LEHRSTUHL UND LABORATORIUM FÜR TECHNISCHE ELEKTRONIK

Vorstand: Prof. Dr.-Ing. D. Seitzer

Friedrich-Alexander-Universität Erlangen-Nürnberg

Definition und Implementierung der Kanalkodierung und Fehlersicherung für ein Funknetz

Diplomarbeit im Fach Elektrotechnik

vorgelegt von

cand. ing. Herbert Thoma

geb. am 25. 03. 1970 in Würzburg

Betreuer:	**Dipl. Ing. Rainer Perthold** **Dipl. Ing. Hans Hauer** Fraunhofer Institut für Integrierte Schaltungen
Beginn der Arbeit:	01. März 1996
Abgabe der Arbeit:	30. August 1996

'All right,' said Deep Thought. 'The Answer to the Great Question ...'
'Of Life, the Universe and Everything ...'
'Is ...'

'**Fourty-two**'

Douglas Adams, The Hitch Hiker's Guide to the Galaxy

Diese Diplomarbeit wurde mit dem Schriftsatzsystem TEX von Donald E. Knuth unter Verwendung des Makropakets LATEX von Leslie Lamport und des Documentstyles `script` von Frank Neukam erstellt.

Zusammenfassung

Im Rahmen dieser Diplomarbeit wurde ein sogenannter Burst Mode Controller entwickelt, der die Kanalcodierung und Fehlersicherung in einem Funknetzwerk durchführt. Der Burst Mode Controller enthält folgende Funktionseinheiten:

- Mikrocontrollerschnittstelle
- Fehlerkorrekturcodierung durch einen (31,16)BCH Code
- Fehlererkennung durch einen CRC Code
- Bit- und Rahmensynchronisation

Der Burst Mode Controller wurde in der Hardwarebeschreibungssprache Verilog HDL entwickelt und in ein FPGA implementiert.

Abstract

In this diploma thesis a Burst Mode Controller which performs channel coding in a radio frequency network is presented. The Burst Mode Controller comprises the following units:

- Microcontrollerinterface
- (31,16)BCH forward error correction coding
- CRC error detection
- Bit and frame synchronization

The Burst Mode Controller was developed in Verilog HDL and implemented into a FPGA.

Erklärung

Ich versichere, daß ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und daß die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Erlangen, den 30. August 1996 _____

Aufgabenstellung

Definition und Implementierung der Kanalkodierung und Fehlersicherung für ein Funknetz

Für Heimanwendungen wird ein Funknetz auf 433 MHz entwickelt. Für die sichere Übertragung der Daten über die Funkstrecke sind sowohl eine angemessene Kanalkodierung als auch Fehlersicherung erforderlich. Darüber hinaus müssen im Empfänger die demodulierten Analogsignale möglichst effizient wieder in Digitalsignale umgewandelt werden (OSI Schicht 1).

Hierfür sind zunächst entsprechende Verfahren und Algorithmen zu untersuchen und zu entwickeln. Anschließend sollen diese in einem FPGA unter Einsatz entsprechender Simulationswerkzeuge realisiert werden. Die Tauglichkeit der gewählten Verfahren wird anschließend an einer realen Funkstrecke demonstriert.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

ARQ	Automatic Repeat Request
ASIC	Application Specific Integrated Circuit
BCH	Bose Chaudhuri Hocquenghem
BMA	Berlekamp-Massey-Algorithmus
BMC	Burst Mode Controller
CCITT	Comité Consultatif International de Télegraphique et Téléphonique
CRC	Cyclic Redundancy Check
CSMA	Carrier Sense Multiple Access
FEC	Forward Error Control
FM	Frequenzmodulation
FPGA	Field Programmable Gate Array
HDL	Hardware Description Language
HF	Hochfrequenz
GFSK	Gaussian Frequency Shift Keying
IIS	Institut für Integrierte Schaltungen
ISM	Industrial Scientific Medical
LON	Local Operating Network
Mbps	Mega bits per second
PLL	Phase Locked Loop
PROM	Programmable Read Only Memory

PWM	Pulse Width Modulation
RAM	Random Access Memory
ROM	Read Only Memory
RX	Receiver
TX	Transmitter
kbps	kilo bits per second

Formelzeichen

E_b	Energie pro Bit
GF(m)	Galoisfeld der Mächtigkeit m
N_0	Rauschleistungsdichte
$S(x)$	Syndrompolynom
$c(x)$	Codewort
$e(x)$	Fehlermuster
$g(x)$	Generatorpolynom
$\nu(x)$	Fehlerstellenpolynom im modifizierten BMA
$p(x)$	Prüfwort
$r(x)$	empfangenes Wort
$\sigma(x)$	Fehlerstellenpolynom im BMA
$u(x)$	Infowort
$\lceil x \rceil$	nächstgrößere ganze Zahl als x

1. Einleitung

Am Fraunhofer Institut für Integrierte Schaltungen wird ein Funknetz für das ISM-Frequenzband auf 433 MHz entwickelt. Die Anwendungsmöglichkeiten eines solchen Funknetzes in der industriellen Automatisierungstechnik, der Gebäudeleittechnik, der Datenerfassung oder bei der Objektüberwachung sind sehr vielfältig.

Das Funkmedium unterscheidet sich von drahtgebundenen Medien in wesentlichen Parametern. Dies erfordert eine aufwendigere Leitungscodierung als bei herkömmlichen Feldbussystemen. Um die Anwendung des Funknetzes zu erleichtern, werden die Codierungs- und HF-Funktionen in einem Transceiver-Modul gekapselt, das dem Anwender eine digitale Datenschnittstelle zur Verfügung stellt. Abbildung 1.1 zeigt ein Blockschaltbild dieses Moduls.

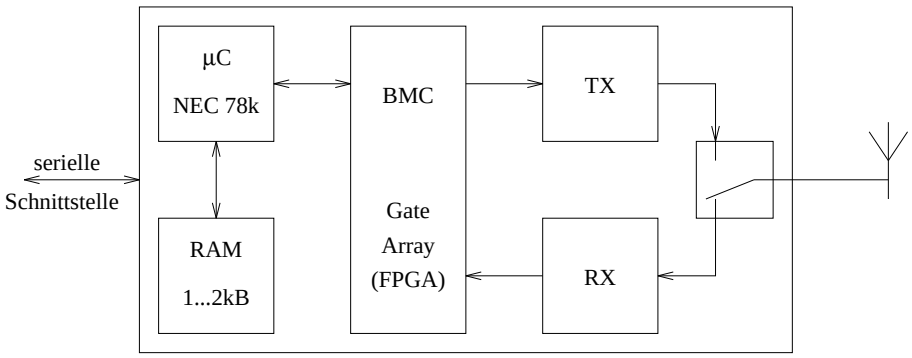


Abbildung 1.1: Blockschaltbild Transceiver-Modul

Neben dem Sender (TX), dem Empfänger (RX) und einem Mikrocontroller (μC) ist der sogenannte Burst Mode Controller (BMC) ein wesentlicher Bestandteil des Transceiver-Moduls. In dem in einem FPGA zu implementierenden BMC sind eine Fehlersicherung durch einen CRC-Code, eine fehlerkorrigierende Kanalcodierung und Synchronisationsfunktionen realisiert.

Der Zugriff auf den Funkkanal wird durch ein CSMA-Verfahren mit Kollisionsvermeidung gesteuert. Optional kann eine Reservierung des Kanals vor der Übertragung erfolgen. Die Datenübertragung erfolgt paketweise mit variabler Paketgröße. Die maximale Paketgröße beträgt 255 Bytes Nutzdaten plus einen kurzen Vorspann für Verwaltungsinformationen.

Als Modulationsart ist Frequenzumtastung mit gaußförmiger Umschaltcharakteristik (GFSK) vorgesehen. Die HF-Datenrate beträgt 20 kbps, optional 40 kbps.

1.1 Bisheriger Stand und Aufgabenstellung

Für die HF-Baugruppen TX und RX ist am IIS ein FM-Transceiver vorhanden. Erste Datenübertragungen wurden mit einem Mikrocontroller und softwarerealisierter Kanalcodierung durchgeführt. Auf Grund der beschränkten Rechenleistung des Mikrocontrollers ist diese Lösung nur für einfache Codes und niedrige Bitraten geeignet.

Die Aufgabenstellung dieser Diplomarbeit konzentriert sich deshalb auf den Burst Mode Controller. Die Ziele dieser Arbeit sind:

- Die Untersuchung verschiedener Kanalcodierungsverfahren und die Auswahl eines geeigneten Algorithmus.
- Die Entwicklung des BMCs in einer Hardwarebeschreibungssprache und die Implementierung in ein FPGA.
- Die Überprüfung des BMCs an einer realen Funkstrecke.

Die Entwicklung von effizienten Kanalzuteilungs- und ARQ-Algorithmen als Mikrocontrollerprogramm ist Gegenstand einer weiteren Diplomarbeit.

1.2 Gliederung der Arbeit

Nachdem in diesem ersten Kapitel zunächst der bisherige Stand und die Aufgabenstellung dargelegt wurden, zeigt das nächste Kapitel verschiedene Möglichkeiten der Fehlerschutzcodierung. Die Gründe für die Auswahl eines speziellen Verfahrens und die Wirkungsweise des gewählten Algorithmus werden dargestellt.

Kapitel 3 zeigt Einzelheiten der Realisierung des BMCs wie zum Beispiel die Hardwareimplementierung des gewählten Codierungsalgorithmus und eine Mi-

krocontrollerschaltung zum Test des BMCs. Ergebnisse und Erfahrungen mit dem BMC sind schließlich in Kapitel 4 dargestellt.

In Kapitel 5 werden die Ergebnisse noch einmal zusammengefaßt und ein Ausblick auf die weitere Entwicklung des Burst Mode Controllers und des Funknetzes wird gegeben.

2. Fehlerschutzcodierung

Die Kanalcodierung stellt Methoden zur Verfügung, mit denen Informationen von einem Sender zu einem Empfänger mit möglichst wenig Fehlern übertragen werden können. Den eigentlichen Informationen wird senderseitig kontrolliert Redundanz hinzugefügt, so daß Übertragungsfehler vom Empfänger erkannt und/oder korrigiert werden können. Demnach sind zwei Verfahren zu unterscheiden:

FEC-Verfahren (Forward Error Control): Die hinzugefügte Redundanz dient empfangsseitig zur *Korrektur* der Übertragungsfehler.

ARQ-Verfahren (Automatic Repeat Request): Hier erfolgt lediglich eine *Erkennung* der Fehler. Wenn ein Übertragungsfehler erkannt wird, wird eine Wiederholung der Übertragung angefordert.

Der Vorteil von ARQ besteht darin, daß zur Fehlererkennung wesentlich weniger Redundanz erforderlich ist als zur Fehlerkorrektur. Als Nachteil ist zu werten, daß sich bei schlechter Kanalqualität der Durchsatz in Folge der Wiederholungen verringert.

In dem in dieser Diplomarbeit entwickelten Burst Mode Controller kommt eine Kombination beider Verfahren zum Einsatz. Mit einem ersten Code können die meisten Übertragungsfehler korrigiert werden. Ein weiterer Code dient zur Erkennung noch verbleibender nicht korrigierbarer Fehler. Durch diese Kombination wird einerseits ein hoher Durchsatz und andererseits eine sehr geringe Fehlerrate erreicht.

Zur Fehlererkennung wird ein CRC-Code nach CCITT eingesetzt. Dieser Code kann 99,9985% aller möglichen Übertragungsfehler erkennen. CRC-Codes sind zur Fehlererkennung besonders gut geeignet, weit verbreitet und einfach zu implementieren, [7], [17] Seite 121ff.

Die Wahl des Fehlerkorrekturcodes hat großen Einfluß auf die Leistung und die Komplexität des Gesamtsystems. Im nächsten Abschnitt sind deshalb verschiedene FEC-Codes und Gründe für die Wahl eines Codes beschrieben. Der übernächste Abschnitt erläutert den gewählten Code im Detail.

2.1 Auswahl eines geeigneten Fehlerkorrekturcodes

Seit den grundlegenden Arbeiten von Shannon sind eine Vielzahl von fehlerkorrigierenden Codes entdeckt und erforscht worden. Die wichtigsten Kriterien für die Wahl eines speziellen Codes sind die Eigenschaften des Übertragungskanals und die angestrebte Restbitfehlerwahrscheinlichkeit. Aber auch andere Randbedingungen wie zum Beispiel die Komplexität der Implementierung oder die zur Verfügung stehende Bandbreite müssen beachtet werden.

Übertragungskanäle lassen sich grob in zwei Gruppen einteilen: In Kanäle mit statistisch unabhänigen Einzelfehlern und solche mit bündelweise auftretenden Fehlern, sogenannten Burstfehlern. In Funkkanälen mit stationären Sendern und Empfängern überwiegen Einzelfehler, während bei Mobilanwendungen Burstfehler häufiger auftreten. Da für das hier betrachtete Funknetz sowohl stationäre als auch Mobilanwendungen in Frage kommen, ist ein günstiger Kompromiß für beide Fehlerarten zu finden.

Die tolerierbare Restbitfehlerwahrscheinlichkeit hängt von den zu übertragenden Daten ab. Unkomprimierte digitalisierte Sprache ist auch noch bei einer Bitfehlerrate von 10^{-3} verständlich, bei einem Computerprogramm dagegen kann schon ein einziges fehlerhaftes Bit zu Fehlfunktionen führen.

Nachdem nun Kriterien für die Auswahl eines Codes aufgezeigt wurden, sollen jetzt die Eigenschaften verschiedener Codes dargestellt werden. Zwei wichtige Kenngrößen von FEC-Codes sind die Coderate und der Codegewinn. Die Coderate gibt den relativen Anteil von Information im Code an. Der Codegewinn ist ein Maß dafür, um wieviel die Sendeleistung gegenüber uncodierter Übertragung verringert werden kann um die gleiche Bitfehlerwahrscheinlichkeit zu erhalten. Die Angabe eines Codegewinns muß immer auf eine bestimmte Bitfehlerwahrscheinlichkeit bezogen sein.

Bei den FEC-Codes unterscheidet man zwei Prinzipien: Faltungscodes und Blockcodes.

Faltungscodes sind besonders gut für Kanäle mit zufälligen Einzelfehlern geeignet. Der bekannteste Algorithmus zur Decodierung von Faltungscodes ist der Viterbi-Algorithmus, [15], [16]. Dieser Algorithmus bietet eine einfache Möglichkeit Zuverlässigkeitsinformationen des Demodulators in die Decodierung einzubeziehen, man spricht dann von Soft Decision. Mit Soft Decision läßt sich der Codegewinn beträchtlich erhöhen. Encoder für Faltungscodes sind sehr einfach realisierbar. Der Nachteil von Faltungscodes ist die Komplexität des Viterbi-Algoritmus, insbesondere benötigt er relativ große Mengen an Speicher. Wegen der hohen Hardwareanforderungen im Decoder wurden Faltungscodes im BMC

nicht eingesetzt.

Das Prinzip der Blockcodierung zeigt Abbildung 2.1. Die zu übertragenden Daten werden in Blöcke der Länge k unterteilt. Diese Blöcke $\boldsymbol{u} = (u_0, \cdots, u_{k-1})$ werden als Infoworte bezeichnet. Der Encoder ordnet den Infoworten umkehrbar eindeutig Codeworte $\boldsymbol{c} = (c_0, \cdots, c_{n-1})$ der Länge n zu. Es gilt stets $k < n$. Am Ausgang des Kanals entsteht das Empfangswort $\boldsymbol{r} = (r_0, \cdots, r_{n-1})$, der Decoder gewinnt daraus eine Schätzung für das Infowort $\hat{\boldsymbol{u}} = (\hat{u}_0, \cdots, \hat{u}_{k-1})$.

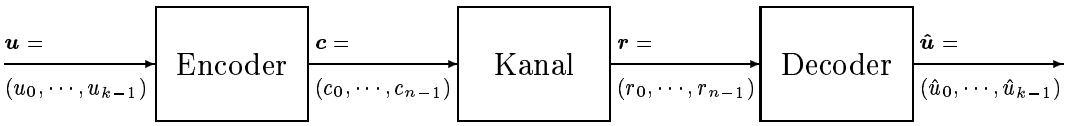


Abbildung 2.1: Prinzip der Blockcodierung

Wie man sieht hat ein (n, k) Blockcode eine Coderate $R = \frac{k}{n} < 1$. Der Codegewinn hängt davon ab, wie die Infoworte den Codeworten zugeordnet werden. Allgemein gilt, daß sich die Codeworte möglichst stark unterscheiden müssen. Wenn sich bei einem Code je zwei Codeworte in mindestens $2t + 1$ Stellen unterscheiden, so kann dieser Code t Fehler korrigieren. Der Codegewinn steigt mit abnehmender Coderate und mit zunehmender Blocklänge.

Bekannte Blockcodes sind zum Beispiel: Hamming Codes, BCH Codes und Reed-Solomon Codes. Reed-Solomon Codes sind besonders für Kanäle mit Burstfehlern geeignet.

