Bachelor Thesis

VHDL-Design eines Rahmengenerators für die Implementierung der Ethernet MAC Protokollschicht

An der Fachhochschule Dortmund
im Fachbereich Elektrotechnik
Studiengang Energie und Umwelt
erstellte Thesis
zur Erlangung des akademischen Grades
Bachelor of Engineering (B. Eng.)

von

Björn Fiegler
geboren am 15.06.1982
Matrikelnummer: 7078020

Betreuung durch:
Prüfer/Betreuer: Prof. Dr-Ing. Michael Karagounis
M. Eng. Andreas Stiller

1. Dezember 2017

Thema der Bachelorthesis

VHDL-Design eines Rahmengenerators für die Implementierung der Ethernet MAC Protokollschicht.

Kurzzusammenfassung

Der Inhalt dieser Bachelorthesis dokumentiert das Design eines Rahmengenerators für Ethernet in der Hardware Beschreibungssprache *Very High Speed Integrated Circuit Hardware Description Language (VHDL)*. Das langfristige Ziel ist eine für das Labor anpassbare Version von Ethernet mit der Kommunikation, Konfiguration und Wartungsaufgaben an *Field Programmable Gate Array (FPGA)* oder *Application Specific Integrated Circuit (ASIC)* Projekten über längere Distanzen stattfinden kann.

Title of the Paper

VHDL design of a framebuilder for the implementation of the Ethernet MAC Layer.

Abstract

The content of this Bachelorthesis is the design of a framebuilder for Ethernet in the hardware description language *VHDL*. The long-term goal is to provide a lab-adaptable version of Ethernet with the communication, configuration, and maintenance tasks available at *FPGA* or *ASIC* projects over longer distances.

Erklärung

Hiermit versichere ich, dass die von mir vorgelegte Prüfungsleistung selbstständig und ohne unzulässige fremde Hilfe erstellt worden ist. Alle verwendeten Quellen sind in der Arbeit so aufgeführt, dass Art und Umfang der Verwendung nachvollziehbar sind.

Ort, Datum	Unterschrift

Inhaltsverzeichnis

Abkürzungsverzeichnis

ASIC	Application Specific Integrated Circuit
AUI	Attachment User Interface
CAN	Controller Area Network
CMOS	Complementary Metal-Oxide-Semiconductor
CRC	Cyclic Redundancy Check
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
FDDI	Fiber Distributed Data Interface
FPGA	Field Programmable Gate Array
FTP	File Transfer Protokoll
FCS	Frame Check Sequence
GAA	Global Administrated Address
IEEE	Institute of Electrical and Electronics Engineers
IP	Intellectual Property
LAA	Local Administrated Address
LSB	Least Significant Bit
MAC	Medium Access Control
MII	Medium Independent Interface
MSB	Most Significant Bit
OSI	Open Systems Interconnection
OUI	Organizationally Unique Identifier
PAD	Padding-Bits
PMA	Physical Medium Attachment
RMII	Reduced Medium Independent Interface
RTDT	Round Trip Delay Time
SFD	Start Frame Delimiter
TTL	Transistor-Transistor-Logik
VHDL	Very High Speed Integrated Circuit Hardware Description Language
VLAN	Virtual LAN

1 Einleitung

1.1 Übersicht

Ethernet hat sich seit seiner Entwicklung im Jahr 1973 zum Standard für Datenkommunikation entwickelt. Das Ethernet Protokoll hat sich in vielen Anwendungsbereichen fest etabliert und hat kürzlich sogar im Automotive Bereich Fuß gefasst. Dank seiner deutlich höheren Übertragungsgeschwindigkeit, gibt es sogar Bestrebungen den Standard Controller Area Network (CAN) durch Ethernet abzulösen. Anfangs nur mit einer sehr geringen Bandbreite von ca. 3 MBit/s spezifiziert, können heutzutage Geschwindigkeiten bis zu 100 GBit/s erreicht werden. Aktuelle arbeitet das Institute of Electrical and Electronics Engineers (IEEE) an Standards für 200- und 400 GBit/s.
Bei CAN ist die Übertragungsgeschwindigkeit auf maximal 1 MBit/s limitiert. Dies ermöglicht die Implementation von schnelleren und intelligenteren Fahrassistenz-/ und Sicherheitssystemen. Das Angebot an Ethernet Bausteinen ist zwar groß, jedoch bieten die Hersteller Ihre Komponenten nur in bestimmter Konfiguration an, die nicht den Kundenwünschen entsprechen und die Schnelligkeit des Systems beeinflussen. Auch die direkte Einbindung fertiger Ethernetblöcke in ASIC-Projekte stellt sich schwierig da. Die Hardware-Designs sind Verschlusssache und werden als Intellectual Property (IP) geführt. Das Ziel des Projektes besteht darin, einen genauen Überblick auf die Spezifikation des Ethernet-Protokolls zu erhalten, dass eine Implementierung in eigenen Hardwaredesigns möglich ist und es auf die Projektbedürfnisse angepasst werden kann.

1.2 Ziele dieser Arbeit

Die IMES Laborgruppe strebt die Entwicklung einer eigenen Ethernetschnittstelle gemäß des IEEE-802.3 - Standards an, um diese in verschiedenen Projekten zur Kommunikation mit Testchips einsetzen zu können. Es soll möglich sein bei ASIC oder FPGA-Designs eine Fernwartung, Updates oder Anfragen durchzuführen. Die bestehenden Lösungen sind entweder auf eine bestimmte Art/Typ von FPGA zugeschnitten oder beinhalten von Firmen Entwickelte IP. Daher besteht der Wunsch nach einer komplett selbst entwickelten Schnittstelle. Hier können wiederkehrende Funktionen fest oder flexibel im Design eingestellt und bei Bedarf optimiert werden. Diese Arbeit behandelt einen Teil des MAC-Layers, den sogenannten Frame Builder oder auch Rahmengenerator. In ihm werden die zu übertragenden Daten auf die im Standard geforderte Formatierung gebracht [Soc16], damit die Kommunikation mit anderen Ethernet kompatiblen Systemen möglich ist. Zusätzlich findet eine Prüfsummen Berechnung auf Grundlage des Cyclic Redundancy Checks (CRC) statt. Diese Prüfsumme kann für die Detektion von Fehlern während der Versendung eines kompletten Ethernet Frames verwendet werden.

2 OSI-Referenzmodell

Ethernet ist ein Protokoll welches in Schicht eins und zwei von den Schichten des Open Systems Interconnection (OSI)-Referenzmodels definiert ist. Zum besseren Verständnis der Kommunikation innerhalb des Netzwerks wird nachfolgend das Model und seine Schichten dargestellt.

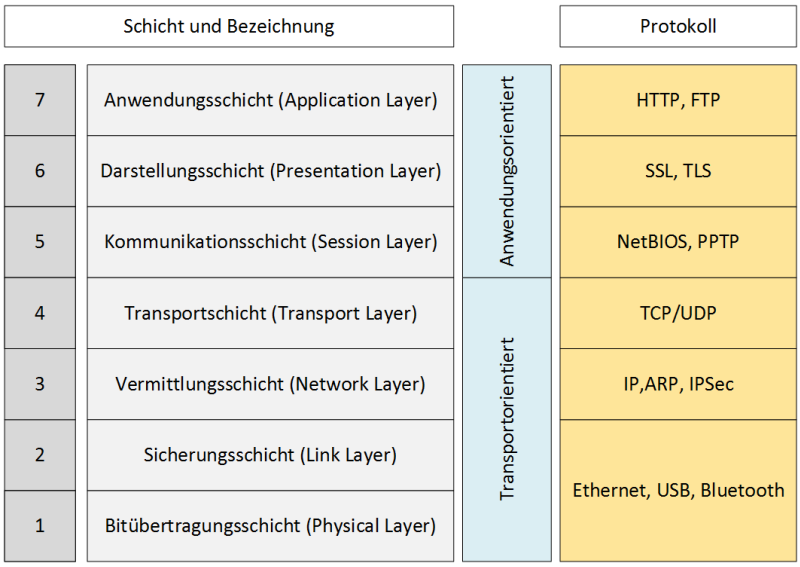

Abbildung 1: *OSI-Referenzmodel*

2.1 Schicht 1: Physical Layer (Bitübertragungsschicht)

Die Bitübertragungsschicht ist für die Übertragung der Informationen (Bits) ohne einen bestimmen Rahmen über das jeweilige Übertragungsmedium (Kupfer, Glasfaser) zuständig. Es werden folgende Parameter[Rec08] definiert:

- Das jeweilige genutzte Medium
- Übertragungsgeschwindigkeit
- Spannungswerte der logischen Größen
- Definition einzelner Leitungsfunktionen (Steuer- oder Datenleitung)
- Übertragungsrichtung: Simplex, Halb- und Voll-Duplex.
  - **Simplex**: Entweder in eine Richtung: Station A –> Station B
  - **Halbduplex**: abwechselnd in beide Richtungen.
    Erst Station A –> Station B dann Station A <– Station B
  - **Duplex**: Gleichzeitig Senden und Empfangen Station A <–> Station B

Die Bitübertragungsschicht hat je nach Geschwindigkeit und Medium einen abweichenden Aufbau. Dies wird in Kapitel 3 näher erläutert.

2.2 Schicht 2: Link Layer (Sicherungsschicht)

Die Sicherungsschicht sorgt für den zuverlässigen und reibungslosen Austausch der Daten zwischen den kommunizierenden Systemen. Nach einer Paketübermittlung führt der Empfänger eine Fehlererkennung durch und quittiert dem Sender gegebenenfalls korrekt übertragene Datenpakete. Sie ist in zwei Schichten unterteilt:

2.2.1 Medium Access Control Layer (MAC-Schicht)

Diese liegt direkt an der Bitübertragungsschicht und definiert zum einen die Sende- und Empfangsadresse im Datenpaket sowie die Nutzung des Mediums.

2.2.2 Logical Link Control Layer (LLC-Schicht)

Die LLC-Schicht regelt den Zugriff der höheren Protokolle mit der der 2. Schicht, da mehrere Schichten die Informationen benötigen z.B. Anwendungsschicht die MAC-Adresse. Weiterhin steuert sie den Datenfluss (Anpassung der Schreib-/Lesegeschwindigkeit) um Kollisionen zu verhindern bzw. minimieren.

2.3 Network Layer (Vermittlungsschicht)

Die Vermittlungsschicht übernimmt Routingaufgaben, um eine Kommunikation über das lokale Netzwerk hinaus zu ermöglichen. Dies geschieht z.B. bei IP[1]-V4 über die jeweiligen Adressen und Subnetz Masken.

2.4 Transport Layer (Transportschicht)

In dieser Schicht befindet sich das Bindeglied zwischen den transportorientierten Schichten 1-3 und den anwendungsorientierten Schichten 5-7. Die Transportschicht sorgt für die richtige Sender/Empfänger-Vermittlung bei Daten von Programmen und einer Fragmentierung von Anwendungs- auf Bit Daten.

2.5 Session Layer (Sitzungsschicht)

Der Session Layer erstellt Sitzungen für die einzelnen Verbindungen. In der Praxis bedeutet dies, dass die Sitzungsschicht, auch bekannt als Kommunikationsschicht, für jede Kommunikation eine Sitzung aufbaut, überwacht und beendet. Zusätzlich werden bestimmte Fixpunkte bei der Kommunikation gesetzt, so dass bei einem Verbindungsabbruch nicht die gesamte Übertragung komplett neu begonnen werden muss.

2.6 Presentation Layer (Darstellungsschicht)

Diese Schicht wird benötigt, um Daten zwischen den Anwendungen und der Sitzungsschicht zu transferieren. Sie stellt Funktionen für die Codierung und Darstellung der übertragenen Informationen bereit. Sie kann externe Ressourcen wie beispielsweise einen Drucker einbinden und legt fest, in welchem Datenformat kommuniziert wird.

2.7 Application Layer (Anwendungsschicht)

Im 7. Layer des OSI-Referenzmodels werden die Daten aus der Darstellungsschicht an die jeweils kommunizierende Anwendung angepasst. Sie ist die direkte Kommunikationsschnittstelle mit dem Netzwerk. Ein Beispiel ist das klassische File Transfer Protokoll (FTP).

