Hochschule Magdeburg-Stendal
Fachbereich Ingenieurwissenschaften und Industriedesign(IWID)
Institut für Elektrotechnik

Masterarbeit

zur Erlangung des Grades eines "Masterarbeit of Engineering"
im Studiengang Funkidentifikation und Nahbereichsfunktechnik

Thema: **"Entwicklung eines QAM-Modulators mit variabler Ausgangsfrequenz"**

Eingereicht von: **Yunxiao Gao**

Angefertigt für: Hochschule Magdeburg-Stendal

Matrikel: E2010

Ausgabetermin: 04. März 2013

Abgabetermin: 21. Juli 2013

Schulischer Betreuer: Herr Prof. Dr. –Ing. Dieter Schwarzenau

Betrieblicher Betreuer: Herr Prof. Dr. –Ing. Olaf Friedewald

---------- ----------

2. Prüfer 1. Prüfer

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden.

Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.

im Juni 2013 , Yunxiao Gao

_____

Datum, Unterschrift

Danksagung

Mein Dank gilt zunächst Herrn Prof. Dr.-Ing. Dieter Schwarzenau, der mich zum Studium am QAM-Modulator ermutigt hat und meine Arbeit betreut. Weiterhin danke ich Herrn Thomas Lange für die Betreuung des 64QAM-Projektes. Insbesondere danke ich Herrn Julius Olbrich für die vielen Ratschläge. Sie gaben mir viele Hilfe, ohne ihren Ansporn könnten diese Arbeit nicht abgeschlossen werden. Unsere Zusammenarbeit war immer vom Streben nach der besseren Lösung geprägt und im Resultat sehr nützlich.

Anschließend möchte ich meinen Freunden und Kommilitonen für die ausgiebige Gespräche und Vorschläge danken. Insbesondere gilt der Dank meinen Eltern, dass sie mich immer gefördert haben, ohne zu fordern. Zum Schluss danke ich meinem Lebensgefährten dafür, dass er mir sehr viel Hilfe und Unterstützung gab.

Yunxiao Gao Magdeburg, im Juni 2013

Inhaltsverzeichnis

I. Abbildungsverzeichnis 5
II. Tabellenverzeichnis 6
III. Abkürzungsverzeichnis 7
1 Einführung 8
1.1 Entwicklung von DVB 8
1.2 Bestandteil von DVB 9
1.3 Zielsetzung 10
1.4 Bedeutung des Themas 11
1.5 Aufbau der Arbeit 11
2 QAM 12
2.1 Allgemeines 12
2.2 Mathematischer Hintergrund 13
2.3 Quantisierte QAM 14
2.3.1 Konstellationsdiagramm 14
2.3.2 Zuordnungen 16
2.3.3 Modulatorstruktur 17
3 Direkte digitale Synthese 17
3.1 Begriff 17
3.2 Prinzip 17
4 DVB-Codierung 18
4.1 Generator von PRBS und Transportströmen 18
4.2 Kanalcodierung 20
4.2.1 Invertierung und Verwürfelung 21
4.2.2 Reed-Solomon Codierung 22
4.2.3 Faltungsverschachtelung 27
4.3 Byte to symbol mapping 29
4.3.1 Differenzcodierung 29
4.3.2 Umwandlung von Byte zu m-Symbolen 30
4.3.3 Mapping 30
4.4 Basisbandfilter 31
5 Entwicklung einer QAM-Modulator-Hardware 32
5.1 Einführung über FPGA 32
5.1.1 FPGA-Arbeitsprinzip 33
5.1.2 FPGA-Entwicklungsboard 34
5.1.3 Eigenschaften vom FPGA-Entwicklungsboard 35
5.1.4 Konfigurationsarten vom FPGA-Entwicklungsboard 35
5.2 Digitaler Aufwärtswandler AD9857 36
5.2.1 Interne Struktur von AD9857 37
5.2.2 Serielle Kommunikation zwischen FPGA und AD9857 40
5.2.3 Schaltungsdesign mit AD9857 43
5.2.4 Parallele Kommunikation zwischen FPGA und AD9857 45
5.2.5 Auswahl des Betriebsmodus von AD9857 46
5.2.6 System-Clock Design 46

6 Entwicklung einer QAM-Modulator-Software 48
6.1 Bauteile und Ablaufdiagramm 48
6.2 Design des digitalen Basisbandsignalverarbeitungsmodus 50
6.2.1 Generator von PRBS und Transportströmen 50
6.2.2 Invertierung und Verwürfelung 51
6.2.3 Synchronisation zwischen Ausgabedaten der Verwürfelung und Eingabedaten der RS-Codierung 52
6.2.4 Reed-Solomon-Codierung 53
6.2.5 Faltungsverschachtelung 55
6.3 Verarbeitung mit Symbolen 57
6.3.1 Umwandlung von Byte zu m-Symbolen 57
6.3.2 Differenzcodierung 58
6.3.3 Konstellation 58
6.4 Digitales Filter 59
6.5 Reset des AD9857 60
6.6 Serielle Kommunikation zwischen FPGA und AD9857 60
6.6.1 Ablaufdiagramm für die serielle Kommunikation 61
6.7 Parallele Kommunikation zwischen FPGA und AD9857 62
6.7.1 Ablaufdiagramm von paralleler Kommunikation 62
6.8 Realisierung der SPI-Schnittstelle in FPGA 62
6.8.1 Realisierung von SPI 63
6.9 Frequenzkontrolle 64
6.10 Gesamtsystem 65
7 Ergebnisse 66
7.1 Simulationsergebnisse 66
7.1.1 Ergebnis der digitalen Basisbandsignalverarbeitung 67
7.1.2 Ergebnis der Kanalcodierung 67
7.1.3 Ergebnis der Umwandelung von Byte zu m-Symbolen 69
7.1.4 Analysierung von Synchronisationsbytes 70
7.1.5 Ergebnis der Konfiguration des AD9857 70
7.1.6 Ergebnis der SPI-Kommunikation 71
7.1.7 Überprüfung der Clock-Systeme 71
7.1.8 Gesamtergebnis 72
7.2 Hardware-Ergebnis 73
7.2.1 Schaltung und Ergebnis 73
8 Zusammenfassung 74
Literaturverzeichnis 76
Anhang 77
A.1 Korrespondenz zwischen den Feld-Elementen und binären Daten 77

I. Abbildungsverzeichnis

Abbildung 1 DVB –C System
Abbildung 2 Modulationsprinzip
Abbildung 3 Demodulationsprinzip
Abbildung 4 Konstellation von 4QAM
Abbildung 5 Konstellationsdiagramme für 16QAM, 32QAM
Abbildung 6 16QAM mit Gray-Code
Abbildung 7 QAM-Modulator für digitale Übertragung
Abbildung 8 Prinzipschaltung eines DDS-Systems
Abbildung 9 Datenformat von TS-Datenströmen
Abbildung 10 Kanalcodierung
Abbildung 11 Invertierung des ersten Synchronisationsbyte jedes Rahmen
Abbildung 12 Format von RS-Codierung zufolge des DVB-C Standards
Abbildung 13 Feedback circuit polynomial
Abbildung 14 das Prinzip der Datenverschachtelungstechnik
Abbildung 15 Prinzip der Verschachtelung und Entschachtelung
Abbildung 16 Differenzcodierung
Abbildung 17 Umwandlung von Byte zu m-Symbolen für 64QAM
Abbildung 18 interne Struktur von FPGA
Abbildung 19 Altera DE2-115 - Development and Education Board V3
Abbildung 20 JTAG-Modus
Abbildung 21 AS-Modus
Abbildung 22 Interne Struktur von AD9857
Abbildung 23 Die Struktur eines CIC-Filters
Abbildung 24 Lowpassfilter mit der Frequenz 82MHz
Abbildung 25 Anschluss zwischen AD9857 und FPGA
Abbildung 26 Zeitlauf von serieller Kommunikation
Abbildung 27 Ablauf der Übertragung der I- und Q-Daten
Abbildung 28 Verteilungsschaltung der Taktsignale
Abbildung 29 Schaltung für Taktsignal
Abbildung 30 Prinzipschaltung des QAM-Modulators
Abbildung 31 gesamtes Ablaufdiagramm
Abbildung 32 PRBS-Generator
Abbildung 33 Realisierung der Verwürfelung
Abbildung 34 Erzeugung der Paritätsprüfbits der RS-Coders durch Polynommultiplizier mit einem Schieberegister
Abbildung 35 Ablauf der RS-Codierung
Abbildung 36 Das logische Blockschaltbild der Faltungsverschachtelung
Abbildung 37 Realisierungsdiagramm von Vergabe der Lese- und Schreib-Adressen
Abbildung 38 Realisierungsschaltung der Differenzcodierung
Abbildung 39 Realisierungsmethode der Konstellation für 64QAM
Abbildung 40 Koeffizienten von digitalem Filter
Abbildung 41 Amplituden- und Phasenfrequenzgang

Abbildung 42 Impulsantwort des Basisbandfilters
Abbildung 43 serielle Kommunikationsschnittstelle
Abbildung 44 Ablaufdiagramm von serieller Kommunikation
Abbildung 45 Ablaufdiagramm von paralleler Kommunikation
Abbildung 46 Konfigurationsprinzip für SPI
Abbildung 47 Arbeitsprozess der Zustandsmaschine für SPI
Abbildung 48 Schaltung von Schalter im FPGA
Abbildung 49 Bauteil des Gesamtsystems
Abbildung 50 die Schaltung des gesamten Systems in RTL
Abbildung 51 Ergebnis der Erzeugung von TS
Abbildung 52 Ergebnis der Invertierung
Abbildung 53 Ergebnis der Verwürfelung
Abbildung 54 Ergebnis von Synchronisation
Abbildung 55 Ergebnis der Reed-Solomon-Codierung
Abbildung 56 Ergebnis der Faltungsverschachtelung
Abbildung 57 Ergebnis der Wandelung von Byte zu m-Symbolen
Abbildung 58 Ergebnis der Konstellation
Abbildung 59 Ergebnis der Ausgabe von I- und Q-Daten nach dem Filter
Abbildung 60 digitale Basisbandsignalverarbeitung
Abbildung 61 Rücksetzen von AD9857
Abbildung 62 Konfigurationen der inneren Register
Abbildung 63 SPI-Kommunikation
Abbildung 64 Clock-System
Abbildung 65 insgesamtes Ergebnis
Abbildung 66 Schaltung und Ergebnis

II. Tabellenverzeichnis

Tabelle 1 Gegenüberstellung der DVB-Übertragungsverfahren
Tabelle 2 gebräuchliche Erzeugungsfunktionen für Schieberegister
Tabelle 3 Definition von Befehlsbyte
Tabelle 4 Definitionen der Datenbytes
Tabelle 5 Befehlsbyte und entsprechende Konfigurationen von inneren Registern
Tabelle 6 Zuordnungstabelle der Lese- und Schreib-Adressen
Tabelle 7 Siganle vom gesamten System

III. Abkürzungsverzeichnis

Abkürzung	Beschreibung
DVB-C	Digital Video Broadcasting – Cable
FPGA	Field Programmable Gate Array
ETSI	European Telecommunications Standards Institute
CENELEC:	European Committee for Electronical Standardization
EBU	European Broadcasting Union
ISDB	Integrated Services Digital Broadcasting
VSB	Vestigial Sideband Modulation
ATSC	Advanced Television Systems Committee
MPEG	Moving Picture Experts Group
SPI	Serial Peripheral Interface
QAM	Quadrature Amplitude Modulation
MQAM	Multilevel QuadratureAmplitude Modulation
DVB-S	Digital Video Broadcasting – Satellite
DVB-H	Digital Video Broadcasting – Handheld
DVB-T	Digital Video Broadcasting – Terrestrial
DTG	Digital Terrestrial Group
DDS	direkte digitale Synthese
ZF	Zwischenfrequenz
CIC	Cascaded-Integrator-Comb Filter
DAC	Digital to analog converter
ROM	Read Only Memory
PRBS	Pseudo Random Binary Sequence
TS	Transport Stream
RS	Reed-Solomon
GF	Galois Field
FIFO	Frist in Frist out
LUT	Look-Up-Table
LCA	Logic Cell Array
CLB	Configurable Logic Block
IOB	Input Output Block
SB	Switch Box
CB	Connection Box
LE	Logikelemente
JTAG	Joint Test Action Group
AS	Active Serial
DC	Direct Current
GMSK	Gaussian Filtered Minimum Shift Keying
CDMA	Code Division Multiple Access
OFDM	Orthogonal Frequency Division Multiplexing
PDCLK	Parallel Data Clock
SYSCLK	System Clock
LC	Lowpass Filter
PLL	Phase Lock Loop
REFCLK	Referenztaktsignal
NCO	Numerical Controlled Oscillator
MCU	Micro Control Unit
SPI	Serial Peripheral Interface
SCLK	Serial Clock
$\overline{CS}$	Slave Select
SDIO	Serial Data In and out
SDO	Serial Data Out
RTL	Real Time Logistics

1 Einführung

Die Aufgabe bestand in der Entwicklung eines QAM-Modulators mit variabler Ausgangsfrequenz, wie er in DVB-C verwendet wird.

Mit dem Aufkommen des digitalen Zeitalters wird das Niveau der Informationstechnologie kontinuierlich verbessert. Der Informationsaustausch und Datenaustausch zwischen Menschen sind weltweit gestiegen. Quadraturamplitudenmodulation wird als eine digitale Modulation mit hoher spektraler Effizienz betrachtet und weithin im Bereich der Breitbandkommunikation verwendet, z.B. bei digitaler Fernsehübertragung, Satellitenkommunikation, digitaler Mikrowellenübertragung usw.

In den vergangenen Jahren haben integrierte Schaltungen und digitale Kommunikationstechniken einen hohem Stand erreicht. FPGA sind in der Elektronik-Industrie weit verbreitet. Weil FPGA als eine allgemeine Logik verwendet werden kann, ist der Marktanteil von FPGA gestiegen. FPGA-Chips sind außerdem sehr hoch integriert und einfach einzusetzen. Codeportabilität ist heute möglich.

1.1 Entwicklung von DVB

DVB ist die Abkürzung für „Digital Video Broadcasting“, DVB ist eine Reihe von international anerkannten offenen Standards für digitale Fernsehsignale. Das DVB-Projekt ist eine Organisation mit über 300 Mitgliedern. Im DVB- Projekt wurden seit 1992 zahlreiche offene Spezifikationen für das digitale Fernsehen entwickelt, die den gesamten Bereich vom Studio bis zum Endgerät abdecken. Diese Spezifikationen wurden an europäische Normenorganisationen ETSI, CENELEC und EBU übergeben, die sie als offizielle europäische Normen und sonstige Dokumenten veröffentlicht haben[1].

Es gibt verschiedene Varianten, die wie folgt zusammengefasst werden können:

1. DVB-S und DVB-S2 für die Satellitenübertragung
2. DVB-C für die Übertragung in den Kabelnetzen
3. DVB-T und DVB-T2 für die terrestrische Funkversorgung
4. DVB-H : Digital Video Broadcasting – Handheld

DVB-S und DVB-C wurden im Jahr 1994 erstellt und DVB-T im Jahr 1997. Der erste kommerzielle DVB-T Rundfunk wurde im Jahr 1998 durch die britische DTG (Digital Terrestrial Group) gestartet. Im Jahr 2003 wurde Berlin das erste Gebiet ohne analoges terrestrisches TV-Signal. Aber bei der Verbreitung von DVB-C hatten die großen Kabel-Anbietern in Deutschland einige Probleme. Lange Zeit hat es keine

vollständig abdeckende Ausstrahlung von privaten Sender über DVB-C gegeben, ausschließlich die öffentlichen rechtlichen Sender und einige Bezahlfernsehen-Angebote konnten empfangen werden. Diese Situation hat sich seit Januar 2006 verbessert[2].

In seinen Herkunftsgebieten Europa, Australien, Südafrika und Indien ist DVB jetzt stark verbreitet. Die Mehrheit der Länder von Asien, Afrika und Südamerika verwendet DVB-Standards im Kabel und über Satellit. In vielen Ländern ist DTTV (Digital terrestrial television) noch nicht bestimmt. Argentinien und Südkorea nutzen einen alternativen Standard: ATSC.

Ein anderer digitaler Fernsehstandard (ISDB) wird in Japan verwendet. Er ist sehr ähnlich wie der DVB-Standard. In Nordamerika wird vor allem DVB-S als Satellitenübertragungsstandard benutzt. Im Kabel benutzt Nordamerika auch DVB-C. Allerdings wird der 8VSB modulierte ATSC-Standard für terrestrisches digitales Fernsehen verwendet.

System	DVB-S	DVB-S2	DVB-C	DVB-T
Norm	ETSI EN 300 421	ETSI EN 302 307	ETSI EN 300 429	ETSI EN 300 744
Modulations-art	QPSK	QPSK, 8PSK, 16APSK, 32APSK	64QAM 256QAM	COFDM, QPSK, 16QAM, 64QAM
Bandbreite	33MHz/36MHz	33MHz/36MHz	7MHz /8MHz	7MHz oder 8MHz
typ. Symbolrate	27,5 Msymb/s	30,9 Msymb/s	6,900 Msymb/s	5,39 Msymb/s
typ. Nutz-datenrate	38,015 Mbit/s	51 Mbit/s	38,1 Mbit/s	13,06 Mbit/s,R=3/4 13,27 Mbit/s,R=2/3
sonstiges	Roll-Off-Faktor: 0,35	Roll-Off-Faktor: 0,35 0,25 0,20	Roll-Off-Faktor: 0,15	veränderbares Guard-Intervall 1/4 oder 1/8

Tabelle 1: Gegenüberstellung der DVB-Übertragungsverfahren [3]

1.2 Bestandteil von DVB

Das auf dem MPEG-2-Standard basierende DVB-C-System verwendet das Kabelnetz als Übertragungsmedium, wobei eine 64-stufige Quadratur-Amplituden-Modulation (64QAM) zum Ersatz kommt. Manchmal werden auch 16QAM, 32QAM, 128QAM oder 256QAM verwendet. Für QAM-Modulation gilt, je höher die Menge der Signalzustände ist, desto schwächer ist die Störungsfertigkeit.

Bei der Kompression und der Formatierung von Bilddaten verwendet DVB die internationalen MPEG-Standards.

Das DVB-System bezieht sich darauf, Video und Audio zu digitalisieren und MPEG-2-Datenströme zu codieren. Die Verarbeitungen von MPEG-2-Datenströmen beinhalten Multiplex und Demultiplex, sowie Kanalcodierung und Kanaldecodierung, Modulation und Demodulation, Filterung usw.

Abbildung1 zeigt eine Übersicht der Anwendung.

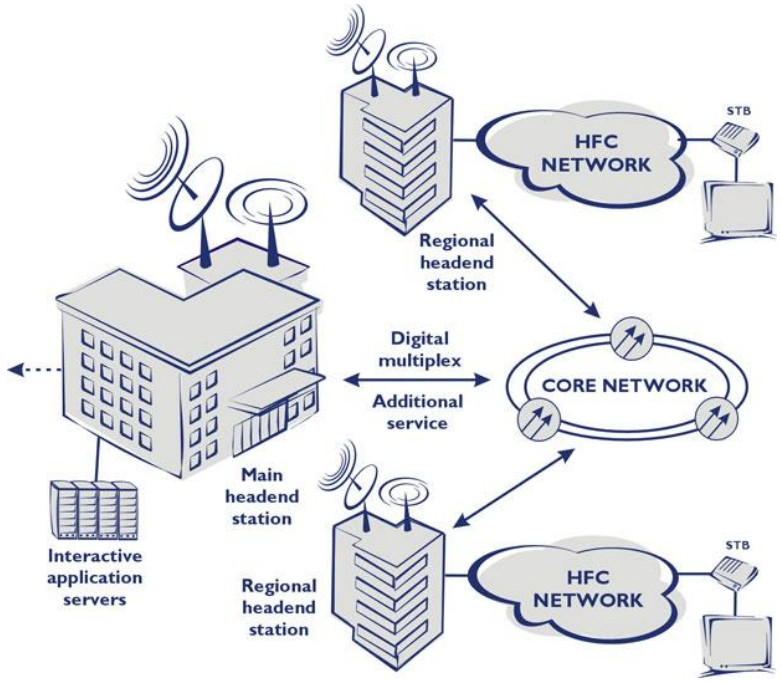


Abbildung 1 DVB –C System[4]

1.3 Zielsetzung

Zielsetzung ist die Entwicklung eines QAM-Modulators mit variabler Ausgangsfrequenz.

In Europäischen Kabelnetzen wird die weit verbreitete DVB-C-Modulation, bzw. das QAM-Verfahren verwendet. Das QAM-Verfahren ist ein aus Amplitudenmodulation und Phasenmodulation kombiniertes Verfahren. Bei dem QAM-Verfahren werden gleichzeitig die Amplitude und Phase des Trägersignals benutzt, um Informationen zu übertragen. Die unterschiedlichen Amplituden und Phasen können für verschiedene codierte Symbole stehen. QAM-Modulation bietet eine größere Anzahl von Konstellationspunkten, dadurch kann eine höhere Bandbreiteneffizienz erreichen[5].

DVB-C belegt 7MHz oder 8MHz Bandbreiten. Die Übertragungsrate ist sehr hoch. Sie

Kann bei 64QAM bis zu 38,1Mbps betragen. 64QAM hat eine hohe Modulations-effizienz, erfordert aber ein hohes Signal-zu-Rausch-Verhältnis.

1.4 Bedeutung des Themas

Das DVB-Übertragungssystem besteht vor allem aus zwei Teilen, der Quellcodierung und der Kanalcodierung. Bei der Quellcodierung wird der MPEG-2-Standard verwendet. Diese Technik ist relativ ausgereift. Im Allgemeinen wird sie durch einen mit MPEG-2 codierten Chip implementiert. Kanalcodierung und Modulationstechnik sind jedoch die Schlüsseltechnologien von DVB-C und es gibt verschiedene Arten den Kombinationen. In der letzten Zeit wurden viele Studien über QAM-Modulation mit Hilfe FPGA und DAC-Chip durchgeführt, aber es ist sehr schwierig, alle Bauteile des QAM-Modulators mit FPGA zu realisieren. Dieses Verfahren erfordert viele Ressourcen in einem FPGA. In dieser Entwicklung wird ein zusätzlicher IQ-Modulator ausgewählt. Dadurch kann die Entwicklung vom QAM-Modulator vereinfacht werden.

1.5 Aufbau der Arbeit

In Kapitel 2 dieser Arbeit wird zunächst die theoretischen Grundlagen des QAM-Prinzips erklärt. Dazu gehört der mathematische Hintergrund und die quantisierte QAM. Bei der quantisierten QAM werden das Konstellationsdiagramm, die Zuordnungen und die Modulatorstruktur eindeutig erklärt.
Im Kapitel 3 wird das DDS-Prinzip vorgestellt.
Anschließend folgt im Kapitel 4 die Beschreibung der DVB-Codierung.
Im Kapitel 5 werden Hardware von FPGA und IQ-Modulator erklärt, dazu zählt zum einen die Erläuterung des FPGA-Arbeitsprinzips, der internen Struktur und des Betriebsmodus des IQ-Modulators.
Anschließend folgt im Kapitel 6 die Beschreibung der Software-Entwicklung eines QAM-Modulators, welche den Schwerpunkt der Arbeit darstellt. Dieses Kapitel beginnt mit der Entwicklung der digitalen Basisbandsignalverarbeitung mit dem Verilog HDL-Programm in Quartus II. Die digitale Basisbandsignalverarbeitung verfügt über die Erzeugung von PRBS und TS-Datenströmen, die Kanalcodierung, die Umwandlung von Byte zu m-Symbolen, die Differenzcodierung, das Mapping und das Basisbandfilter. Weiterhin gibt es den Codierungsprozess bzw. das Ablaufdiagramm für die Konfigurationen der inneren Register des IQ-Modulators über die serielle Schnittstelle, um die Betriebsart und Arbeitsfrequenz des IQ-Modulators zu

kontrollieren. Danach wird der Codierungsprozess über die parallele Kommunikation zwischen FPGA und IQ-Modulator geschrieben, um die Daten zu senden und im IQ-Modulator zu modulieren.
Anschließend sind die Ergebnisse dargestellt. Dazu gehören Simulationsergebnisse in ModelSim Altera 6.6d und Hardware -Ergebnis.
Die Arbeit schließt mit Zusammenfassung und Ausblick.

2 QAM

„Die Quadraturamplitudenmodulation ist ein Modulationsverfahren in der elektronischen Nachrichtentechnik, das die Amplitudenmodulation und Phasenmodulation miteinander kombiniert. Sie wird in der Fachliteratur überwiegend zu den digitalen Modulationsverfahren gezählt, wenngleich auch Formen der analogen Quadraturamplitudenmodulation unter der Bezeichnung Quadraturmodulation existieren."[6]

2.1 Allgemeines

Zur Erzeugung der QAM-Modulation werden zwei um 90° phasenverschobene Sinusträger verwendet, darauf werden zwei voneinander unabhängige Basisbandsignale je mithilfe einem Multiplizier durch diese Träger moduliert. Danach werden die beiden modulierten Signale addiert, um das QAM-Signal zu bekommen. Die zwei Basisbandsignale werden in der englischen Literatur auch als „*I*" für „In-phase Signal" und *„Q"* für „Quadrature Signal" beschrieben, was zu dem Namen IQ-Modulation führte[7].
Die beiden *I*- und *Q*-Basisbandsignale können nicht nur voneinander abhängig, sondern auch unabhängig sein. Nur wenn die beiden Basisbandsignale voneinander unabhängige Informationen tragen, wird das Verfahren QAM genannt.
Gebräuchlich sind 16-, 64-, und 256-stufige QAM. Jedem Kennzustand sind $log_2 M$ Bits zugeordnet, wobei der Wert von *M* die Stufenzahl ist. Das QAM-Verfahren benötigen einen sehr linearen Übertragungsweg, weil bei höherstufigen QAM-Verfahren geringe Amplituden- und Laufzeitverzerrungen zu Fehlern führen können. Von besonderer Bedeutung sind heute die 64 QAM und die 256 QAM für die Übertragung von DVB-Signal in Kabelnetzen[8]. Es werden aber auch bis zu 1024 Zustände realisiert.
Zur Demodulation von QAM muss der Träger im Empfänger sowohl die gleiche Frequenz wie bei dem Sender haben, als auch eine identische Phasenlage. Das

Verfahren wird auch als kohärente Demodulation bezeichnet[9].

2.2 Mathematischer Hintergrund

Das QAM-Signal s(t) wird wie in Abbildung 2 erzeugt[10].

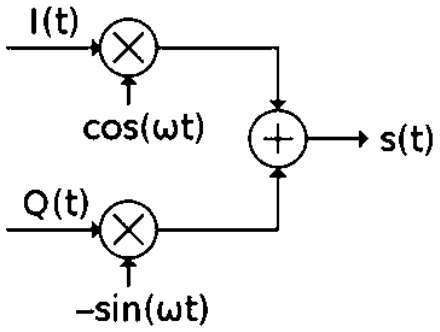


Abbildung 2 Modulationsprinzip

Das QAM-Signal s(t) wird durch die folgende Gleichung (2-1):

$$s(t) = I(t) \cdot \cos(\omega t) - Q(t) \cdot \sin(\omega t)$$
$$= I(t) \cdot \cos(\omega t) + Q(t) \cdot \cos(\pi/2 + \omega t) \qquad (2\text{-}1)$$

in dem Modulator gebildet. Wobei $\omega = 2\pi f$ die Kreisfrequenz ist und f für die Trägerfrequenz steht.

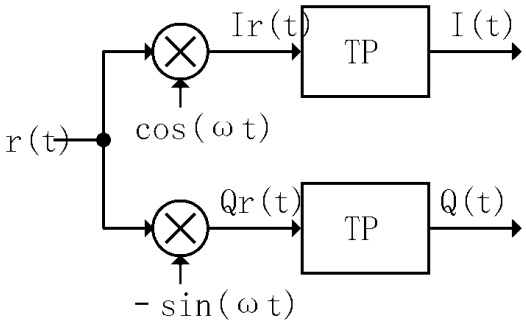


Abbildung 3 Demodulationsprinzip

Zur Demodulation ist es erforderlich, dass die Frequenz und Phasenlage zum Sender identisch konfiguriert sind. Wenn es in einem Übertragungskanal keine Störungen gibt, dann ist das Empfangssignal *r(t)* gleich dem QAM-Signal *s(t)*. Wenn dieser Kanal mit Störung ist, dann addieren sich die Fehleranteile *e(t)*, es gilt die folgende Gleichung[11]:

$$r(t) = s(t) + e(t). \qquad (2\text{-}2)$$

Wenn es keine Fehler gibt, h.d. *e(t)*=0, gilt für die Gewinnung des Basisbandsignals I_r [12]:

$$I_r(t) = s(t)\cos(\omega t)$$
$$= I(t)cos(\omega t)\cos(\omega t) - Q(t)sin(\omega t)\cos(\omega t)$$
$$= \frac{1}{2} I(t)[1 + \cos(2\omega t)] - \frac{1}{2} Q(t)sin(2\omega t)$$

$$= \frac{1}{2} I(t) + \frac{1}{2} [I(t)\cos(2\omega t) - Q(t)sin(2\omega t)] \qquad (2\text{-}3)$$

An dieser Gleichung kann man sehen, dass es im Signal $I_r(t)$ zwei zusätzliche Multiplikationsprodukte mit der doppelten Frequenz neben dem gewünschten Basisbandsignal $I(t)$ gibt. Durch einen Tiefpassfilter (TP) können diese unerwünschten Multiplikationsprodukte weggefiltert werden, somit kann das originale Signal $I(t)$ am Ausgang von einem Demodulator gewonnen werden.

Die Formation von $Q_r(t)$ ist fast gleich wie $I_r(t)$, der Unterschied ist die Multiplikation mit einem negativen Sinussignal[13]:

$$\begin{aligned} Q_r(t) &= s(t)\,(-\sin(\omega t)) \\ &= -I(t)cos(\omega t)\sin(\omega t) + Q(t)sin(\omega t)\sin(\omega t) \\ &= -\frac{1}{2}\, I(t)sin(2\omega t) + \frac{1}{2} Q(t)[1 - \cos(2\omega t)] \\ &= \frac{1}{2} Q(t) - \frac{1}{2}[I(t)\sin(2\omega t) + Q(t)cos(2\omega t)] \qquad (2\text{-}4) \end{aligned}$$

Für die Bildung von $Q_r(t)$ wird auch einen Tiefpassfilter am Ausgang benötigt, um die unerwünschten Frequenzanteile wegzufiltern.

In der Gleichung (2-3) und (2-4) gibt es je einen konstanten Faktor von 1/2 vor $I(t)$ und $Q(t)$, dieser Faktor kann durch eine Verstärkung ausgeglichen werden.

2.3 Quantisierte QAM

Bei der allgemeinen QAM wird ein Analogsignal übertragen. Bei der quantisierten QAM werden wertdiskrete und zeitdiskrete Signalfolgen übertragen, die auch als Digitalsignal bezeichnet werden.

2.3.1 Konstellationsdiagramm

Das Konstellationsdiagramm oder Zustandsdiagramm ist eine grafische Darstellung, die die digital modulierten Signale in einem zweidimensionalen Koordinatensystem für QAM, QPSK, COFDM usw. beschreibt.