Code	Rate	Gewinn	Eignung für Burstfehler	Komplexität
uncodiert	1	0,0 dB	-	-
(7,4)Hamming	0,57	2,3 dB	nein	gering
(31,21)BCH	0,68	3,0 dB	nein	moderat
(31,16)BCH	0,52	3,9 dB	bedingt	moderat
(63,31)RS	0,49	4,9 dB	gut	hoch
$K = 7$, $R = \frac{1}{2}$ Faltungscode	0,50	4,5 dB	schlecht	sehr hoch
Faltungscode, Soft Decision	0,50	7,4 dB	schlecht	sehr hoch

Tabelle 2.1: Vergleich verschiedener Codes bei einer Bitfehlerrate von 10^{-5}

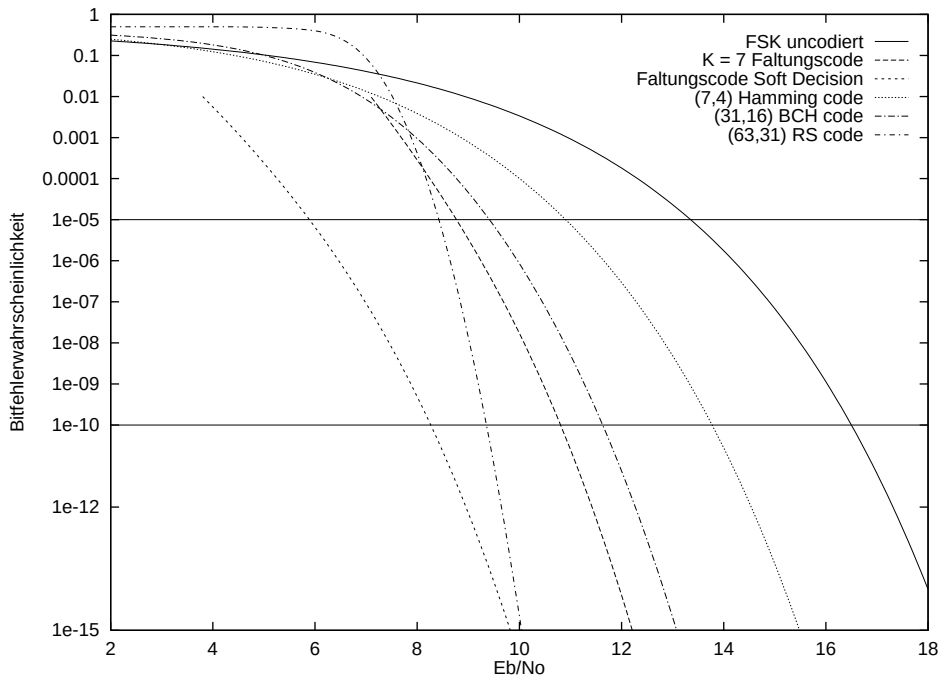


Abbildung 2.2: Vergleich verschiedener Codes

In Tabelle 2.1 und Abbildung 2.2 sind verschiedene Codes einander gegenübergestellt. Die Angaben zum Codegewinn beziehen sich auf eine Bitfehlerwahrscheinlichkeit von 10^{-5}. Weiterhin ist zu beachten, daß die Codes bei gleicher Kanaldatenrate verglichen werden, das bedeuted, daß Codes mit gringerer Rate auch geringeren Durchsatz haben. Diese Art der Darstellung wurde gewählt, da bei der Anwendung des BMCs im Funknetz die HF-Datenrate festgelegt ist.

Für den Einsatz im BMC wurde der (31,16)BCH Code ausgewählt. Dieser Code bietet folgende Vorteile:

- Geeignet für zufällige Fehler und kurze Burstfehler
- Guter Codegewinn
- Mäßige Komplexität der Implementierung

2.2 BCH Codes

Die nach ihren Entdeckern Bose, Chaudhuri und Hocquenguem benannten BCH Codes bilden eine große Klasse zyklischer Blockcodes. Unter anderem kann man

auch die Reed-Solomon Codes als Unterklasse der BCH Codes auffassen. Im folgenden werden nur binäre BCH Codes mit sogenannter primitiver Blocklänge behandelt.

Ausführlichere Abhandlungen zu BCH Codes finden sich in allen Grundlagenwerken zur Codierungstheorie, z. B. in [1], [3], [8], [9] oder [17]. Die hier gegebene kurze Einführung lehnt sich an die Darstellungen in [5] und [17] an. Einige Hinweise zur Arithmetik über Galoisfeldern werden in Anhang A gegeben.

2.2.1 Definition der BCH Codes

Die Codeworte eines binären BCH Codes der Blocklänge n bilden eine Teilmenge der Menge der binären n-Tupel. Für primitive BCH Codes gilt: $n = 2^m - 1$. Eine übliche Art der Darstellung von n-Tupeln ist die Polynomdarstellung, bei der die Elemente des n-Tupels als Koeffizienten eines Polynoms aufgefaßt werden. Wenn zum Beispiel

$$(a_0, a_1, a_2, \cdots, a_{n-1}), \qquad a_i \in \mathrm{GF}(2)$$

ein binäres n-Tupel ist, so ist

$$f(x) = \sum_{i=0}^{n-1} a_i x^i, \qquad a_i \in \mathrm{GF}(2)$$

das zughörige Polynom.

Ein BCH Code ist definiert als die Menge aller $f(x)$, die Vielfache eines Generatorpolynoms $g(x)$ sind, wobei $g(x)$ ein binäres Polynom vom Grad kleiner als $n-1$ ist. Anders ausgedrückt ist der Code die Menge aller Polynome $f(x)$, die als Nullstellen auch die Nullstellen von $g(x)$ besitzen.

Das Generatorpolynom eines BCH Codes, der t Fehler korrigieren kann, ist das Polynom mit dem kleinsten Grad über $\mathrm{GF}(2^m)$, das $\alpha, \alpha^2, \cdots, \alpha^{2t}$ als Nullstellen hat.

2.2.2 Encoder

Die naheliegenste Methode ein Codewort $c(x)$ zu einem Infowort $u(x)$ zu berechnen ist die Multiplikation mit dem Generatorpolynom $g(x)$:

$$c(x) = u(x)g(x)$$

Diese Methode hat allerdings den Nachteil, daß sie nicht-systematische Codeworte erzeugt.

Unter einem systematischen Code versteht man einen Code, bei dem das Infowort explizit Bestandteil des Codeworts ist:

$$\boldsymbol{c} = (\underbrace{p_0, \cdots, p_{n-k-1}}_{n-k \text{ Prüfstellen}}, \underbrace{u_0, \cdots, u_{k-1}}_{k \text{ Infostellen}})$$

Eine Codierung in systematischer Form wird in drei Schritten ausgeführt:

1. Multiplikation des Infoworts $u(x)$ mit x^{n-k}.
2. Berechnung des Divisionsrests $p(x)$ der Polynomdivision von $x^{n-k}u(x)$ durch das Generatorpolynom $g(x)$.
3. Kombination von $p(x)$ und $x^{n-k}u(x)$. Das Codepolynom lautet dann:

$$c(x) = p(x) + x^{n-k}u(x)$$

2.2.3 Decoder

Bei einem Empfangswort $r(x)$ besteht die Möglichkeit, daß das gesendete Codewort $c(x)$ durch Übertragungsfehler $e(x)$ verfälscht ist.

$$r(x) = c(x) + e(x)$$

Die Korrektur dieser Fehler geschieht in folgenden Schritten:

1. Das Syndrom $S(x)$ wird berechnet.
2. Aus dem Syndrom $S(x)$ wird das Fehlerstellenpolynom $\sigma(x)$ berechnet.
3. Die inversen Nullstellen von $\sigma(x)$ sind die Fehlerorte in $r(x)$.

Das Syndrom

Sei $r(x) = c(x) + e(x)$, dann gilt:

$$r(\alpha^j) = c(\alpha^j) + e(\alpha^j) = e(\alpha^j), \qquad j = 1, 2, \cdots, 2t$$

$$S_j = r(\alpha^j), \qquad j = 1, 2, \cdots, 2t \quad S_j \in \mathrm{GF}(2^m)$$

wird als j-te Syndromkomponente bezeichnet. Wie man sieht, sind die Syndromkomponenten nur vom Fehlermuster und nicht vom gesendeten Codewort abhängig.

Der Berlekamp-Massey-Algorithmus

Der weitaus aufwendigste Schritt bei der Decodierung von BCH Codes ist die Berechnung des Fehlerstellenpolynoms $\sigma(x)$ aus den Syndromkomponenten S_j. Der beste bekannte Algorithmus zur Lösung dieses Problems ist der Berlekamp-Massey-Algorithmus (BMA) [1], [11]. Der BMA berechnet $\sigma(x)$ rekursiv wie folgt:

Definiere:

$$1 + S = 1 + S_1 x + S_2 x^2 + \cdots + S_{2t} x^{2t}$$
$$\sigma^{(0)} = 1, \qquad \tau^{(0)} = 1$$

Dann führe folgende rekursive Berechnungen durch:

Wenn S_{2k+1} nicht bekannt ist, beende die Rekursion. Sonst setze $\Delta^{(2k)}$ gleich dem Koeffizienten von x^{2k+1} in dem Produkt $(1+S)\sigma^{(2k)}$. Berechne:

$$\sigma^{(2k+2)} = \sigma^{(2k)} + \Delta^{(2k)} \tau^{(2k)} x \tag{2.1}$$

$$\tau^{(2k+2)} = \begin{cases} x^2 \tau^{(2k)}, & \text{wenn } \Delta^{(2k)} = 0 \text{ oder Grad}\,\sigma^{(2k)} > k \\ \dfrac{x\sigma^{(2k)}}{\Delta^{(2k)}}, & \text{wenn } \Delta^{(2k)} \neq 0 \text{ und Grad}\,\sigma^{(2k)} \leq k \end{cases} \tag{2.2}$$

Das Polynom $\sigma^{(2t)}$ ist das gesuchte Fehlerstellenpolynom. Die multiplikativen Inversen der Nullstellen dieses Polynoms bestimmen die Fehlerorte in $r(x)$.

2.2.4 (31,16)BCH Code

Die Ausführungen über BCH Codes sollen am Beispiel des (31,16)BCH Codes verdeutlicht werden. Die Blocklänge dieses Codes beträgt 31 Bit, davon sind 16 Bit Infobits und 15 Prüfbits. Dieser Code ist in der Lage bis zu drei Fehler je Codewort zu korrigieren. Das Generatorpolynom lautet:

$$g_{(31,16)}(x) = 1 + x + x^2 + x^3 + x^5 + x^7 + x^8 + x^9 + x^{10} + x^{11} + x^{15}$$

Es soll das Infowort

$$u(x) = x + x^4$$

übertragen werden. Dem entspricht die Bitfolge

$$\boldsymbol{u} = (0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0).$$

Die Multiplikation von $u(x)$ mit $x^{(n-k)}$ ergibt:

$$u(x)x^{15} = x^{16} + x^{19}.$$

Der Divisionsrest von $(u(x)x^{15})/g(x)$ ist:

$$p(x) = 1 + x^9 + x^{11} + x^{13} + x^{14}$$

Für das Codewort ergibt sich also:

$$c(x) = 1 + x^9 + x^{11} + x^{13} + x^{14} + x^{16} + x^{19}$$

bzw.:

$$\boldsymbol{c} = (1,0,0,0,0,0,0,0,0,1,0,1,0,1,1,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0)$$

Wird nun

$$r(x) = 1 + x^9 + x^{11} + x^{14}$$

empfangen, so ergibt sich als Syndrom

$$S(x) = \alpha x + \alpha x^2 + \alpha^{29}x^3 + \alpha x^4 + \alpha^{23}x^5 + \alpha^{27}x^6.$$

Der Berlekamp-Massey-Algorithmus läuft wie folgt ab:

k	$\sigma^{(2k)}$	$\tau^{(2k)}$	$\Delta^{(2k)}$
0	1	1	1
1	$1+x$	x	α^3
2	$1+x+\alpha^3x^2$	$\alpha^{28}x+\alpha^{28}x^2$	α^{20}
3	$1+x+\alpha^{16}x^2+\alpha^{17}x^3$	-	-

$$\begin{aligned}\sigma^{(6)} &= 1 + x + \alpha^{16}x^2 + \alpha^{17}x^3 \\ &= (1+\alpha^{13}x)(1+\alpha^{16}x)(1+\alpha^{19}x)\end{aligned}$$

ist das gesuchte Fehlerstellenpolynom. Es besitzt die Nullstellen α^{18}, α^{15} und α^{12}, deren Inverse sind α^{13}, α^{16} und α^{19}. Das Fehlerpolynom ist demnach

$$e(x) = x^{13} + x^{16} + x^{19}$$

und das geschätzte Codewort ist

$$\hat{c}(x) = r(x) - e(x) = 1 + x^9 + x^{11} + x^{13} + x^{14} + x^{16} + x^{19}.$$

In der Darstellung als 31-Tupel:

$$\begin{aligned}\boldsymbol{r} &= (1,0,0,0,0,0,0,0,0,1,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0) \\ \boldsymbol{e} &= (0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0) \\ \hat{\boldsymbol{c}} &= (1,0,0,0,0,0,0,0,0,1,0,1,0,1,1,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0) \\ \hat{\boldsymbol{u}} &= (0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)\end{aligned}$$

3. Realisierung

Bevor auf die Einzelheiten der Realisierung des BMCs eingegangen wird, soll hier das im Funknetz verwendete Paketformat vorgestellt werden. Abbildung 3.1 zeigt den Aufbau eines Pakets.

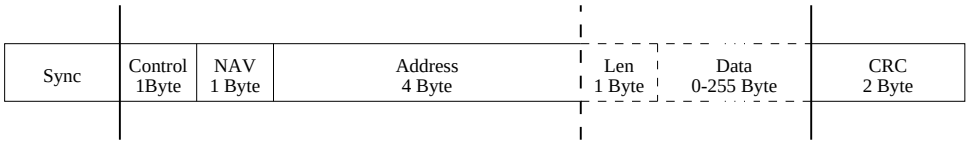


Abbildung 3.1: Paketformat

Ein Paket besteht aus drei Teilen: Am Beginn des Pakets wird eine Synchronisationssequenz gesendet, dann folgen die Nutzdaten, am Schluß der CRC Wert. Die Nutzdaten und der CRC Wert sind mit dem (31,16)BCH Code gegen Übertragungsfehler geschützt. Je zwei Byte werden für ein Codewort zusammengefaßt, soll eine ungerade Anzahl Bytes übertragen werden, so wird ein zusätzliches Nullbyte eingefügt.

Die Nutzdaten bestehen aus einem sechs Byte langen Vorspann für Adressen und Netzwerkfunktionen, einem Byte, das die Anzahl der Datenbytes angibt, und den eigentlichen Daten. Das Längenbyte und die Daten können auch entfallen. Das höchstwertige Bit des ersten Bytes zeigt an, ob Daten folgen oder nicht.

3.1 Burst Mode Controller

Für den Burst Mode Controller wurde zunächst eine Beschreibung in der Hardwarebeschreibungssprache Verilog HDL auf Register Transfer Ebene erstellt und durch Simulation getestet. Diese Beschreibung wurde anschließend unter Verwendung von Logiksynthesewerkzeugen in ein FPGA vom Typ Xilinx XC 4013 implementiert.

In diesem Abschnitt wird detailiert die Funktionsweise des BMCs vorgestellt. Der Verilog HDL Quelltext kann Anhang B entnommen werden. Die Verilog Beschreibung ist hierarchisch in Module aufgeteilt. Im folgenden werden `Signal`namen aus dem Verilog Quelltext in Schreibmaschinenschrift gesetzt, Modulnamen in serifenloser Schrift.

Der BMC kennt 3 Betriebszustände: Ruhezustand, Sendebetrieb und Empfangsbetrieb. Ein Übergang vom Ruhezustand in den Sendebetrieb wird vom steuernden Mikrocontroller ausgelöst, ein Übergang in den Empfangsbetrieb erfolgt, wenn am Empfängereingang ein Paketanfang erkannt wird. Nachdem ein komplettes Paket gesendet bzw. empfangen wurde, kehrt der BMC in den Ruhezustand zurück. Ein Abbruch des Empfangsbetriebs und direkter Wechsel zum Sendebetrieb ist möglich, der Sendebetrieb kann durch den Empfänger jedoch nicht unterbrochen werden.

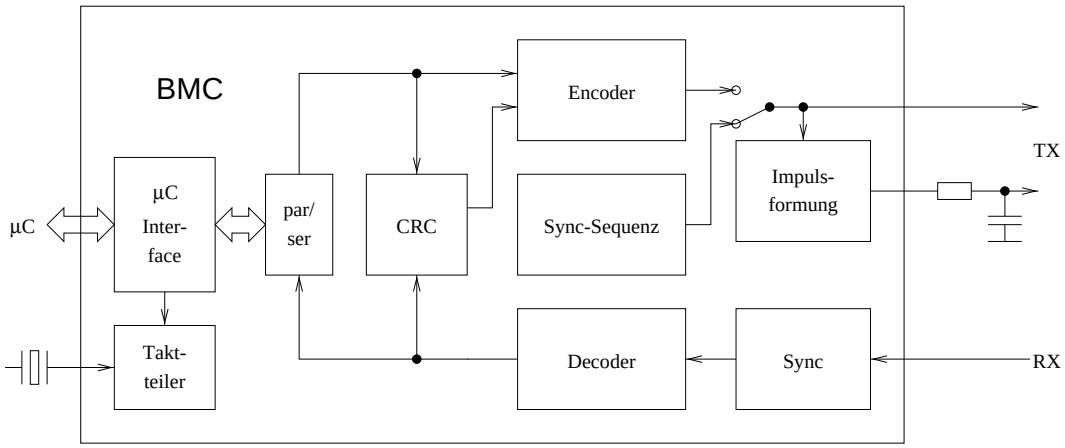


Abbildung 3.2: Blockschaltbild BMC

An Hand des Blockschaltbildes 3.2 soll nun der Signalfluß im Sende- und Empfangsbetrieb dargestellt werden.

Im Sendebetrieb erhält der BMC über das Mikrocontrollerinterface Datenbytes. Diese werden in einen seriellen Bitstrom umgesetzt und vom Encoder codiert. Gleichzeitig wird der CRC Wert berechnet, der nach den Nutzdaten gesendet wird. Vor den Nutzdaten wird eine Synchronisationssequenz gesendet. Durch eine Impulsformungsstufe wird in Verbindung mit einem externen Tiefpaßfilter ein annähernd gaußförmiges Steuersignal für den Sender erzeugt.

Im Empfangsbetrieb durchläuft der Eingangsbitstrom eine Synchronisationsstufe und wird dann decodiert. Der decodierte Bitstrom wird in Bytes umgesetzt und

an die Mikrocontrollerschnittstelle weitergegeben. Der CRC Wert der empfangen Nutzdaten wird berechnet und mit dem übertragenen CRC Wert verglichen.

Da ein Teilnehmer am Funknetz nicht gleichzeitig Senden und Empfangen kann, sind die Funktionen zur parallel-seriell Umsetzung und zur Berechnung des CRCs nur einmal vorhanden. Ein einstellbarer Frequenzteiler dient zur Erzeugung des internen Systemtaktes des BMCs. Die HF-Bitrate beträgt ein Achtel der Systemtaktfrequenz.

3.1.1 Innerer Aufbau und Funktion des BMCs

Die Verilog HDL Beschreibung des BMCs ist hierarchisch in Module gegliedert, die einzelnen Funktionseinheiten des BMCs entsprechen. In diesem Unterabschnitt wird diese Gliederung aufgezeigt und die Funktionsweise des BMCs erläutert. Verschiedene wichtige Teilsysteme sind in weiteren Unterabschnitten genauer beschrieben. Diese Beschreibung kann nicht auf alle Einzelheiten der Funktion des BMCs eingehen, hierzu sei auf die kommentierten Verilog Quelltexte in Anhang B verwiesen.