[1] Internet Protokoll

3 Der Physical Layer

Ethernet kommuniziert auf Bit Ebene über die erste Schicht, dem Physical Layer. Im Laufe der Zeit wurden in dieser Schicht Anpassungen vorgenommen, um eine höhere Geschwindigkeit zu erreichen. Wie in Abb. 2 zu sehen, wurden seit dem ersten Ethernet, über Koaxial (1/10MBit/s auf der linken Seite) bis Geschwindigkeiten $\geq$ 100MBit/s bei Twisted-Pair- oder Glasfaserkabeln, einige Schichten hinzugefügt, die als PHY-Schicht zusammengefasst werden. Beim ersten Ethernet gab es zusätzlich zum eigentlichen Übertragungsmedium ein Attachment User Interface (AUI), welches das Bindeglied zwischen Netzwerkkarte und Übertragungsmedium darstellte. Vorteil war, dass die MAC-Hardware (Schicht 2) an jede PHY Einheit für jedes Medium verwendet werden konnte ohne diese anpassen zu müssen. Vor dem PHY liegen zwei Zwischenschichten die das Routing und die Zuordnung zu den Datenpunkten bereitstellen.

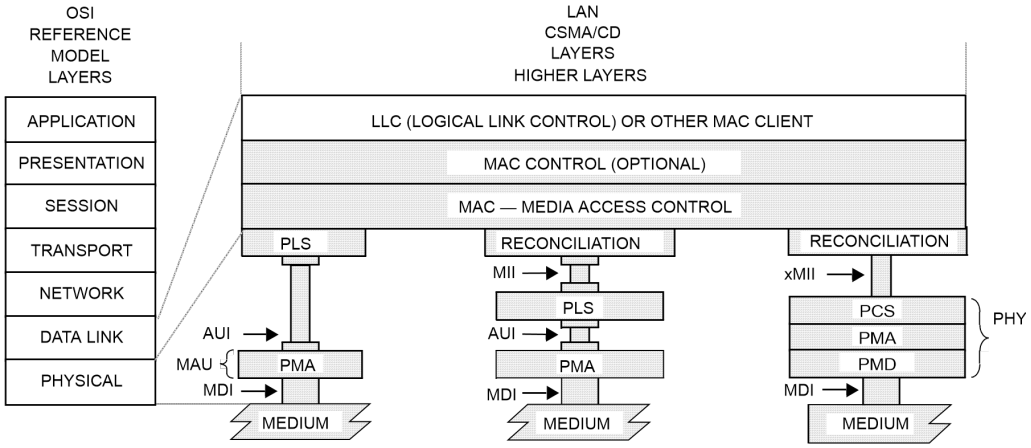

Abbildung 2: *(1MBit/s, 10MBit/s) $\leftrightarrow$ (10MBit/s) $\leftrightarrow \geq$ (100MBit/s)[Soc16]*

3.1 Reconcilation Sublayer (RS)

Diese Unterschicht verknüpft die MAC/LLC Schicht mit dem Medium Independent Interface (MII) und weißt die Signale den jeweiligen Diensten zu. Er sorgt für eine Anpassung der Daten an die angeschlossene PHY-Schicht

3.2 Medium Independent-Interface (MII)

Das Medium Independent Interface ist die Schnittstelle des PHY mit dem RS. Es wurde mit Fast Ethernet eingeführt, um die MAC-Schicht mit dem jeweiligen PHY zu verbinden. Das MII kann sowohl direkt über die Platine verbunden werden, als auch über einen 40poligen MII-Connector. Zur eigentlichen Kommunikation werden Transistor-Transistor-Logik (TTL)-Signale verwendet, damit eine Schnittstelle mit günstigen CMOS-Bausteinen realisiert werden kann. Es gibt für verschiedene Geschwindigkeiten, Einsatzbereiche und Übertragungsmedien unterschiedliche MII-Arten. Eine Übersicht der gebräuchlichsten[Neu05]:

- **MII** –> Standard Interface für die Datenraten 10/100MBit/s. unterschiedlichen Taktraten: 25MHz bei 100MBit/s und 2.5MHz bei 10MBit/s. Das MII überträgt in Sende-/ und Empfangsrichtung jeweils mit vier Bit, auch als Nibble bezeichnet.
- **Reduced Media-Independent Interface (RMII)** –> Um das MII für andere Einsatzzwecke wie z.B. FPGA Implementierung zu verkleinern wurden einige Signale vereinfacht oder entfernt. Im Gegensatz zum MII mit jeweils einem Takt für

Senden und Empfangen wird hier auf einen zentralen Takt zugegriffen, was eine Implementierung in Multi Port Geräten, wie einem Switch vereinfacht. Allerdings ist die Datenschnittstelle von vier auf zwei Bits verringert, so dass sich eine Verdopplung des Basistaktes auf 50MHz ergibt. Des Weiteren fällt das Signal für die Kollisionserkennung weg und die Signale für die Datenvalidität werden zu einem einzigen Signal zusammengelegt.

- **Gigabit Media-independent Interface (GMII)** –> Interface für Datenraten bis 1GBit/s. Hier wird mit einem Takt von 125MHz und einer Sende-/ und Empfangsbandbreite von jeweils acht Bit gearbeitet, um auf die Übertragungsgeschwindigkeit von 1000MBit/s zu kommen. Zusätzlich werden sechs Statusleitungen für die Verwaltung und Fehlererkennung genutzt. Zusammen mit den Leitungen für den Takt besitzt das GMII 24 Pins. Es ist Abwärtskompatibel und kann auch mit 10MBits/s übertragen.
- **Reduced Gigabit Media-Independent Interface (RGMII)** –> Das RGMII arbeitet mit 12 Pins. Um dies zu realisieren wurden die Datenleitungen halbiert und jeweils zwei Leitungen für Takt und Fehlererkennung genutzt. Um eine Anpassung des Taktes zu vermeiden, werden beim RGMII sowohl bei der fallenden als auch bei der steigenden Taktflanke Daten gesendet.

3.3 PHY

Die Zusammenfassung der folgenden Schichten wird als PHY bezeichnet.

3.3.1 Physical Coding Sublayer (PCS)

Der Physical Coding Sublayer hat folgende Aufgaben[Neu05]:

- Realisierung der Datenkodierung. Es werden je nach angestrebter Geschwindigkeit Manchester, 4B/5B oder 8B6T-Code in Sender-/ und Empfängerrichtung genutzt.
- Generierung der Carrier Sense/ Collision Detection Signale für die MAC-Schicht.
- Zuweisung von Sende-/Empfangsdaten, CS und CD zwischen dem MII und dem Physical Medium Attachment (PMA).

3.3.2 Physical Medium Attachment (PMA)

Das Physical Medium Attachment bereitet die vom PCS erhaltenen Daten für das angeschlossene Medium auf. Die drei Funktionsgruppen sind:

- Bestimmung/Synchronisation des Clock-Signals.
- Festlegung Übertragungsbegin pro Kanal.
- Kommunikation mit dem PCS.

Des Weiteren sind Funktionen für Carrier Sense und Link Integrity[2] implementiert.

3.3.3 Physical Medium Dependent Sublayer (PMD)

Die letzte Schicht des PHY ist der mediumbhängige Physical Medium Dependent Layer. Dieser Sublayer wird nur bei Kombinationen mit Fiber Distributed Data Interface (FDDI) genutzt und liefert dahingehende Dienste und Codierungen.

[2]Signal das prüft, ob physikalische Verbindung besteht

3.4 Medium Dependent Interface (MDI)

Dies ist der Physikalische Anschluss an das Übertragungsmedium wie z.B. Twisted-Pair-Kabel.

4 Topologien

Eine Übersicht der unterschiedlichen Verkabelungen und Hierarchien bei Ethernet. Die anfängliche Verkabelung bei Ethernet stellte der Bus da. Aufgrund der nachfolgend genannten Gründe hat sich dies mittlerweile Richtung Maschentopologien gewandelt.

4.1 Bus

Hierbei werden die Rechner an nur einem Kabel hintereinandergeschaltet und an den Enden durch einen Widerstand terminiert. Der Vorteil ist eine unaufwändige Verkabelung und ein kostengünstiger Aufbau. Durch die Verschaltung in Reihe kann jedoch keine Abhörsicherheit gewährleistet und kein hoher Datendurchsatz erreicht werden. Diese Verkabelung wird aus diesen Gründen in der Praxis bei Ethernet nicht mehr eingesetzt.

4.2 Stern

Bei dieser Topologie werden die Rechner an einen zentralen Verteiler angeschlossen (Hub oder Switch). Dadurch können im Gegensatz zur Busstruktur deutlich höhere Geschwindigkeiten realisiert sowie Abhörsicherheit gewährleistet werden. Die Anschaffungskosten sind deutlich höher und bei Ausfall des Verteilers wird das komplette Netz lahmgelegt. Der Stern ist die aktuell vorwiegende Verkabelungsart bzw. Topologie

4.3 Maschentopologie

Es werden alle Geräte miteinander verbunden, um eine sehr hohe Ausfallsicherheit zu erhalten. Bei Zusammenschluss aller Geräte spricht man von einer "vollständigen Maschenbildung". Bei Geräteausfall existieren immer noch alternative Verbindungen. Der Kostenaufwand sowie die Realisierungsdauer sind sehr hoch In der Praxis werden Mischtopologien zusammengeschaltet und an die jeweiligen Netz- sowie Nutzeranforderungen angepasst.

5 Kabel

IEEE 802.3 beinhaltet auch eine Glasfaser Komponente, die zwar vollständigkeitshalber erwähnt, aufgrund des Fokus auf Elektrische Übertragung jedoch nicht weiter vertieft werden soll. Ethernet ist seit seiner Einführung auf die höchstmögliche Kompatibilität mit den genutzten Übertragungsmedien entwickelt worden. Aus diesem Grund werden auf den nachfolgenden Seiten die elektrischen Datenleitungen beleuchtet.

5.1 Koaxialkabel RG-58/RG-8

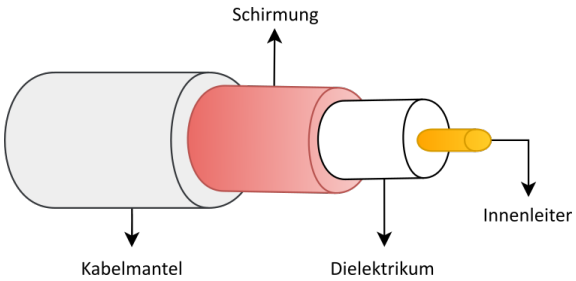

Abbildung 3: *Koaxialkabel*

In den Anfängen des Ethernet wurde hauptsächlich mit Koaxialkabeln[6] gearbeitet. Zum einen konnten die elektrischen Eigenschaften aufgrund der Leitungsgeometrie sehr genau berechnet werden. Zum anderen konnte auf ein Jahrzehnte altes Wissen zurück gegriffen werden, da Koaxialkabel seid ihrer Markteinführung 1934 ständig optimiert wurden. In der langsamstem Variante kam das Kabel Typ RG58 mit 5mm Außendurchmesser zum Einsatz, welches aus dem Militärbereich übernommen wurde. Der Nachfolger für eine höhere Geschwindigkeit ist das mit 10mm Außendurchmesser RG8 Kabel, auch als "Yellow Cable"bezeichnet.
Der Aufbau ist bei beiden Kabeln identisch. Unter einem Kabelmantel aus Kunststoff sitzt ein Außenleiter/Schirmung (meist eine Netzstruktur aus Kupfer). Der Zwischenraum zum Innenleiter ist das Dielektrikum, eine Isolationsmasse. Die Kabeldurchmesser sind beim Koaxialkabel Namensgebend. Thin-Ethernet bei 5mm-, und Thick-Ethernet bei 10 mm- Außendurchmesser. Da das 10mm-Kabel sich nicht problemlos von System zu System verlegen lässt, wurde mit einem externen Transceiver gearbeitet (MAU) der an die AUI der Netzwerkkarte geschaltet wurde. Aufgrund von besserer Erweiterbarkeit, Geschwindigkeit, Störanfälligkeit und Kostensenkung wurde auf Twisted-Pair-Kabel umgeschwenkt.

