aus diesem Grund eine Zugriffskontrolle auf das Kommunikationsmedium. Eine Übertragung zu mehreren Knoten ist strukturbedingt einfach möglich.

- **Verschaltbare Punkt-zu-Punkt-Verdrahtungen** kombinieren die Vorteile einer Busverdrahtung in Bezug auf die einfache Umsetzbarkeit von Multi- oder Broadcast-Übertragungen mit den Vorteilen einfacher Punkt-zu-Punkt-Verbindungen wie der parallelen Kommunikation mehrerer unterschiedlicher Teilnehmer untereinander.

6.1.1 Voraussetzung für den Vergleich von FPGA-Cluster-Kommunikationsstrukturen

Die Bedeutung der Latenz innerhalb einer Kommunikationsstruktur wird häufig im Hinblick auf die Bandbreite vernachlässigt. Jedoch spielt die Latenz der Übertragung gerade bei Anwendungen mit einer hohen Parallelität von nicht autonom berechenbaren Teilaufgaben eine entscheidende Rolle. In [267] wird gezeigt, dass die Fortschritte in der Elektronikentwicklung dazu geführt haben, dass sich der Faktor, mit der sich die Bandbreite vergrößert, quadratisch bis kubisch zum Faktor der Verringerung der Latenz entwickelt hat. Zu erklären ist der Unterschied mit der die durch Moore vorausgesagten Skalierung der Transistorstrukturen, die hauptsächlich Vorteile für die Bandbreite ergibt. So profitiert die Bandbreite von der Entwicklung schnellerer Transistoren, von einer höheren Anzahl an Transistoren auf der gleichen Die-Fläche und schließlich von der steigenden Anzahl der für die Kommunikation verfügbaren IO-Pins in modernen Chip-Gehäusen. Zwar reduziert die gesteigerte Verarbeitungsgeschwindigkeit durch den Einsatz von schnelleren Transistoren auch die Latenz, da aber zeitgleich die Chip-Fläche ebenfalls gewachsen ist und somit auch die zu überbrückende Strecke, lassen sich nur geringe Steigerungsraten in Bezug auf die Übertragungszeit von Daten verzeichnen [267].

In den Vergleich der Leistungsfähigkeit der Kommunikationsstruktur von FPGA-Clustern werden nur solche Kommunikationspfade miteinbezogen, die eine konstante, nicht von Faktoren außerhalb des Clusters abhängige Latenz aufweisen. Dies bedeutet, dass Schnittstellen wie PCIe oder Ethernet, die in einem Großteil der vorgestellten Systeme vorhanden sind, dennoch nicht in der Klassifikation betrachtet werden. Äußere, nicht von der tatsächlich auf den FPGAs implementierten Anwendung abhängige Umstände wie Netzwerklast, Anzahl an Busteilnehmern, Prozessorlast etc. haben einen direkten, für den Ablauf der Anwendung unbekannten Einfluss auf die Verzögerung im Datenaustausch. Aus diesem Grund werden in der graphentheoretischen Betrachtung der Kommunikationsinfrastruktur nur solche Kanten zugelassen, die zur Laufzeit ein festes Gewicht in Form der bestimmbaren Latenz besitzen. Dadurch kann es passieren, dass Knoten außerhalb der transitiven Hülle des beschreibenden Graphen liegen und somit das System nicht als eng gekoppelter Cluster verstanden werden kann, sondern lediglich eine lose Kopplung von verschiedenen, möglicherweise lokal eng gekoppelten FPGAs darstellt.

6.1.2 Flexibilität

Um beurteilen zu können, ob eine vorhandene Kommunikationsinfrastruktur innerhalb eines Cluster-Systems die Darstellung von Implementierungen aus verschiedenen Anwendungsbereichen erlaubt, wird eine Metrik für die Flexibilität, die Anpassbarkeit an unterschiedliche Topologien, Latenzen und Bandbreiten aufgestellt. Zur Darstellung der Flexibilität muss das Verhältnis von Logikressourcen zu

Kommunikationsressourcen formuliert werden. Da die Gruppe der Kommunikationsressourcen nicht homogen ist, sondern je nach Anforderung unterschiedliche Qualitäten bietet, speziell in Bezug auf die Latenz der Übertragung und die Bandbreite, erfolgt eine Unterteilung in die folgenden zwei Bereiche:

1. Inter-Board-Kommunikation
2. Intra-Board-Kommunikation

Sind die Logikressourcen eines einzelnen FPGAs nicht ausreichend, empfiehlt sich zunächst eine Erweiterung auf Logikressourcen, die sich auf dem gleichen Trägersystem befinden, unter Ausnutzung der typischerweise vorhandenen parallelen Kommunikationsstrukturen. Neben der in [162] nachgewiesenen Energieeffizienz paralleler Schnittstellen gegenüber seriellen Hochgeschwindigkeitsprotokollen ist die für direkte Verbindungen garantierbare feste Latenz ein Vorteil. Sollten diese Ressourcen ausgeschöpft sein, muss über die Intra-Board-Kommunikation auf FPGAs anderer Trägersysteme zugegriffen werden.

Darstellen lässt sich die Flexibilität als Funktion der Datenrate über der Anzahl der bidirektional erreichbaren FPGAs. Dabei wird die Anzahl der erreichbaren FPGAs ausschließlich im Single-Hop ohne mechanische Änderungen der Übertragungsstrecke betrachtet. Als Single-Hop wird eine Verbindung bezeichnet, wenn zwischen der Ausgangsschnittstelle des Quell-FPGA und der Eingangsschnittstelle des Ziel-FPGA keine Logikressourcen anderer FPGAs passiert werden müssen, wie dies beispielsweise bei Topologien mit dedizierten Kommunikationsknoten in Form von weiteren FPGAs geschieht. Relevant ist diese Unterscheidung bei der Betrachtung der Latenz der Übertragung. Die Übertragungszeit setzt sich mindestens aus der Summe der Komponenten T_{out}, T_{Verb} und T_{out} (vergleiche Kapitel 5.10) zusammen. Werden weitere Elemente in der Übertragungsstrecke hinzugefügt, wie Crosspoint-Switche oder FPGAs, so nimmt die Laufzeitverzögerung der Daten entsprechend zu. Dabei beträgt der Verzögerungsanteil durch dedizierte Crosspoint-Switche typischerweise 500 ps bis 520 ps [239, 240], während eine Datenweiterleitung durch die Logikressourcen von FPGAs schnell einen Bereich größer 100 ns erreicht.

Valenz-Bandbreiten-Verhältnis Eine Besonderheit des RAPTOR-XPress-Cluster ist es, dass die Datenrate aufgeteilt werden kann. Auf diese Weise ergibt sich das Valenz-Bandbreiten-Verhältnis. In Abbildung 6.1 wird das Verhältnis der Datenrate zur Anzahl der verbunden FPGAs dargestellt. Die Latenz der Verbindung unterscheidet sich bei der Aufteilung der Bandbreite zur Anbindung von bis zu 16 bzw. mit einem erweiterten Trägersystem theoretisch 20 anderen FPGAs nicht von der Latenz bei der Kommunikation mit der maximalen Datenrate von 230 Gbit/s mit einem einzelnen benachbarten FPGA. Für diesen Verbindungsaufbau ist es irrelevant, wie in Kapitel 5.9 gezeigt, ob die Kommunikation Inter-Board, d.h. zwischen

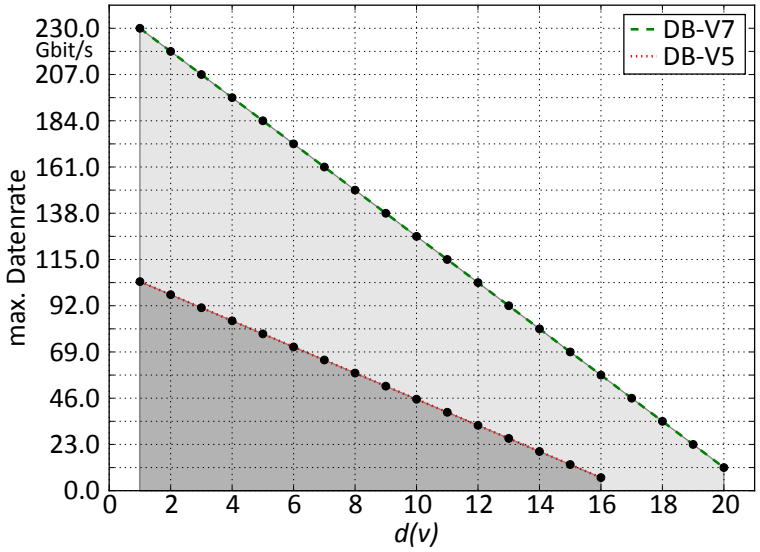


Abbildung 6.1: Valenz-Bandbreiten-Verhältnis im RAPTOR-XPress-Cluster

zwei Modulen eines RAPTOR-XPress-Trägersystems, oder Intra-Board, zwischen unterschiedlichen Modulen zweier RAPTOR-XPress, erfolgt. Davon unberührt bleibt die nicht-MGT-getriebene Intra-Board-Kommunikation eines Trägersystems, welche eine breitbandige und niederlatente Ergänzung zu den Inter-Board-Verbindungen darstellt.

Möglich wird eine solche Abhängigkeit der Bandbreite von der Anzahl der direkt zu erreichenden FPGAs innerhalb des FPGA-Clusters durch den Einsatz des in Kapitel 5.2.1 vorgestellten Crosspoint-Switch-Array. Die somit erzielte hohe Flexibilität gegenüber starren Strukturen führt zu einer geringfügig erhöhten Latenz bei der Übertragung von Daten. Auf der anderen Seite erfolgen durch die einzelnen Crosspoint-Switche eine Signalaufbereitung und eine Signalvorverzerrung, die längere Übertragungsstrecken ermöglichen. Der notwendige Einsatz von fehlerdetektierenden und/oder korrigierenden Maßnahmen bei der Übertragung wird so auf ein Minimum reduziert, was höhere Nettodatenraten erlaubt und für die Kommunikation benötigte Logikressourcen reduziert.

6.1.3 Logikdichte

Zur Ermittlung eines Vergleichswertes bezüglich der dem Anwender zur Verfügung gestellten Logikressourcen muss ein Ressourcenbegriff für die Logikdichte innerhalb des Clusters gefunden werden. In verschiedenen Ansätzen zur Bestimmung der

Logikdichte eines Systems werden dabei unterschiedliche Zielsetzungen verfolgt. Die in [268] gegebene Definition der Logikdichte, als *FPGA Computational Density* bezeichnet, bezieht neben der Funktionalität des FPGAs auch den Anteil der benötigten Die-Fläche mit ein. Die verschiedenen in [269] aufgezeigten Metriken zur Bestimmung der Logikdichte für unterschiedliche Fertigungstechnologien und Berechnungen erfassen auch die für die Rechenschritte benötigte Energie. Zur Darstellung der Vorteile der dynamischen Rekonfiguration stellt [270] ein als *functional density* bezeichnetes Maß auf, das die Kosten der Rekonfiguration zur Laufzeit dem Nutzen gegenüberstellt. Die betrachteten Definitionen orientieren sich an einem einzelnen FPGA und ermöglichen keine Definition der Logikdichte eines Clusters aus FPGAs. Aus diesem Grund wird in diesem Abschnitt eine Metrik eingeführt, die sich an der physikalischen Massendichte $\rho = \frac{m}{V}$ orientiert und die Besonderheiten der beschriebenen Cluster-Struktur erfasst.

Neben der Flexibilität der Kommunikationsinfrastruktur entscheidet die Latenz bei der Übertragung von Daten darüber, ob vorgegebene Echtzeitbedingungen bei der Ergebnisermittlung eingehalten werden können oder nicht. Beeinflussen lässt sich die Latenz, wie in Kapitel 5.9 gezeigt, durch die Pfadlänge des Übertragungsweges im Verbindungsgraphen des Clusters. Mit der Anzahl der im System verfügbaren FPGAs und dem Durchmesser des die Kommunikationsstruktur beschreibenden Graphen kann die Logikdichte ρ des FPGA-Clusters nach Formel 6.1 bestimmt werden. Die Anzahl der Logikzellenäquivalente, der Durchmesser $D(G)$ der Kommunikationsinfrastruktur und die Bandbreite B, die jedem FPGA zur Kommunikation mit seinen Nachbarn zur Verfügung steht, bilden die Faktoren der Logikdichte. In der Darstellung der Logikdichte in Abbildung 6.2 zeigt sich die enge Kopplung der RAPTOR-XPress-FPGAs bei ψ-optimaler Anordnung der einzelnen Trägersysteme.

$$\rho = \frac{\text{Anzahl Logikzellen-Äquivalente}}{D(G)} \cdot B \tag{6.1}$$

Die durchsatzstarke Kommunikationsinfrastruktur des BEE4 gekoppelt mit modernen FPGAs, die eine große Anzahl an Logikzellen bereitstellen, führt zu einer hohen Logikdichte. Deutlich nachteilig wirkt sich die geringe Anzahl an Logikzellen-Äquivalenten der Virtex-5-FPGAs des DB-V5-Moduls gegenüber den DB-V7-Modulen bei gleichbleibender Architektur des RAPTOR-XPress-Trägersystems aus. Darüber hinaus können die DB-V5-Module Daten nur mit 6,5 Gbit/s übertragen, während die MGTs des DB-V7-Moduls bis zu 11,18 Gbit/s erreichen. Die FPGA-Cluster mit Spartan-3- und Spartan-6-FPGAs haben im Vergleich zu Clustern mit neueren Virtex-Generationen wenig Logikzellen zur Verfügung. Lediglich die Systemarchitekturen CUBE und DNBFC_S12_12 können aufgrund der hohen Anzahl an FPGAs relevante Werte erreichen. In Bezug auf die Logikdichte wirken sich jedoch der Durchmesser und die relativ geringe Bandbreite der Verbindungen negativ auf die Bewertung des CUBE-Clusters aus.

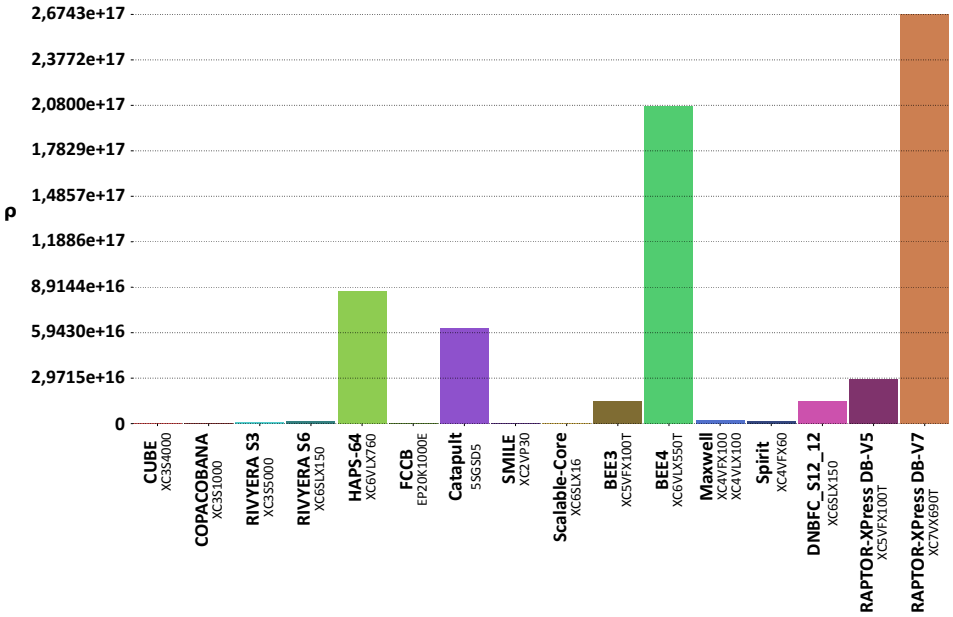


Abbildung 6.2: Logikdichte ρ der vorgestellten FPGA-Cluster

6.2 Gegenüberstellung der FPGA-Cluster-Topologien

Neben der in Kapitel 4.1 vorgestellten Unterteilung der Systeme in UNNS- und NNUS-Architekturen unterscheiden sich die vorgestellten FPGA-Cluster in der Verbindungsstruktur ihrer Elemente. Zum Vergleich wird in Tabelle 6.1 die Organisationsstruktur der FPGAs aufgezeigt. Die Systeme lassen sich dabei in die folgenden drei Organisationstypen klassifizieren:

1. Monolithische Systeme
2. Trägersysteme mit Einzel-FPGA-Modulen
3. Trägersysteme mit Multi-FPGA-Modulen

Monolithische Systeme vereinen alle FPGAs in einem System, wobei mehrere dieser Systeme miteinander verbunden werden können. Ein Vorteil dieses Aufbaus ist, dass die Verbindungsstrukturen zwischen den FPGAs lokal auf einer Leiterkarte geführt werden können und somit auch hochparallele Verbindungen realisierbar sind. Auf den relativ kurzen Strecken variieren die Leitungslängen der häufig als Punkt-zu-Punkt-Verbindungen ausgelegten Kupferbahnen in nur geringem Umfang,

wodurch sich hohe Übertragungsraten erzielen lassen und MGT-Ressourcen für die Kopplung mehrerer Systeme oder zur Anbindung an Hostsysteme zur Verfügung stehen. Nachteilig ist die geringe Flexibilität der teilweise asymmetrischen, fest verdrahteten Kommunikationsinfrastruktur, weshalb ein erhöhter Aufwand bei der Partitionierung der Applikation entsteht.

Trägersysteme mit einzelnen FPGA-Modulen erlauben die individuelle Zusammenstellung mit verschiedenen Funktionalitäten und FPGA-Typen. Somit ist die auf dem Trägersystem umgesetzte Kommunikationsstruktur relevant für den Datenaustausch. Auch hier ist die Verwendung paralleler Punkt-zu-Punkt-Verbindungen möglich. Dieses Konzept stellt allerdings durch die Anbindung der Module über Steckverbinder erhöhte Anforderungen an die Signalqualität der Leitungsführung. Die Cluster-Bildung von mehreren dieser Trägersysteme erfolgt entweder über die Module oder über die Trägersysteme, die ihrerseits die Daten an die Module weiterreichen.

Für Multi-FPGA-Module und ihre Trägersysteme gelten ebenfalls die im vorherigen Abschnitt aufgeführten Anforderungen bezüglich der Signalqualität. Im Bereich der Intra-Modul-Kommunikation verhalten sich diese Systeme äquivalent zu den monolithischen Ansätzen. Nachteilig im Hinblick auf einen universell einsetzbaren FPGA-Cluster ist der Aspekt, dass die Module jeweils eine eigene starre Kommunikationsinfrastruktur besitzen und nicht alle FPGAs Zugriff auf die Ressourcen des Trägersystems haben. Die Anordnung des flüchtigen und persistenten Speichers, der den einzelnen FPGAs zur Verfügung steht, variiert in Abhängigkeit vom jeweiligen System.

Für die Kategorisierung der einzelnen FPGA-Cluster-Systeme in Tabelle 6.1 gelten, sofern eine mehrdeutige Zuordnung denkbar ist, die folgenden Überlegungen. Je nach eingesetztem FPGA-Typ innerhalb des RIVYERA-Cluster variiert die Anzahl der FPGAs pro Modul zwischen einem und acht. Werden ausschließlich Virtex-7-Module eingesetzt, müssen diese der Kategorie *Trägersysteme mit Einzel-FPGA-Modulen* zugeordnet werden. Bei Bestückung mit anderen Modulen entspricht der Aufbau einem *Trägersystem mit Multi-FPGA-Modulen*. Da der RIVYERA V7-2000T jedoch bereits vom Hersteller SciEngines abgekündigt wurde, wird er von der weiteren Betrachtung ausgeschlossen. Die Organisation des ScalableCore-FPGA-Cluster stellt insofern eine Besonderheit dar, als dass jeder Cluster-Baustein jeweils genau einen FPGA aufweist und somit dem Organisationstyp 2 zugeordnet werden könnte. Da jedoch kein Trägersystem benötigt wird und der Aufbau bereits mit der ersten ScalableCore-Unit betrieben werden kann, erfolgt eine Einordnung in die Kategorie 1. Aus dem gleichen Grund werden der Maxwell- und der Spirit-FPGA-Cluster ebenfalls der ersten Kategorie zugeordnet.

Um die realisierbaren Topologievarianten beurteilen zu können, kommen die in Kapitel 3 vorgestellten Metriken zur Anwendung. Die sich daraus ergebenden Werte sind in Tabelle 6.2 aufgeführt. Die Angaben beschreiben den jeweiligen in den Veröffentlichungen vorgestellten FPGA-Cluster, da die meisten Systeme

Tabelle 6.1: Kommunikationsinfrastruktur der FPGA-Cluster

System	Skalierbar	Struktur			Organisationstyp
		Inter-Modul	**Inter-FPGA** (Intra-Modul)	**Inter-System**	
CUBE	ja	2-regulär	2-regulär	2-regulär	3
COPACOBANA	max. 20 Module	Backplane-Bus	Modul-Bus	-	3
RIVYERA	max. 16 Module	2-regulär	2-regulär	-	3 & 2
HAPS-64	ja	-	3-regulär	2-regulär	1
FCCB	max. 15 PUs	-	2-regulär	-	1
Catapult	ja	-	-	4-regulär	1
ScalableCore	ja	-	-	4-regulär	1
BEE3	ja	-	2-regulär	8-regulär	1
BEE4	ja	-	3-regulär	8-regulär	1
Maxwell	ja	1-regulär	-	-	1
Spirit	max. 64 Module	Crosspoint	-	-	1
DNBFC_S12_12	max. 12 DNBFC_S12	2-regulär	3-regulär	2-regulär	3
SMILE	ja	-	-	3-regulär	1
RAPTOR-XPress	ja	Crosspoint, 2-regulär, 3-regulär	-	4-regulär	2

beliebig skalierbar sind, wie Tabelle 6.1 zu entnehmen ist. Die angegebene k-Regularität bezieht sich auf die Systemkomponenten, die die Knoten des Clusters bilden, während der Durchmesser D dem maximalen Pfad zwischen zwei FPGAs des Systems entspricht. Ebenso spiegelt die in der Tabelle aufgeführte Anzahl der FPGAs die Anzahl der insgesamt im FPGA-Cluster für den Anwender zur Verfügung stehenden FPGAs wider. Die folgenden Anmerkungen sind bei der Interpretation der Tabelle zu berücksichtigen.

Bei der Betrachtung des CUBE-FPGA-Clusters wird eine offene Linientopologie angenommen, da andernfalls eine Kommunikation mit einem Hostsystem nicht möglich wäre. Sollten die Endverbindungen zu einem Ring geschlossen sein, verringert sich der Durchmesser entsprechend auf $D\left(G_{CUBE}\right) = 256$.

Aufgrund der systemübergreifenden Busstruktur des COPACOBANA-FPGA-Clusters ist es theoretisch möglich, von jedem Modul an jedes Modul und somit von jedem FPGA an jeden FPGA direkt Daten zu senden. Diese Architekturvariante schlägt sich in Tabelle 6.2 in der Valenz $k = 20$ nieder. Die hierfür notwendige Arbitrierung der exklusiven Kommunikationsressource limitiert jedoch die tatsächlich nutzbare Bandbreite zwischen beliebigen FPGAs.