Abbildung 3.3 zeigt die oberen Ebenen der Modulhierarchie. An der Spitze steht das Modul bmc. Es beinhaltet die Module io, clkdiv und core. Wie in allen Modulen, die weitere Untermodule enthalten, sind in bmc nur Verbindungen zwischen diesen Untermodulen und keine logischen Funktionen definiert. Diese Vorgehensweise ist günstig für die Logiksynthese.

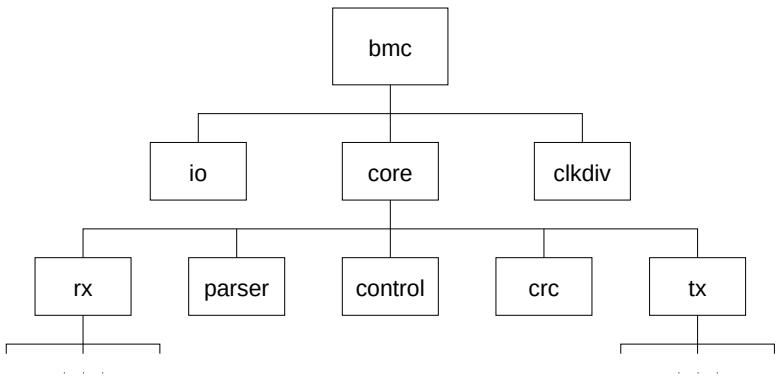


Abbildung 3.3: Modulhierarchie 1

Die Ein- und Ausgangssignale von bmc entsprechen den externen Anschlüssen des BMCs. Dies sind neben den Signalen der Mikrocontrollerschnittstelle, die durch das Modul io definiert und im Abschnitt 3.1.2 beschrieben wird, folgende Signale:

`in_clk`: Eingang für das externe Taktsignal.

`reset`: Ein Low-Pegel an diesem Eingang setzt den BMC in einen definierten Ausgangszustand zurück.

`rx_in`: Eingang für das Empfängersignal.

`receiving`: Dieses Ausgangssignal zeigt an, ob sich der BMC im Empfangsbetrieb befindet. Es kann verwendet werden, um die Zeitkonstante einer Bitentscheidungsschaltung umzuschalten.

`tx_out`: Ausgang für die vom BMC erzeugte Bitfolge.

`tx_pulse`: Dieses PWM Ausgangssignal dient in Verbindung mit einem Tiefpaßfilter zur Erzeugung eines annähernd gaußförmigen Steuersignals für den Sender. Die Grenzfrequenz des externen Tiefpaßfilters muß gleich der halben Bitrate sein.

Das Modul clkdiv realisiert einen einstellbaren Frequenzteiler, der den internen Systemtakt `sys_clk` des BMCs erzeugt. Der Teiler kann durch jede gerade Zahl zwischen 2 und 256 teilen. Die Systemtaktfrequenz beträgt das achtfache der Bitrate, für 20 kbps also 160 kHz.

Die eigentliche Funktionalität des BMCs ist in core und den weiteren Untermodulen beschrieben. tx faßt die Sendefunktionen zusammen, rx die Empfangsfunktionen. Die Module control, parser und crc werden sowohl für Sende- als auch für Empfangsbetrieb benötigt. control bestimmt den Betriebszustand des BMCs, parser dient zur parallel-seriell bzw. seriell-parallel Umsetztung und crc berechnet den CRC Wert von gesendeten bzw. empfangenen Paketen. Abbildung 3.4 zeigt die Module, die die Sendefunktionen ausführen. Die Modulhierarchie der Empfangsfunktionen ist in Abbildung 3.6 auf Seite 17 dargestellt.

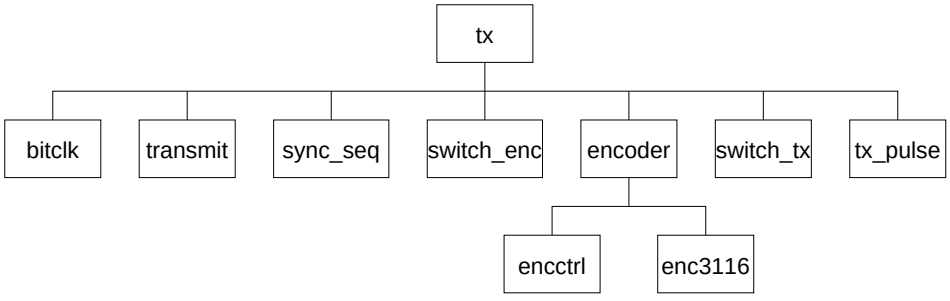


Abbildung 3.4: Modulhierarchie 2

Das Modul bitclk erzeugt den Bittakt `bit_clk` für den Sendebetrieb. `bit_clk` ist während acht Perioden des Systemtakts für eine Periode aktiv. Im Sendebetrieb wird, wenn `bit_clk` aktiv ist, ein neues Bit gesendet.

Die Ablaufsteuerung des Sendebetriebs führt das Modul transmit durch. Eine Übertragung wird vom Modul io durch das Signal `start_tx` gestartet. transmit löst dann zuerst die Übertragung der Synchronisationssequenz aus. Danach wird der Encoder gestartet und die eigentliche Nachricht beginnt. Der Encoder fordert durch `enc_poll` ein Datenbit an und transmit erzeugt die Signale `tx_shift` und `tx_rd` zum Schieben bzw. Laden des parallel-seriell Umsetzers. Das Modul transmit wertet das höchstwertige Bit des ersten übertragenen Bytes und gegebenenfalls das siebte Byte aus, um die Länge des zu sendenden Pakets zu bestimmen. Nach der Nachricht wird noch der CRC Wert übertragen und dann der Sendebetrieb beendet.

Das Modul sync_seq erzeugt die Synchronisationssequenz. Der Encoder für den (31,16)BCH Code ist in dem Modul encoder und den zugehörigen Untermodulen realisiert. Der Encoder wird in Abschnitt 3.1.3 genauer beschrieben.

Die Module switch_enc und switch_tx steuern den Datenfluß im Sendebetrieb. Das Modul tx_pulse erzeugt ein pulsweitenmoduliertes Signal mit dem in Verbindung mit einem externen Tiefpaßfilter ein annähernd gaußförmiges Steuersignal für den Sender erzeugt werden kann.

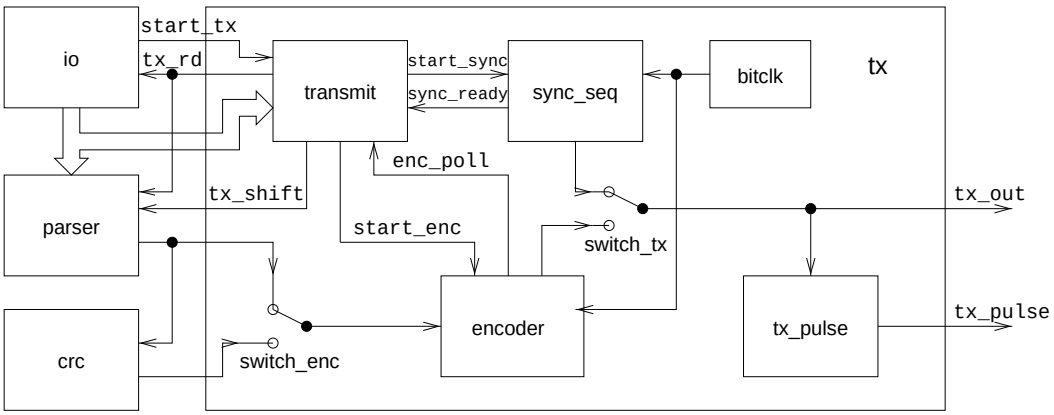


Abbildung 3.5: Signalfluß im Sendebetrieb

Abbildung 3.5 verdeutlicht das Zusammenwirken der verschiedenen Module im Sendebetrieb.

In Abbildung 3.6 ist die Modulhierarchie der Empfangsfunktionen dargestellt. Die Empfangsfunktionen sind unterteilt in die beiden großen Blöcke Decoder und

Synchronisation, denen die Module decoder und sync mit ihren Untermodulen entsprechen, sowie das Modul receive, das den zeitlichen Ablauf des Empfangs steuert. Decoder und Synchronisation werden in den Abschnitten 3.1.4 bzw. 3.1.6 genauer beschrieben.

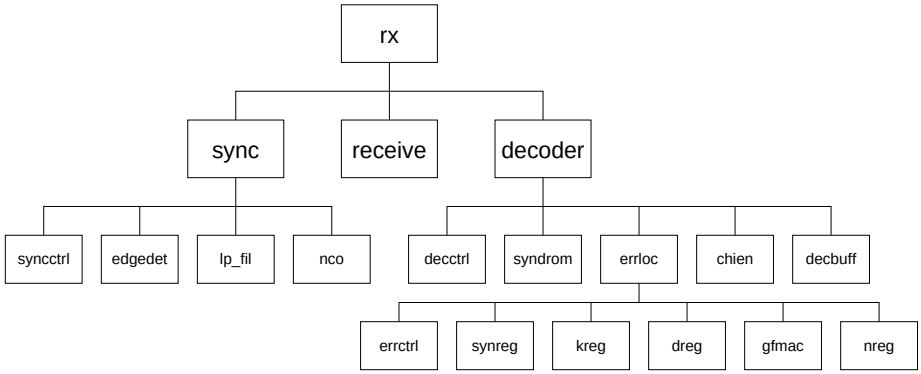


Abbildung 3.6: Modulhierarchie 3

Die Synchronisation erkennt Paketanfänge an Hand der Synchronisationssequenz und startet darauf den Empfangsbetrieb durch `rx_start`. sync liefert den empfangenen Bitstrom und den rückgewonnenen Bittakt an den Decoder. Der Decoder korrigiert Fehler im empfangenen Bitstrom und zeigt durch `valid` an, ob an seinem Ausgang gültige Daten anstehen.

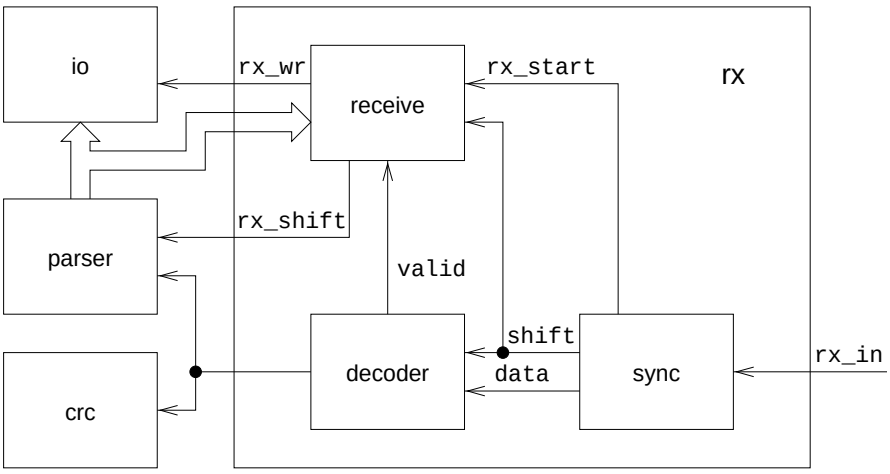


Abbildung 3.7: Signalfluß im Empfangsbetrieb

Das Modul receive erzeugt die Signale `tx_shift` und `tx_wr` um den seriellen Da-

tenstrom in einen parallen umzuwandeln und an die Mikrocontrollerschnittstelle weiterzugeben. receive wertet das höchstwertige Bit des ersten Bytes der empfangenen Daten und gegenbenenfalls das siebte Byte aus, um die Länge des empfangenen Pakets zu ermitteln.

Abbildung 3.7 zeigt das Zusammenwirken der Module im Empfangsbetrieb.

3.1.2 Schnittstelle zum Mikrocontroller

Die Steuerung des BMCs erfolgt durch einen Mikrocontroller. Das Businterface des BMCs ist kompatibel zu den Mikrocontrollerfamilien 8051 und NEC 78k. Tabelle 3.1 zeigt die Signale des Businterfaces.

Signal	Richtung	Bedeutung
`data[7:0]`	bidirektional	Datenbus
`addr[1:0]`	Eingang	Adressbus
`cs0`	Eingang	chip select (active low)
`cs1`	Eingang	chip select (active high)
`rd`	Eingnag	read (active low)
`wr`	Eingang	write (active low)
`intr`	Ausgang	Interrupt (active low)

Tabelle 3.1: Businterface

Der BMC ist selektiert, wenn an `cs0` Low-Pegel und an `cs1` High-Pegel anliegt. Low-Pegel an `rd` zeigt einen Lesezugriff an, Low-Pegel an `wr` einen Schreibzugriff. Über die zwei Adressleitungen `addr[1:0]` wird eines von vier Registern ausgewählt. Der Datenaustausch erfolgt über den 8 Bit breiten Datenbus `data[7:0]`. Durch eine fallende Flanke auf `intr` teilt der BMC dem Mikrocontroller eine Unterbrechungsanforderung mit.

Der BMC verfügt über vier Register, die vom Mikrocontroller wie Speicherzellen angesprochen werden. Die Bedeutung der einzelnen Register ist in Tabelle 3.2 dargestellt.

Register 0 ist das Datenregister. Im Sendebertieb werden die zu übertragenden Daten in dieses Register geschrieben, im Empfangsbetrieb werden die empfangenen Daten von hier gelesen. Befindet sich der BMC im Ruhezustand oder Empfangsbetrieb, so löst ein Schreibzugriff auf dieses Register den Übergang in den Sendebetrieb aus.

Nr.	addr 1 0	**Zugriff**	**Register**
0	0 0	read	Empfangsbyte
		write	Sendebyte
1	0 1	read	Status
		write	-
2	1 0	read	Taktteilerfaktor
		write	(Bits 6–0)
3	1 1	read	untere Grenzfrequenz
		write	(Bits 3–0)

Tabelle 3.2: Registerbelegung

Der Inhalt von Register 1 stellt wichtige interne Zustände des BMCs dar. Die Bedeutung der einzelnen Bits von Register 1 ist in Tabelle 3.3 zusammengefaßt. Wenn alle Bits des Statusregisters gesetzt sind, so bedeutet das, daß der BMC zur Zeit keine Zugriffe verarbeiten kann. Auf die Register 0, 2 und 3 darf nur zugegriffen werden, wenn vorher das Register 1 gelesen wurde und überprüft wurde, daß der Inhalt nicht FF_{hex} ist.

Bit	**Bedeutung**
0	Empfangspuffer voll
1	Sendepuffer leer
2	Kanal belegt
3	Empfangsbetrieb
4	Sendebetrieb
5	Empfangsfehler
6	CRC Fehler
7	Sendefehler

Tabelle 3.3: Statusregister

Bit 0 zeigt an, daß ein Byte empfangen wurde. Dieses Byte kann aus Register 0 gelesen werden, der Lesezugriff auf Register 0 setzt Bit 0 zurück. Wird ein weiteres Byte empfangen, bevor Register 0 gelesen wurde, so wird Bit 5, Empfangsfehler, gesetzt.

Bit 1 gesetzt bedeutet, daß der BMC ein weiteres Sendebyte verarbeiten kann. Dieses Byte wird in Register 0 geschrieben, der Schreibzugriff auf Register 0 setzt

Bit 1 zurück. Wird nicht rechtzeitig ein neues Sendebyte übergeben, so setzt der BMC Bit 7, Sendefehler, und bricht die Übertragung ab.

Bit 2 kann zur Entscheidung, ob eine Übertragung begonnen werden soll, herangezogen werden. Wenn dieses Bit gesetzt ist und der BMC sich im Leerlauf befindet, so bedeutet das, daß mit hoher Wahrscheinlichkeit in Kürze der Begin einer Übertragung erkannt wird.

Die Bits 3 und 4 zeigen den Betriebszustand des BMCs an. Beide Bits gleich Null bedeuten Leerlauf, Bit 3 gesetzt bedeutet Empfangsbetrieb, Bit 4 gesetzt bedeutet Sendebetrieb.

Die Bits 5 bis 7 weisen auf Fehler hin. Bit 5 gesetzt bedeutet, daß beim letzten Paket ein Empfangsfehler aufgetreten ist. Bit 6 zeigt an, daß beim letzten empfangenen Paket der CRC Wert fehlerhaft war. Bit 7 signalisiert einen Fehler beim letzten Sendeversuch.

Wenn die Bits 0, 1, 5, 6 oder 7 gesetzt werden löst der BMC einen Interrupt aus. Ein anschließender Lesezugriff auf das Statusregister setzt den Interrupt zurück.

Register 2 stellt den Teilfaktor des internen Taktteilers ein. Dies beeinflußt die Kanaldatenrate, sie beträgt $f_{CLK}/16(n+1)$ Bits pro Sekunde, wobei f_{CLK} die externe Taktfrequenz und n der Inhalt von Register 2 ist. n kann einen Wert zwischen 0 und 127 annehmen. Nach einem Reset ist ein Wert von 30 eingestellt, damit ergibt sich bei einer externen Taktfrequenz von 10 MHz eine interne Systemtaktfrequenz von 161,3 kHz und eine Bitrate von 20,2 kbps.

Der Wert von Register 3 bestimmt die maximale Anzahl von gleichen Bits, die im Ausgangsdatenstrom aufeinanderfolgen können. Dieser Mechanismus ist für die Synchronisation von Bedeutung und wird im Abschnitt 3.1.6 näher erläutert. Der zulässige Wertebereich ist 1 bis 15, nach einem Reset ist 4 eingestellt.

Im Leerlauf ist der Inhalt des Statusregisters typischerweise 02_{hex}, wenn beim letzten Sende- oder Empfangsversuch Fehler aufgetreten sind können auch die Bits 5 bis 7 gesetzt sein. Im folgenden wird der Ablauf der Kommunikation zwischen Mikrocontroller und BMC beim Sende- und Empfangsvorgang beschrieben.