5.2 Twisted-Pair

Bei den gebräuchlichsten Ethernet-Kabeln gibt es je nach Einsatzbereich unterschiedliche Ausführungen. Es gibt 3 verschiedene Schirmungsmöglichkeiten[Kat15]:

- Adernpaarschirmung ohne Gesamtschirm
- Aderpaar ungeschirmt mit Gesamtschirm
- Adernpaarschirmung mit Gesamtschirm

Bei den eingesetzten Schirmungen gibt es folgende Ausführungen:

Schirmung mit:

- ...einem Drahtgeflecht
- ...einer Metallfolie
- ...einem Drahtgeflecht und Metallfolie

Abkürzung	Name	Gesamtschirmung	Adernpaarschirmung
ungeschirmte Twisted Pair (UTP)			
U/UTP	Unshielded Twisted Pair	keine	keine
S/UTP	Screened Unshielded Twisted Pair	Drahtgeflecht	keine
F/UTP	Screened Unshielded Twisted Pair	Folie	keine
SF/UTP	Screened Unshielded Twisted Pair	Folie und Drahtgeflecht	keine
geschirmte Twisted Pair mit Drahtgeflecht (STP)			
U/STP	Unscreened Shielded Twisted Pair	keine	Drahtgeflecht
S/STP	Screened Shielded Twisted Pair	Drahtgeflecht	Drahtgeflecht
F/STP	Screened Shielded Twisted Pair	Folie	Drahtgeflecht
SF/STP	Screened Shielded Twisted Pair	Folie und Drahtgeflecht	Drahtgeflecht
geschirmte Twisted Pair mit Folie (FTP)			
U/FTP	Unscreened Foiled Twisted Pair	keine	Folie
S/FTP	Screened Foiled Twisted Pair	Drahtgeflecht	Folie
F/FTP	Screened Foiled Twisted Pair	Folie	Folie
SF/FTP	Screened Foiled Twisted Pair	Folie und Drahtgeflecht	Folie

Tabelle 1: *Übersicht Kabelschirmungen*

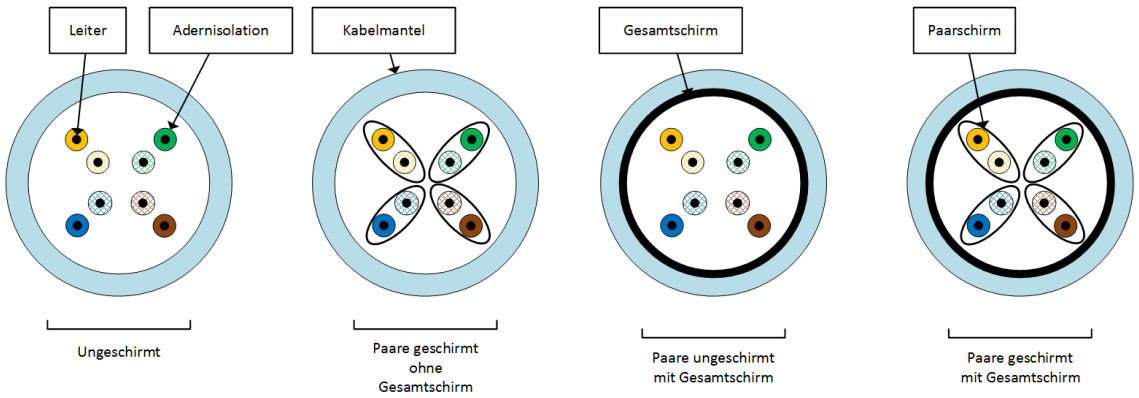

Abbildung 4: *Bauformen Twisted Pair Kabel*

6 Ethernet-Varianten

Die folgende Übersicht[Kat15] bezieht sich auf die gängigsten, in Unternehmen und Industrie am häufigsten vorkommenden Arten von Ethernet bzw. dessen Standards. Die elektrischen Eigenschaften sind immer abwärtskompatibel, z.B. kann man (ungeschirmte Twisted Pair) UTP-Kabel, die für einen Betrieb als 10GBase-T ausgelegt sind, auch für 100Base-T4 oder 1000Base-T verwenden.

$$\textbf{Beispiel Ethernet-Bezeichnung} = \underbrace{100}_{\mathbf{1}}\underbrace{BASE}_{\mathbf{2}}-\underbrace{TX}_{\mathbf{3}}$$

1. Die zu Beginn stehende Zahl gibt die Geschwindigkeit in MBit/s an. Folgt auf die Zahl der Buchstabe G handelt es sich um einen GBit/s Wert
2. Der Begriff BASE sagt aus, dass die gesamte Bandbreite benötigt wird
3. Zuletzt folgen nach einem Bindestrich weitere Angaben:
   - -2 –> maximale Segmentlänge [2]00m
   - -5 –> maximale Segmentlänge [5]00m
   - -T –> [T]wisted Pair CAT3
   - -TX –> [T]wisted Pair CAT5, [X]=4B/5B Codierung (für 100MBit)
   - -T2 –> [T]wisted Pair, [2] Doppel paarig
   - -T4 –> [T]wisted Pair, [4] Doppel paarig
   - -FX –> [F]ibre, [X]=4B/5B Codierung (für 100MBit)
   - -SX –> [S]hort-range multi-mode Fibre, [X]=4B/5B Codierung (für 100MBit)
   - -CX –> [C]opper, [X]=8B/10B Codierung (für GBit)
   - -LX –> [L]ong Wavelength, [X]=8B/10B Codierung (für GBit)
   - -CX4 –> [C]opper, [X]=8B/10B Codierung (für GBit), 4 Doppelpaare
   - -SR –> [S]hort-range multi-mode fibre, [R]=64B/66B Codierung
   - -LR –> [L]ong-range single- or multi-mode fibre, [R]=64B/66B Codierung

Die in der folgenden Kategorienübersichtstabelle aufgezeigten Kabel sind zwar über den Standard ISO/IEC 11801 vereinheitlicht, aber die jeweiligen Bezeichnungen nicht geschützt. Es entstehen dadurch missverständliche Bezeichnungen wie CAT5^{e},CAT5$_{e}$,CAT6E oder spezielle, vom Standard abweichende Herstellertitel. Des Weiteren ist darauf zu achten, auf welches Teil Bezug genommen wird (Kabel, Stecker, Buchse).

Bezeichnung	Name	Datenrate bit/s	Kabeltyp	Adernpaare bzw. Kanäle	ISO Kategorie	Maximale Übertragungslänge in m	Übertragungsfreq. od. Wellenlänge	IEEE 802.3 Norm
10Base-2	Thin-Ethernet	10M	Koax RR-58U	1		185		Clause 10
10Base-5	Thick-Ethernet	10M	Koax RG-8A/U	1		500		Clause 8
10Base-T		10M	UTP	2	CAT3	100	2x10MHz	Clause 14
10Base-FL		10M	LWL			100	850nm	Clause 18
100Base-T	Fast-Ethernet	100M	UTP	2	CAT5	100		Clause 25
100Base-TX	Fast-Ethernet	100M	UTP	2	CAT5	100	2x31,25MHz	Clause 24
100Base-T2	Fast-Ethernet	100M	UTP	2	CAT3			Clause 32
100Base-T4	Fast-Ethernet	100M	UTP	4	CAT3			Clause 23
100Base-FX	Fast-Ethernet	100M	LWL	1		400	1300nm	Clause 26
100Base-SX	Fast-Ethernet	100M	LWL	1			850nm	
100Base-SX	Fast-Ethernet	100M	LWL	1		10K	1300nm	Clause 58
1000Base-T	Gigabit-Ethernet	1G	UTP	4	CAT5	100	2x65,5MHz	Clause 40
1000Base-T	Gigabit-Ethernet	1G	UTP	4	CAT5e	100	2x65,5MHz	Clause 40
1000Base-CX	Gigabit-Ethernet	1G	STP	2		25		Clause 39
1000Base-SX	Gigabit-Ethernet	1G	LWL-Multimode			550	850nm	Clause 38
1000Base-LX	Gigabit-Ethernet	1G	LWL-Multimode			550	1310nm	Clause 38
10GBase-T	10GbE	10G	UTP	4	CAT5e	45	4x417MHz	
10GBase-T	10GbE	10G	STP	4	CAT5e	>45	4x417MHz	
10GBase-T	10GbE	10G	S/FTP	4	CAT6	100	4x417MHz	
10GBase-T	10GbE	10G	S/FTP	4	CAT6	100		an
10GBase-T	10GbE	10G	S/FTP	4	CAT6A	100		
10GBase-T	10GbE	10G	S/FTP	4	CAT7	100		
10GBase-CX4	10GbE	10G	Twinax-Kabel	2		15		
10GBase-FX	10GbE	10G	LWL			100		
10GBase-SX	10GbE	10G	LWL					
10GBase-SR	10GbE	10G	LWL-Multimode			300	850nm	
10GBase-LR	10GbE	10G	LWL-Singlemode			10K	1310nm	
40GBase-CR4	40GbE	40G	Twinax-Kabel	4		10		ba
40GBase-KR4	40GbE	40G	PCB-Leitung	4		1		ba
40GBase-SR4	40GbE	40G	LWL-Multimode	4		100		ba
40GBase-LR4	40GbE	40G	LWL-Monomode	4		10K		ba
100GBase-CR10	100GbE	100G	Twinax-Kabel	10		10		ba
100GBase-SR10	100GbE	100G	LWL-Multimode	10		100		ba
100GBase-LR10	100GbE	100G	LWL-Monomode	10		10K		ba
100GBase-ER4	100GbE	100G	LWL-Monomode	4		40K		ba

Tabelle 2: *Übersicht der bei Ethernet zum Einsatz kommende Verkabelung[Kat15]*

7 Codierung

In der Digitaltechnik legen die Spannungspegel das logische Signal fest. Bei einem Wert von 0V handelt es sich um eine logische 0 und bei 5V um eine 1. Je nach verwendeter Technologie (TTL, Complementary Metal-Oxide-Semiconductor (CMOS)) kann der Pegel einer Logischen 1 zwischen 5V, 3,3V, 2,5V und 1,8V variieren. Eine serielle Datenübertragung ist bei dieser Darstellung aufgrund der fehlenden Taktinformation, die einen klaren Pegelwechsel anzeigt, nicht möglich. Hier kommt die Codierung der Informationen zwischen Sender und Empfänger zum Einsatz. Damit der jeweilige Sender/Empfänger die Daten zweifelsfrei zuordnen und verarbeiten kann, werden bei Ethernet folgende Varianten genutzt:

- Manchester
- NRZ-I
- 4B/5B
- 8B/10B
- MLT-3
- 8B/6T

7.1 Manchester Codierung

Die von G.E. Thomas an der Universität von Manchester entwickelte Codierung, wird bei Ethernet in einer abgewandelten Form genutzt. Der logische Wert 0 wird durch eine fallende Spannungsflanke beschrieben, der logische Wert 1 durch eine steigende (klassische NRZ-Codierung). Im Ursprungscode war dies invertiert und wurde an die Digitaltechnik angepasst. Der Flankenwechsel codiert das Bit. Aus Sicht der Digitaltechnik wird aus dem Systemtakt und den Daten über eine XOR-Verknüpfung das Ausgangssignal generiert. Bei Netzwerken mit einer Geschwindigkeit >100MBit/s kommt diese Kodierung nicht mehr zum Einsatz, da bei jedem übertragenen Bit zweimal der Zustand wechselt, wodurch die doppelte Bandbreite belegt wird, welche eine wirtschaftliche und technische Ressourcenverschwendung darstellt.

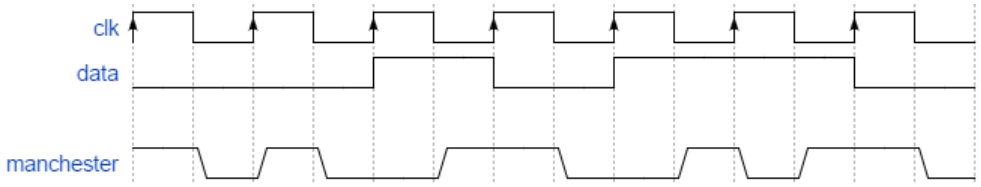

Abbildung 5: *Manchester Codierung*

7.2 Non Return to Zero-Inverted-Codierung

Non Return to Zero Inverted (NRZ-I) ist eine der Einfachsten Codierungen, die bei Ethernet, USB sowie bei der Datenaufzeichnung bei HDD und CD-ROM genutzt werden. Die beiden binären Signale werden durch den Spannungspegel und dessen Zustandswechsel bestimmt. Praktisch bedeutet dies, dass die Codierung einem Bit-Wert den aktuellen Leitungszustand zuordnet und dem anderen den Wert invertiert. Hieraus ergibt sich eine Polaritätsfreiheit, d.h. eine Verpolung der Übertragungsleitungen ändert nicht die Bitfolge. Für lange Bitfolgen von 0 oder 1 sollte dieses Codierung nicht verwendet werden, da es sonst Synchronisationsprobleme geben kann.