Die geteilten externen Verbindungen (vgl. Abbildung 4.7; A7 – D7) des HAPS-64 können genutzt werden, um eine busartige Struktur zwischen verschiedenen HAPS-64-Systemen aufzubauen. Andere ebenfalls eingebundene Systeme haben Zugriff auf die jeweils zusammengehörigen Verbinder A7/B7 und C7/D7 sowie die anliegenden FPGAs. Durch den asymmetrischen Aufbau der beiden Schnittstellen AD8 und BC8 und die damit verbundenen Einschränkungen werden diese Verbindungen nicht in die Valenzbetrachtung miteinbezogen. Dennoch ergibt sich für das HAPS-64-System eine Valenz von $k = 28$, da eine Kopplung mehrerer Systeme über jeden der Konnektoren 1 – 6 zuzüglich der geteilten Konnektoren A7/B7, C7/D7, BC8 und AD8 möglich ist. Eine potentielle Weiterverteilung der eingehenden Daten benötigt Logikressourcen und erfordert einen Routing-Ansatz, der Multi-FPGA-Hops verwalten kann.

Der SMILE-FPGA-Cluster nutzt zwar für alle Knoten gleichartige Hardware-Komponenten, verwendet für die Verbindung der einzelnen Elemente allerdings eine Kombination aus systolischem Array und Ringtopologie. Aus diesem Grund basiert die Angabe zur k-Regularität auf der theoretisch mit der eingesetzten Hardware erzielbaren Umsetzung.

Der gleiche Ansatz der Darstellung wurde für den ScalableCore-FPGA-Cluster gewählt, bei dem die Baugruppen, die die Außengrenze des Clusters bilden, lediglich zwei oder drei ihrer vier möglichen Schnittstellen nutzen. Beiden BEE-Systemen gemein ist, dass Teile ihrer MGT-Verbindungen als PCIe-Schnittstelle verwendet werden können. Da es in der FPGA-Logik möglich ist, auch andere Protokolle für die Verbindung umzusetzen können ebenso verschiedene BEE- Systeme über diese Schnittstellen gekoppelt werden, weshalb sie in der Valenzbetrachtung mit berücksichtigt worden sind. Die Gigabit-Ethernet-Schnittstellen sind über einen

externen PHY-Baustein realisiert worden und können deshalb nicht mit einem beliebigen Übertragungsprotokoll betrieben werden, weshalb sie in Tabelle 6.2 nicht erfasst sind.

Die homogenen Komponenten des Spirit-FPGA-Clusters würden einen 6-regulären Aufbau des Clusters unterstützen. Da allerdings innerhalb einer Ebene kein vollständiger 2D-Torus aufgebaut wird, beträgt der Grad der Knoten in der oberen und unteren Reihe einer Ebene lediglich fünf, mit der Folge, dass der Durchmesser des Systems bei $D\left(G_{Spirit}\right) = 7$ liegt.

Für den in Tabelle 6.2 angeführten Vergleich der vorhandenen Logikressourcen wird die Darstellung der Logikzellen-Äquivalente gewählt, welche sich für die LUTs mit vier Eingängen je Logikzelle der Xilinx FPGAs nach Formel 6.2 berechnet [271]. Die Auslastung erfasst einen von Hersteller Xilinx angegebenen Wert, der widerspiegelt, dass abgebildete Funktionen die LUTs und D-Flip-Flops nie ganz vollständig auslasten können. Entsprechend wird der Auslastungswert von 1,125 für LUT4-Architekturen auch für die Altera Logikbausteine des FCCB-FPGA-Clusters angenommen [272]. Neuere FPGAs mit sechs Logikeingängen je LUT bestimmen die Logikzellen-Äquivalente nach Formel 6.3, welche die gegenüber LUT-4-FPGAs gesteigerte Logikkapazität berücksichtigt [273].

$$\text{Logikzellen-Äquivalente}_{\text{LUT4}} = \text{Anzahl LUTs} \times 1,125\ \text{Auslastung} \tag{6.2}$$

$$\text{Logikzellen-Äquivalente}_{\text{LUT6}} = \text{Anzahl LUTs} \times 1,6\ \text{Auslastung} \tag{6.3}$$

Die Spalte *Hostanbindung* beschreibt die Verbindungsart und die Granularität mit denen von einem oder mehreren Hostsystemen auf die Cluster-Komponenten zugegriffen werden kann. Unterschieden wird dabei zwischen Zugriffsmöglichkeiten auf einzelne FPGAs, Knoten im Cluster, die durch Trägersysteme mit mehreren FPGAs dargestellt werden, und einem globalen Zugriff auf den Cluster. Der globale Zugriff erfolgt dabei über eine Standardschnittstelle wie beispielsweise Ethernet oder im Fall von FPGA-Clustern des Organisationstyps 1 über ein auf dem lokalen Server laufendes Betriebssystem.

6.3 Abbildbarkeit von Cluster-Architekturen mit dem RAPTOR-XPress-Cluster

Ein FPGA-Cluster kann nur als universell verwendbares System betrachtet werden, sofern es möglich ist, sowohl die Strukturen als auch die Funktionalität anderer Systeme abzubilden. Im Folgenden werden die zwei beispielhaft ausgewählten Systemarchitekturen des Maxwell-FPGA-Clusters und des Catapult-FPGA-Clusters

Tabelle 6.2: Topologiebewertung der FPGA-Cluster

System	Struktur				Funktionalität		
	k	***D***	**Anzahl Knoten**	**Anzahl FPGAs**	**Logikzellen-Äquivalente**	**Speicher**	**Host-Anbindung**
CUBE	2	511	8	512	31 850 496	-	IDE / Global
COPACOBANA	20	1	20	120	2 073 600	-	USB / Global
RIVYERA S3	2	15	16	128	9 584 640	-	PCIe / Global
RIVYERA S6	2	15	16	128	18 872 704	65 536 MB	PCIe / Global
HAPS-64	28	1	1	4	3 035 136	-	PCIe / FPGA
FCCB	2	7	1	15	648 000	keine Angabe	System Bus / Global
Catapult	4	7	48	48	21 936 000	384 GB	PCIe / FPGA
SMILE	3	10	32	32	986 112	64 GB	Ethernet / Global
ScalableCore	4	14	64	64	933 056	32 768 kB	USB / Global
BEE3	16	2	1	4	409 600	64 GB	OS / Global
BEE4	12	1	1	4	2 199 552	128 GB	OS / Global
Maxwell	4	8	64	64	6 575 616	48 896 MB	PCIe / FPGA
Spirit	5 / 6	7	64	64	3 640 320	20 480 MB	Ethernet / Global
DNBFC_S12_12	2	18	12	144	21 231 792	288 GB	PCIe / Trägersystem
RAPTOR-XPress DB-V5	4	3	16	64	6 553 600	256 GB	PCIe / FPGA
RAPTOR-XPress DB-V7	4	3	16	64	44 359 680	512 GB	PCIe / FPGA

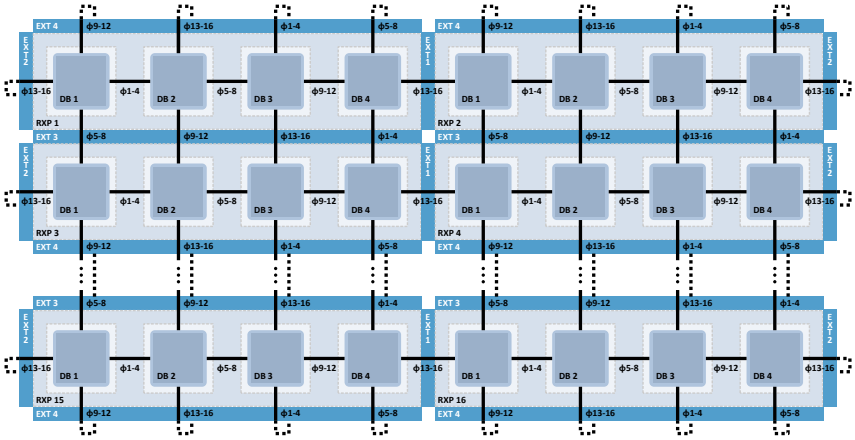


Abbildung 6.3: Abbildung des Maxwell-FPGA-Clusters auf dem RAPTOR-XPress-System

mit Hilfe eines RAPTOR-XPress-Clusters in ihrer Struktur nachgebildet und die somit verfügbare Funktionalität den Originalsystemen gegenübergestellt.

6.3.1 Maxwell

Im Gegensatz zu dem einzelnen FPGA der Nallatech- und Alpha-Data-FPGA-Boards verfügt das RAPTOR-XPress über bis zu vier FPGA-Module. Aufgrund des ähnlichen Aufbaus der Nallatech- und Alpha-Data-Systeme, insbesondere im Hinblick auf deren Verbindungsmöglichkeiten, werden beide im weiteren Verlauf als homogen angesehen. Der Aufbau des Maxwell-FPGA-Clusters mit 8×8 FPGAs lässt sich mit 16 RAPTOR-XPress-Trägersystemen und 64 DBs nachbilden (Abbildung 6.3). Sämtliche dargestellte Verbindungen zwischen den einzelnen DBs laufen über die Crosspoint-Switche des jeweiligen RAPTOR-XPress. Aus Gründen der Übersichtlichkeit werden die Crosspoint-Switche jedoch in der Abbildung nicht dargestellt.

Dank der spezifischen Architektur des RAPTOR-XPress ist eine flexible Verschaltung zwischen den Intra-Board- und Inter-Board-Verbindungen, in der Abbildung als *EXT* bezeichnet, möglich. Aufgrund der geraden Anzahl an RAPTOR-XPress-Trägersystemen können im Rahmen der Inter-Board-Verbindungen gleich bezeichnete Stecker jeweils miteinander verbunden werden (bspw. EXT3 auf EXT3). Zu beachten ist hierbei, dass die Bezeichnungen der horizontalen Verbindungen alternierende Nummern aufweisen (bspw. Ebene 1 zu Ebene 2 über EXT3; Ebene 2 zu Ebene 3 über EXT4; Ebene 3 zu Ebene 4 wieder über EXT3). Eine äquivalente Umsetzung wie im folgenden Abschnitt 6.3.2 ist ebenfalls realisierbar.

Die DB-V5-Module ermöglichen eine in Bezug auf die Anzahl der Logikzellen-Äquivalente annähernd gleichwertige Realisierung, wobei jedem der FPGAs doppelt so viel externer Speicher zur Verfügung steht wie den FPGAs der Nallatech- bzw. Alpha-Data-Boards. In der folgenden Beschreibung (6.4) wird die Verbindung der MGTs des ersten RAPTOR-XPress beispielhaft aufgeführt. Im Gegensatz zu der Einfach-Lane-Verbindung zwischen den einzelnen FPGAs des Maxwell-Systems können beim RAPTOR-XPress-Aufbau je Verbindung vier MGT-Lanes verwendet werden. Darüber hinaus sind die MGTs des RAPTOR-XPress auf eine Übertragungsrate von 6,25 Gbit/s ausgelegt. Somit können bis zu 25 Gbit/s zwischen den FPGAs übertragen werden im Gegensatz zu den 2,5 Gbit/s beim Maxwell-FPGA-Cluster. Bei Verwendung von DB-V7 Modulen erhöht sich die Übertragungsrate auf bis zu 44,72 Gbit/s zwischen zwei FPGAs.

$$
\begin{aligned}
((RXP_1, DB_1), \phi_{1-4}) &\longleftrightarrow ((RXP_1, DB_2), \phi_{1-4}),\\
((RXP_1, DB_1), \phi_{5-8}) &\xleftrightarrow{\text{EXT}_3} ((RXP_3, DB_1), \phi_{5-8}),\\
((RXP_1, DB_1), \phi_{9-12}) &\xleftrightarrow{\text{EXT}_4} ((RXP_{15}, DB_1), \phi_{9-12}),\\
((RXP_1, DB_1), \phi_{13-16}) &\xleftrightarrow{\text{EXT}_2} ((RXP_2, DB_4), \phi_{13-16}),\\
((RXP_1, DB_2), \phi_{5-8}) &\longleftrightarrow ((RXP_1, DB_3), \phi_{5-8}),\\
((RXP_1, DB_2), \phi_{9-12}) &\xleftrightarrow{\text{EXT}_3} ((RXP_3, DB_2), \phi_{9-12}),\\
((RXP_1, DB_2), \phi_{13-16}) &\xleftrightarrow{\text{EXT}_4} ((RXP_{15}, DB_2), \phi_{13-16}),\\
((RXP_1, DB_3), \phi_{9-12}) &\longleftrightarrow ((RXP_1, DB_4), \phi_{9-12}),\\
((RXP_1, DB_3), \phi_{13-16}) &\xleftrightarrow{\text{EXT}_3} ((RXP_3, DB_3), \phi_{13-16}),\\
((RXP_1, DB_3), \phi_{1-4}) &\xleftrightarrow{\text{EXT}_4} ((RXP_{15}, DB_3), \phi_{1-4}),\\
((RXP_1, DB_4), \phi_{13-16}) &\xleftrightarrow{\text{EXT}_1} ((RXP_2, DB_1), \phi_{13-16}),\\
((RXP_1, DB_4), \phi_{1-4}) &\xleftrightarrow{\text{EXT}_3} ((RXP_3, DB_4), \phi_{1-4}),\\
((RXP_1, DB_4), \phi_{5-8}) &\xleftrightarrow{\text{EXT}_4} ((RXP_{15}, DB_4), \phi_{5-8})
\end{aligned}
\tag{6.4}
$$

6.3.2 Catapult

Ebenso wie bei der Übertragung der Maxwell-Architektur auf einen RAPTOR-XPress-Cluster führt bei einer Abbildung des Catapult-Systems die Bündelung von vier FPGAs auf einem RAPTOR-XPress zu einer veränderten Aufteilung der FPGAs, wobei sich jedoch die Topologie der Verbindungsstruktur vollständig darstellen lässt.

Die 36 FPGAs eines Catapult-FPGA-Cluster-Aufbaus können mit neun vollständig mit DBs ausgestatteten RAPTOR-XPress-Trägersystemen nachgebildet werden. Auf einem RAPTOR-XPress-System können jeweils zwei benachbarte FPGAs sowie die in der dahinter liegenden Ebene vorhandenen FPGAs zusammengefasst werden. Entsprechend ergibt sich die in Abbildung 6.4 dargestellte Verbindungsstruktur der RAPTOR-XPress. Zur Konfiguration der Crosspoint-Switche kann wie im vorherigen Beispiel die feingranulare Modellierung aus Kapitel 3.2 verwendet werden. Dargestellt am ersten RAPTOR-XPress ergibt sich die folgende Beschreibung.

$$
\begin{aligned}
((RXP_1, DB_1), \phi_{1-4}) &\xleftrightarrow{\text{EXT}_1 - \text{EXT}_2} ((RXP_3, DB_2), \phi_{1-4}),\\
((RXP_1, DB_1), \phi_{5-8}) &\xleftrightarrow{\text{EXT}_4 - \text{EXT}_3} ((RXP_{10}, DB_3), \phi_{5-8}),\\
((RXP_1, DB_1), \phi_{9-12}) &\longleftrightarrow ((RXP_1, DB_2), \phi_{9-12}),\\
((RXP_1, DB_1), \phi_{13-16}) &\longleftrightarrow ((RXP_1, DB_3), \phi_{13-16}),\\
((RXP_1, DB_2), \phi_{1-4}) &\xleftrightarrow{\text{EXT}_2 - \text{EXT}_1} ((RXP_2, DB_3), \phi_{1-4}),\\
((RXP_1, DB_2), \phi_{5-8}) &\longleftrightarrow ((RXP_1, DB_4), \phi_{5-8}),\\
((RXP_1, DB_2), \phi_{13-16}) &\xleftrightarrow{\text{EXT}_4 - \text{EXT}_3} ((RXP_{10}, DB_4), \phi_{13-16}),\\
((RXP_1, DB_3), \phi_{1-4}) &\longleftrightarrow ((RXP_1, DB_4), \phi_{1-4}),\\
((RXP_1, DB_3), \phi_{5-8}) &\xleftrightarrow{\text{EXT}_3 - \text{EXT}_4} ((RXP_4, DB_1), \phi_{5-8}),\\
((RXP_1, DB_3), \phi_{9-12}) &\xleftrightarrow{\text{EXT}_1 - \text{EXT}_2} ((RXP_3, DB_4), \phi_{9-12}),\\
((RXP_1, DB_4), \phi_{9-12}) &\xleftrightarrow{\text{EXT}_2 - \text{EXT}_1} ((RXP_2, DB_3), \phi_{9-12}),\\
((RXP_1, DB_4), \phi_{13-16}) &\xleftrightarrow{\text{EXT}_3 - \text{EXT}_4} ((RXP_4, DB_2), \phi_{13-16}),
\end{aligned}
\tag{6.5}
$$

Die Verbindung der RAPTOR-XPress-Systeme in einer 3-$\times$-4-Torusstruktur (Abbildung 6.4) führt dazu, dass sich in der Ebene sechs sowie spaltenweise jeweils acht FPAGs zu einem Ring verbinden lassen, was der Originaltopologie des Catapult-FPGA-Clusters entspricht. Aufgrund der Bündelung von vier MGTs in jeder Verbindung zwischen zwei FPGAs erhöht sich die Bandbreite mit DB-V7-Modulen von 20 Gbit/s beim Catapult-Cluster auf 44,72 Gbit/s beim RAPTOR-XPress-Cluster. Während die Stratix-V-FPGAs des Catapult-FPGA-Clusters in Summe über 21 936 000 Logikzellenäquivalente verfügen, beträgt die Anzahl der Logikzellenäquivalente beim RAPTOR-XPress-Cluster 33 269 760. Somit ergibt sich eine Steigerung um ca. 51 % von der Originalarchitektur hin zum Aufbau mit RAPTOR-XPress-Systemen. In Bezug auf den jedem FPGA zur Verfügung stehenden externen Speicher, sind beide Systeme mit jeweils 8 GB je FPGA gleichrangig.

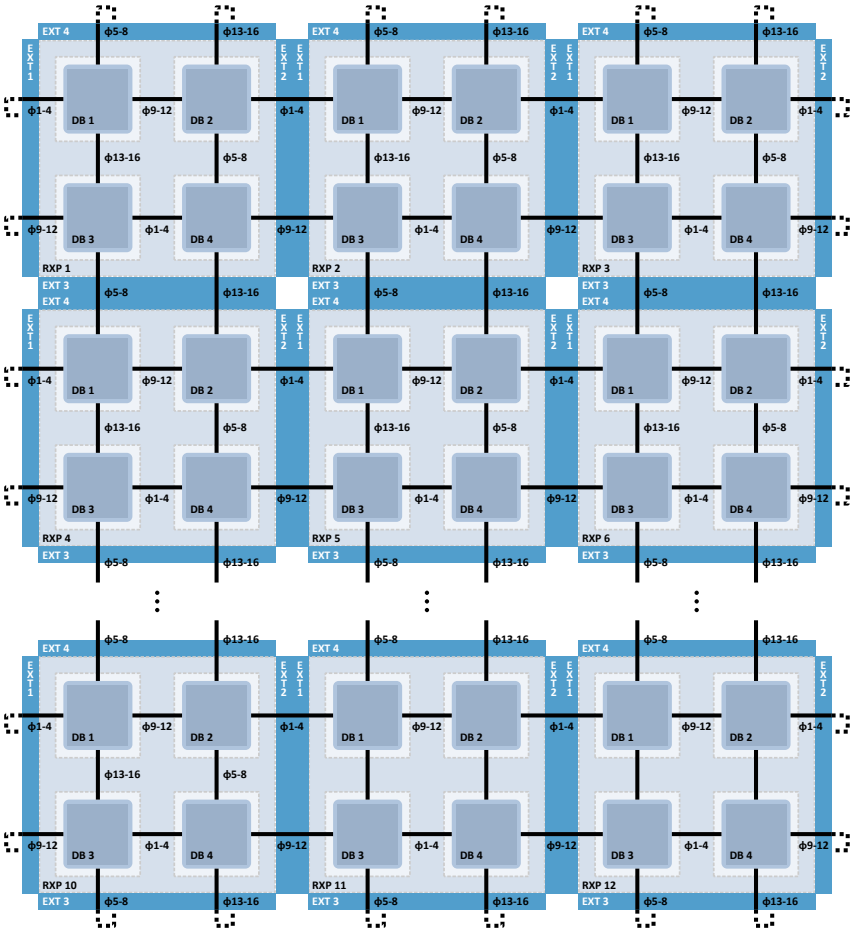


Abbildung 6.4: Abbildung des Catapult-FPGA-Clusters auf dem RAPTOR-XPress-System

6.4 Auswertung

Die verschiedenen FPGA-Cluster unterscheiden sich nicht nur in der Anzahl der dem Anwender zur Verfügung gestellten technologieabhängigen Logikressourcen, sondern auch in der Menge des FPGA-lokalen Speichers. Bedingt durch den Vergleich verschieden alter Systeme, ergeben sich FPGA-Bausteine unterschiedlicher Generationen. So verfügen aktuellere Systeme bereits über FPGAs, die LUT-6 für die Abbildung der Logikfunktionen verwenden, während ältere FPGA-Cluster noch

LUT-4-Technologie implementieren. Dies wird durch die Berechnung der Logikzellen-Äquivalente für die jeweilig eingesetzten FPGAs abstrahiert. Aus diesem Grund kann der CUBE-FPGA-Cluster bedingt durch seine große Anzahl an FPGAs durchaus mit modernen Systemen im Hinblick auf die zur Verfügung stehenden Logikzellen konkurrieren. Die Aufteilung auf viele FPGAs erfordert jedoch eine extreme Partitionierung der Umsetzung, was unter anderem durch die offene Ringtopologie der Kommunikationsinfrastruktur nur für spezielle Aufgabenstellungen sinnvoll möglich ist. Insbesondere Systeme des Organisationstyps 1 und 3 weisen einen deutlich starreren Aufbau der Kommunikationsinfrastruktur auf, was die universelle Verwendung zusätzlich einschränkt.

Generell können für ein dediziertes Aufgabenfeld entworfene FPGA-Cluster trotz möglicherweise ausreichend vorhandener Ressourcen aufgrund der starren Kommunikationsinfrastruktur nur bedingt für andere Zwecke eingesetzt werden. Deutlich wird dies angesichts des systolischen Array des CUBE-Systems. Eine dynamische Auslastung ungenutzter FPGAs durch mehrere parallele Anwender mit unterschiedlichen Aufgabenstellungen ist mit Ausnahme der Systeme SMILE, BEE3, BEE4, Maxwell, Spirit und RAPTOR-XPress-Cluster nicht realisierbar, da keine Möglichkeit für einen gezielten Zugriff auf die einzelnen FPGAs besteht, während Berechnungen auf anderen Teilen des Clusters erfolgen. Dieser Einsatzzweck ist lediglich bei den Systemen Spirit und RAPTOR-XPress vorgesehen.

Die Hostanbindung erfolgt bei den meisten hier aufgeführten Systemen über eine dedizierte globale Schnittstelle und erlaubt keine direkte Kommunikation mit einzelnen Knoten. Die beiden FPGA-Cluster Maxwell und RAPTOR-XPress-Cluster bestehen aus einzelnen PCIe-Einsteckkarten, welche in normalen Serversystemen verwendet werden können. Dank dieses Ansatzes ist ein Ansprechen der Anwendung über den direkt im jeweiligen FPGA realisierten PCIe-Endpunkt möglich.