Der Sendevorgang wird vom Mikrocontroller durch das Schreiben des ersten zu übertragenden Bytes in das Register 0 gestartet. An Hand von Bit 7 dieses Bytes entscheidet der BMC, ob ein Paket mit oder ohne Nutzdaten gesendet werden soll. Ist das Bit gesetzt, werden Nutzdaten übertragen. Wenn der BMC das erste Byte verarbeitet hat, löst er einen Interrupt aus. Der Inhalt des Statusregisters ist dann typisch 12_{hex}. Der Mikrocontroller übergibt das nächste Byte, dem wieder ein Interrupt folgt, usw. Bei einem Paket ohne Nutzdaten werden so 6 Byte übergeben, bei einem Paket mit Nutzdaten folgt als siebtes Byte die Anzahl der

Datenbytes und danach die eigentlichen Daten. Der BMC wertet das siebte Byte zur Steuerung des weiteren Sendeablaufs aus. Wenn der BMC die Übertragung beendet hat löst er einen weiteren Interrupt aus. Der Mikrocontroller kann dann an Hand von Bit 7 des Statusregisters feststellen, ob ein Sendefehler aufgetreten ist. Zum Zeitpunkt des letzten Interrupts befindet sich der BMC wieder im Ruhezustand, der Inhalt des Statusregisters ist typischerweise 02_{hex}, wenn keine Fehler aufgetreten sind.

Erkennt der BMC eine gültige Synchronisationsequenz am Empfängereingang, so wechselt er in den Empfangsbetrieb. Nach jedem empfangenen Byte löst der BMC einen Interrupt aus. Der Inhalt des Statusregisters ist dann typischerweise $0F_{\text{hex}}$. Der BMC wertet das höchstwertige Bit des ersten Bytes und das siebte Byte der empfangenen Daten zur Steuerung des Empfangsablaufs aus. Nachdem auch der CRC empfangen wurde, wird ein weiterer Interrupt erzeugt. Der Mikrocontroller kann dann an Hand von Bit 5 und 6 des Statusregisters feststellen, ob ein Empfangs- oder CRC-Fehler aufgetreten ist. Zum Zeitpunkt des letzten Interrupts befindet sich der BMC wieder im Ruhezustand, der Inhalt des Statusregisters ist typischerweise 02_{hex}, wenn keine Fehler aufgetreten sind.

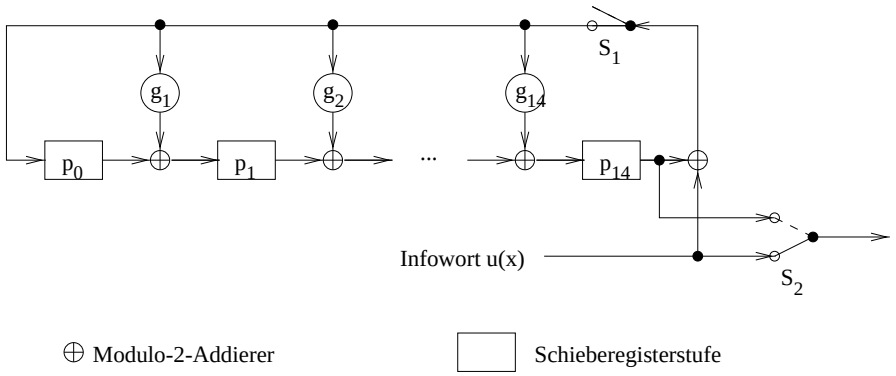
3.1.3 Encoder

Die Schaltung des Encoders für den (31,16)BCH Code ist in Abbildung 3.8 dargestellt. Es handelt sich um eine Schaltung aus rückgekoppelten Schieberegistern. Viele Hinweise und Anregungen zur Implementierung von Encodern und Decodern für zyklische Codes sowie Galoisfeldarithmetik mit Schieberegistern sind in [1], [3] und [9] zu finden.

Ein Codewort wird folgendermaßen berechnet: Zunächst werden mit der gezeigten Schalterstellung 16 Infobits $u(x)$ nacheinander in den Encoder und gleichzeitig in den Übertragungskanal geschoben, dann entspricht der Inhalt des Schieberegisters dem Divisionsrest von $u(x) \cdot x^{15}/g(x)$, also den Prüfbits. Die Prüfbits werden jetzt bei geöffnetem Schalter S_1 bzw. umgelegten S_2 in den Kanal geschoben.

Das Verilog Modul encoder enthält die zwei Untermodule enc3116 und encctrl. enc3116 realisiert den eigentlichen Encoder wie in Abblidung 3.8 gezeigt, während encctrl den zeitlichen Ablauf des Codiervorgangs steuert.

Neben der Codierung hat der Encoder die Aufgabe in regelmäßigen Abständen Bitwechsel (Flanken) in den codierten Datenstrom einzufügen, um dem Empfänger eine Synchronisation auch dann zu ermöglichen, wenn lange Folgen von 0 oder 1 Bits übertragen werden (siehe 3.1.6). Die Anzahl der Datenbits,



$$g(x) = \sum_{i=0}^{15} g_i x^i = 1 + x + x^2 + x^3 + x^5 + x^7 + x^8 + x^9 + x^{10} + x^{11} + x^{15}$$

Abbildung 3.8: Encoder

nach denen ein Bitwechsel eingefügt wird, ist durch das Register 3 der Mikrocontrollerschnittstelle einstellbar.

3.1.4 Decoder

Das Decoderblockschaltbild 3.9 zeigt die Funktionsweise des Decoders für den (31,16)BCH Code. Die empfangenen Datenbits werden in ein Pufferregister geschoben, während gleichzeitig die Syndromkomponenten berechnet werden. Wenn ein komplettes Codewort eingeschoben wurde, wird aus den Syndromkomponenten das Fehlerstellenpolynom berechnet. Die inversen Nullstellen des Fehlerstellenpolynoms werden mit der sogenannten Chien-Suche ermittelt und die Übertragungsfehler werden korrigiert.

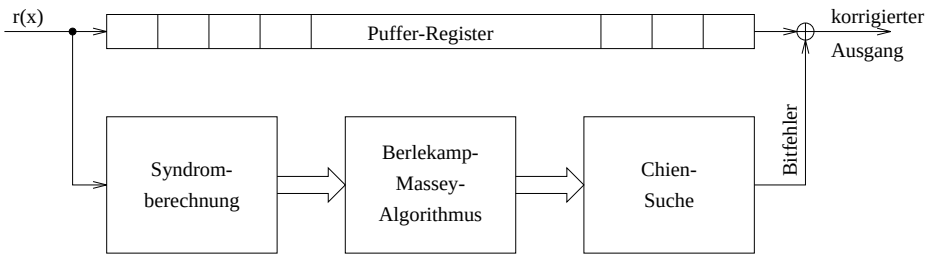


Abbildung 3.9: Decoderblockschaltbild

Das Pufferregister ist 38 Bit lang. So erreicht ein empfangenes Codewort genau dann den Ausgang des Pufferregisters, wenn die Berechnung des Fehlerstellenpolynoms abgeschlossen ist. Auf diese Weise ist kontinuierlicher Betrieb möglich: Die empfangenen Bits werden am Eingang des Decoders eingeschoben und erscheinen mit einer Verzögerung von 38 Bitperioden korrigiert am Ausgang.

Den in Abbildung 3.9 gezeigten Blöcken entsprchen die Verilog Module decbuff, syndrom, errloc und dessen Untermodule sowie chien. Das Modul decctrl steuert den zeitlichen Ablauf des Zusammenwirkens der Teilmodule des Decoders.

Syndromberechnung

Die Syndromkomponenten S_i sind wie folgt definiert:

$$S_i = r(\alpha^i) = r_0 + r_1\alpha^i + r_2(\alpha^i)^2 + \cdots + r_{30}(\alpha^i)^{30}, \qquad i = 1, 2, \cdots, 6$$

Die Schaltung, die diese Berechnungen durchführt ist in Abbildung 3.10 gezeigt.

Die Schaltung zur Syndromberechnung nutzt einige mathematische Eigenschaften von Polynomen über Galoisfeldern aus. Sei $\Phi_i(x)$ das Minimalpolynom von α^i und $b_i(x)$ der Divisionsrest der Division $r(x)/\Phi_i(x)$. Dann gilt, [9]:

$$r(\alpha^i) = b_i(\alpha^i)$$

Die Minimalpolynome von $\alpha, \alpha^2, \cdots, \alpha^6$ in GF(32) sind, [17]:

$$\begin{aligned} \alpha, \alpha^2, \alpha^4 &: 1 + x^2 + x^5 \\ \alpha^3, \alpha^6 &: 1 + x^2 + x^3 + x^4 + x^5 \\ \alpha^5 &: 1 + x + x^2 + x^4 + x^5 \end{aligned}$$

Wie man sieht, gilt:

$$b_1(x) = b_2(x) = b_4(x), \qquad b_3(x) = b_6(x)$$

Die $b_i(x)$ werden durch die drei rückgekoppelten Schieberegister in Abbildung 3.10 berechnet. In die $b_i(x)$ müssen nun noch die α^i eingesetzt werden. Dazu dienen die Modulo-2-Addierer an den Ausgängen. An Hand von zwei Beispielen sei die Vorgehensweise exemplarisch gezeigt.

Ein $b_i(x)$ hat die Form:

$$b_i(x) = b_{i0} + b_{i1}x + b_{i2}x^2 + b_{i3}x^3 + b_{i4}x^4$$

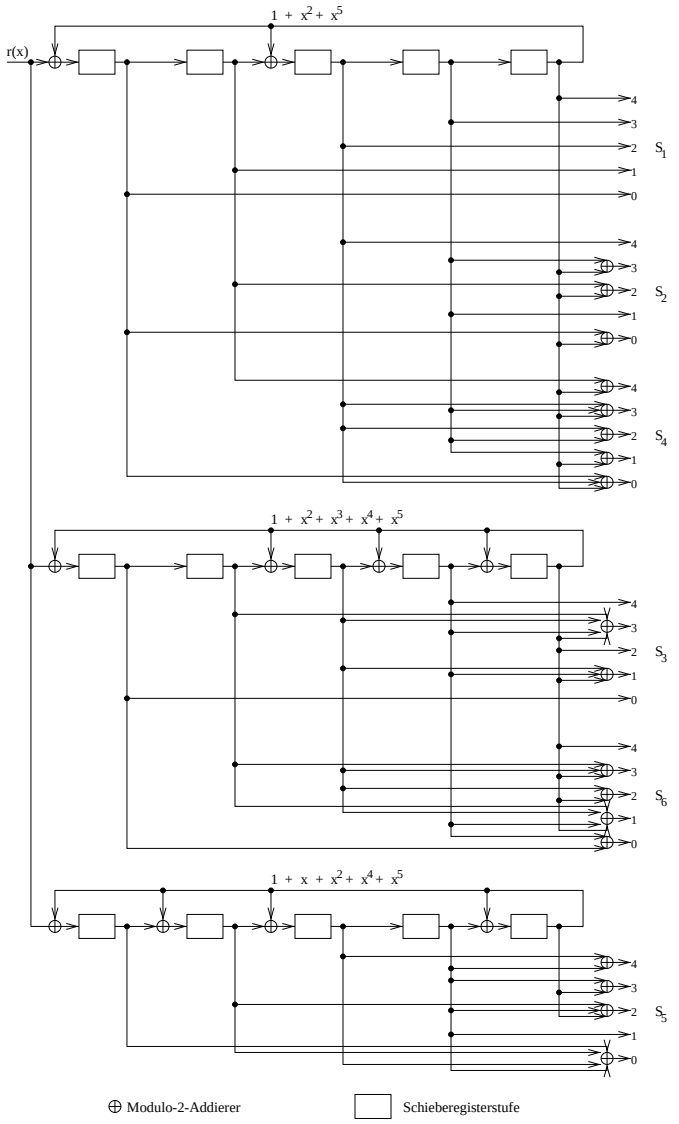


Abbildung 3.10: Syndromberechnug

Für S_2 gilt:

$$\begin{aligned} r(\alpha^2) &= b_2(\alpha^2) \\ &= b_{20} + b_{21}\alpha^2 + b_{22}\alpha^4 + b_{23}\alpha^6 + b_{24}\alpha^8 \\ &= b_{20} + b_{21}\alpha^2 + b_{22}\alpha^4 + b_{23}(\alpha^3 + \alpha) + b_{24}(\alpha^3 + \alpha^2 + 1) \\ &= (b_{20} + b_{24}) + b_{23}\alpha + (b_{21} + b_{24})\alpha^2 + (b_{23} + b_{24})\alpha^3 + b_{22}\alpha^4 \end{aligned}$$

Für S_3 gilt:

$$\begin{aligned} r(\alpha^3) &= b_3(\alpha^3) \\ &= b_{30} + b_{31}\alpha^3 + b_{32}\alpha^6 + b_{33}\alpha^9 + b_{34}\alpha^12 \\ &= b_{30} + b_{31}\alpha^3 + b_{32}(\alpha^3 + \alpha) + b_{33}(\alpha^4 + \alpha^3 + \alpha) + b_{34}(\alpha^3 + \alpha^2 + \alpha) \\ &= b_{30} + (b_{32} + b_{33} + b_{34})\alpha + b_{34}\alpha^2 + (b_{31} + b_{32} + b_{33} + b_{34})\alpha^3 + b_{33}\alpha^4 \end{aligned}$$

Berechnung des Fehlerstellenpolynoms

Das Fehlerstellenpolynom wird aus den Syndromkomponenten nach dem Berlekamp-Massey-Algorithmus berechnet. Im ursprünglichen Berlekamp-Massey-Algorithmus ist nach Gleichung 2.2 unter Umständen eine Division duchzuführen. Da eine Division eine sehr aufwendige Rechenoperation ist, wird hier ein modifizierter Algorithmus verwendet, bei dem kein Quotient berechnet werden muß.

Modifizierter BMA, [5]: Definiere:

$$1 + S = 1 + S_1 x + S_2 x^2 + \cdots + S_{2t} x^{2t}$$

$$\nu^{(0)} = 1, \qquad \kappa^{(0)} = 1, \qquad \delta^{(-2)} = 1$$

Dann führe folgende rekursive Berechnungen durch:

Wenn S_{2k+1} nicht bekannt ist, beende die Rekursion. Sonst setze $d^{(2k)}$ gleich dem Koeffizienten von x^{2k+1} in dem Produkt $(1 + S)\nu^{(2k)}$. Berechne:

$$\begin{aligned} \nu^{(2k+2)} &= \delta^{(2k-2)}\nu^{(2k)} + d^{(2k)}\kappa^{(2k)}x \\ \kappa^{(2k+2)} &= \begin{cases} x^2\kappa^{(2k)}, & \text{wenn } d^{(2k)} = 0 \text{ oder Grad}\,\nu^{(2k)} > k \\ x\nu^{(2k)}, & \text{wenn } d^{(2k)} \neq 0 \text{ und Grad}\,\nu^{(2k)} \leq k \end{cases} \\ \delta^{(2k)} &= \begin{cases} \delta^{(2k-2)}, & \text{wenn } d^{(2k)} = 0 \text{ oder Grad}\,\nu^{(2k)} > k \\ d^{(2k)}, & \text{wenn } d^{(2k)} \neq 0 \text{ und Grad}\,\nu^{(2k)} \leq k \end{cases} \end{aligned}$$

Die Schaltung, die das Fehlerstellenpolynom berechnet ist in Abbildung 3.11 dargestellt. Die Schaltung besteht aus Registern für $s(x)$, $\kappa(x)$, $\nu(x)$ und $\nu(x)$ der letzten Iteration, d und δ, sowie einer Schaltung um zwei Elemente aus GF(32) zu multiplizieren oder zu addieren. Diesen Funktionseinheiten entsprechen die Module synreg, kreg, nreg, dreg und gfmac. Das Modul errctrl steuert den Ablauf der Berechnung.

Der erste Schritt in jeder Iteration ist die Berechnung von $d^{(2k)}$. Dazu ist der Koeffizient von x^{2k+1} in dem Produkt $(1 + S)\nu^{(2k)}$ zu berechnen:

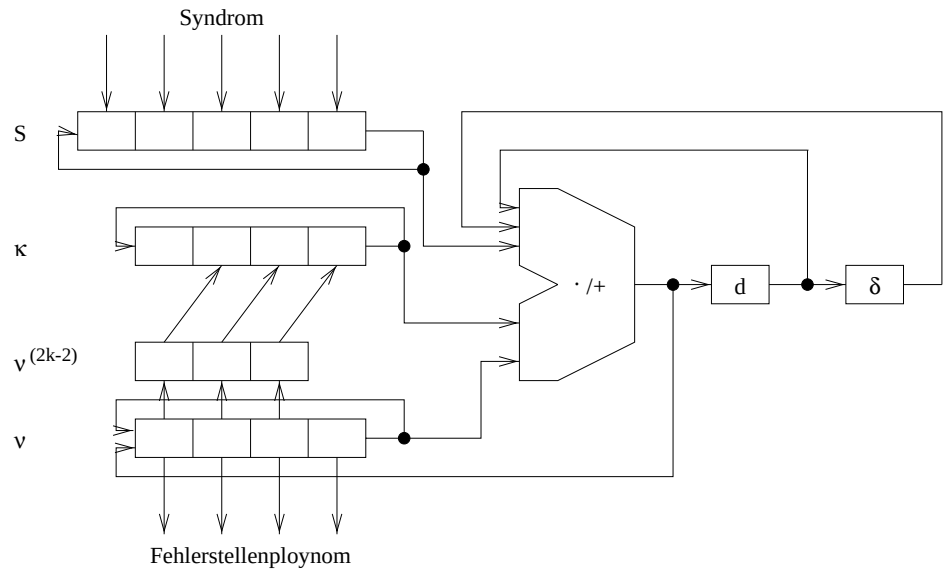


Abbildung 3.11: Berechnung des Fehlerstellenpolynoms

$$(\nu_0 + \nu_1 x + \nu_2 x^2 + \nu_3 x^3)(1 + S_1 x + S_2 x^2 + S_3 x^3 + S_4 x^4 + S_5 x^5 + S_6 x^6)$$

$$\begin{array}{rcl}
 & & \downarrow \qquad\qquad \downarrow \qquad\qquad \downarrow \\
 & = & \nu_0 \;+ \nu_0 S_1 x + \nu_0 S_2 x^2 + \nu_0 S_3 x^3 + \nu_0 S_4 x^4 + \nu_0 S_5 x^5 + \nu_0 S_6 x^6 \\
 & + & \nu_1 S_6 + \;\nu_1 x \;+ \nu_1 S_1 x^2 + \nu_1 S_2 x^3 + \nu_1 S_3 x^4 + \nu_1 S_4 x^5 + \nu_1 S_5 x^6 \\
 & + & \nu_2 S_5 + \nu_2 S_6 x + \;\nu_2 x^2 \;+ \nu_2 S_1 x^3 + \nu_2 S_2 x^4 + \nu_2 S_3 x^5 + \nu_2 S_4 x^6 \\
 & + & \nu_3 S_4 + \nu_3 S_5 x + \nu_3 S_6 x^2 + \;\nu_3 x^3 \;+ \nu_3 S_1 x^4 + \nu_3 S_2 x^5 + \nu_3 S_3 x^6
\end{array}$$

Für die $d^{(2k)}$ ergibt sich somit:

$$\begin{array}{rcll}
d^{(0)} & = & \nu_0^{(0)} S_1 + \nu_1^{(0)} + \nu_2^{(0)} S_6 + \nu_3^{(0)} S_5 & \\
 & = & S_1 & \text{, da } \nu_0^{(0)} = 1,\ \nu_1^{(0)} = \nu_2^{(0)} = \nu_3^{(0)} = 0 \\
d^{(2)} & = & \nu_0^{(2)} S_3 + \nu_1^{(2)} S_2 + \nu_2^{(2)} S_1 + \nu_3^{(2)} & \\
d^{(4)} & = & \nu_0^{(4)} S_5 + \nu_1^{(4)} S_4 + \nu_2^{(4)} S_3 + \nu_3^{(4)} S_2 &
\end{array}$$

Wie man sieht, wird die Syndromkomponente S_6 zur Berechnung des Fehlerstellenpolynoms nicht benötigt. Mit

$$\nu^{(0)} = 1, \qquad \kappa^{(0)} = 1, \qquad d^{(0)} = S_1, \qquad \delta^{(-2)} = 1$$

gilt weiter:

$$\nu^{(2)}(x) = 1 + S_1 x$$

Die Register werden entsprechend vorbesetzt und die Berechnung des Fehlerstellenpolynoms beginnt mit der Berechnung von $\kappa^{(2)}$.

