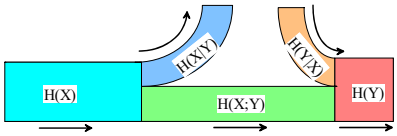
HOCHSCHULE DÜSSELDORF

FACHBEREICH
ELEKTRO- UND INFORMATIONSTECHNIK

Praktikum

Nachrichtencodierung



Versuch 5:

Lineare Blockcodes und ihre Eigenschaften bei Verwendung von AWGN-Kanälen

Prof. Dr. P. Pogatzki
Dipl.-Ing. D. Spengler

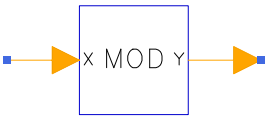
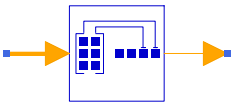
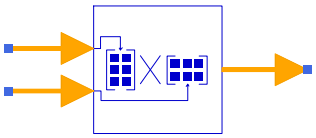
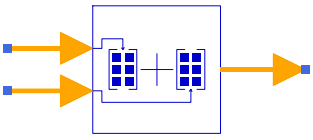
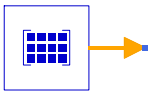
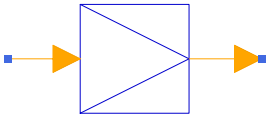
Name: Matr.-Nr.:

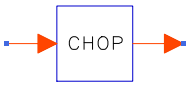
Testat:

1. Einleitung

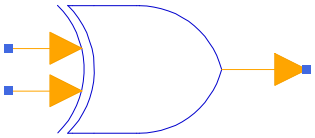
Ergänzend zur Vorlesung und Übung sollen in diesem Praktikumsversuch die Eigenschaften Linearer Blockcodes (Fehlererkennung/Fehlerkorrektur) bei Verwendung von AWGN-Kanälen untersucht werden. Es wird ein (7,4)-Hamming-Code verwendet.
Da die Kanalcodierung mit Linearen Blockcodes auf Vektor- und Matrix-Operationen beruht, werden in der Simulation mit ADS neben den Standard-Elementen einige Bauteile der Palette *Numeric Matrix* eingesetzt. Da die mathematischen Operationen im GF(2) durchgeführt werden, erfolgen nur Integer-Operationen mit anschließender Modulo-2-Rechnung!
Zusätzlich zu den Ihnen bekannten ADS-Elementen werden für die Simulation folgende Bauteile benötigt **(weitere Erläuterungen bzgl. der Bauteile: siehe ADS-Help)**:

http://edocs.soco.agilent.com/display/ads2008U1/Numeric+Components

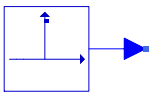
Schematic-Symbol	Schematic-Symbol
 PackInt_M P1 NumRows=1 NumCols=7	 ModuloInt M3 Modulo=2
 UnPkInt_M U2 NumRows=1 NumCols=7	 MpyInt_M M1
 AddInt2_M A1	 Int_M I1 NumRows=4 NumCols=7 IntMatrixContents="1 0 0 0 1 1 1.... 0"
 BitsToInt B1 nBits=3	 TableInt_M T1 NumRows=1 NumCols=7 IntTable="0 0 0 0 0 0 0 0 ....0 0"



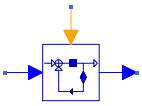
Chop
C1
nRead=7
nWrite=4
Offset=0
UsePastInputs=YES



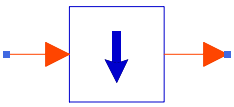
LogicXOR2
L1



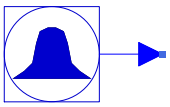
ImpulseFloat
I6
Level=1.0
Period=4
Delay=0



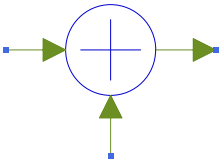
Integrate
I5
FeedbackGain=1.0
Top=0.0
Bottom=0.0
Saturate=YES
State=0.0



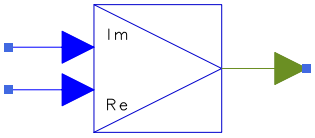
DownSample
D1
Factor=4
Phase=0



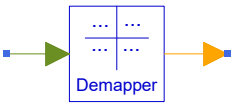
IID_Gaussian
I1
Mean=0.0
Variance=Sigma



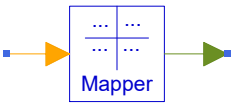
AddCx2
A1



RectToCx
R2

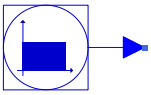


Demapper
D1
ModType=BPSK
MappingTable=""



Mapper
M1
ModType=BPSK
MappingTable=""

ParamSweep
Sweep1
SweepVar="Sigma"
SimInstanceName[1]="DF"
SimInstanceName[2]=
SimInstanceName[3]=
SimInstanceName[4]=
SimInstanceName[5]=
SimInstanceName[6]=
Start=0.01
Stop=10
Step=



Rect
R1
Height=1.0
Width=2
Period=4

2. Versuchsvorbereitung

2.1 Generator-Matrix

- Entwerfen Sie eine Generator-Matrix $\vec{\vec{G}}$ für einen (7,4)-Hamming-Code und berechnen Sie die zugehörige Prüf-Matrix $\vec{\vec{H}}$ bzw. $\vec{\vec{H}}^T$.
  Stellen Sie die Syndrom-Tabelle $\vec{S}$ für die Fehler auf, die auch korrigiert werden können!