Verschiedene FPGA-Cluster gewährleisten das Hinzufügen von lokalem Speicher über Standardschnittstellen wie beispielsweise DIMM-Module und bieten somit eine größtmögliche Flexibilität zur Abbildung verschiedener Anwendungen. Diesen Ansatz verfolgen die Systeme Spirit, BEE3, BEE4, Maxwell und RAPTOR-XPress. Dem HAPS64-FPGA-Cluster können über die verschiedenen Erweiterungssteckplätze Speichermodule zugeordnet werden. Nachteilig ist jedoch, dass die erzielbaren Datenraten bedingt durch die Leitungslängen und Steckerübergänge und die damit einhergehende Verschlechterung der Signalqualität bei der Datenübertragung geringer ausfallen müssen als bei Systemen mit lokalem, über kurze Verbindungen angebundenem Speicher. Einen fest verbauten Erweiterungsspeicher bieten die FPGA-Cluster FCCB, ScalableCore und DNBFC_S12_12. Anwendungen, die auf den übrigen Clustern implementiert sind, müssen mit den FPGA-internen Speichern auskommen. In Bezug auf die Menge des lokalen Speichers stechen die BEE-Systeme mit 32 GB bzw. 16 GB hervor.

Fügt man die gewonnenen Informationen zusammen, so zeigt sich, dass der RAPTOR-XPress-Cluster als System des Organisationstyps 2 mit Hilfe seiner ein-

zelnen Trägersysteme jeweils vier eng gekoppelte FPGAs mit auf bis zu 8 GB erweiterbarem lokalem Speicher zur Verfügung stellt. Gleichzeitig wird eine hoch flexible und skalierbare Verbindung mehrerer RAPTOR-XPress ermöglicht, welche eine Anpassung der Bandbreitenbedürfnisse an die Anzahl der über eine direkte Verbindung miteinander verbundenen FPGAs erlaubt. Da bereits während des Entwurfsprozesses die parallele Verwendung des auf diese Weise entstehenden FPGA-Clusters von mehreren Anwendern vorgesehen wurde, kann eine gute Auslastung des Systems erreicht werden. Die Flexibilität der Kommunikationsinfrastruktur erlaubt eine Anpassung an die Topologieanforderungen der jeweiligen Zielanwendung. Während andere FPGA-Cluster beispielsweise über größere lokale Speicher verfügen, ergibt sich aus der Kombination von dem modularen Aufbau, der flexiblen Kommunikationsinfrastruktur, welche eine nach oben offene Skalierung des Systems gewährleistet, der integrierten PCIe-Hostanbindung jedes der vier DBs je RAPTOR-XPress und der Größe des lokalen Speichers eine echte universell und für den Mehrbenutzerbetrieb geeignete FPGA-Cluster-Lösung.

Die Auslegung der Kommunikationsinfrastruktur hat nicht nur Einfluss auf die Flexibilität des Gesamtsystems, sondern bestimmt auch maßgeblich die erzielbare Logikdichte. Nicht optimierbare Topologien können die notwendigen Verbindungsschritte nicht minimieren, weshalb längere Wege zurückgelegt werden müssen. Anhand der Logikdichte zeigt sich ein deutlicher Vorteil des RAPTOR-XPress.

6.5 Zusammenfassung

Um beurteilen zu können, ob die Vorgabe, einen universellen und skalierbaren FPGA-Cluster zu entwickeln, erreicht wurde, erfolgte im Rahmen dieses Kapitels eine Gegenüberstellung der verschiedenen in den Kapiteln 4 und 5.6 vorgestellten FPGA-Clustern. Die realisierbaren Kommunikationsinfrastrukturmodelle wurden dabei besonders herausgestellt. Als Vergleichsgröße wurde die Flexibilität der Cluster-Topologie benannt. Es wurde gezeigt, dass sich die Flexibilität einer Kommunikationsstruktur anhand der Eigenschaften des sie beschreibenden Graphen darstellen lässt. Entsprechend ihres Systemaufbaus wurden die Cluster kategorisiert und anhand ihrer dem Anwender zur Verfügung stehenden Ressourcen beurteilt. In der abschließenden Auswertung konnte gezeigt werden, dass die verschiedenen Systeme, wie zu erwarten, über individuelle Stärken und Schwächen verfügen. Insbesondere die Flexibilität und Leistungsfähigkeit der Kommunikationsinfrastruktur einschließlich der Hostanbindung bestätigen das Systemkonzept für einen universell verwendbaren, skalierbaren FPGA-Cluster, welcher aus den in Kapitel 5 vorgestellten RAPTOR-XPress-Trägersystemen gebildet wird. Die erzielte Logikdichte für einen Beispielaufbau mit 64 FPGAs liegt bei $2,6743 \cdot 10^{17}$ und somit über den Werten anderer Cluster. Layout-bedingte Restriktionen sind an der Anzahl möglicher Verbindungen zu erkennen. So zeichnen sich die meisten Cluster dadurch aus, dass

sie maximal 8-reguläre Topologien im Bereich der Inter-Systemkommunikation bilden können. Zur Steigerung der Bandbreite oder aus mechanischen Gründen, etwa einer beschränkten Leiterkartengröße, werden jedoch 2- bis 4-reguläre Strukturen am häufigsten umgesetzt. Das RAPTOR-XPress-System realisiert zur Bündelung mehrerer MGT-Verbindungen und der damit einhergehenden Steigerung der Bandbreite ebenfalls einen 4-regulären Aufbau.

Aufgrund des flexiblen Aufbaus des RAPTOR-XPress konnte gezeigt werden, dass es möglich ist, alternative Architekturen mit dem RAPTOR-XPress abzubilden. Die Struktur der zwei FPGA-Cluster Maxwell und Catapult als Beispiele für derzeit relevante Systeme konnte mit Hilfe mehrerer RAPTOR-XPress-Trägersysteme und DBs vollständig dargestellt werden.

7 Zusammenfassung und Fazit

Im Rahmen dieser Arbeit wurde ein Verfahren aufgezeigt, mit dessen Hilfe kompakte regelmäßige Graphen für eine gegebene Anzahl an Knoten gefunden werden können. Relevanz gewinnt dieses Vorgehen bei der Auffindung einer optimalen Kommunikationstopologie für ein neu zu entwickelndes FPGA-Cluster-System.

Um eine universell einsetzbare Lösung zu realisieren, wurden typische Anwendungsgebiete von FPGAs und FPGA-Clustern identifiziert und hinsichtlich ihrer Anforderungen an die verwendete Hardware-Architektur untersucht. Darüber hinaus wurde die Motivation für eine Implementierung auf rekonfigurierbarer Logik erläutert. Als die drei wesentlichen Gründe, FPGAs Umsetzungen auf CPUs und GPGPUs vorzuziehen, sind die Beschleunigung der Verarbeitung, eine harte Echtzeitfähigkeit und die Energieeffizienz herausgearbeitet worden. Deutlich wurde bei der Untersuchung, dass insbesondere die Kommunikationsinfrastruktur zwischen den einzelnen FPGAs eines Clusters entscheidend dafür ist, ob sich eine Vielzahl unterschiedlicher Anwendungen auf einem bestimmten System implementieren lässt.

Um die Kommunikationsressourcen eines FPGA-Clusters beschreiben zu können, wurde eine abstrahierende Darstellung vorgestellt, die es ermöglicht, sowohl feingranular auf Schnittstellenebene der eingesetzten Hardware als auch auf Systemebene die Verbindungsstruktur des Systems abzubilden. Weiterführend wurde eine Methode zur Auffindung latenzminimaler Topologien entwickelt. Mit Hilfe dieses Verfahrens ist es möglich, im Gegensatz zu typischerweise in Cluster-Architekturen eingesetzten regelmäßigen Strukturen kompaktere Varianten umzusetzen. Möglich wird dieser Ansatz in der Praxis aufgrund der Tatsache, dass, anders als zum Beispiel in klassischen Computer-Netzwerken, bereits im Voraus bekannt ist, welche Kommunikationsressourcen während der anstehenden Berechnungen benötigt werden. Aus diesem Grund erscheint die Kommunikation von außen betrachtet statisch, und eine dynamische Routenfindung für jede einzelne Übertragung entfällt.

Bereits in der Forschung eingesetzte und in Veröffentlichungen erwähnte FPGA-Cluster-Systeme wurden hinsichtlich ihrer individuellen Eigenschaften untersucht. Dabei fiel auf, wie aufgrund der Untersuchung der Anwendungsgebiete von FPGAs zu erwarten, dass sich dediziert für eine bestimmte Klasse von Anwendungen oder sogar für eine einzelne Implementierung optimierte Architekturen nur sehr eingeschränkt als Zielplattform für andere Applikationen eignen. Die meisten der betrachteten Systeme schließen architekturbedingt darüber hinaus eine parallele Nutzung durch unterschiedliche Benutzer mit verschiedenen Implementierungen

aus. In Folge dessen reduziert sich die Auslastungseffizienz der Systeme für allgemeine Anwendungen. Die herausgestellten Eigenschaften der unterschiedlichen Systeme schaffen im weiteren Verlauf der Arbeit eine Vergleichsbasis hinsichtlich des neu entwickelten modularen RAPTOR-XPress und des darauf aufbauenden RAPTOR-XPress-Clusters.

Mit Hilfe der vorangegangenen Untersuchungen wurde schließlich das modulare FPGA-Trägersystem RAPTOR-XPress entwickelt, bei dem bereits zum Entwurfszeitpunkt ein besonderer Fokus auf die Kommunikationsinfrastruktur gelegt wurde. Zur Steigerung der Flexibilität und somit auch der offenen Skalierbarkeit des aufbauenden FPGA-Clusters wurde auf Systemebene ein Crosspoint-Switch-Array implementiert. Diese Architekturentscheidung erlaubt eine Bündelung oder Aufteilung der Kommunikationsbandbreite jedes FPGAs. So ist es möglich, dass ein FPGA mit bis zu 16 weiteren im Cluster verteilten FPGAs eine direkte Verbindung aufbauen kann, ohne dass weitere FPGAs Logikressourcen aufwenden müssen, um als Kommunikationsbrücke zu dienen. Die Anwendungsentwicklung unterstützende Logikblöcke ermöglichen eine Abstraktion der zugrunde liegenden Hardware, so dass sich die verbundenen FPGAs wie eine zusammenhängende Logikfläche darstellen. Die für die Anwendung vorgesehenen Erweiterungsmodule DB-V5 und DB-V7, welche parallel zum im Rahmen dieser Arbeit entstandenen RAPTOR-XPress in der Arbeitsgruppe *Kognitronik und Sensorik* der Universität Bielefeld entwickelt worden sind, wurden ebenfalls vorgestellt. Die weiteren speziellen Architekturmerkmale des RAPTOR-XPress und des RAPTOR-XPress-Clusters wurden herausgearbeitet. Zusammen mit der auf den Hostsystemen auch im Fernzugriff verfügbaren Software-Schnittstelle, die als Teil einer parallelen Entwicklung in der Arbeitsgruppe *Kognitronik und Sensorik* entstanden ist, können verschiedene Benutzer gleichzeitig die Ressourcen des Clusters nutzen. In der Anforderungsverifikation konnte festgestellt werden, dass das RAPTOR-XPress die Möglichkeit bietet, ein skalierbares, universell einsetzbares und von mehreren Anwendern gleichzeitig verwendbares Cluster-System aufzubauen.

Anhand des Vergleichs der verschiedenen FPGA-Cluster konnte gezeigt werden, dass durch die besondere Kommunikationsarchitektur des RAPTOR-XPress ein in Bezug auf die Verwendbarkeit einzigartiges Cluster-Konzept realisiert worden ist. Der konsequent auf Skalierbarkeit und Modularität hin ausgerichtete Aufbau ermöglicht über die DBs das Einbringen von Logik- und Speicherressourcen, welche verglichen mit den anderen Systemen mindestens ebenso leistungsstark sind. Im Hinblick auf die Konnektivität innerhalb des Clusters zeigt sich eine deutliche Überlegenheit des RAPTOR-XPress. Alternative aktuelle Cluster-Konzepte lassen sich in ihrer Struktur nachbilden, wobei die Bandbreite der Verbindungen auf dem RAPTOR-XPress gegenüber den Originalansätzen verdoppelt bis verzehnfacht werden kann.

Durch die Flexibilität der Kommunikationsstruktur ist eine Umsetzung einer ψ-optimalen Topologie möglich, was zu deutlich verringerten Latenzen in der Kom-

munikation führt, da die Pfadlängen innerhalb des Clusters minimiert werden. Am Beispiel eines Cluster-Aufbaus mit 64 FPGAs konnte für den Betrieb mit mehreren Anwendungen, welche zu unterschiedlichen Zeitpunkten starten können, durch Verwendung einer ψ-optimalen Topologie ein um 8,5 % vergrößerter Platzierungsraum gegenüber einer 2D-Torus Umsetzung erreicht werden. Darüber hinaus reduziert sich die Exzentrizitätssumme um bis zu 31 % im Vergleich zu klassischen regelmäßigen Strukturen.

Die aufgestellte Metrik der Logikdichte ermöglicht einen Vergleich verschiedener FPGA-Cluster unabhängig von der eingesetzten FPGA-Technologie. Hier konnte gezeigt werden, dass ein mit DB-V7-Modulen ausgestatteter RAPTOR-XPress-Cluster im Vergleich eine um 28 % höhere Logikdichte erreicht als das BEE4-System mit der nächst kleineren Logikdichte.

Im Hinblick auf künftige Systementwürfe können die gewonnenen Erkenntnisse Entscheidungshilfen bei der Auslegung der Cluster-bildenden Kommunikationsinfrastruktur sein. Wie gezeigt wurde, gewährleistet die Kopplung von Teilkomponenten mit einer hohen Valenz die Realisierung von Systemen mit der höchsten Logikdichte. Die vorgestellte Methode zur Optimierung der Cluster-Topologie kann auch in Zukunft für kabelgebundene Übertragungsstandards verwendet werden, um einen kompakten Cluster umzusetzen. Durch den Entwurf eines alternativen High-Speed-Piggyback für das RAPTOR-XPress ist die Realisierung komplexerer Cluster-Strukturen ohne erforderliche Veränderungen am eigentlichen Trägersystem und an den DBs möglich. Eine Revision des RAPTOR-XPress-Trägersystems mit 20 MGT-Verbindungen sowie die Verwendung von DB-V7-Modulen gewährleisten eine weitere Leistungssteigerung in Bezug auf die Kommunikationsbandbreite. Die grundlegende Systemarchitektur mit dem Alleinstellungsmerkmal des Crosspoint-Switch-Array zur Umsetzung der Intra- und Inter-Systemkommunikation bleibt selbst im Fall kleiner Änderungen erhalten und unterstreicht somit den zukunftssicheren Ansatz.

Literaturverzeichnis

[1] FAGGIN, F. ; HOFF, M. E. ; MAZOR, S. ; SHIMA, M.: The history of the 4004. In: *IEEE Micro* 16 (1996), Dezember, Nr. 6, S. 10 – 20. – ISSN 0272-1732

[2] SCROFANO, R. ; CHOI, Seonil ; PRASANNA, V. K.: Energy Efficiency of FPGAs and Programmable Processors for Matrix Multiplication. In: *2002 IEEE International Conference on Field-Programmable Technology, 2002. (FPT). Proceedings.*, Dec 2002, S. 422 – 425

[3] ADHINARAYANAN, V. ; KOEHN, T. ; KEPA, K. ; FENG, W. c. ; ATHANAS, P.: On the Performance and Energy Efficiency of FPGAs and GPUs for Polyphase Channelization. In: *2014 International Conference on ReConFigurable Computing and FPGAs (ReConFig14)*, Dec 2014, S. 1 – 7. – ISSN 2325-6532

[4] MOORE, Gordon E.: Cramming more components onto integrated circuits, Reprinted from Electronics, volume 38, number 8, April 19, 1965, pp.114 ff. In: *Solid-State Circuits Society Newsletter, IEEE* 11 (2006), Sept, Nr. 5, S. 33 – 35. – ISSN 1098-4232

[5] INTERNATIONAL TECHNOLOGY ROADMAP FOR SEMICONDUCTORS (ITRS): ITRS Executive Summary / International Technology Roadmap for Semiconductors (ITRS). 2011. – Forschungsbericht

[6] MENCER, O. ; TSOI, Kuen H. ; CRAIMER, S. ; TODMAN, T. ; LUK, W. ; WONG, Ming Y. ; LEONG, Philip: Cube: A 512-FPGA cluster. In: *Programmable Logic, 2009. SPL. 5th Southern Conference on*, April 2009, S. 51 – 57

[7] XILINX: Xilinx Stacked Silicon Interconnect Technology Delivers Breakthrough FPGA Capacity, Bandwidth, and Power Efficiency / Xilinx. Dezember 2012 (WP380 (v1.2)). – White Paper

[8] DIN: *DIN 1333 Zahlenangaben*. Deutsches Institut für Normung (Veranst.), Februar 1992

[9] DIN: *DIN 461 Graphische Darstellung in Koordinatensystemen*. Deutsches Institut für Normung (Veranst.), März 1973

[10] DIN: *DIN 1338 Formelschreibweise und Formelsatz*. Deutsches Institut für Normung (Veranst.), März 2011

[11] DIN: *DIN 1302 Allgemeine mathematische Zeichen und Begriffe*. Deutsches Institut für Normung (Veranst.), 1999

[12] DIN: *DIN EN 80000-13 Grössen und Einheiten – Teil 13: Informationswissenschaft und -technik*. Deutsches Institut für Normung (Veranst.), Januar 2009

[13] DIN: *DIN 5473 Logik und Mengenlehre*. Deutsches Institut für Normung (Veranst.), Juli 1992

[14] DEISER, Oliver: Einführung in die Mengenlehre. In: *Einführung in die Mengenlehre - Die Mengenlehre Georg Cantors und ihre Axiomatisierung durch Ernst Zermelo*. Springer Berlin Heidelberg, 2010 (Springer-Lehrbuch). – ISBN 978-3-642-01445-1

[15] CARTER, William S. ; DUONG, Khue ; FREEMAN, Ross H. ; HSIEH, Hung-Cheng ; JA, Jason Y. ; MAHONEY, John E. ; NGO, Luan T. ; SZE, Shelly L.: A User Programmable Reconfigurable Logic Array. In: *Proceedings of the IEEE Custom Integrated Circuits Conference* IEEE (Veranst.), Mai 1986, S. 233 – 235

[16] TESSIER, R. ; POCEK, K. ; DEHON, A.: Reconfigurable Computing Architectures. In: *Proceedings of the IEEE* 103 (2015), März, Nr. 3, S. 332 – 354. – ISSN 0018-9219

[17] DEHON, A.: The density advantage of configurable computing. In: *Computer* 33 (2000), April, Nr. 4, S. 41 – 49. – ISSN 0018-9162

[18] XILINX: *7 Series FPGAs Overview*. DS180 (v2.0), September 2016

[19] XILINX: *7 Series FPGAs Configurable Logic Block*. UG474 (v1.7), November 2014

[20] AMDAHL, Gene M.: Validity of the single processor approach to achieving large scale computing capabilities. In: *Proceedings of the April 18-20, 1967, spring joint computer conference*. New York, NY, USA : ACM, 1967 (AFIPS '67 (Spring)), S. 483 – 485

[21] GREENLAW, Raymond ; HOOVER, H. J. ; RUZZO, Walter L.: *Limits to Parallel Computation: P-Completeness Theory*. OXFORD UNIVERSITY PRESS, 1995

[22] MACK, C.A.: Fifty Years of Moore's Law. In: *Semiconductor Manufacturing, IEEE Transactions on* 24 (2011), Nr. 2, S. 202 – 207. – ISSN 0894-6507

[23] KLAUER, Bernd: The Convey Hybrid-Core Architecture. In: VANDERBAUWHEDE, Wim (Hrsg.) ; BENKRID, Khaled (Hrsg.): *High-Performance Computing Using FPGAs*. Springer New York, 2013, S. 431 – 451. – ISBN 978-1-4614-1790-3

[24] HOANG, Trang ; NGUYEN, Van L.: An Efficient FPGA Implementation of the Advanced Encryption Standard Algorithm. In: *Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), 2012 IEEE RIVF International Conference on*, Februar 2012, S. 1 – 4

[25] LI, Jian ; SARUNIC, M.V. ; SHANNON, L.: Scalable, High Performance Fourier Domain Optical Coherence Tomography: Why FPGAs and Not GPGPUs. In: *Field-Programmable Custom Computing Machines (FCCM), 2011 IEEE 19th Annual International Symposium on*, Mai 2011, S. 49 – 56

[26] KESTUR, S. ; DANTARA, D. ; NARAYANAN, V.: SHARC: A streaming model for FPGA accelerators and its application to Saliency. In: *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, März 2011, S. 1 – 6. – ISSN 1530-1591

[27] GMBH af inventions: *Ethernet Switch IP-Core Product Brief*. v1.1, April 2015. – URL `http://www.af-inventions.de/de/fpga-loesungen/ethernet-switch-ip-core/`

[28] XILINX: *4K2K (Ultra HD) and 8K4K (Super Hi-Vision) Digital Television*. online. April 2014. – URL `http://www.xilinx.com/applications/broadcast/digital-television.html`

[29] GUPTA, Pratishtha: A survey of techniques and applications for real time image processing. In: *Journal of Global Research in Computer Science* 4 (2013), Nr. 8, S. 30 – 39

[30] HANUMANTHARAJU, M.C. ; VISHALAKSHI, G.R. ; HALVI, Srinivas ; SATISH, S.B.: A Novel FPGA Based Reconfigurable Architecture for Image Color Space Conversion. In: KRISHNA, P.Venkata (Hrsg.) ; BABU, M.Rajasekhara (Hrsg.) ; ARIWA, Ezendu (Hrsg.): *Global Trends in Information Systems and Software Applications* Bd. 270. Springer Berlin Heidelberg, 2012, S. 292 – 301. – ISBN 978-3-642-29215-6

[31] LU, Ching-Hsi ; HSU, Hong-Yang ; WANG, Lei: A New Contrast Enhancement Technique Implemented on FPGA for Real Time Image Processing. In: *Intelligent Information Hiding and Multimedia Signal Processing, 2009. IIH-MSP '09. Fifth International Conference on*, Sept 2009, S. 542 – 545

[32] TAN, Xin ; LAI, Shiming ; WANG, Bin ; ZHANG, Maojun ; XIONG, Zhihui: A simple gray-edge automatic white balance method with FPGA implementation. In: *Journal of Real-Time Image Processing* (2013), S. 1 – 11. – ISSN 1861-8200

[33] VANISHREE ; REDDY, K.V.R.: Implementation of pipelined sobel edge detection algorithm on FPGA for High speed applications. In: *Emerging Trends in Communication, Control, Signal Processing Computing Applications (C2SPCA), 2013 International Conference on*, Oktober 2013, S. 1 – 5

[34] BROUSSEAU, B. ; ROSE, J.: An Energy-Efficient, Fast FPGA Hardware Architecture for OpenCV-Compatible Object Detection. In: *Field-Programmable Technology (FPT), 2012 International Conference on*, Dezember 2012, S. 166 – 173

[35] RUMMELE-WERNER, M. ; PERSCHKE, T. ; BRAUN, L. ; HUBNER, M. ; BECKER, J.: A FPGA based fast runtime reconfigurable real-time Multi-Object-Tracker. In: *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on*, Mai 2011, S. 853 – 856. – ISSN 0271-4302

[36] TOMASI, M. ; VANEGAS, M. ; BARRANCO, F. ; DIAZ, J. ; ROS, E.: Real-Time Architecture for a Robust Multi-Scale Stereo Engine on FPGA. In: *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* 20 (2012), Dezember, Nr. 12, S. 2208 – 2219. – ISSN 1063-8210

[37] GRIESSL, René ; HERBRECHTSMEIER, Stefan ; PORRMANN, Mario ; RÜCKERT, Ulrich: A Low-Power Vision Processing Platform for Mobile Robots, 2011 (Proceedings of the FPL2011 Workshop on Computer Vision on Low-Power Reconfigurable Architectures)

[38] NEEDLEMAN, Saul B. ; WUNSCH, Christian D.: A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins. In: *Journal of Molecular Biology* 48 (1970), Nr. 3, S. 443 – 453. – ISSN 0022-2836