7.3 4B/5B Codierung

Bei der 4B/5B Codierung werden die Daten in 4 Bit Einheiten zusammenfasst und anschließend in 5 Bit kodiert, in so genannte Symbole. Durch die Codierung werden Bitkombination vermieden, in denen mehr als 3-mal hintereinander der Wert 0 vorkommt, um die Sender-/Empfängercodierung zu erleichtern.

Bezeichnung	4B-Muster	5B-Muster	Funktion o. Signal
0	0000	11110	Hex data 0
1	0001	01001	Hex data 1
2	0010	10100	Hex data 2
3	0011	10101	Hex data 3
4	0100	01010	Hex data 4
5	0101	01011	Hex data 5
6	0110	01110	Hex data 6
7	0111	01111	Hex data 7
8	1000	10010	Hex data 8
9	1001	10011	Hex data 9
A	1010	10110	Hex data A
B	1011	10111	Hex data B
C	1100	11010	Hex data C
D	1101	11011	Hex data D
E	1110	11100	Hex data E
F	1111	11101	Hex data F
Q	—	00000	Quiet (Signalverlust)
I	—	11111	Idle (Pause)
J	—	11000	Start #1
K	—	10001	Start #2
T	—	01101	End
R	—	00111	Reset
S	—	11001	Set
H	—	00100	Halt

Tabelle 3: *Übersicht 4B/5B Code*

7.4 8B/10B Codierung

Für Gigabit Ethernet wird die 8B/10B Codierung genutzt. Hier werden ähnlich wie bei der 4B/5B Codierung die 8 Bit Blöcke in 10 Bit gewandelt. Alle für die Datenübertragung gültigen Bitkombinationen sind so aufgebaut, das fünf die maximale Länge gleicher Bits ist. Spätestens danach wird ein Pegelwechsel erfolgen um eine Gleichstromfreiheit zu garantieren. Des Weiteren weisen diese Kombinationen mindestens 3-mal einen Zustandswechsel von 1 auf 0 oder umgekehrt auf, um auf der Empfängerseite eine mögliche Taktrückgewinnung zu ermöglichen.

7.5 8B/6T Codierung

Bei dieser Codierung handelt es sich um 8 Bit word to 6 Ternary[3] Symbols. Hierbei werden 8 Bit Datenwörter in 6 dreiwertigen Symbolen dargestellt. Die elektrischen Pegel für +, 0 und - werden hier genutzt, was direkt übertragen werden kann.

7.6 MLT-3-Codierung

Bei der Multi-Level-Transmission, oder MLT-3 handelt es sich auch wie bei der 8B/6T-Codierung um ein Terniärverfahren. Es wird empfängerseitig ausgewertet und ändert den Wert nur bei einer Pegeländerung zur logischen 1.

[3] Ein Multipegelverfahren

8 Übersicht des Ethernet Rahmens

Ethernet ist ein Paket vermittelndes Datenprotokoll. Das Daten- bzw. Ethernet Paket besteht aus dem sogenannten Frame und zwei zusätzlichen Datenfeldern, dem Start Frame Delimiter (SFD) und der Präambel. Diese werden vor das Frame gesetzt, und werden für die Synchronisation von Sender u. Empfänger und die Taktrückgewinnung verwendet. Seit Einführung von Ethernet wurde das Datenpaket 3-mal überarbeitet und befindet sich aktuell in seiner 4. Version, dem Tagged MAC-Frame. Der in dieser Thesis entworfene Generator erstellt ein Paket der ersten Version, in dem die Virtual LAN (VLAN)-Funktionalität nicht implementiert ist. Vollständigkeitshalber wird das VLAN-TAG aber trotzdem erwähnt. Die nachfolgende Grafik zeigt die grundsätzliche Form des Datenrahmens und die Sendeprioritäten.

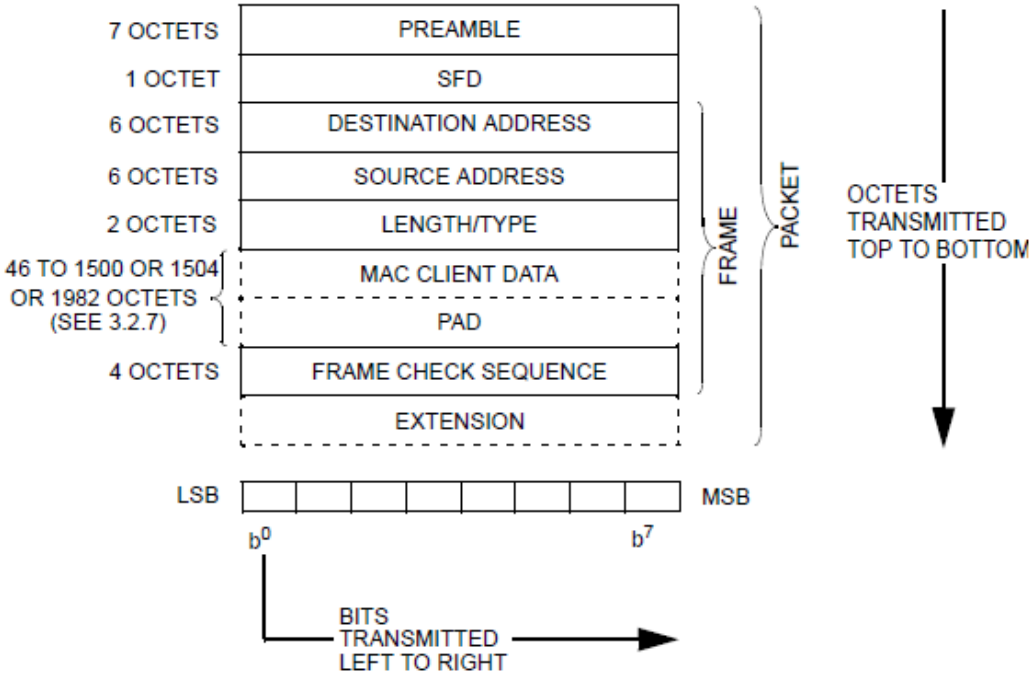

Abbildung 6: *Übersicht Ethernet Paket bzw. Frame[Soc16]*

8.1 Präambel

Das erste Datenfeld im Ethernet Paket ist die Präambel. Sie ist eine Rechteckschwingung mit einer 10101010 Bitfolge, die der Taktsynchronisation dient. Diese Schwingung wird 7-mal wiederholt.

8.2 Start Frame Delimiter

Der SFD folgt auf die Präambel und ist eine feste Bitfolge, welche den Start des eigentlichen Ethernet-Frames kennzeichnet. Es besteht aus einem Oktett mit der Bitfolge 10101011. Zusammen mit der Präambel wird hiermit der Takt vorgegeben und eine frühzeitige Kollisionserkennung ermöglicht.

8.3 MAC-Adresse

Bei der MAC-Adresse handelt es sich um eine eindeutige und, je nach Art, weltweit einzigartige Adresse. Abgesehen von den ersten zwei Bits können die ersten 3 Bytes einer MAC-Adresse bei der IEEE gegen eine Gebühr fest reserviert werden. Diese Reservierung wird von Hardwareherstellern genutzt, um der Verpflichtung gerecht zu werden, in ihren Produkten weltweit einmalige Adressen zu verwenden. Diese eindeutigen Kennungen, auch Organizationally Unique Identifier (OUI) genannt, sind bei der IEEE verzeichnet und einsehbar [IEE]. Da es möglich ist, einen Adressbereich zu erwerben ohne namentlich genannt zu werden, existieren auch private Einträge in dieser Liste. Insgesamt besteht die Adresse aus 6 Bytes (48 Bits), die meist hexadezimal angegeben und entweder mit Bindestrichen oder Doppelpunkten, in Oktetten gruppiert, dargestellt wird.
Beispiel: **D0-17-C2-18-E3-C6**. Diese Darstellungsform wird auch als kanonisches Format oder Little Endian[4] bezeichnet. Bei der Datenübertragung wird von links nach rechts übertragen mit dem Least Significant Bit (LSB) zuerst.
In der gezeigten MAC-Adresse wäre die übertragene Bitfolge:

Binär	0000	1011	1110	1000	0100	0011	0001	1000	1100	0111	0110	0011
Hex	0	B	E	8	4	3	1	8	C	7	6	3

Tabelle 4: *Beispiel MAC-Bitfolge*

Die beiden niederwertigen Bits des ersten Bytes haben bestimmte Funktionen:

- **LSB –> Individual/Group Bit**
  Bei einer logischen 0 handelt es sich um eine weltweit einmalig vergebene Adresse (Unicast Adresse). Ist dieser Wert auf 1 gesetzt stellt die Adresse eine Broad-, oder Multicastadresse da. Multicast spricht nur bestimmte, Broadcast alle im Netzwerk vorhandenen Stationen an.

- **Bit 1 –> (G/L-Bit)**
  Das Bit sagt aus, ob es sich bei der MAC-Adresse entweder um eine Global Administrated Address (GAA), oder eine Local Administrated Address (LAA) handelt. Bei GAA werden von den jeweiligen Herstellern weltweit einmalige Adressen vergeben. Bei LAA können die Adresse vom jeweiligen Nutzer (Administrator, Programmierer etc.) zur Verwendung im eigenen Netzwerk vergeben werden. Diese Option kann nur lokal von Personen angewendet werden, die genaue Kenntnis über den Aufbau Ihres Netzwerkes besitzen. Bei einer Nutzung im Globalen Adressbereich können Fehler wie unvollständige Übertragungen wegen Adresskollisionen oder Verbindungsabbrüchen der beteiligten Teilnehmer auftreten.

Es gibt eine spezielle MAC-Adresse, die sogenannte Broadcast-Adresse. Diese Adresse enthält nur den hexadezimalen Wert F (FF-FF-FF-FF-FF-FF). Pakete an dieses Ziel werden von allen Endstellen im Netzwerk verarbeitet.

[4] auch klein ender, ähnlich der Deutschen Datumsformatierung DD.MM.YYYY

8.4 Virtual LAN-Tag

Das VLAN bietet die Möglichkeit bestimmte Ports von bestimmen Switches im Netzwerk virtuell zusammen zu fassen. Dadurch können Systeme zu einer eigenständigen Gruppe zusammengelegt werden. Vorteile liegen bei einer besseren Administrierbarkeit und einer höheren Sicherheit durch Separierung von sensiblen Arbeitsgruppen.

8.5 Ethernet Typ/Länge

Das Ethernet Typ-Feld beinhaltet entweder Informationen, welche Protokolle auf der nächst höheren OSI-Schicht auswerten, oder bei Hex-Werten zwischen 0x0000-0x05DC, die Länge der Nutzdaten. Die Protokollinformation wird mit einem ab dem Hex-Wert 0x0600 beginnenden Code angegeben.

Ethertype-Feld	Protokoll
0x0000-0x05DC	IEE802.3 Length Field
0x0600	XEROX IDP
0x0800	IP Internet Protocol, Version 4 (IPv4)
0x0805	X.25
0x0806	Address Resolution Protocol (ARP)
0x0835	Reverse Address Resolution Protocol (RARP)
0x0842	Wake on LAN (WoL)
0x809B	AppleTalk (EtherTalk)
0x80F3	Appletalk Address Resolution Protocol (AARP)
0x8100	VLAN Tag (VLAN)
0x8137	Novell IPX (alt)
0x8138	Novell
0x86DD	IP Internet Protocol, Version 6 (IPv6)
0x8847	MPLS Unicast
0x8848	MPLS Multicast
0x8863	PPPoE Discovery
0x8864	PPPoE Session
0x8870	Jumbo Frames (veraltet)
0x888E	802.1X Port Access Entity
0x8892	Echtzeit-Ethernet PROFINET
0x88A2	ATA over Ethernet Coraid AoE
0x88A4	Echtzeit-Ethernet EtherCAT
0x88AB	Echtzeit-Ethernet Ethernet POWERLINK
0x88CC	Link Layer Discovery Protocol LLDP

Tabelle 5: *Übersicht einiger nennenswerten Ethernet-Typen[Rec08]*

8.6 Daten

Das Datenfeld bzw. die Nutzerdaten haben eine Mindestgröße von 46 Bytes und eine Maximalgröße von 1500 Bytes. Auch in diesem Feld beginnt die Übertragung mit dem LSB.

8.7 Padding

Padding-Bits (PAD) werden für die korrekte Ausführung des Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Protokolls benötigt. Mit Hilfe des PAD-Feldes wird das Ethernet Paket auf eine Mindestgröße von 64 Byte (46 Byte Daten, 12 Byte Adressen, 2 Byte Länge und 4 Byte Frame Check Sequence (FCS)) gebracht, wodurch die Detektion von Kollisionen verschiedener Busteilnehmer garantiert wird. Bei Unterschreitung der Mindestgröße können Probleme mit der Kollisionserkennung auf dem Datenbus auftreten. Die Präambel und der SFD werden bei der Berechnung der Framelänge nicht einbezogen, während der VLAN-Tag berücksichtigt wird.

