Digitale AV-Technik
Prof. Uwe Hahne
Sommersemester 2025

Digitale AV-Technik - Aufgabenblatt 10

Thema: Fehlererkennung und -korrektur

Ziele:

- Rechnen mit **XOR** üben und Parität verstehen
- Prüfsummen kennen lernen
- Verständnis der Hamming-Code-Struktur und der Platzierung von Daten- und Paritätsbits.

Aufgabe 1: Paritäten ermitteln

Ermitteln Sie die (gerade) Parität der folgenden Bitketten:

- 11010110010
- 01000100

.....

.....

.....

.....

Prüfen Sie ob man mit dem folgenden Python-Code `parity.py` die Parität ermitteln kann:

```python
def compute_parity(binary_string):
    parity = 0
    for bit in binary_string:
        parity ^= int(bit)
    return parity
```

Erklären Sie, wie der Code funktioniert und ob er die Parität korrekt berechnet.

.....

.....

.....

.....

Aufgabe 2: EAN Prüfziffer

Ermitteln Sie online wie die Prüfziffer beim EAN Code berechnet wird und prüfen Sie nach ob diese auf dem folgenden Bild korrekt ist.

Aufgabe 3: CRC ermitteln

Ermitteln Sie für die Bitfolge 11011 und das Prüfpolynom

$$x^5 + x^4 + x^2 + 1$$

den Restwert, der an die Bitfolge angehängt wird.
Hinweis: Für die Bitfolge 11011 und den Prüfwert 110101 (Länge n = 6), ergibt sich der Rahmen 1101100000 (Bitfolge um n-1 = 5 Nullen verlängert). Anschließend wird der Rahmen durch das Generatorpolynom dividiert.

. . . . . .
. . . . . .
. . . . . .
. . . . . .

. . . . . .
. . . . . .
. . . . . .
. . . . . .

Das Ergebnis hat nun nur noch drei relevante Stellen, also werden die letzten n-1 (=5) Stellen als CRC-Wert übernommen. Die Bitfolge und der Rest ergeben die zu übertragenden Daten. Verifizieren Sie als Empfänger die Korrektheit der Daten, indem Sie eine Polynomdivision mit dem Rahmen und dem Prüfwert durchführen:

. . . . . .
. . . . . .
. . . . . .
. . . . . .

. . . . . .
. . . . . .
. . . . . .
. . . . . .

Hinweis: Die Lösung dieser Aufgabe findet sich hier.

Aufgabe 4: Fehlererkennung mit CRC

In der Vorlesung wurde die Nachricht "Hi" mit dem CRC-Polynom $x^2 + 1$ codiert und mit dem Rest "11" angehängt übertragen. Welcher Restwert würde bei der fehlerhaften Übertragung der Nachricht "Hh" entstehen?

Hinweis: Verwenden Sie die ASCII-Codes für "H" und "h", also 72 und 104. Die Nachricht wird in eine Bitfolge umgewandelt, indem die ASCII-Codes in Binärzahlen umgewandelt werden. Anschließend wird der CRC-Wert aus der Vorlesung angehängt.

. . . . . .
. . . . . .
. . . . . .
. . . . . .

. . . . . .
. . . . . .
. . . . . .
. . . . . .

Hinweis: Schauen Sie sich den Python Code `crc8.py` und seine Quelle an, um zu verstehen, wie man die Stelle des Bitfehlers findet.

Aufgabe 5: Fülle den Hamming-Block korrekt aus

Ein **Hamming(15,11)-Code** erweitert **11** Datenbits mit 4 Paritätsbits also insgesamt **15** Bits, um Fehler erkennen und korrigieren zu können. In dieser Aufgabe sollst du lernen, wie man den **Hamming-Block** korrekt mit Daten- und Paritätsbits füllt. Diese Aufgabe wird auch im Video von 3blue1brown gestellt.

Hinweis: Verwende immer das **Even Parity-Verfahren** (gerade Anzahl von 1en).

Gegeben sind die folgenden 11 Datenbits:

$$01001101100$$

Platziere diese Datenbits an den richtigen Positionen im **Hamming-Block** und ergänze die **Paritätsbits** so, dass der Code fehlerfrei übertragen werden kann.

Jedes Paritätsbit prüft eine bestimmte Gruppe von Bits. Führe die Berechnungen für die Paritätsbits an den Stellen 1, 2, 4 und 8 durch.

Um dir klarzuwerden, welches Paritätsbit welchen Bereich überprüft, schreibe die Indizes aller Zellen in binärer Form auf.

Prüfe, wie der Hamming-Code auf einen Bitfehler reagiert.

- Nimm an, dass das Bit an Position **6** fehlerhaft übertragen wurde.
- Berechne das **Syndrom**, um den Fehler zu erkennen.
- Korrigiere den Fehler im Hamming-Code.

Zur Überprüfung deiner Lösung kannst Du den Python Code `hamming_3b1b.py` verwenden und dir den zweiten Teil des Videos von 3blue1brown anschauen.