[39] WATERMAN, M.S ; SMITH, T.F ; BEYER, W.A: Some Biological Sequence Metrics. In: *Advances in Mathematics* 20 (1976), Juni, Nr. 3, S. 367 – 387. – ISSN 0001-8708

[40] SMITH, T.F. ; WATERMAN, M.S.: Identification of Common Molecular Subsequences. In: *Journal of Molecular Biology* 147 (1981), mar, Nr. 1, S. 195 – 197. – ISSN 0022-2836

[41] ALTSCHUL, Stephen F. ; GISH, Warren ; MILLER, Webb ; MYERS, Eugene W. ; LIPMAN, David J.: Basic Local Alignment Search Tool. In: *Journal of Molecular Biology* 215 (1990), oct, Nr. 3, S. 403 – 410. – ISSN 0022-2836

[42] RAMDAS, T. ; EGAN, G.: A Survey of FPGAs for Acceleration of High Performance Computing and their Application to Computational Molecular Biology. In: *TENCON 2005 2005 IEEE Region 10*, November 2005, S. 1 – 6

[43] SARKAR, S. ; MAJUMDER, T. ; KALYANARAMAN, A. ; PANDE, P.P.: Hardware Accelerators for Biocomputing: A Survey. In: *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, Mai 2010, S. 3789 – 3792

[44] ALACHIOTIS, N. ; SOTIRIADES, E. ; DOLLAS, A. ; STAMATAKIS, A.: Exploring FPGAs for Accelerating the Phylogenetic Likelihood Function. In: *Parallel Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, Mai 2009, S. 1 – 8. – ISSN 1530-2075

[45] CONTI, V. ; VITABILE, S. ; MILITELLO, C. ; LANZA, B. ; SORBELLO, F.: An Embedded Processor for Metabolic Networks Optimization. In: *Complex, Intelligent and Software Intensive Systems (CISIS), 2011 International Conference on*, Juni 2011, S. 77 – 84

[46] ŠKODA, P. ; MEDVED ROGINA, B. ; SRUK, V.: FPGA implementations of data mining algorithms. In: *MIPRO, 2012 Proceedings of the 35th International Convention*, Mai 2012, S. 362 – 367

[47] NARAYANAN, R. ; HONBO, D. ; MEMIK, G. ; CHOUDHARY, A. ; ZAMBRENO, J.: An FPGA Implementation of Decision Tree Classification. In: *Design, Automation Test in Europe Conference Exhibition, 2007. DATE '07*, April 2007, S. 1 – 6

[48] DAS, A. ; NGUYEN, D. ; ZAMBRENO, J. ; MEMIK, G. ; CHOUDHARY, A.: An FPGA-Based Network Intrusion Detection Architecture. In: *Information Forensics and Security, IEEE Transactions on* 3 (2008), März, Nr. 1, S. 118 – 132. – ISSN 1556-6013

[49] LIN, Zhongduo ; LO, C. ; CHOW, P.: K-means implementation on FPGA for high-dimensional data using triangle inequality. In: *Field Programmable Logic and Applications (FPL), 2012 22nd International Conference on*, August 2012, S. 437 – 442

[50] BREIMAN, L. ; FRIEDMAN, J. ; STONE, C.J. ; OLSHEN, R.A.: *Classification and Regression Trees*. Taylor & Francis Ltd, Januar 1984 (The Wadsworth and Brooks-Cole statistics-probability series). – ISBN 9780412048418

[51] BURGES, Christopher J. C.: A Tutorial on Support Vector Machines for Pattern Recognition. In: *Data Mining and Knowledge Discovery* 2 (1998), Juni, Nr. 2, S. 121 – 167. – ISSN 1384-5810

[52] SMITH, Kate A.: Neural Networks for Combinatorial Optimization: A Review of More Than a Decade of Research. In: *INFORMS Journal on Computing* 11 (1999), Februar, Nr. 1, S. 15 – 34. – ISSN 1526-5528

[53] ZHANG, Q. ; CHAMBERLAIN, R.D. ; INDECK, R.S. ; WEST, B.M. ; WHITE, J.: Massively Parallel Data Mining Using Reconfigurable Hardware: Approximate String Matching. In: *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, April 2004, S. 259 – 266

[54] BROULIM, P. ; BROULIM, J. ; GEORGIEV, V. ; MOLDASCHL, J.: Very high resolution time measurement in FPGA. In: *Telecommunications Forum Telfor (TELFOR), 2014 22nd*, November 2014, S. 745 – 748

[55] CAPPELLI, L. ; CRETI, P. ; GRANCAGNOLO, F.: A Cluster Timing Algorithm for drift chambers readout electronics. In: *Advances in Sensors and Interfaces (IWASI), 2011 4th IEEE International Workshop on*, Juni 2011, S. 43 – 44

[56] SCHRYVER, Christian de ; SCHRYVER, Christian de (Hrsg.): *FPGA Based Accelerators for Financial Applications*. 1. Springer International Publishing, 2015

[57] BRUGGER, C. ; SCHRYVER, C. de ; WEHN, N. ; OMLAND, S. ; HEFTER, M. ; RITTER, K. ; KOSTIUK, A. ; KORN, R.: Mixed precision multilevel Monte Carlo on hybrid computing systems. In: *Computational Intelligence for Financial Engineering Economics (CIFEr), 2104 IEEE Conference on*, März 2014, S. 215 – 222

[58] POTTATHUPARAMBIL, R. ; COYNE, J. ; ALLRED, J. ; LYNCH, W. ; NATOLI, V.: Low-Latency FPGA Based Financial Data Feed Handler. In: *Field-Programmable Custom Computing Machines (FCCM), 2011 IEEE 19th Annual International Symposium on*, Mai 2011, S. 93 – 96

[59] DENHOLM, S. ; INOUE, H. ; TAKENAKA, T. ; LUK, W.: Application-Specific Customisation of Market Data Feed Arbitration. In: *Field-Programmable Technology (FPT), 2013 International Conference on*, Dezember 2013, S. 322 – 325

[60] KLIR, George J. ; FOLGER, Tina A.: *Fuzzy sets, uncertainty, and information*. Englewood Cliffs, NJ: Prentice Hall, 1988. – xi + 355 S. – ISBN 0-13-345984-5

[61] SULAIMAN, Nasri ; OBAID, Zeyad A. ; MARHABAN, MH ; HAMIDON, MN: FPGA-Based Fuzzy Logic: Design andApplications – a Review. In: *IACSIT International Journal of Engineering and Technology* 1 (2009), Nr. 5, S. 491 – 503

[62] HASSAN, Mohammed Y. ; SHARIF, Waleed F.: Design of FPGA based PID-like Fuzzy Controller for Industrial Applications. In: *IAENG International Journal of Computer Science* 34 (2007), November, Nr. 2

[63] ALVAREZ, J. ; LAGO, A. ; NOGUEIRAS, A. ; MARTINEZ-PENALVER, C. ; MARCOS, J. ; DOVAL, J. ; LOPEZ, O.: FPGA Implementation of a Fuzzy Controller for Automobile DC-DC Converters. In: *Field Programmable Technology, 2006. FPT 2006. IEEE International Conference on*, Dezember 2006, S. 237 – 240

[64] WANG, Hsiao-Ping: Design of Fast Fuzzy Controller and its Application on Position Control of DC Motor. In: *Consumer Electronics, Communications and Networks (CECNet), 2011 International Conference on*, April 2011, S. 4902 – 4905

[65] KUMAR, S. ; PAAR, C. ; PELZL, J. ; PFEIFFER, G. ; SCHIMMLER, Manfred: COPACOBANA A Cost-Optimized Special-Purpose Hardware for Code-Breaking. In: *Field-Programmable Custom Computing Machines, 2006. FCCM '06. 14th Annual IEEE Symposium on*, April 2006, S. 311 – 312

[66] ABBAS, A. ; RATHJE, C.A. ; WIENBRANDT, L. ; SCHIMMLER, M.: Dictionary Attack on TrueCrypt with RIVYERA S3-5000. In: *Parallel and Distributed Systems (ICPADS), 2012 IEEE 18th International Conference on*, Dezember 2012, S. 93 – 100. – ISSN 1521-9097

[67] GÜNEYSU, T. ; KASPER, T. ; NOVOTNÝ, M. ; PAAR, C. ; RUPP, A.: Cryptanalysis with COPACOBANA. In: *Computers, IEEE Transactions on* 57 (2008), November, Nr. 11, S. 1498 – 1513. – ISSN 0018-9340

[68] GÜNEYSU, T. ; PAAR, C. ; PFEIFFER, G. ; SCHIMMLER, Manfred: Enhancing COPACOBANA for advanced applications in cryptography and cryptanalysis. In: *Field Programmable Logic and Applications, 2008. FPL 2008. International Conference on*, September 2008, S. 675 – 678

[69] PUTTMANN, Christoph: *Ressourceneffiziente Hardware-Software-Kombinationen für Kryptographie mit elliptischen Kurven*, Universität Bielefeld, Dissertation, 2013

[70] XILINX: *LogiCORE IP DSP48 Macro*. PG148 (v3.0), Dezember 2013

[71] NASCIMENTO, P.S.B. ; SOUZA, H.E.P. de ; NEVES, F.A.S. ; DOMINGUES, M.A.O.: FPGA Design Methodology for DSP Industrial Applications - A Case Study of a Three-Phase Positive-Sequence Detector. In: *Integrated Circuits and Systems Design (SBCCI), 2012 25th Symposium on*, August 2012, S. 1 – 6

[72] BROSSER, F. ; CHEAH, Hui Y. ; FAHMY, S.A.: Iterative floating point computation using FPGA DSP blocks. In: *Field Programmable Logic and Applications (FPL), 2013 23rd International Conference on*, September 2013, S. 1 – 6

[73] AKKALA, V. ; RAJALAKSHMI, P. ; KUMAR, P. ; DESAI, U.B.: FPGA Based Ultrasound Backend System with Image Enhancement Technique. In: *Biosignals and Biorobotics Conference (2014): Biosignals and Robotics for Better and Safer Living (BRC), 5th ISSNIP-IEEE*, Mai 2014, S. 1 – 5

[74] BIRK, M. ; KRETZEK, E. ; FIGULI, P. ; WEBER, M. ; BECKER, J. ; RUITER, N.: High-Speed Medical Imaging in 3D Ultrasound Computer Tomography. In: *Parallel and Distributed Systems, IEEE Transactions on* PP (2015), Nr. 99, S. 1–1. – ISSN 1045-9219

[75] AL-ASHMOUNY, K.M. ; HAMED, H.M. ; MORSY, A.A.: FPGA-based Sleep Apnea Screening Device for Home Monitoring. In: *Engineering in Medicine and Biology Society, 2006. EMBS '06. 28th Annual International Conference of the IEEE*, August 2006, S. 5948 – 5951. – ISSN 1557-170X

[76] WENG, Chi-Kai ; CHEN, Jeng-Wen ; HUANG, Chih-Chung: A FPGA-based Wearable Ultrasound Device for Monitoring Obstructive Sleep Apnea Syndrome. In: *2015 IEEE International Ultrasonics Symposium Proceedings*, 2015

[77] CARVALHO, H.H. ; CINTRA, E.R.F. ; PIMENTA, T.C. ; MORENO, R.L.: An electrocardiogram diagnostic system implemented in FPGA. In: *Biosignals and Biorobotics Conference (BRC), 2012 ISSNIP*, Januar 2012, S. 1 – 5

[78] CORDES, B. ; DY, J. ; LEESER, M. ; GOEBEL, J.: Enabling a real-time solution for neuron detection with reconfigurable hardware. In: *Rapid System Prototyping, 2005. (RSP 2005). The 16th IEEE International Workshop on*, Juni 2005, S. 128 – 134. – ISSN 1074-6005

[79] RUDRA, Angsuman: FPGA-based applications for software radio. In: *RF Design Magazine* (2004), S. 24 – 35

[80] XILINX: *Virtex-II Platform FPGAs*. DS031 (v3.5), November 2007

[81] XILINX: *XtremeDSP for Virtex-4 FPGAs User Guide*. UG073 (v2.7), Mai 2008

[82] CHEN, Hao ; CHEN, Yu ; SUMMERVILLE, D.H.: A Survey on the Application of FPGAs for Network Infrastructure Security. In: *Communications Surveys Tutorials, IEEE* 13 (2011), August, Nr. 4, S. 541 – 561. – ISSN 1553-877X

[83] LOCKWOOD, J.W. ; MCKEOWN, N. ; WATSON, G. ; GIBB, G. ; HARTKE, P. ; NAOUS, J. ; RAGHURAMAN, R. ; LUO, Jianying: NetFPGA – An Open Platform for Gigabit-Rate Network Switching and Routing. In: *Microelectronic Systems Education, 2007. MSE '07. IEEE International Conference on*, june 2007, S. 160 – 161

[84] BYMA, S. ; STEFFAN, J.G. ; CHOW, P.: NetThreads-10G: Software packet processing on NetFPGA-10G in a virtualized networking environment demonstration abstract. In: *Field Programmable Logic and Applications (FPL), 2013 23rd International Conference on*, September 2013, S. 1–1

[85] XILINX: *XILINX DELIVERS VIRTEX-II PRO X FPGAS - WORLD'S MOST CAPABLE HIGH-SPEED SERIAL TRANSCEIVER SOLUTION Xilinx Press Release # 03154*. online. November 2003. – URL `http://www.xilinx.com/prs_rls/silicon_vir/03154v2prox_ships.htm`

[86] CALABRETTA, N. ; DI LUCENTE, S. ; LUO, J. ; ROHIT, A. ; WILLIAMS, K. ; DORREN, H.: Flow controlled scalable optical packet switch for low latency flat data center network. In: *Transparent Optical Networks (ICTON), 2013 15th International Conference on*, Juni 2013, S. 1 – 4. – ISSN 2161-2056

[87] ENOSH, S.M. ; GEORGE, N.S.: An efficient implementation of protocol operation control unit of FlexRay communication controller. In: *Computational Systems and Communications (ICCSC), 2014 First International Conference on*, Dezember 2014, S. 256 – 259

[88] ZHU, Jihan ; SUTTON, Peter: FPGA Implementations of Neural Networks – A Survey of a Decade of Progress. In: CHEUNG, Peter (Hrsg.) ; CONSTANTINIDES, GeorgeA. (Hrsg.): *Field Programmable Logic and Application* Bd. 2778. Springer Berlin Heidelberg, 2003, S. 1062 – 1066. – ISBN 978-3-540-40822-2

[89] JAMES-ROXBY, P. ; BLODGET, B.: Adapting constant multipliers in a neural network implementation. In: *Field-Programmable Custom Computing Machines, 2000 IEEE Symposium on*, April 2000, S. 335 – 336

[90] POHL, Christopher: *Konfigurierbare Hardwarebeschleuniger für selbst-organisierende Karten*, Universität Paderborn, Dissertation, 2010

[91] SANNI, K. ; GARREAU, G. ; MOLIN, J.L. ; ANDREOU, A.G.: FPGA Implementation of a Deep Belief Network Architecture for Character Recognition Using Stochastic Computation. In: *Information Sciences and Systems (CISS), 2015 49th Annual Conference on*, März 2015, S. 1 – 5

[92] KIM, Lok-Won ; ASAAD, Sameh ; LINSKER, Ralph: A Fully Pipelined FPGA Architecture of a Factored Restricted Boltzmann Machine Artificial Neural Network. In: *ACM Trans. Reconfigurable Technol. Syst.* 7 (2014), Februar, Nr. 1, S. 5:1 – 5:23. – ISSN 1936-7406

[93] POHL, C. ; FRANZMEIER, M. ; PORRMANN, M. ; RÜCKERT, U.: gNBX - Reconfigurable Hardware Acceleration of Self-organizing Maps. In: *Field-Programmable Technology, 2004. Proceedings. 2004 IEEE International Conference on*, Dezember 2004, S. 97 – 104

[94] QIU, Jiantao ; WANG, Jie ; YAO, Song ; GUO, Kaiyuan ; LI, Boxun ; ZHOU, Erjin ; YU, Jincheng ; TANG, Tianqi ; XU, Ningyi ; SONG, Sen u. a.: Going Deeper with Embedded FPGA Platform for Convolutional Neural Network. In: *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays* ACM (Veranst.), 2016, S. 26 – 35

[95] OVTCHAROV, Kalin ; RUWASE, Olatunji ; KIM, Joo-Young ; FOWERS, Jeremy ; STRAUSS, Karin ; CHUNG, Eric S.: Accelerating Deep Convolutional Neural Networks Using Specialized Hardware / Microsoft Research. 2015. – Forschungsbericht

[96] CHEN, X. W. ; LIN, X.: Big Data Deep Learning: Challenges and Perspectives. In: *IEEE Access* 2 (2014), S. 514 – 525. – ISSN 2169-3536

[97] MISHRA, D. ; HATTO, M. ; KUHARA, T. ; OSANA, Y. ; FUJITA, N. ; AMANO, H.: FPGA Implementation of Viscous Function in a Package for Computational Fluid Dynamics. In: *Computing and Networking (CANDAR), 2014 Second International Symposium on*, Dezember 2014, S. 608 – 610

[98] LIU, Ming ; KUEHN, W. ; LU, Zhonghai ; JANTSCH, A.: System-on-an-FPGA Design for Real-time Particle Track Recognition and Reconstruction in Physics Experiments. In: *Digital System Design Architectures, Methods and Tools, 2008. DSD '08. 11th EUROMICRO Conference on*, September 2008, S. 599 – 605

[99] FROHLICH, I. ; GABRIEL, A. ; KIRSCHNER, D. ; LEHNERT, J. ; LINS, E. ; PETRI, M. ; PEREZ-CAVALCANTI, T. ; RITMAN, J. ; SCHAFER, D. ; TOIA, A. ; TRAXLER, M. ; KUEHN, W.: Pattern recognition in the HADES spectrometer: an application of FPGA technology in nuclear and particle physics. In: *Field-Programmable Technology, 2002. (FPT). Proceedings. 2002 IEEE International Conference on*, Dezember 2002, S. 443 – 444

[100] ABBA, S. ; LEE, Jeong-A: Examining the Performance Impact of NoC Parameters for Scalable and Adaptive FPGA-Based Network-on-Chips. In: *Computational Intelligence, Modelling and Simulation (CIMSim), 2013 Fifth International Conference on*, September 2013, S. 364 – 372

[101] LIMA, E. R. de ; QUEIROZ, A. F. R. ; ALVES, D. C. ; SILVA, G. S. da ; CHAVES, C. G. ; MERTES, J. G. ; MARSON, T. M.: A detailed DVB-S2 receiver implementation: FPGA prototyping and preliminary ASIC resource estimation. In:

2014 IEEE Latin-America Conference on Communications (LATINCOM), Nov 2014, S. 1 – 6. – ISSN 2330-989X

[102] XILINX: *Kintex UltraScale Architecture Data Sheet: DC and AC Switching Characteristics*. DS892 (v1.2), Mai 2014

[103] INTERNATIONAL TECHNOLOGY ROADMAP FOR SEMICONDUCTORS (ITRS): ITRS International Roadmap Committee Overview / International Technology Roadmap for Semiconductors (ITRS). 2013. – Forschungsbericht

[104] JUNGEBLUT, Thorsten: *Entwurfsraumexploration ressourceneffizienter VLIW-Prozessoren*, Technische Fakultät, Universität Bielefeld, Dissertation, Mai 2011

[105] XILINX: *Virtex-II Pro and Virtex-II Pro X Platform FPGAs*. DS083 (v5.0), Juni 2011

[106] KUMAR M, V. ; SELVAKUMAR A, D. ; SOBHA, P.M.: Area and Frequency optimized 1024 point Radix-2 FFT Processor on FPGA. In: *VLSI Systems, Architecture, Technology and Applications (VLSI-SATA), 2015 International Conference on*, Januar 2015, S. 1 – 6

[107] BAI, Yonghong ; WU, Liji ; WANG, Beibei ; ZHANG, Xiangmin: A Novel Java Coprocessor with Data Hazard Handling on FPGA for IC Bank Card. In: *Solid-State and Integrated Circuit Technology (ICSICT), 2014 12th IEEE International Conference on*, Oktober 2014, S. 1 – 3

[108] CALLANAN, O. ; GREGG, D. ; NISBET, A. ; PEARDON, M.: High Performance Scientific Computing Using FPGAs with IEEE Floating Point and Logarithmic Arithmetic for Lattice QCD. In: *Field Programmable Logic and Applications, 2006. FPL '06. International Conference on*, August 2006, S. 1 – 6

[109] MONMASSON, E. ; CIRSTEA, M.N.: FPGA Design Methodology for Industrial Control Systems – A Review. In: *Industrial Electronics, IEEE Transactions on* 54 (2007), August, Nr. 4, S. 1824 – 1842. – ISSN 0278-0046

[110] BUCCELLA, C. ; CECATI, C. ; LATAFAT, H.: Digital Control of Power Converters – A Survey. In: *Industrial Informatics, IEEE Transactions on* 8 (2012), August, Nr. 3, S. 437 – 447. – ISSN 1551-3203

[111] SANCHEZ, P.M. ; MACHADO, O. ; PEÑA, E.J.B. ; RODRIGUEZ, F.J. ; MECA, F.J.: FPGA-Based Implementation of a Predictive Current Controller for Power Converters. In: *Industrial Informatics, IEEE Transactions on* 9 (2013), August, Nr. 3, S. 1312 – 1321. – ISSN 1551-3203

[112] CHEN, Shin-Ju ; YANG, Sung-Pei ; WONG, Ruei-Hong: FPGA-Based Digital Control for Boost Converters with Power Factor Correction. In: *Industrial Electronics and Applications (ICIEA), 2012 7th IEEE Conference on*, Juli 2012, S. 1102 – 1106

[113] DUBEY, R. ; AGARWAL, P. ; VASANTHA, M. K.: Programmable Logic Devices for Motion Control – A Review. In: *Industrial Electronics, IEEE Transactions on* 54 (2007), Februar, Nr. 1, S. 559 – 566. – ISSN 0278-0046

[114] PAIZ, C. ; HAGEMEYER, J. ; POHL, C. ; PORRMANN, M. ; RÜCKERT, U. ; SCHULZ, B. ; PETERS, W. ; BOCKER, J.: FPGA-Based Realization of Self-Optimizing Drive-Controllers. In: *Industrial Electronics, 2009. IECON '09. 35th Annual Conference of IEEE*, November 2009, S. 2848 – 2853. – ISSN 1553-572X

[115] PAIZ GATICA, Carlos Vladimir: *Dynamically Reconfigurable Hardware for Embedded Control Systems*, Universität Bielefeld, Dissertation, Dezember 2012

[116] BELHADJ, Hichem ; AGGRAWAL, Vishal ; PRADHAN, Ajay ; ZERROUKI, Amal: Power-aware FPGA design. In: *Actel Corporation White Paper* 75 (2009)

[117] SUNDARARAJAN, Priya ; GAYASEN, Aman ; VIJAYKRISHNAN, N. ; TUAN, T.: Thermal Characterization and Optimization in Platform FPGAs. In: *Proceedings of the 2006 IEEE/ACM International Conference on Computer-aided Design*. New York, NY, USA : ACM, 2006 (ICCAD '06), S. 443 – 447. – ISBN 1-59593-389-1

[118] REYNOLDS, R. O. ; SMITH, P. H. ; BELL, L. S. ; KELLER, H. U.: The design of Mars lander cameras for Mars Pathfinder, Mars Surveyor '98 and Mars Surveyor '01. In: *IEEE Transactions on Instrumentation and Measurement* 50 (2001), Februar, Nr. 1, S. 63 – 71. – ISSN 0018-9456