8.8 Frame Check Sequence

Die FCS ist der letzte Teil des Ethernet Pakets und dient der Sicherung des Datensatzes gegen Verfälschung oder Fehler bei der Übertragung. Dies geschieht mit Hilfe des Cyclic Redundancy Check (CRC)-Verfahrens. Die CRC-Checksumme ist ein durch Polynomdivision berechneter Datenwert, welcher jedem zu übertragendem Datenpaket, angefügt wird. Im Falle von Ethernet ist die CRC-Checksumme ein 32 Bit Wert, der aus den MAC-Adressen für -Sender und Empfänger, .Typ, -Daten und PAD-Feld generiert wird. Gebildet wird dieser Wert über das folgende Generator Polynom:
$\mathbf{x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1}$.

Die Funktionsweise wird im Kapitel Komponente: crc32 näher beschrieben.

9 Übersicht des Rahmengenerators

Der Rahmengenerator besteht aus 6 Komponenten und ist in der Lage Daten Anhand des IEEE 802.3-Standards zu formatieren und auszugeben.

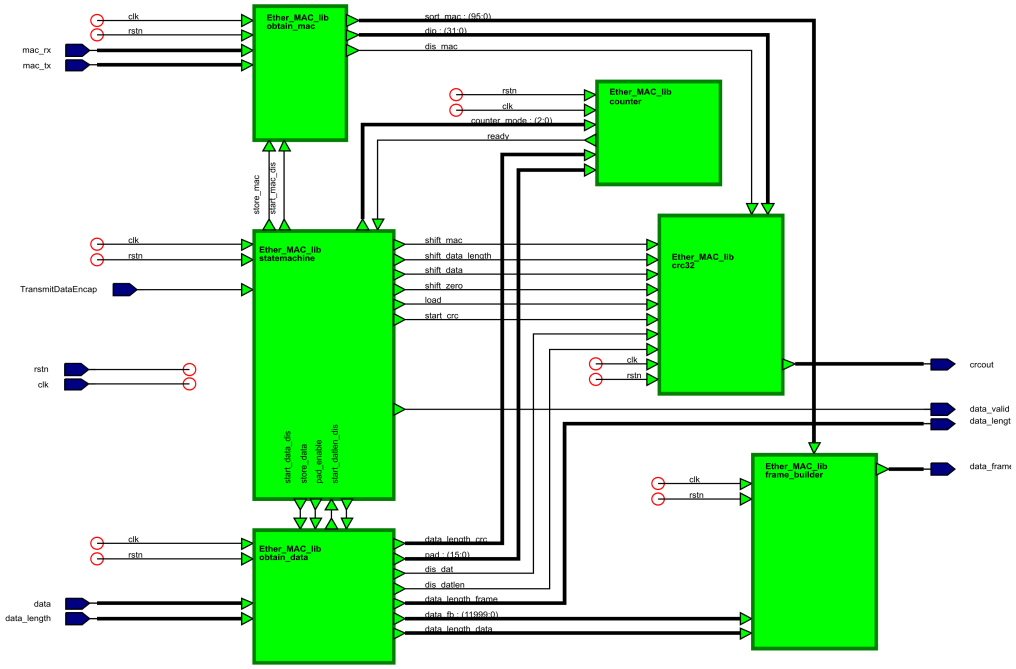

Abbildung 7: *Gesamtübersicht des Rahmengenerators und seiner Komponenten*

Die zu sendenden Daten werden von **obtain_data** und **obtain_mac** übernommen und verarbeitet. Zusätzlich erhält die zentrale Komponente **statemachine** ein Signal zum Start des Funktionsablaufs. Die aufbereiteten Daten werden, gesteuert durch die **statemachine**, **counter** und **crc32** zur Auswertung übergeben. Final werden die Nutzdaten von **obtain_data** und **obtain_mac** durch **frame_builder** in der richtigen Reihenfolge ausgegeben. **crc32** gibt die errechnete Prüfsumme aus.
Die zu sendenden Daten müssen bei der Übergabe an den Rahmengenerator der folgenden Formatierung folgen, damit eine fehlerfreie Funktion der CRC-Komponente gewährleistet werden kann.

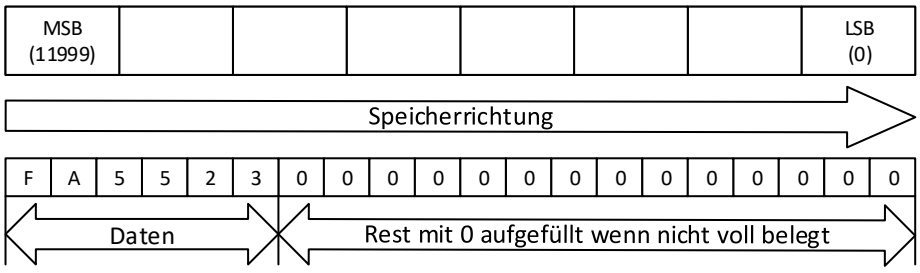

Abbildung 8: *Übersicht des Aufbaus der Daten*

9.1 Signalübersicht

Der Rahmengenerator besitzt 7 Eingänge, 4 Ausgänge und 23 interne Steuersignale. Nachfolgend wird eine kurze Übersicht der Ein-/Ausgänge gegeben. Die internen Signale werden bei Bedarf in den jeweiligen Komponenten näher erörtert.

INPUT:

- **mac_rx**
  Beinhaltet die 6 Byte große MAC-Adresse des Empfängers.
- **mac_tx**
  Enthält die 6 Byte großeMAC-Adresse des Senders.
- **clk**
  Systemtakt mit 50MHz für den Betrieb mit Reduced Medium Independent Interface (RMII).
- **rstn**
  Entspricht dem Reset-Signal. Der Reset wird ausgeführt, wenn das Signal den Wert '0' hat.
- **TransmitDataEncap**
  Steuersignal das dem Rahmengenerator anzeigt, dass Daten anliegen, die gesendet werden soll.
- **data**
  Daten variabler Länge von 1 bis 1500 Byte, je nach Übertragungsprotokoll und Inhalt.
- **data_length**
  Größe der anliegenden Daten in Bytes.

OUTPUT:

- **crc_out**
  Beinhaltet die 4 Byte große Prüfsumme, d.h. das FCS.
- **data_length_frame** Summer aller Datengrößen außer FCS für die RMII-Komponente.
- **data_frame_out**
  Gesamter Datensatz der Präambel bis zu dem letzten Byte der Daten. Der komplette Rahmen wird erst durch die RMII-Komponente mit dem abschließenden CRC-Feld zusammengeführt.
- **data_valid**
  Signal zur Anzeige dass der Rahmenbildungsprozess abgeschlossen ist.

9.2 Komponente: obtain_data

Die Komponente obtain_data speichert die anliegenden Daten und bereitet sie durch Umsortierung der Bitfolge auf die Versendung vor. Es existieren zwei asynchrone Prozesse[5] und ein synchroner Prozess. Der synchrone reagiert auf 3 Steuersignale: **store_data,** **start_data_dis** und **start_datlen_dis**.

```
data_check: process (clk,rstn)
begin
if (rstn = '0') then
    s_data <= (others => '0');
    s_data_length <= (others => '0');
  elsif (rising_edge(clk)) then
    if (store_data = '1') then
     s_data_length <= f_endian_change_length(data_length);
     s_data <= f_endian_change_data(data);
    elsif (start_data_dis = '1') then
      s_data (11999 DOWNTO 0) <= s_data(11998 DOWNTO 0) & s_data(11999);
    elsif (start_datlen_dis = '1') then
      s_data_length <= s_data_length(14 DOWNTO 0) & s_data_length (15);
    else s_data <=s_data;
  end if;
end if;
```

Listing 1: synchroner Prozess von obtain_data

store_data sorgt dafür, dass die Eingangsdaten welche an **data** und **data_length** anliegen mit umgedrehter Bitreihenfolge abgespeichert werden, um die byteweise **LSB first** Versendung zu vereinfachen. Dementsprechend wird in jedem Byte die Bit Reihenfolge umgekehrt und in den Signalen **s_data** und **s_data_length** abgelegt. Die Umsortierung der Bits geschieht dabei an Hand der in der Funktion **f_endian_change_length** festgelegten Methodik, welche in Kapitel 3.7 noch weiter beschrieben wird.

Wenn das Signal **start_data_dis** gesetzt wird, beginnt die Komponente mit einer bitweise Schiebeoperation des Inhaltes von **s_data** die, solange das Signal gesetzt bleibt, mit jeder steigenden Flanke des Taktsignals ausgeführt wird. Dabei werden die umsortierten Daten nach links geschoben und das hochwertigste Bit des Singals dem CRC Block zur Verfügung gestellt.
Die letzte Funktion **start_datlen_dis** startet die Schiebeoperation der Bits des **s_data_length**-Signals, welche die Information über die Länge des Datenfelds tragen. Auch in diesem Fall werden die Bits des Signals nach links geschoben und das hochwertigste Bit des Signals an den CRC Block weitergeleitet. Die beiden asynchrone Prozesse tragen den Namen **count_pad** und **count_data**

```
count_pad: process (data_length)
begin
  if (data_length > x"002D") then
   pad <= (others => '0');
   pad_enable <= '0';
  else
   pad <= x"002E" - data_length;
   pad_enable <= '1';
  end if;
end process;
```

Listing 2: Asynchroner Prozess von obtain_data: count_pad

count_pad wertet das Signal **data_length** aus und ermittelt die Anzahl der benötigten Padding-Bytes falls die zu versendenden Daten weniger als 46 Byte umfassen. Zusätzlich wird das für die Komponente **statemachine** benötigte Signal **pad_enable** gesetzt. Ist die Datenmenge größer als 46 Byte wird der Zähler und das **pad_enable** Signal nicht gesetzt.

[5] ohne Abhängigkeit eines clk-Signals, reagiert auf Änderung von **data_length**

```
count_data: process (data_length)
begin
  if (data_length > x"002D") then
   data_length_frame  <= data_length + c_framebase;
  else
   data_length_frame <= x"0044";
  end if;
end process;
```

Listing 3: Asynchroner Prozess von obtain_data: count_data

Der Prozess **count_data** dient dazu, der angeschlossenen RMII-Steuereinheit die Größe der Gesamtdaten (Präamble, SFD, MAC(RX und TX), Data Length und Data) mitzuteilen. Die RMII Steuereinheit kann an Hand der Signalinformation die genaue Position der CRC-Prüfsumme im Datenstrom befinden. Diese festen Rahmenbestandteile haben immer eine Größe von 22 Byte. Dieser Wert ist im Signal **c_framebase** hinterlegt. Es wird geprüft ob die Datenlänge unter- oder oberhalb der Mindestgröße von 46 Byte[6] liegt. Ist der Wert größer, wird die Datenlänge mit der Größe des fixen Datenteils summiert. Ist der Wert kleiner, wird die Mindestgröße von 46 Byte zusammen mit dem fixen Datenteil summiert, was dem Hex Wert 0044 entspricht. Hierdurch kann aus dem Wert der Datenlänge immer eine Gesamtgröße errechnet werden, die von der RMII-Schnittstelle für die Positionierung der FCS ausgewertet wird.

[6]Entspricht dem Hexadezimalwert x"002d"

9.3 Komponente: obtain_mac

Ähnlich wie in der Komponente **obtain_data** werden hier die angelegten MAC-Adressen gespeichert. Dieser Prozess reagiert auf 2 Steuersignale: **store_mac** und **start_mac_dis**.

```
process(clk, rstn)
begin
if (rstn = '0') then
  s_mac <= (others => '0');
elsif (rising_edge(clk)) then
 if (store_mac = '1') then
  s_mac  <= f_endian_change_mac(mac_rx(47 DOWNTO 0)) & f_endian_change_mac(mac_tx(47 DOWNTO 0));
 elsif (start_mac_dis = '1') then
  s_mac (63 DOWNTO 0) <= s_mac (62 DOWNTO 0) & s_mac (63);
 end if;
end if;
```

Listing 4: Übersicht Prozess obtain_mac

Durch setzen des **store_mac** Steuersignals werden die Adressen mit Hilfe des Konkatinierungsoperators & miteinander verknüpft und in **s_mac** gespeichert. Die Adressbits werden genau wie die Sendedaten mit umgedrehter Bitreihenfolge gespeichert. Für die Anpassung der Bitreihenfolge wird die Funktion **f_endian_change_mac** genutzt, welche näher in Kapitel 9.8 beschrieben wird. Wenn das Signal **start_mac_dis** gesetzt wird, beginnt eine bitweise Schiebeoperation nach links, wobei auch in diesem Fall das hochwertigste Bit an das CRC-Register geführt wird.