- Das Ergebnis der Syndrom-Berechnung im Kanal-Decodierer ist eine 3-stellige Dualzahl. Zu jedem Syndrom gehört **ein** möglicher 7-stelliger Fehlervektor $\vec{e}$. Alle möglichen Fehlervektoren bilden damit eine Tabelle/Matrix. Wandeln Sie die 3-stelligen Syndrome in Dezimalzahlen um und verwenden Sie diese, um die Fehlervektoren mittels ***TableInt_M*** während des Versuchs zu adressieren!

2.2 BPSK-Modulation

In diesem Versuch wird formal BPSK-Modulation verwendet. Dazu werden die Elemente *Mapper* bzw. das inverse Element *Demapper* verwendet. Machen Sie sich vor Versuchsbeginn mit der **BPSK-Modulation vertraut**. Während des Versuches wird als Trägerfrequenz 0Hz verwendet.

3. Versuchsdurchführung

Dieser Versuch behandelt die Definition eines gestörten Kanals und den Aufbau der Kanal-Codierung bzw. Kanal-Decodierung.

3.1 Aufbau des gestörten Kanals

Starten Sie mit dem Aufbau des gestörten Kanals. Dazu werden die Elemente ***Mapper*** und ***Demapper*** benötigt. Um deren Funktion zu verstehen, ist zunächst eine Testschaltung zu entwerfen.

- Generieren Sie eine Schaltung bestehend aus *Rect* und 2 Anzeigeinstrumenten *NumerikSink*. Verwenden Sie 10 Abtastwerte. Modifizieren Sie den *Rect* so, daß jeder Impuls eine Dauer von einem Abtastwert bei einer Periodenlänge von 2 Abtastwerten hat. Dieses ist nun Ihre Signalquelle X für die folgenden Untersuchungen. Schalten Sie *Mapper* und *Demapper* in Kette und verbinden Sie den *Mapper* mit der Signalquelle. Verwenden Sie sowohl für den *Mapper* als auch für den *Demapper* **BPSK-Modulation**. Betrachten Sie die Ausgangssignale des *Mappers* und des *Demappers* und interpretieren Sie diese!

- Fügen Sie nun die Störung in den Kanal ein, in dem Sie zwischen *Mapper* und *Demapper* ***Normalverteiltes Rauschen*** mit dem Mittelwert Null hinzufügen. Stören Sie lediglich den **Realteil** des Signals (Daten-Typ beachten). Verwenden Sie für die Varianz der Rauschquelle eine Variable (*VAR-Element*). Was ist für verschiedene Werte der Varianz zu beobachten?

- Speichern Sie Ihre Schaltung als **Unternetzwerk** mit den dafür notwendigen Modifikationen ab (siehe Versuche 1-4)! Verwenden Sie als Dateinamen *AWGN_Kanal.dsn*.

3.2 Aufbau der Kanalcodierung

Der Aufbau der Kanalcodierung sollte schrittweise erfolgen, da nur so die einzelnen Stufen der Codierung bzw. Decodierung überprüft werden können. Um die Simulationsergebnisse übersichtlich zu gestalten, sollte ***DefaultNumericStop=0*** gesetzt werden!

- Definieren Sie im Schematic die zuvor entworfene (7,4)-Generatormatrix mittels ***Int_M***. Testen Sie Ihre Eingabe!

- Implementieren Sie nun die Multiplikation $\vec{c} = \vec{d} \cdot \vec{\vec{G}}$. Verwenden Sie für erste Tests $\vec{d} = (1100)$. Überprüfen Sie das Ergebnis. Sollten Sie mit dem Resultat zufrieden sein, dann variieren Sie den Datenvektor und überprüfen die Ergebnisse erneut.

- Erweitern Sie Ihre bisherige Schaltung um die Modulo-2-Arithmetik mittels ***UnPkInt_M*** und ***ModuloInt***. Wandeln Sie das Signal nach erfolgter Modulo-Berechnng wieder in den Typ Matrix um (wie?). Überprüfen Sie das Ergebnis!