[119] BHOJ, S. ; BHATIA, D.: Thermal Modeling and Temperature Driven Placement for FPGAs. In: *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*, Mai 2007, S. 1053 – 1056

[120] CHO, Jung U. ; LE, Quy N. ; JEON, Jae W.: An FPGA-Based Multiple-Axis Motion Control Chip. In: *Industrial Electronics, IEEE Transactions on* 56 (2009), March, Nr. 3, S. 856 – 870. – ISSN 0278-0046

[121] PETKO, M. ; GAC, K. ; GORA, G. ; KARPIEL, G. ; OCHONSKI, J.: Acceleration of Parallel Robot Trajectory Generation in FPGA. In: *Industrial Electronics and Applications (ICIEA), 2013 8th IEEE Conference on*, June 2013, S. 1123 – 1128

[122] NAZIR, L. ; MIR, R.N.: Performance Analysis of Various scheduling algorithms using FPGA Platforms. In: *VLSI Systems, Architecture, Technology and Applications (VLSI-SATA), 2015 International Conference on*, Januar 2015, S. 1 – 4

[123] TANG, Yi ; BERGMANN, N.W.: A Hardware Scheduler Based on Task Queues for FPGA-Based Embedded Real-Time Systems. In: *Computers, IEEE Transactions on* 64 (2015), Mai, Nr. 5, S. 1254 – 1267. – ISSN 0018-9340

[124] MAFI, R. ; SIROUSPOUR, S. ; MAHDAVIKHAH, B. ; MOODY, B. ; ELIZEH, K. ; KINSMAN, A.B. ; NICOLICI, N.: A Parallel Computing Platform for Real-Time Haptic Interaction with Deformable Bodies. In: *Haptics, IEEE Transactions on* 3 (2010), Juli, Nr. 3, S. 211 – 223. – ISSN 1939-1412

[125] SHEN, Hao ; QIU, Qinru: An FPGA-Based Distributed Computing System with Power and Thermal Management Capabilities. In: *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*, Juli 2011, S. 1 – 6. – ISSN 1095-2055

[126] CHEN, Poki ; SHIE, Mon-Chau ; ZHENG, Zhi-Yuan ; ZHENG, Zi-Fan ; CHU, Chun-Yan: A Fully Digital Time-Domain Smart Temperature Sensor Realized With 140 FPGA Logic Elements. In: *Circuits and Systems I: Regular Papers, IEEE Transactions on* 54 (2007), Dezember, Nr. 12, S. 2661 – 2668. – ISSN 1549-8328

[127] BHATTACHARJEE, A. ; CONTRERAS, G. ; MARTONOSI, M.: Full-System Chip Multiprocessor Power Evaluations Using FPGA-Based Emulation. In: *Low Power Electronics and Design (ISLPED), 2008 ACM/IEEE International Symposium on*, August 2008, S. 335 – 340

[128] WÄLTERMANN, Peter: „Hardware-in-the-Loop": Die Technologie zum Test elektronischer Steuerungen und Regelungen in der Fahrzeugtechnik. In: *6. Paderborner Workshop: Entwurf mechatronischer Systeme*. dSPACE GmbH, April 2009

[129] FISCHER, Till: Entwurf eines FPGA-Cores zur Simulationsbeschleunigung zeitkontinuierlicher Modelle im HiL-Kontext. In: HALANG, Wolfgang A. (Hrsg.): *Herausforderungen durch Echtzeitbetrieb*. Springer Berlin Heidelberg, 2012 (Informatik aktuell), S. 75 – 80. – ISBN 978-3-642-24657-9

[130] XILINX: *Virtex-5QV Family Overview*. DS192 (v1.4), November 2014

[131] RUAN, Zhuo ; HAN, Yuzhang ; CAI, Hongbo ; JIN, Shengzhen ; HAN, Yuzhang: A Dynamically Partial-reconfigurable FPGA-based Architecture for Data Processing on Space Solar Telescope. In: *Industrial Embedded Systems, 2007. SIES '07. International Symposium on*, Juli 2007, S. 194 – 199

[132] HAGEMEYER, J. ; HILGENSTEIN, A. ; JUNGEWELTER, D. ; COZZI, D. ; FELICETTI, C. ; RUECKERT, U. ; KORF, S. ; KOESTER, M. ; MARGAGLIA, F. ; PORRMANN, M. ; DITTMANN, F. ; DITZE, M. ; HARRIS, J. ; STERPONE, L. ; ILSTAD, J.: A Scalable Platform For Run-time Reconfigurable Satellite Payload Processing. In: *Adaptive Hardware and Systems (AHS), 2012 NASA/ESA Conference on*, Juni 2012, S. 9 – 16

[133] SHARMA, S. ; KULKARN, S. ; PUJARI, V. ; VANITHA, M. ; LAKSHMINARSIMHAN, P.: FPGA Implementation of M-PSK Modulators for Satellite Communication. In: *Advances in Recent Technologies in Communication and Computing (ARTCom), 2010 International Conference on*, Oktober 2010, S. 136 – 139

[134] KOIZUMI, K. ; INABA, M. ; HIRAKI, K. ; ISHII, Y. ; MIYOSHI, T. ; YOSHIZOE, K.: Triple Line-Based Playout for Go - An Accelerator for Monte Carlo Go. In: *Reconfigurable Computing and FPGAs, 2009. ReConFig '09. International Conference on*, Dezember 2009, S. 161 – 166

[135] SKLIAROVA, Iouliia ; VALLEJO, Tiago ; SKLYAROV, Valery ; SUDNITSON, Alexander ; KRUUS, Margus: Solving Computationally Intensive Problems in Reconfigurable Hardware: A Case Study. In: *Journal of Convergence Information Technology* 8 (2013), Nr. 3

[136] STIAKOGIANNAKIS, Ioannis N. ; KAKLAMANI, Dimitra I.: A Radio Resource Management Framework for Multi-User Multi-Cell OFDMA Networks Based on Game Theory. In: *Wireless Personal Communications* 69 (2013), Nr. 2, S. 745 – 770. – ISSN 1572-834X

[137] HUANG, J.R. ; ONG, C.K. ; CHENG, K.-T. ; WU, C.W.: An FPGA-based Reconfigurable Functional Tester for Memory Chips. In: *Test Symposium, 2000. (ATS 2000). Proceedings of the Ninth Asian*, 2000, S. 51 – 57. – ISSN 1081-7735

[138] OH, Kyung-Soo ; YOON, Sang-Yong ; CHAE, Soo-Ik: Emulator environment based on an FPGA prototyping board. In: *Rapid System Prototyping, 2000. RSP 2000. Proceedings. 11th International Workshop on*, 2000, S. 72 – 77. – ISSN 1074-6005

[139] ELDREDGE, J.G. ; HUTCHINGS, B.L.: Density Enhancement of a Neural Network Using FPGAs and Run-Time Reconfiguration. In: *FPGAs for Custom Computing Machines, 1994. Proceedings. IEEE Workshop on*, April 1994, S. 180 – 188

[140] GUCCIONE, StevenA. ; GONZALEZ, MarioJ.: Classification and Performance of Reconfigurable Architectures. In: MOORE, Will (Hrsg.) ; LUK, Wayne

(Hrsg.): *Field-Programmable Logic and Applications* Bd. 975. Springer Berlin Heidelberg, 1995, S. 439 – 448. – ISBN 978-3-540-60294-1

[141] PAPADIMITRIOU, Kyprianos ; DOLLAS, Apostolos ; HAUCK, Scott: Performance of Partial Reconfiguration in FPGA Systems: A Survey and a Cost Model. In: *ACM Trans. Reconfigurable Technol. Syst.* 4 (2011), Dezember, Nr. 4, S. 36:1 – 36:24. – ISSN 1936-7406

[142] PAPADIMITRIOU, Kyprianos D.: *Partial Reconfiguration Cost Calculator (PRCC)*. Web application. Juni 2015. – URL `http://users.isc.tuc.gr/~kpapadimitriou/prcc.html`

[143] LIANG, Chen ; WU, Chenlu ; ZHOU, Xuegong ; CAO, Wei ; WANG, Shengye ; WANG, Lingli: An FPGA-cluster-accelerated Match Engine for Content-based Image Retrieval. In: *Field-Programmable Technology (FPT), 2013 International Conference on*, Dezember 2013, S. 422 – 425

[144] GUYETANT, Stéphane ; GIRAUD, Mathieu ; L'HOURS, Ludovic ; DERRIEN, Steven ; RUBINI, Stéphane ; LAVENIER, Dominique ; RAIMBAULT, Frédéric: Cluster of Re-configurable Nodes for Scanning Large Genomic Banks. In: *Parallel Comput.* 31 (2005), Januar, Nr. 1, S. 73 – 96. – ISSN 0167-8191

[145] PATTERSON, C. ; MARTIN, B. ; ELLINGSON, S. ; SIMONETTI, J. ; CUTCHIN, S.: FPGA Cluster Computing in the ETA Radio Telescope. In: *Field-Programmable Technology, 2007. ICFPT 2007. International Conference on*, Dec 2007, S. 25 – 32

[146] LIN, Edward C. ; RUTENBAR, Rob A.: A Multi-FPGA 10x-Real-Time High-Speed Search Engine for a 5000-Word Vocabulary Speech Recognizer. In: *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*. New York, NY, USA : ACM, 2009 (FPGA '09), S. 83 – 92. – ISBN 978-1-60558-410-2

[147] CHRYSOS, Grigorios ; DAGRITZIKOS, Panagiotis ; PAPAEFSTATHIOU, Ioannis ; DOLLAS, Apostolos: HC-CART: A Parallel System Implementation of Data Mining Classification and Regression Tree (CART) Algorithm on a multi-FPGA System. In: *ACM Trans. Archit. Code Optim.* 9 (2013), Januar, Nr. 4, S. 47:1 – 47:25. – ISSN 1544-3566

[148] TIAN, Xiang ; BENKRID, K.: Massively parallelized Quasi-Monte Carlo financial simulation on a FPGA supercomputer. In: *High-Performance Reconfigurable Computing Technology and Applications, 2008. HPRCTA 2008. Second International Workshop on*, November 2008, S. 1 – 8

[149] CASTILLO, J. ; BOSQUE, J.L. ; CASTILLO, E. ; HUERTA, P. ; MARTINEZ, J.I.: Hardware accelerated montecarlo financial simulation over low cost FPGA cluster. In: *Parallel Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, Mai 2009, S. 1 – 8. – ISSN 1530-2075

[150] CONG, J. ; HUANG, Muhuan ; ZOU, Yi: Accelerating Fluid Registration Algorithm on Multi-FPGA Platforms. In: *Field Programmable Logic and Applications (FPL), 2011 International Conference on*, September 2011, S. 50 – 57

[151] GOTHANDARAMAN, Akila: *Accelerating Quantum Monte Carlo Simulations with Emerging Architectures*, University of Tennessee, Dissertation, August 2009

[152] SASS, R. ; KRITIKOS, W.V. ; SCHMIDT, A.G. ; BEERAVOLU, S. ; BEERAKA, P.: Reconfigurable Computing Cluster (RCC) Project: Investigating the Feasibility of FPGA-Based Petascale Computing. In: *Field-Programmable Custom Computing Machines, 2007. FCCM 2007. 15th Annual IEEE Symposium on*, April 2007, S. 127 – 140

[153] AGAKICHIEV, G. et al.: The high-acceptance dielectron spectrometer HADES. In: *The European Physical Journal A* 41 (2009), Nr. 2, S. 243 – 277. – ISSN 1434-6001

[154] LIU, Ming ; LANG, J. ; YANG, Shuo ; PEREZ, T. ; KUEHN, W. ; XU, Hao ; JIN, Dapeng ; WANG, Qiang ; LI, Lu ; LIU, Zhenan ; LU, Zhonghai ; JANTSCH, A.: ATCA-based computation platform for data acquisition and triggering in particle physics experiments. In: *Field Programmable Logic and Applications, 2008. FPL 2008. International Conference on*, September 2008, S. 287 – 292

[155] MAHDAVIKHAH, Behzad ; MAFI, Ramin ; SIROUSPOUR, Shahin ; NICOLICI, Nicola: A Multiple-FPGA Parallel Computing Architecture for Real-time Simulation of Soft-object Deformation. In: *ACM Trans. Embed. Comput. Syst.* 13 (2014), März, Nr. 4, S. 81:1 – 81:23. – ISSN 1539-9087

[156] CHEN, Yuan ; DINAVAHI, V.: Multi-FPGA digital hardware design for detailed large-scale real-time electromagnetic transient simulation of power systems. In: *Generation, Transmission Distribution, IET* 7 (2013), Mai, Nr. 5, S. 451 – 463. – ISSN 1751-8687

[157] KWON, Young-Su ; KYUNG, Chong-Min: Performance-Driven Event-Based Synchronization for Multi-FPGA Simulation Accelerator With Event Time-Multiplexing Bus. In: *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 24 (2005), September, Nr. 9, S. 1444 – 1456. – ISSN 0278-0070

[158] SALDANA, M. ; CHOW, P.: TMD-MPI: An MPI Implementation for Multiple Processors Across Multiple FPGAs. In: *Field Programmable Logic and Applications, 2006. FPL '06. International Conference on*, August 2006, S. 1 – 6

[159] ASAAD, Sameh ; BELLOFATTO, Ralph ; BREZZO, Bernard ; HAYMES, Chuck ; KAPUR, Mohit ; PARKER, Benjamin ; ROEWER, Thomas ; SAHA, Proshanta ; TAKKEN, Todd ; TIERNO, José: A Cycle-accurate, Cycle-reproducible multi-FPGA System for Accelerating Multi-core Processor Simulation. In: *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*. New York, NY, USA : ACM, 2012 (FPGA '12), S. 153 – 162. – ISBN 978-1-4503-1155-7

[160] CHEN, G. ; WANG, X. ; LI, X.: *Fundamentals of Complex Networks: Models, Structures and Dynamics*. Wiley, 2014. – ISBN 9781118718148

[161] GUIMERÀ, R. ; DÍAZ-GUILERA, A. ; VEGA-REDONDO, F. ; CABRALES, A. ; ARENAS, A.: Optimal Network Topologies for Local Search with Congestion. In: *Phys. Rev. Lett.* 89 (2002), November, S. 248701

[162] STRUGHOLZ, Manuel: *Analyse der Ressourceneffizienz leitungsgebundener Kommunikation in Multiprozessorsystemen*, Universität Paderborn, Dissertation, 2013

[163] DIESTEL, Reinhard: *Graphentheorie*. 4., neu bearb. und erw. Aufl. Berlin [u.a.] : Springer, 2012. – ISBN 3-540-21391-0, 9783540213918, 3540213910, 978-3-540-21391-8

[164] BÜSING, Christina ; SPEKTRUM, Akad. V. (Hrsg.): *Graphen- und Netzwerkoptimierung*. Spektrum, Akad. Verl., 2010. – 265 S

[165] XILINX: *Virtex-4 RocketIO Multi-Gigabit Transceiver User Guide*. UG076 (v4.1), November 2008

[166] XILINX: *Virtex-5 FPGA RocketIO GTX Transceiver User Guide*. UG198 (v3.0), Oktober 2009

[167] MERINGER, Markus: Fast Generation of Regular Graphs and Construction of Cages. In: *J. Graph Theory* 30 (1999), Februar, Nr. 2, S. 137 – 146. – ISSN 0364-9024

[168] MERINGER, Markus: *Connected regular graphs*. 2016. – URL `http://www.mathe2.uni-bayreuth.de/markus/reggraphs.html`. – Zugriffsdatum: 2016

[169] GÜTING, R.H. ; DIEKER, S.: *Datenstrukturen und Algorithmen*. Vieweg+Teubner Verlag, 2004 (Lehrbuch : Informatik). – ISBN 9783519221210

[170] PARVEZ, H. ; MARRAKCHI, Z. ; MEHREZ, H.: ASIF: Application Specific Inflexible FPGA. In: *2009 International Conference on Field-Programmable Technology*, Dec 2009, S. 112 – 119

[171] AWAD, Mariette: FPGA supercomputing platforms: A survey. In: *Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on*, August 2009, S. 564 – 568. – ISSN 1946-1488

[172] SHAN, Amar: *The Cray XD1 Technical Overview*. Mai 2004

[173] EL-GHAZAWI, T. ; EL-ARABY, E. ; HUANG, Miaoqing ; GAJ, K. ; KINDRATENKO, V. ; BUELL, D.: The Promise of High-Performance Reconfigurable Computing. In: *Computer* 41 (2008), Februar, Nr. 2, S. 69 – 76. – ISSN 0018-9162

[174] DINI GROUP: *DN9000K10 User Manual*. 1. 7469 Draper Ave. La Jolla, CA92037 USA: , April 2009

[175] ARNOLD, Jeffrey M. ; BUELL, Duncan A. ; DAVIS, Elaine G.: Splash 2. In: *Proceedings of the Fourth Annual ACM Symposium on Parallel Algorithms and Architectures*. New York, NY, USA : ACM, 1992 (SPAA '92), S. 316 – 322. – ISBN 0-89791-483-X

[176] CASSELMAN, S.: Virtual computing and the Virtual Computer. In: *FPGAs for Custom Computing Machines, 1993. Proceedings. IEEE Workshop on*, Apr 1993, S. 43 – 48

[177] RENCHER, M. ; HUTCHINGS, B.L.: Automated target recognition on SPLASH 2. In: *Field-Programmable Custom Computing Machines, 1997. Proceedings., The 5th Annual IEEE Symposium on*, April 1997, S. 192 – 200

[178] KUNG, H. T.: Why systolic architectures? In: *Computer* 15 (1982), Januar, Nr. 1, S. 37 – 46. – ISSN 0018-9162

[179] ARNOLD, J.M.: The Splash 2 software environment. In: *FPGAs for Custom Computing Machines, 1993. Proceedings. IEEE Workshop on*, April 1993, S. 88 – 93

[180] ABBOTT, A.L. ; ATHANAS, P.M. ; CHEN, L. ; ELLIOTT, R.L.: Finding lines and building pyramids with SPLASH 2. In: *FPGAs for Custom Computing Machines, 1994. Proceedings. IEEE Workshop on*, April 1994, S. 155 – 163

[181] ARNOLD, J.M. ; BUELL, D.A. ; HOANG, D.T. ; PRYOR, Daniel V. ; SHIRAZI, N. ; THISTLE, M.R.: The Splash 2 processor and applications. In: *Computer Design: VLSI in Computers and Processors, 1993. ICCD '93. Proceedings., 1993 IEEE International Conference on*, Oktober 1993, S. 482 – 485

[182] PRYOR, Daniel V. ; THISTLE, M.R. ; SHIRAZI, N.: Text searching on Splash 2. In: *FPGAs for Custom Computing Machines, 1993. Proceedings. IEEE Workshop on*, April 1993, S. 172 – 177

[183] HOANG, D.T.: Searching genetic databases on Splash 2. In: *FPGAs for Custom Computing Machines, 1993. Proceedings. IEEE Workshop on*, April 1993, S. 185 – 191

[184] ÅKERSTEDT, H. ; MUSCHTER, S. ; DRAKE, G. ; ANDERSON, K. ; BOHM, C. ; OREGLIA, M. ; TANG, F.: Reliable and Redundant FPGA Based Read-Out Design in the ATLAS TileCal Demonstrator. In: *IEEE Transactions on Nuclear Science* 62 (2015), Oktober, Nr. 5, S. 2129 – 2133. – ISSN 0018-9499

[185] KUMAR, Sandeep ; PAAR, Christof ; PELZL, Jan: How to Break DES for BC 8,980. In: *IN INTERNATIONAL WORKSHOP ON SPECIAL-PURPOSE HARDWARE FOR ATTACKING CRYPTOGRAPHIC SYSTEMS – SHARCS06*, 2006, S. 3 – 4

[186] GÜNEYSU, Tim ; KASPER, Timo ; NOVOTNÝ, Martin ; PAAR, Christof ; WIENBRANDT, Lars ; ZIMMERMANN, Ralf: High-Performance Cryptanalysis on RIVYERA and COPACOBANA Computing Systems. In: VANDERBAUWHEDE, Wim (Hrsg.) ; BENKRID, Khaled (Hrsg.): *High-Performance Computing Using FPGAs*. Springer New York, 2013, S. 335 – 366. – ISBN 978-1-4614-1790-3

[187] POSPÍŠIL, Jan ; NOVOTNÝ, Martin: Lightweight cipher resistivity against brute-force attack: Analysis of PRESENT. In: *Design and Diagnostics of Electronic Circuits Systems (DDECS), 2012 IEEE 15th International Symposium on*, April 2012, S. 197 – 198

[188] POSPÍŠIL, Jan ; NOVOTNÝ, Martin: Evaluating Cryptanalytical Strength of Lightweight Cipher PRESENT on Reconfigurable Hardware. In: *Digital System Design (DSD), 2012 15th Euromicro Conference on*, September 2012, S. 560 – 567

[189] STEMBERA, P. ; NOVOTNY, M.: Breaking Hitag2 with Reconfigurable Hardware. In: *Digital System Design (DSD), 2011 14th Euromicro Conference on*, August - September 2011, S. 558 – 563

[190] ZIMMERMANN, R. ; GUNEYSU, T. ; PAAR, C.: High-Performance Integer Factoring with Reconfigurable Devices. In: *Field Programmable Logic and Applications (FPL), 2010 International Conference on*, August - September 2010, S. 83 – 88. – ISSN 1946-1488

[191] SCIENGINES GMBH: *COPACOBANA V4-SX35 DSP-optimized reconfigurable architecture Virtex based variant of the original COPACOBANA*. Website. Juli 2013. – URL http://sciengines.com/products/computers-and-clusters/copacobana-v4-sx35.html

[192] SCIENGINES GMBH: *RIVYERA S3-5000 FPGA-computing for restricted budgets Spartan-3 5000 based massively parallel system*. Website. September 2013. – URL http://sciengines.com/products/computers-and-clusters/rivyera-s3-5000.html

[193] SCIENGINES GMBH: *RIVYERA S6-LX150 128 FPGA Cluster The next generation of performance-optimized reconfigurable computing*. Website. September 2013. – URL http://sciengines.com/products/computers-and-clusters/rivyera-s6-lx150.html

[194] SCIENGINES GMBH: *RIVYERA V6-LX550T Virtex-6 based FPGA-computer Highest density of processing resources*. Website. September 2013. – URL http://sciengines.com/products/computers-and-clusters/rivyera-v6-lx550t.html

[195] SCIENGINES GMBH: *RIVYERA V7-2000T The largest Xilinx FPGAs*. Website. September 2013. – URL http://sciengines.com/products/computers-and-clusters/v72000t.html

[196] WIENBRANDT, Lars ; SIEBERT, Daniel ; SCHIMMLER, Manfred: Improvement of BLASTp on the FPGA-Based High-Performance Computer RIVYERA. In: BLERIS, Leonidas (Hrsg.) ; MANDOIU, Ion (Hrsg.) ; SCHWARTZ, Russell (Hrsg.) ; WANG, Jianxin (Hrsg.): *Bioinformatics Research and Applications* Bd. 7292. Springer Berlin Heidelberg, Mai 2012, S. 275 – 286. – ISBN 978-3-642-30190-2

[197] BOGDANOV, A. ; KAVUN, E.B. ; TISCHHAUSER, E. ; YALCIN, T.: Efficient reconfigurable hardware architecture for accurately computing success probability and data complexity of linear attacks. In: *Reconfigurable Computing and FPGAs (ReConFig), 2012 International Conference on*, Dezember 2012, S. 1 – 6