9.4 Komponente: counter

Für alle Zähloperationen bei der Erstellung der CRC-Summe wurde die Komponente **counter** erstellt. Abgesehen von den Frame Bestandteilen mit fixer Datengröße (MAC-Adressen, CRC-Summe, Datenlänge) existieren auch Datenfelder mit variabler Größe. Dabei handelt es sich um das eigentliche Daten- sowie das Paddingfeld, welches bei Unterschreitung der Datenmindestgröße eingefügt werden muss. Die Komponente besteht aus je einem synchronen und asynchronen Prozess.

```
process (clk,rstn)

begin
if (rstn = '0') then
        s_counter <= 0;
elsif (rising_edge(clk)) then
    case counter_mode is
        when "001" => s_counter <= 63;
        when "010" => s_counter <= 15;
        when "100" => s_counter <= (to_integer(unsigned(data_length_crc))*8)-1;
        when "101" => s_counter <= (to_integer(unsigned(pad))*8)-1;
        when "110" => s_counter <= s_counter - 1;
        when "111" => s_counter <= 31;
        when others => s_counter <= s_counter;
    end case;
end if;
```

Listing 5: Synchroner Prozess von counter

Die Zählweite wird über das Signal **counter_mode** bestimmt. Es existieren folgende Zählmöglichkeiten:

- **mode:"001"**
  Feste Zählweite von 63 für die Schiebeoperation der MAC-Adressen.
- **mode:"010"**
  Feste Zählweite von 15 für die Schiebeoperation des Datenlängenfeldes.
- **mode:"100"**
  Hierbei wird das Eingangssignal **data_length_crc** mit Hilfe der Funktion $(to\_integer(unsigned(data\_length\_crc)) * 8) - 1$ von einem hexadezimalen Wert in eine Dezimalzahl gewandelt. Das Signal bestimmt die Anzahl der zur versendenden Daten-Bytes. Die Anzahl der Bits berechnet sich daher aus der Multiplikation mit 8. Für eine wertgenaue Zuordnung wird zuletzt noch 1 abgezogen.
- **mode:"101"**
  Zählweite für die Schiebeoperation der Padding Bits.
- **mode:"110"**
  Dekrementierung des Zählerstandes mit jeder steigenden Flanke des Taktsignals.
- **mode:"111"**
  Zur abschließenden CRC Generierung wird nach der letzten Daten- oder Paddingschiebeaktion 32 mal eine logische 0 bitweise in das Register **s_crcdata** geschoben.

Bei allen anderen Modi behält der Zähler seinen Wert und findet keine Änderung statt.

```
process (s_counter)
begin
if (s_counter = 1) then
   ready <= '1';
else
   ready <= '0';
end if;
```

Listing 6: asynchroner Prozess von counter

Der asynchrone Prozess soll der Komponente: statemachine mit dem Signal **ready** anzeigen, dass der Zähler am Ende des Zählzyklus angelangt und bereit ist, einen neuen Funktionsmodus von der Komponente: statemachine zu erhalten. Da die Übernahme eines Wertes bei der statemachine einen Takt dauert, wird das Signal **ready** bereits bei 1 gesetzt.

9.5 Komponente: crc32

Eine der wichtigsten Komponenten stellt das Modul CRC da. Hier werden die zu prüfenden Daten anhand des im Kapitel 8.8 auf Seite 19 erwähnten Polynoms verarbeitet und nach Abschluss der Rahmenbildung bzw. Erreichen des Zustandes **data_valid = '1'** der nächsten Komponente zur Verfügung gestellt.

3.2.9 Frame Check Sequence (FCS) field

A cyclic redundancy check (CRC) is used by the transmit and receive algorithms to generate a CRC value for the FCS field. The FCS field contains a 4-octet (32-bit) CRC value. This value is computed as a function of the contents of the protected fields of the MAC frame: the Destination Address, Source Address, Length/ Type field, MAC Client Data, and Pad (that is, all fields except FCS). The encoding is defined by the following generating polynomial.

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

Mathematically, the CRC value corresponding to a given MAC frame is defined by the following procedure:

a) The first 32 bits of the frame are complemented.
b) The n bits of the protected fields are then considered to be the coefficients of a polynomial $M(x)$ of degree $n - 1$. (The first bit of the Destination Address field corresponds to the $x^{(n-1)}$ term and the last bit of the MAC Client Data field (or Pad field if present) corresponds to the x^0 term.)
c) $M(x)$ is multiplied by x^{32} and divided by $G(x)$, producing a remainder $R(x)$ of degree ≤ 31.
d) The coefficients of $R(x)$ are considered to be a 32-bit sequence.
e) The bit sequence is complemented and the result is the CRC.

The 32 bits of the CRC value are placed in the FCS field so that the x^{31} term is the left-most bit of the first octet, and the x^0 term is the right most bit of the last octet. (The bits of the CRC are thus transmitted in the order $x^{31}, x^{30}, \ldots, x^1, x^0$.) See Hammond, et al. [B36].

Abbildung 9: *Funktionsbeschreibung für die CRC-Summenbrechnung nach der IEEE-Norm[Soc16]*

Anhand Abbildung 9 wird nachfolgend die CRC-Prüfsumme generiert. Zu einem besseren Verständnis wird im Laufe der Funktionsbeschreibung darauf Bezug genommen.

```
process (clk, rstn)
begin
if (rstn = '0') then
        s_crcdata <= (others => '0');
elsif rising_edge(clk) then
  if load = '1' then
   s_crcdata <= not dip;
  elsif start_crc = '1' then
    if s_crcdata(31)= '1' then
      s_crcdata <= ((s_crcdata (30 downto 0) & s_crc_bit) xor c_crcpoly (31 downto 0)) ;
    else
      s_crcdata <= s_crcdata (30 downto 0) & s_crc_bit;
    end if;
  end if;
end if;
```

Listing 7: synchroner Prozess des CRC's

Der Prozess reagiert auf das **load** Signal. Ist das Signal **load** gesetzt, wird der invertierte Wert des Signals **dip** abgespeichert und in das Signal **s_crcdata** geschrieben. Dies entspricht den Punkten a) und b)[7] der CRC-Übersicht in Abbildung 9. Um die Hauptfunktion des CRC's zu starten wird das Signal **start_crc** gesetzt, wobei die CRC Berechnung so lange weiter läuft, wie das Signal **start_crc** gesetzt bleibt. Die Funktion des CRC Blocks hängt dabei von der Wertigkeit des Most Significant Bit (MSB) des Signals **s_crcdata** ab. Dementsprechend gibt es zwei Fälle:

- **s_crcdata(31) = '1'**
  Das in **s_crcdata** (Bit 30 bis Bit 0) gespeicherte Bitmuster wird mit dem Wert des nächsten am Signal **s_crc_bit** anliegenden Bit über den Konkatinierungsoperator & verknüpft, was im Prinzip einer Schiebeoperation nach links entspricht, bei der das hochwertigste Bit des Schieberegisters verloren geht, und gleichzeitig wird eine XOR-Operation mit dem CRC-Polynom durchgeführt. Dieser Schritt entspricht den Punkten c) und d) der CRC-Übersicht in Abbildung 9

- **s_crcdata(31) = '0'**
  In diesem Fall wird nur die Schiebeoperation, aber kein XOR durchgeführt.

Der Wert wird außerhalb des Prozesses invertiert ausgegeben, was Punkt e) der CRC-Übersicht in Abbildung 9 entspricht

```
crcout <= not s_crcdata;
```

Listing 8: Output des CRC's

Der Inhalt des Signals **s_crc_bit** ist variabel und wird von einem asynchronen Prozess gesteuert.

```
crc_state : process (shift_data, shift_zero, shift_mac, dis_mac, dis_dat, shift_data_length,
    dis_datlen)
begin
if shift_data = '1' then
  s_crc_bit <= dis_dat;
elsif shift_data_length = '1' then
  s_crc_bit <= dis_datlen;
elsif shift_mac = '1' then
  s_crc_bit <= dis_mac;
elsif shift_zero = '1' then
  s_crc_bit <= '0';
else
  s_crc_bit <= '0';
end if;
```

Listing 9: asynchroner Prozess des CRC's

Je nachdem welches Signal gesetzt wird, nimmt **s_crc_bit** den Wert von dem nächst anliegenden Datenbit, Datenlängenbit oder Adressbit an. Zu diesem Zweck werden die Steuersignale **data**, **data_length**, **shift_mac** bzw. **shift_zero** ausgewertet. Mit dem Signal **shift_zero** kann auch eine logische Null an den den Eingang des CRC Registers geschrieben werden, wie es zum Abschluss der CRC Rechnung bei den letzten 32 Bit der Fall ist.

[7] Die Daten werden direkt so gespeichert

Beispiel einer Polynomen Division bis zum 1. XOR:

```
  E    F   0    D    2    9    F    2    9    B     Datenwert HEX
1110 1111 0000 1101 0010 1001 1111 0010 1001 1011   Datenwert Binär
                                                    (32 Bit werden
                                                    direkt übernommen)

1111 0111 1011 0000 1001 0100 0100 1111 1101 1001   little endian Änderung

0000 1000 0100 1111 0110 1011 1011 0000             invertiert

0000 0100 1100 0001 0001 1101 1011 0111             CRC-Polynom(x 04C11DB7)
--------------------------------------------------------------------------------
     1000 0100 1111 0110 1011 1011 0000 1101        4 mal geschoben bis 1

      000 0100 1111 0110 1011 1011 0000 1101 1      Datensatz für XOR
      000 0010 0110 0000 1000 1110 1101 1011 1      CRC-Polynom(x 04C11DB7)

      000 0110 1001 0110 0011 0101 1101 0110 0      Ergebnis des 1. XOR
```

Grundsätzlich existieren 2 verschiedene Möglichkeiten eine CRC-Prüfsumme zu berechnen und zwar parallel und seriell. Die in der Urfassung des Standards [XER80] und in diesem Design genutzte Weg ist die serielle Berechnung. Bei einer Verwendung des CRC mit Geschwindigkeiten höher als 100 MBit muss die CRC-Checksummenberechnung parallel durchgeführt werden. Je nach genutzter Datenbreite[8] steht hier jedem Bitwert eine XOR-Funktion gegenüber die bei jedem Takt einen validen CRC-Wert liefert. Näheres dazu findet man in der Literatur bei [CPR03] oder [NSJ].