Jedes Signal kann als ein Vektor mit einer horizontalen Komponente *I* (In Phase) und vertikalen Komponente *Q* (Quadrature), bzw. mit einer Realteil *I* und Imaginärteil *Q* in einer komplexen Ebene betrachtet werden[14].

In dieser komplexen *I/Q*-Ebene können die Anzahl der verfügbaren Symbole, die die Punkte in dieser komplexen Ebene darstellen, durch eine genaue Zahl ausgedrückt werden. Zum Beispiel hat 64QAM 64 verfügbare Symbole.

2.3.1.1 Prinzip

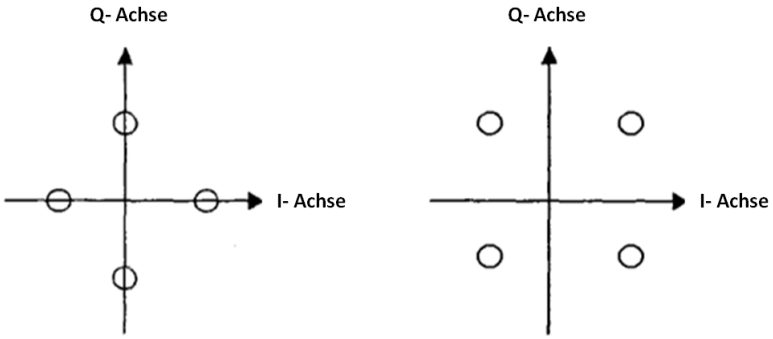


Abbildung 4 Konstellation von 4QAM

Vier mögliche Kombinationen (00, 01, 10, 11) von zwei aufeinander folgenden Bits eines binären Codesignals entsprechen vier Phasenlagen einer Sinuswelle:

$$s_i(t)=\cos(\omega_0 t + Q_i) \quad \text{wobei i=1,2,3,4; } -T/2 \leq t \leq T/2 \qquad (2\text{-}5)$$

Hier kann Q_i 0, $\pm\pi/2$, π oder $\pm\pi/4$, $\pm 3\pi/4$ sein. Dies ist 4 QAM. Die obige Gleichung (2-5) kann auch wie folgende Gleichung (2-6) beschrieben werden:

$$s_i(t)=A_m \cdot cos(\omega_0 t)+B_m \cdot sin(\omega_0 t) \quad \text{wobei } -T/2 \leq t \leq T/2 \qquad (2\text{-}6)$$

Wenn Q_i 0, $\pm\pi/2$ und π ist, dann wird:

$$(A_m, B_m)=(1, 0), (0, 1), (-1, 0), (0,-1) \qquad (2\text{-}7)$$

Wenn Q_i $\pm\pi/4$, $\pm 3\pi/4$ ist, dann wird:

$$(\sqrt{2}A_m, \sqrt{2}B_m)=(1, 1), (-1, 1), (-1, -1), (1,-1) \qquad (2\text{-}8)$$

Die Konstellation wird durch die Punkte auf dem zweidimensionalen Koordinatensystem ausgedrückt, wie in der Abbildung 4 gezeigt wird. Wobei die horizontale Komponente A_m als die Inphasekomponente und die vertikale Komponente B_m als die Quadraturkomponente bezeichnet werden.

Durch 4QAM-Modulation können 2 Bits innerhalb eines Taktes übertragen werden, die Datenrate wird bei gleicher Bandbreite verdoppelt. Um Intersymbolinterferenzen zu vermeiden, die durch die aufgrund von Kanalcharakteristiken verursachte Verlängerung von Impulsen zur nächsten Entscheidungszeit hervorgerufen werden, sollten die Signale geformt werden.

Wenn A_m und B_m verschiedene Werte annehmen, dann wird die Verarbeitung als „Quadraturamplitudenmodulation“ bezeichnet. In Abbildung 5 werden die Konstellationsdiagramme von 16QAM und 32QAM dargestellt. Abbildung 5 zeigt, dass in der In-Phase- Achse und Quadratur-Phase-Achse die Amplitudenstufen nicht mehr gleich 2, sondern gleich 4 (16QAM) oder 5 (32QAM) sind und die übertragenen Datenraten sind auch vierfach oder fünffach höher als die ursprünglichen (ohne Rücksicht auf

den Roll-off-Faktor). Jedoch können die Stufenzahlen nicht unbegrenzt erhöht werden, um die Datenraten zu erhöhen. Da mit der Zunahme der Stufenzahl das Intervall zwischen den Zuständen reduziert wird, erhöht sich bei gleichem Rauschen die Bitfehlerwahrscheinlichkeit. Im Zeitbereich nehmen die Intersymbol- interferenz und der Jitter zu. Diese Effekte können zu einem schlechten Empfang führen.

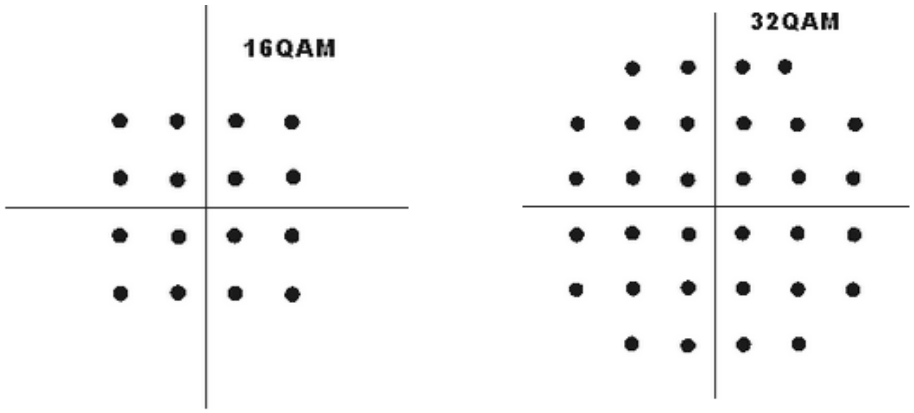


Abbildung 5 Konstellationsdiagramme für 16QAM, 32QAM [15]

2.3.2 Zuordnungen

QAM ist eine Vektor-Modulation. Die Eingangsbits werden Signalzuständen zugeordnet (Konstellationsdiagramm). Bei dieser Zuordnung wird im Allgemeinen der Gray-Code verwendet. Die besondere Eigenschaft von diesem Code ist, dass sich von einem Nachbarwert zum nächsten nur ein Bit ändert, somit kann der Gray-Code Fehler reduzieren. Wegen dieser Eigenschaft kann die Möglichkeit von Auftreten eines ernsten Fehlers sehr stark reduziert werden. In der Abbildung 6 gibt es als Beispiel die Zuordnung von 16QAM mit Gray-Code[16]. Diese Abbildung zeigt, dass jedes Symbol nur in einer Bitstelle vom benachbarten Symbol verschieden ist.

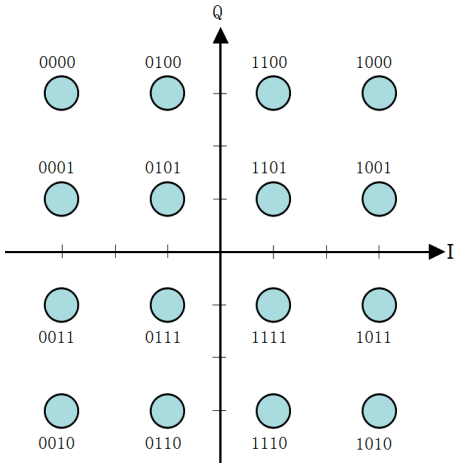


Abbildung 6 16QAM mit Gray-Code

2.3.3 Modulatorstruktur

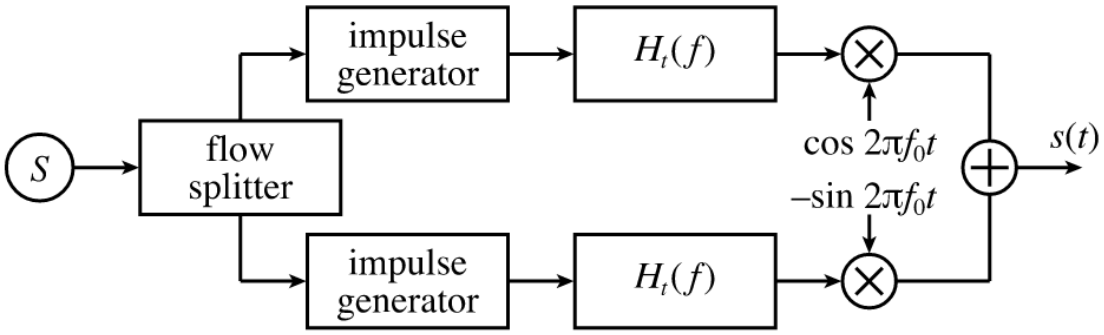


Abbildung 7 QAM-Modulator für digitale Übertragung [17]

Ein QAM-Modulator setzt sich aus einem Flow-Splitter, zwei Impulsgeneratoren, zwei Formungsfilter, zwei Multiplizier und einem Addierwerk zusammen. Der Flow-Splitter formt den Eingangsbitstrom um in zwei parallele Bitströme. Nach dem Flow-Splitter ist die Bitrate jedes Bitstroms die Hälfte vom Eingangsbitstrom. Der Impuls-generatoren kann einer bestimmten Anzahl von Bits ein bestimmtes Signalniveau am Ausgang zuweisen. Beispielsweise werden bei 64QAM jeweils Zweig jeweils 3 Bits vom Impulsgenerator kombiniert und bilden 8 Stufen mit den bipolaren Niveaus −14, −10, -6, -2, 2, 6, 10 und 14. Diese Werte entsprechen den Koordinatenswerten der *I*- und *Q*-Komponenten auf der *I*- und *Q*-Achse im Konstellationsdiagramm[18].

3 Direkte digitale Synthese

3.1 Begriff

„Die Direct Digital Synthesis oder direkte digitale Synthese (kurz DDS) ist ein Verfahren in der digitalen Signalverarbeitung zur Erzeugung periodischer, bandbegrenzter Signale mit praktisch beliebig feiner Frequenzauflösung. Das Verfahren stellt heute neben der rational gebrochenen Phasenregelschleife die dominierende Methode zur Generierung von Signalen fein einstellbarer Frequenz dar und hat weite Verbreitung in der Nachrichten- und Messtechnik gefunden.“ [19]

3.2 Prinzip

Das Grundprinzip von DDS ist die Verwendung des Abtasttheorems und die Erzeugung einer Wellenform mit Hilfe einer Tabelle. Es gibt eine Vielzahl von Strukturen der DDS. Eine grundlegende Prinzipschaltung ist in Abbildung 8 dargestellt.

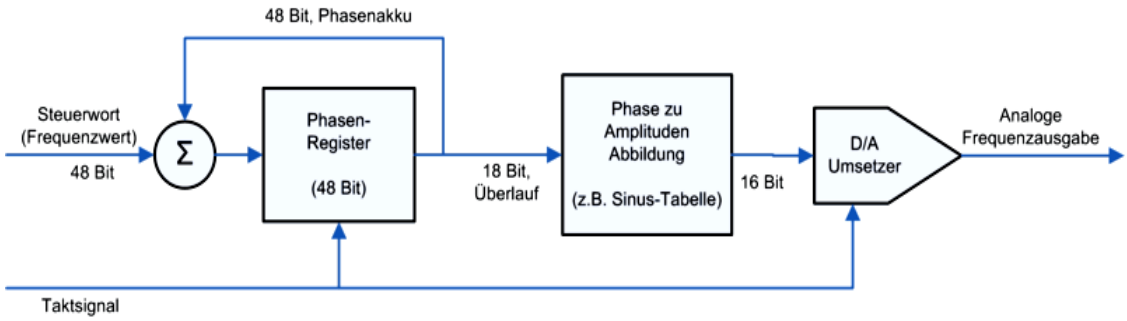


Abbildung 8 Prinzipschaltung eines DDS-Systems [20]

Der Phasenakkumulator besteht aus einer Kaskade von N Bits Akkumulationsregister und N Bits Addierwerk. Bei jedem Takt addiert das Addierwerk die Steuerwörter der Frequenz mit den durch das Akkumulationsregister ausgegebenen addierten Phasendaten. Die Summe wird zum Dateneingangsanschluß des Akkumulationsregisters gesendet. Das Akkumulationsregister koppelt die bei dem vorherigen Taktimpuls neu erzeugten Phasendaten vom Addierwerk zu dem Eingang des Addierers zurück, damit das Addierwerk bei dem nächsten Taktimpuls weiterhin mit den Steuerwörtern der Frequenz addieren kann. Somit kann das Akkumulationsregister mit den Taktimpulsen die Steuerwörter der Frequenz kontinuierlich linear addieren. Die Ausgangsdaten von dem Phasenakkumulator sind die Phasen des synthetisierten Signals, die Überlaufsfrequenz vom Phasenakkumulator ist die Ausgangsfrequenz.

Die durch den Phasenakkumulator ausgegebenen Daten werden als die Abtastphasen-Adresse vom Signalformspeicher benutzt, damit die im Signalformspeicher gespeicherten Abtastwerte (Binärer Code) der Signalforms durch eine Tabelle (z.B. Sinus-Tabelle) nachgeschlagen werden können, um die Phasen in Amplituden umzuwandeln. Die Ausgangsdaten vom Signalformspeicher werden dem D/A-Wandler zugeführt. Der D/A-Wandler kann die digitalen Signalamplituden in analoge Signale mit der gewünschten synthetisierten Frequenz umrechnen.

4 DVB-Codierung

4.1 Generator von PRBS und Transportströmen

Um ein DVB-Signal zu realisieren, müssen zuerst die Datenströme erzeugt, danach verarbeitet werden. Hier wird ein Generator von PRBS und TS-Datenströmen erklärt.

„Die Pseudorandom Binary Sequence (PRBS) ist ein binäres Signal, welches das Spektrum von weißem Rauschen approximiert und durch einen deterministischen Zufallsgenerator erzeugt wird.“[21]

PRBS hat die Eigenschaft von Zufälligkeit, weil die binären Zahlen „*0*" und „*1*" zufällig in PRBS erscheinen. Aber die Zufälligkeit von PRBS ist beschränkt. Nur in einem internen Zyklus sind „*0*" und „*1*" zufällig verteilt. Ihre Zufälligkeit ist von einer Erzeugungsfunktion des Codestroms und einem Anfangszustand abhängig.

Die Pseudozufallsfolge wird durch ein linear rückgekoppeltes Schieberegister erzeugt. Durch die Wahl der Stufenzahl und Anzapfungen können unterschiedliche Folgen gewonnen werden. In der Tabelle 2 gibt es einige gebräuchliche Erzeugungsfunktionen mit verschiedenen Stufenzahlen und Anzapfungen.

Stufenzahl	Erzeugungsfunktionen	Stufenzahl	Erzeugungsfunktionen
2	$x^2 + x + 1$	14	$x^{14} + x^{10} + x^6 + x + 1$
3	$x^3 + x + 1$	15	$x^{15} + x + 1$
4	$x^4 + x + 1$	16	$x^{16} + x^{12} + x^3 + x + 1$
5	$x^5 + x^2 + 1$	17	$x^{17} + x^3 + 1$
6	$x^6 + x + 1$	18	$x^{18} + x^7 + 1$
7	$x^7 + x^3 + 1$	19	$x^{19} + x^5 + x^2 + x + 1$
8	$x^8 + x^4 + x^3 + x^2 + 1$	20	$x^{20} + x^3 + 1$
9	$x^9 + x^4 + 1$	21	$x^{21} + x^2 + 1$
10	$x^{10} + x^3 + 1$	22	$x^{22} + x + 1$
11	$x^{11} + x^2 + 1$	23	$x^{23} + x^5 + 1$
12	$x^{12} + x^6 + x^4 + x + 1$	24	$x^{24} + x^7 + x^2 + x + 1$
13	$x^{13} + x^4 + x^3 + x + 1$	25	$x^{25} + x^3 + 1$

Tabelle 2 gebräuchliche Erzeugungsfunktionen für Schieberegister

Das Schieberegister setzt sich aus einer kaskadierten Folge von binären Speicherzellen (Flip-Flops) zusammen. Im Takt der Eingangsbits wird der binäre Zustand jeder Stufe zur nächsten Stufe zuführen. Das Schieberegister wird an bestimmten Stufen angezapft, die betreffenden Signale werden durch Modulo-2-Addition berechnet und die Summe wird zur ersten Stufe zurückgeführt[22].

Die Periode der Pseudozufallsfolge ist von der Stufenzahl des Schieberegisters, der Wahl von Anzapfungen und auch dem Anfangswert abhängig. Aber der Anfangswert darf nicht alles gleich 0 sein. Da der Ausgangswert mit einem Anfangswert mit 0 in allen Schieberegister unverändert bleibt, ist deshalb nicht sinnvoll. Die maximale Länge einer Pseudozufallsfolge mit einem n-stufigen Schieberegister wird wie folgt definiert:

$$l = 2^n - 1. \qquad (4\text{-}1)$$

Eine solche Folge, die die maximale Länge hat, wird als Maximalcode genannt.

Wenn die gewünschte Pseudozufallsfolge erzeugt wird, dann muss sie noch in das Format gleich eines TS-Datenstroms gewandelt werden, um die Daten am Eingang zu simulieren. Das Datenformat von TS-Datenströmen ist in Abbildung 9 dargestellt. TS-Datenströme basieren auf Rahmen. Jeder Rahmen hat 8 Pakete, jedes Paket hat insgesamt 188 Bytes und jedes Paket enthält Synchronisationsbyte 47h und 187 Bytes Informationsdaten[23].

Abbildung 9 Datenformat von TS-Datenströmen

4.2 Kanalcodierung

„Als Kanalcodierung bezeichnet man in der Nachrichtentechnik das Verfahren, digitale Daten bei der Übertragung über gestörte Kanäle durch Hinzufügen von Redundanz gegen Übertragungsfehler zu schützen.“[24]

Kanalcodierung wird auch Fehlerkorrekturcodierung genannt, sie bezieht sich hier auf dem Codierungsprozess von digitalen Fernsehsignalen. Mit der Kanalcodierung können Übertragungsfehler erkannt und korrigiert werden. Ihre wesentliche Zielstellung ist, die Zuverlässigkeit der Informationensübertragung zu verbessern, das heißt, die Störfestigkeit eines digitalen TV-Systems zu verbessern. Die Kanalcodierung ist ein signifikanter Unterschied zwischen der digitalen Kommunikation und der analogen Kommunikation.

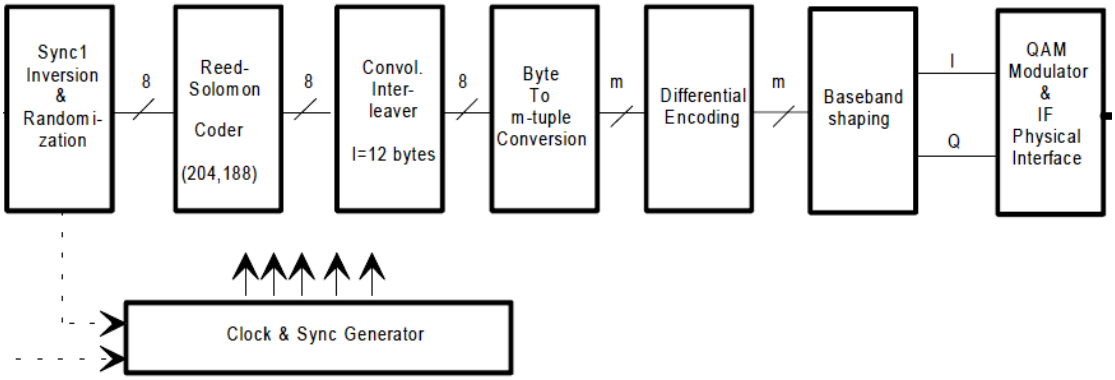


Abbildung 10 Kanalcodierung [25]

Die Kanalcodierung besteht gemäß dem DVB-Standard (Norm 300_429) [26] aus der Verwürfelung, der Reed-Solomon-Codierung, der Faltungsverschachtelung, der Differenzcodierung, dem Mapping und anderen digitalen Verarbeitungen. Der

Prozess ist in Abbildung 10 gezeigt.

4.2.1 Invertierung und Verwürfelung

4.2.1.1 Invertierung

Jeder TS-Datenstrom hat 8 Pakete. Ein Paket des TS-Datenstroms besteht aus einer konstanten Länge von 188 Bytes, wobei das erste Byte jedes Pakets das Synchronisationsbyte ist. Der Wert des Synchronisationsbytes ist immer gleich 47H. Dieser dient zum Kennzeichen des Beginns eines TS-Pakets.

Das erste Funktionsblock ist Invertierung. In diesem Block wird jedes achte Synchronisationsbyte invertiert (siehe Abbildung 11). Durch die Bitinversion ergibt sich dann aus 47H das invertierte Synchronisationsbyte B8H zur Synchronisation der Verwürfelung. Die anderen Synchronisationsbytes bleiben unverändert[27].

Die Pseudozufallsfolge für Verwürfelung muss jedes achte Paket initialisiert werden. Das invertierte Synchronisationsbyte B8H im Abstand von achte Pakten kann gewissermaßen eine Zeitmarke darstellen, die sowohl auf der Senderseite als auch auf der Empfangsseite für den Initialisierungsvorgang der Pseudozufallsfolge benötigt wird.

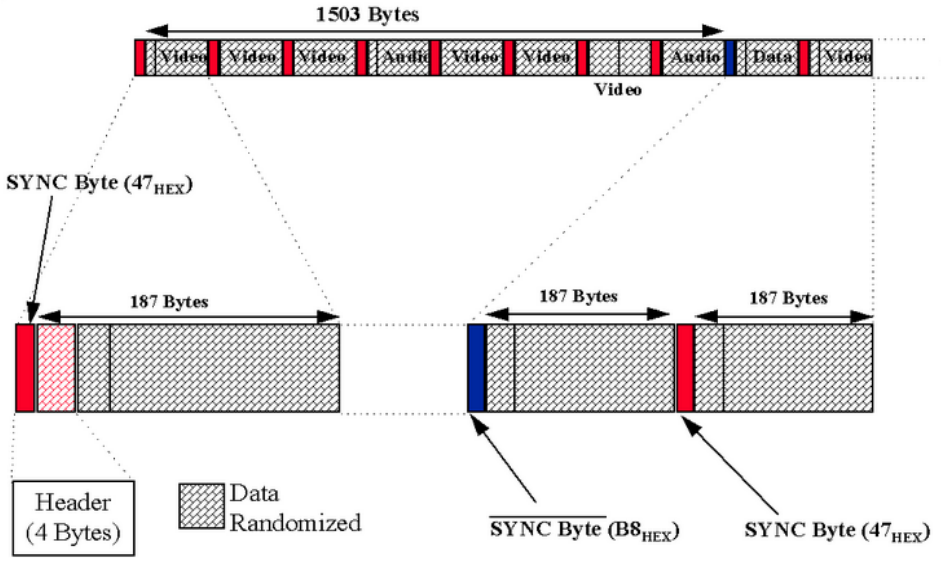


Abbildung 11 Invertierung des ersten Synchronisationsbyte jedes Rahmen[28]

4.2.1.2 Verwürfelung

Es gibt eine Annahme bei der Konstruktion eines Kommunikationssystems, dass die Eintrittswahrscheinlichkeit von *„0“* und *„1“* in den übertragenen Bitströmen gleich 50% ist. In der praktischen Anwendung basiert die Berechnung der Leistung eines Kommunikationssystems auch auf dieser Annahme. In den TS-Datenströmen können

lange Folgen von *„0“*- oder *„1“*-Elementen nach dem Codierungsprozess enthalten sein. Dann gibt es zwei Probleme. Einerseits wird die Voraussetzung des Systemdesigns zerstört, sodass sich die Leistung dieses Systems mit den gewohnten Methoden bestimmen lässt. Andererseits muss der Bittakt am Empfänger vor der Kanaldecodierung entnommen werden. Die Entnahme des Bittakts wird aus den Wechseln zwischen *„0“* und *„1“* in den TS-Datenströmen gewonnen, die bei aufeinander folgenden *„0“* oder *„1“* fehlen. Die Verwürfelung kann lange Folgen von *„0“* oder *„1“* und periodische Bitfolgen in dem übertragenden Bitstrom verhindern. Durch die Verwürfelung erhält das Datensignal einen stärkeren Zufallscharakter. Die Verwürfelung gewährleistet eine sichere Entnahme vom Bittakt in den Modulationspausen oder bei konstanten oder periodischen Signalen. Weiter ist mit der Verwürfelung eine Energieverwischung im Frequenzbereich verbunden, die die Gefahr von Nachbarkanalstörungen reduziert. Außerdem bietet die Verwürfelung einen Schutz gegen zufälliges Auftreten von Synchronisationsbytes[29].

4.2.1.3 Verwürfelung mit einer Pseudozufallsfolge

Um die gleiche Wahrscheinlichkeit von *„0“* oder *„1“* in den Datenströmen des DVB-Übertragungssystems in jedem Fall zu gewährleisten, sollten die binären TS-Datenströme durch logische Addition (modulo-2-Addition) mit einer Pseudozufallsfolge verwürfelt werden. Solche Zufallsfolgen können durch einige rückgekoppelten Schieberegister erzeugt werden (siehe Abschnitt 4.1). Die Verwürfelung verändert die ursprünglichen TS-Datenströme, deshalb braucht das Kommunikationssystem auch einen entsprechenden Entwürfelungsprozess auf der Empfängerseite nach der Fehlerkorrekturdecodierung, um die ursprünglichen TS-Datenströme wiederherzustellen.

4.2.2 Reed-Solomon Codierung

4.2.2.1 Allgemeines

„Reed-Solomon-Codes (kurz RS-Codes) sind leistungsfähige Codierungsverfahren, die beim Lesen oder Empfangen der mit ihnen codierten digitalen Daten erlauben, Fehler zu erkennen und zu korrigieren (Vorwärtsfehlerkorrektur). Bei nach dem DVB-Standard (Norm 300 429) ausgesendeten Fernsehsignalen wird beispielsweise ein RS-Code verwendet, der es dem Empfänger ermöglicht, die Bitfehlerrate des empfangenen Signals um mehr als sechs Zehnerpotenzen zu verbessern.

Reed-SolomonCodes sind besonders zur Korrektur von Burstfehlern bei der Datenübertragung geeignet.“ [30]

Die RS-Codes sind nicht-binäre und zyklische Codes. Aus der strukturellen Sicht ist der RS-Code ein (*n,k*) linearer Blockcode mit der Länge *n* für die Codewörter und der Länge *k* für die Datenwörter, wobei der Blockcode *n* Bytes Codewörter hat. Der aus *k* Bytes Informationsdaten und *r (r=n-k)* Bytes Symbolen für die Prüfsumme gemäß des Codierungsregels bestehende RS-Code wird als *RS(n,k)* gezeichnet. Der lineare Blockcode bedeutet, dass die Beziehung der Transformation zwischen Codewörter der Prüfsumme und Codewörter der Informationen linear ist. In der Fehlerkorrektur-codierung ist der Abstand zwischen Codewörtern, besonders der Mindestabstand von Codewörtern, ein Maß für die Entstörungsfähigkeit eines Codeworts. Je größer der Mindestabstand zwischen den Codewörtern ist, desto größer ist die kleinste Differenz zwischen zwei beliebigen Codewörtern und desto stärker ist die Entstörungsfähigkeit. Für alle lineare Blockcodes ist die Hammingdistanz der RS-Codes maximal, deswegen ist die Fehlerkorrekturfähigkeit der RS-Codes optimal.

RS-Codierung ist eine sehr effektive Blockcodierungstechnik. Sie unterscheidet sich von den anderen Blockcodierungstechniken, die auf einem einzigen Codewort basieren.

In den *RS(n, k)*-Codes wird das Eingangssignal in $k*m$ Bits eine Gruppe unterteilt, wobei jede Gruppe *k* Bytes Codesymbole umfasst, der Parameter *m* hat bei RS-Codes eine besondere Bedeutung. Er stellt die Zahl der Bit dar, die ein Codesymbol bilden. RS-Codes sind für die Korrektur von $t = r/2$ Fehlerssymbolen geeignet und RS-Codes können in der Regel mit einer Fehlerkorrektur von *t* fehlerhaften Bytes als RS*(n, k, t)* ausgedrückt werden.

In dem DVB-System wird die Kanalcodierung *RS(204,188,t=8)* (siehe Abbildung 12) verwendet[31]. Das heißt, $n= 204$ Bytes und $k= 188$ Bytes. Je 188 Informations-symbole werden 16 Bytes Prüfsummen benutzt, es ergeben sich aber Paketlänge von 204 Symbolen. Jedes Symbol umfasst $m= 8$ Bits (1 Byte), d.h. ein Codesymbol kann als ein Byte betrachtet werden.

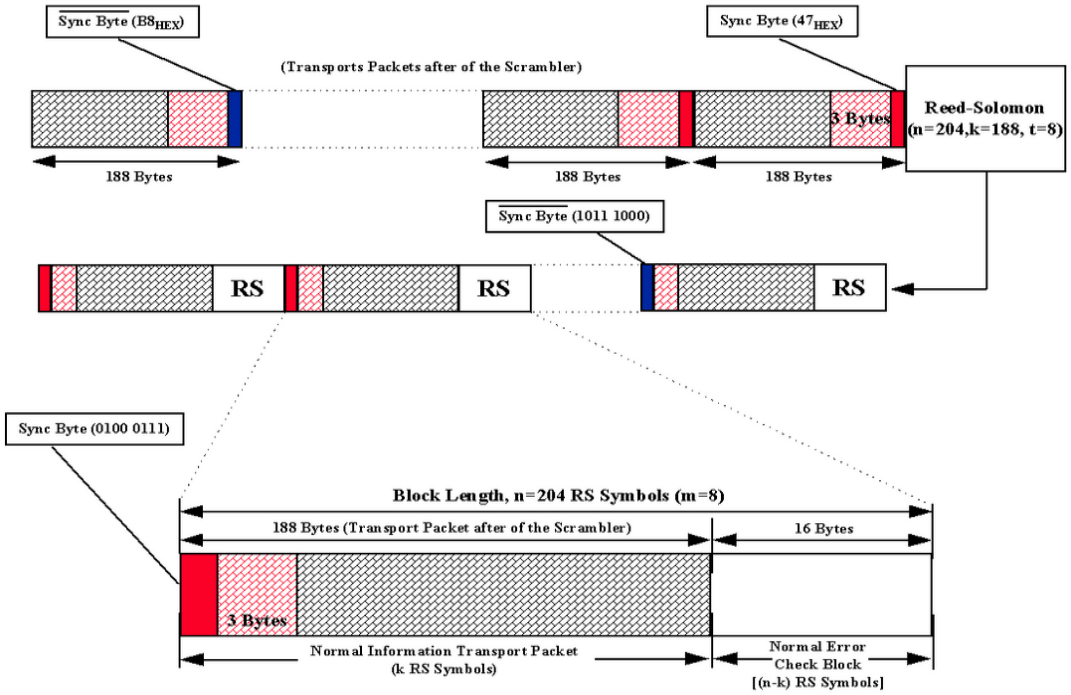


Abbildung 12 Format von RS-Codierung zufolge des DVB-C Standards[32]

4.2.2.2 Prinzip

RS-Codes können jegliche Fehlerkombinationen korrigieren, die kleiner als *t* sind. *t* wird nach folgender Formel ausgerechnet:

$$t = \left[\frac{d_{min}-1}{2}\right] = \left[\frac{n-k}{2}\right] \qquad (4\text{-}2)$$

Diese Konstruktion von RS-Codes erfolgt auf der Basis von Galoiskörper[33]. Bei der Beschreibung der Methoden zur Codierung bzw. Decodierung wird hier nur auf die Verwendung von GF(2^m) eingegangen.