[198] WIENBRANDT, Lars: Bioinformatics Applications on the FPGA-Based High-Performance Computer RIVYERA. In: VANDERBAUWHEDE, Wim (Hrsg.) ; BENKRID, Khaled (Hrsg.): *High-Performance Computing Using FPGAs*. Springer New York, 2013, S. 81 – 103. – ISBN 978-1-4614-1790-3

[199] SYNOPSYS INC.: *HAPS-64 Virtex-6 Motherboard*. 700 East Middlefield Road Mountain View, CA 94043: , 2012

[200] YOKOTA, T. ; NAGAFUCHI, M. ; MEKADA, Y. ; YOSHINAGA, T. ; OOTSU, K. ; BABA, T.: A Scalable FPGA-based Custom Computing Machine for a Medical Image Processing. In: *Field-Programmable Custom Computing Machines, 2002. Proceedings. 10th Annual IEEE Symposium on*, 2002, S. 307 – 308

[201] CAULFIELD, Adrian ; CHUNG, Eric ; PUTNAM, Andrew ; ANGEPAT, Hari ; FOWERS, Jeremy ; HASELMAN, Michael ; HEIL, Stephen ; HUMPHREY, Matt ; KAUR, Puneet ; KIM, Joo-Young ; LO, Daniel ; MASSENGILL, Todd ; OVTCHAROV, Kalin ; PAPAMICHAEL, Michael ; WOODS, Lisa ; LANKA, Sitaram ; CHIOU, Derek ; BURGER, Doug: A Cloud-Scale Acceleration Architecture. In: *Proceedings of the 49th Annual IEEE/ACM International Symposium on Microarchitecture*, IEEE Computer Society, October 2016

[202] PUTNAM, A. ; CAULFIELD, A. M. ; CHUNG, E. S. ; CHIOU, D. ; CONSTANTINIDES, K. ; DEMME, J. ; ESMAEILZADEH, H. ; FOWERS, J. ; GOPAL, G. P. ; GRAY, J. ; HASELMAN, M. ; HAUCK, S. ; HEIL, S. ; HORMATI, A. ; KIM, J. Y. ; LANKA, S. ; LARUS, J. ; PETERSON, E. ; POPE, S. ; SMITH, A. ; THONG, J. ; XIAO, P. Y. ; BURGER, D.: A Reconfigurable Fabric for Accelerating Large-Scale Datacenter Services. In: *IEEE Micro* 35 (2015), Mai, Nr. 3, S. 10 – 22. – ISSN 0272-1732

[203] CASTILLO, E. ; PEDRAZA, C. ; CASTILLO, J. ; CAMARERO, C. ; BOSQUE, J.L. ; MENENDEZ, R. ; MARTINEZ, J.I.: SMILE: Scientific Parallel Multiprocessing based on Low-Cost Reconfigurable Hardware. In: *Field-Programmable Custom Computing Machines, 2008. FCCM '08. 16th International Symposium on*, April 2008, S. 277 – 278

[204] XILINX: *Xilinx University Program Virtex-II Pro Development System Hardware Reference Manual*. UG069 (v1.0), März 2005

[205] PEDRAZA, C. ; CASTILLO, E. ; CASTILLO, J. ; CAMARERO, C. ; BOSQUE, J.L. ; MARTINEZ, J.I. ; MENENDEZ, R.: Cluster architecture based on low cost reconfigurable hardware. In: *Field Programmable Logic and Applications, 2008. FPL 2008. International Conference on*, September 2008, S. 595 – 598

[206] TAKAMAEDA-YAMAZAKI, Shinya ; SANO, Shintaro ; SAKAGUCHI, Yoshito ; FUJIEDA, Naoki ; KISE, Kenji: ScalableCore System: A Scalable Many-Core Simulator by Employing over 100 FPGAs. In: *Proceedings of the 8th international conference on Reconfigurable Computing: architectures, tools and applications*. Berlin, Heidelberg : Springer-Verlag, 2012 (ARC'12), S. 138 – 150. – ISBN 978-3-642-28364-2

[207] TAKAMAEDA-YAMAZAKI, Shinya ; SASAKAWA, Ryosuke ; SAKAGUCHI, Yoshito ; KISE, Kenji: An FPGA-Based Scalable Simulation Accelerator for Tile Architectures. In: *SIGARCH Comput. Archit. News* 39 (2011), Dezember, Nr. 4, S. 38 – 43. – ISSN 0163-5964

[208] DAVIS, John D. ; THACKER, Charles P. ; CHANG, Chen: BEE3: Revitalizing Computer Architecture Research / Microsoft Corporation. April 2009 (MSR-TR-2009-45). – TechReport

[209] CHANG, Chen ; KUUSILINNA, K. ; RICHARDS, B. ; CHEN, A. ; CHAN, N. ; BRODERSEN, R.W. ; NIKOLIC, B.: Rapid Design and Analysis of Communication Systems Using the BEE Hardware Emulation Environment. In: *Rapid System Prototyping, 2003. Proceedings. 14th IEEE International Workshop on*, 2003, S. 148 – 154. – ISSN 1074-6005

[210] RICHARDS, B. ; CHANG, Chen ; BRODERSEN, R.: DSP system design using the BEE hardware emulation environment. In: *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Seventh Asilomar Conference on* Bd. 1, 2003, S. 895 – 899

[211] ARVIND ; ASANOVIĆ, Krste ; CHIOU, Derek ; EMER, Joel ; HOE, James C. ; KOZYRAKIS, Christoforos ; LU, Shih-Lien ; PATTERSON, David ; RENAU, Jose ; WAWRZYNEK, John: Research Accelerator for Multiple Processors (RAMP) - A Shared Experimental Parallel HW/SW Platform / MIT, UT Austin, CMU, Standford, Intel, U Washington, UC Berkeley. Dezember 2008. – Forschungsbericht

[212] ASANOVIC, Krste ; BODIK, Ras ; CATANZARO, Bryan C. ; GEBIS, Joseph J. ; HUSBANDS, Parry ; KEUTZER, Kurt ; PATTERSON, David A. ; PLISHKER, William L. ; SHALF, John ; WILLIAMS, Samuel W. ; YELICK, Katherine A.: The Landscape of Parallel Computing Research: A View from Berkeley / Electrical Engineering and Computer Sciences University of California at Berkeley. Dezember 2006 (UCB/EECS-2006-183). – Forschungsbericht

[213] ELLITHORPE, J.D. ; TAN, Zhangxi ; KATZ, R.H.: Internet-in-a-Box: Emulating datacenter network architectures using FPGAs. In: *Design Automation Conference (DAC), 2009 46th ACM/IEEE*, Juli 2009, S. 880 – 883. – ISSN 0738-100X

[214] BANI ASADI, Narges ; FLETCHER, Christopher W. ; GIBELING, Greg ; GLASS, Eric N. ; SACHS, Karen ; BURKE, Daniel ; ZHOU, Zoey ; WAWRZYNEK, John ; WONG, Wing H. ; NOLAN, Garry P.: ParaLearn: A Massively Parallel, Scalable System for Learning Interaction Networks on FPGAs. In: *Proceedings of the 24th ACM International Conference on Supercomputing*. New York, NY, USA : ACM, 2010 (ICS '10), S. 83 – 94. – ISBN 978-1-4503-0018-6

[215] CHRYSOS, G. ; SOTIRIADES, E. ; ROUSOPOULOS, C. ; DOLLAS, A. ; PAPADOPOULOS, A. ; KIRMITZOGLOU, I. ; PROMPONAS, V. ; THEOCHARIDES, T. ; PETIHAKIS, G. ; LAGNEL, J. ; VAVYLIS, P. ; KOTOULAS, G.: Opportunities from

the use of FPGAs as platforms for bioinformatics algorithms. In: *Bioinformatics & Bioengineering (BIBE), 2012 IEEE 12th International Conference on*, November 2012, S. 559 – 565

[216] STEVENS, K. ; CHEN, H. ; FILIBA, T. ; MCMAHON, P. ; SONG, Y.S.: SeqHive: A Reconfigurable Computer Cluster for Genome Re-sequencing. In: *Field Programmable Logic and Applications (FPL), 2010 International Conference on*, September 2010, S. 442 – 447. – ISSN 1946-1488

[217] MÜHLBACH, S. ; KOCH, A.: A Scalable Multi-FPGA Platform for Complex Networking Applications. In: *Field-Programmable Custom Computing Machines (FCCM), 2011 IEEE 19th Annual International Symposium on*, 2011, S. 81 – 84

[218] SONMEZ, Nehir ; ARCAS, Oriol ; SAYILAR, Gokhan ; UNSAL, Osman S. ; CRISTAL, Adrián ; HUR, Ibrahim ; SINGH, Satnam ; VALERO, Mateo: From Plasma to BeeFarm: Design Experience of an FPGA-based Multicore Prototype. In: *Proceedings of the 7th international conference on Reconfigurable computing: architectures, tools and applications*. Berlin, Heidelberg : Springer-Verlag, 2011 (ARC'11), S. 350 – 362. – ISBN 978-3-642-19474-0

[219] ROTHMAN, J. ; CHANG, Chen: BEE technology overview. In: *Embedded Computer Systems (SAMOS), 2012 International Conference on*, 2012

[220] BEECUBE: *BEEcube Platform Studio BPS*, Oktober 2013

[221] BEECUBE: *BEE4 All Programmable Rack Servers*. Website. September 2013. – URL `http://www.beecube.com/products/BEE4.asp`

[222] BAXTER, R. ; BOOTH, S. ; BULL, M. ; CAWOOD, G. ; PERRY, J. ; PARSONS, M. ; SIMPSON, A. ; TREW, A. ; MCCORMICK, A. ; SMART, G. ; SMART, R. ; CANTLE, A. ; CHAMBERLAIN, R. ; GENEST, G.: Maxwell - A 64 FPGA Supercomputer. In: *Adaptive Hardware and Systems, 2007. AHS 2007. Second NASA/ESA Conference on*, August 2007, S. 287 – 294

[223] TIAN, Xiang ; BENKRID, K.: Design and Implementation of a High Performance Financial Monte-Carlo Simulation Engine on an FPGA Supercomputer. In: *ICECE Technology, 2008. FPT 2008. International Conference on*, Dezember 2008, S. 81 – 88

[224] KASAP, S. ; BENKRID, K.: A high performance implementation for Molecular Dynamics simulations on a FPGA supercomputer. In: *Adaptive Hardware and Systems (AHS), 2011 NASA/ESA Conference on*, Juni 2011, S. 375 – 382

[225] SCHMIDT, Andrew G. ; KRITIKOS, William V. ; GAO, Shanyuan ; SASS, Ron: An Evaluation of an Integrated On-chip/Off-chip Network for High-performance Reconfigurable Computing. In: *Int. J. Reconfig. Comput.* 2012 (2012), Januar, S. 5:5 – 5:5. – ISSN 1687 - 7195

[226] SCHMIDT, A.G. ; KRITIKOS, W.V. ; SHARMA, R.R. ; SASS, R.: AIREN: A Novel Integration of On-Chip and Off-Chip FPGA Networks. In: *Field Programmable Custom Computing Machines, 2009. FCCM '09. 17th IEEE Symposium on*, April 2009, S. 271 – 274

[227] SCHMIDT, A.G. ; KRITIKOS, W.V. ; DATTA, S. ; SASS, R.: Reconfigurable Computing Cluster Project: Phase I Brief. In: *Field-Programmable Custom Computing Machines, 2008. FCCM '08. 16th International Symposium on*, April 2008, S. 300 – 301

[228] RAJASEKHAR, Yamuna ; SASS, Ron: Architecture and applications for an All-FPGA parallel computer. In: *Cluster Computing* (2013), Juni, S. 1 – 11. – ISSN 1386-7857

[229] DINI GROUP: *Dini Group - Big FPGA Boards High Performance Computing*. online. August 2013. – URL `http://www.dinigroup.com`

[230] DINI GROUP: *DNV6F6PCIe Godzilla's Son-In-Law ASIC Prototyping Engine Featuring Xilinx Virtex-6*. 1.4, Juli 2010

[231] DINI GROUP: *DNK7_F5_8_Cluster Xilinx Kintex 7 Rack Mount HPC Cluster*. 1.0, Oktober 2012

[232] KALTE, Heiko: *Einbettung dynamisch rekonfigurierbarer Hardwarearchitekturen in eine Universalprozessorumgebung*, Universität Paderborn, Heinz Nixdorf Institut, Schaltungstechnik, Dissertation, 2004

[233] WENNEMERS, Pascal: *Realisierung eines Hochleistungs-FPGA-Moduls für eine skalierbare Rapid-Prototyping-Umgebung*, Universität Paderborn, Masterarbeit, April 2013

[234] BOGATIN, Eric: *Signal and Power Integrity – Simplified*. 2nd. Prentice Hall PTR, 2010

[235] JOHNSON, H.W. ; GRAHAM, M.: *High-speed Signal Propagation: Advanced Black Magic*. Prentice Hall PTR, 2003. – ISBN 9780130844088

[236] GRANBERG, T.: *Handbook of Digital Techniques for High-speed Design: Design Examples, Signaling and Memory Technologies, Fiber Optics, Modeling and Simulation to Ensure Signal Integrity*. Prentice Hall PTR, 2004 (Prentice Hall Signal Integrity Library). – ISBN 9780131422919

[237] XILINX: *Virtex-5 FPGA User Guide*. UG190 (v5.3), Mai 2010

[238] KHALID, Mohammed A.: *Routing Architecture and Layout Synthesis for Multi-FPGA Systems*, University of Toronto, Dissertation, 1999

[239] VITESSE: *VSC3008 11.5 Gbps 8×8 Asynchronous Crosspoint Switch Datasheet*. 2.2. 741 Calle Plano Camarillo, CA 93012: Vitesse Semiconductor Corporation (Veranst.), April 2010

[240] VITESSE: *VSC3308 11.5 Gbps 8 x 8 Asynchronous Crosspoint Switch and Signal Conditioner Datasheet*. 4.2. 741 Calle Plano Camarillo, CA 93012: Vitesse Semiconductor Corporation (Veranst.), Dezember 2010

[241] XILINX: *Virtex-5 FPGA Data Sheet: DC and Switching Characteristics*. DS202 (v5.3), Mai 2010

[242] XILINX: *Virtex-5 FPGA Configuration User Guide*. UG191 (v3.9.1), August 2010

[243] BUDRUK, Ravi ; ANDERSON, Don ; SHANLEY, Tom ; WINKLES, Joe (Hrsg.): *PCI Express System Architecture*. Addison Wesley, 2004

[244] PLX TECHNOLOGY: *ExpressLane PEX8311AA PCI Express-to-Generic Local Bus Bridge Data Book*. PLX Technology (Veranst.), Dezember 2009

[245] PLX TECHNOLOGY: *PCI 9656BA Data Book*. PLX Technology (Veranst.), Januar 2009

[246] PCI SPECIAL INTEREST GROUP: *PCI Express Card Electromechanical Specification Revision 1.1*. März 2005. – URL `http://www.pcisig.com/`

[247] PCI SPECIAL INTEREST GROUP: *PCI Local Bus Specification Revision 2.2*. Dezember 1998. – URL `http://www.pcisig.com/`

[248] PLX TECHNOLOGY: PCI 9xxx/PEX 8311 Local Bus Primer / PLX Technology. Januar 2010. – Forschungsbericht

[249] PLX TECHNOLOGY: *ExpressLane PEX 8648-AA, AB, and BB 48-Lane/12-Port PCI Express Gen 2 Switch Data Book*. PLX Technology (Veranst.), Oktober 2010

[250] SIMPSON, Chester: Linear and Switching Voltage Regulator Fundamental Part 1 / Texas Instruments. 2011 (SNVA558). – Forschungsbericht

[251] MIKA, Kevin: *Entwurf und Layout eines FPGA-Moduls für eine skalierbare Prototyping Plattform*, Hochschule Ostwestfalen-Lippe, Studienarbeit, August 2015

[252] XILINX: *MicroBlaze Processor Reference Guide*. UG081 (v12.0), März 2011

[253] NALLATECH: *FSB FPGA Accelerator Platform Reference Guide*. NT120-342 (v1.2). Nallatech (Veranst.), Februar 2010

[254] XILINX: *ChipScope Pro Software and Cores*. UG029 (v14.2), Juli 2012

[255] ALTERA: *Quartus II Handbook Version 12.0*. Bd. 3: Verification. Kap. 13 Design Debugging Using The SignalTap II Logic Analyzer, Altera, Juni 2012

[256] JUNGEBLUT, Thorsten ; AX, Johannes ; PORRMANN, Mario ; RUCKERT, Ulrich: A TCMS-based architecture for GALS NoCs. In: *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on*, Mai 2012, S. 2721 – 2724. – ISSN 0271-4302

[257] TUMEO, Antonino ; BRANCA, Marco ; CAMERINI, Lorenzo ; CERIANI, Marco ; MONCHIERO, Matteo ; PALERMO, Gianluca ; FERRANDI, Fabrizio ; SCIUTO, Donatella: Prototyping Pipelined Applications on a Heterogeneous FPGA Multiprocessor Virtual Platform. In: *Proceedings of the 2009 Asia and South Pacific Design Automation Conference*. Piscataway, NJ, USA : IEEE Press, Januar 2009 (ASP-DAC '09), S. 317 – 322. – ISBN 978-1-4244-2748-2

[258] DENHOLM, Stewart ; TSOI, Kuen H. ; PIETZUCH, Peter ; LUK, Wayne: Efficient communication for FPGA clusters. In: *Proceedings of the 8th international conference on Reconfigurable Computing: architectures, tools and applications*. Berlin, Heidelberg : Springer-Verlag, 2012 (ARC'12), S. 335 – 341. – ISBN 978-3-642-28364-2

[259] PELLAUER, M. ; ADLER, M. ; KINSY, M. ; PARASHAR, A. ; EMER, J.: HAsim: FPGA-Based High-Detail Multicore Simulation Using Time-Division Multiplexing. In: *High Performance Computer Architecture (HPCA), 2011 IEEE 17th International Symposium on*, Februar 2011, S. 406 – 417. – ISSN 1530-0897

[260] SAMMAN, F.A. ; PHILIPP, F. ; GLESNER, M.: Reconfigurable Interconnect Infrastructure for Multi-FPGA-based Adaptive Multiprocessing Systems. In: *Computing in Heterogeneous, Autonomous 'N' Goal-Oriented Environments (CHANGE), 2011 1st International Workshop on*, März 2011, S. 1 – 8

[261] XILINX: *16-Channel, DDR LVDS Interface with Per-Channel Alignment*. XAPP855 (v1.0), Oktober 2006

[262] XILINX: *16-Channel, DDR LVDS Interface with Real-Time Window Monitoring*. XAPP860 (v1.1), Juli 2008

[263] XILINX: *Aurora 8B/10B Protocol Specification*. SP002 (v2.2), April 2010

[264] XILINX: *Aurora 64B/66B Protocol Specification*. SP011 (v1.2), Juli 2010

[265] HENDRICKSON, Bruce ; KOLDA, Tamara G.: Graph partitioning models for parallel computing. In: *Parallel Computing* 26 (2000), Nr. 12, S. 1519 – 1534. – ISSN 0167-8191

[266] NIU, Xin Y. ; TSOI, Kuen H. ; LUK, W.: Reconfiguring Distributed Applications in FPGA Accelerated Cluster with Wireless Networking. In: *Field Programmable Logic and Applications (FPL), 2011 International Conference on*, Sept 2011, S. 545 – 550

[267] PATTERSON, David A.: Latency Lags Bandwith. In: *Commun. ACM* 47 (2004), Oktober, Nr. 10, S. 71 – 75. – ISSN 0001-0782

[268] DEHON, André: *Reconfigurable Architectures for General-Purpose Computing*, Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, Dissertation, 1996

[269] WILLIAMS, Jason ; GEORGE, Alan D. ; RICHARDSON, Justin ; GOSRANI, Kunal ; SURESH, Siddarth: Computational Density of Fixed and Reconfigurable Multi-Core Devices for Application Acceleration. In: *Proceedings of Reconfigurable Systems Summer Institute* (2008)

[270] WIRTHLIN, M. J. ; HUTCHINGS, B. L.: Improving functional density using run-time circuit reconfiguration [FPGAs]. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 6 (1998), Juni, Nr. 2, S. 247 – 256. – ISSN 1063-8210

[271] XILINX: *Spartan-3 FPGA Family Data Sheet*. DS099 (v3.1), Juni 2013

[272] ALTERA: *APEX 20K Programmable Logic Device Family*. v5.1, März 2004

[273] XILINX: *Spartan-6 FPGA Configurable Logic Block*. UG384 (v1.1), Februar 2010

[274] SCIENGINES GMBH: *COPACOBANA S3-1000 The original 'massively parallel FPGA-computer' 120 Spartan-3 1000 FPGAs for number-crunching tasks*. Website. Juli 2013. – URL `http://sciengines.com/products/computers-and-clusters/copacobana-s3-1000.html`

[275] DINI GROUP: *DNBFC_S12_12_Cluster Xilinx Spartan-6 FPGA Rack Mount FPGA/HPC Cluster*. online. Oktober 2013. – URL `http://www.dinigroup.com/new/DNBFC_S12_12_Cluster.php`

Eigene Veröffentlichungen

[276] ROMOTH, Johannes ; PORRMANN, Mario ; RÜCKERT, Ulrich: Survey of FPGA applications in the period 2000 – 2015. URL `https://doi.org/10.13140/RG.2.2.16364.56960`, März 2017. – Forschungsbericht

[277] PFAU, Timo ; PEVELING, Ralf ; SAMSON, Florian ; ROMOTH, Johannes ; HOFFMANN, Sebastian ; BHANDARE, Suhas ; IBRAHIM, Selwan K. ; SANDEL, D ; ADAMCZYK, Olaf ; PORRMANN, Mario ; NOE, Reinhold ; HAUDEN, Y. ; GROSSARD, N. ; PORTE, H. ; SCHLIEDER, D. ; KOSLOVSKY, A. ; BENARUSH, Y. ; ACHIAM, Y.: Polarization-Multiplexed 2.8 Gbit/s Synchronous QPSK Transmission with Real-Time Digital Polarization Tracking. In: *Proceedings of ECOC* ECOC (Veranst.), ECOC, Januar 2007

[278] PORRMANN, Mario ; HAGEMEYER, Jens ; ROMOTH, Johannes ; STRUGHOLTZ, Manuel ; ABDEL-WAHAB, Mohammed: Rapid Prototyping of Next-Generation Multiprocessor SoCs. In: *Proceedings of Semiconductor Conference Dresden, SCD 2009*. Dresden, Germany, April 2009

[279] POHL, C. ; HAGEMEYER, J. ; ROMOTH, J. ; PORRMANN, M. ; RUCKERT, U.: Using a Reconfigurable Compute Cluster for the Acceleration of Neural Networks. In: *Field-Programmable Technology, 2009. FPT 2009. International Conference on*, Dezember 2009, S. 368 – 371

[280] PORRMANN, Mario ; HAGEMEYER, Jens ; POHL, Christopher ; ROMOTH, Johannes ; STRUGHOLTZ, Manuel: RAPTOR – A Scalable Platform for Rapid Prototyping and FPGA-based Cluster Computing. In: *Parallel Computing: From Multicores and GPU's to Petascale* Bd. 19, IOS Press, 2010, S. 592 – 599

[281] ROMOTH, Johannes ; JUNGEWELTER, Dirk ; HAGEMEYER, Jens ; PORRMANN, Mario ; RÜCKERT, Ulrich: Optimizing Inter-FPGA Communication by Automatic Channel Adaptation. In: *Reconfigurable Computing and FPGAs (ReConFig), 2012 International Conference on*, Dezember 2012, S. 1 – 7

[282] ROMOTH, Johannes ; HAGEMEYER, Jens ; PORRMANN, Mario ; RÜCKERT, Ulrich: Fast Design-space Exploration with FPGA Cluster. In: *DATE 2011 Workshop on Design Methods and Tools for FPGA-Based Acceleration of Scientific Computing*. Grenoble, France, März 2011

Betreute Arbeiten

[283] GEHRING, Sebastian: *Dezentrales Kommunikationssystem für verteilte Recheneinheiten in modularen, eingebetteten Systemen*, Universität Paderborn, Diplomarbeit, April 2012

[284] GRIESSL, René: *Eine partiell rekonfigurierbare Bildverarbeitungsarchitektur für ressourceneffiziente Systeme*, Universität Paderborn, Diplomarbeit, 2012

[285] URBAN, Holger: *Entwurf eines FPGA-basierten Prototyping Systems unter besonderer Berücksichtigung der Signalintegrität*, Universität Paderborn, Diplomarbeit, April 2010

[286] HILGENSTEIN, Arne: *Multi-Gigabit-Switch für Inter-FPGA-Kommunikation*, Universität Paderborn, Diplomarbeit, Juli 2011

[287] JUNGEWELTER, Dirk: *Systemintegration einer modernen Rapidprototyping Plattform RAPTOR-XPress*, Universität Paderborn, Studienarbeit, Juli 2010

[288] JUNGEWELTER, Dirk: *Skalierbare Datenübertragungslösung zur breitbandigen Kopplung von FPGAs*, Universität Paderborn, Diplomarbeit, Mai 2011

Anhang A

RAPTOR-XPress-Aufbau

Im Folgenden wird der RAPTOR-XPress-Aufbau detailliert dargestellt (siehe hierzu auch Kapitel 5). Abbildung A.1 erweitert das Blockschaltbild 5.10. Der vollständige Lagenaufbau zur Fertigung des RAPTOR-XPress-Trägersystems wird in Abbildung A.2 aufgezeigt.