[8] Datengröße die pro Takt verarbeitet wird

9.6 Komponente: statemachine

Die statemachine, oder auch Zustandsmaschine, übernimmt alle Steuerungsaufgaben des Designs und wurde komplett über das Programm **HDL-Designer** als Zustandsdiagramm erstellt. Hierdurch ist eine Visualisierung der einzelnen states möglich, welche durch das Programm in VHDL-Code abgebildet wird.
Zu Beginn wurden folgende Rahmenbedingungen für die Funktion definiert:

- Starten der Zustandsmaschine bei **init**
- Asynchrones Reset der Zustandsmaschine bei **rstn = '0'**
- Wechsel der Zustände mit der steigenden Flanke (**clk'event AND clk = '1'**)
- Alle Signale haben, wenn sie nicht gesetzt werden, den Standardwert **'0'**

Die Zustandsmaschine ist hierarchisch aufgebaut und besteht aus 2 Ebenen bzw. Funktionsbereiche: **Top Level** und **CRC Level**

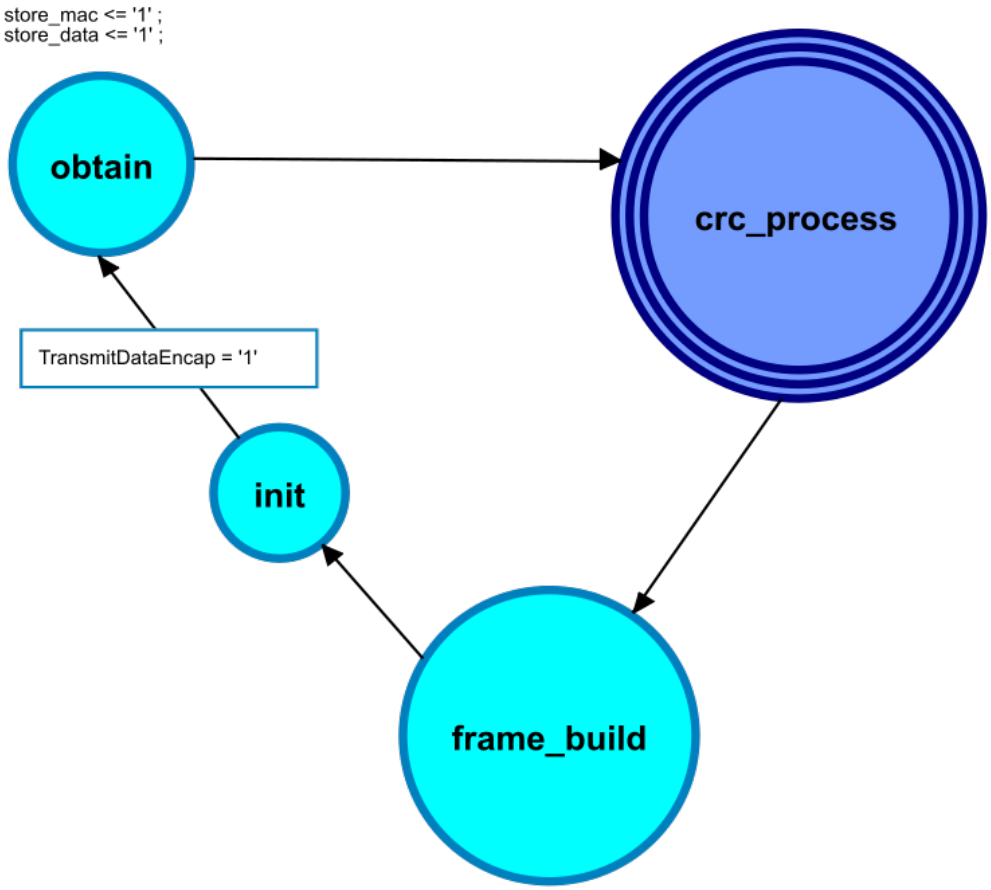

Abbildung 10: *statemachine: Top level*

Nach setzen des Input Signals **TransmitDataEncap ='1'** wechselt die Zustandsmaschine mit der nächsten steigenden Taktflanke in den Zustand **obtain**. Nach dem Zustandswechsel liegen auch die Signale des Zustands an. Nachfolgend ist das Zustandsdiagramm des hierarchischen Zustands CRC-Level abgebildet.

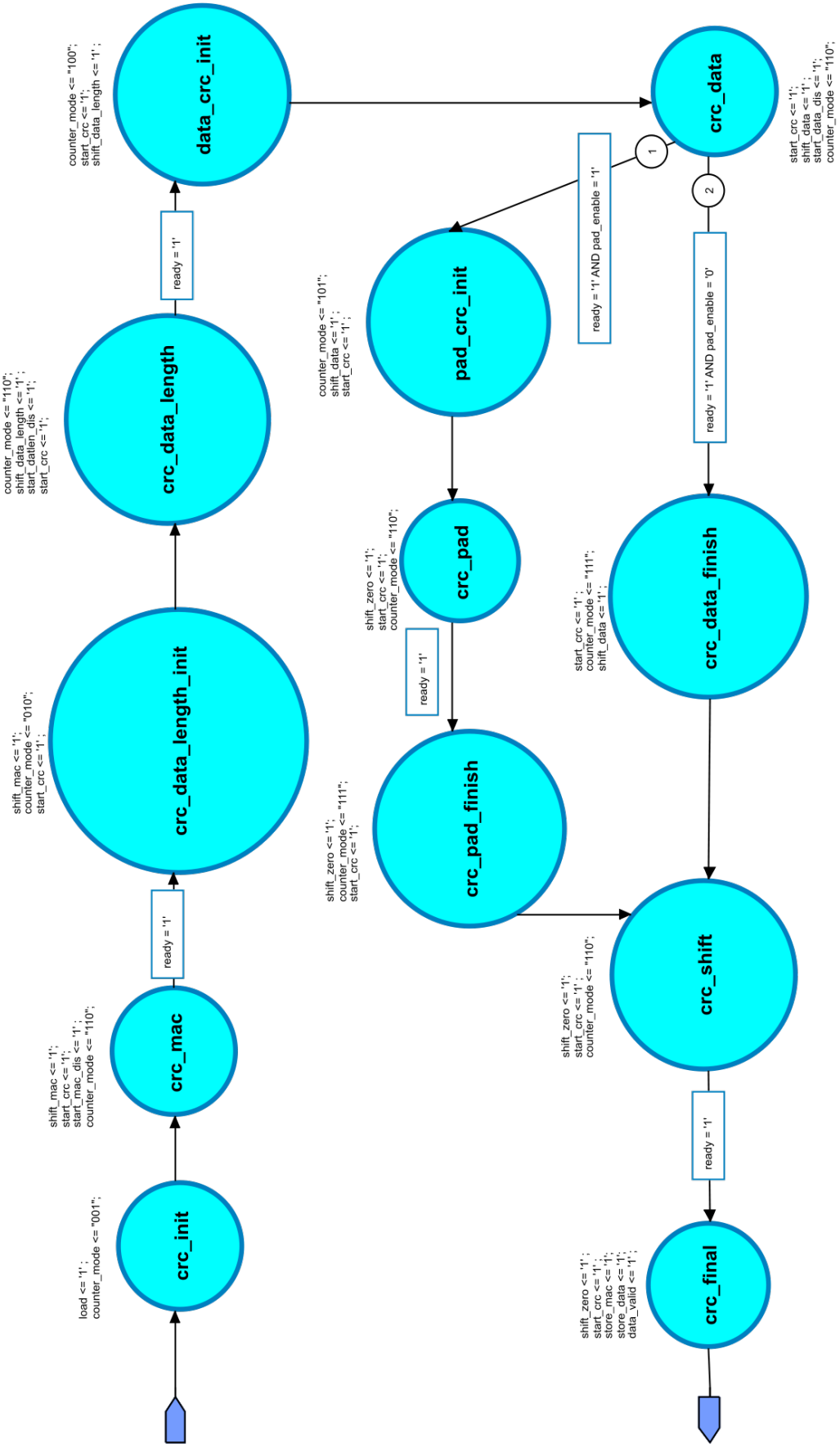

Abbildung 11: *Zustandsdiagramm: CRC level*

Die Funktion des CRC-Level ist die einzelnen Prozesse für die Generierung der Prüfsumme nacheinander und Taktgenau anzusprechen. Die Reihenfolge entspricht dem des im Standard vorgegebenen Rahmen. Erst werden nach dem **init**-Befehl und der Konfiguration des Zählers die ersten 32 Bit der MAC-Daten parallel vorgeladen. Dann beginnt der Zähl-/CRC-Prozess. Der Zähler zählt bis auf 0 runter, wobei schon bei 1 das Signal **ready** gesetzt wird. Anschließend wird der Zähler erneut vorgeladen und der Multiplexer auf die nächste Datenquelle umgestellt. Dieser Prozess wiederholt sich für die in **data_length**- und **data** abgespeicherten Daten. Danach findet eine Abfrage statt, ob das Signal **pad_enable** gesetzt ist.

- **pad_enable = '1'** Der Zähler wird mit der Zahl der Paddingbits vorgeladen, welche benötigt werden, um das Ethernetpatket, um auf die Mindestgröße von 46 Byte zu kommen.
- **pad_enable = '0'** Nach einem weiteren Takt wird die Schiebeoperation der Daten finalisiert.

Abschließend wird der Zähler auf den Wert 32 vorgelanden und das Signal shift zero gesetzt, um 32 logische Nullen in das CRC-Register zu schieben.

9.7 Komponente: frame_builder

Alle dem Datenpaket zugehörigen Daten, werden durch den frame_builder in die richtige Reihenfolge des IEEE-Standards gebracht, und mit Präambel und SFD versehen.

```
ARCHITECTURE behavioural OF frame_builder IS

constant c_pream : std_logic_vector (55 DOWNTO 0) :=
     "10101010101010101010101010101010101010101010101010101010101010";
constant c_sfd : std_logic_vector (7 DOWNTO 0) := "10101011";

BEGIN

data_frame_out (11999 DOWNTO 0) <= data_fb;
data_frame_out (12175 DOWNTO 12120) <= c_pream;
data_frame_out (12119 DOWNTO 12112) <= c_sfd;
data_frame_out (12111 DOWNTO 12016) <= sort_mac;
data_frame_out (12015 DOWNTO 12000) <= data_length_data;

END ARCHITECTURE behavioural;
```

Listing 10: Inhalt von frame_builder

9.8 Paket: ethercustom

Die verwendeten Funktionen zur Umsortierung der Bitreihenfolge sind in einem Package zentral abgelegt. Alle Funktionen werden mit Hilfe von for-Schleifen realisiert.

- **f_endian_change_crc** wird im jetzigen Design nicht verwendet. Allerdings gibt es Varianten des CRC's die dies benötigen. Ändert, wenn aufgerufen, die Bitreihenfolge der die CRC-Prüfsumme. byteweise auf LSB-First.
- **f_reverse_crc** wird wie die vorhergehende Funktion bisher nicht verwendet. Kehrt bei Aufruf die Reihenfolge der CRC-Bit's um.
- **f_endian_change_data** ändert die Bitfolge der Daten für die Maximalgröße des Datenfeldes. Jedes Byte wird auf LSB-first geändert.
- **f_endian_change_mac** ändert die Bitfolge der MAC-Adressen.
- **f_endian_change_length** ändert die Bitfolge des data_length Signals.

10 Gesamtsystem Rahmengenerator

Das Design des Rahmengenerators wurde nach seiner Fertigstellung mit 4 unterschiedlichen Datensätzen getestet. 1, 45, 46 und 1500 Bytes. Die ersten 3 Datensätze wurden verwendet, um die Padding Funktion zu testen. Die gewählte Datengrößen werden genutzt, um zu prüfen, ob das Design die Notwendigkeit von Paddingbits erkennt und die richtige Anzahl an Bits hinzugefügt werden. Getestet wird mit einem Takt von 100 MHz. Dies entspricht einem Taktzyklus von $\frac{1}{(100*10^6)Hz} = 1 * 10^{-9}s = 10ns$.

```
rstn <= '0';
TransmitDataEncap <= '0';
data (11999 DOWNTO 11640) <= x"381
     e828691f20becc19ffab4f20b9cf569541ab49f04ee5d7e6eb457cb05c8b08d31ec4586ac66ff3e42d8fe93";
data (11639 DOWNTO 0) <= (others => '0');
data_length <= x"002D";
mac_rx <= x"ef0d29f29b0e";
mac_tx <= x"509a4c0ed81f";
  wait for clk_period*3.7;
rstn <= '1';
  wait for clk_period;
TransmitDataEncap <= '1';
  wait on data_valid;
  wait for clk_period*3;
```

Listing 11: Testbench des Rahmenbilders

Die Testdaten des Rahmengenerators haben eine Größe von 45 Byte Nutzdaten. Der Rest wird mit Nullen gefüllt. Dies entspricht dem Aufbau der Daten wie in Abbildung 8, damit kein Registerwert undefiniert ist. Dazu kommen die jeweiligen MAC-Adressen und die Datenlänge. In den Abbildungen 12 und 13 wird Anfang und Ende des CRC-Prozesses dargestellt.