Galoiskörper besteht aus einer Reihe von Nicht-Null-Elementen, die Anzahl von Elementen in dem Körper wird die Ordnung des Körpers genannt. Ein Galoiskörper mit *q* Ordnungen wird typischerweise als GF (*q*) dargestellt. Eine wichtige Eigenschaft des Körpers ist, dass es in dem Körper mindestens ein Element *a* gibt; *a* wird üblicherweise als ein primitives Element bezeichnet. Durch die Verwendung mit *a* können die andere *q-1* Elementen in diesem Körper ausgedrückt werden, d.h. *q-1* Nicht-Null-Elemente können durch $a, a^1, a^2, \ldots, a^{q-1}$ bezeichnet werden[34].

Das Hauptmerkmal des RS-Codes ist, dass die Codewörter und die Lösungen vom RS-Code in diesem endlichen Körper GF (*q*) sind. Da

$$x^{q-1} = \coprod_{a^i \epsilon \mathrm{GF\,(q)}} (x \cdot a^i) \qquad (4\text{-}3)$$

Die Länge des RS Codes ist $n = q - 1$.

Generatorpolynom vom RS-Code mit dem Mindestabstand δ ist:

$$g(x) = \coprod_{i=0}^{n-k-1}(x - a^{m_0+i}) \quad (4\text{-}4)$$

Wobei $\delta = D_{min} = n - k + 1$ ist, deswegen ist $n - k - 1 = \delta - 2$.

Normalerweise ist m_0=0, dann

$$\begin{aligned} g(x) &= (x - a^0)(x - a^1)\cdots(x - a^{\delta-2}) \\ &= x^{n-k} + g_1 x^{n-k-1} + \cdots + g_{n-k-1}x + 1 \end{aligned} \quad (4\text{-}5)$$

Der minimale Hanmming-Abstand ist:

$$\delta = 2t+1 \quad (4\text{-}6)$$

Wobei *t* die fehlerkorrekturfähige Länge des RS-Codes ist.

In der Codierung müssen die Rückkopplungsfaktoren in eine Dezimalzahl verwandelt werden, um sie einfacher berechnen zu können. Die obige Funktion *g(x)* wird Generatorpolynom von Codewörtern genannt. Für die Durchführung der Codierung wird auch ein anderes Polynom verwendet. Allgemein wird es als ein Körper-Erzeugungspolynom oder primitives Polynom bezeichnet. Beim DVB- System wird RS (204,188) verwendet, die Fehlerkorrekturfähigkeit einer Periode mit der Paketlänge von 204 Bytes ist:

$$t = (n - k)/2 \text{=(204-188)/2 = 8 Bytes} \quad (4\text{-}7)$$

Die benutzten Körpergeneratorpolynom und Generatorpolynom von Codewörtern in der Codierung nach dem DVB-C-Standard (Norm 300_429) sind:

$$„p(x) = x^8 + x^4 + x^3 + x^2 + 1 \quad (4\text{-}8)$$

$$\begin{aligned} g(x) &= (x - a^0)(x - a^1)\cdots(x - a^{15}) \\ &= x^{16} + a^{120}x^{15} + a^{104}x^{14} + a^{107}x^{13} + a^{109}x^{12} + a^{102}x^{11} + a^{161}x^{10} + a^{76}x^9 \\ &\quad + a^3x^8 + a^{91}x^7 + a^{191}x^6 + a^{147}x^5 + a^{169}x^4 + a^{182}x^3 + a^{194}x^2 + a^{225}x \\ &\quad + a^{120}" \end{aligned} \quad (4\text{-}9)$$

$$\begin{aligned} &= x^{16} + 59x^{15} + 13x^{14} + 104x^{13} + 189x^{12} + 68x^{11} + 209x^{10} + 30x^9 + 8x^8 \\ &\quad + 163x^7 + 65x^6 + 41x^5 + 229x^4 + 98x^3 + 50x^2 + 36x + 59 \end{aligned} \quad (4\text{-}10)$$

Zur Korrespondenz zwischen den Körper-Elementen und binären Daten siehe Anhang A.1.

Zur Realisierung des RS-Codierers sind diese Konstruktionen von Polynom-Addierer und Polynommultiplizier im Galoiskörper sehr wichtig. Ein Polynom-Addierer ist relativ einfach zu erreichen und kann direkt durch die Modulo-2-Addition von den beiden Polynomkoeffizienten realisiert werden. Ein Polynommultiplizier ist komplexer zu erreichen. a^{76} wird als ein Beispiel genommen, um das Realisierungs-prinzip des Polynommultiplizierers in GF(2^8) zu beschreiben. Jedes Element in dem

Körper GF(2^8) kann durch eine lineare Kombination von 1, a^1, a^2, a^3, a^4, a^5, a^6, a^7 ausgedrückt werden.

z.B. *B*= $b_7 \cdot a^7 + b_6 \cdot a^6 + b_5 \cdot a^5 + b_4 \cdot a^4 + b_3 \cdot a^3 + b_2 \cdot a^2 + b_1 \cdot a^1 + b_0$ (4-11)

Dann multipliziert man mit a^{76}, um Element *C* zu beschreiben.

C=*B*· a^{76}

= $b_7 \cdot a^{83} + b_6 \cdot a^{82} + b_5 \cdot a^{81} + b_4 \cdot a^{80} + b_3 \cdot a^{79} + b_2 \cdot a^{78} + b_1 \cdot a^{77} + b_0 \cdot a^{76}$

= $c_7 \cdot a^7 + c_6 \cdot a^6 + c_5 \cdot a^5 + c_4 \cdot a^4 + c_3 \cdot a^3 + c_2 \cdot a^2 + c_1 \cdot a^1 + c_0$ (4-12)

Nun wird *a* als die Wurzel von $p(x) = 0$ definiert. Nach der Formel (4-8) erhält man:

$a^8 = a^4 + a^3 + a^2 + 1$ (4-13)

Durch $a^8 = a^4 + a^3 + a^2 + 1$ und die Formel (4-12) können die c_i berechnet werden.

C= $b_7 \cdot a^{83} + b_6 \cdot a^{82} + b_5 \cdot a^{81} + b_4 \cdot a^{80} + b_3 \cdot a^{79} + b_2 \cdot a^{78} + b_1 \cdot a^{77} + b_0 \cdot a^{76}$

$= b_7 \cdot (a^7 + a^5 + a^4 + a^3 + a^1 + a^0)$

$+ b_6 \cdot (a^7 + a^6 + a^5 + a^1 + a^0)$

$+ b_5 \cdot (a^7 + a^6 + a^5 + a^2 + a^1 + a^0)$

$+ b_4 \cdot (a^7 + a^6 + a^5 + a^4 + a^3 + a^2 + a^0)$

$+ b_3 \cdot (a^7 + a^6 + a^5 + a^4)$

$+ b_2 \cdot (a^6 + a^5 + a^4 + a^3)$

$+ b_1 \cdot (a^5 + a^4 + a^3 + a^2)$

$+ b_0 \cdot (a^4 + a^3 + a^2 + a^1)$

$= a^7 \cdot (b_7 + b_6 + b_5 + b_4 + b_3)$

$+ a^6 \cdot (b_6 + b_5 + b_4 + b_3 + b_2)$

$+ a^5 \cdot (b_7 + b_5 + b_4 + b_3 + b_2 + b_1)$

$+ a^4 \cdot (b_7 + b_6 + b_5 + b_4 + b_3 + b_2 + b_1 + b_0)$

$+ a^3 \cdot (b_7 + b_4 + b_2 + b_1 + b_0)$

$+ a^2 \cdot (b_5 + b_4 + b_1 + b_0)$

$+ a^1 \cdot (b_7 + b_6 + b_5 + b_0)$

$+ a^0 \cdot (b_7 + b_6 + b_5 + b_4)$

$= c_7 \cdot a^7 + c_6 \cdot a^6 + c_5 \cdot a^5 + c_4 \cdot a^4 + c_3 \cdot a^3 + c_2 \cdot a^2 + c_1 \cdot a^1 + c_0$

Gemäß der obigen Gleichungen können die folgenden Gleichungen abgeleitet werden.

$c_7 = b_7 \hat{} b_6 \hat{} b_5 \hat{} b_4 \hat{} b_3$

$c_6 = b_6 \hat{} b_5 \hat{} b_4 \hat{} b_3 \hat{} b_2$

$c_5 = b_7 \hat{} b_5 \hat{} b_4 \hat{} b_3 \hat{} b_2 \hat{} b_1$

$$c_4 = b_7 \hat{} b_6 \hat{} b_5 \hat{} b_4 \hat{} b_3 \hat{} b_2 \hat{} b_1 \hat{} b_0$$
$$c_3 = b_7 \hat{} b_4 \hat{} b_2 \hat{} b_1 \hat{} b_0$$
$$c_2 = b_5 \hat{} b_4 \hat{} b_1 \hat{} b_0$$
$$c_1 = b_7 \hat{} b_6 \hat{} b_5 \hat{} b_0$$
$$c_0 = b_7 \hat{} b_6 \hat{} b_5 \hat{} b_4$$

Nach dem obigen Verfahren können beliebige Potenzen von α berechnet werden, womit der Polynommultiplizier realisiert wird.

Die Schaltung vom Körper-Generatorpolynom ist wie Abbildung 13.

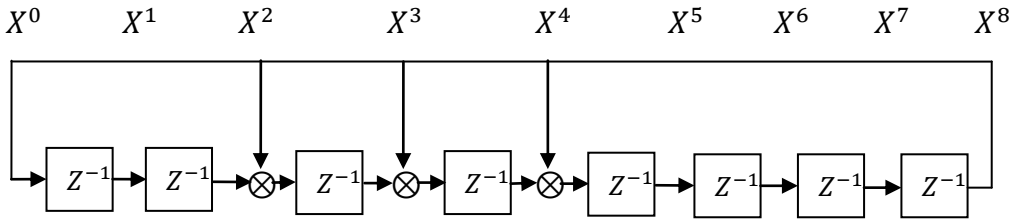


Abbildung 13 Feedback circuit polynomial

$$p(x) = a^8 + a^4 + a^3 + a^2 + 1 \qquad (4\text{-}14)$$

Diese Schaltung erzeugt $2^m - 1$ *(m=8)* nicht-Null-Elemente in dem Galoiskörper. Von dieser Abbildung 13 ist es klar, dass die Schaltungsanschlüsse gleich wie die Polynomkoeffizienten sind. Der Anfangszustand des Polynoms darf nicht gleich Null sein, zum Beispiel wird 8'b1000_0000 genommen. Jeder Takt erreicht eine Verschiebung nach rechts, und $2^m - 1$ kann periodisch im Status des Schieberegisters erscheinen.

4.2.3 Faltungsverschachtelung

4.2.3.1 Allgemeines

Die RS-Codes haben eine sehr starke Fähigkeit, Burstfehler zu korrigieren. Aber durch die Verschachtelungsoperation von den Daten kann die Korrekturfähigkeit weiter verbessert werden. Die Datenverschachtelung braucht keine zusätzlichen Codewörter für Fehlerkorrektur, sondern benutzt die Methode, dass die Übertragungsordnung der Codewörter verändert wird. Der vom RS-Codierer ausgegebene Datenstrom wird auf der Senderseite so umgerechnet, dass die aufeinander folgenden Bits auf die weiter auseinanderliegenden Positionen getrennt werden[35]. Durch die Faltungsverschachtelung können die Korrelation zwischen den Daten geschwächt und können die Fehler verteilt werden. Diese Effekte können zu weniger Fehler in einem Übertragungsprozess führen.

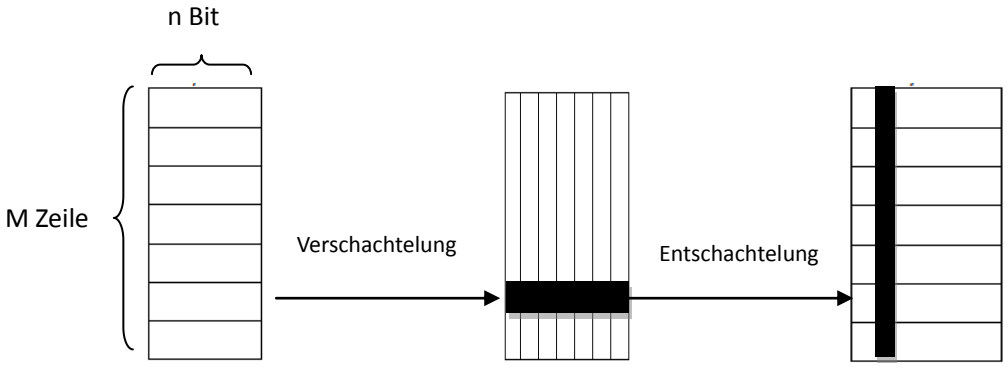


Abbildung 14 das Prinzip der Datenverschachtelungstechnik

Das Prinzip der Datenverschachtelungstechnik wird in Abbildung 14 gezeigt. Diese Abbildung beschreibt, dass $m*n$ Bits eine Gruppe bilden[36]. Jede horizontale Linie hat n Bits und insgesamt hat m Zeilen (je Zeile ein Codewort). Die $m*n$ Bits werden gleichzeitig in ein Register geschrieben. Danach werden die Bits aus den senkrechten Spalten herausgelesen, gegebenenfalls auf einen Träger moduliert und anderen Schaltungen zugeführt. Im Empfänger werden die Daten im Entschachtelter im gleichen Rechteckformat gespeichert und aus den horizontalen Zeilen ausgelesen.

Wenn es bei der Übertragung einen Burst-Fehler gibt, dann werden die Fehlerbits im Entschachtelungsregister im Bitstrom verteilt werden. Hier wird die Spaltenzahl n als die Verschachtelungstiefe bezeichnet. Je größer die Verschachtelungstiefe ist, desto stärker ist die Verteilung von Burstfehlern, aber desto längere Verzögerungszeiten werden erfordert. Die Verschachtelungstiefen sind im ATSC-Standard 52, und im DVB-T- Standard 12.

5.2.3.2 Prinzip

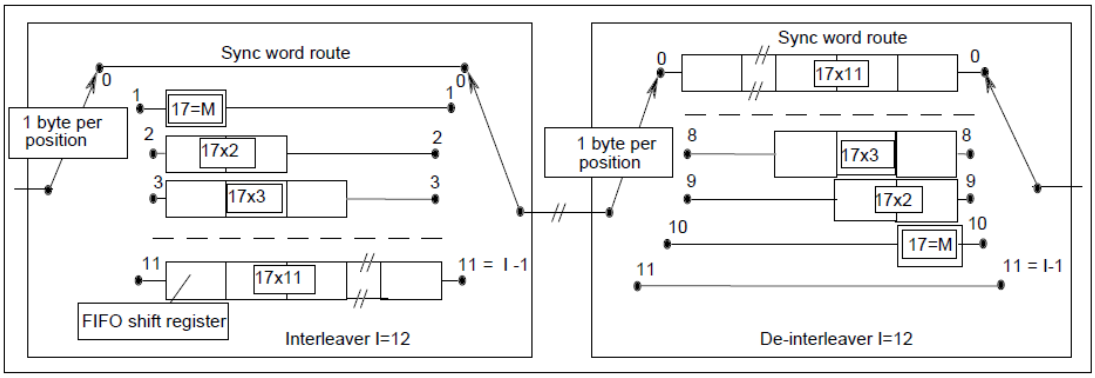


Abbildung 15 Prinzip der Verschachtelung und Entschachtelung[38]

Abbildung 15 zeigt die schematische Darstellung der Verschachtelung und Entschachtelung. Die Faltungsverschachtelung erfolgt anhand des Forney Ansatzes. Das Prinzip von Forney und Ramsey Typ III bei I = 12 ist kompatibel. Die verschachtelten Rahmen bestehen aus geschützten Rahmen, die durch die

Burst-Fehler gestört sind. Der Verschachteler umfasst *I* = 12 Zweige und die Zweige werden durch einen Eingangsschalter periodisch mit den Eingabedaten verbunden. Jeder Zweig sollte ein FIFO- Schieberegister haben, die Tiefe der FIFO ist M * j („*hier M = 17 = N / I, N = 204 = error protected frame length, I = 12 = interleaving depth, j = branch index"*) [37].

Der Schalter von Eingang und Ausgang sollte synchronisiert werden, deshalb sollte das FIFO-Schreiberegister im ersten Zweig ein Synchronisationsbyte enthalten. In diesem Zweig gibt es keine Verzögerung, sodass das Synchronisationsbyte 47H und das invertierte Synchronisationsbyte B8H direkt übertragen werden können.

Durch die schematische Darstellung kann man den Arbeitsprozess des Faltungs-verschachtelers und Entschachtelters erkennen: Das Eingangssymbol im Verschachtelter tritt im *B-ten* Zweig ein. Jeder Zweig hat einen Verzögerer und verzögert um unterschiedliche Symbol-Perioden. Der erste Zweig hat keine Verzögerungszeit, der zweite Zweig verzögert *M (M=17)* Symbol-Perioden, und der dritte Zweig verzögert *2M (2M=34)* Symbol-Perioden, so kann der *B-te* Zweig *(B-1)∗M* Symbol-Perioden verzögern. Gleichzeitig werden die verzögerten Daten entsprechend dem Takt am Ausgang des Verschachtelers ausgegeben. Die Verzögerung der Symboldaten von der Faltungsverschachtelung jedes Zweigs ist

$$d_i = (i-1) \cdot M, \quad i=1,2,3,\ldots,B \tag{4-15}$$

4.3 Byte to symbol mapping

4.3.1 Differenzcodierung

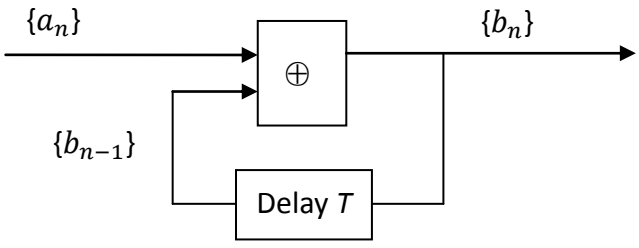


Abbildung 16 Differenzcodierung [39]

Mit der Differenzcodierung wird die Eingangsbitfolge $\{a_n\}$ in eine neue Folge $\{b_n\}$ umgewandelt. Abbildung 16 zeigt das Realisierungsschema der Differenzcodierung bei Duobinärsignalen. Durch die folgende Beziehung:

$$b_n = a_n \oplus b_{n-1} \tag{4-16}$$

wird die Eingangsbitfolge differenzcodiert. D.h. der Ausgangswert b_n wird durch Modulo-2-Addition von dem aktuellen a_n-Wert und dem aus vorhergehendem

Intervall erhaltenen Wert b_{n-1} gewonnen[40].

4.3.2 Umwandlung von Byte zu m-Symbolen

In der MQAM-Modulation entspricht ein Symbol *m* Bits ($m=log_2 M$). Wenn *M* {16, 32, 64, 128, 256} ist, dann ist *m* gleich 4, 5, 6, 7, 8 Bits. Durch die Umwandlung von Byte zu *m*-Symbolen kann ein entsprechender Wert von *m* ausgewählt werden, somit wird eine entsprechende MQAM-Modulation erhalten.

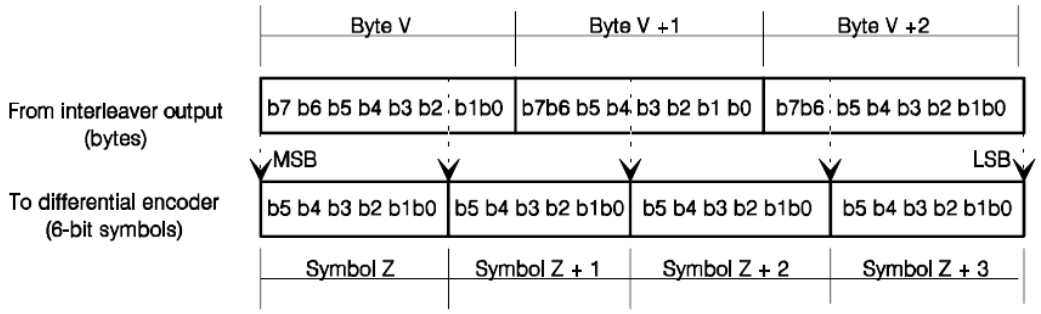


Abbildung 17 Umwandlung von Byte zu m-Symbolen für 64QAM[41]

64QAM (*m*=6, *k*=3, *n*=4) wird als ein Beispiel genommen, um den Mapping-Prozess zu beschreiben. Das Mapping-Verfahren ist, dass das höchste Bit des Symbols *Z* (siehe Abbildung 17) dem höchsten Bit des Bytes *V* entspricht, dann entspricht das nächste höchste Bit des Symbols Z dem nächsten höheren Bit des Symbols V usw. Wie in Abbildung 30 dargestellt, können jeweils 3 Bytes Daten zu vier 6 Bits Daten konvertiert werden.

4.3.3 Mapping

Die Zuordnung sollte einer bestimmten Abbildungslogik folgen. Sie verbindet die ursprünglichen Symbole mit einen entsprechenden Konstellationspunkt in einem Konstellationsdiagramm. Die Koordinatenwerte(natürlicher Code) dieses Punktes in der Quadratur- und Inphase-Achse sind die Ergebnisse nach dem Mapping. Die Ausgangswerte von den Natürlichen Codes in den beiden orthogonalen Achsen sind die sogenannten *I*- und *Q*-Basisbandsignale. Um die ausgezeichneten Modulations-signale zu erhalten, sollte der minimale Abstand zwischen Konstellationspunkten so groß wie möglich sein. Aber der minimale Abstand wird auch durch die Anzahl der Eingangsbits des Formungsfilters eingeschränkt. Das Konstellationsdiagramm kann verändert werden, aber die Breite der Eingangssymbole vom Formungsfilter ist im Allgemeinen fixiert, deshalb kann eine geeignete Mapping-Sammlung gewählt werden, um eine gute Nutzung der Genauigkeit von einem Formfilter zu erzielen.

Nachfolgend gibt es eine mögliche Koordinatenzuordnung der Konstellationspunkte

in dem zweidimensionalen Koordinatensystem. Bei diesem Verfahren ist der minimale Abstand zwischen Konstellationspunkten optimal.

16QAM: $\{\pm 5, \pm 15\}$;

32QAM: $\{\pm 3, \pm 9, \pm 15\}$;

64QAM: $\{\pm 2, \pm 6, \pm 10, \pm 14\}$;

128QAM: $\{\pm 1, \pm 3, \pm 5, \pm 7, \pm 9, \pm 11\}$;

256QAM: $\{\pm 1, \pm 3, \pm 5, \pm 7, \pm 9, \pm 11, \pm 13, \pm 15\}$;

4.4 Basisbandfilter

Vor der Modulation sollten die *I*- und *Q*-Signale durch ein Quadratwurzel-Raised-Cosine-Filter geformt werden. Die Übertragungsfunktion des Quadratwurzel-Raised-Cosine-Filters wird durch den folgenden Ausdruck definiert[42]:

$$\begin{cases} H(f) = 1 \quad \text{für} \quad |f| < f_N(1-\alpha) \\ H(f) = \left\{\frac{1}{2} + \frac{1}{2} \cdot \sin\frac{\pi}{2f_N}\left[\frac{f_N - |f|}{\alpha}\right]\right\}^{1/2} \quad \text{für} \quad f_N(1-\alpha) \leq |f| \leq f_N(1+\alpha) \\ H(f) = 0 \quad \text{für} \quad |f| > f_N(1+\alpha) \end{cases} \tag{4-17}$$

Wobei $f_N = \frac{1}{2T_s} = \frac{R_S}{2}$ der Nyquist-Bandbreite entspricht. Als Maß für die Stärke des Abfalls von H(f) wird der sogenanten Roll-off- Faktor [43]

$$\alpha = \frac{\Delta f}{f_N} = \frac{f_Ü - f_N}{f_N} \tag{4-18}$$

definiert. Die Werte von α können zwischen *0* und *1* liegen. Für α=0 ergibt sich bei f=f_N ein abrupter Abfall der Übertragungscharakteristik. Für $\alpha > 0$ ergibt sich gegenüber der Nyquist-Bandbreite f_N eine erhöhte Übertragungsbandbreite von

$$f_Ü = f_N \cdot (1 + \alpha). \tag{4-19}$$

Für DVB-C ist ein Roll-off-Faktor α= 0,15 erforderlich.

5 Entwicklung einer QAM-Modulator-Hardware

Als Hardware-Plattform dieser Konstruktion wurden ein FPGA von der Firma ALTERA und der AD9857 von der Firma Analog Devices verwendet. Der AD9857 realisiert die Interpolationsfilterung, Quadraturmodulation, DA-Wandlung und andere Funktionen von den digitalen Basisbandsignalen in der digitalen Domäne. Das FPGA-Entwicklung-Board DE2-115 gehört zu der Familie Cyclone IV von der Firma ALTERA. Dieses Entwicklungsboard wurde ausgewählt, um die digitale Basisbandsignalverarbeitung und die Kontrolle des AD9857 zu erreichen. Zur Software-Entwicklung werden Quartus II und Modelsim-Altera verwendet. Als Hardwarebeschreibungssprache wurde Verilog HDL im Quartus II verwendet, um das digitale Basisband-signalverarbeitungsmodul (Erzeugung von PRBS und TS-Datenströmen, Kanal-codierung, Umwandlung von Byte zu m-Symbolen, Basisbandfilter etc.) und den Kommunikationsmodul zwischen AD9857 und FPGA (die serielle Kommunikations-schnittstelle und die parallele Datenübertragung) zu verwirklichen. Es wurden alle Module simuliert und die Simulationsergebnisse erfüllt an die Anforderungen an diese Konstruktion.
In diesem Kapitel wird zuerst über die Hardware diskutiert.

5.1 Einführung über FPGA

Ein FPGA verfügt über viele Logikelemente, hauptsächlich FlipFlops (FF) und kombinatorische Logikschaltungen, die vor den FlipFlops angeordnet sind. Diese Schaltungenen sind Verknüpfungen mit verschiedenen Logikgattern oder Nachschla-getabellen (LUTs : Look-Up-Table). Diese kombinatorische Logikschaltungen können durch einen elektronischen Schalter nach der vom Entwickler gewünschten Funktion miteinander verbunden werden.
Ein FPGA basiert auf dem Konzept von Logikzellenfeldern LCA (Logic Cell Array). Diese beinhalten den konfigurierbaren Logikblock CLB (Configurable Logic Block), das Ausgangs- und Eingangsmodul IOB (Input Output Block) und die interne Verdrahtung (Interconnect). Ein FPGA verwendet kleine Nachschlagetabellen, um eine beliebige kombinatorische Funktion beispielsweise NAND, XOR, AND, Multiplexer usw. aus den Eingangssignalen zu realisieren[44]. Die Anzahl von den Eingangssignalen pro Nach-schlagetabelle ist von dem FPGA abhängig und ist meistens zwischen 4 und 6. Jede Nachschlagetabelle wird mit dem Eingangsanschluß eines D-Flip-Flops verbunden, danach treiben diese D-Flip-Flops andere Logikschaltungen oder IOB an, wodurch

eine grundlegende Logikeinheit zusammengesetzt wird, die nicht nur sequentielle Logikfunktionen, sondern auch kombinationslogische Funktionen realisieren kann [45]. Diese FlipFlops werden benutzt, um die Werte vom Signal zwischenzuspeichern, damit sie im nächsten Taktimplus weiterverarbeitet werden können. Meistens ist das Verhältnis zwischen der Anzahl von Nachschlagetabellen und der Anzahl von Flip-Flops 1:1. Gegenwärtige FPGAs verfügt über bis zu einige zehntausend Logik-elemente[46].

Die grundlegende Struktur des FPGA-Chips ist in der Abbildung 18 dargestellt. Weiterhin ist eine Schaltbox (Switch Box, Abkürzung als SB) enthalten, um die horizontalen und vertikalen Verbindungsressourcen umzuschalten. Noch dazu gehört eine Anschlussbox (Connection Box, Abkürzung als CB), die mit dem Eingang und Ausgang von CLB verbindet.

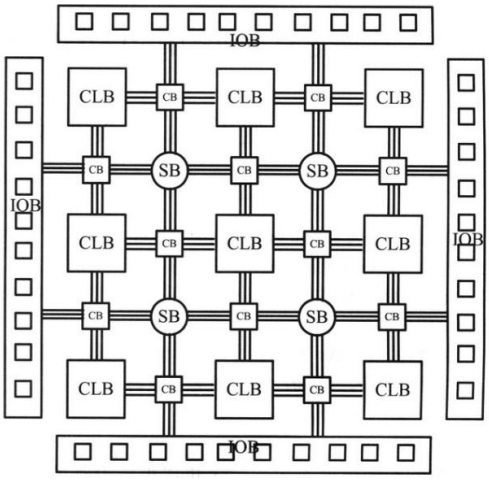


Abbildung 18 interne Struktur von FPGA

5.1.1 FPGA-Arbeitsprinzip

Diese Module sind durch Verwendung eines Metalldrahts miteinander oder mit dem I/O Modul verbunden. FPGA-Logik wird durch Laden der Programmierung von Daten nach der internen statischen Speichereinheit konfiguriert. Die Werte, die in der Speichereinheit gespeichert sind, können die logischen Funktionen der Logikeinheit bestimmen. Gleichzeitig können sie die Verbindungen zwischen den verschiedenen Modulen oder die Verbindungen zwischen einem Modul und I/O festlegen.

Die Konfigurationsinformationen im FPGA werden in der Regel aus einer internen statischen Speichereinheit laden. Weiterhin können die meisten FPGAs für diesen Konfigurationsvorgang mehrere Modie (z.B. seriell, parallel, Master/Slave) anbieten. FPGA ist ein auf SRAM basierendes Gerät, wenn die Versorgungsspannung abge-

schaltet wird, dann werden die Inhalte im SRAM-Speichereinheit sorfort verloren. Wenn ein benutzer FPGA verwenden möchte, muss FPGA bei jedem Einschalten neu konfiguiert werden. Bevor FPGA vollständig betrieben wird, braucht noch einige Millisekunden bis zu einigen Sekunden[47]. Schließlich können die durch FPGA verwirklichten Funktionen bestimmt werden.

5.1.2 FPGA-Entwicklungsboard

Das Altera DE2-115-Development and Education Board V3 ist in Abbildung 19 dargestellt.