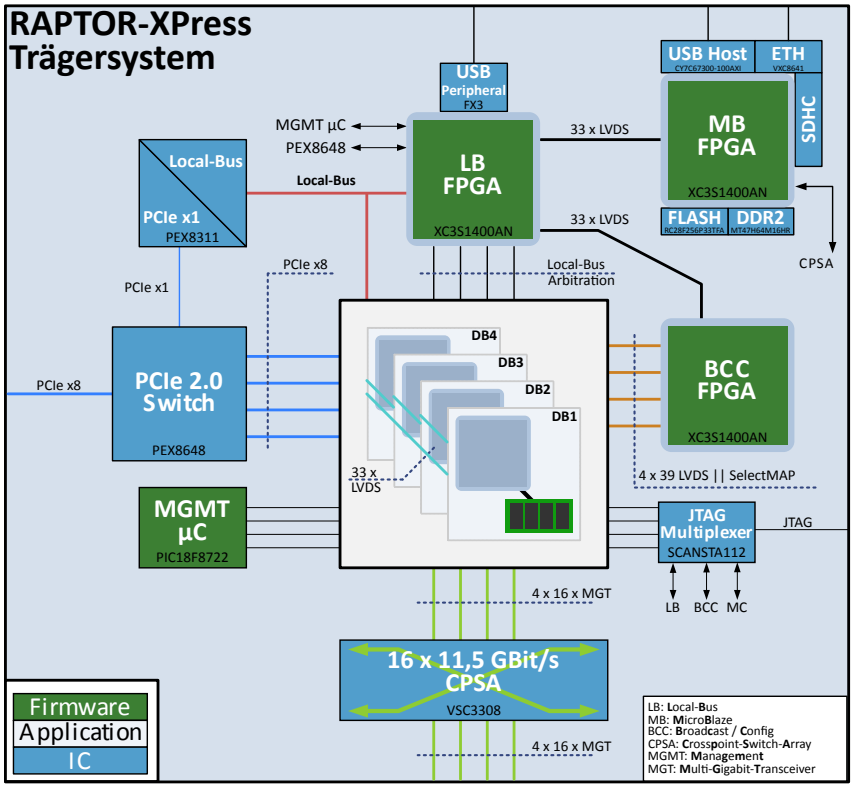


Abbildung A.1: Architekturübersicht des RAPTOR-XPress

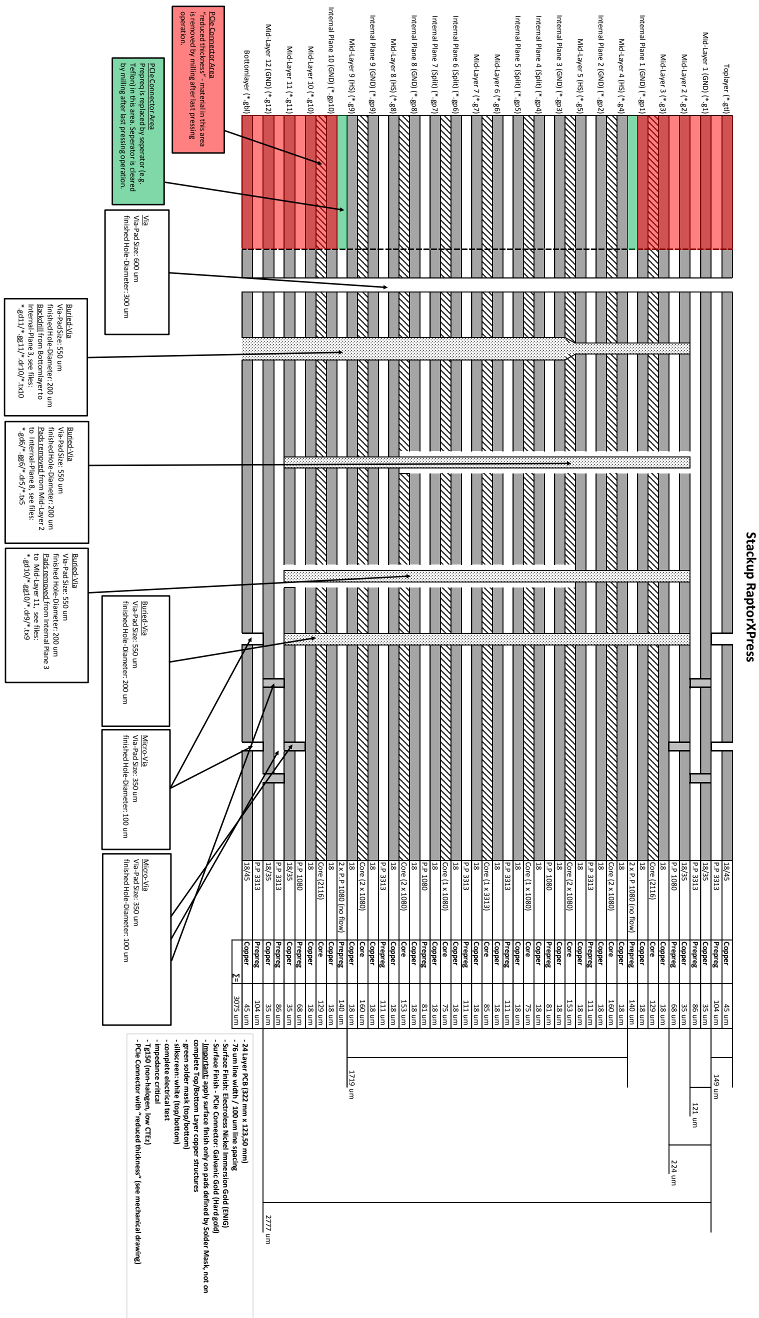


Abbildung A.2: Lagenaufbau des RAPTOR-XPress

Anhang B

Architekturübersicht verschiedener FPGA-Cluster

Ergänzend zu den in Kapitel 4.2 vorgestellten und mit Blick auf die Kommunikationsinfrastruktur und die Speicheranbindung fokussierten Darstellungen der FPGA-Cluster erfolgt in diesem Abschnitt eine Auflistung der unveränderten Blockschaltbilder der jeweiligen Hersteller.

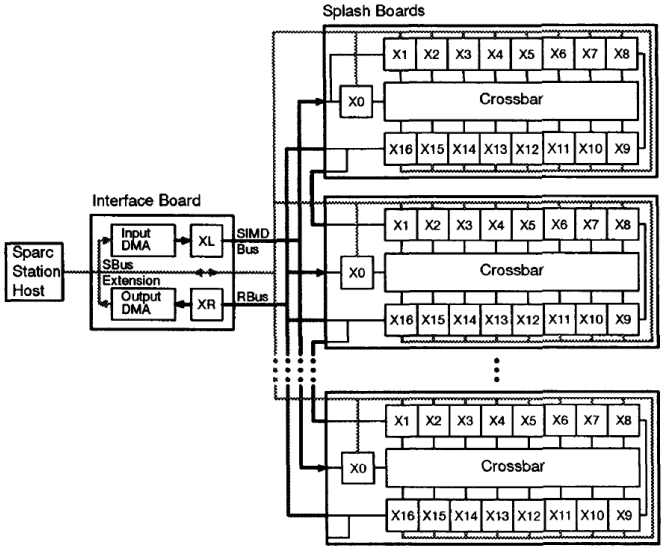


Abbildung B.1: Architekturübersicht des Splash-2-FPGA-Clusters aus Abschnitt 4.2 [179]

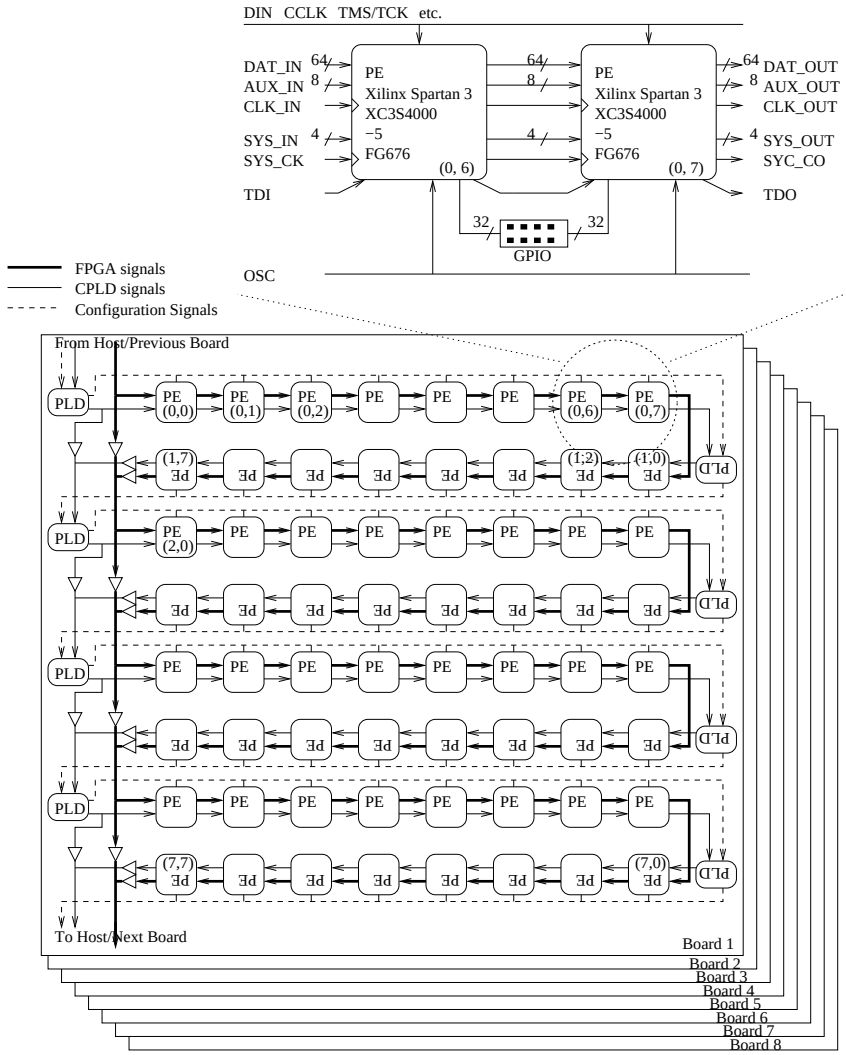


Abbildung B.2: Architekturübersicht des CUBE-FPGA-Clusters aus Abschnitt 4.2.1 [6]

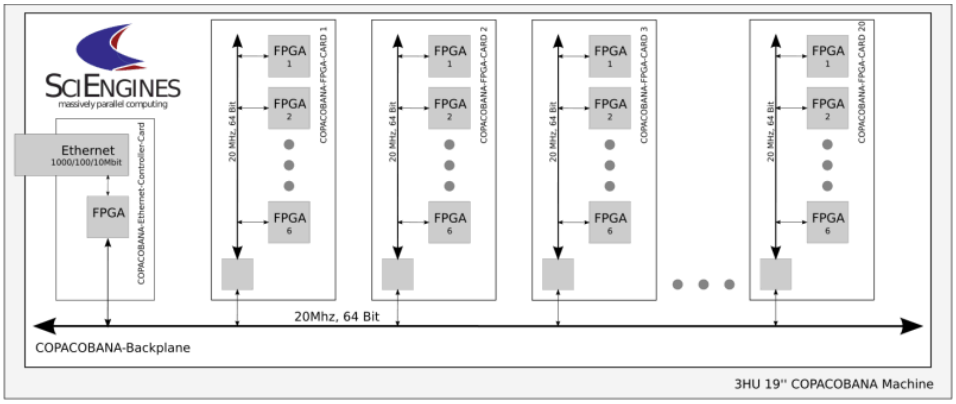


Abbildung B.3: Architekturübersicht des COPACOBANA-FPGA-Clusters aus Abschnitt 4.2.1 [274]

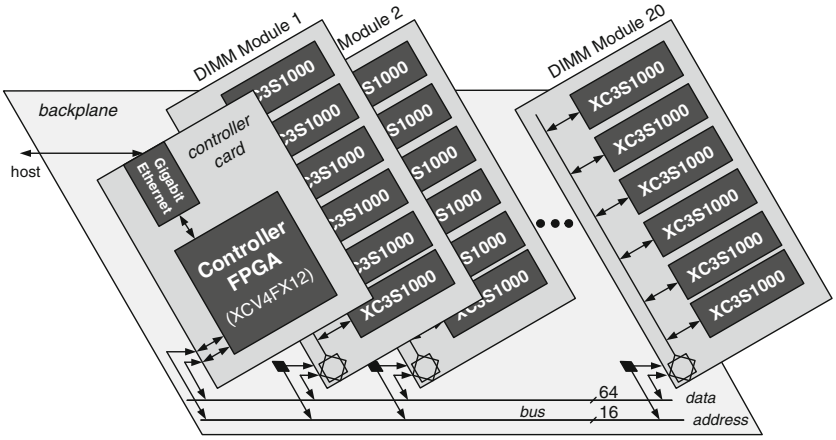


Abbildung B.4: Architekturübersicht des COPACOBANA2-FPGA-Clusters aus Abschnitt 4.2.1 [186]

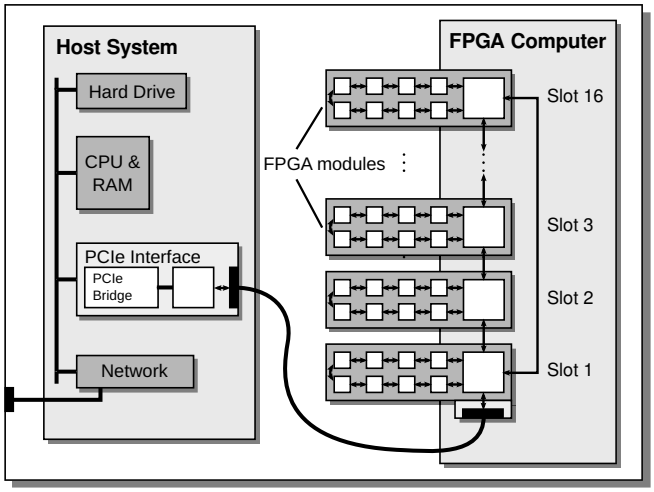


Abbildung B.5: Architekturübersicht des RIVYERA-S3-FPGA-Clusters aus Abschnitt 4.2.1 [66]

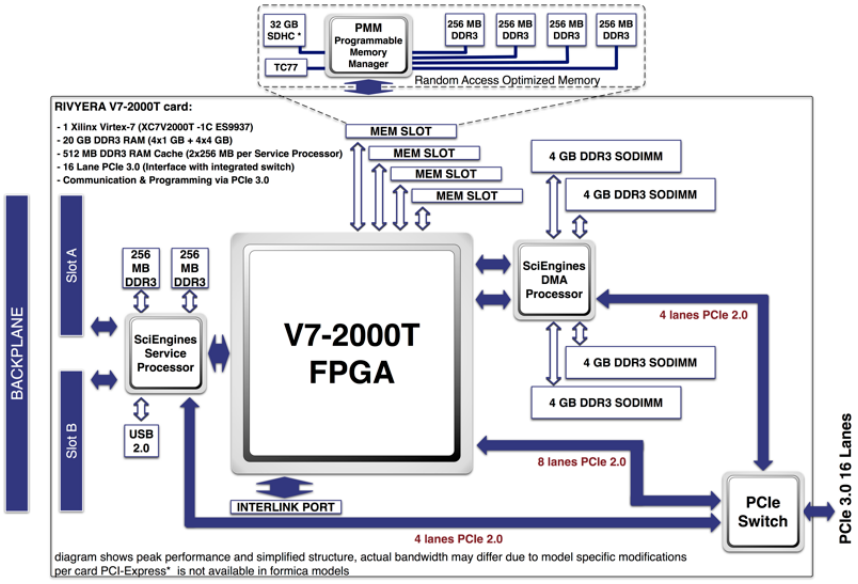


Abbildung B.6: Architekturübersicht eines RIVYERA-V7-Moduls aus Abschnitt 4.2.1 [195]

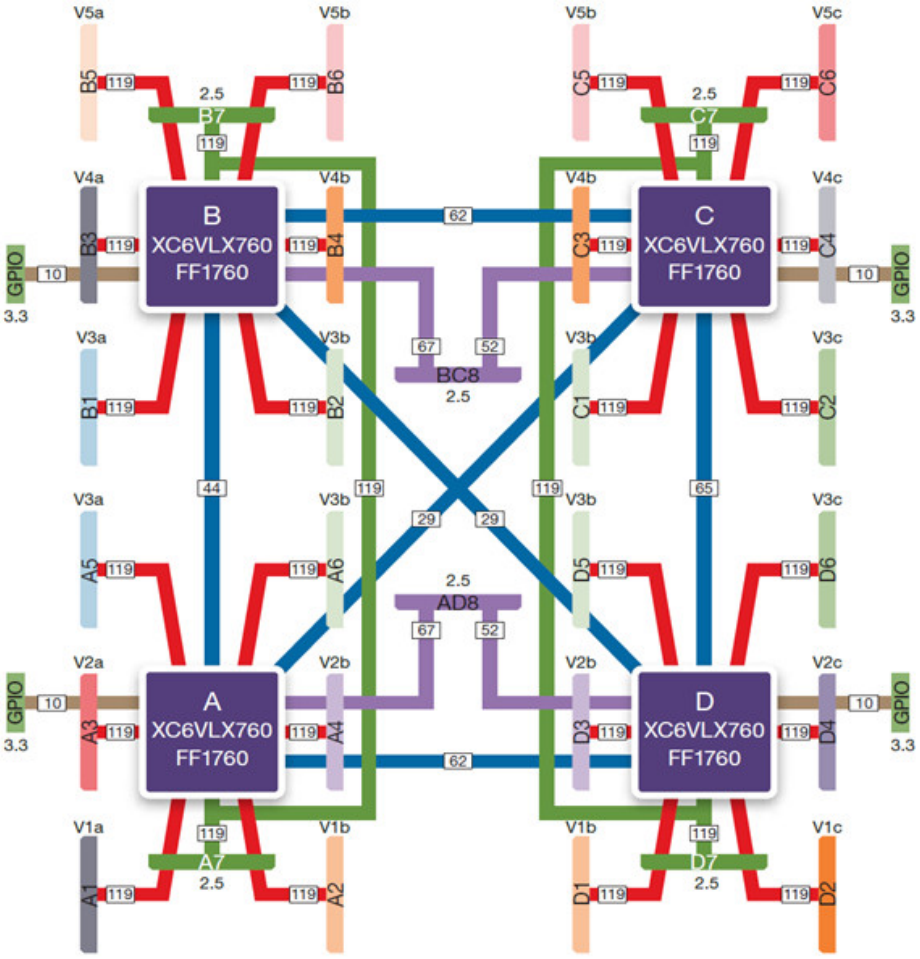


Abbildung B.7: Architekturübersicht eines HAPS-64 aus Abschnitt 4.2.1 [199]

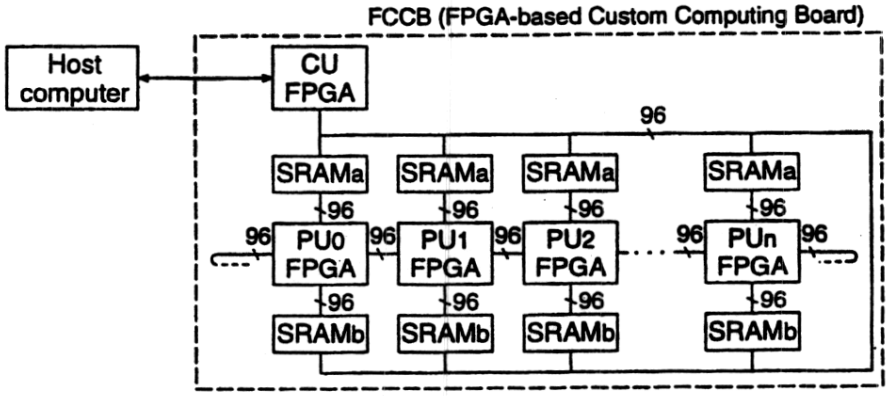


Figure 3: System Configuration

Abbildung B.8: Architekturübersicht des FCCB aus Abschnitt 4.2.1 [200]

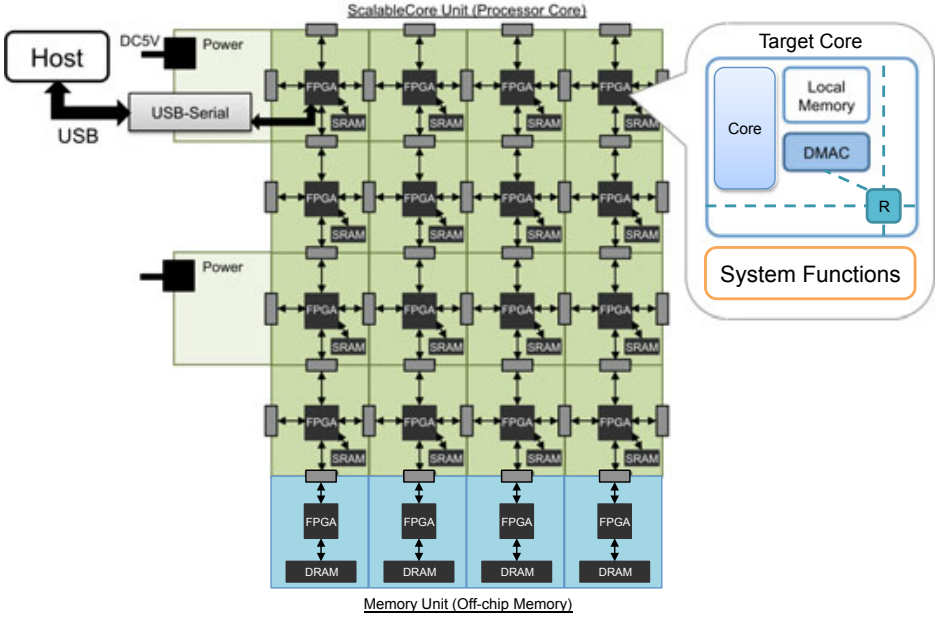


Abbildung B.9: Architekturübersicht eines ScalableCore-FPGA-Clusters aus Abschnitt 4.2.2 [206]