Das Modul gfmac ist das Herzstück der Schaltung zur Berechnung des Fehlerstellenpolynoms. Es führt Multiplikationen und Additionen in GF(32) aus. Zur Multiplikation zweier Elemente aus GF(2^m) sind in der Literatur verschiedene Schaltungen mit rückgekoppelten Schieberegistern vorgeschlagen, [1], [9]. Diese Schaltungen haben den Nachteil, daß eine Multiplikation in GF(32) sechs Taktzyklen dauert. Deshalb wurde eine Methode entwickelt, mit der das Produkt von zwei Elementen aus GF(32) durch kombinatorische Logik berechnet wird.

Für $a, b, c \in \mathrm{GF}(32)$ gilt:

$$\begin{aligned} a &= a_0 + a_1\alpha + a_2\alpha^2 + a_3\alpha^3 + a_4\alpha^4 \\ b &= b_0 + b_1\alpha + b_2\alpha^2 + b_3\alpha^3 + b_4\alpha^4 \\ c &= c_0 + c_1\alpha + c_2\alpha^2 + c_3\alpha^3 + c_4\alpha^4 \end{aligned}$$

$$\begin{aligned} a &= b \cdot c \\ &= (b_0 + b_1\alpha + b_2\alpha^2 + b_3\alpha^3 + b_4\alpha^4) \cdot (c_0 + c_1\alpha + c_2\alpha^2 + c_3\alpha^3 + c_4\alpha^4) \\ &= (b_0c_0 + b_1c_4 + b_2c_3 + b_3c_2 + b_4c_1 + b_4c_4) \\ &+ (b_0c_1 + b_1c_0 + b_2c_4 + b_3c_3 + b_4c_2)\,\alpha \\ &+ (b_0c_2 + b_1c_1 + b_1c_4 + b_2c_0 + b_2c_3 + b_3c_2 + b_3c_4 + b_4c_1 + b_4c_3 + b_4c_4)\,\alpha^2 \\ &+ (b_0c_3 + b_1c_2 + b_2c_1 + b_2c_4 + b_3c_0 + b_3c_3 + b_4c_2 + b_4c_4)\,\alpha^3 \\ &+ (b_0c_4 + b_1c_3 + b_2c_2 + b_3c_1 + b_3c_4 + b_4c_0 + b_4c_3)\,\alpha^4 \end{aligned}$$

In der Harwarerealisierung dieser Gleichung entspricht jeder b_ic_j Term einer UND Verknüpfung und jede Addition einer EXKLUSIV-ODER Verknüpfung. Die kombinatorische Schaltung zur Multiplikation benötigt geringfügig mehr Ressourcen des FPGAs als eine Schaltung mit rückgekoppelten Schieberegistern. Da die Berechnung des Fehlerstellenpolynoms aber wesentlich schneller erfolgt, kann das Pufferregister des Decoders kürzer ausfallen. Dies gleicht den größeren Aufwand für den Multiplizierer mehr als aus und verkürzt die Latenzzeit des Decoders. In Tabelle 3.4 sind die beiden Realisierungen gegenübergestellt. Der Ressourcenverbrauch ist in CLB angegeben, das sind Funktionseinheiten des XC 4013 FPGAs.

	Kombinatorik	**Schieberegister**
gfmac	29 CLB	27 CLB
decbuff	19 CLB (38 Bit)	27 CLB (53 Bit)
Summe	48 CLB	54 CLB

Tabelle 3.4: Multiplizierer in GF(32)

Berechnung der Fehlerstellen

Die Fehlerstellen im empfangenen Wort $r(x)$ werden durch die multiplikativen Inversen der Nullstellen des Fehlerstellenpolynoms bestimmt. Das bedeutet, daß Bit 30 des empfangenen Worts fehlerhaft ist, wenn α^1 Nullstelle von $\nu(x)$ ist, Bit 29 ist fehlerhaft, wenn α^2 Nullstelle ist, usw. Um zu bestimmen, ob Bit k fehlerhaft ist, ist folgende Bedingung zu prüfen:

$$\sum_{i=0}^{3} \nu_i (\alpha^i)^{31-k} \stackrel{?}{=} 0, \qquad k = 0, 1, 2, \cdots, 30$$

Die Nullstellen des Fehlerstellenpolynoms werden nach einer Methode bestimmt, die als Chien-Suche bekannt ist, [6]. Das empfangene Wort erscheint mit Bit 30 zuerst am Ausgang des Pufferregisters. Dieses Bit ist fehlerhaft, wenn gilt: $\nu_0 + \nu_1\alpha + \nu_2\alpha^2 + \nu_3\alpha^3 = 0$. Wird nun Bit 29 an den Ausgang geschoben, heißt die Bedingung für einen Fehler $\nu_0 + \nu_1\alpha^2 + \nu_2\alpha^4 + \nu_3\alpha^6 = 0$, usw. Die Chien-Suche muß demnach jedesmal, wenn ein neues Bit an den Ausgang des Pufferregisters geschoben wird, folgende Berechnung durchführen:

$$\nu_0 + \nu_1\alpha^n \cdot \alpha + \nu_2(\alpha^2)^n \cdot \alpha^2 + \nu_3(\alpha^3)^n \cdot \alpha^3$$

Die Schaltung, die diese Berechnung durchführt zeigt Abbildung 3.12. Diese Schaltung ist im Modul chien realisert.

Die Register für ν_1, ν_2 und ν_3 sind so verschaltet, daß sie eine Multiplikation mit α, α^2 bzw. α^3 ausführen können. Dies sei an Hand von ν_1 verdeutlicht:

ν_1 hat die Form:

$$\nu_1 = \nu_{10} + \nu_{11}\alpha + \nu_{12}\alpha^2 + \nu_{13}\alpha^3 + \nu_{14}\alpha^4$$

Es gilt:

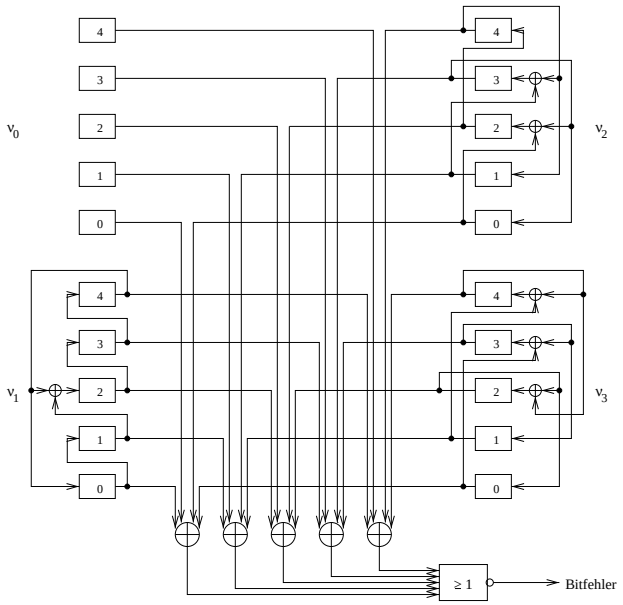


Abbildung 3.12: Chien-Suche

$$\begin{aligned}\nu_1 \cdot \alpha &= \nu_{10}\alpha + \nu_{11}\alpha^2 + \nu_{12}\alpha^3 + \nu_{13}\alpha^4 + \nu_{14}\alpha^5 \\ &= \nu_{10}\alpha + \nu_{11}\alpha^2 + \nu_{12}\alpha^3 + \nu_{13}\alpha^4 + \nu_{14}(1+\alpha^2) \\ &= \nu_{14} + \nu_{10}\alpha + (\nu_{11}+\nu_{14})\alpha^2 + \nu_{12}\alpha^3 + \nu_{13}\alpha^4\end{aligned}$$

3.1.5 Cyclic Redundancy Check

Die Schaltung zur Berechnung des CRC Werts eines gesendeten bzw. empfangenen Pakets entspricht der Schaltung des Encoders mit folgenden Unterschieden:

- Das Schieberegister ist 16 Bit lang.
- Das Polynom, das die Rückkopplung definiert lautet:

$$g_{CRC}(x) = 1 + x^5 + x^{12} + x^{16}$$

Im Empfangsbetrieb werden die empfangen Daten und der empfangene CRC Wert seriell in die Schaltung zur CRC Berechnung geschoben. Wenn kein Übertragungsfehler aufgetreten ist, ist der Inhalt des Schieberegisters nach Empfang des CRC Werts gleich Null.

3.1.6 Synchronisation

Die Synchronisation hat in jedem Datenübertragungssystem eine große Bedeutung. Der Empfänger muß aus dem empfangenen Datenstrom den Takt ableiten, mit dem die Daten gesendet wurden. Diesen Vorgang nennt man Bitsynchronisation. Bei einer paketorientierten Übertragung wie mit dem BMC muß zudem der Anfang eines Pakets erkannt werden. Dies wird als Rahmensynchronisation bezeichnet. Um dem Empfänger Bit- und Rahmensynchronisation zu ermöglichen, müssen bereits im Sender Vorkehrungen getroffen werden.

Der Sender sendet zu Beginn jedes Pakets eine Synchronisationssequenz (Präambel), die zur Bit- und Rahmensynchronisation dient. Abbildung 3.13 zeigt den Aufbau dieser Bitfolge.

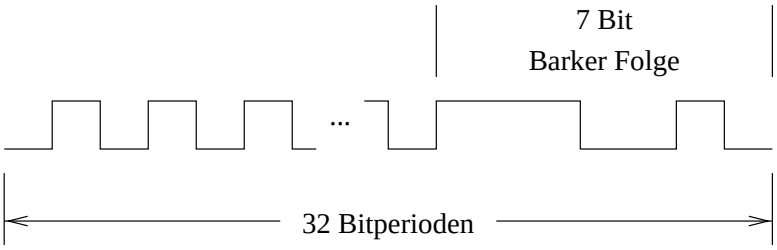


Abbildung 3.13: Synchronisationssequenz

Die Sequenz, die 32 Bitperioden lang ist, beginnt mit einer abwechselnden Folge von 0 und 1 Bits, die zur Bitsynchronisation dienen. Den Abschluß der Synchronisationssequenz bildet eine 7 Bit Barker Folge zur Kennzeichnung des Paketanfangs. Barker Folgen sind auf Grund ihrer Autokorrelationseigenschaften besonders zur Rahmensynchronisation geeignet, [12].

Um während der Übertragung eines Pakets die Bitsynchronisation nicht zu verlieren, müssen im Datenstrom in regelmäßigen Abständen Flanken auftreten. Der BMC erfüllt diese Forderung, indem nach einer einstellbaren Anzahl von Datenbits ein Bitwechsel eingefügt wird. Die Anzahl der Datenbits ist über Register 3 der Mikrocontrollerschnittstelle (siehe 3.1.2) einstellbar. Der Normalwert beträgt vier, das bedeutet, daß nach jedem vierten Datenbit ein Füllbit eingefügt wird, das der Synchronisation dient.

Zur Bittaktrückgewinnung im Empfänger wird eine volldigitale PLL eingesetzt. Abbildung 3.14 zeigt ein Blockschaltbild der PLL. Ein numerisch gesteuerter Oszilator (NCO) erzeugt den Bittakt des Empfängers, der durch den Phasenkomparator und das Schleifenfilter auf den Takt der empfangenen Daten geregelt

wird. Diesen Funktionseinheiten entsprechen die Verilog Module nco, edgedet und lp_fil.

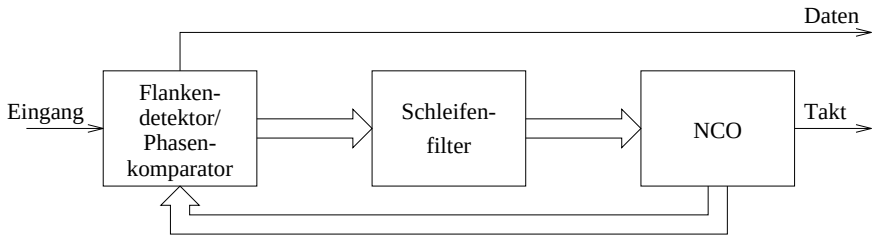


Abbildung 3.14: PLL

Abbildung 3.15 zeigt die Funktionsweise des NCOs. Der NCO besteht aus zwei Registern und einem Addierer. Bei jeder steigenden Flanke des Systemtakts wird das Phaseninkrement zum Inhalt des Phasenakkumulators addiert. Bei einem Überlauf des Phasenakkumulators wird der Bittaktausgang aktiv. Das Phasenakkumulatorregister hat eine Breite von 13 Bit. Mit einem Phaseninkrement von 1024 ergibt sich eine Bittaktfrequenz von einem Achtel der Systemtaktfrequenz. Ein Phaseninkrement größer als 1024 erzeut eine Bittaktfrequenz größer als ein Achtel der Systemtaktfrequenz und umgekehrt.

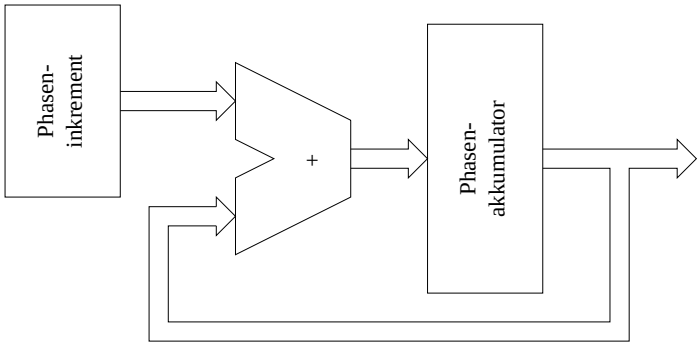


Abbildung 3.15: NCO

Durch das Ausgangssignal des Schleifenfilters wird das Phaseninkrement verändert, der Inhalt des Phasenakkumulators wird vom Phasenkomparator zur Bestimmung des Phasenfehlers verwendet.

Der Phasenkomparator ermittelt einen Wert für den Phasenfehler jedesmal, wenn im Eingangssignal eine Flanke auftritt. Der Phasenfehler zu diesem Zeitpunkt ist gleich dem Inhalt des Phasenakkumulators geteilt durch die Anzahl von Bits, die

zwischen dieser Flanke und der vorhergehenden Flanke übertragen wurden. Um die Komplexität der Hardware zu begrenzen, wird hier durch die Zweierpotenz geteilt, die den kleinsten Fehler ergibt.

Zur Flankenerkennung wird das Eingangssignal mit der Systemtaktfrequenz abgetastet und die letzten acht Abtastwerte gespeichert. Eine Flanke wird abhängig vom Betriebszustand des BMCs auf unterschiedliche Weise erkannt. Befindet sich der BMC nicht im Empfangszustand, so müssen die acht Abtastwerte dem Bitmuster 11111110 entsprechen und die letzte Flanke muß eine steigende Flanke gewesen sein, damit eine fallende Flanke erkannt wird. Eine steigende Flanke wird erkannt, wenn die Abtastwerte dem Bitmuster 00000001 entsprechen und die letzte Flanke eine fallende war. Diese restriktiven Kriterien für eine Flankenerkennung sind nötig, damit die PLL nicht auf zufällige Rauschereignisse synchronisiert. Im Empfangsbetrieb ist die PLL eingeschwungen. Eine Flanke wird erkannt, wenn die letzten 4 Abtastwerte den Bitmustern 1110 oder 0001 entspechen und die Flanke zu einem Zeitpunkt maximal zwei Periodendauern des Systemtakts vor oder nach einem Zeitpunkt auftrit, zu dem eine Flanke erwartet wird. Dadurch wird erreicht, daß eine eimal erreichte Bitsynchronisation auch bei Störungen gehalten werden kann.

Neben der Flankenerkennung und der Ermittlung des Phasenfehlers bestimmt das Modul `edgedet` den Wert des aktuell empfangenen Bits. Zu diesem Zweck werden die sieben Abtasterte vor einer Flanke betrachtet. Sind von diesen sieben Abtastwerten mehr als drei 1, so wird auf 1 entschieden, andernfalls auf 0.