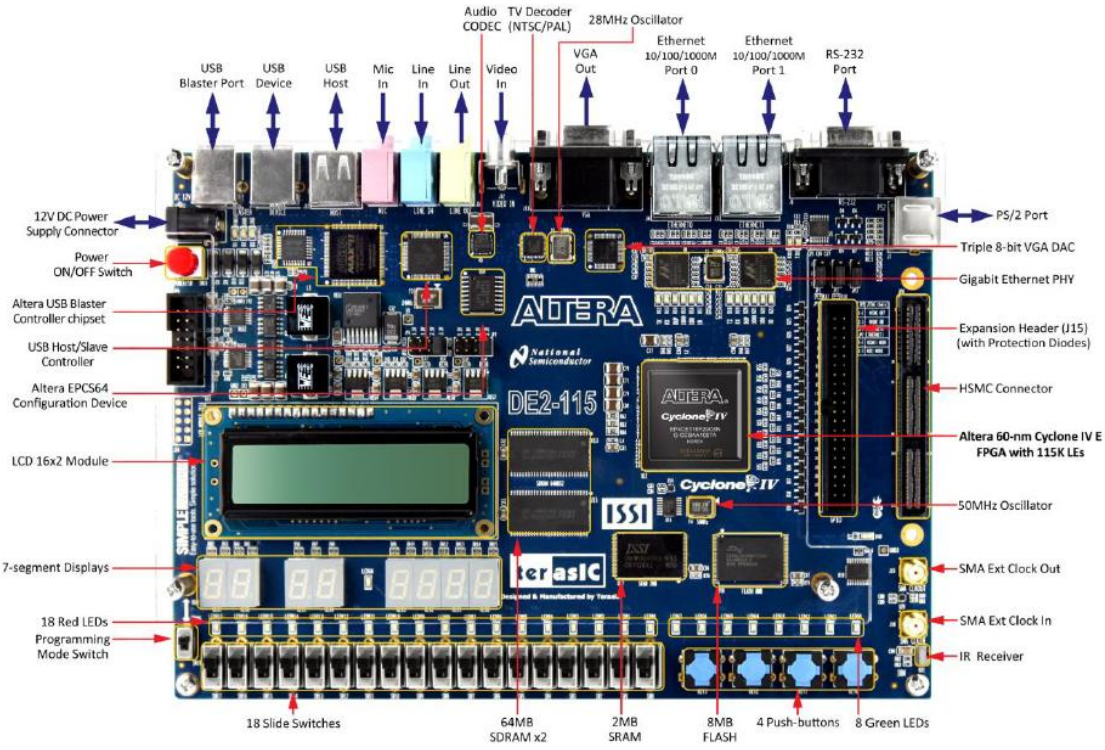


Abbildung 19 Altera DE2-115 - Development and Education Board V3 [48]

Das Altera Cyclone IV FPGA ist preiswert und hat einen niedrigen Stromverbrauch. Es bietet 114,480 Logikelemente (LE) und hat gegenüber anderen FPGAs zwei Vorteile [49]:

1. Reduktion der Systemkosten

Alle Cyclone IV FPGA brauchen nur zwei Versorgungsleitungen, dadurch werden die Stromversorgung vereinfacht, die Kosten für die Leiterplatten reduziert, die die Leiterplattenfläche verringert, sowie die Entwicklungszeit verkürzt.

2. Reduktion des Stromverbrauchs

Durch die Verwendung eines 60-nm Low-Power-Prozesses kann die interne Core-Spannung reduziert werden und im Vergleich mit den vorherigen Generationen von

Produkten hat Cyclone IV E eine 25% geringe Stromaufnahme.

5.1.3 Eigenschaften vom FPGA-Entwicklungsboard

- Altera Cyclone IV EP4CE115F29C7 FPGA mit 115000 LEs
- Altera Serial Configuration device EPCS64
- USB Blaster zur Programmierung und für User API
- User-Applikations und Steuerungssoftware
- 128 MB SDRAM, 8 MB Flash Memory, 2 MB SRAM
- SD Card Sockel, RS-232, 2x GBit-Ethernet,
- 8bit VGA, 24bit Audio CODEC
- TV Decoder (NTSC/PAL), IrDA, USB (Host + Slave)
- 18 Schalter, 4 Drucktasten, PS2-Maus/Tastatur-Anschluss
- 27 LEDs, 16x2-LCD-Display, 8 7-Segment-Anzeigen
- IR-Fernsteuerung
- HSMC-Stecker
- Viele Beispiele mit Source Code wie TV, SD Music Player, Audiorecorder[50]

5.1.4 Konfigurationsarten vom FPGA-Entwicklungsboard

Es gibt 2 konfigurationsarten im FPGA-Entwicklungsboard: JTAG und AS[51].

5.1.4.1 JTAG Modus

Im JTAG-Modus können die Konfigurationsinformationen direkt in das FPGA Entwicklungsboard, also in die SRAM-Speichereinheit heruntergeladen werden.

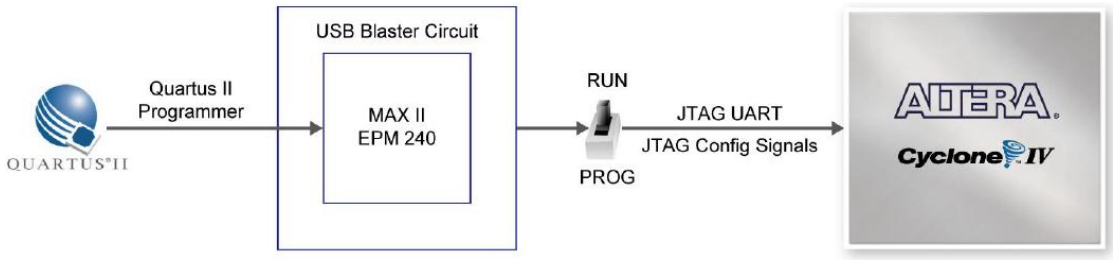


Abbildung 20 JTAG-Modus

5.1.4.2 AS Modus

Im AS-Modus werden die Konfigurationsinformationen in einem Konfigurations-device EPCS64 gespeichert.

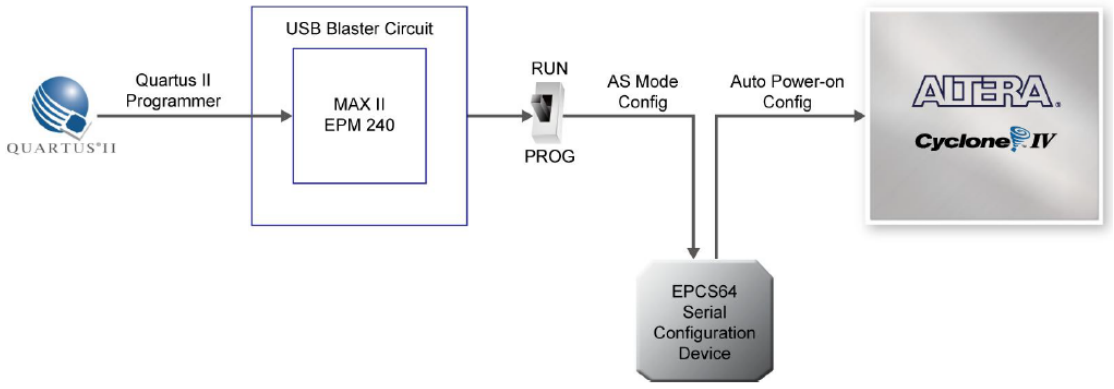


Abbildung 21 AS-Modus

5.2 Digitaler Aufwärtswandler AD9857

Um die QAM-Modulation zu realisieren, wird der Chip AD9857 von der Firma Analog Devices benutzt. Der AD9857 enthält Basisband-Eingänge für die quadratischen IQ-Daten und eine direkte digitale Synthesetechnologie.

Der AD9857-Chip integriert einen schnellen 32 Bit-Direkt-Digital-Synthesizer (DDS), einen schnellen 14 Bit-Digital-Analog-Wandler (DAC), eine Takt-Multiplizierschaltung, einen digitalen Filter und andere DSP-Funktionen auf einem einzigen Chip, um einen vollständigen quadratischen digitalen Aufwärtskonverter zu bilden. Die Abtastrate ist 200MPS. Dieser Chip kann digitale Ausgangsfrequenzen von DC bis 80MHz ausgeben. Der AD9857 hat gute dynamische Eigenschaften. Wenn ein analoges Signal mit der Frequenz von 65MHz ausgegeben wird, weist der AD9857 einen Dynamikbereich von 80dB auf. Der AD9857 verfügt über einen 200 MHz internen Takt, integriert mit einem 4-20 fachen programmierbaren Takt-Multiplizier. Er kann einen hochpräzisen Systemtakt als Eintakt oder Differenztakt anbieten. Der AD9857 umfasst vier Sätze von programmierbaren Registern („Profiles“). Die Steuerschnittstelle ist eine 10MHz serielle Schnittstelle und kompatibel mit SPI. Der AD9857 hat ein inverses CIC-Filter, ein programmierbares CIC-Filter und noch ein festes CIC-Filter, mit Hilfe der eine schnelle und saubere Abtastrate realisiert werden kann. Außerdem hat der AD9857 eine inverse Funktion von SINC, die dient dazu, dass der durch die DAC-Schaltung verursachte Schwund kompensiert wird. Der AD9857 kann auch eine ausgegebene Amplitude mit 8 Bits Daten kontrollieren. Die Spannung vom AD9857 ist 3,3 V DC, Betriebstemperatur ist -40～ +85℃[52].

Die maximale Taktfrequenz vom AD9857 kann bis zu 200 MHz sein. Um Störungen zu vermeiden, sollte die maximale Ausgangsfrequenz vom AD9857 kleiner als 40% von der inneren Systemclock sein. Die Anforderung dieser Konstruktion ist die Ausgangs-

frequenz von 5 bis 65MHz, daher kann an die Anforderung erfüllt werden. AD9857 kann auch die Integration und Stabilität des digitalen Kommunikationssystems erhöhen.

Durch den AD9857 können GMSK, CDMA, OFDM und mehrere QAM-Modulationen realisiert werden.

5.2.1 Interne Struktur von AD9857

Abbildung 22 zeigt die innere Struktur des AD9857, wenn er im quadratischen Modus arbeitet. Der digitale Teil ist in eine Digitalverarbeitungseinheit und eine Logiksteuereinheit unterteilt. Die Digitalverarbeitungseinheit besteht aus den Dateneingänge, dem CIC und inversen CIC-Filter, dem festen Interpolationsfilter, dem Quadratur-Modulator, DDS, dem inversen SINC-Filter, dem Amplitudenkontroller für Ausgangsdaten, dem DA-Wandler und dem Multiplizierer für Referenztakt usw.

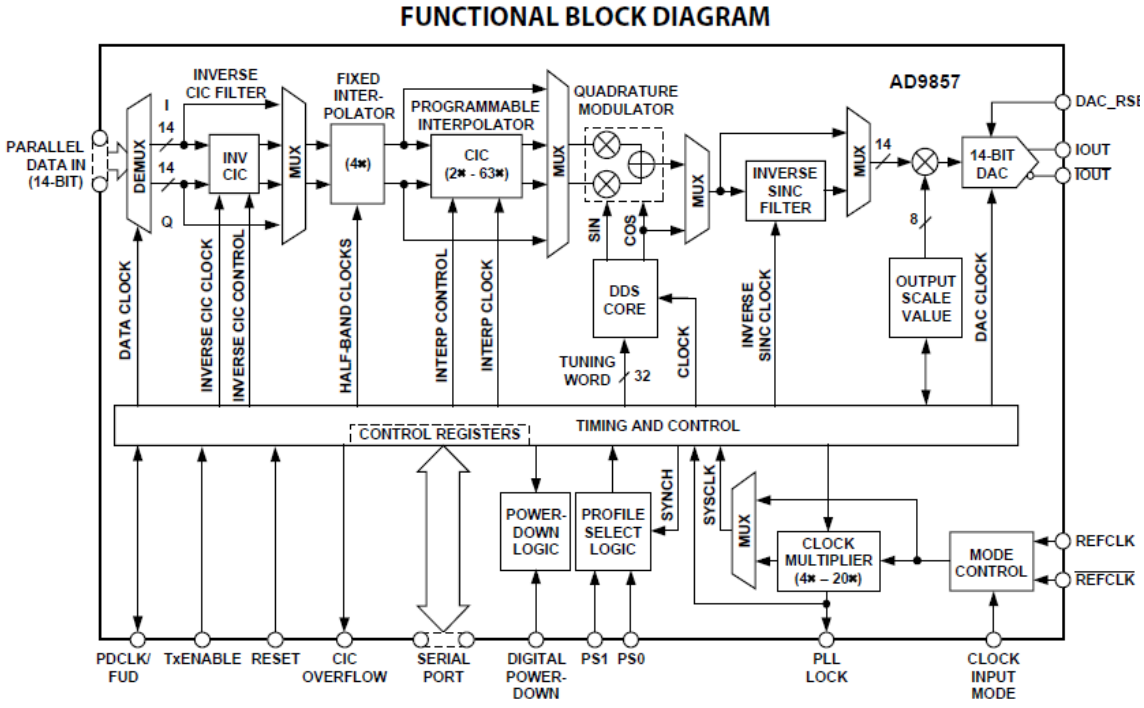


Abbildung 22 Interne Struktur von AD9857[53]

Dateneingänge: zuerst müssen die digitalen Basisbandsignale in 14 Bits parallele Daten umgewandelt werden. Beim Arbeiten in dem quadratischen Modus werden die IQ-Komponenten des Basisbandsignals abwechselnd eingegeben. Dann muss sicherstellt werden, dass die IQ-Komponenten mit dem Eingangstakt PDCLK synchron sind, sodass die Eingangsdaten in zwei parallelen I/Q-Datenströmen umgewandelt werden können, um die I/Q- Datenströmen zur nächsten Schaltung zu senden.

Die CIC und inverses CIC-Filter: CIC (interpolierte kaskadierte Integrator-Kammfilter)

ist ein programmierbares Oversamplingfilter (2-fach bis 64-fach). Seine Struktur ist in Abbildung 23 dargestellt. Wegen der Tiefpasscharakteristik vom CIC-Filter sollte ein inverses CIC-Filter vor dem CIC-Filter benutzt werden, um zu kompensieren. Durch die Verwendung der Kombination von einem festen Interpolationsfilter und einem programmierbaren CIC-Filter können 8-fache bis 256-fache Interpolationsraten bereitgestellt werden. Die Interpolationsraten sind der Schlüssel zum Erreichen der beliebigen Basisbandsymbolraten. Ein inverses kaskadiertes Interpolationsfilter wird verwendet, um die durch ein kaskadiertes Interpolationskammfilter gedämpfte Amplitude hauptsächlich vorzukompensieren. Das inverse kaskadierte Interpolationsfilter ist ein digitales FIR-Filter. Der Amplitudenfrequenzgang von einem inversen kaskadierten Interpolationsfilter ist dem Amplitudenfrequenzgang von einem kaskadierten Interpolationskammfilter entgegengesetzt.

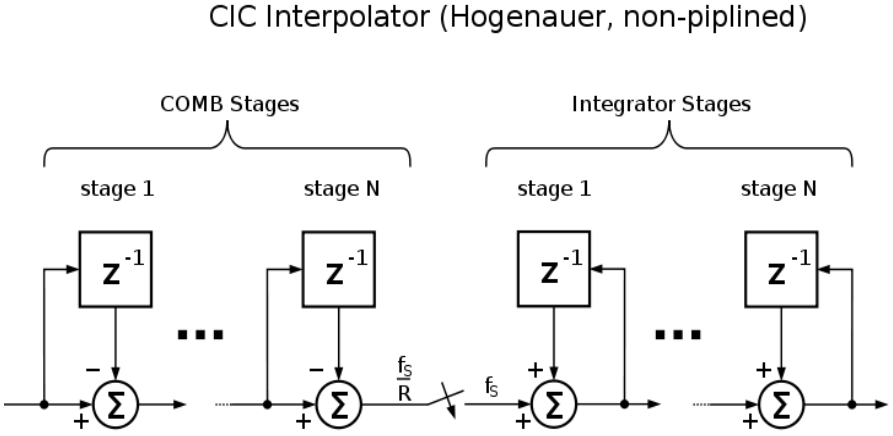


Abbildung 23 Die Struktur eines CIC-Filters [54]

Festes Interpolationsfilter: Das feste Interpolationsfilter wird durch zwei *Halbbandfilter* gebildet. Das feste Interpolationsfilter hat einen Amplitudenfrequenzgang wie das CIC-Filter, hat also auch Tiefpasscharakteristik. Der Interpolationskoeffizient des festen Interpolationsfilters ist gleich vier.

Quadraturmodulation: Quadraturmodulation wird benutzt, um die orthogonalen modulierten Signale zu bekommen.

DDS: DDS wird benutzt, um das Sinussignal und das Cosinussignal des Trägers zu erzeugen. Zwischen dem Träger f_{out}, und den Kontrollwörtern der Frequenz „FTWORD“ und dem Systemtakt „SYSCLK“ besteht eine Beziehung, wie folgende

„Halbbandfilter[55]: Ein Halbbandfilter zeichnet sich dadurch aus, dass der „halbe“ Frequenzbereich Durchlassbereich ist, während die zweite Frequenzhälfte Sperrbereich ist. Der Übergangsbereich zwischen Durchlass und Sperrbereich liegt also symmetrisch zu $\omega = \pi/2$. Man setzt daher Halbbandfilter bei der Umsetzung der Abtastrate um einen Faktor 2 ein. Macht man das Halbbandfilter punktsymmetrisch bezüglich $\omega = \pi/2$, so zeigt sich, dass die Impulsantworten der Halbbandfilter für alle geraden Indizes bis auch einen Wert alle Null sind.“

Gleichung (5-1) gezeigt.

$$f_{out} = (\mathrm{FTWORD} * SYSCLK)/2^{32} \quad (5\text{-}1)$$

Wobei „FTWORD“ die Dezimalzahl von 0 bis $(2^{32} - 1)$ ist.

Multiplizierer für Ausgangsamplitude: Dieser Multiplizierer wird benutzt, um die endgültigen Ausgangssignalamplituden einzustellen. Dessen Wert wird durch das entsprechende programmierbare Register bestimmt, der Bereich ist von 0 bis 1,9921875. Der Standardwert ist B5h entsprechend wie $\sqrt{2}$ in Dezimalzahl, um den 3dB- Eigenverlust der Quadraturmodulation zu kompensieren.

14Bit-DAC: Der 14Bit-DAC wird genutzt, um die digitalen Daten in analoge Signale umzuwandeln. Die Eingabedaten des DA-Wandlers im AD9857 sind abgetastete Trägerdatenströme. Durch die DA-Wandlung können Interferenzsignale bei den Frequenzen [n*SYSCLK +/-fout] (n=1,2,3....) erzeugt werden. Daher sollte ein LC-Lowpass-Filter hinter dem AD9857 angeschlossen werden, um die Interferenzsignale zu beseitigen. In Abbildung 24 gibt es ein Lowpass-Filter im I/Q-Modulator AD9857 mit der Frequenz 82MHz, dieses kann direkt benutzt werden.

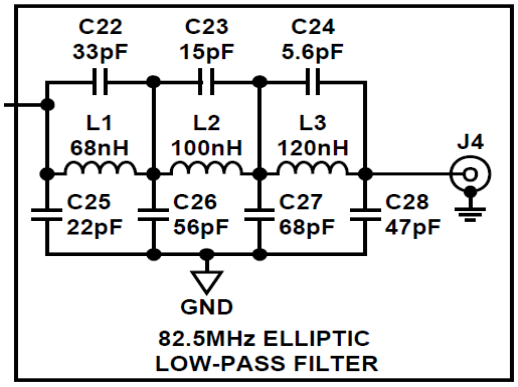


Abbildung 24 Lowpassfilter mit der Frequenz 82MHz

Inverses SINC-Filter: Im DA-Wandler gibt es ein *Halteglied nullter Ordnung*, mit Hilfe dem die Digital-Analog-Umwandlung erreicht werden kann. Aufgrund von dem Halteglied nullter Ordnung verursacht der DA-Wandler immer einen SINC-Schwund im Frequenzbereich. Wegen dieser Eigenschaft können die Ausgangssignale durch die Hüllkurve von SINC gewichtet werden. Das inverse SINC-Filter kann die Eingabedaten vorverarbeiten, um die durch Hüllkurve von SINC verursachten Verzerrungen zu reduzieren.

Multiplizierer für den Referenztakt: Der AD9857 enthält eine PLL. Sie kann das Refe-

„Halteglied nullter Ordnung[56] ist eine elektronische Vorrichtung die es erlaubt, analoge Spannungswerte kurzzeitig auf einem definierten Wert zu halten. Wichtige Kenngrößen sind Einstellzeit, maximale Anstiegsgeschwindigkeit und Haltedrift.“

renztaktsignal (REFCLK) des Eingangs 4-fach bis 20-fach multiplizieren. Damit wird der erforderliche Systemtakt (SYSCLK) erzeugt. Auf diese Weise kann aus einem externen Niederfrequenzoszillator bis zu 200 MHz Systemtakt erhalten werden. Der interne Takt vom AD9857, bzw. der Systemtakt (SYSCLK) wird durch die Frequenzmultiplikation vom Referenztakt bekommen. „SYSCLK“ bietet alle inneren Zeitsteuerungen. Die Abtastrate der IQ-Daten durch Ausgeben des CIC-Filters und die Abtastrate des digitalen Trägers von DDS sind gleich wie „SYSCLK“. Daher ist das modulierte Signal tatsächlich ein Satz von Datenströmen mit der Abtastrate gleich wie „SYSCLK“.

5.2.2 Serielle Kommunikation zwischen FPGA und AD9857

Um die serielle Kommunikation zwischen FPGA und AD9857 zu realisieren, müssen die inneren Register des AD9857 konfiguriert werden.

5.2.2.1 Kommunikationswort

Der AD9857 hat 26 Register. Jedes Register hat 8 Bits. Die Versatzadressen von diesen Registern sind von 00H bis 19H. Von 02H bis 19H sind die Register in vier Gruppen zu je 6 Register mit unterschiedlichen Interpolationsfaktoren und Frequenzsteuer-wörtern eingestellt. Durch die Ansteuerung der Pins „PS0“ und „PS1“ mit hohem oder niedrigem Pegel kann eine entsprechende Gruppe ausgesucht werden.
Bei der seriellen Kommunikation wird jeder Kommunikationszyklus in 2 Teile aufgeteilt. Der erste Teil ist ein 8 Bits Befehlszyklus, anschließend folgt ein Datenzyklus, der Single-Byte oder Multi-Byte-Datenzyklus sein kann. Befehlsbytes spezifizieren die Adressen der inneren Register und die Operationen „Auslesen“ oder „Schreiben“. Nach dem Empfang von Befehlsbytes decodiert der AD9857 den Befehl. Datenbytes entsprechen den Kontrollwörtern der inneren Registern[57].

5.2.2.2 Befehlsbyte

Das Befehlsbyte bietet die benötigten Informationen für die durchgeführten Daten-verarbeitungen an. Das Spezifisches Befehlsbyte ist in der folgenden Tabelle 3 dargestellt.

MSB	D6	D5	D4	D3	D2	D1	D0
R/$\overline{\mathbf{W}}$	N1	N0	A4	A3	A2	A1	A0

Tabelle 3 Definition von Befehlsbyte

Das höchste Bit spezifiziert „Auslesen“ oder „Schreiben“ vom Befehlsbyte im AD9857,

logisch *1* bedeutet „Auslesen“ und logisch *0* ist „Schreiben“. N1 und N0 können zeigen, wie viele Bytes in den nachfolgenden übertragenden Datenzyklus haben, *00* repräsentiert ein Single-Byte Datenzyklus, andere Werte von N1 und N0 entsprechen einem Datenzyklus mit mehreren Bytes. A0 bis A4 entsprechen 5 Bits Adressen. Die Adresse gibt an, welche innere Register betrieben werden. Bei der Übertragung des Datenzyklen mit mehreren Bytes hat die Veränderung der Adressen eine Beziehung zum Befehlsbyte. Wenn der MSB-Modus benutzt wird, dann können die entsprechende Mehrzahl von Registern automatisch mit den abnehmenden Adressen betrieben werden. Wenn der LSB-Modus verwendet wird, dann können die entsprechende Mehrzahl von Registern automatisch mit den zunehmenden Adressen betrieben werden[58].

5.2.2.3 Datenbytes

Table 8. Control Register Quick Reference

Register Address	(MSB) Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	(LSB) Bit 0	Def. Value	Profile
00h	SDIO Input Only	LSB First	PLL Lock Control	**REFCLK Multiplier** 01h: *Bypass* PLL 04h– 14h: 4×– 20×					21h	N/A
01h	CIC Clear	Inverse SINC Bypass	Reserved: Must Be 0	Reserved	Full Sleep	Auto Power-Down	Operating mode 00h: Quad. Mod. 01h: Single-Tone 02h: Intrp. DAC		00h	N/A
02h	Frequency Tuning Word #1 <7:0>								00h	0
03h	Frequency Tuning Word #1 <15:8>								00h	0
04h	Frequency Tuning Word #1 <23:16>								00h	0
05h	Frequency Tuning Word #1 <31:24>								00h	0
06h	**CIC Interpolation Rate** 01h: Bypass CIC Filter 02h–3Fh: Interpolation Factor (2–63, Decimal)						Spectral Invert	Inverse CIC Bypass	08h	0
07h	**Output Scale Factor** Bit Weighting: MSB = 2^0, LSB = 2^{-7}								B5h	0

Tabelle 4 Definitionen der Datenbytes [59]

In der Tabelle 4 gibt die ausführliche Konfigurationen für die innere Register des AD9857. In dieser Entwicklung wird die erste Gruppe der Register verwendet. Die Adressen sind von 00H bis 07H. Durch die Konfiguration der Register werden die Betriebsart und die Arbeitsfrequenz usw. des AD9857 bestimmt.

5.2.2.4 Konfiguration der Resgister von AD9857

Die inneren Register des AD9857 müssen zuerst konfiguriert werden, dann werden die Konfigurationsinformationen vom FPGA zum AD9857 transportiert.

In dieser Arbeit wird die Konfiguration der inneren Register des AD9857 wie folgt eingestellt:

In der Adresse 00H der inneren Registern: Der Systemclock von AD9857 ist 200 MHz.

Die Frequenz vom Quarzoszillator im FPGA ist 50 MHz, diese Frequenz ist gleich dem Referenztakt im AD9857, danach kann der interne PLL-Verstärkungskoeffizient 4 (200/50) bekommen werden. In dieser Arbeit wird der MSB-Modus verwendet. Dann wird die 00H Register mit der binären Zahl 10000100 konfiguriert werden.

In dieser Arbeit arbeitet AD9857 mit QAM-Modus, dann kann in der Adresse 01H das Konfigurationswort 00000000 benutzt werden.

In den Adressen von 02H bis 05H steht das Frequenzwort, das nach der Formel

$$\text{FTWORD}=\frac{\mathrm{f_{out}}}{\mathrm{SYSCLK}} * 2^{32} \qquad (5\text{-}2)$$

ausgerechnet werden kann. Aber in dieser Arbeit wird das Frequenzwort draußen durch die Schalter im FPGA-Entwicklungsboard kontrolliert. Das bedeutet, wenn die Frequenz durch die Kontrolle der Schalter verändert wird, dann kann das Frequenzwort in der Software zu einer entsprechenden Änderung geführt werden. In dieser Arbeit sollte ein Ausgangsfrequenzbereich von 5 MHz bis 65 MHz eingestellt werden.

In der Adresse 06H: Die Frequenz vom Quarzoszillator im FPGA-Entwicklungsboard ist 50 MHz. Die Datengeschwindigkeit von IQ-Daten wird auf 6,25 MHz eingestellt. Systemclock von AD9857 ist 200 MHz. Die Abtastungsrate von IQ-Daten ist gleich wie Abtastungsrate der DDS, hier sind die beiden Abtastungsraten gleich wie Systemclock. „PDCLK“ ist das Rückkopplungstaktsignal von AD9857 zu FPGA, es sollte zweimal die Datengeschwindigkeit von IQ-Daten sein, dann wird 12,5 MHz eingestellt. Im AD9857 gibt es zwei Interpolationsfilter, ein fester Interpolationsfilter mit der Interpolations-rate 4, ander ist ein CIC Filter mit einer programmierbaren Interpolationsrate N. Der Bereich von N ist 2 $\sim$ 63. Die Interpolationsrate N hat eine Beziehung auf dem Systemclock „SCLK“:

$$SYSCLK = PDCLK * 4N \qquad (5\text{-}3)$$

Durch diese Gleichung kann die Interpolationsrate N gleich 8 berechnet wird, dann können die binäre Zahl 00100000 in der Adresse 06H geschrieben werden.

In der Adresse 07H des inneren Registers: der interne Verlust der QAM-Modulation ist 3dB ($1/\sqrt{2}$), deshalb wenn das System in dem QAM-Modus arbeitet, sollte der Gewinn von dem ausgegangenen Verstärker $\sqrt{2}$ konfiguriert werden, um die Datenmenge zu erholen. Da das standardmäßige Verfahren des AD9857 ist QAM-Modus, wird der Standardwert B5h verwendet. Mit einer Binärzahl wird als 10110101 ausgedrückt.

Zufolge der Konstruktionsanforderungen werden die Konfigurationen der inneren

Register des AD9857 wie folgende Tabelle angezeigt.

Befehlsbytes	Adresse	Konfigurationen
0000_0000	00h	10000100
0000_0001	01h	00000000
0000_0010	05h	freq2[31:24]
	04h	freq2[23:16]
	03h	freq2[15:8]
	02h	freq2[7:0]
0000_0110	06h	00100000
0000_0111	07h	10110101

Tabelle 5 Befehlsbyte und entsprechende Konfigurationen von inneren Registern

5.2.3 Schaltungsdesign mit AD9857

In dieser Arbeit wurden der FPGA und auf DDS-Technologie basierende orthogonale Frequenzwandler AD9857 verwendet und eine Hardwareplattform mit der vollständigen programmierbaren Steuerung aufgebaut. DDS (Direct Digital Synthesis) kann zwei digitale Träger mit $90°$ Phasenverschiebung erzeugen, die zur Quadratur-modulation erforderlich sind. Das gesamte Systemschaltbild ist in Abbildung 25 gezeigt.

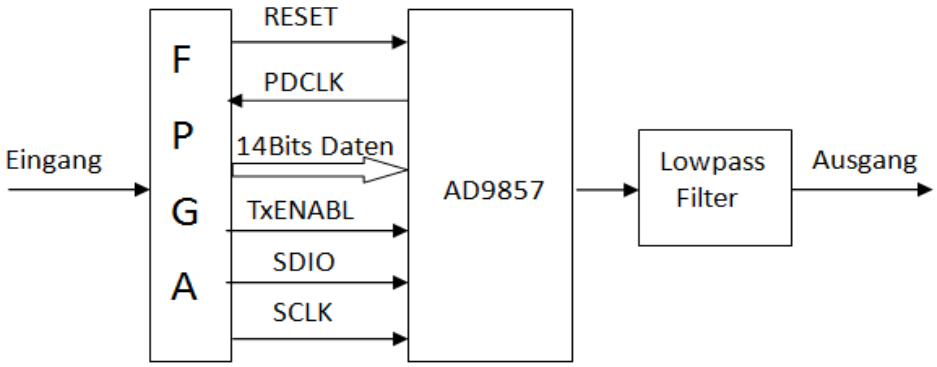


Abbildung 25 Anschluss zwischen AD9857 und FPGA

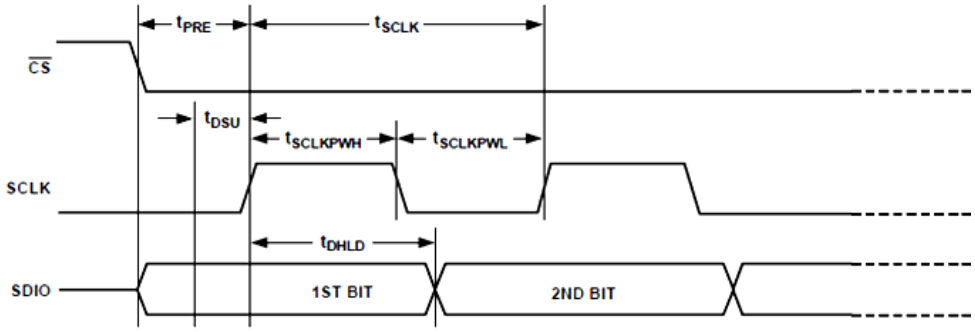
Von dieser Abbildung ist es klar, dass die digitalen Basisbandsignalverarbeitungen des Aufwärtsumsetzers und die Steuerung des Systems durch FPGA kontrolliert werden. Mit AD9857 werden die Interpolationsfilterung und Aufwärtskonvertierung in dem digitalen Bereich vervollständigt. NCO(numerical controlled oscillator) kann vom FPGA durch die Verwendung von AD9857 befreit, und durch DDS in diesem Chip AD9857 erreicht werden. DDS wird verwendet, um die Basisbandsignalverarbeitung komplett im FPGA zu erzielen. Durch die Anwendung von DDS kann nicht nur die

High-Speed-Digital-Signal-Modulation erleichtern, sondern auch die Eigenschaften von High-Speed und High-Precision von DDS komplett nutzen. Daher kann die Modulation des Basisbandsignals in einigen Megahertz Träger sehr leicht erreichen.

