Digitaltechnik 1. Semester

Übung 5

Besprechung: 29.10.2020 Abgabe: 05.11.2020

Aufgabe 1: **Zweierkomplement**

a) Führen Sie folgende Rechnungen im Zweierkomplement durch. Die Wortlänge betrage dabei 8 Bit. Geben Sie auch den dezimalen Wert des Resultats an.

12+27	33-17	-12-14
127+1	-88-77	5·22

b) Stellen Sie 0.1 als binäre Fixkommazahl mit 6 Bit nach dem Komma dar und errechnen Sie den Fehler.

c) Die folgenden Berechnungen sollen mit binären Fixkommazahlen mit je 4 Bit vor und nach dem Komma ausgeführt werden.

0.125+2.875	0.5-3.75	0.7-0.2-0.5

Aufgabe 2: **Kodes/Zahlensysteme**

(2) Binär-Kode: Übliche Darstellung einer Zahl als Zweierpotenzen

(8) Oktale Zahlendarstellung: Darstellung als Potenzen von 8

(10) "Normale" Dezimalzahlendarstellung

(H) Hexadezimale Zahlendarstellung, bei der eine Zahl als Potenzen von 16 dargestellt wird. Für die Ziffern "0" bis "9" werden die entsprechenden Zahlen verwendet. Die Ziffern "10" bis "15" werden mit Hilfe der Buchstaben "A" bis "F" kodiert.
Beispiel: $1F_{(H)} \rightarrow 1 \cdot 16 + 15 \cdot 1 = 31_{(10)}$ (häufig wird auch ein $-Zeichen als Kennzeichnung einer Hexadezimal-Zahl verwendet: $1F)

(BCD) Binär kodierte Dezimalzahl, bei der jede Stelle einer Dezimalzahl als 4-bit breite Binärzahl kodiert ist.
Beispiel: $59_{(10)} \rightarrow 01011001_{(BCD)}$

(Gray) Der Gray-Kode is ein einschrittiger Kode, d.h. beim Wechsel von einem Zustand in den Folgezustand wechselt jeweils nur ein Bit. Folgende Tabelle kann zu Hilfe genommen werden:

1a) $12+27$

	±	64	32	16	8	4	2	1
+	0	0	0	0	1	1	0	0
	0	0	0	1	1	0	1	1
	0	0	1	0	0	1	1	1

$= 39_{10}$ ✓

ii) $33-17$

	±	64	32	16	8	4	2	1
		0	1	0	0	0	0	1
(−)		~~0~~	~~1~~	~~0~~	~~0~~	~~0~~	~~0~~	~~1~~
+		1	0	1	1	1	1	1
	0	0	0	1	0	0	0	0

$= 16_{10}$ ✓

iii) $-12-14$

	±	64	32	16	8	4	2	1
(−)					~~1~~	~~1~~	~~0~~	~~0~~
				1	0	1	0	0
(−)					~~1~~	~~1~~	~~1~~	~~0~~
				1	0	0	1	0
			1	0	0	1	1	0

$= -26_{10}$ ✓

iv) $127+1$

	±	64	32	16	8	4	2	1
	0	1	1	1	1	1	1	1
								1
0 1	0	0	0	0	0	0	0	0

$= -128_{10}$ ✓

v) $-88-77$

	±	64	32	16	8	4	2	1
(−)	~~1~~	~~0~~	~~1~~	~~1~~	~~0~~	~~0~~	~~0~~	
	1	0	1	0	1	0	0	0
(−)	~~1~~	~~0~~	~~0~~	~~1~~	~~1~~	~~0~~	~~1~~	
	1	0	1	1	0	0	1	1
1	0	1	0	1	1	0	1	1

$= -165_{10}$ ✗

vi) $5 \cdot 22$

	64	32	16	8	4	2	1
x					1	0	1
			1	0	1	1	0
			1	0	1	1	0
		0	0	0	0	0	0
	1	0	1	1	0	0	0
	1	1	0	1	1	1	0

$= 110_{10}$ ✓

b)

2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
0	0	0	1	1	0

$= 0.09375$

$E_{rror} = 0.1 - 0.09375$

$= 6.25 \times 10^{-3}$ ✓

c) $0.125 + 2.875$

2^1	2^0 .	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
0	0.	0	0	1	0	0	0
1	0.	1	1	1	0	0	0
1	1.	0	0	0	0	0	0

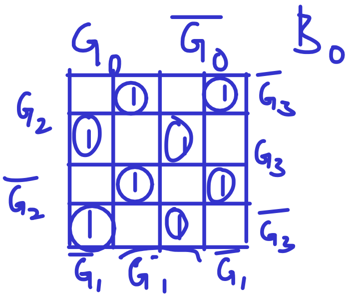
$= 3$ ✓

ii) $0.5 + 3.75$

	2^2	2^1	2^0 .	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
		0	0.	1	0	0	0	0	0
		1	1.	1	1	0	0	0	0
1	0	0	0	1	0	0	0	0	0

4.25

Binär (B_0)	Gray ($3 2 1 G_0$)
0000	0000
0001	0001 ✓
0010	0011
0011	0010 ✓
0100	0110
0101	0111 ✓
0110	0101
0111	0100
1000	1100
1001	1101
1010	1111
1011	1110
1100	1010
1101	1011
1110	1001
1111	1000



B_0 = All Minterms

a) Vervollständigen Sie nun die folgende Liste von Kodeumsetzungen, indem Sie den verlangten Zielkode aus dem angegebenen Kode errechnen.