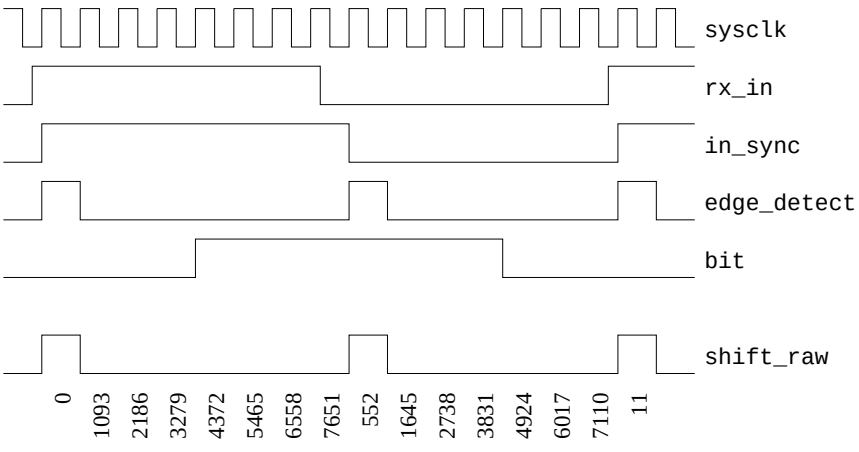


Abbildung 3.16: Impulsdiagramm der PLL

Abbildung 3.16 zeigt ein Beispiel der Abläufe in der PLL. Das Eingangssignal `tx_in` stammt von einem Sender, dessen Taktfrequenz etwa 6,7% höher als die des

Empfängers ist. Das Eingangssignal wird mit der Systemtaktfrequenz abgetastet (`in_sync`), nach jeder Flanke wird der Ausgang `edge_detect` aktiv. Der Ausgang `bit` entspricht dem Bit vor der Flanke. Der NCO wird so geregelt, daß der Bittakt `shift_raw` synchron zu `edge_detect` ist. In Abbildung 3.16 ist der Inhalt des Phasenakkumulators im eingeschwungenen Zustand für zwei Zyklen angegeben.

Das Schleifenfilter bestimmt wesentlich die dynamischen Eigneschaften der PLL. Das Schleifenfilter übernimmt bei jeder Flanke den Wert des Phasenfehlers und berechnet einen neuen Korrekturwert für den NCO. Dieser Korrekturwert bleibt bis zur nächsten Flanke gültig. Abbildung 3.17 zeigt die Struktur des Schleifenfilters.

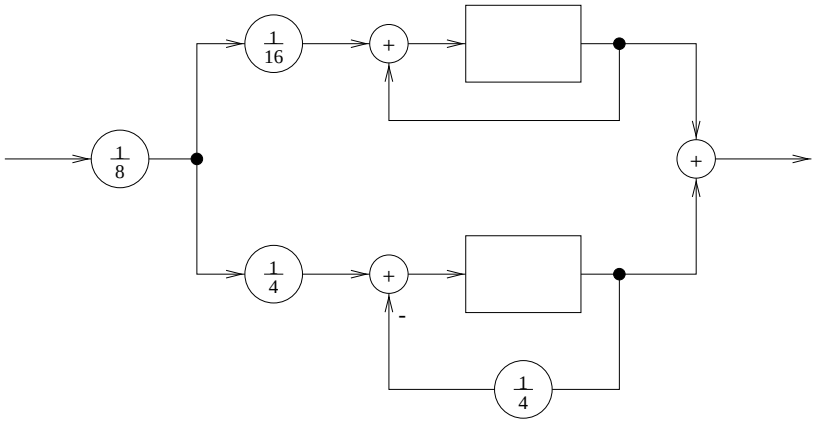


Abbildung 3.17: Schleifenfilter

Das Schleifenfilter besteht aus zwei Zweigen, die einen integrierenden und einen proportionalen Anteil des Phasenregelkreises bilden. Bei der Analyse dieser PLL versagen klassische Methoden der Regelungstechnik, da der Phasenfehler und ein Korrekturwert nur ermittelt wird, wenn eine Flanke auftritt. Der intergrierende Anteil ist so ausgelegt, daß dieser Anteil während der Synchronisationssequenz auf einen stabilen Wert einschwingt. Durch den proportionalen Anteil wird ein Phasenfehler in vier Bitperioden ausgeregelt, da im Normalbetrieb des BMCs spätestens nach vier Bitperioden eine Flanke auftritt. Der Faktor $\frac{1}{8}$ am Eingang des Schleifenfilters gleicht die Steilheit des NCOs aus. Um die Komplexität der Hardware des Schleifenfilters gering zu halten, werden als arithmetische Operationen nur Additionen und Multiplikationen mit Zweierpotenzen, die durch Bitschiebeoperationen realisert werden, eingesetzt.

Die Rahmensynchronisation erfolgt in zwei Schritten. Wenn im Eingangsdatenstrom 8 aufeinanderfolgende abwechselnde 0 und 1 Bits auftreten wird das Signal

carrier_detected aktiv. Dies wird im Bit 2 des Statusregisters der Mikrocontrollerschnittstelle angezeigt. Wenn nun innerhalb von 32 Bitperioden die 7 Bit Barker Folge im Eingangsdatenstrom erscheint, wird der BMC in den Empfangsbetrieb versetzt. Durch diese aufwendige Rahmensynchronisation in Verbindung mit der oben beschriebenen Flankenerkennung ist die Wahrscheinlichkeit, daß ein zufälliges Rauschereignis als Paketanfang erkannt wird, sehr gering.

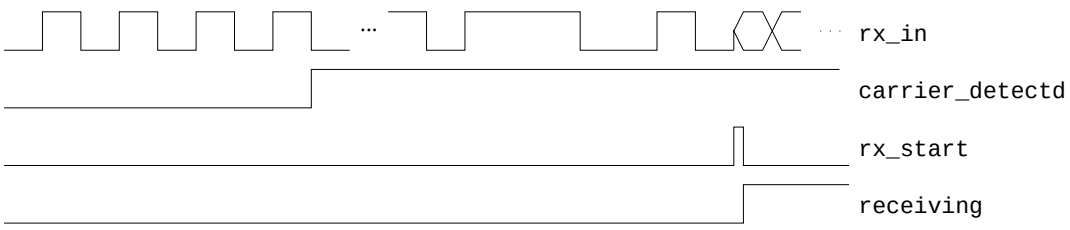


Abbildung 3.18: Rahmensynchronisation

Abbildung 3.18 verdeutlicht den Ablauf der Rahmensynchronisation. Der zeitliche Ablauf der Rahmensynchronisation wird vom Modul syncctrl gesteuert. Während des Empfangsbetriebs hat syncctrl die Aufgabe, die zur Bitsynchronisation eingefügten Füllbits zu verwerfen. Der Vergleich des Eingangsdatenstroms mit der Barker Folge erfolgt nicht innerhalb des sync Moduls, sondern im Modul decbuff, da dort das benötigte Schieberegister schon vorhanden ist.

3.2 FPGA Realisierung des BMCs

Die Verilog HDL Beschreibung des BMCs wurde mit Hilfe des Synopsys FPGA Compilers in ein FPGA vom Typ XC 4013E-4PQ240 der Firma Xilinx implementiert. Die verwendeten Befehlsdateien für den FPGA Compiler sind in Anhang B abgedruckt. Der FPGA Compiler erzeugt aus der Verilog HDL Beschreibung des BMCs eine Xilinx Netzlisten Datei, die mit Xilinx Werkzeugen weiterbearbeitet werden muß um eine Programmierdatei für das FPGA zu erhalten. Die Erstellung der Programmierdatei soll hier nicht weiter erläutert werden, da der Aufruf der Werkzeuge je nach Instalation unterschiedlich sein kann und es nicht Aufgabe einer Diplomarbeit ist, Programmaufrufe zu erklären. Bei Bedarf sei auf die entsprechende Dokumentation verwiesen.

Die FPGAs der XC 4000 Familie sind im eingebauten Zustand beliebig oft wiederprogrammierbar. Diese Eigenschaft macht sie für den Einsatz in Prototypen besonders geeignet.

Der Typ XC 4013E-4PQ240 wurde gewählt, da er genügend Kapazität für die Logik des BMCs besitzt und noch etwas Platz für zusätzliche Erweiterungen bleibt. Sollen gößere Erweiterungen hinzugefügt werden, stehen Xilinx FPGAs mit höherer Kapazität im gleichen Gehäuse zur Verfügung, so daß ein einmal erstelltes Platinenlayout weiterverwendet werden kann. Wenn der BMC seinen entgültigen Stand erreicht hat und in größerem Umfang eingesetzt werden soll, kann der FPGA Typ XC 4013E-4PQ160 eingesetzt werden, der ein deutlich kleineres Gehäuse besitzt. In dieser Gehäusevariante sind allerdings keine FPGAs mit höherer Kapazität erhältlich.

Neben den für die Funktion des BMCs unbedingt nötigen Signalen sind im FPGA auch einige interne Signale nach außen geführt. Diese sollen den Test und eine eventuelle Fehlersuche erleichtern. Tabelle 3.5 zeigt die Anschlußbelegung des FPGAs. Das gewählte FPGA besitzt 240 Anschlußpins, alle in Tabelle 3.5 nicht aufgeführten Pins sind nicht belegt.

Signal	**Pin**	**Beschreibung**	**Bemerkung**
`in_clk`	178	externer Takt	
`reset`	100	Rücksetzten	
`addr[0]`	43	Adresse 0	Mikrocontroller-schnittstelle
`addr[1]`	44	Adresse 1	
`data[0]`	23	Daten 0	
`data[1]`	24	Daten 1	
`data[2]`	25	Daten 2	
`data[3]`	26	Daten 3	
`data[4]`	27	Daten 4	
`data[5]`	28	Daten 5	
`data[6]`	31	Daten 6	
`data[7]`	32	Daten 7	
`cs0`	51	chip select low	
`cs1`	50	chip select high	
`rd`	57	Lesezugriff	
`wr`	239	Schreibzugriff	
`intr`	67	Interrupt	
`rx_in`	142	Eingang	Verbindung zum HF-Teil
`receiving`	96	Empfangsbetrieb	
`tx_out`	134	Ausgang	
`tx_pulse`	128	PWM Ausgang	

Tabelle 3.5: Anschlußbelegung des FPGAs

Signal	Pin	Beschreibung	Bemerkung
carrier_detected	35	Synchronisation	interne Signale für Testzwecke
transmitting	38	Sendebetrieb	
rx_error	33	Empfangsfehler	
crc_error	41	CRC Fehler	
tx_error	144	Sendefehler	
tx_start	141	Beginn Sendebetrieb	
tx_ready	139	Ende Sendebetrieb	
rx_ready	215	Ende Empfangsbetrieb	
full	48	Empfangspuffer voll	
empty	39	Sendepuffer leer	
bit	88	entschiedenes Bit	
shift_raw	47	Empfangsbittakt	
sys_clk	99	Systemtakt	
bit_clk	107	Sendebittakt	
M0	60	Mode 0	Programmierung des FPGAs
M1	58	Mode 1	
M2	62	Mode 2	
DIN	177	Konfigurationsdaten	
CCLK	179	Konfigurationstakt	
DONE	120	Konfiguration	
INIT	89	Konfiguration	
PROG	122	Konfiguration	
VDD	19, 30, 40, 61, 80, 90, 101, 121, 140, 150, 161 180, 201, 212, 222, 240		
GND	1, 14, 29, 45, 59, 75, 91, 106, 119, 135, 151 166, 182, 196, 211, 227		

Tabelle 3.5: Anschlußbelegung des FPGAs

Die Signale aus dem ersten Teil von Tabelle 3.5 wurden bereits in den Abschnitten 3.1.1 und 3.1.2 beschrieben, hier folgt eine Beschreibung der zusätzlichen Testsignale:

`carrier_deteced`: Dieses Signal wird High, wenn die Synchronisationsschaltung den Anfang einer Präambel erkennt.

`transmitting`: Dieses Signal zeigt an, daß der BMC sich im Sendebetrieb befindet.

`rx_error`, `crc_error` und `tx_error`: Diese Signale zeigen Empfangsfehler, CRC Fehler bzw. Sendefehler an.

`tx_start`, `tx_ready` und `rx_ready` werden zu Beginn des Sendebetriebs, zum Ende des Sendebetriebs bzw. zum Ende des Empfangsbetriebs aktiv.

`full` und `empty` zeigen den Zustand des Sendepuffers bzw. des Empfangspuffers der Mikrocontrollerschnittstelle an.

`bit`: Dies ist der Bit-Ausgang der Synchronisation bzw. das Eingangsignal des Decoders.

`shift_raw` ist der aus dem Empfangssignal gewonnene Bittakt.

`sys_clk` ist der interne Systemtakt des BMCs.

`bit_clk`: Der Sendebittakt.

Der Zustand der Signale `carrier_detected`, `transmitting`, `rx_error`, `crc_error`, `tx_error`, `full` und `empty` wird zwar auch im Statusregister der Mikrocontrollerschnittstelle angezeigt, die Ausgänge des FPGAs folgen den Zustandsänderungen der Signale aber ohne Verzögerung, was für Messungen unerläßlich ist.

Die Programmierung des FPGAs kann auf unterschiedliche Weise erfolgen, zum Beispiel über ein serielles PROM oder über eine Schnittstelle von einem PC. Die Art der Programmierung bestimmen die drei Eingänge `M0`, `M1` und `M2`. Für die Programmierung durch ein serielles PROM müssen diese Eingänge auf Low-Pegel liegen. Zur Programmierung über eine Schnittstelle vom PC müssen die Eingänge auf High-Pegel liegen. Die Datenübertragung während der Programmierung erfolgt über die Leitungen `DIN` und `CCLK`. Die Signale `DONE`, `INIT` und `PROG` dienen zur Steuerung der Datenübertragung. Erfolgt die Programmierung durch ein serielles PROM, so liest das FPGA dieses PROM beim Einschalten der Versogungsspannung aus.

Die Versorgungsspannungs- (`VDD`) und Masse- (`GND`) Anschlüsse müssen alle verbunden werden.

3.3 Testsystem für den BMC

Da der Burst Mode Controller durch einen Mikrocontroller angesteuert werden muß, wurde zum Test des BMCs eine Schaltung mit dem 8 Bit Mikrocontroller μPD 78054 der Firma NEC entwickelt.

3.3.1 Schaltungsbeschreibung des Testsystems

Die Schaltpläne des Testsystems für den BMC sind in Anhang C abgebildet. Die Schaltung gliedert sich in 3 Teile:

- Der Mikrocontroller mit externem RAM und einer seriellen Schnittstelle
- Das FPGA mit einem seriellen PROM und einer Schnittstelle zur Programmierung durch einen PC
- Einer Bitentscheidungsschaltung, die das analoge Ausgangssignal des Funkempfängers in ein Digitalsignal umwandelt. Die Bitentscheidungsschaltung wurde von Frank Mayer entwickelt.

Die Versorgungsspannung wird der Schaltung über die Buchsen J1 und J2 zugeführt und von einem integrierten Spannungsregler vom Typ 7805 (U1) stabilisiert. Ein TL 7705A (U2) erzeugt beim Einschalten der Versorgungsspannung ein Resetsignal. Durch den Taster SW1 kann manuell ein Reset ausgelöst werden.

Der Mikrocontroller μPD 78054 wurde gewählt, da für diesen Mikrocontrollertyp ein Entwicklungssystem mit C Compiler und In Circuit Emulator zur Verfügung steht.

Der μPD 78054 arbeitet mit einer Taktfrequenz von 5 MHz. Er enthält 32 kByte ROM und 1 kByte RAM. Als externer Speicher ist ein 8 kByte großes statisches RAM (U3) angeschlossen.

Da der μPD 78054 einen im Multiplexbetrieb arbeitenden Adress/Datenbus verwendet, müssen die unteren 8 Adressbits durch das Latch U6 zwischengespeichert werden um das RAM und den BMC ansteuern zu können. Das RAM belegt im Adressraum des μPD 78054 den Bereich von 8000_{hex} bis $9FFF_{\text{hex}}$, der BMC belegt die Adressen von $C000_{\text{hex}}$ bis $C003_{\text{hex}}$.

Der Mikrocontroller verfügt über eine bidirektionale asynchrone serielle Schnittstelle. Die Signale dieser Schnittstelle werden von einem MAX 232 auf RS 232 Pegel gebracht und sind auf die 9-polige Sub D Buchse CON1 geführt. Diese Buchse ist so beschaltet, das eine direkte Verbindung mit der seriellen Schnittstelle eines PCs möglich ist. Tabelle 3.6 zeigt die Belegung der Buchse CON1. Über die Leitung TxD sendet der Mikrocontroller Daten, über RxD empfängt er Daten. Die Verbindungen der Pins 4 und 6 bzw. 7 und 8 sind notwendig, wenn die Gegenstelle mit Hardware-Handshake arbeitet.

Die digitalen Ein-/Ausgabeports des μPD 78054 sind auf die Stiftleisten J3 bis J7 geführt. Die analogen Ein- und Ausgänge stehen an J8 zur Verfügung. Die

Pin	Signal	Pin	Signal
1	offen	2	TxD
3	RxD	4	mit 6 verbunden
5	GND	6	mit 4 verbunden
7	mit 8 verbunden	8	mit 7 verbunden
9	offen		

Tabelle 3.6: Serielle Schnittstelle

Pinbelegung der Stiftleisten J3 bis J8 wird hier nicht wiedergegeben, sie kann bei Bedarf dem Schaltplan entnommen werden. Zur genauen Funktionsbeschreibung der verschiedenen Ein-/Ausgabeports sei auf [10] verwiesen.

Der BMC ist über die Signale des Mikrocontrollerinterfaces (siehe 3.1.2) an den Mikrocontrollerbus angeschlossen. Über die Stiftleiste J10 sind verschiedene interne Signale des BMCs zugänglich. Die Verbindung mit dem HF-Transceiver erfolgt durch die Stiftleiste J16. Tabelle 3.7 erläutert die Belegung der Stiftleiste J16.

Pin	Beschreibung
1	+5 V
2	Eingang für das analoge Ausgangssignal des Funkempfängers
3	Eingang für ein digitales Ausgangssignal des Funkempfängers
4	Dieses Signal wird High, wenn der BMC sich im Empfangsbetrieb befindet. Es kann verwendet werden, um die Zeitkonstante einer Bitentscheidungsschaltung umzuschalten.
5	Der Port P120 des Mikrocontrollers dient zur Umschaltung des Funktransceivers zwischen Sende- und Empfangsbetrieb.
6	Der Ausgang des BMCs zur Ansteuerung des Senders. Dieses Signal ist rechteckförmig und hat TTL-Pegel.
7	Der PWM Ausgang des BMCs wird über ein Tiefpaßfilter auf diesen Anschluß geführt.
8	GND

Tabelle 3.7: Verbindung zum Funktransceiver

Über die Stiftleiste J15 kann gewählt werden, ob das analoge oder das digitale Ausgangssignal des Funktranceivers dem BMC zugeführt wird. Das Tiefpaßfilter bestehend aus R9 und C19 ist mit einer Grenzfrequenz von 20 kHz für eine

HF Bitrate von 40 kbps ausgelegt.