Die Kommunikationsschnittstelle vom AD9857 ist eine SPI-Schnittstelle (Serial Peripheral Interface Bus). Über SPI können MCU (Micro Control Unit) mit unterschiedlichen Peripheriegeräten durch ein serielles Verfahren kommunizieren, um die Informationen auszutauschen.

SPI-Schnittstelle wird in der Regel mit fünf Leitungen verwendet: SCLK, $\overline{CS}$, SYNCIO, SDIO, SDO. Durch diese Schnittstelle können die Konfigurationsinformationen der inneren Register vom FPGA zum AD9857 übertragen werden.

5.2.3.1 Ablauf der SPI-Schnittstelle



SYMBOL	DEFINITION	MIN
t_{PRE}	$\overline{CS}$ SETUP TIME	40ns
t_{SCLK}	PERIOD OF SERIAL DATA CLOCK	100ns
t_{DSU}	SERIAL DATA SETUP TIME	30ns
$t_{SCLKPWH}$	SERIAL DATA CLOCK PULSE WIDTH HIGH	40ns
$t_{SCLKPWL}$	SERIAL DATA CLOCK PULSE WIDTH LOW	40ns
t_{DHLD}	SERIAL DATA HOLD TIME	0ns

Abbildung 26 Zeitlauf von serieller Kommunikation [60]

Das Freigabesignal „$\overline{CS}$“ ist bei einem Low-Pegel aktiv. Es kann die seriellen Eingänge und Ausgänge des AD9857 aktivieren. Nach der Aktivierung wird die Kommunikation zwischen Master- und Slave-Device durch das Signal „SCLK“ entschieden. Bei der ansteigenden Flanke vom Signal „SCLK“ werden die Daten durch die serielle Kommunikationsschnittstelle übernommen.

Wie in Abbildung 26 dargestellt wird, werden die Daten bei der ansteigenden Flanke des Signals „SCLK“ in der seriellen Kommunikationsschnittstelle geschrieben. t_{SCLK} ist der Taktzyklus der seriellen Kommunikationsschnittstelle und beträgt mindestens 100 ns. Das bedeutet, dass die maximale Kommunikationsfrequenz der SPI-

Schnittstelle 10 MHz ist.

5.2.4 Parallele Kommunikation zwischen FPGA und AD9857

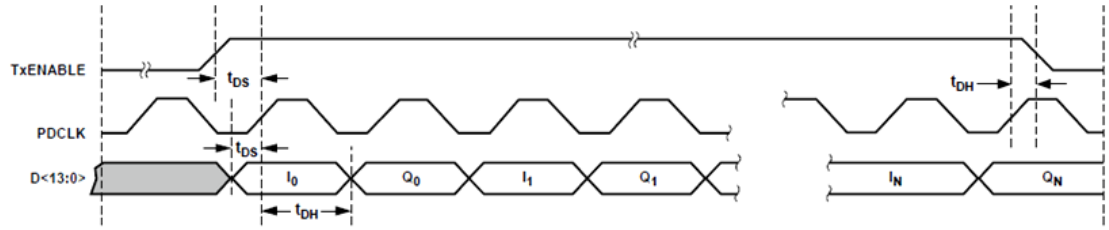


Abbildung 27 Ablauf der Übertragung der I- und Q-Daten[61]

14 Bits Daten werden parallel in den AD9857 eingegeben. Der Zeitablauf ist in Abbildung 27 dargestellt. t_{DS} ist die Einrichtzeit der Daten, die vor dem Abtasten bei der ansteigenden Flanke dauert, und t_{DS} währt mindestens 4 ns. t_{DH} ist die aufrechterhaltene Zeit von Daten nach dem Abtasten und dauert mindestens 0. Wenn „TxEnable“ mit hohem Pegel aktiv ist, dann werden die I/Q-Daten bei der ansteigenden Flanke von „PDCLK“ abwechselnd zu AD9857 eingegeben. Der AD9857 beginnt die Speicherung von Daten bei der ersten ansteigenden Flanke von „PDCLK“ nach der anstiegenden Flanke von „TxEnable“. Standardmäßig wird zuerst einmal *I*-Data gespeichert, anschließend wird einmal *Q*-Data bei der nächsten anstiegenden Flanke von „PDCLK“ gespeichert. Weil AD9857 kein Signal für Auslesen und Schreiben hat, muss das System synchron sein. Wenn die Daten von FPGA zum AD9857 geschrieben werden, müssen zudem die Daten gespeichert werden, um die gültigen Daten zu halten, bis die ansteigende Flanke von „PDCLK“ beendet ist. Wenn „TxEnable“ unwirksam ist, gehen die Daten von *I* und *Q* auf 0.

Die auf der QAM-Modulation wartenden Daten von *I* und *Q* werden zeitversetzt zum Demultiplexer im AD9857 übertragen und die Daten werden in zwei parallelen Zweigen von *I*- und *Q*-Daten demultiplexiert. Die parallelen Anschlussleitungen vom AD9857 bestehen aus den 14 parallelen Leitungen für Daten, sowie „TxEnable“ und „PDCLK“. Wenn der AD9857 im QAM-Modus arbeitet, wird „TxEnable“ zur Synchronisierung eines externen Gerätes mit dem AD9857 benutzt. „TxEnable“ ist mit logisch-1 aktiv. „PDCLK“ kann einen parallelen synchronen Takt ausgeben. Die Frequenz wird durch den Interpolationsfaktor und den Systemclock festgelegt.

$$SYSCLK = PDCLK * 4N \qquad (5\text{-}4)$$

N: Interpolationsfaktor von CIC Filter

Wenn der AD9857 im QAM-Modus arbeitet, sind die I/Q-Daten mit der ansteigenden Flanke von „PDCLK“ zeitversetzt aktiv. Da die I/Q-Daten abwechselnd zum AD9857

übertragen werden, sollte die Taktfrequenz von „PDCLK“ zweimal die Datengeschwindigkeit der einzelnen *I*- oder *Q*-Daten sein.
Basisbanddaten werden nach der Erzeugung von PRBS und TS-Datenströmen, der Kanalcodierung, den digitalen Datenverarbeitungen und dem Basisbandfilter im RAM gespeichert. Das RAM ist in dem FPGA eingebettet. Der Ausgabe-Pin „PDCLK/ FUD“ vom AD9857 wird als Lesetaktsignal genutzt, um die Daten im RAM zu entnehmen.

5.2.5 Auswahl des Betriebsmodus von AD9857

Der AD9857 hat drei verschiedene Arbeitsarten: Quadraturmodulation, Single-Frequenz-Modus und Interpolationsmodus. Hier wird der Modus in der Quadraturmodulation verwendet, um die 64QAM zu realisieren.
Im Modus der Quadraturmodulation empfängt der AD9857 die 14 Bits parallelen *IQ*-Daten. Die *IQ*-Daten werden abwechselnd zum AD9857 eingegeben, dann in zwei Zweige aufgeteilt, jeder Zweig übertragt *I*-Daten oder *Q*-Daten. Anschließend werden die *I*- oder *Q*-Daten je über ein inverses CIC-Filter, ein festes Interpolator und ein programmierbares Interpolator usw. zu dem Quadraturmodulator gesendet.

5.2.6 System-Clock Design

Zum Erreichen eines sehr geringen Rauschens wird zuerst auf der Auswahl von einem Quarzoszillator geachtet. Denn gemäß der Datenanalyse ist das Phasenrauschen des Ausgangssignals von dem Phasenrauschen des Taktsignals abhängig. Die Abhängigkeit wird durch $20\log(f_{out}/f_{clk})$ beschrieben. Das bedeutet, wenn die Ausgangsfrequenz unveränderlich ist, dann ist das Phasenrauschen, das durch ein Taktsignal mit 10 MHz erzeugt wird, 20 dB schlechter als das durch ein Taktsignal mit 100 MHz erzeugte Phasenrauschen. Deshalb muss ein Quarzoszillator mit einer hohen Frequenz und einem extremen niedrigen Phasenrauschen als Taktsignal verwendet werden. Der externe Referenztakt sollte weniger als 1ps Jitter-Leistung erreichen.

5.2.6.1 Clock-System von FPGA-Entwicklungsboard

Das DE2-115-Board enthält einen aktiven Quarzoszillator, der ein Taktsignal mit 50MHz Frequenz erzeugt, sowie einen Taktpuffer. Durch die Verwendung des Taktpuffers steht ein Taktsignal mit geringem Jitter zur Verfügung. Dieses Taktsignal wird verwendet, um die Logikschaltungen innerhalb des FPGAs anzutreiben. Das

Board enthält noch zwei SMA-Verbindungsanschlüsse. Diese werden benutzt, um ein externes Taktsignal zu empfangen, oder um ein Taktsignal vom FPGA-Entwiklungs-board nach außen auszugeben. Darüber hinaus werden alle Eingänge der Taktsignale mit PLL-Modul im internen FPGA-Entwiklungsboard verbunden, womit der Benutzer diese Taktsignale als Eingangstakt für die PLL-Schaltung nehmen kann[62].
Abbildung 28 zeigt die Verteilungsinformation der Taktsignale im DE2-115 Board.

Abbildung 28 Verteilungsschaltung der Taktsignale [63]

5.2.6.2 Clock-System von AD9857

Das AD9857-Evaluationsboard hat einen programmierbaren PLL-Multiplizierer. Das Taktsignal des Chips wird vom inneren Systemtakt „SYSCLK“ abgeleitet, der durch einen Phasenregelkreis PLL auf dem Chip erzeugt wird. Der Systemtakt „SYSCLK“ wird durch die verdoppelte Frequenz von dem lokalen Oszillator bekommen. Die Anzahl der Frequenzvervielfachung ist für den Benutzer wählbar. Der Bereich für den Faktor ist 4 bis 20. Aber es gibt eine Grenzbedienung, dass der Systemtakt nicht größer als 200 MHz sein darf. Mithilfe dieser Funktion kann ein innerer Systemtakt von 200 MHz durch ein Taktsignal, das durch einen Benutzer eingegeben wird, aus einer nahe zu beliebigen Frequenz gewonnen werden. „SYSCLK“ wird an das Steuermodul gesendet, danach wird „SYSCLK“ durch das Steuermodul zu den anderen Modulen im Chip bereitgestellt. Beispielsweise bietet der Chip den Systemtakt für „DDS“ an. Dann fungiert „DDS“ mit diesem Systemtakt als Referenztakt.
Der innere Systemtakt (SYSCLK) vom AD9857 kann alternativ auch direkt durch den Pin „REFCLK“ in das Board eingespeist werden (siehe Abbildung 29). Der Referenzclock hat nicht nur einen unsymmetrischen Eingang sondern auch einen Differenzeingang. Durch die Belegung des Pins „REFCLK ENABLE“ mit high oder low kann der unsymmetrische Eingang oder der Differenzeingang ausgewählt werden.
In dieser Entwicklung wird der unsymmetrische Eingang verwendet und der Pin „DIFFCLKEN“ mit logisch *0* konfiguriert.

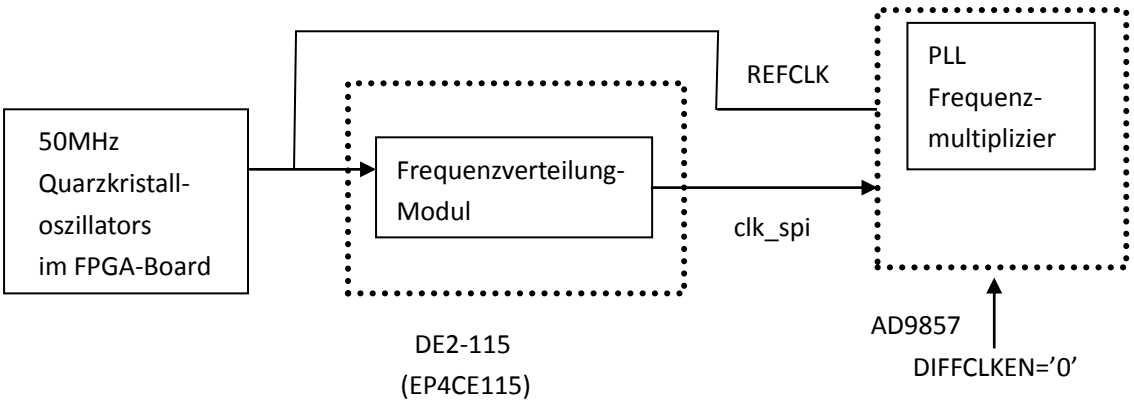


Abbildung 29 Schaltung für Taktsignal

6 Entwicklung einer QAM-Modulator-Software

Die Software für die FPGA-Schaltung und die Kontrolle der AD9857-Schaltung kann in Quartus II v10.0 mit der Hardwarebeschreibungssprache Verilog HDL realisiert werden.

Quartus II ist ein synthetisches FPGA-Entwicklungssoftware. Quartus II hat eine Vielzahl von Eingabentypen, beispielsweise Schematic, VHDL, Verilog HDL, AHDL (Altera Hardware Description Language) usw. In Quartus II hat einen eigenen Synthesizer und Simulator eingebettet, sodass sie den vollständigen FPGA-Design-ablauf von der Konzeption zu der Konfiguration des Hardwares erledigen kann.

Verilog HDL ist eine Hardwarebeschreibungssprache. Diese Sprache beschreibt, wie die logischen Schaltungen aufgebaut und miteinander verbunden sind. Verilog HDL erlaubt, dass Hardware (z.B. ICs) auf einer höheren Abstraktionsebene beschrieben wird. Das Verhalten, die Struktur und niedrigere Abstraktionslevel auf Gatterebene können beschrieben werden[64].

6.1 Bauteile und Ablaufdiagramm

Abbildung 30 zeigt die wichtigen Blöcke in dieser Arbeit.

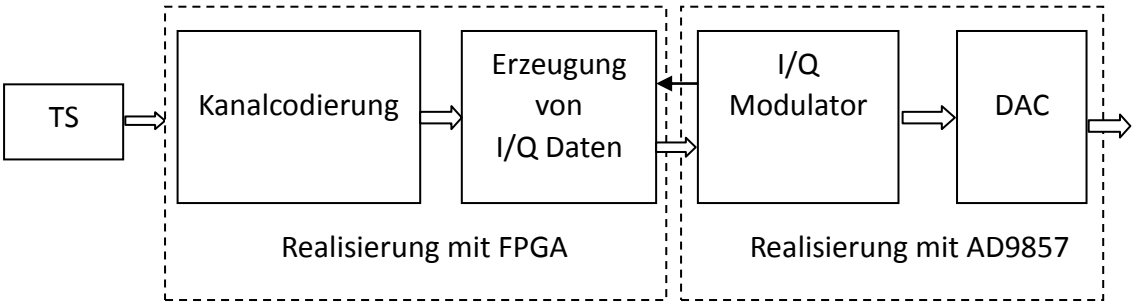


Abbildung 30 Prinzipschaltung des QAM-Modulators

Der erste Schritt ist die Erzeugung von TS-Datenströmen, die Datenquellen in diesem gesamten System sind. Die Kanalcodierung enthält die Verwürfelung, die Reed-Solomon-Codierung und die Faltungsverschachtelung. Bei dem Bauteil der Erzeugung von I/Q-Daten werden zuerst die Datenströme nach der Kanalcodierung mit Hilfe der Umwandlung von Byte zu m-Symbolen 6 Bits bekommen, die 3 Bits *I*-Data und 3 Bits *Q*-Data enthalten. Anschließend werden die beiden höchstwertigen Bits der 6 Bits differenzcodiert, danach das Mapping durchgeführt, weiterhin einem Basisbandfilter zugeführt. Die Ausgangsdaten vom *I*- oder *Q*-Zweig sind die *I*- oder *Q*-Daten. Weiterhin werden die *I/Q*-Daten zum AD9857-Entwicklungsboard übertragen. Schließlich wird das 64QAM-Signal erzeugt. Abbildung 31 zeigt das gesamte Ablaufdiagramm.

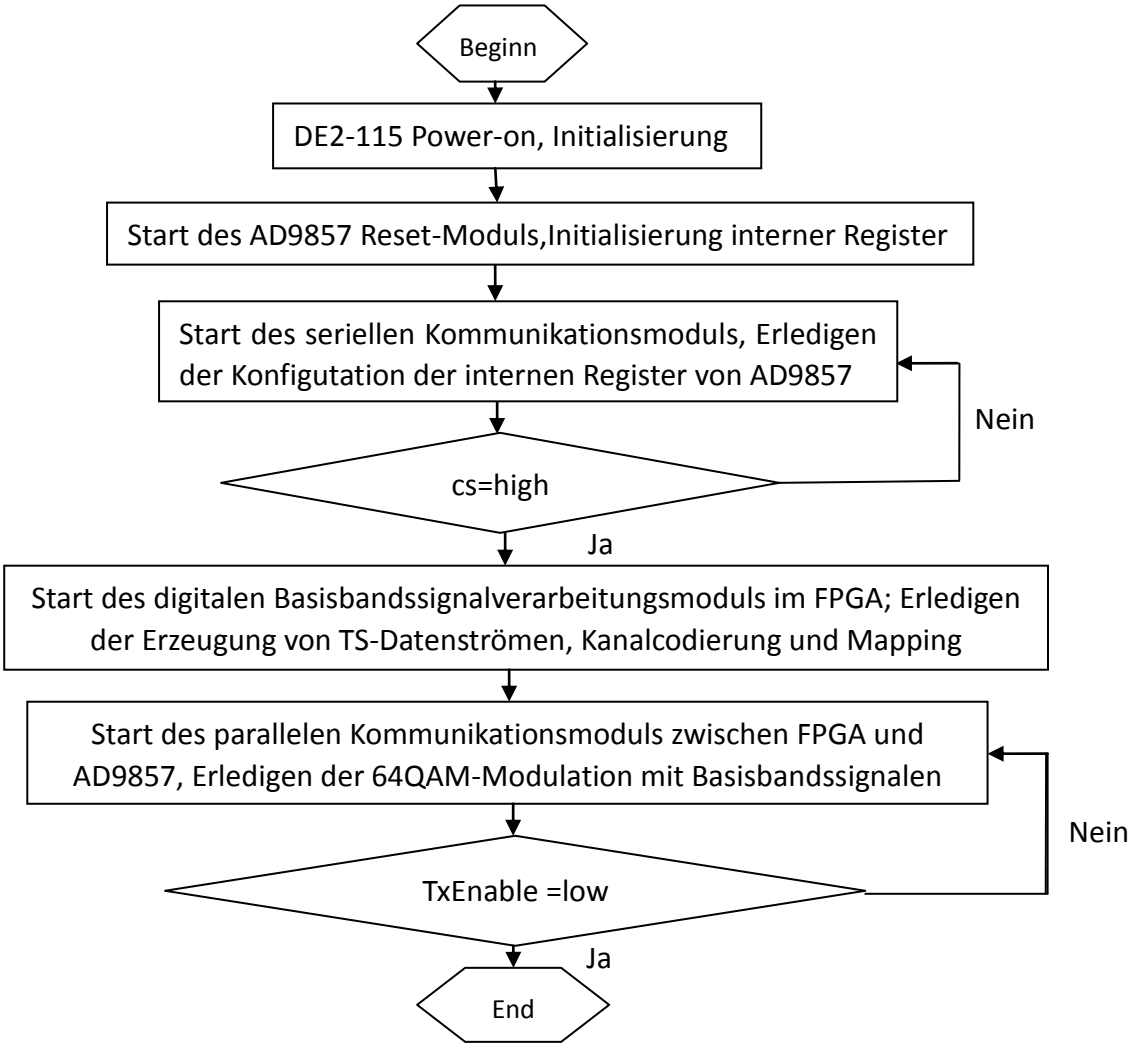


Abbildung 31 gesamtes Ablaufdiagramm

Nach dem Einschalten der Betriebsspannung wird zuerst der FPGA initialisiert, danach können durch einige Schalter im FPGA-Entwicklungsboard die Zurücksetzung,

der Beginn von der seriellen Übertragung der Kontrollwörter und der Beginn von der parallelen Übertragung der *IQ*-Daten gesteuert werden. Das Zurücksetzungssignal dauert mindestens fünf Taktzyklen, um AD9857 zurückzusetzen und auch um die internen Register vom AD9857 zu initialisieren. Wenn die Initialisierung vom FPGA abgeschlossen ist, kann die serielle Kommunikation durch die Auslösung eines Schalters gestartet werden. Dann können die Kontrollwörter der internen Register vom AD9857 über die serielle Schnittstelle durch „SDIO“ eingestellt werden. Nachdem die Kontrollwörter der internen Register vom AD9857 konfiguriert worden sind, werden das digitale Basisbandsignalverarbeitungsmodul im FPGA gestartet und die auf Übertragung wartenden digitalen Basisbandsignale verarbeitet. Gleichzeitig sendet der AD9857 den synchronisierten Takt „PDCLK“ zum FPGA als Dateneingangstakt. Wenn die digitale Basisbandsignalverarbeitung in FPGA erledigt ist, wird „TxEnbale“ des AD9857 auf high gesetzt, um die Daten über „PDCLK“ an den AD9857 zu senden. Die verarbeiteten digitalen Basisbandsignale schließen die Interpolationsfilterung, Quadraturmodulation, und die DA-Wandlung im AD9857 ab. Der differentielle Ausgangsstrom kann über einen Transformator und ein analoges Filter das gewünschte ZF-Signal erzeugt werden.

6.2 Design des digitalen Basisbandsignalverarbeitungsmodus

In dieser Arbeit wird der Kodierungsprozess vom digitalen Signal im FPGA erledigt. Der Kodierungsprozess enthält die Erzeugung von TS-Datenströmen, die Kanalcodierung, das Mapping etc.

6.2.1 Generator von PRBS und Transportströmen

Der PRBS-Generator besteht aus 31 Schieberegistern und einem XOR-Gatter, die Schaltung wird in Abbildung 32 dargestellt：

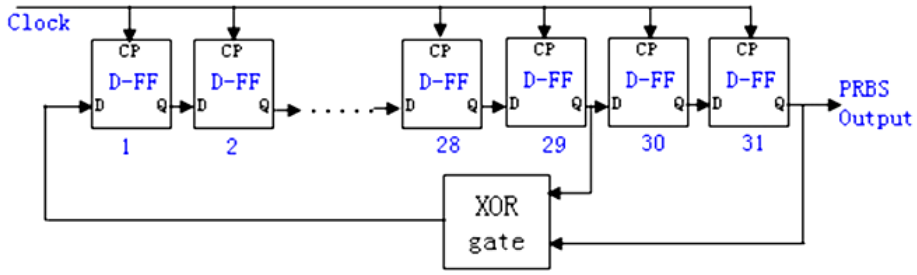


Abbildung 32 PRBS-Generator

In Dieser Arbeit wird die erzeugende Funktion von PRBS $1 + X^{29} + X^{31}$ genommen, weil je größer die Ordnung der Funktion ist, desto größer ist die Reichweite und

zufälliger sind die Daten, deshalb wird die Ordnung von 31 benutzt. Die Rückkopplungen sind hinter der 29ten und 31ten Stufe. Durch den folgenden Code wird die Pseudozufallsfolge im Register „in“ erzeugt.

PRBS_31[31:2]<=PRBS_31[30:1];

PRBS_31[1]<=PRBS_31[29]^PRBS_31[31];

in_1<= PRBS_31[31];

Durch ein Schieberegister, das in der Software definiert wird, kann die Pseudozufallsfolge Byte für Byte ausgegeben werden. Die ausgegebenen Bytes werden im Register „PRBS_reg“ gespeichert.

In der Software werden 2 Zählwerke „count_8“ für die 8 Pakete jedes Rahmens und „count_1503“ für die 188 Bytes Daten jedes Pakets definiert. „count_8“ hat 8 Zustände. Bei jedem Zustand zählt das Zählwerk „count_1503“ 1504. Bei jeden ersten 8 Zahlen wird das Synchronisationsbyte dem Register „PRBS_1503“ zugeführt, in dem die TS-Datenströme ansteht. Bei den anderen 1496 (187 Bytes) Zahlen werden die im Register „PRBS_reg“ gespeicherten Daten dem Register „PRBS_ 1503“ zugeführt. So erzeugt man die TS-Datenströme.

6.2.2 Invertierung und Verwürfelung

6.2.2.1 Realisierung der Invertierung

Um die Invertierung zu realisieren, ist auch das Zählwerk „count_8“(siehe Abschnitt 6.2.1) vom Modul „Erzeugung der Transportströme“ benötigt. Wenn „count_8“ gleich *0* ist , wird das erste Paket gezeigt, danach das Synchronisationsbyte des Pakets invertiert. Wenn „count_ 8“ gleich *1* bis *7* sind, ist die Invertierung der Synchronisationsbytes nicht benötigt.

6.2.2.2 Realisierung der Verwürfelung

Wie in Abschnitt 4.2.1.1 beschrieben wurde, wird zur Realisierung der Invertierung ein 14 Bit langes Schieberegister benötigt.

Bei der Verwürfelung bleiben alle Synchronisationsbytes unverändert und die anderen Daten von TS-Datenströmen werden mit Pseudozufallszahlen Bit für Bit verwürfelt. Abbildung 33 zeigt der Schieberegister mit Rückkopplungen hinter der vierzehnten und fünfzehnten Stufe.

Vor der Verwürfelung sollte des Register der Pseudozufallsfolge mit 15b’10010_ 10100_00000 initialisiert werden. Das höchste Bit des Anfangswerts vom Register entspricht dem niedrigsten Bit des Polynoms, die zufälligen Werte des Registers

verschieben immer mit dem Taktimplus von links nach rechts. Das erste Ausgangsbit der Verwürfelungscodes ist das höchste Bit des ersten Bytes hinter einem invertierten Synchronisationsbyte B8h(siehe Abbildung 33).

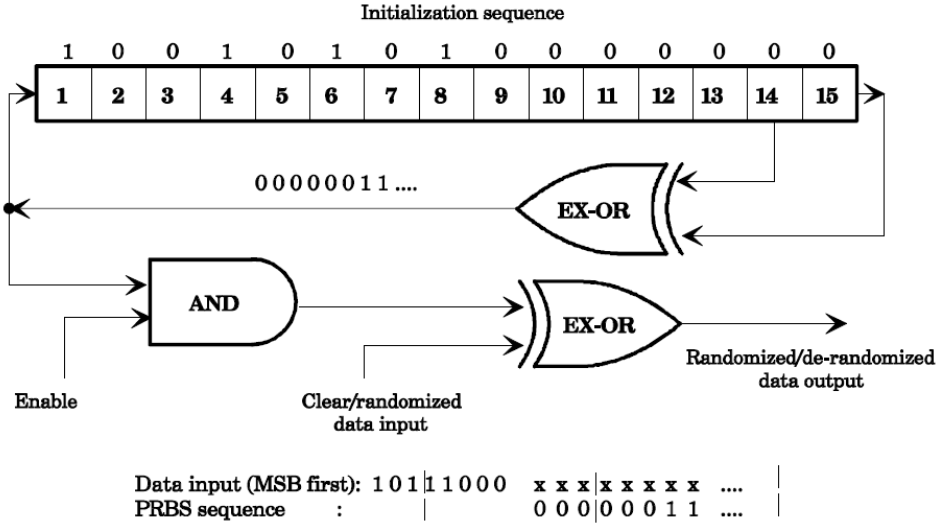


Abbildung 33 Realisierung der Verwürfelung[65]

Um die Synchronisation zu verbessern, dürfen die Synchronisationsbytes von nachgehenden 7 Paketen mit dem Taktimplus nicht verwürfelt werden, das heißt, dass die Verwürfelung kontinuierlich arbeiten sollte, aber die Ausgabe ist deaktiviert, wobei jedes Synchronisationsbyte (B8h und 47h) nicht verwürfelt ist. Daher ist der Zyklus der erzeugten Sequenz vom Pseudozufallsfolgengenerator 1503 Bytes.

In der Software wird mit Hilfe einem Freigabesignal „Enable“ die Funktion realisiert. Wenn die Synchronisationsbytes gezeigt werden, wird das Signal „Enable“ auf „0000_0000“ gesetzt, im anderen Fall ist „Enable“ gleich „1111_1111“. Danach wird die Konjunktion von „Enable“ und der Pseudozufallsfolge *„feedback_8“* gerechnet und das Konjunktionsergebnis im Resgister *„rdm_seq*“ gespeichert. Weiterhin werden die Daten im *„rdm_seq*“ mit den durch die Invertierung gewonnenen Datenströmen Modulo-2-addiert und die Ausgabedaten im Resgister *„rdm_data_ out*“ gespeichert. Folgende Funktionen werden verwendet:

assign rdm_seq = feedback_8 & enable;

assign rdm_data_out = rdm_seq ^ data_in_inv;

6.2.3 Synchronisation zwischen Ausgabedaten der Verwürfelung und Eingabedaten der RS-Codierung

Jedes Paket der Eingabedaten von Reed-Solomom-Codierung hat 188 Bytes, und jedes Paket der Ausgabedaten der Reed-Solomom-Codierung hat 204 Bytes. Aber die

Eingabedaten der Reed-Solomom-Codierung sind kontinuierlich, es gibt keine Plätze im Zeitbereich für 16 Bytes Fehlerkorrekturcodes mehr, deshalb wird ein zusätzliches Modul nach der Verwürfelung geschrieben. Im Modul wird ein FIFO- Schieberegister verwendet, um die Daten zu puffern, damit die Ausgabedaten von der Verwürfelung mit der Eingabedaten von der RS-Codierung synchronisieren.