- Implementieren Sie nun die Matrix $\vec{\vec{H}}^T$ und multiplizieren Sie diese mit dem empfangenen Codevektor $\vec{c}$. Führen Sie anschließend die für die Modulo-2-Berechnung notwendigen Schritte durch (siehe oben)! Wandeln Sie das Ergebnis mittels ***BitsToInt*** in eine Integer-Zahl um und verwenden Sie diese zur Adressierung der in der **Versuchsvorbereitung** ermittelten Tabelle der möglichen Fehlervektoren $\vec{e}$ mittels ***TableInt_M.*** Überprüfen Sie den berechneten Fehlervektor $\vec{e}$!

- Überprüfen Sie nun ihren Kanal-Decodierer, in dem Sie senderseitig kontrolliert Fehlervektoren $\vec{e}$ zu den berechneten Codevektoren $\vec{c}$ addieren und diese dann dem **Kanaldecodierer** zuführen. Implementieren Sie die Fehlerkorrektur[1]. Beachten Sie wieder die Modulo-2-Berechnung!

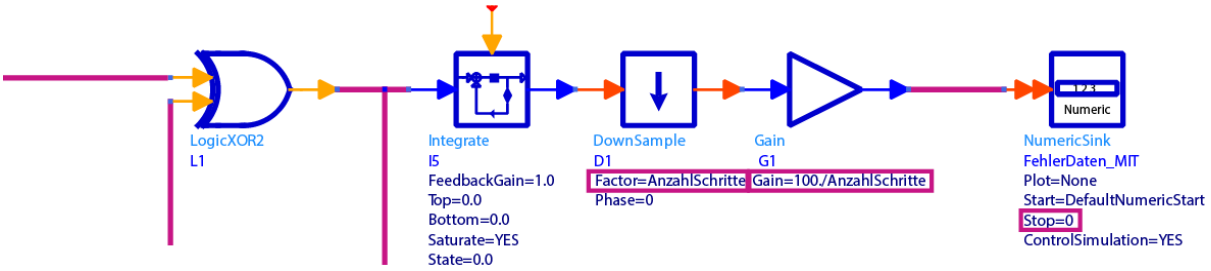
[1] Siehe Vorlesungsscript

3.3 Testen der Kanalcodierung

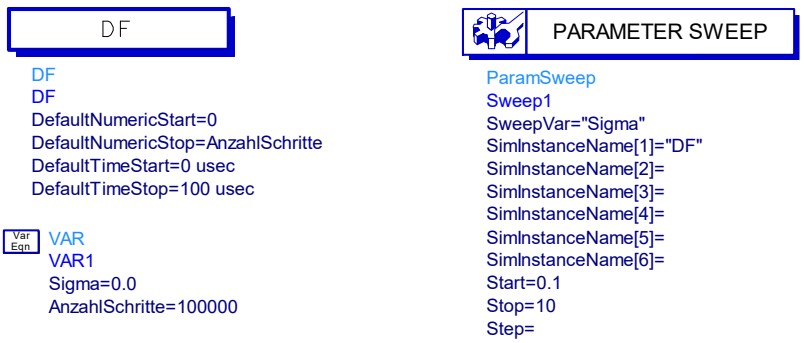
Um die folgenden Untersuchungen übersichtlich zu gestalten, speichern Sie einzelne Blöcke (Codewort-Berechnung, Syndrom-Berechnung, Fehlerkorrektur, AWGN-Kanal, etc.) des Schematics in Unternetzwerken ab.
Erzeugen Sie nun ein neues Schematic, welches unter Verwendung der zuvor gespeicherten Unternetzwerke für verschiedene Werte der Varianz der Gauß-verteilten Störung des AWGN-Kanals (Sigma) die Bitfehlerraten berechnen kann. Dazu ist ein Parameter-Sweep erforderlich (Sigma).
Es sollen nun 4 verschiedene Fälle untersucht werden, dazu sind **nur 4 NumericSinks** erforderlich:

- Bitfehlerrate der Daten-Bits **ohne Korrektur** als Funktion der Rauschleistungsdichte Sigma
- Bitfehlerrate der Daten-Bits **nach erfolgter Korrektur** als Funktion der Rauschleistungsdichte Sigma
- Bitfehlerrate der Codewort-Bits **ohne Korrektur** als Funktion der Rauschleistungsdichte Sigma
- Bitfehlerrate der Codewort-Bits **nach erfolgter Korrektur** als Funktion der Rauschleistungsdichte Sigma

Deaktivieren Sie alle anderen möglicherweise vorhandenen Sinks!
Verwenden Sie zur Berechnung der Fehlerrate in Prozent die folgende Anordnung! Das Element „*DownSample*“ dient zur Reduktion der anfallenden Datenmenge!
Stellen Sie im NumericSink **Stop auf Null**!



Die Simulation erfolgt mittels der folgenden Einstellungen!



Was ist zu beobachten? Stellen Sie Ihre Ergebnisse sowohl doppelt linear als auch doppelt logarithmisch dar!