Der externe Takt des BMCs kann über die Stiftleiste J12 von verschiedenen Quellen gewählt werden. Zum einen von dem 10 MHz Quarzoszillator X2, zum anderen von drei unterschiedlichen Ausgängen des Mikrocontrollers, die je ein Taktsignal einstellbarer Frequenz erzeugen können. Ebenso kann der Interruptausgang des BMCs durch die Stiftleiste J11 auf vier verschiedene Interrupteingänge des Mikrocontrollers gelegt werden.

Die Programmierung des FPGAs kann entweder durch das PROM U8 oder über die Stiftleiste J13 durch eine Schnittstelle von einem PC aus erfolgen. Soll die Programmierung durch das PROM erfolgen, so sind die Anschlüsse 1 und 2, 3 und 4, 5 und 6 der Stiftleiste J9 zu verbinden. Bei einer Programmierung durch eine Schnittstelle vom PC müssen diese Verbindungen geöffnet sein. Die Stiftleiste J13 ist so belegt, daß eine 1 zu 1 Verbindung zu der Xilinx Xchecker Schnittstelle möglich ist. Das FPGA kann so programmiert werden, daß ein Auslesen der Registerinhalte und ein Boundary Scan Test durchgeführt werden kann. Durch die Stiftleiste J14 können die entsprechenden Verbindungen zur Xchecker Schnittstelle hergestellt werden. Die aktuelle Version des BMCs macht jedoch keinen Gebrauch von diesen Möglichkeiten.

Die Bitentscheidungsschaltung hat die Aufgabe aus dem analogen Ausgangssignal des Funkempfängers ein Digitalsignal zu erzeugen. Hierzu wird der obere und untere Spitzenwert des analogen Signals gemessen. Die Spitzenwerte werden in Kondensatoren gespeichert. Die Entladewiderstände der Kondensatoren und damit die Zeitkonstante der Bitentscheidungsschaltung sind umschaltbar. Zur Umwandlung in ein Digitalsignal wird das analoge Signal mit dem Mittelwert aus oberem und unterem Spitzenwert verglichen.

Im Normalbetrieb arbeitet die Bitentscheidungsschaltung mit einer Zeitkonstanten von 820 μs, so daß sich die Schaltung schnell an unterschiedliche Eingangssignale anpassen kann. Erkennt der BMC eine Präambel, so wird die Zeitkonstante durch das Signal `receiving` auf 4,7 ms umgeschaltet, damit sich kurzzeitige Störungen nicht auswirken können.

3.3.2 Hinweise zur Programmierung des Mikrocontrollers

Obwohl die Programmierung des Mikrocontrollers nicht Gegenstand dieser Diplomarbeit ist, sollen hier doch einige Hinweise zur Ansteuerung des BMCs durch den Mikrocontroller gegeben werden.

Die Kommunikation des BMCs mit dem Mikrocontroller erfolgt interruptgesteuert. Nach jedem Interrupt muß der Mikrocontroller das Statusregister des BMCs

auslesen um den Interrupt zurückzusetzen und den Grund für der Interrupt zu erfahren. Die Bits 3 und 4 des Statusregisters zeigen an in welchem Betriebszustand der BMC sich befindet (siehe Abschnitt 3.1.2, Tabelle 3.3). An Hand des Betriebszustandes kann der weitere Programmablauf gesteuert werden.

Vor dem Senden eines Pakets sollte das Paket im Speicher aufgebaut werden. Dabei ist besonders das erste und das siebte Byte des Pakets korrekt zu bestimmen. Ist das höchstwertige Bit des ersten Bytes gleich Null, so wird ein 6 Byte langes Paket gesendet, das nur Steuerinformationen für das Funknetzwerk enthält. Ist das höchstwertige Bit des ersten Bytes gleich Eins, so gibt das siebte Byte des Pakets an, wie viele Datenbytes folgen. Der BMC bestimmt an Hand dieser Daten die Länge des Pakets und steuert seine interne Ablaufsteuerung entsprechend.

Das Senden eines Pakets wird durch das Schreiben des ersten Bytes des Pakets in das Datenregister des BMCs begonnen. Vorher muß der Funktransceiver auf Sendebetrieb umgeschaltet werden. Es sollte auch geprüft werden, ob gerade ein Paket empfangen wird. Der Mikrocontroller kann den Empfangsbetrieb jedoch auch abbrechen. Der weitere Ablauf erfolgt nun interruptgesteuert. Nachdem der BMC ein Byte verarbeitet hat, wird ein Interrupt ausgelöst. Der BMC befindet sich im Sendebetrieb und sein Sendpuffer ist leer. Der Mikrocontroller muß nun das nächste Byte in das Datenregister schreiben. Auch nach der Verarbeitung des letzten Bytes eines Pakets löst der BMC einen Interrupt aus, der Mikrocontroller schreibt jetzt jedoch nicht mehr in das Datenregister.

Beim Empfang eines Pakets löst der BMC jeweils einen Interrupt aus, wenn ein Byte vollständig empfangen wurde. Der BMC befindet sich im Empfangsbetrieb und sein Empfangspuffer ist voll. Der BMC muß das empfangene Byte aus dem Datenregister lesen.

Nachdem der BMC ein Paket vollständig, d. h. inklusive CRC Wert, gesendet bzw. empfangen hat und aus dem Sendebetrieb bzw. Empfangsbetrieb in den Leerlaufzustand zurückkehrt, löst der BMC einen weiteren Interrupt aus. Abhängig vom vorhergehenden Betriebszustand kann der Mikrocontroller jetzt prüfen ob ein Sendefehler bzw. ein Empfangsfehler oder CRC Fehler aufgetreten ist.

Das Struktogramm in Abblidung 3.19 verdeutlicht den Ablauf einer Interruptbearbeitung durch den Mikrocontroller. Die Betriebszustände Sendebetrieb, Empfangsbetrieb und Leerlauf des BMCs sind abkürzend mit RX, TX und IDLE bezeichnet.

Abschließend soll noch die Bedeutung der möglichen Fehlerzustände des BMCs für das Mikrocontrollerprogramm und das Zeitverhalten des BMCs erläutert werden.

Ein Sendefehler trit im Sendebetrieb auf, wenn der Mikrocontroller nach einem

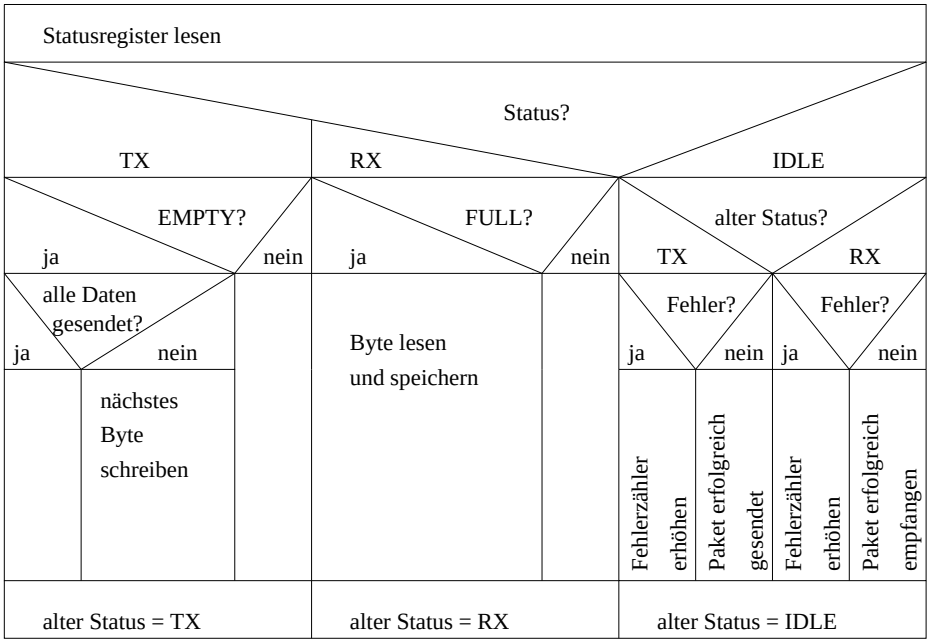


Abbildung 3.19: Interruptbearbeitung

Interrupt nicht rechtzeitig eine neues Byte an den BMC schickt. Bei einer Datenrate von 20 kbps stehen dafür 400 μs zur Verfügung, bei einer Datenrate von 40 kbps entsprechend 200 μs. Trit ein Sendefehler auf, so bricht der BMC die Übertragung ab.

Ein Empfangsfehler trit analog zum Sendefehler im Empfangsbetrieb auf. Auch in diesem Fall stehen bei 20 kbps 400 μs zur Verfügung um das empfangenge Byte nach einem Interrupt zu lesen. Bei einem Empfangsfehler wird der Empfang jedoch nicht abgebrochen.

Ein CRC Fehler bedeutet, daß der empfangene CRC Wert nicht mit dem CRC Wert übereinstimmt, den der BMC aus den empfangengen Daten errechnet hat. Das empfangene Paket ist in diesem Fall fehlerhaft und muß verworfen werden.

Die Gesamtdauer der Übertragung eines Datenpakets ist abhängig von der Länge des Pakets, der Bitrate und der Anzahl der zusätzlich zur Synchronisation eingefügten Bits.

Der BMC codiert jeweils zwei Datenbytes in ein 31 Bit langes Codewort, hinzu kommen zwei Bytes für den CRC Wert. Die Anzahl der Bits, die sich hier ergibt, muß um die Synchronisationsbits während der Übertragung und die Synchronisationssequenz ergänzt werden. Es ergibt sich:

$$b = 32 + \left\lceil \frac{n+2}{2} \right\rceil \frac{s+1}{s}$$

Dabei ist:

- b: Anzahl der übertragenen Bits
- n: Anzahl der übertragenen Bytes
- s: Inhalt des Registers 3 des BMCs

Die Anzahl der Bits muß mit der Dauer eines Bits multipliziert werden um die Dauer der Übertragung eines Pakets zu bestimmen. Tabelle 3.8 zeigt die Übertragungszeit verschieden großer Pakete bei unterschiedlichen Bitraten. Die Kenntnis der Übertragungszeit eines Pakets ist wichtig für die Entwicklung eines Kanalzuteilungsalgorithmus.

	kurzes Paket (6 Byte)	**langes Paket** (262 Byte)
Codeworte	4	132
Bits gesammt	187	5147
Zeit (20 kbps)	9,35 ms	257 ms
Zeit (40 kbps)	4,68 ms	129 ms

Tabelle 3.8: Übertragungszeit

4. Ergebnisse

4.1 Logiksynthese des BMCs

In diesem Abschnitt werden die Ergebnisse der Logiksynthese zur Implementierung des BMCs in ein FPGA vom Typ XC 4013E-4PQ240 dargestellt.

In den FPGAs der XC 4000 Familie werden logosche Funktionen mit sogenannten CLBs realisiert [18]. Der Baustein XC 4013 verfügt über 576 solcher CLBs. Von diesen werden 98% für die Implementierung des BMCs verwendet. Es muß jedoch beachtet werden, daß dabei etwa 25% der CLBs zur Realisierung von Verbindungen zwischen anderen CLBs herangezogen werden und keine logische Funktion ausführen. Tabelle 4.1 zeigt welchen Anteil einige wichtige Teilsysteme des BMCs an der gesammten Implemantierung des BMCs haben.

Teilschaltung	**CLBs**
Mikrocontrollerinterface	6,4%
Encoder	7,8%
Synchronisation	21,6%
Decoder	39,3%

Tabelle 4.1: Flächenanteil von Teilsystemen

Zum Vergleich wurde eine Logiksynthese des BMCs auch mit einer ASIC Technologie mit einer Strukturbreite von 0,8 μm und zwei Metallisierungslagen durchgeführt. Diese Synthese ergibt einen Flächenbedarf von 2,16 mm^2 und eine Komplexität von etwa 5000 Gatteräquivalenten, allerdings ohne Ein- und Ausgangszellen. Dieses Ergebnis relativiert die Angabe von Xilinx, daß das FPGA XC 4013 eine Komplexität von 13000 Gatteräquivalenten besitzt.

Tabelle 4.2 zeigt die maximal möglichen Taktfrequenzen für die Implementierung des BMCs im XC 4013.

externe Taktfrequenz	33 MHz
interne Taktfrequenz	10 MHz
Bitrate	1,25 Mbps

Tabelle 4.2: Geschwindigkeit des BMCs

4.2 Übertragungsversuche

Das in Abschnitt 3.3 beschriebene Testsystem für den BMC wurde dreimal aufgebaut. Die Stromaufnahme des Systems beträgt 65 mA. Die Funktion des Systems und des BMCs hat sich als stabil und reproduzierbar erwiesen.

Maximilian Hofmann hat im Rahmen seiner Diplomarbeit eine vorläufige Software zur Ansteuerung des BMCs entwickelt. Die Software ermöglicht das Lesen und Verändern der Registerinhalte des BMCs sowie das Senden und Empfangen von Datenpaketen. Die Steuerung dieser Software erfolgt durch ein Terminal, das an der seriellen Schnittstelle des Testsystems angeschlossen wird.

Als Funktransceiver wurde ein Transceivermodul eingesetzt, das für die Anwendung in einem LON Netzwerk entwickelt wurde. Dieses Transceivermodul arbeitet auf einer Frequenz von 433,92 Mhz. Die Senderausgangsleistung beträgt 10 mW.

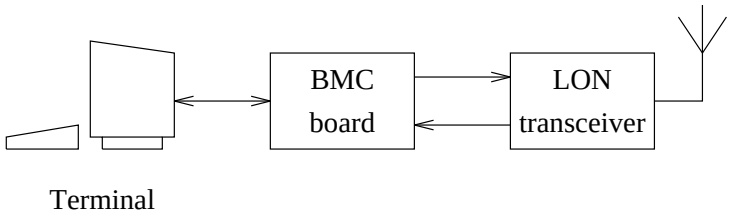


Abbildung 4.1: Versuchsaufbau

Abbildung 4.1 zeigt den Versuchsaufbau. Der Sender wird dabei mit dem TTL Ausgangssignal des BMCs angesteuert, da sich herausgestellt hat, daß die Erzeugung eines annähernd gaußförmigen Signals nicht ausreichend gut gelingt. Da der Systemtakt nur eine Frequenz der achtfachen Bitrate hat, ist es nicht möglich, eine Bitperiode in hinreichend kleine Zeitabschnitte zu unterteilen um die Gaußform durch Pulsweitenmodulation oder Deltamodulation zu erzeugen.

Mit zwei der in Abbildung 4.1 gezeigten Versuchsaufbauten konnte die Funktionsfähigkeit der Datenübertragung zwischen zwei Büros am IIS und über einige

hundert Meter im Freien nachgewiesen werden. Aussagekräftige Messungen der Fehlerrate der Übertragung konnten wegen der vorläufigen Software, die noch keine Fehlerstatistik führt, nicht durchgeführt werden.

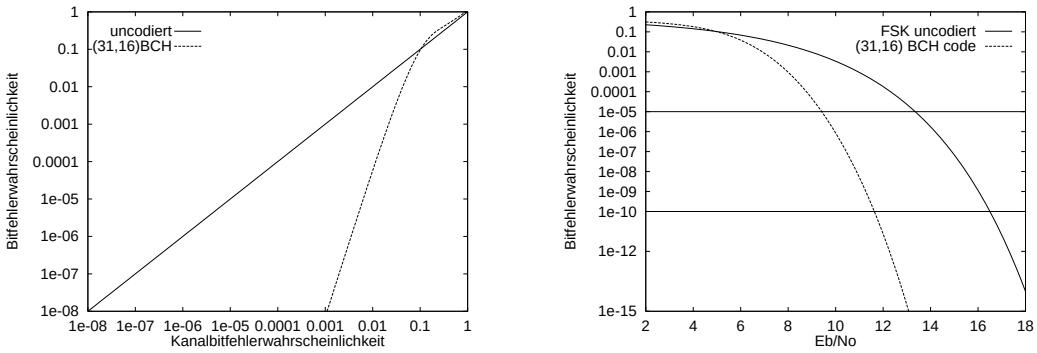
4.3 Theoretische Leistung des (31,16)BCH Codes

Da noch nicht genügend Meßergebnisse zur Beurteilung der Übertragungssicherheit einer mit dem BMC realisierten Datenübertragung vorliegen, wird hier kurz die theoretische Leistungsfähigkeit des (31,16)BCH Codes dargestellt.

Der (31,16)BCH Code kann pro Codewort drei Bitfehler korrigieren. Treten pro Codewort mehr als drei Fehler auf, so fügt der Decoder im ungünstigsten Fall drei weitere Fehler hinzu. Die Bitfehlerwahrscheinlichkeit P_B in den decodierten Daten beträgt somit in Abhängigkeit von der Kanalbitfehlerwahrscheinlichkeit P_K:

$$P_B = \sum_{i=4}^{31} \min(1, \frac{i+3}{31}) \binom{31}{i} P_K^i (1-P_K)^{31-i}$$

Diese Abhängigkeit ist in Abbildung 4.2a graphisch dargestellt.



Abbildungen 4.2a,b: Leistungfähigkeit des (31,16)BCH Codes

Die Kanalbitfehlerwahrscheinlichkeit bei Frequenzumtastung als Modulationsverfahren ist, [12]:

$$P_K = \frac{1}{2} e^{-\frac{1}{2}\frac{E_b}{N_0}}$$

Die Abhängigheit der Bitfehlerwahrscheinlichkeit von $\frac{E_b}{N_0}$ ist in Abbildung 4.2b dargestellt. Beim Vergleich von codierter und uncodierter Übertragung ist zu beachten, daß der Durchsatz bei codierter Übertragung um die Coderate geringer ist als bei uncodierter Übertragung.