In der Software wird ein FIFO-Schieberegister mit 141 Bytes definiert und die Datengeschwindigkeit vom Auslesen ist 4mal die vom Schreiben. Wenn die 141 Bytes Daten im FIFO fertig geschrieben werden, wird ein Freigabesignal für das Auslesen erzeugt. Dann können Schreiben und Auslesen von 188 Bytes Daten gleichzeitig fertig sein. Von der Formel (6-1) kann man den Vorgang nachvollziehen:

$$(141+47\times1)\ \text{Bytes}= (47\times4)\ \text{Bytes} = 188\ \text{Bytes} \qquad (6\text{-}1)$$

$$(188\times Trd+141\times Twr)/Trd=188\times1+141\times4=752\ \text{Bytes} \qquad (6\text{-}2)$$

Durch die Gleichung (6-2) können 752 Bytes erzeugt werden, davon sind 188 Bytes für Nutzdaten, 564 Bytes für Lücken. Nach der RS-Codierung gibt es 16 Bytes Nutzdaten mehr für Fehlerkorrekturcodes und die Nutzdaten sind 188+16=204 Bytes, die Lücken haben noch 752-16-18= 548 Bytes. Bei den Lücken wird das Signal „valid_out“ auf low gesetzt, damit die Lücken nicht ausgegeben werden.

6.2.4 Reed-Solomon-Codierung

Eine mögliche Realisierungsmethode der RS-Codierung zeigt Abbildung 34. Der Codierer besteht hauptsächlich aus einem r-stufigen rückgekoppelten Schiebe-register (r=n-k) und einer Steuerschaltung. Das benutzte Rückkoppelmuster wird durch die Faktoren g_i angegeben, die *1* oder *0* sind. *1* bedeutet, dass ein Rückkoppelungs- zweig an der betreffenden Stelle vorhanden ist. Bei *0* ist er nicht vorhanden. Die g_i sind die Koeffizienten vom Generatorpolynom $g(x)$. Die modulo-2-Addierer sind auch als Exklusiv-Oder-Elemente genannt.

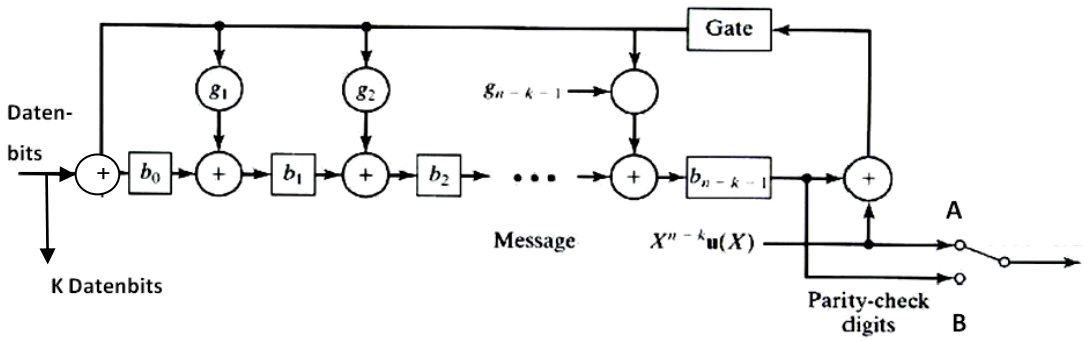


Abbildung 34 Erzeugung der Paritätsprüfbits der RS-Coders durch Polynommultiplizier mit einem Schieberegister[66]

Mit jedem Taktschritt wird in der Stellung A des Schalters am Ausgang der Schaltung jeweils ein Bit der k Datenbits in die Schaltung eingelesen. Zur gleichen Zeit werden diese Bits auf die Ausgangsleitung gegeben, weil sie die ersten k Bits eines n-Bit-Codewortes formen. Die Datenbits sollten das Schieberegister durchlaufen, bis das k-te Bit die letzte Stufe von diesem Schieberegister verlässt. Dann werden die Paritätsprüfbits durch den Inhalt von r(r=16) Schieberegisterzellen dargestellt. Diese werden in der Stellung B des Schalters am Ausgang mit den folgenden Taktschritten ausgeschoben und der Ausgangsanschluß zugeführt[67].

In der Software werden 4 Zustände *idle, rs_encoder, shift_out und initialization* definiert. Der Ablauf von den 4 Zuständen wird in Abbildung 35 gezeigt, *counter_188* stellt die Zahl jedes Pakets dar. *counter_16* entspricht die Zahl der Fehlerkorrekturcodes. Der Grundzustand ist *idle*.

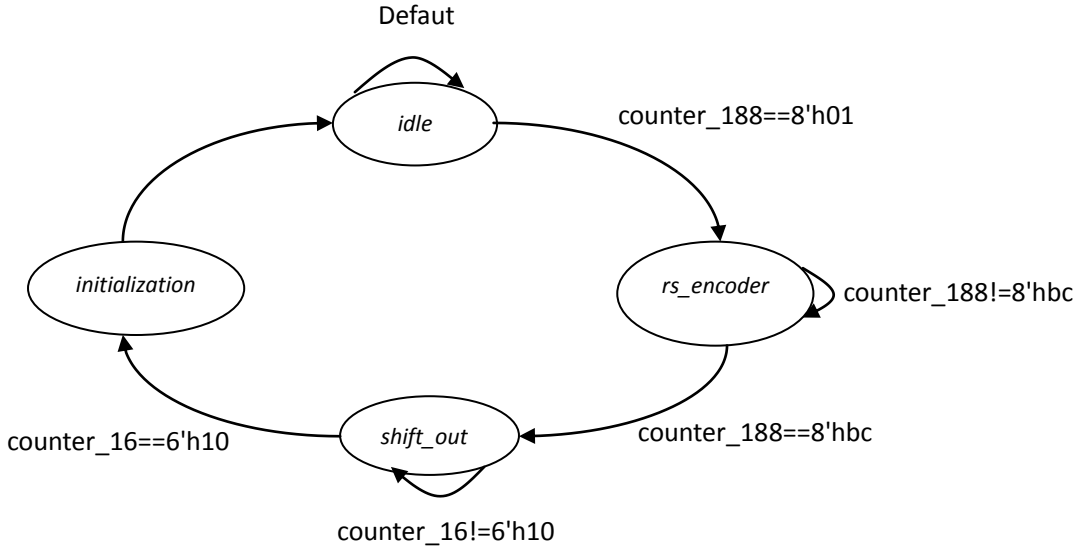


Abbildung 35 Ablauf der RS-Codierung

Wenn ein Paket kommt, dann beginnt der zweite Zustand *rs_encoder*. In diesem Zustand gibt der RS-Codierer die 188 Bytes Informationsdaten aus. Wenn die Ausgabe von 188 Bytes Informationsdaten nicht fertig ist, dann wiederholt sich der Ablauf. Wenn 188 Bytes Informationsdaten schon ausgegeben worden sind, beginnt der dritte Zustand *shift_out*. In diesem Zustand werden die 16 Bytes Fehlerkorrekturcodes mit dem Taktimplus nach den Informationsdaten ausgegeben. Die Fehlerkorrekturcodes sind vorher schon durch einen entsprechenden Code erzeugt und in 16 Schieberegister von *shift00* bis *shift15* gespeichert worden. Bei dem Zustand *shift_out* sollte *counter_16* überprüft werden. Wenn *counter_16* gleich 16 ist, d.h. die Ausgabe von 16 Bytes Fehlerkorrekturcodes fertig ist, dann startet der

nächste Zustand. Wenn nicht gleich 16 ist, werden die nächsten Bytes ausgegeben. Wenn der Zustand *initialization* ist, werden das Freigabesignal *Valid_188* und Synchronisations- signal *Sync_188* initialisiert, um den nächsten Ablauf zu vorbereiten.

6.2.5 Faltungsverschachtelung

Verschachteler und Entschachteler können durch Dual-Port-RAM realisiert werden. Durch Dual-Port-RAM werden die Zeitverzögerungen von Eingabedaten realisiert. In dieser Entwicklung werden die Lese- und Schreib-Adressen nach einem bestimmten Gesetz von den RAM-Speicherzellen gelesen und geschrieben. Dual-Port-RAM hat zwei Eingänge und zwei Ausgänge, die jeweils unabhängig voneinander arbeiten können. Das Realisierungsdiagramm ist in Abbildung 36 dargestellt.

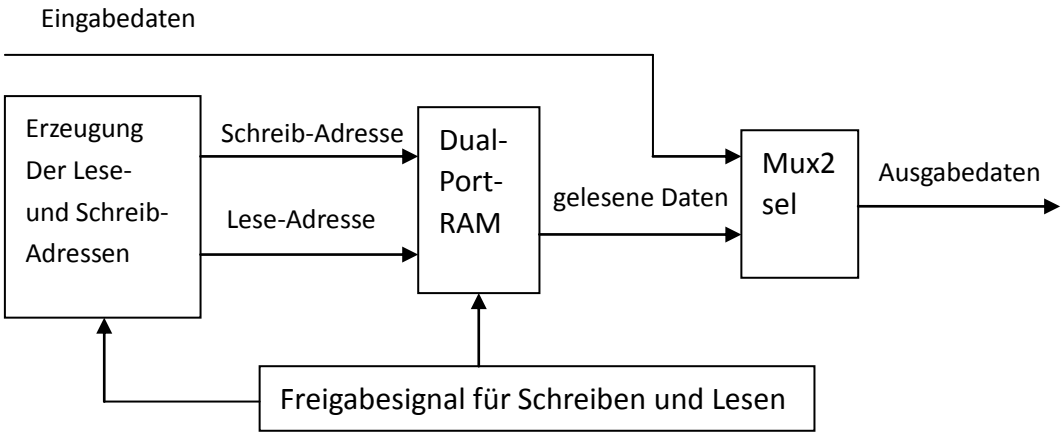


Abbildung 36 Das logische Blockschaltbild der Faltungsverschachtelung

Schlüsselpunkt der Konstruktion ist die Adressvergaben von RAM, um die Lese- und Schreib-Adressen zu erzeugen. Das Realisierungsdiagramm ist in Abbildung 33 dargestellt. Die Lese- und Schreib-Adressen von dem *i*-ten Zweig sind von der Adresse nach rechts (*i*-1)×17 Einheiten weit. Um diese Schaltung zu konzipieren, gibt es für jeden Zweig außer dem ersten Zweig einen Zähler, wobei jeder Zähler unabhängig ist und der Zahlenwert von dem *i*-ten Zweig a_i ist. Der erste Zweig hat keinen Verzögerer, damit das Synchronisationsbyte zügig ausgegeben werden kann. Aber um den Arbeitstakt anzupassen, muss ein Byte Speichereinheit dem ersten Zweig zugeordnet werden. Der zweite Zweig braucht 17×1 Symbol-Perioden Zeitverzögerungen, deshalb müssen 18 Bytes Speichereinheiten zugeordnet werden. Der Zahlenwert a_2 ist von 0 bis 17. Der dritte Zweig braucht 17×2 Symbol-Perioden Zeitverzögerungen, deshalb müssen 35 Bytes Speichereinheiten zugeordnet werden, der Zahlenwert a_3 ist von 0 bis 34;... ; Der zwölfte Zweig braucht 17×11

Symbol-Perioden Zeitverzögerungen, deshalb müssen 188 Bytes Speichereinheiten zugeordnet werden, der Zahlenwert a_{12} ist von 0 bis 187. Es braucht insgesamt 1+ 18+… + 188= 1134 Bytes Speichereinheiten, deswegen wird ein 2K RAM verwendet, d.h. der Adressbus umfasst 11 Bits.

Angenommen, die erste Adresse von dem *i*-ten Zweig ist b_i und sie wird durch b_i-creat erzeugt. Die letzte Adresse ist c_i, sie wird durch c_i-creat erzeugt. Die Regel zur Bildung von Lese- und Schreib-Adressen für das RAM ist:

Lese-Adresse vom i-ten Zweig: $a_i + b_i$;

Schreib-Adresse vom i-ten Zweig:
$$\begin{cases} a_i + b_i - 1, & a_i \neq 0; \\ c_i, & a_i = 0; \end{cases} \qquad (6\text{-}3)$$

Das Realisierungsdiagramm der Vergabe der Lese- und Schreib-Adressen wird in Abbildung 37 gezeigt.

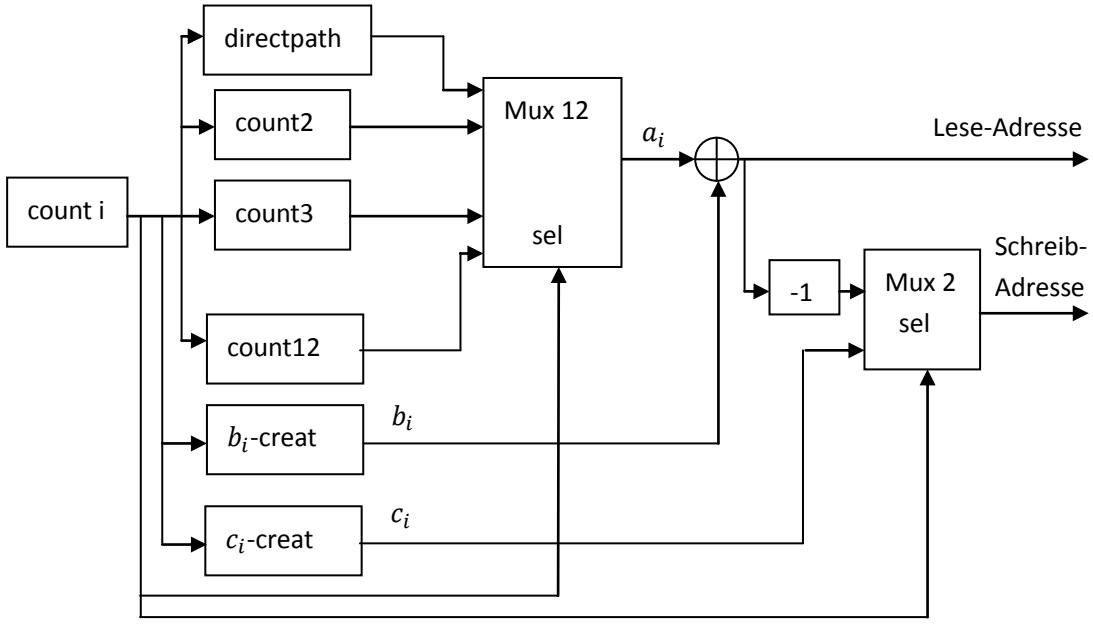


Abbildung 37 Realisierungsdiagramm von Vergabe der Lese- und Schreib-Adressen

Tabelle 5 gibt die Adressvergaben vom Verschachteler wieder. Die Variation von a_i wird als Zahlenwert in der Tabelle 6 beschrieben.

Tabelle 6 Zuordnungstabelle der Lese- und Schreib-Adressen

Zweig	Erste Adresse	Letze Adresse	Zahlenwert
0	0		0
1	1	18	0~17× 1
2	19	53	0~17× 2
3	54	105	0~17× 3
4	106	174	0~17× 4
5	175	260	0~17× 5

6	261	363	$0\sim17\times 6$
7	364	483	$0\sim17\times 7$
8	484	620	$0\sim17\times 8$
9	621	774	$0\sim17\times 9$
10	775	945	$0\sim17\times 10$
11	946	1133	$0\sim17\times 11$

Wenn die Adressvergabe fertig zugeordnet wurde, dann können die Daten eingegeben und ausgegeben werden.

Zum ersten Zeitpunkt: Die Daten werden in der ersten Adresse des ersten Zweigs eingegeben, das heißt: Die Lese-Adresse des FIFO-Schreiberegisters ist 0, die Schreib-Adresse ist auch 0.

Zum zweiten Zeitpunkt: Die Lese-Adresse des FIFO-Schreiberegisters ist 18, die Schreib-Adresse ist 1.

.....

Zum zwölften Zeitpunkt: Die Lese-Adresse des FIFO-Schreiberegisters ist 1133, die Schreib-Adresse ist 946.

So erreicht man die Faltungsverschachtelung in der Software.

6.3 Verarbeitung mit Symbolen

6.3.1 Umwandlung von Byte zu m-Symbolen

Nach der Faltungsverschachtelung werden die Daten den m-Symbolen zugeordnet. Bei 2^m-QAM Modulation sind *k* Bytes in *n* Symbole mappen: $8k=n\times m$.

In der Software wird ein FIFO-Schieberegister verwendet, um dem Umwandlungs-Prozess zu realisieren. Bevor die Daten im FIFO-Schieberegister geschrieben werden, wird zuerst der Schreibtakt jede 3 Byte-Perioden wiederholt, damit jeweils 3 Bytes Daten zu 24 Bits zusammengefasst werden. Dann werden die 24 Bits in das FIFO-Schieberegister geschrieben. Der Lesetakt soll jeweils 4 Symbol-Perioden wiederholen. In jedem Lesetakt werden 6 Bits gelesen. Nach der 4 Symbol-Perioden können der 24 Bits Data aus dem FIFO-Schieberegister ausgelesen werden. Der Zuordnungscode ist wie folgt:

3: data_out[5:0] = data_24_out[23:18]; *2: data_out[5:0] = data_24_out[17:12];*

1: data_out[5:0] = data_24_out[11:6]; *0: data_out[5:0] = data_24_out[5:0];*

Wobei *„data_24_out“* das durch 3 Symbol-Perioden gelesene 24 Bits-Datenwort ist und im FIFO-Schieberegister gespeichert wird.

Die Datenrate am Ausgang dieses Blocks ist: $\mathrm{r_{aus}} = 8/6 \times 204/752 \times \mathrm{r_{ein}}$, dudurch

werden die durch den Block „Synchronisation“ erzeugten Lücken gelöscht.

6.3.2 Differenzcodierung

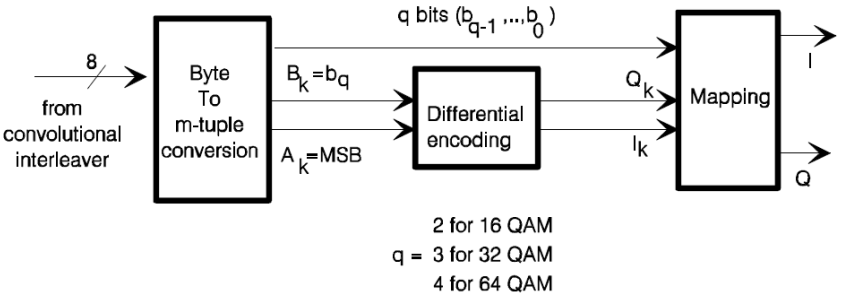


Abbildung 38 Realisierungsschaltung der Differenzcodierung [68]

Die zwei höchstwertigen Bits von jedem m-Symbol werden der Differenzcodierung zugeführt, um eine $\pi/2$ rotationsinvariante QAM-Konstellation zu erhalten. Die Differenzcodierung der beiden höchstwertigen Bits werden durch die folgenden Booleschen Ausdrücke gegeben:

$$I_k = \left(\overline{A_k \oplus B_k}\right)(A_k \oplus I_{k-1}) \; + (A_k \oplus B_k)(A_k \oplus Q_{k-1}) \qquad (6\text{-}4)$$

$$Q_k = \left(\overline{A_k \oplus B_k}\right)(B_k \oplus Q_{k-1}) \; + (A_k \oplus B_k)(A_k \oplus I_{k-1}) \qquad (6\text{-}5)$$

6.3.3 Konstellation

Die Koordinatenwerte der Konstellation werden durch binäre Zahlen dargestellt, so dass der nachfolgende Interpolationsfilter sie verarbeiten kann.

Abbildung 39 zeigt die Konstellation für 64QAM nach dem DVB-C Standard(Norm 30_429).

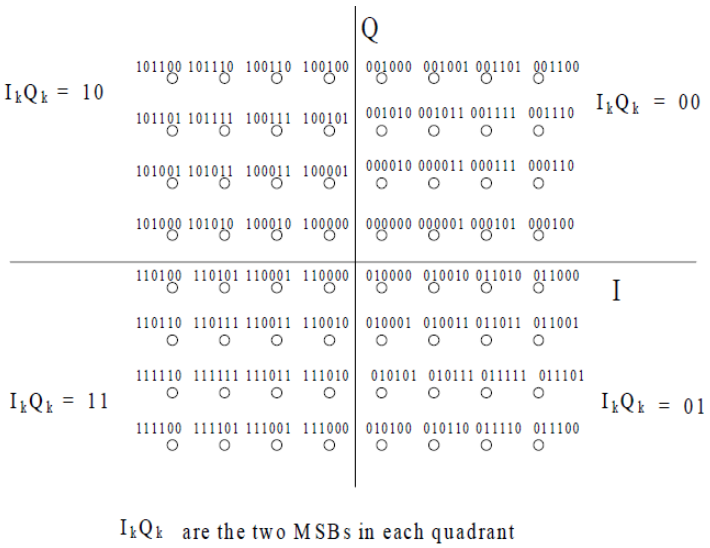


Abbildung 39 Realisierungsmethode der Konstellation für 64QAM [69]

Die Realisierungsmethode der Konstellation ist, dass wenn der eingegebene Wort z.B. 101100 (siehe Abbildung 36) ist, dann ist der Koordinatenswert von *I*-Komponente gleich -14 (Binärzahl 1_1110) und der von *Q*-Komponente gleich 14 (binäre Zahl 0_1110), usw. Mit Hilfe der „case“-Funktionen kann die Methode realisiert werden.

6.4 Digitales Filter

Das digitale Filter wurde in MATLAB durch „FDATool“ generiert. Mit MATLAB können die Koeffizienten von einem digitalen Filter berechnet werden, danach können diese Koeffizienten durch „coeff=round(Num/max(abs(Num))*(2^(5-1)-1))“ quantifiziert werden. Abbildung 40 zeigt die Koeffizienten von dem digitalen Filter, die Koeffizienten sind gleich wie „Num“ in der obigen Formel. Diese Koeffizienten werden nachher in Quartus II verwendet.

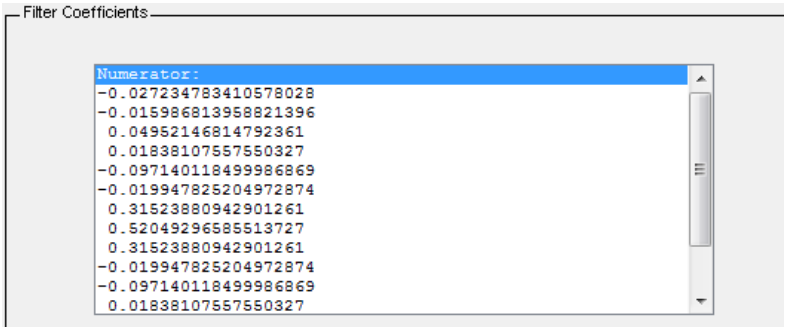


Abbildung 40 Koeffizienten von digitalem Filter

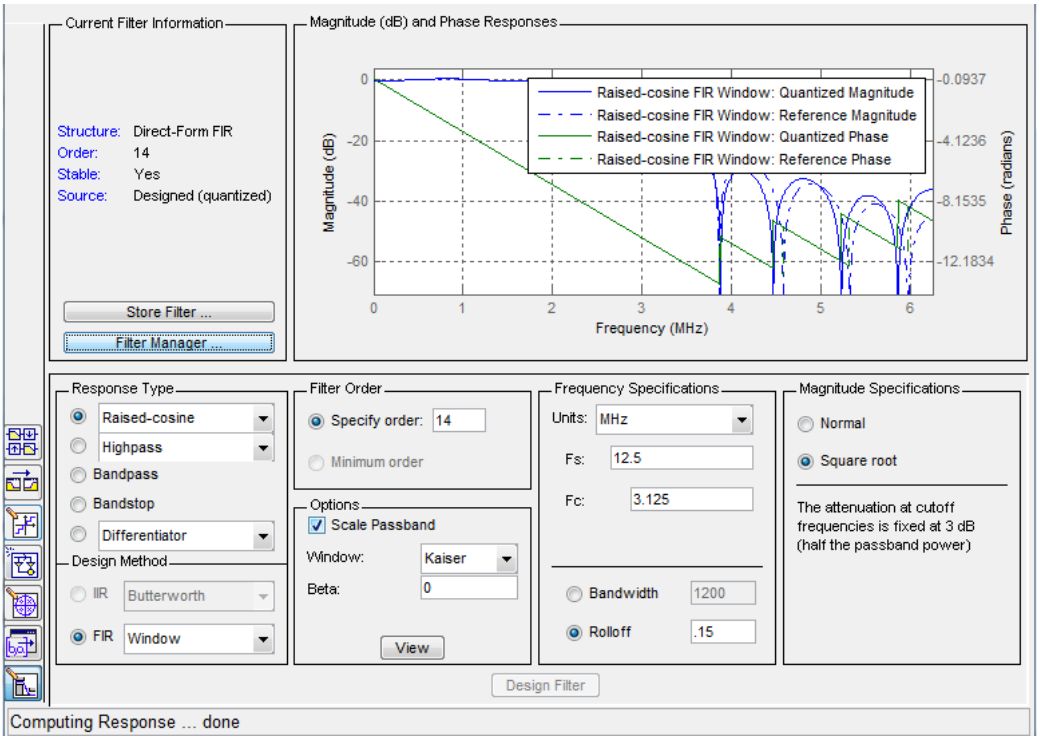


Abbildung 41 Amplituden- und Phasenfrequenzgang

Abbildung 41 zeigt die Amplituden- und Phasenfrequenzgänge, weiter ist auch die

Konfiguration von diesem digitalen Filter gezeigt. In dieser Entwicklung wird eine Symbolrate von 6,25Mbd verwendet, deshalb sind die Cut-Off-Frequenz „F_c“ 3,125 MBd und die Sampling-Frequenz „F_s“ 12,5 MBd. In Abbildung 41 kann man sehen, dass das digitale Filter ein Quadratwurzel-Raised-Cosine-Filter ist, der Roll-Off-Faktor davon ist 0,15, wie im DVB-C Standard(Norm 300_429) gefordet. Als Stufenzahl wird 14 genommen. Abbildung 42 zeigt die Impulsantwort.

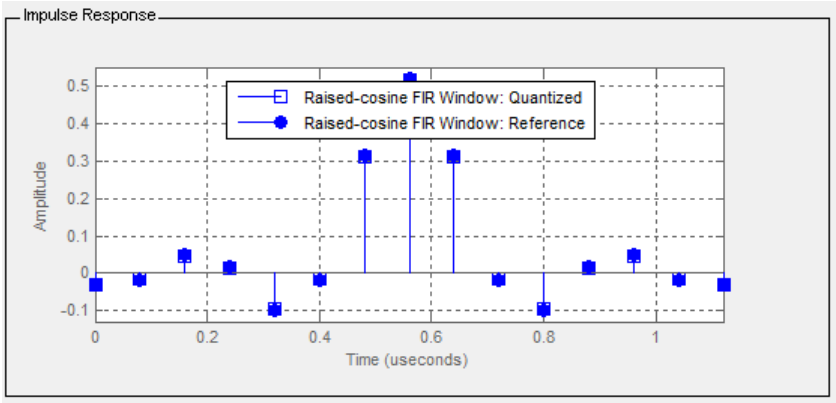


Abbildung 42 Impulsantwort des Basisbandfilters

6.5 Reset des AD9857

Wenn „reset“ high ist, dann wird AD9857 zurückgesetzt, und der Hochpegel sollte mindestens 5 Referenztakte auftreten. Nach der Zurücksetzung wird „reset“ auf low geschaltet, um den AD9857 normal zu betreiben. In dieser Entwicklung werden 10 Referenztakte genommen.

6.6 Serielle Kommunikation zwischen FPGA und AD9857

Die Kommunikationsschnittstelle des AD9857 arbeitet im SPI- (Serial Peripheral Interface) Modus. Die seriellen Pins der SPI-Kommunikationsschnittstelle bestehen hauptsächlich aus: $\overline{CS}$, SCLK, SYNCIO, SDIO, SDO.

Durch SPI-Schnittstelle kommunizieren FPGA und AD9857 (siehe Abbildung 43).

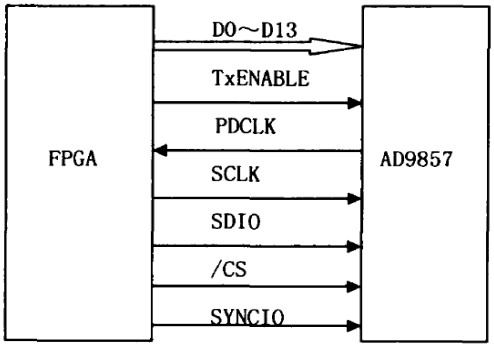


Abbildung 43 serielle Kommunikationsschnittstelle

In der Software sind zwei Module „Configure“ und „spi“ für die serielle Kommunikation enthalten. Das Modul „Configure“ wird benutzt, um die Zuordnungen der Konfigurationswörter des AD9857 zu realisieren. In diesem Modul werden 5 Register für die Befehlsbytes und 8 Register für die Datenbytes definiert. Um die serielle Kommunikation zu erreichen, wird noch ein Register „data_1“ verwendet. In dem Register werden die Konfigurationsinformationen gespeichert. Die Realisierungsmethode ist, dass zum ersten Zeitpunkt das erste Befehlsbyte dem „data_1“ zugeführt wird. Zum zweiten Zeitpunkt wird das entsprechende Datenbyte dem „data_1“ zugeordnet. Zum dritten Zeitpunkt kommt das zweite Befehlsbyte zum „data_1“, dann folgt das zweite Datenbyte, so geht der Ablauf weiter. Das Modul „spi“ wird verwendet, um zuerst die im „data_1“ gespeicherten Daten durch einen Schieberegister die seriellen Daten zu konvertieren, danach die seriellen Daten durch „SDIO“ vom FPGA zum AD9857 zu senden.

6.6.1 Ablaufdiagramm für die serielle Kommunikation

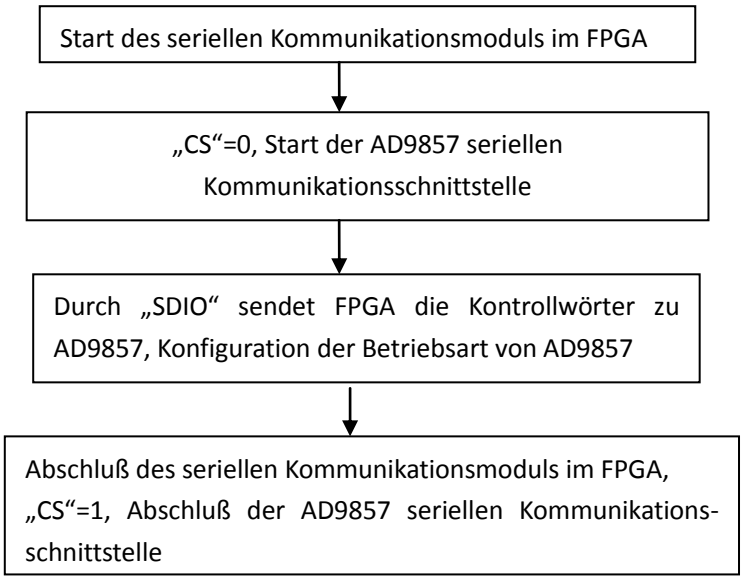


Abbildung 44 Ablaufdiagramm von serieller Kommunikation

„CS“ ist das Freigabesignal und mit logisch-0 aktiv. In der Software wird ein anderes Signal „Start“ definiert und mit einem Schalter im FPGA-Entwicklungsboard verbunden. Wenn dieser Schalter ausgelöst wird, wird das Freigabesignal „CS“ auf low gesetzt, um das serielle Kommunikationsmodul starten zu können. Durch „SDIO“ werden die seriellen Kontrollwörter zum AD9857 gesendet. Wenn der FPGA die seriellen Kontrollwörter fertig übertragen hat, wird sofort das Signal *„CS“* auf high gesetzt, um die Kommunikation abzuschließen.