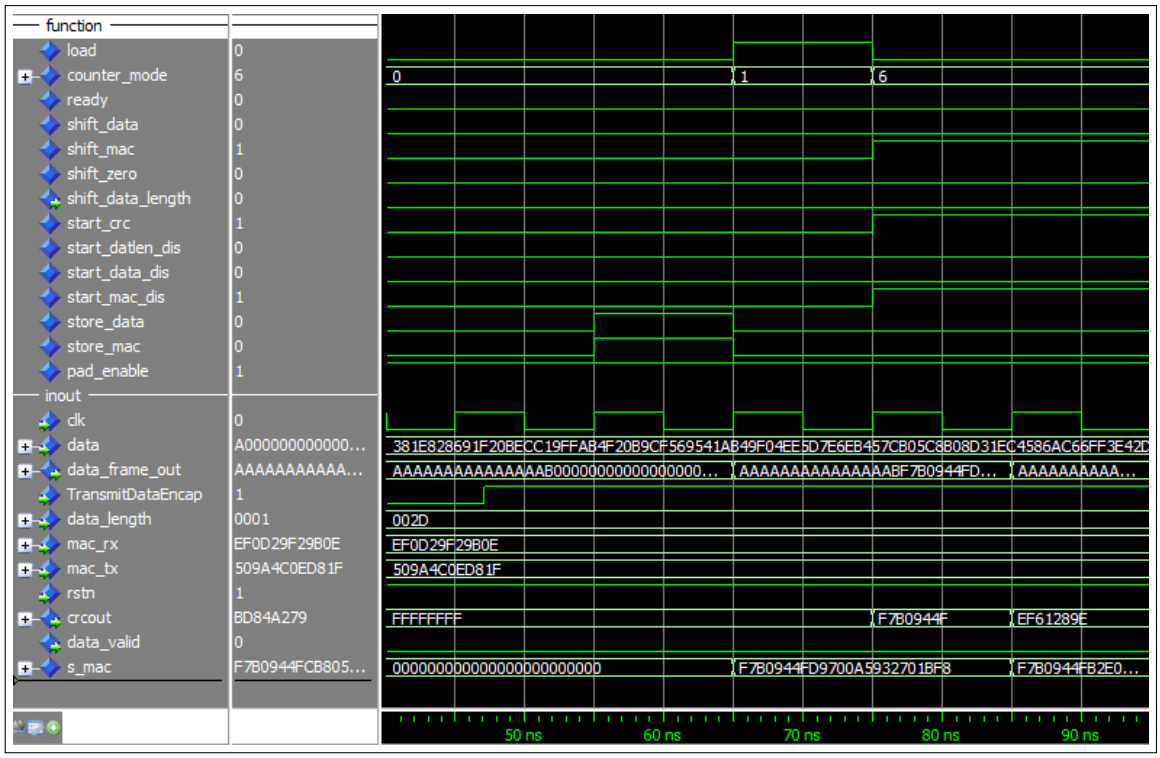

Abbildung 12: *Begin der 45 Byte Testdaten Übertragung*

Mit **TransmitDataEncap = '1'** startet der Prozess. Die anliegenden Daten werden sortiert und mit der nächsten steigenden Taktflanke gespeichert. Bei setzen des **load**-Signals werden die ersten 32 Bit von **s_mac** im CRC-Register gespeichert. In der folgenden steigenden Taktflanke beginnt der CRC-Prozess mit der ersten Schiebeoperation, gestartet durch setzen des **start_mac_dis**-, **start_crc**- und **shift_mac**-Signals.

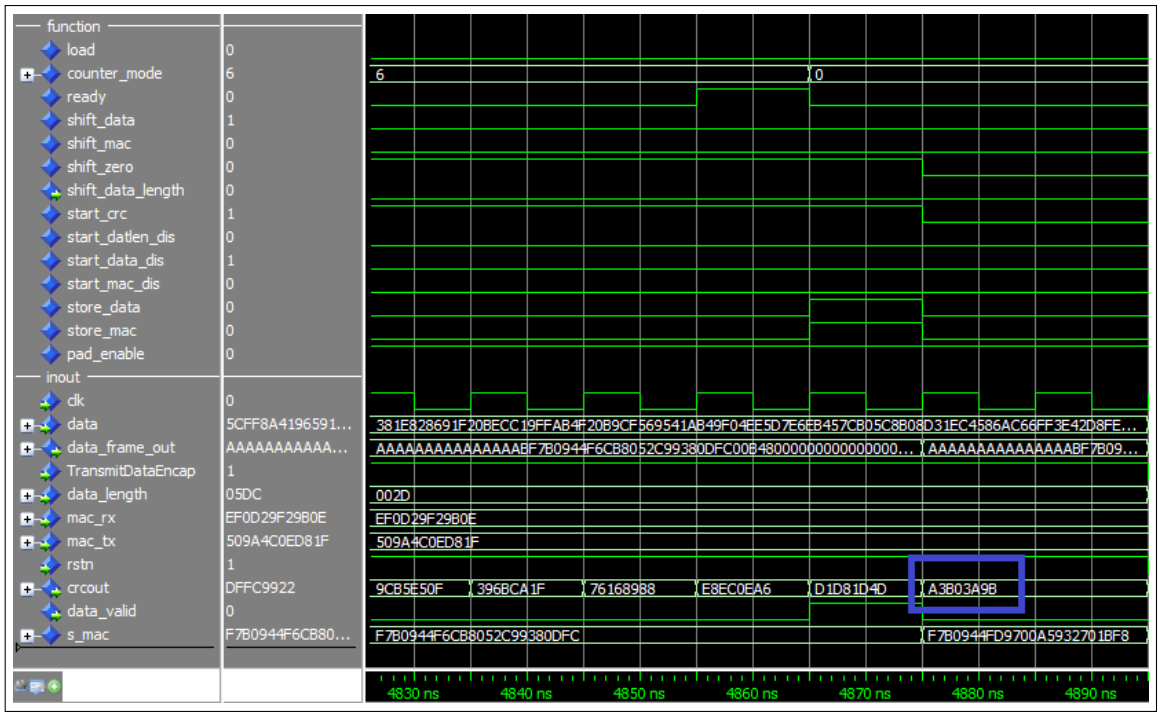

Abbildung 13: *Ende des 45 Byte Datensatzes. Valide CRC-Summe wird angezeigt.*

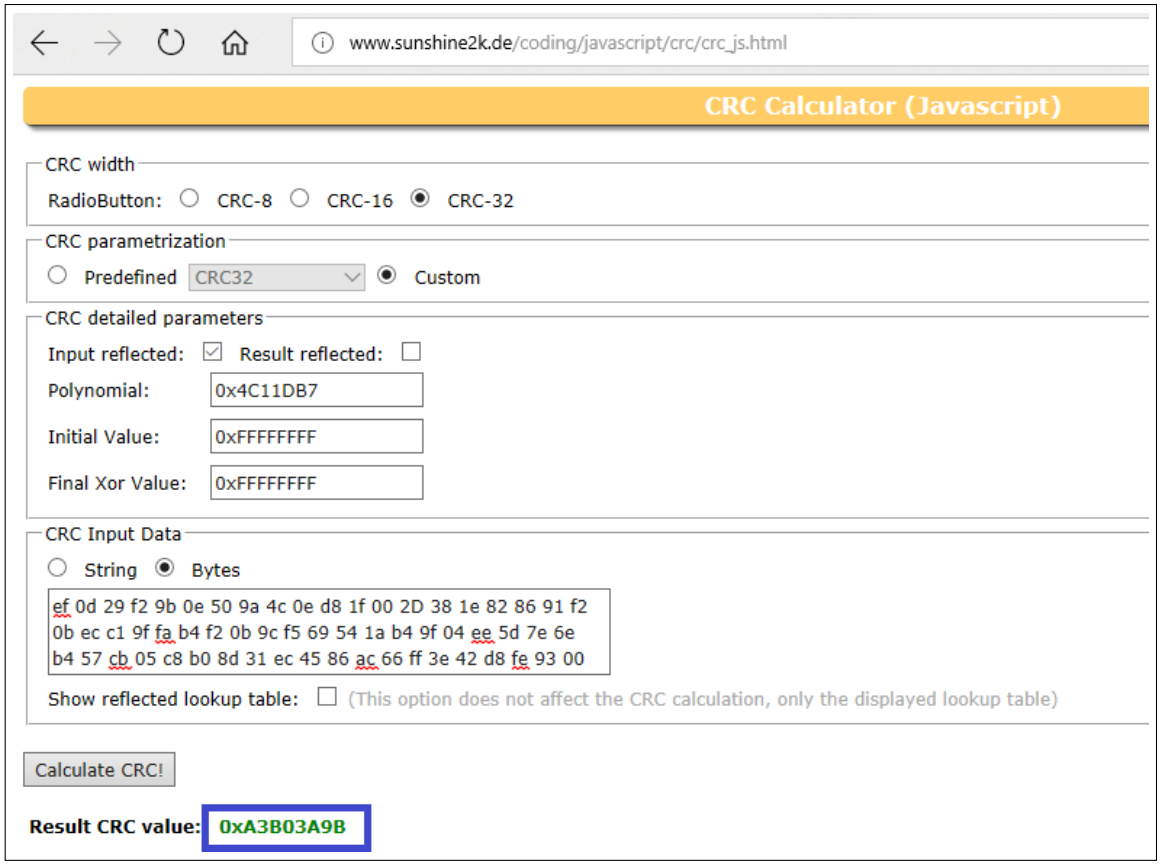

Abbildung 14: *Ergebnis eines Onlinetools zur CRC-Generierung[Mol16]*

Wie in den Abbildungen 13 und 14 markiert, wird die gleiche CRC-Summe mit dem Hex-Wert **0xA3B03A9B** errechnet.

11 Ausblick

Der Rahmengenerator funktioniert, und gibt die Daten gemäß Norm und gültigem CRC aus. Die Unterschiede der Datengröße werden berücksichtigt und ein Padding findet bei Bedarf statt. Für eine maximale Paketgröße von 1518 Byte (1500 Byte Daten + MAC-Adressen, Länge, CRC-Prüfsumme) benötigt der CRC eine Zeit von 0,12ms. Damit kann eine theoretische Übertragungsgeschwindigkeit von 123,9 MByte/s bei einem Takt von 100 MHz erreicht werden. Bei Bearbeitung der RMII-Schnittstelle ist aufgefallen, dass falls eine Kollision bei der Durchführung des CSMA/CD-Verfahrens erkannt wird, die Daten des Rahmengenerators nicht gehalten werden. Aktuell geht die Zustandsmaschine nach Beendigung der CRC Summenberechnung direkt auf den Zustand **init**. Bei einer Überarbeitung muss die Funktionalität so geändert und angepasst werden, dass die Daten des Rahmengenerators so lange gehalten werden, bis die Daten komplett versendet worden sind. Die RMII-Schnittstelle muss dem Rahmengenerator den Abschluss der Datenversendung signalisieren, damit die Zustandsmaschine des Rahmengenerators in den Basiszustand **init** übergehen kann.

Abbildungsverzeichnis

Listings

Literatur

[CPR03] Campobello, G. ; Patane, G. ; Russo, M.: Parallel CRC realization. In: *IEEE Transactions on Computers* 52 (2003), Oct, Nr. 10, S. 1312–1319. http://dx.doi.org/10.1109/TC.2003.1234528. – DOI 10.1109/TC.2003.1234528. – ISSN 0018–9340

[IEE] IEEE: *OUI, Übersicht vergebenen MAC-Adressbereiche*. http://standards-oui.ieee.org/oui/oui.txt

[Kat15] Katzier, Helmut: *Elektrische Kabel und Leitungen. Technologien, Anwendungen und Anforderungen ; mit 94 Tabellen*. Bad Saulgau : Leuze, 2015. – ISBN 978–3–87480–284–0

[Mol16] Molkenthin, Bastian: *CRC Calculator (Javascript)*. Online. http://www.sunshine2k.de/coding/javascript/crc/crc_js.html. Version: November 2016

[Neu05] Neuhaus, Ralf: *A Beginner's Guide to Ethernet 802.3*. pdf. http://www.analog.com/media/en/technical-documentation/application-notes/EE-269.pdf. Version: Juni 2005

[NSJ] Naveen, B ; Swaraja, K ; Jagdissh, MCP: Parallel CRC Generation for High Speed Applications.

[Rec08] Rech, Jörg: *Ethernet. Technologien und Protokolle für die Computervernetzung ; Standard-Ethernet ; Fast Ethernet ; Gigabit-Ethernet ; 10Gigabit-Ethernet ; Power-over-Ethernet*. 2., aktualisierte und überarb. Aufl. Hannover : Heise, 2008. – ISBN 978–3–936931–40–2; 3–936931–40–2

[Soc16] Society, IEEE C.: *IEEE Std 802.3^{TM}-2015 (Revision of IEEE Std 802.3-2012)*. PDF. http://ieeexplore.ieee.org/browse/standards/get-program/page/series?id=68. Version: Mai 2016

[XER80] XEROX, digital Intel: *The Ethernet (The BLUE Book)*. 11 1980

Anhang

A CD Inhalt: Quellcode und Dokumentation

LaTex
Thesis in der Textsatzumgebung LaTex

- **code**
  Für die LaTex-Einbindung optimierter Quellcode
- **images**
  Bilder für LaTex

Quellen
Zusätzliche Ressourcen sowie Projektdaten

- **doc**
  Dokumentation zur parallelen Berechnung des CRC
- **Ethernet**
  Unveränderter Quellcode und HDL-Designer Projekt
- **MAC Layer**
  HTML-Ansicht des HDL-Designer Projekts