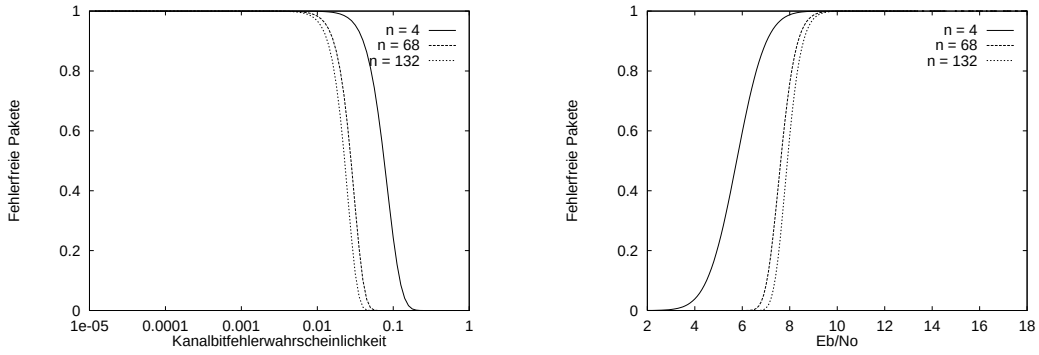
Wichtiger als die Bitfehlerwahrscheinlichkeit ist die Wahrscheinlichkeit, mit der ein komplettes Paket fehlerfrei empfangen wird. Ein Paket besteht aus mindestens 4 und maximal 132 Codeworten. Die Wahrscheinlichkeit, mit der ein Codewort fehlerfrei empfangen wird ist:

$$P_C = \sum_{i=0}^{3} \binom{31}{i} P_K^i (1 - P_K)^{31-i}$$

Für ein Paket git:

$$P_P = P_C^n, \qquad n = 4, 5, \cdots, 132.$$

Abbildung 4.3 zeigt die Wahrscheinlichkeit, daß ein Paket fehlerfrei empfangen wird, in Abhängigkeit von der Kanalbitfehlerwahrscheinlichkeit bzw. $\frac{E_b}{N_0}$ für verschiedene Paketgrößen.



Abbildungen 4.3a,b: Wahrscheinlichkeit für den Empfang fehlerfreier Pakete

Die hier angestellten Überlegungen gelten nur für Kanäle mit zufällig verteilten Fehlern. Solche Fehlerstrukturen sind typisch für Funkstrecken mit fest installierten Sendern und Empfängern ohne Hindernisse zwischen Sender und Empfänger.

Das Funknetzwerk, für das der BMC entwickelt wurde, soll auch innerhalb von Gebäuden und mit mobilen Stationen eingesetzt werden. Hier kann es zu zusätzlichen Störungen durch Reflektionen an Wänden oder durch Fading im Mobilbetrieb kommen. Die theoretische Betrachtung der Übertragungssicherheit über

solche Kanäle ist sehr schwierig und von der genauen Kenntnis der Fehlerstatistik des jeweiligen Kanals abhängig. Die Übertragungssicherheit über solche Kanäle muß also durch weitere Versuche ermittelt werden.

4.4 Synchronisationsverhalten

Die Synchronisation spielt in jedem Datenübertragungssystm eine sehr wichtige Rolle, da eine Decodierung nur bei Kenntnis des zeitlichen Ablaufs des empfangenen Datenstroms möglich ist.

Abbildung 4.4 zeigt die Fähigkeit des BMCs Datenpakete zu empfangen, die von Sendern stammen, deren Bittaktfrequenz von der des Empfängers abweicht.

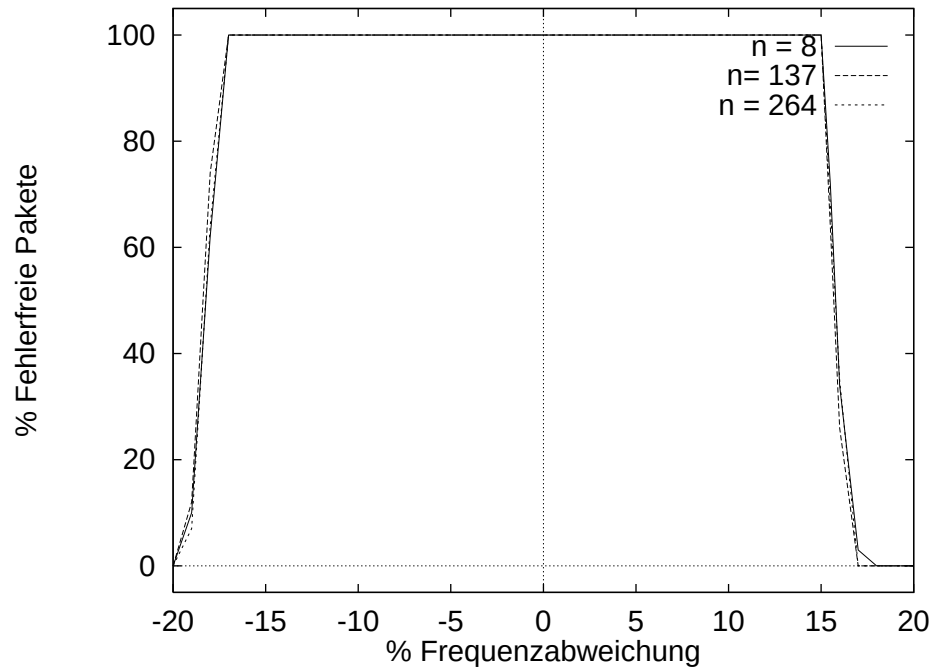


Abbildung 4.4: Synchronisationsverhalten

In Abbildung 4.4 ist der Anteil der fehlerfrei empfangenen Pakete in Abhängigkeit von der Frequenzabweichung zwischen Sender und Empfänger für verschiedene Paketlängen dargestellt. Für diese Messung wurden zwei BMC Testsysteme direkt ohne zwischengeschaltete Funkstrecke verbunden. Die BMCs arbeiten dabei mit den Normaleinstellungen, d. h. daß nach jedem vierten Bit eine Flanke zur Synchronisation in den Datenstrom eingefügt wird.

Wenn die Anzahl der Bits nach denen eine Flanke zur Synchronisation eingefügt vergrößert wird, zeigt es sich, daß eine zuverlässige Synchronisation nur bis zu einem Wert von 6 möglich ist. Bei größeren Werten wird die Synchronisation abhängig von den übertragenen Daten. Pakete, die lange Folgen von 0 oder 1 Bits enthalten, können nicht mehr zuverlässig empfangen werden.

Abbildung 4.5 zeigt das Verhalten des Schleifenfilters, wenn ein Datenpaket empfangen wird, dessen Bittaktfrequenz 8% höher als die des Empfängers ist. Diese Darstellung ist Ergebnis einer Simulation.

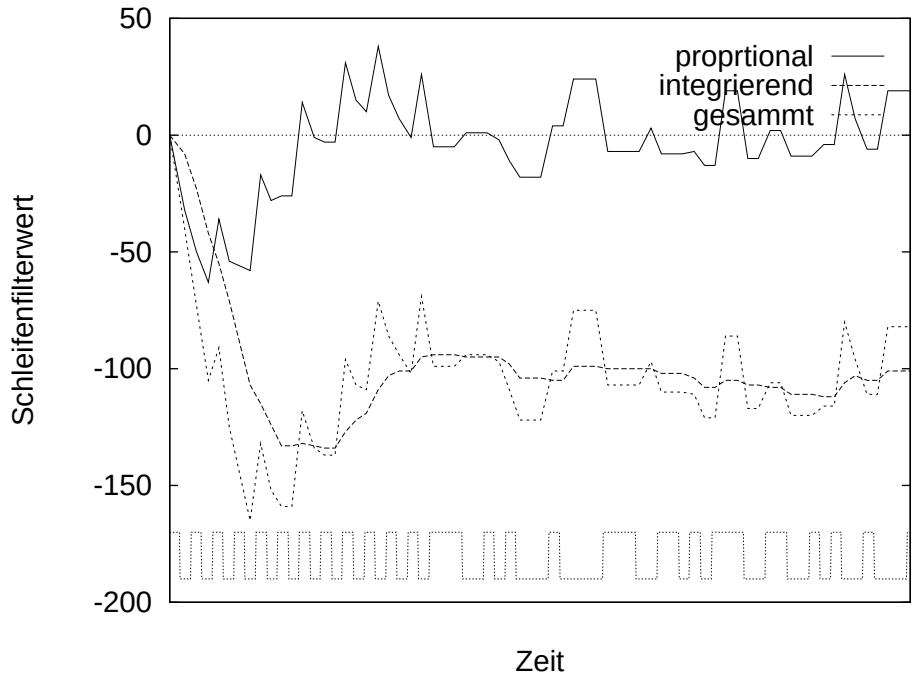


Abbildung 4.5: Verhalten des Schleifenfilters der PLL

Im unteren Teil von Abbildung 4.5 ist das empfangene Datenpaket dargestellt. Man erkennt, daß der integrierende Anteil während der Synchronisationssequenz einschwingt und daß der proportionale Anteil dann um einen Mittelwert von 0 schwankt.

Die Untersuchung der Rahmensynchronisation hat ergeben, daß mit dem verwendeten Transceiver und Bitentscheider pro Stunde durchschnittlich 20 bis 30 Pakete auf Grund von zufälligen Rauschereignissen erkannt werden.

5. Zusammenfassung und Ausblick

Im Rahmen dieser Diplomarbeit wurde ein Burst Mode Controller entwickelt, der die Kanalcodierung und Fehlersicherung in einem Funknetz durchführt. Der Burst Mode Controller wurde in der Hardwarebeschreibungssprache Verilog HDL entwickelt und in ein FPGA vom Typ XC 4013 implementiert. Die Funktionsfähigkeit des BMCs wurde durch Funkübertragungen demonstriert.

Die Steuerung des BMCs erfolgt durch einen Mikrocontroller. Die Mikrocontrollerschnittstelle des BMCs ist kompatibel zu den weit verbreiteten Controllern der 8051 Familie und zu der 78k Familie von NEC.

Im BMC wird ein (31,16)BCH Code zur Fehlerkorrektur und ein CRC Code zur Fehlererkennung eingesetzt. Der (31,16)BCH Code erreicht einen Codegewinn von 3,9 dB bei einer Bitfehlerrate von 10^{-5}.

Zur Synchronisation im Empfangsbetrieb wird eine leistungsfähige volldigitale PLL verwendet.

Für die Weiterentwicklung des BMCs sind insbesondere folgende Punkte von Interesse:

- Entwicklung eines besseren Konzeptes zur Sendeimpulsformung.
- Optimierung der Synchronisation durch genaue Analyse der PLL und Verbesserung des Schleifenfilters.
- Realisierung des BMCs als ASIC statt als FPGA. In ein ASIC kann auch ein Mikrocontroller mitintegriert werden, wodurch ein sehr kompakter Aufbau des Funkmoduls möglich würde. Die Implementierung des BMCs in ein ASIC ist durch die technologieunabhängige Verilog HDL Beschreibung mit geringem Aufwand möglich.
- Für Anwendungen mit geringen Anforderungen an die Übertragungssicherheit ist eine Version des BMCs ohne oder mit vereinfachter Fehlerkorrekturcodierung, z. B. einem (7,4)Hamming Code, denkbar. Die Komplexität des BMCs würde sich dadurch erheblich verringern.

Die weitere Entwicklung des Funknetzwerkes wird sich auf die Software konzentrieren. Hier sind die folgenden beiden Entwicklungsschritte geplant:

- Zunächst müssen Strategien zur Kanalzuteilung für die einzelnen Teilnehmer am Funknetz und zur Übertragungswiederholung bei fehlerhaftem Empfang entwickelt werden. Diese Aufgabe wird bereits im Rahmen einer Diplomarbeit bearbeitet.
- In einem zweiten Schritt soll das Funknetzwerk um Routing-Funktionen erweitert werden, d. h. Datenpakete können über Zwischenstationen zwischen Funknetzteilnehmern ausgetauscht werden, die sich gegenseitig nicht direkt erreichen können.

An dieser Stelle möchte ich mich bei meinen Betreuern Reiner Perthold und Hans Hauer sowie bei Frank Mayer und Maximilian Hofmann für die gute Unterstützung und Zusammenarbeit bedanken.

Literaturverzeichnis

[1] Berlekamp, E. R.: *Algebraic Coding Theory.* Aegan Park Press, Laguna Hills, 1968.

[2] Best, R.: *Theorie und Anwendungen des Phase-locked Loop.* AT-Verlag, 4. Aufl., Aarau, 1987.

[3] Blahut, R. E.: *Theorie and Practice of Error Control Codes.* Addison-Wesley, Reading, 1983.

[4] Bossert, M.: *Kanalcodierung.* Teubner, Stuttgart, 1992.

[5] Burton, H.: Inversionless Decoding of Binary BCH Codes. In *IEEE Transactions on Information Theory*, Vol. IT-17, No. 4, Juli 1971, Seite 464–466.

[6] Chien, R. T.: Cyclic Decoding Procedure for the Bose-Chaudhuri-Hocquenghem Codes. In *IEEE Transactions on Information Theory*, Vol. IT-10, No. 5, Oktober 1964, Seite 357–363.

[7] Deliano, R.: Auf Fehlersuche. In *Elrad*, Heft 2, Februar 1996, Seite 30–33.

[8] Friedrichs, B.: *Kanalkodierung: Grundlagen und Anwendungen in modernen Kommunikationssystemen.* Springer, Berlin Heidelberg New York, 1995.

[9] Lin, S. und Costello, D. J.: *Error Controll Coding.* Prentice-Hall, Englewood Cliffs, 1983.

[10] NEC: *Users's Manual μPD78054, 78054Y Subseries 8-Bit Single-Chip Microcontroller*, 1995.

[11] Massey, J. L.: Shift-Register Synthesis and BCD Decoding. In *IEEE Transactions on Information Theory*, Vol. IT-15, No. 1, Januar 1969, Seite 122–127.

[12] Sklar, B.: *Digital Communications.* Prentice-Hall, Englewood Cliffs, 1988.

[13] SWEENEY, P.: *Codierung zur Fehlererkennung und Fehlerkorrektur.* Hanser, München, 1992.

[14] THOMAS, D. E. UND MOORBY, P. R.: *The Verilog Hardware Description Language.* Kluwer Academic Publishers, 2. Aufl., Boston Dordrecht London, 1994.

[15] VITERBI, A. J.: Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. In *IEEE Transactions on Information Theory*, Vol. IT-13, No. 2, April 1967, Seite 260–269.

[16] VITERBI, A. J.: Convolutional Codes and Their Performance in Communication Systems. In *IEEE Transactions on Communications Technology*, Vol. COM-19, No. 5, October 1971, Seite 751–771.

[17] WICKER, S.: *Error Control Systems for Digital Communication and Storage.* Prentice-Hall, Englewood Cliffs, 1995.

[18] Xilinx: *XC4000 Series Field Programmable Gate Arrays Product Specification.* San Jose, 1996.

Anhang

A. Arithmetik über Galoisfeldern

Ein Galoisfeld ist eine Menge mit q Elementen, zwischen denen zwei Rechenoperationen definiert sind und die die mathematische Struktur eines Körpers bilden. In jedem Grundlagenwerk zur Codierungstheorie findet sich ein ausführliches Kapitel zu Galoisfeldern.

Im Rahmen dieser Diplomarbeit sind nur die Galoisfelder GF(2) und GF(32) von Bedeutung. GF(2) enthält die Elemente $\{0, 1\}$, die Operationen $+$ und $\cdot$ sind wie folgt definiert:

$+$	0	1
0	0	1
1	1	0

$\cdot$	0	1
0	0	0
1	0	1

GF(32) wird aus dem primitiven Element α und dem Minimalpolynom $1 + \alpha^2 + \alpha^5$ konstruiert. Es sind zwei verschiedene Darstellungen von GF(32) möglich: Exponential- und Polynomform. Tabelle A.1 zeigt die Darstellungen von GF(32).

Tabelle A.1: Darstellung von GF(32)

Exponentialform	**Polynomform**	**Bitmuster**
0	0	00000
α^0	1	00001
α^1	α	00010
α^2	α^2	00100
α^3	α^3	01000
α^4	α^4	10000
α^5	$\alpha^2 + 1$	00101
α^6	$\alpha^3 + \alpha$	01010
α^7	$\alpha^4 + \alpha^2$	10100
α^8	$\alpha^3 + \alpha^2 + 1$	01101
α^9	$\alpha^4 + \alpha^3 + \alpha$	11010

Tabelle A.1: Darstellung von GF(32)

Exponentialform	**Polynomform**	**Bitmuster**
α^{10}	$\alpha^4 + 1$	10001
α^{11}	$\alpha^2 + \alpha + 1$	00111
α^{12}	$\alpha^3 + \alpha^2 + \alpha$	01110
α^{13}	$\alpha^4 + \alpha^3 + \alpha^2$	11100
α^{14}	$\alpha^4 + \alpha^3 + \alpha^2 + 1$	11101
α^{15}	$\alpha^4 + \alpha^3 + \alpha^2 + \alpha + 1$	11111
α^{16}	$\alpha^4 + \alpha^3 + \alpha + 1$	11011
α^{17}	$\alpha^4 + \alpha + 1$	10011
α^{18}	$\alpha + 1$	00011
α^{19}	$\alpha^2 + \alpha$	00110
α^{20}	$\alpha^3 + \alpha^2$	01100
α^{21}	$\alpha^4 + \alpha^3$	11000
α^{22}	$\alpha^4 + \alpha^2 + 1$	10101
α^{23}	$\alpha^3 + \alpha^2 + \alpha + 1$	01111
α^{24}	$\alpha^4 + \alpha^3 + \alpha^2 + \alpha$	11110
α^{25}	$\alpha^4 + \alpha^3 + 1$	11001
α^{26}	$\alpha^4 + \alpha^2 + \alpha + 1$	10111
α^{27}	$\alpha^3 + \alpha + 1$	01011
α^{28}	$\alpha^4 + \alpha^2 + \alpha$	10110
α^{29}	$\alpha^3 + 1$	01001
α^{30}	$\alpha^4 + \alpha$	10010

Die Addition in GF(32) wird mit der Polynomform ausgeführt. Es gilt:

$$\begin{aligned} a &= a_0 + a_1\alpha + a_2\alpha^2 + a_3\alpha^3 + a_4\alpha^4 \\ b &= b_0 + b_1\alpha + b_2\alpha^2 + b_3\alpha^3 + b_4\alpha^4 \\ a + b &= (a_0 + b_0) + (a_1 + b_1)\alpha + (a_2 + b_2)\alpha^2 + (a_3 + b_3)\alpha^3 + (a_4 + b_4)\alpha^4 \\ a_i, b_j &\in GF(2) \end{aligned}$$

Die Multiplikation in GF(32) wird mit der Exponentialform durchgeführt. Es gilt:

$$\begin{aligned} a &= \alpha^i \\ b &= \alpha^j \\ a \cdot b &= \alpha^{(i+j) \bmod 31} \end{aligned}$$