6.7 Parallele Kommunikation zwischen FPGA und AD9857

Im QAM-Modus sind die I- und Q-Datenzweige benutzbar. „PDCLK/PDU“ (Parallel Data Clock) wird als Ausgangstakt für die parallelen Daten verwendet. Die Eingabedaten werden mit der ansteigenden Flanke von „PDCLK“ in den AD9857 eingeschrieben. Ein I-Datenpaket und ein Q-Datenpaket bilden eine innere Datengruppe, sie übertragen an den internen Datenpfade in einem parallelen Verfahren.

6.7.1 Ablaufdiagramm von paralleler Kommunikation

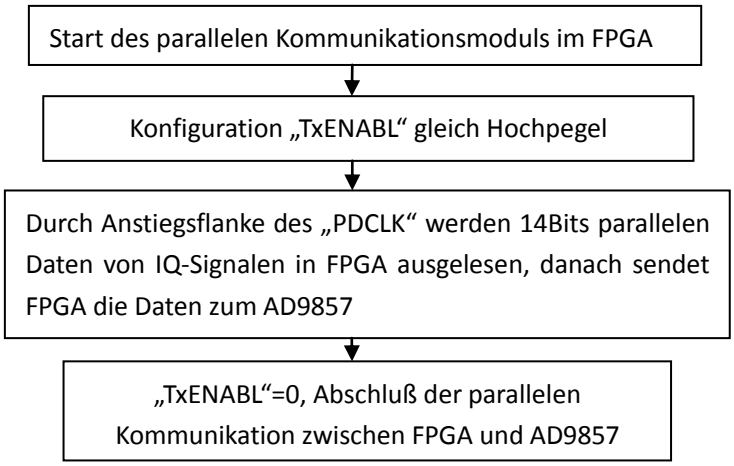


Abbildung 45 Ablaudiagramm von paralleler Kommunikation

„TxENABL“ ist ein Freigabesignal und mit logisch-1 aktiv. Es wird mit einem Schalter im FPGA-Entwicklungsboard verbunden. Wenn dieser Schalter ausgelöst wird, wird das parallele Kommunikationsmodul gestartet. Danach wird das Rückkopplungssignal „PDCLK“ durch den AD9857 erzeugt und zum FPGA als Dateneingangstakt gesendet. Daher können die *I*- und *Q*-Daten mit der ansteigenden Flanke von „PDCLK“ abwechselnd zum AD9857 übertragen. Um diesen Schritt zu realisieren, ist noch ein binäres Signal „bit“ benötigt. Wenn „bit“ gleich 1 ist, dann werden die I-Daten zum AD9857 gesendet; Wenn „bit“ gleich 0 ist, dann werden die Q-Daten zum AD9857 gesendet; Durch das Ausschalten des „TxENABL“ wird die parallele Kommunikation abgeschlossen werden.

6.8 Realisierung der SPI-Schnittstelle in FPGA

Im FPGA-Entwicklungsboard gibt es keine SPI-Schnittstelle, deswegen muss eine SPI-Schnittstelle im FPGA realisiert werden, damit die Kommunikation mit dem AD9857 erfolgen kann.

6.8.1 Realisierung von SPI

Der FPGA kann die Konfiguration von AD9857 dadurch realisieren, dass er die Konfigurationsports („SCLK“, „CS“, „SDIO“) von AD9857 gesteuert.

Das Konfigurationsprinzip ist wie in Abbildung 46 dargestellt. Die Daten, die in die internen Register von AD9857 geschrieben werden sollen, können durch das Taktsignal in das Schieberegister verschoben werden. Anschließend müssen die 16 Bits Daten (8 Bits für Befehlsbyte und 8 Bits für Datenbyte) Bit für Bit zu dem Port „SDIO“ im AD9857 ausgegeben werden, um die Konfigurationen der inneren Register von AD9857 zu erfüllen.

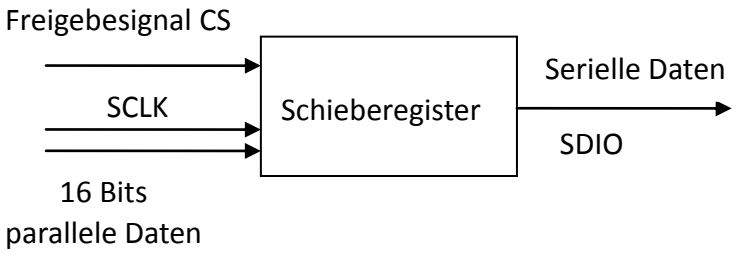


Abbildung 46 Konfigurationsprinzip für SPI

Die Daten des Schieberegisters werden im Voraus berechnet. Sie beinhalten nicht nur 8 Bits Befehlsdaten, die die Betriebsarten und Adressen der inneren Register umfassen, sondern auch die in die entsprechenden Adresse zu schreibenden 8 Bit-Konfigurationswörter.

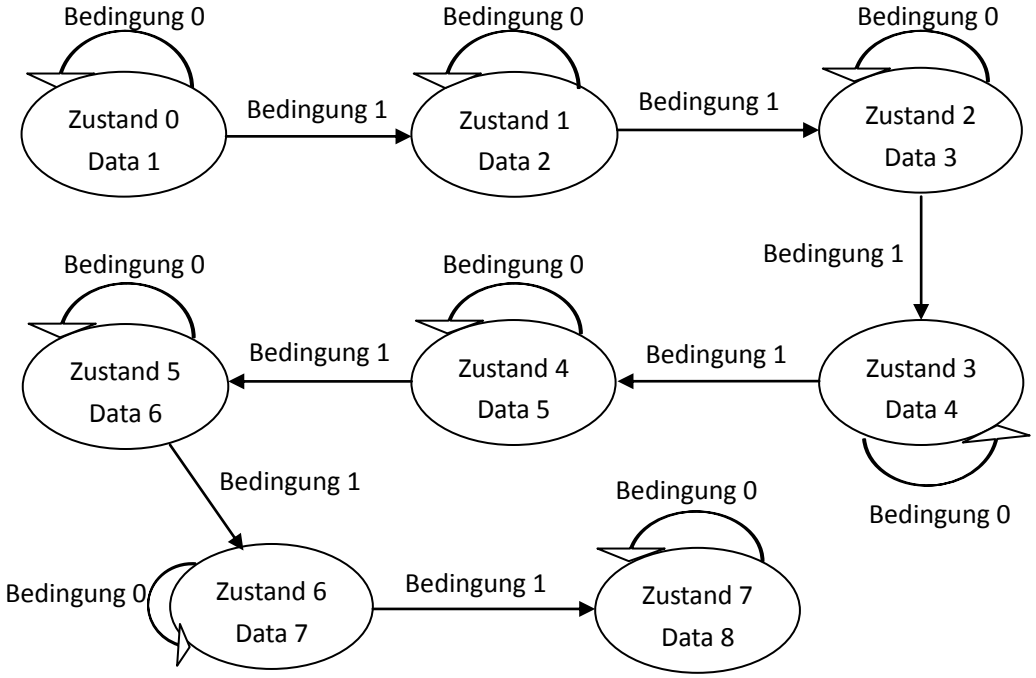


Abbildung 47 Arbeitsprozess der Zustandsmaschine für SPI

Durch den Mechanismus einer Zustandsmaschine können die Konfigurationswörter der inneren Register bei den unterschiedlichen Zuständen im Schieberegister geschrieben werden. Das Arbeitsverfahren der Zustandsmaschine ist in Abbildung 47 gezeigt. Bedingung 1 in der Abbildung ist bei fallender Flanke von SCLK. Bedingung 0 ist bei ansteigender Flanke von SCLK.

In diseser Konstruktion werden ausschließlich 8 Register benutzt, deshalb hat diese Zustandsmaschine nur 8 Zustände. Wenn mehr interne Register vom AD9857 benötigt werden, werden nur die Zustände der Zustandsmaschine erhöht.

6.9 Frequenzkontrolle

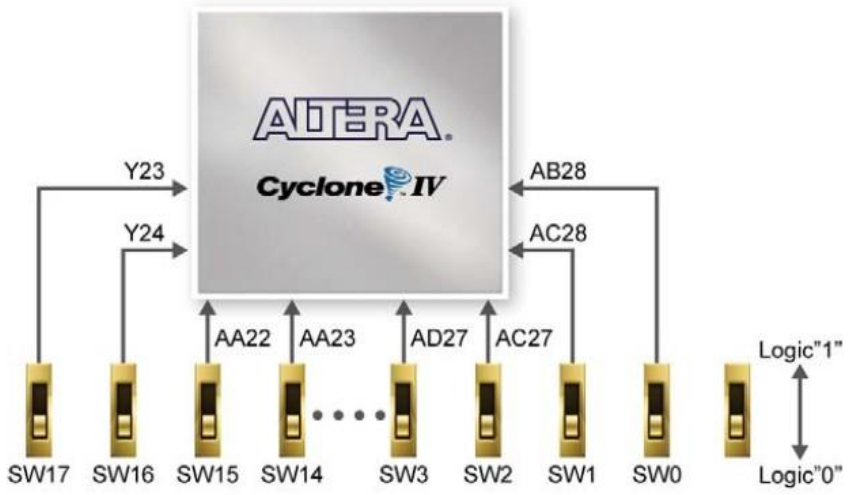


Abbildung 48 Schaltung von Schalter im FPGA [70]

Über die 10 vorhandenen Schalter (SWITCH0 bis SWITCH9) soll die Frequenz eingestellt werden können. In dieser Entwicklung wird eine Schrittfrequenz von 0,1MHz vorgesehen. Der Änderungsbereich ist von 0000000000 bis 1111111111, deshalb ist die maximale Frequenz 2^10×0.1MHz, d.h. 102.4 MHz. In dieser Entwicklung wird auch eine Grundfrequenz von 5 MHz gefordert. Dann liegt der einstellbare Frequenzbereich schließlich zwischen 5 MHz und 107.4 MHz.

In der Software wurde diese Funktion mit den folgenden Funktionen realisiert:

```
assign freq1=({30'b0,freq}+{34'b0,6'b110010})<<28;
assign freq2=freq1/125;
```

„freq“ ist die tatsächlich eingegebene Frequenz, die mithilfe der wichtigen Gleichung

$$\text{FTWORD}=\frac{freq}{SYSCLK}*2^{32} \qquad (6\text{-}6)$$

berechnet werden kann, dabei ist „SYSCLK“ der Systemtakt vom AD9857, hier also 200 MHz. Die Grundfrequenz ist 5 MHz, Schrittfrequenz ist 0,1 MHz. Daraus wurden

die oberen Funktionen gewonnen.

6.10 Gesamtsystem

Das gesamtes System dieser Entwicklung wurde zuerst mit der Hardwarebeschreibungssprache geschrieben, danach wurde es durch ein schematisches Bild in Quartus II realisiert. Das Schema ist in Abbildung 49 dargestellt.

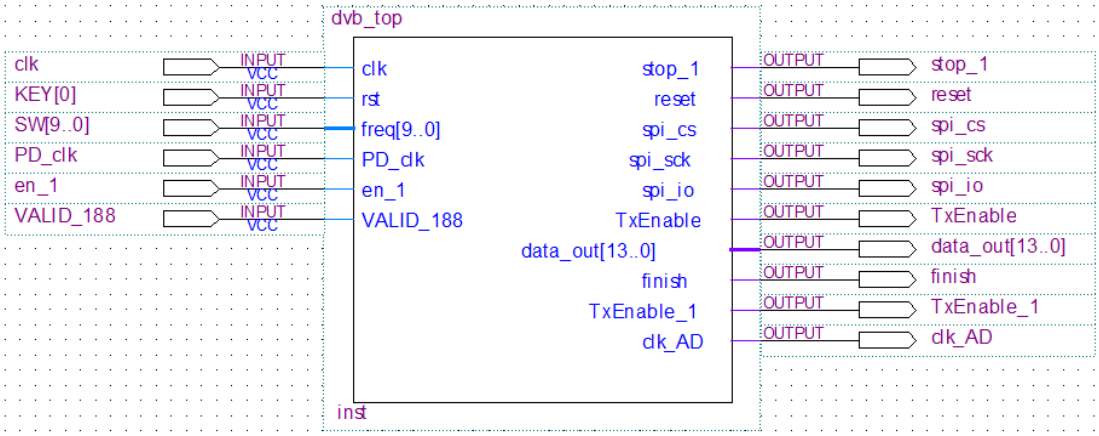


Abbildung 49 Schema des Gesamtsystems

Block *dvb_top* enthält alle externen Eingangspins und Ausgangspins dieses Systems. Diese Pins sind den entsprechenden Pins im FPGA-Entwicklungsboard zugeordnet. Die betreffenden Eingangspins werden mit Schaltern oder Druckknöpfen verbunden, um die FPGA-Schaltung zu kontrollieren. Die betreffenden Ausgangspins werden GPIO-Pins zugeordnet, danach werden diese GPIO-Pins durch Leitungen mit den entsprechenden Pins im AD9857-Entwicklungsboard verbunden, um die Kontrolle vom AD9857 und die Datenübertragung zu realisieren. Beispielsweise wird der Eingangspin „rst“ im Bauteil mit einem Schalter KEY[0] im FPGA-Entwicklungsboard verbunden. Mit diesem Schalter kann die Zurücksetzung dieses Systems ausgelöst werden. Der Ausgangspin „spi_io“ entspricht einem GPIO-Pin des FPGA-Entwicklungsboards und wird mit dem SDIO-Pin des AD9857-Entwicklungsboards verbunden, damit die Konfigurationswörter übertragen werden können.

Port name	Daten Breite	Output /input	Funktionsbeschreibung
clk	1	input	Zeitsignal
rst	1	input	Zurücksetzsignal
freq[9..0]	10	input	Frequenz wird draußen kontrolliert.
PD_clk	1	input	Rückkopplungstaktsignal vom AD9857 zum FPGA
en_1	1	input	en_1=1, wird die Konfiguration erlaubt

en	1	input	en=1, beginnt man serielle Daten zu senden
stop_1	1	output	Wenn die Konfiguration unfertig ist, ist stop_1= 1
reset	1	output	Zurücksetzsignal für AD9857
spi_cs	1	output	Chipauswahlsignal des AD9857, 0 ist aktiv
spi_sck	1	output	Taktsignal für Kommunikationsschnittstelle
spi_io	1	output	Kontrollwörter bei Bit für Bit zum AD9857
TxEnable	1	output	Bei TxEnable=1 sind IQ-Daten zu senden
data_out [13..0]	14	output	Parallen ausgegebene 14 Bits I / Q Daten nach Konstellation
finish	1	output	wenn ein Register fertig konfuguriert wird, dann ist finish=1

Tabelle 7 Siganle vom gesamten System

In der Tabelle 7 werden die Signale von diesem gesamten System bezeichnet.

Abbildung 50 zeigt die Schaltung vom gesamten System in RTL(Real Time Logistics). Darin kann man einige wichtige Module erkennen.

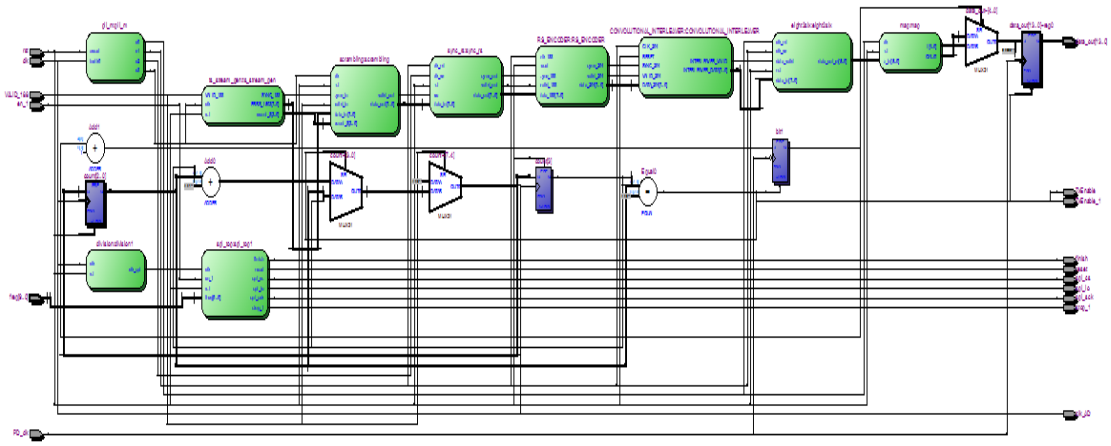


Abbildung 50 die Schaltung des gesamten Systems in RTL

In dieser Entwicklung wurde zuerst der JTAG-Modus verwendet. Als die Codierung keine Fehler mehr aufwies, wurde der AS-Modus benutzt und der Quellcode ist im Konfigurationsdevice EPCS64 speichert.

7 Ergebnisse

7.1 Simulationsergebnisse

Im Folgenden werden die durch die Simulationssoftware „Modelsim-Altera 6,6d“ erhaltenen Ergebnisse präsentiert.

7.1.1 Ergebnis der digitalen Basisbandsignalverarbeitung

7.1.1.1 Ergebnis der Erzeugung von TS

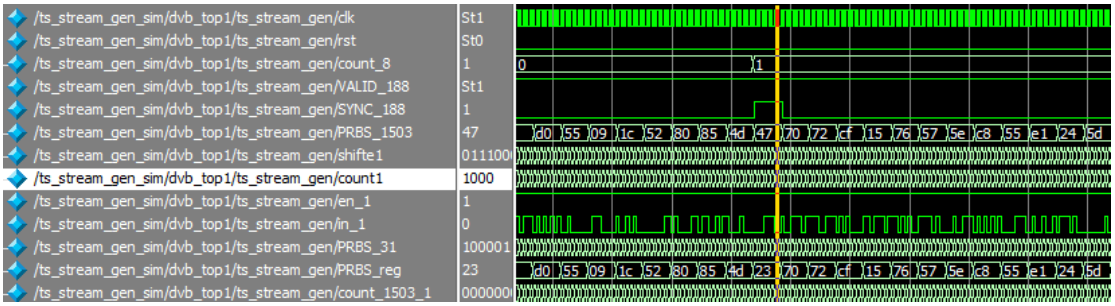


Abbildung 51 Ergebnis der Erzeugung von TS

Im Register „PRBS_1503“ steht der TS-Datenstrom an. Wie in Abschnitt 4.1 erklärt, hat der TS-Datenstrom ein Synchronisationsbyte 47H und 187 Bytes Informationsdaten.

7.1.2 Ergebnis der Kanalcodierung

7.1.2.1 Ergebnis der Invertierung und Verwürfelung

Abbildung 52 gibt das Ergebnis der Verwürfelung wieder. Darin kann man sehen, dass das Synchronisationsbyte von dem ersten Paket jedes Rahmens invertiert, also von 47H zu B8H verändert wurde.

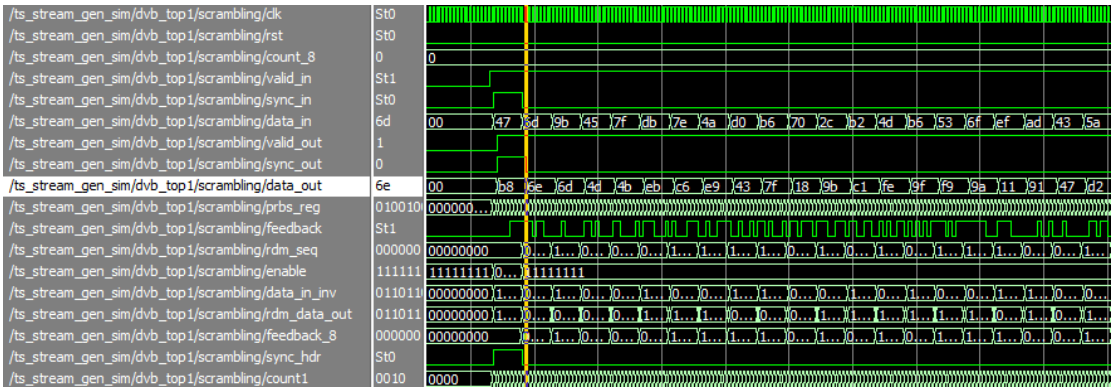


Abbildung 52 Ergebnis der Invertierung

Die anderen Synchronisationsbytes bleiben unverändert. In Abbildung 53 kann man sehen, dass die Synchronisationsbytes 47H in den Datenströmen stehen. Bei der Verwürfelung dürfen die Synchronisationsbytes nicht verwürfelt werden, d.h. B8H und 47H sind unverändert. Andere Daten der TS-Datenströme werden mit PRBS verwürfelt (siehe Abschnitt 4.2.1.1).

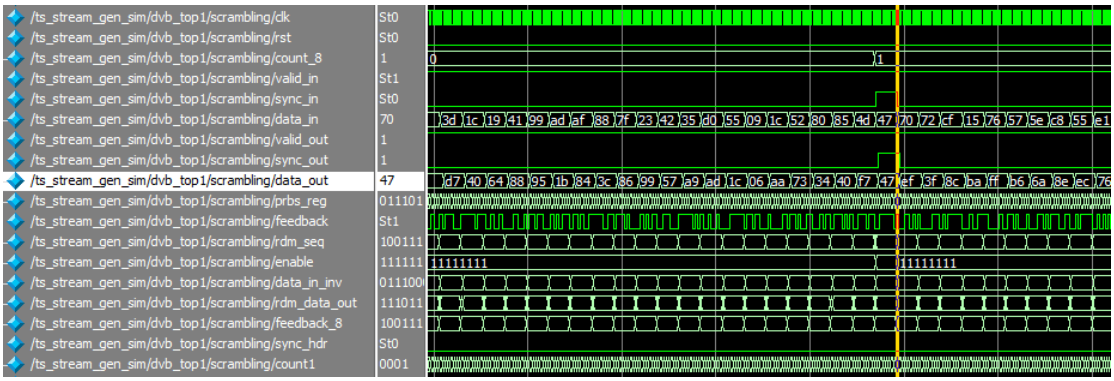


Abbildung 53 Ergebnis der Verwürfelung

7.1.2.2 Ergebnis von Synchronisation

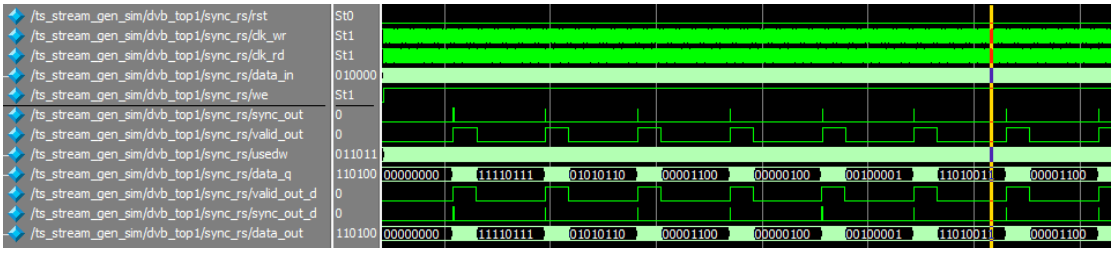


Abbildung 54 Ergebnis von Synchronisation

Wie in Abschnitt 4.2.1.1 erklärt, hat das Ergebnis nach der Synchronisierung 548 Bytes Lücken und 204 Bytes Informationen.

7.1.2.3 Ergebnis der Reed-Solomon-Codierung

Abbildung 55 zeigt das Ergebnis der Reed-Solomon-Codierung. Bei der RS-Codierung müssen die Synchronisationsbytes und die Nutzdaten unverändert bleiben und es gibt zusätzliche 16 Bytes Daten mehr für die Fehlerkorrektur.

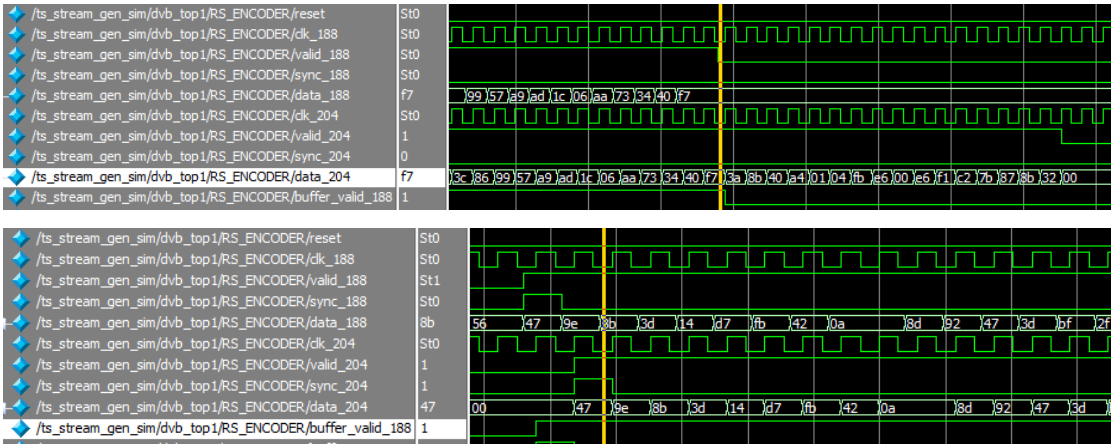


Abbildung 55 Ergebnis der Reed-Solomon-Codierung

7.1.2.4 Ergebnis der Faltungsverschachtelung

Abbildung 56 zeigt das Ergebnis der Faltungsverschachtelung. In dieser Abbildung

kann man sehen, dass die Synchronisationsbytes unverändern geblieben sind und die Nutzdaten verschachtelt wurden, die Fehlerkorrekturcodes eingeschlossen.

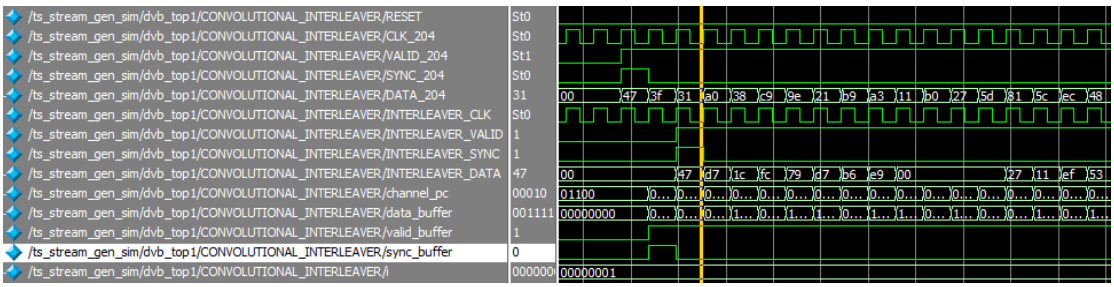


Abbildung 56 Ergebnis der Faltungsverschachtelung

7.1.3 Ergebnis der Umwandelung von Byte zu m-Symbolen

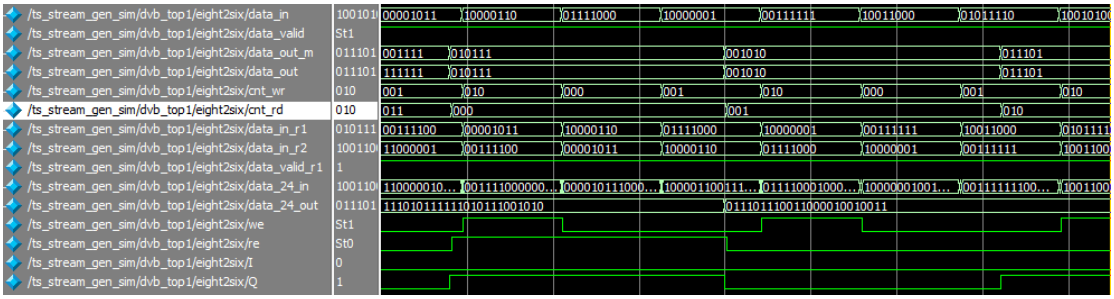


Abbildung 57 Ergebnis der Wandelung von Byte zu m-Symbolen

Jeweils 3 Bytes (24 Bits) Daten werden genommen, um vier 6 Bits Daten zu bekommen.

7.1.3.1 Ergebnis der Konstellation

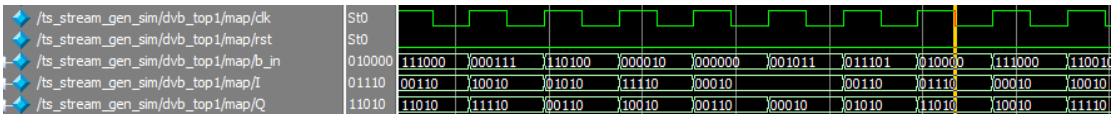


Abbildung 58 Ergebnis der Konstellation

Jeweils „b_in“ ist ein 6 Bits Wort und enthält 3 Bits I-Data und 3 Bits Q-Data. Wie in Abschnitt 6.3.3 beschrieben wird, bedeutet „b_in“ = 000111 (siehe Abbildung 58), z.B. *I*-Data gleich 01010 und *Q*-Data gleich 00110 (einen Takt verzögert). Der Koordinatenswert im Konstellationsdiagramm ist *(10, 6)*.

7.1.3.2 Ausgabe von I- und Q-Daten nach dem Filter

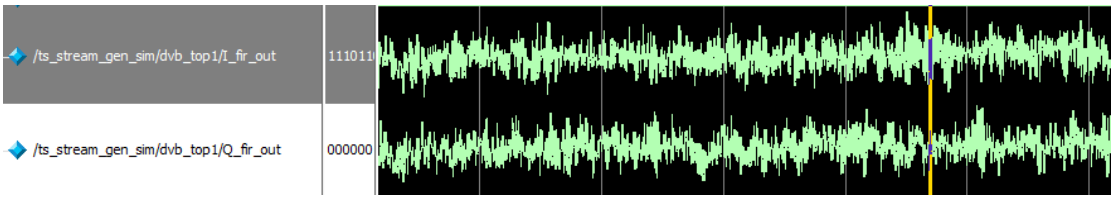


Abbildung 59 Ergebnis der Ausgabe von I- und Q-Daten nach dem Filter

7.1.4 Analysierung von Synchronisationsbytes

Aus diesem Ergebnis kann man sehen, dass die Synchronisationsbytes nach der Verwürfelung, der RS-Codierung und Faltungsverschachtelung bestehen bleiben.

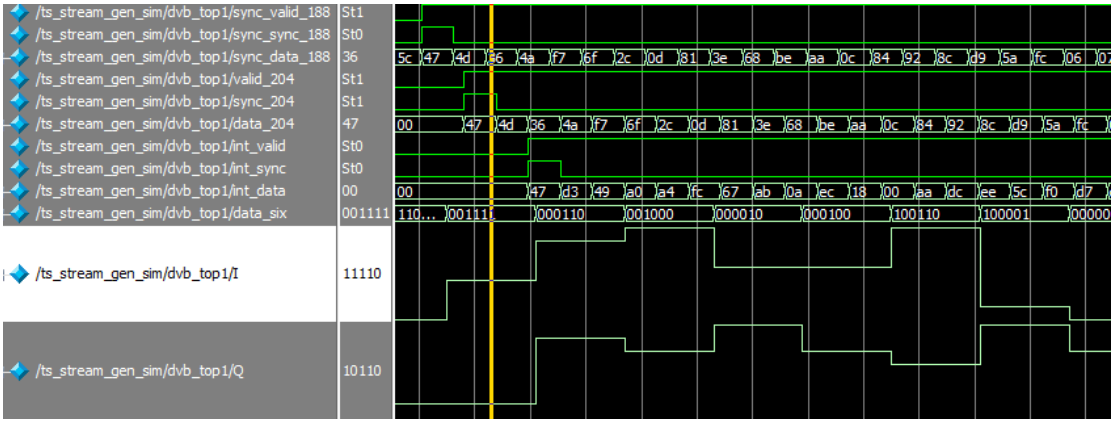


Abbildung 60 digitale Basisbandsignalverarbeitung

7.1.5 Ergebnis der Konfiguration des AD9857

Die Zurücksetzung vom AD9857 erfolgt wie in Abbildung 64 gezeigt.