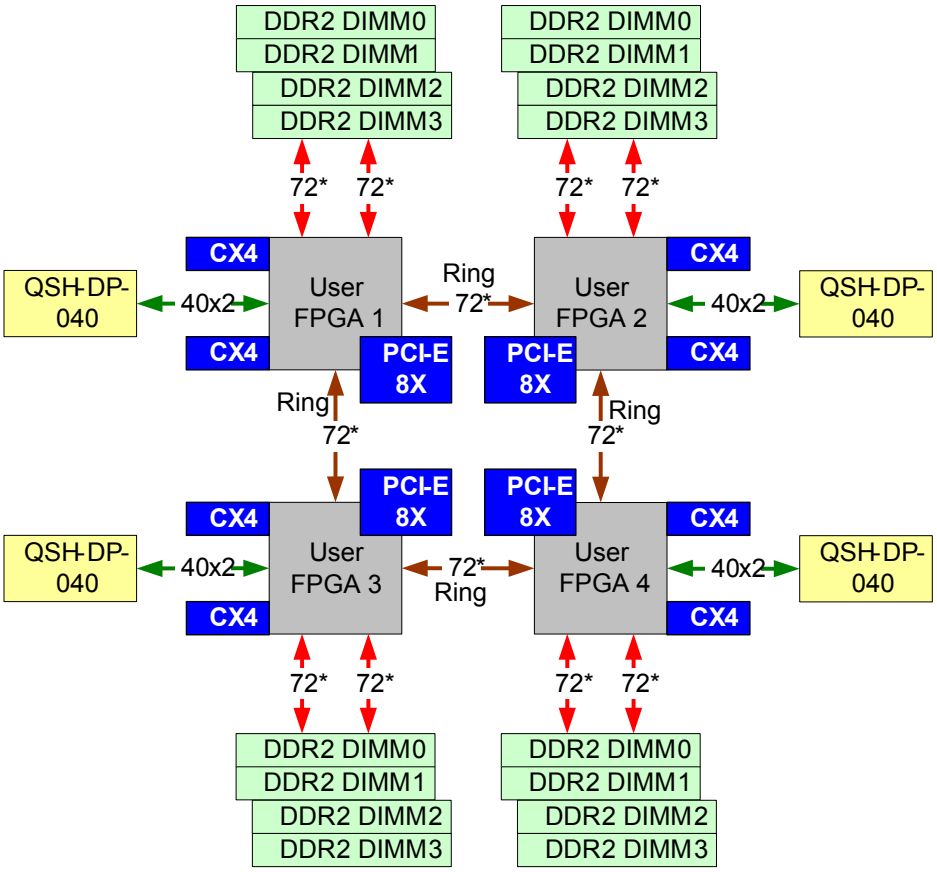


Abbildung B.10: Architekturübersicht eines BEE3-Basissystems aus Abschnitt 4.2.2 [208]

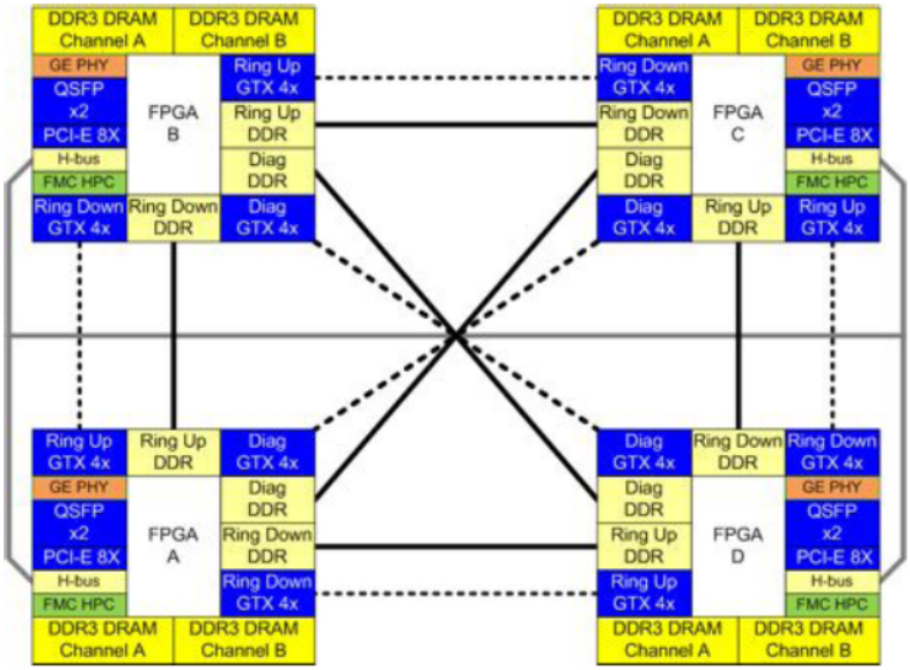


Abbildung B.11: Architekturübersicht eines BEE4-Basissystems aus Abschnitt 4.2.2 [219]

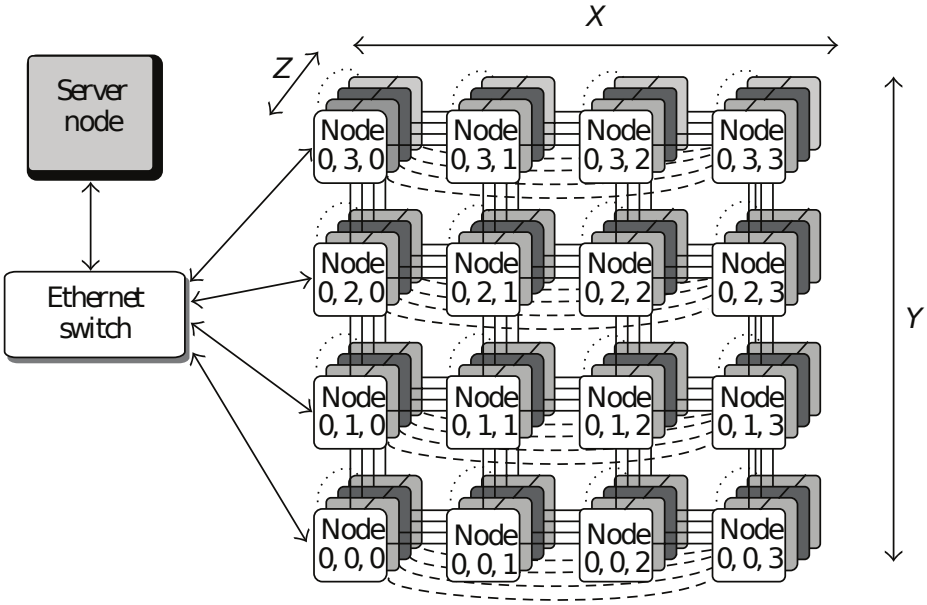
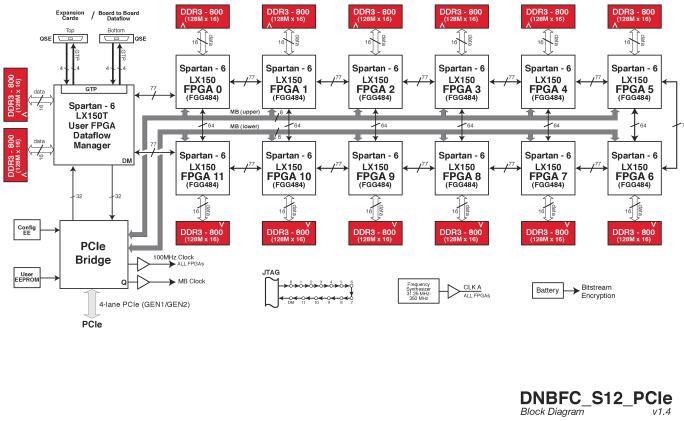
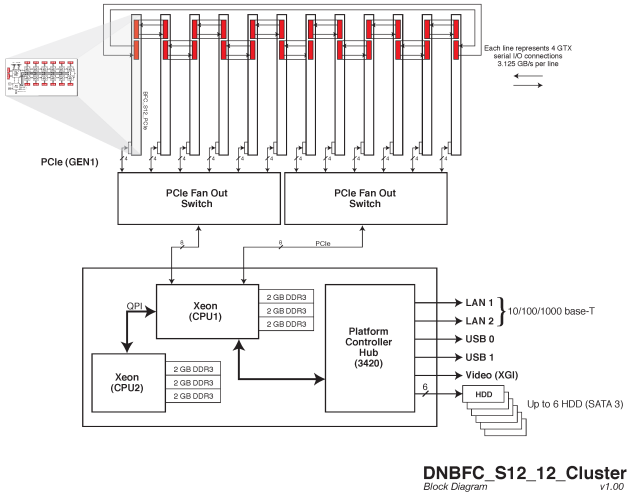


Abbildung B.12: Architekturübersicht des Spirit-FPGA-Clusters aus Abschnitt 4.2.2 [225]



(a) Architekturübersicht eines DNBFC_S12_PCIe-Systems aus Abschnitt 4.2.2 [275]



(b) Architekturübersicht des DNBFC_S12_12-FPGA-Clusters aus Abschnitt 4.2.2 [275]

Abbildung B.13: Systemkomponenten des DNBFC_S12_12-FPGA-Clusters aus Abschnitt 4.2.2 [275]

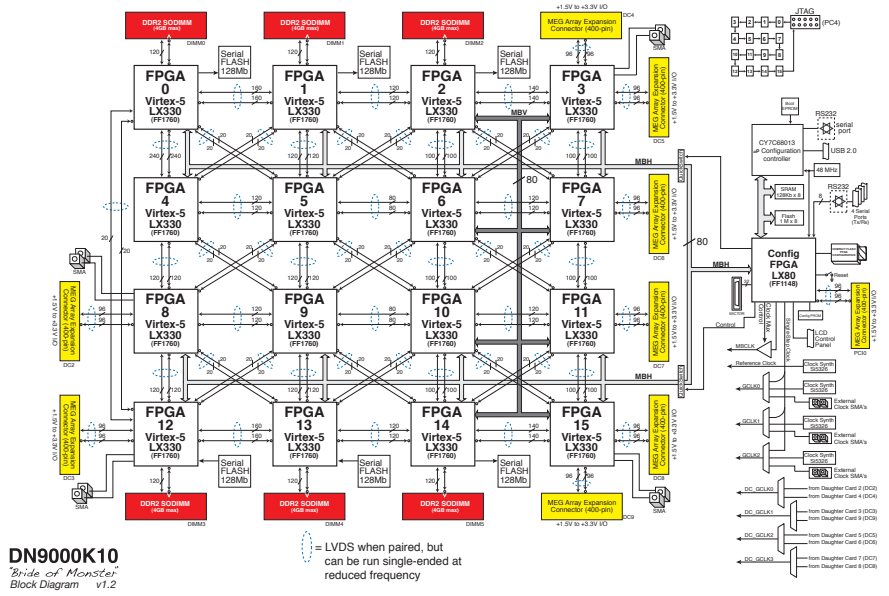


Abbildung B.14: Architekturübersicht eines DN9000K10 MFS aus Abschnitt 4.1 [174]

Anhang C

Vollständige Verbindungsvarianten des RAPTOR-XPress

Im Folgenden werden die im Beispiel von Kapitel 3.2 nur in Ausschnitten aufgeführten Schnittstellenmodellierungen mit zwei Modulen à zwei Schnittstellen vollständig dargestellt.

$$
\begin{aligned}
\Xi\left(M \times S\right) =& E((m_1, s_1)) \cup E((m_1, s_2)) \cup E((m_2, s_1)) \cup E((m_2, s_2)) \\
=& \{\ \emptyset, \\
& \{(m_1, s_1) \to (m_1, s_1)\}, \{(m_1, s_2) \to (m_1, s_1)\}, \\
& \{(m_2, s_1) \to (m_1, s_1)\}, \{(m_2, s_2) \to (m_1, s_1)\}, \\
& \{(m_1, s_1) \to (m_1, s_2)\}, \{(m_1, s_2) \to (m_1, s_2)\}, \\
& \{(m_2, s_1) \to (m_1, s_2)\}, \{(m_2, s_2) \to (m_1, s_2)\}, \\
& \{(m_1, s_1) \to (m_2, s_1)\}, \{(m_1, s_2) \to (m_2, s_1)\}, \\
& \{(m_2, s_1) \to (m_2, s_1)\}, \{(m_2, s_2) \to (m_2, s_1)\}, \\
& \{(m_1, s_1) \to (m_2, s_2)\}, \{(m_1, s_2) \to (m_2, s_2)\}, \\
& \{(m_2, s_1) \to (m_2, s_2)\}, \{(m_2, s_2) \to (m_2, s_2)\}, \\
& \{(m_1, s_1) \to \{(m_1, s_1), (m_1, s_2)\}\}, \{m_1, s_1 \to \{(m_1, s_1), (m_2, s_1)\}\}, \\
& \{(m_1, s_1) \to \{(m_1, s_2), (m_2, s_1)\}\}, \{m_1, s_1 \to \{(m_1, s_1), (m_2, s_2)\}\}, \\
& \{(m_1, s_1) \to \{(m_1, s_2), (m_2, s_2)\}\}, \{m_1, s_1 \to \{(m_2, s_1), (m_2, s_2)\}\}, \\
& \{(m_1, s_2) \to \{(m_1, s_1), (m_1, s_2)\}\}, \{m_1, s_2 \to \{(m_1, s_1), (m_2, s_1)\}\}, \\
& \{(m_1, s_2) \to \{(m_1, s_2), (m_2, s_1)\}\}, \{m_1, s_2 \to \{(m_1, s_1), (m_2, s_2)\}\}, \\
& \{(m_1, s_2) \to \{(m_1, s_2), (m_2, s_2)\}\}, \{m_1, s_2 \to \{(m_2, s_1), (m_2, s_2)\}\}, \\
& \{(m_2, s_1) \to \{(m_1, s_1), (m_1, s_2)\}\}, \{m_2, s_1 \to \{(m_1, s_1), (m_2, s_1)\}\}, \\
& \{(m_2, s_1) \to \{(m_1, s_2), (m_2, s_1)\}\}, \{m_2, s_1 \to \{(m_1, s_1), (m_2, s_2)\}\}, \\
& \{(m_2, s_1) \to \{(m_1, s_2), (m_2, s_2)\}\}, \{m_2, s_1 \to \{(m_2, s_1), (m_2, s_2)\}\}, \\
& \{(m_2, s_2) \to \{(m_1, s_1), (m_1, s_2)\}\}, \{m_2, s_2 \to \{(m_1, s_1), (m_2, s_1)\}\},
\end{aligned}
$$

$$\{(m_2,s_2) \rightarrow \{(m_1,s_2),(m_2,s_1)\}\},\{m_2,s_2 \rightarrow \{(m_1,s_1),(m_2,s_2)\}\},$$
$$\{(m_2,s_2) \rightarrow \{(m_1,s_2),(m_2,s_2)\}\},\{m_2,s_2 \rightarrow \{(m_2,s_1),(m_2,s_2)\}\},$$
$$\{(m_1,s_1) \rightarrow \{(m_1,s_1),(m_1,s_2),(m_2,s_1)\}\},$$
$$\{(m_1,s_1) \rightarrow \{(m_1,s_1),(m_1,s_2),(m_2,s_2)\}\},$$
$$\{(m_1,s_1) \rightarrow \{(m_1,s_1),(m_2,s_1),(m_2,s_2)\}\},$$
$$\{(m_1,s_1) \rightarrow \{(m_1,s_2),(m_2,s_1),(m_2,s_2)\}\},$$
$$\{(m_1,s_2) \rightarrow \{(m_1,s_1),(m_1,s_2),(m_2,s_1)\}\},$$
$$\{(m_1,s_2) \rightarrow \{(m_1,s_1),(m_1,s_2),(m_2,s_2)\}\},$$
$$\{(m_1,s_2) \rightarrow \{(m_1,s_1),(m_2,s_1),(m_2,s_2)\}\},$$
$$\{(m_1,s_2) \rightarrow \{(m_1,s_2),(m_2,s_1),(m_2,s_2)\}\},$$
$$\{(m_2,s_1) \rightarrow \{(m_1,s_1),(m_1,s_2),(m_2,s_1)\}\},$$
$$\{(m_2,s_1) \rightarrow \{(m_1,s_1),(m_1,s_2),(m_2,s_2)\}\},$$
$$\{(m_2,s_1) \rightarrow \{(m_1,s_1),(m_2,s_1),(m_2,s_2)\}\},$$
$$\{(m_2,s_1) \rightarrow \{(m_1,s_2),(m_2,s_1),(m_2,s_2)\}\},$$
$$\{(m_2,s_2) \rightarrow \{(m_1,s_1),(m_1,s_2),(m_2,s_1)\}\},$$
$$\{(m_2,s_2) \rightarrow \{(m_1,s_1),(m_1,s_2),(m_2,s_2)\}\},$$
$$\{(m_2,s_2) \rightarrow \{(m_1,s_1),(m_2,s_1),(m_2,s_2)\}\},$$
$$\{(m_2,s_2) \rightarrow \{(m_1,s_2),(m_2,s_1),(m_2,s_2)\}\},$$
$$\{(m_1,s_1) \rightarrow \{(m_1,s_1),(m_1,s_2),(m_2,s_1),(m_2,s_2)\}\},$$
$$\{(m_1,s_2) \rightarrow \{(m_1,s_1),(m_1,s_2),(m_2,s_1),(m_2,s_2)\}\},$$
$$\{(m_2,s_1) \rightarrow \{(m_1,s_1),(m_1,s_2),(m_2,s_1),(m_2,s_2)\}\},$$
$$\{(m_2,s_2) \rightarrow \{(m_1,s_1),(m_1,s_2),(m_2,s_1),(m_2,s_2)\}\}\ \}$$

Die im Folgenden dargestellte Distanzmatrix vervollständigt die im Beispiel von Kapitel 3.4 gegebenen Informationen.

$$
\begin{array}{c}
 & \begin{array}{cccccccccccccccccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 & v_9 & v_{10} & v_{11} & v_{12} & v_{13} & v_{14} & v_{15} & v_{16} & v_{17} & v_{18} & v_{19} & v_{20} & v_{21} & v_{22} \end{array} \\
\begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \\ v_9 \\ v_{10} \\ v_{11} \\ v_{12} \\ v_{13} \\ v_{14} \\ v_{15} \\ v_{16} \\ v_{17} \\ v_{18} \\ v_{19} \\ v_{20} \\ v_{21} \\ v_{22} \end{array} &
\left(\begin{array}{cccccccccccccccccccccc}
0 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 4 & 4 & 4 & 4 \\
1 & 0 & 2 & 2 & 1 & 1 & 2 & 3 & 2 & 3 & 2 & 2 & 3 & 3 & 4 & 3 & 3 & 4 & 3 & 3 & 4 & 4 \\
1 & 2 & 0 & 2 & 2 & 3 & 1 & 1 & 3 & 3 & 3 & 4 & 2 & 2 & 2 & 3 & 3 & 3 & 3 & 4 & 3 & 4 \\
1 & 2 & 2 & 0 & 2 & 3 & 3 & 3 & 1 & 1 & 3 & 4 & 3 & 3 & 3 & 2 & 2 & 2 & 4 & 3 & 4 & 3 \\
2 & 1 & 2 & 2 & 0 & 2 & 1 & 3 & 1 & 3 & 3 & 3 & 2 & 3 & 4 & 2 & 3 & 4 & 4 & 4 & 3 & 3 \\
2 & 1 & 3 & 3 & 2 & 0 & 3 & 3 & 3 & 3 & 1 & 1 & 3 & 2 & 3 & 3 & 2 & 3 & 2 & 2 & 3 & 3 \\
2 & 2 & 1 & 3 & 1 & 3 & 0 & 2 & 2 & 3 & 3 & 4 & 1 & 3 & 3 & 3 & 2 & 4 & 3 & 4 & 2 & 3 \\
2 & 3 & 1 & 3 & 3 & 3 & 2 & 0 & 3 & 3 & 2 & 3 & 3 & 1 & 1 & 2 & 3 & 2 & 2 & 3 & 3 & 3 \\
2 & 2 & 3 & 1 & 1 & 3 & 2 & 3 & 0 & 2 & 3 & 4 & 3 & 2 & 4 & 1 & 3 & 3 & 4 & 3 & 3 & 2 \\
2 & 3 & 3 & 1 & 3 & 3 & 3 & 3 & 2 & 0 & 2 & 3 & 2 & 3 & 2 & 3 & 1 & 1 & 3 & 2 & 3 & 3 \\
3 & 2 & 3 & 3 & 3 & 1 & 3 & 2 & 3 & 2 & 0 & 2 & 2 & 1 & 3 & 2 & 1 & 3 & 3 & 3 & 3 & 3 \\
3 & 2 & 4 & 4 & 3 & 1 & 4 & 3 & 4 & 3 & 2 & 0 & 3 & 3 & 2 & 3 & 3 & 2 & 1 & 1 & 2 & 2 \\
3 & 3 & 2 & 3 & 2 & 3 & 1 & 3 & 3 & 2 & 2 & 3 & 0 & 3 & 3 & 3 & 1 & 3 & 2 & 3 & 1 & 2 \\
3 & 3 & 2 & 3 & 3 & 2 & 3 & 1 & 2 & 3 & 1 & 3 & 3 & 0 & 2 & 1 & 2 & 3 & 3 & 3 & 3 & 2 \\
3 & 4 & 2 & 3 & 4 & 3 & 3 & 1 & 4 & 2 & 3 & 2 & 3 & 2 & 0 & 3 & 3 & 1 & 1 & 2 & 2 & 3 \\
3 & 3 & 3 & 2 & 2 & 3 & 3 & 2 & 1 & 3 & 2 & 3 & 3 & 1 & 3 & 0 & 3 & 3 & 3 & 2 & 2 & 1 \\
3 & 3 & 3 & 2 & 3 & 2 & 2 & 3 & 3 & 1 & 1 & 3 & 1 & 2 & 3 & 3 & 0 & 2 & 3 & 3 & 2 & 3 \\
3 & 4 & 3 & 2 & 4 & 3 & 4 & 2 & 3 & 1 & 3 & 2 & 3 & 3 & 1 & 3 & 2 & 0 & 2 & 1 & 3 & 2 \\
4 & 3 & 3 & 4 & 4 & 2 & 3 & 2 & 4 & 3 & 3 & 1 & 2 & 3 & 1 & 3 & 3 & 2 & 0 & 2 & 1 & 2 \\
4 & 3 & 4 & 3 & 4 & 2 & 4 & 3 & 3 & 2 & 3 & 1 & 3 & 3 & 2 & 2 & 3 & 1 & 2 & 0 & 2 & 1 \\
4 & 4 & 3 & 4 & 3 & 3 & 2 & 3 & 3 & 3 & 3 & 2 & 1 & 3 & 2 & 2 & 2 & 3 & 1 & 2 & 0 & 1 \\
4 & 4 & 4 & 3 & 3 & 3 & 3 & 3 & 2 & 3 & 3 & 2 & 2 & 2 & 3 & 1 & 3 & 2 & 2 & 1 & 1 & 0
\end{array} \right)
\end{array}
\tag{C.1}
$$

Abbildungsverzeichnis

Tabellenverzeichnis

Gedruckt auf alterungsbeständigem Papier °° ISO 9706