Aus diesem Bild können wir sehen, dass der AD9857 nach 10 Referenztaktes zurückgesetzt wird.

Abbildung 61 Rücksetzen von AD9857

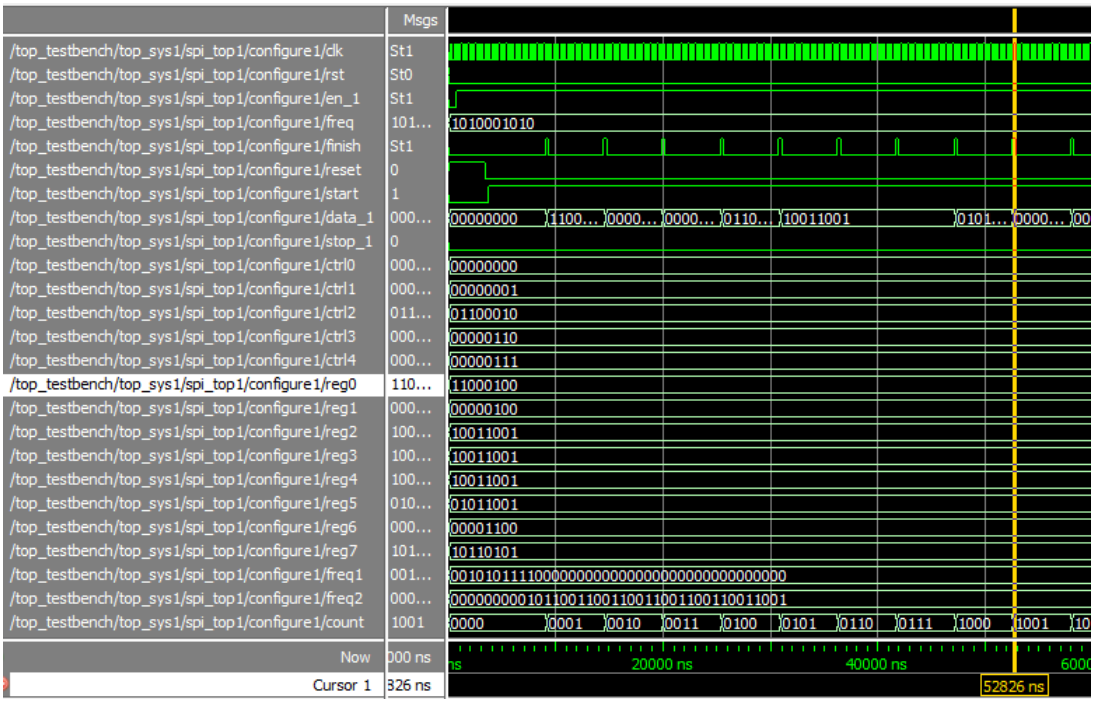


Abbildung 62 Konfigurationen der inneren Register

„en_1“ ist das Signal für die Zulässigkeit der Konfigurationen. Wenn „en_1“ gleich 1

ist, dann wird die Konfiguration erlaubt. Nach der Zurücksetzung wird ein high-Pegel zu „start“ gegeben und zu diesem Zeitpunkt ist „en_1“ schon gleich 1, dann kann die Konfiguration tatsächlich gestartet werden.

Aus dieser Abbildung kann man sehen, dass das Ergebnis der Konfiguration der inneren Register von AD9857 korrekt ist.

7.1.6 Ergebnis der SPI-Kommunikation

Nach der Zurücksetzung des AD9857 wird das logisch-0 aktive Chipauswahlsignal „spi_cs“ auf low gesetzt, um die Kommunikation über die SPI-Schnittstelle anzufangen. „spi_clk“ ist das Taktsignal für die serielle Kommunikation. „spi_io“ entspricht den ausgegebenen Kontrollwörtern zum AD9857. Die Kontrollwörter im Port „data_1“ werden bei der fallenden Flanke von „spi_clk“ Bit für Bit an „spi_io“ mit dem MSB-Modus gesendet.

In dieser Arbeit wird zuerst der Befehl „00000000“ vom FPGA zum AD9857 übertragen, dann steht in der Abbildung 63 ein acht Referenztakte dauernder niedriger Pegel an. Danach wird das Kontrollwort „10000100“ übertragen. In dieser Abbildung kann man sehen, dass es zuerst einen 1 Referenztakt dauernden hohen Pegel gibt, dann werden 4 Referenztakte mit niedrigem Pegel ausgegeben, anschließend wird ein 1 Referenztakt dauernder hoher Pegel ausgegeben, schließlich kommt ein 2 Referenztakte dauernder niedriger Pegel. Schrittsweise werden die anderen Kontrollwörter übertragen.

Wenn alle inneren Register fertig eingerichtet sind, dann wird „stop_1“ als ein Endsignal auf 1 gesetzt.

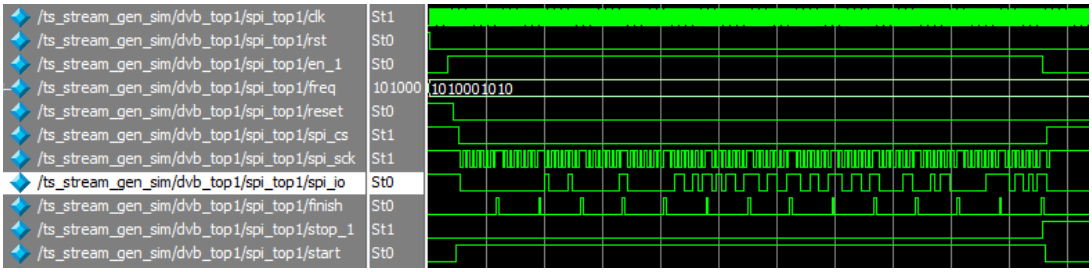


Abbildung 63 SPI-Kommunikation

Aus dieser Abbildung kann man sehen, dass das Ergebnis der SPI- Kommunikation auch korrekt ist.

7.1.7 Überprüfung der Clock-Systeme

In dieser Arbeit gibt es insgesamt 7 unterschiedliche Taktsignale. Das erste ist die

Frequenz vom Quarzoszillator im FPGA-Entwicklungsboard mt 50 MHz. Das zweite ist der Referenztakt „clk_AD“ für den AD9857. Das dritte Taktsignal ist das Zeitsignal für die Module „Umwandlung von Byte zu Symbol“ und „Differenzcodierung und Mapping“. Der Takt „clk_188“ für das Modul „sync_rs“. Das vierte Zeitsignal „clk_204“ ist für die Module „RS-Codierung“ und „Faltungsverschachtelung“. Das Zeitsignal „clk_5m“ ist für die „Erzeugung von TS-Datenströmen“ und „Verwürfelung“. Das letzte Taktsignal „clk_spi“ ist der Takt DER SPI- Kommunikationsschnittstelle.

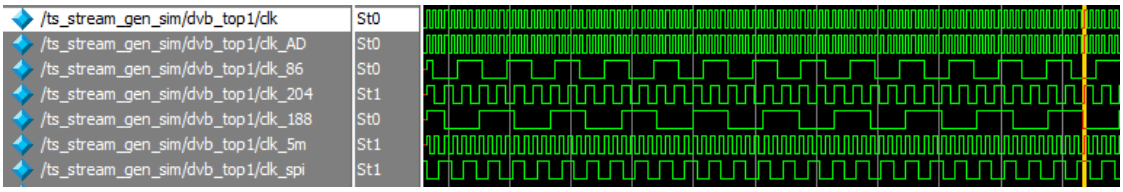


Abbildung 64 Clock-System

7.1.8 Gesamtergebnis

In Abbildung 65 wird das Gesamtergebnis dargestellt.

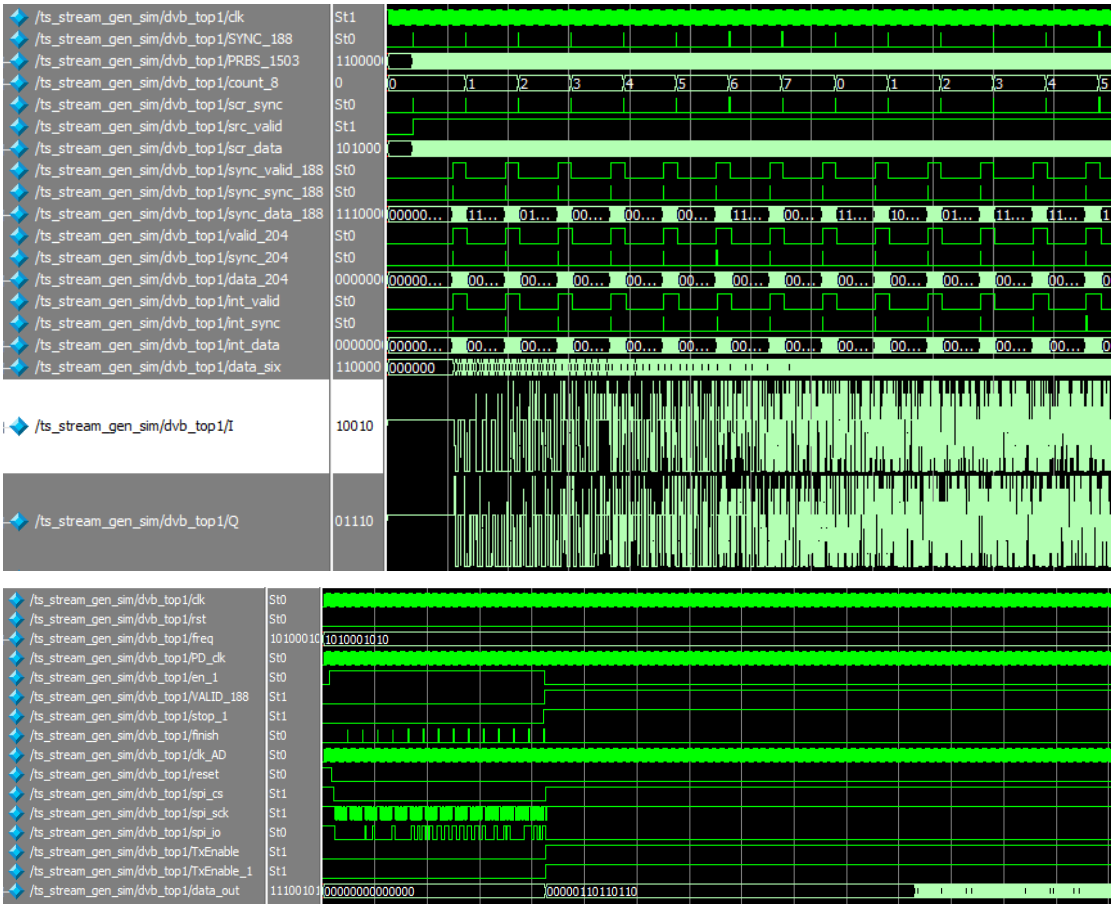


Abbildung 65 gesamtes Ergebnis

In dieser Abbildung können wir deutlich sehen, dass zuerst die SPI-Kommunikations-schnittstelle konfiguriert wird, um die Kontrollwörter Bit für Bit zum AD9857 zu

übertragen, damit die Arbeitsart und Frequenz usw. vom AD9857 konfiguriert werden. Wenn die Kontrollwörter fertig übertragen wurden, dann wird „stop_1“ auf 1 gesetzt. Zur gleichen Zeit wird „spi_cs“ auf logisch 1 gesetzt, um die serielle Kommunikationsschnittstelle zu schließen. Wenn „stop_1“ gleich 1 ist, dann wird „VALID_188“ mit Hochpegel eingestellt, damit die parallele Übertragung von I- und Q-Daten stattfinden kann. Nach der Öffnung der Datenübertragungsschnittstelle wird „TxEnable“ auf high gesetzt, dann wurden die I/Q Daten unter Zuhilfennahme des Flag-Signals „bit“ zeitversetzt zum AD9857 übertragen.

Wenn „bit“ gleich 1 ist, dann werden die I-Daten zu AD9857 gesendet;

Wenn „bit“ gleich 0 ist, dann werden die Q-Daten zu AD9857 gesendet.

7.2 Hardware-Ergebnis

7.2.1 Schaltung und Ergebnis

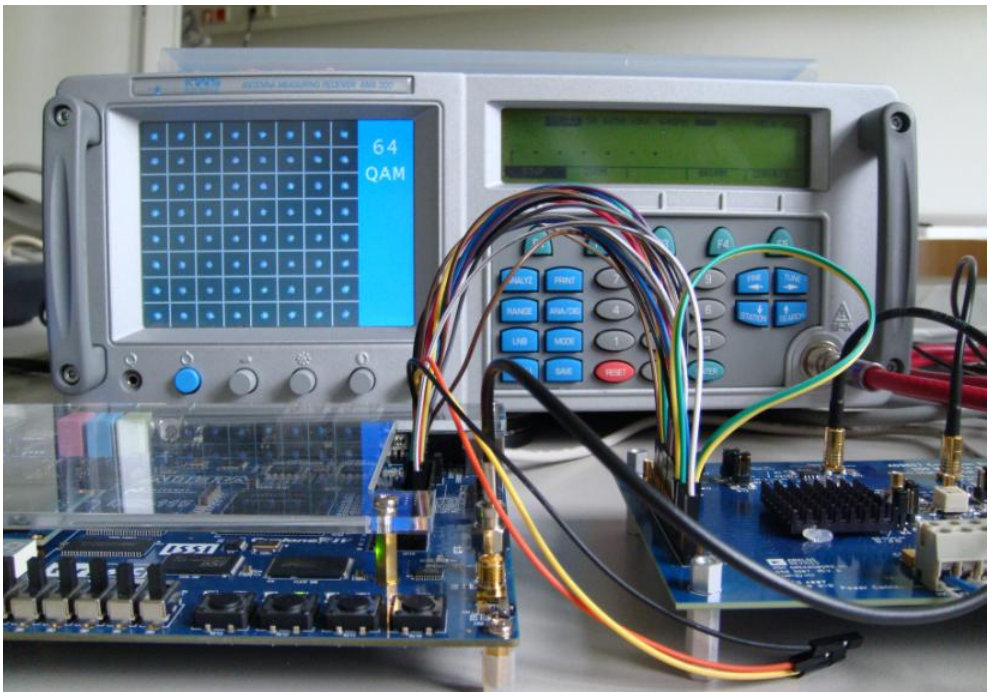


Abbildung 66 Schaltung und Ergebnis

Zur Untersuchung des Ausgangssignals wurde das Messgerät „Antennen-Mess-empfänger AMA 300“ von der Firma KWS verwendet. *„Das AMA 300 ist ein qualitativ hochwertiges Antennen-Messgerät das hinsichtlich Verarbeitung, Funktionen bw. Optionen (zB. COFDM, CI, Drucker) oder Bedienungsfreundlichkeit kaum Wünsche offen läßt.“* [71]

Eigenschaften:

- Für Analog-TV und DVB geeignet

- Frequenzbereich von 5 MHz bis 2150 MHz
- BER(Bitfehlerrate-Messung) und Konstellationsdiagramm für QPSK und QAM
- MER(Modulationsfehler-Messung) für QAM
- S/N-Messung für QPSK

Auf der Abbildung 66 sind links das FPGA-Entwicklungsboard und rechts das AD9857-Entwicklungsboard zu sehen.

Abbildung 66 zeigt auch die gemessene Konstellation der 64QAM-Modulation. Das Konstellationsdiagramm sieht richtig aus.

8 Zusammenfassung

Gemäß der Anforderung wurde ein auf dem programmierbaren Aufwärtsumsetzer AD9857 und dem FPGA Cyclone IV basierender Entwurf der Digitalaufwärtswandlung implementiert. In dieser Arbeit wurden das Hardware-Design und die Software-Konfiguration von FPGA und AD9857-System beschrieben.

1. In dieser Arbeit werden die Basisbandsignale durch ein FPGA verarbeitet. Im FPGA können TS-Datenströme, Kanalcodierung, Mapping, Basisbandfilter usw. nach dem DVB-C-Standard in Software umgesetzt werden. Die Funktionen wurden simuliert. Die Simulationsergebnisse haben die Anforderungen erfüllt.
2. Der FPGA wird auch zur Konfiguration des AD9857 verwendet.
3. In dieser Arbeit wurde ein paralleles Kommunikationsprogramm geschrieben und durch Simulation überprüft. Das Ergebnis zeigt, dass die 14 Bits Daten über eine parallele Datenschnittstelle zum AD9857 gesendet werden können. Danach werden die Interpolation, Filterung und Quadraturmodulation im AD9857 realisiert.
4. Aus den Simulationsergebnissen und dem Hardware-Ergebnis können wir sehen, dass die Entwicklung eines 64QAM-Modulators ist erfolgreich. Das Konstellationsdiagramm sieht perfekt aus. Aber für das DVB-System ist die Entwicklung noch nicht erledigt.

Problem： Es gibt ein offenes Problem in der Arbeit, dass die Synchronisationsbytes durch das Messgerät nicht erkannt werden können, obwohl die Simulationsergebnisse zeigen, dass die Synchronisationsbytes in jedem Paket ordnungsgemäß vorhanden sind. Zur Fehlersuche wurden die folgenden Methoden versucht:

1. Zunächst war die Symbolrate von I/Q-Daten ein unperiodischer Dezimalbruch,

das Messgerät konnte die Symbolrate nicht genau anzeigen. Daher wurde die Symbolrate in der Software verändert, aber es gab keine Verbesserung für das Problem.

2. Die Ordnung, der Roll-off-Faktor und die Kardinalzahl der Quantifizierung vom Basisbandformfilter wurden verändert. Es ergaben sich auch keine Auswirkungen auf die Ergebnisse.
3. Die Interpolationskoeffizienten des Interpolationsfilters wurden modifiziert. Das verbesserte die Konstellation, aber das Problem wurde nicht gelöst.
4. Die Bestandteile der RS-Codierung und der Faltungsverschachtelung wurden vorher mit der Hardwarebeschreibungssprache Verilog HDL erstellt. Um die Richtigkeit der Programmierung zu überprüfen, wurden die entsprechenden IP-Cores im Quartus II benutzt, aber das Ergebnis blieb gleich wie vorher.

Als Ursache wurden folgende Punkte vermutet:

1. Jedes Modul verwendet ein verschiedenes Taktsignal in der Entwicklung. Die Symbolraten zwischen den Modulen sind vielleicht nicht angepasst.
2. Zwischen der Ausgangssymbolrate und der Symbolrate der Umwandlung von Byte zu m-Symbolen(m=6) hat eine Beziehung:

   $$204\times 8=n\times 6;$$

   $$752/fwr=n/frd;$$

   Wobei n die Anzahl von Lesezyklen, fwr der Schreibtakt und frd der Lesetakt ist. Durch die beiden Gleichungen wird $frd=272\times fwr/752$ bekommen. Daraus kann man sehen, wenn die Ausgangssymbolrate einen genauen Dezimalbruch (z.B. 6,25 MBaud) eingestellt wurde, war die Symbolrate der Umwandlung von Byte zu m-Symbolen ein unperiodischer Dezimalbruch. Somit kann zum Überlauf der Symbole im FIFO-Schieberegister führen. Der Verlust der Symbole lässt die Synchronisationsbytes nicht verschlüsseln.
3. Der DA-Wandler hat vielleicht ein Problem der Richtigkeit. Da die Umwandlung vom digitalen Signal zum analogen Signal mehr oder wenigen zu einigen Fehler führt. Diese Fehler wurden durch die interne Eigenschaft des DA-Chips bestimmt und sind schwer zu ändern.

Während dieser Arbeit habe ich nicht nur das an der Hochschule gelernte Wissen geprüft, sondern auch mich trainiert, wie man eine Sache erfassen kann, wie man dafür arbeiten kann und wie man die Sache vollbringen kann.

Durch diese Arbeit habe ich viele Fähigkeit verbessert. Von der Auswahl und der

Bestimmung der Konzeption, der Auswahl des Hardwares bis zu der Auswahl des Softwares und der Hardwarebeschreibungssprache habe ich mein Denkvermögen und meine Fähigkeit des Problemlösens verbessert. Durch diese Arbeit habe ich sowohl mein Verständnis für die Sprache auf Verilog HDL, als auch meine Programmierfähigkeit erhöht. Das war meine erste Erfahrung, mit Verilog HDL ein Programm zu schreiben, deshalb habe ich während dieser Programmierung auf viele Schwierigkeiten gestoßen, damit habe ich viel gelernt.

Durch diese Arbeit wurde mein Wissen gefestigt und erweitert, das ich in der Vorlesung „digitale Nachrichtenübertragung“ gelernt habe. Diese Arbeit lässt mich ein detailliertes Verständnis für das DVB-System haben und erregt mir viele Interesse an dem DVB-System.

Im Vergleich mit der analogen TV-Technik kann die digitale TV-Technik einen besseren Sinnengenuss den Menschen geben, deshalb hat die digitale TV-Technik eine sehr gute Marktaussicht. Weiterhin ist auf dem Markt ein QAM-Modulator mit dem Frequenzbereich ab 5 MHz noch nicht vorhanden, diese Arbeit bietet Dibkom eine Möglichkeit, einen 64QAM-Modulator mit dem Frequenzbereich von 5 MHz bis 107,4 MHz zu benutzen.

Literaturverzeichnis

[1] [3] [8]
Kabelnetz-Handbuch, Richtlinien und Hinweise für die Planung und Installation von Multimedia-Kabelnetzen. 4. Überarbeitete Auflage-Mai 2007. Salzland Druck GmbH. ISBN 978-3-9811630-0-1

[2] http://de.wikipedia.org/wiki/Digital_Video_Broadcasting

[4] http://tvdigital13.wordpress.com/2011/03/14/dvb-c/

[5] http://de.wikipedia.org/wiki/DVB-C

[6] [7][9] [10] [11][12] [13][16] [17] [18]
http://de.wikipedia.org/wiki/Quadraturamplitudenmodulation

[14]http://files.schulbuchzentrum-online.de/onlineanhaenge/files/978-3-14-235050-9-5-l.pdf

[15]http://www.radio-electronics.com/info/rf-technology-design/pm-phase-modulation/8qam-16qam-32qam-64qam-128qam-256qam.php

[19] [20] http://de.wikipedia.org/wiki/Direct_Digital_Synthesis

[21] http://de.wikipedia.org/wiki/Pseudo-random_bit_stream
[22][25][31][36][37][38][40][43][67] Erich Pehl: Digitale und analoge Nachrichten-übertragung: Signale, Codierung, Modulation, Anwendung. 2., überarbeitete und erweiterte Auflage. Heidelberg: Hüthig, 2001. ISBN 3-7785-2801-7
[23][29] [26][35] [41] [42] [65][68] [69]EN 300 429 V1.2.1 (1998-04): Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for cable systems.
[24] http://de.wikipedia.org/wiki/Kanalkodierung
[27] Wolfgang Eustach, Digitale Fernsehtechnik: Eine Einführung, Books on Demand GmbH, Norderstedt,2009. ISBN: 978-8391-7329-9
[28] http://www.une.edu.ve/~jduran/Dvb.htm
[30] http://de.wikipedia.org/wiki/Reed-Solomon-Code
[32] http://www.une.edu.ve/~jduran/Dvb.htm
[33] Codierungstheorie WS0506
[34] ftp://ftp.radionetworkprocessor.com/pub/reed-solomon/chap6.pdf
[39] Vorlesung: digitale Nachrichtenübertragung. Prof. Schwauzenau
[44] [46][47] http://www.mikrocontroller.net/articles/FPGA
[45]http://baike.baidu.com/view/51371.htm
[48] [50] [51] [62] [63] [70]PDF, DE2-115 User Manual
[49]http://www.altera.com/devices/fpga/cyclone-iv/cyiv-index.jsp
[52][53][57] [58] [59] [60] [61] PDF,AD9857 Data Sheet Rev C, 05/2004
[54] http://de.wikipedia.org/wiki/Cascaded-Integrator-Comb_Filter
[55] Mezer-Bäse Uwe: schnelle digitale Signalverarbeitung : Algorithmen, Architek-turen, Anwendungen, Berlin; Heidelberg: Springer, 2000. ISBN 3-540-67662-7
[56] http://de.wikipedia.org/wiki/Sample-and-Hold-Schaltung
[64] http://de.wikipedia.org/wiki/Verilog
[66] PDF, Kanalcodierung
[71] PDF, KWS AMA 300 & AMA 301

Anhang

A.1 Korrespondenz zwischen den Feld-Elementen und binären Daten

Wenn a als die Wurzel von $p(x) = 0$ definiert ist, dann kann $a^8 = a^4 + a^3 + a^2 + 1$

nach $p(x) = x^8 + x^4 + x^3 + x^2 + 1$ bekommen werden. Die Korrespondenz zwischen den Körper-Elementen und binären Daten ist in Tabelle A.1 aufgelistet.

Binär-zahl	GF(2^8)	Binär-zahl	GF(2^8)	Binär-zahl	GF(2^8)	Binär-zahl	GF(2^8)	Binär-zahl	GF(2^8)
00H	0	01H	a^0	02H	a^1	03H	a^{25}	04H	a^2
05H	a^{50}	06H	a^{26}	07H	a^{198}	08H	a^3	09H	a^{223}
0AH	a^{51}	0BH	a^{238}	0CH	a^{27}	0DH	a^{104}	0EH	a^{199}
0FH	a^{75}	10H	a^4	11H	a^{100}	12H	a^{224}	13H	a^{14}
14H	a^{52}	15H	a^{141}	16H	a^{239}	17H	a^{129}	18H	a^{28}
19H	a^{193}	1AH	a^{105}	1BH	a^{248}	1CH	a^{200}	1DH	a^8
1EH	a^{76}	1FH	a^{113}	20H	a^5	21H	a^{138}	22H	a^{101}
23H	a^{47}	24H	a^{225}	25H	a^{36}	26H	a^{15}	27H	a^{33}
28H	a^{53}	29H	a^{147}	2AH	a^{142}	2BH	a^{218}	2CH	a^{240}
2DH	a^{18}	2EH	a^{130}	2FH	a^{69}	30H	a^{29}	31H	a^{181}
32H	a^{194}	33H	a^{125}	34H	a^{106}	35H	a^{39}	36H	a^{249}
37H	a^{185}	38H	a^{201}	39H	a^{154}	3AH	a^9	3BH	a^{120}
3CH	a^{77}	3DH	a^{228}	3EH	a^{114}	3FH	a^{166}	40H	a^6
41H	a^{191}	42H	a^{139}	43H	a^{98}	44H	a^{102}	45H	a^{221}
46H	a^{48}	47H	a^{253}	48H	a^{26}	49H	a^{152}	4AH	a^{37}
4BH	a^{179}	4CH	a^{16}	4DH	a^{145}	4EH	a^{34}	4FH	a^{136}
50H	a^{54}	51H	a^{208}	52H	a^{148}	53H	a^{206}	54H	a^{143}
55H	a^{150}	56H	a^{219}	57H	a^{189}	58H	a^{241}	59H	a^{210}
5AH	a^{19}	5BH	a^{92}	5CH	a^{131}	5DH	a^{56}	5EH	a^{70}
5FH	a^{64}	60H	a^{30}	61H	a^{66}	62H	a^{182}	63H	a^{163}
64H	a^{195}	65H	a^{72}	66H	a^{126}	67H	a^{110}	68H	a^{107}
69H	a^{58}	6AH	a^{40}	6BH	a^{84}	6CH	a^{250}	6DH	a^{133}
6EH	a^{186}	6FH	a^{61}	70H	a^{202}	71H	a^{94}	72H	a^{155}
73H	a^{159}	74H	a^{10}	75H	a^{21}	76H	a^{121}	77H	a^{43}
78H	a^{78}	79H	a^{212}	7AH	a^{229}	7BH	a^{172}	7CH	a^{115}
7DH	a^{243}	7EH	a^{167}	7FH	a^{87}	80H	a^7	81H	a^{112}
82H	a^{192}	83H	a^{247}	84H	a^{140}	85H	a^{128}	86H	a^{99}
87H	a^{13}	88H	a^{103}	89H	a^{74}	8AH	a^{222}	8BH	a^{237}
8CH	a^{49}	8DH	a^{197}	8EH	a^{254}	8FH	a^{24}	90H	a^{227}
91H	a^{165}	92H	a^{153}	93H	a^{119}	94H	a^{38}	95H	a^{184}

96H	a^{180}	97H	a^{124}	98H	a^{17}	99H	a^{68}	9AH	a^{146}
9BH	a^{217}	9CH	a^{35}	9DH	a^{32}	9EH	a^{137}	9FH	a^{46}
A0H	a^{55}	A1H	a^{63}	A2H	a^{209}	A3H	a^{91}	A4H	a^{149}
A5H	a^{188}	A6H	a^{307}	A7H	a^{205}	A8H	a^{144}	A9H	a^{135}
AAH	a^{151}	ABH	a^{178}	ACH	a^{220}	ADH	a^{252}	AEH	a^{190}
AFH	a^{97}	B0H	a^{244}	B1H	a^{86}	B2H	a^{211}	B3H	a^{171}
B4H	a^{20}	B5H	a^{42}	B6H	a^{93}	B6H	a^{158}	B8H	a^{132}
B9H	a^{60}	BAH	a^{57}	BBH	a^{83}	BCH	a^{71}	BDH	a^{109}
BEH	a^{65}	BFH	a^{162}	C0H	a^{31}	C1H	a^{45}	C2H	a^{67}
C3H	a^{216}	C4H	a^{183}	C5H	a^{123}	C6H	a^{164}	C7H	a^{118}
C8H	a^{196}	C9H	a^{23}	CAH	a^{73}	CBH	a^{236}	CCH	a^{127}
CDH	a^{12}	CEH	a^{111}	CFH	a^{246}	D0H	a^{108}	D1H	a^{161}
D2H	a^{59}	D3H	a^{82}	D4H	a^{41}	D5H	a^{157}	D6H	a^{85}
D7H	a^{170}	D8H	a^{251}	D9H	a^{96}	DAH	a^{134}	DBH	a^{177}
DCH	a^{187}	DDH	a^{204}	DEH	a^{62}	DFH	a^{90}	E0H	a^{203}
E1H	a^{89}	E2H	a^{95}	E3H	a^{176}	E4H	a^{156}	E5H	a^{169}
E6H	a^{160}	E7H	a^{81}	E8H	a^{11}	E9H	a^{245}	EAH	a^{22}
EBH	a^{235}	ECH	a^{122}	EDH	a^{117}	EEH	a^{44}	EFH	a^{215}
F0H	a^{79}	F1H	a^{174}	F2H	a^{213}	F3H	a^{233}	F4H	a^{230}
F5H	a^{231}	F6H	a^{173}	F7H	a^{232}	F8H	a^{116}	F9H	a^{214}
FAH	a^{244}	FBH	a^{234}	FCH	a^{168}	FDH	a^{80}	FEH	a^{214}
FFH	a^{175}								

Tabelle A.1