A F 9: 1010 1111 1001

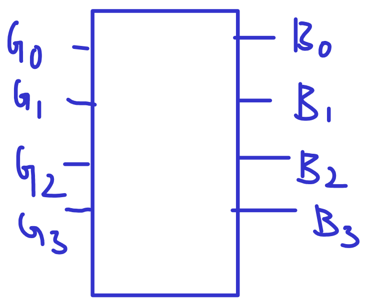
6 13=D: 0110 1101

A F: 1 1010 1111 =

$111111111_{(2)}$	→	BCD $2^9 + 2^8 + \ldots = 511_{10}$ ✓
$AF9_{(H)}$	→	Binär 101011111001_2 ✓
$10111111_{(2)}$	→	Dezimal 191_{10} ✓
$109_{(10)}$	→	Hexadezimal $6D_{16}$ ✓
$1110_{(Gray)}$ 1011	→	Dezimal 11_{10} ✓
$657_{(8)}$	→	Hexadezimal $1AF_{16}$ ✓
$FFF_{(H)}$	→	Dezimal 4095_{10} ✓
$10011000_{(2)}$ 98	→	Hexadezimal 98_{16} ✓
$126.375_{(10)}$	→	Binär 1111110.011 ✓

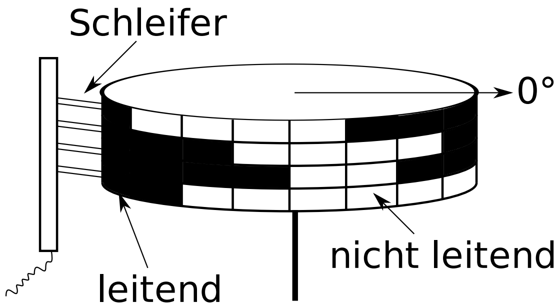
b) Es soll nun ein Kodeumwandler in Hardware realisiert werden, der die Gray-Kode-Signale in binär-kodierte Signale umwandelt. Diese umgewandelten Signale können dann in einem Computer weiterverarbeitet werden. Als Hilfsmittel soll die obige Tabelle verwendet werden.

1. Erstellen Sie ein Blockschaltbild der zu realisierenden Schaltung.
2. Bestimmen Sie eine disjunktive Zusammenfassung anhand der Karnaughdiagramme.
3. Zeichnen Sie die entsprechende zweistufige Schaltung mit UND und ODER Toren.



Aufgabe 3: **Einschrittiger Kode**

Ein zylindrischer Massstab zur elektronischen Messung eines Winkels kann, wie unten gezeichnet, konstruiert werden. Der Winkel soll mit vier Bit kodiert werden. Pro Bit führen je zwei Drähte auf eine Trommel, die aus leitenden und nicht leitenden Flächenstücken besteht. Liegt ein Schleifer auf einem leitenden Teilstück, wird der Stromkreiss geschlossen und das als "1" interpretiert, bei einem nicht leitenden Teilstück entsprechend als "0".



a) Warum wird hier ein einschrittiger und nicht z.B. ein Binär-Kode auf den Zylinder gebracht?

Nur eine Teilstück ändert sich => mehr toleranz gegen Fehlern und eventuell weniger aufwand bei dem herstellungsprozess

b) Wie gross ist der maximale Fehler in Grad (Abweichung des wahren analogen Wertes vom codierten Wert), der beim Ablesen entstehen kann, wenn ein einschrittiger Kode verwendet wird? Die Flächenstücke haben einen Positionierungsfehler von max. ϵ ($\epsilon > 0$, unendlich klein).

16 states $\rightarrow \frac{360}{16} = 22.5°, \div 2 = 11.25° = \epsilon$

c) Der Kode soll zirkulär sein, d.h. dass der Übergang von 15 (bei 4Bit) auf 0 wieder einschrittig sein soll. Die folgenden Werte seien schon gegeben:

0:	1010	4:	0101	8:	0011	12:	......
1:	0010	5:	0100	9:	1011	13:	......
2:	0110	6:	0000	10:	1111	14:	......
3:	...... 0111 ~~0100~~	7:	0001	11:	......	15:	...... ~~1011~~ 1000 1110 ~~0010~~

Geben Sie den (die) möglichen Wert(e) für 3 und 15 an.

d) Vervollständigen Sie den Kode mit den Werten für 11, 12, 13 und 14. Wieviele verschiedene Möglichkeiten gibt es?

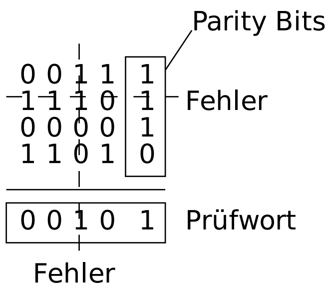
Aufgabe 4: **Fehlerkorrigierende Datenübertragung**

In Datenübertragungen treten fast immer Fehler auf. Werden zusätzlich zu den Daten Kontroll-Informationen übertragen, können die Fehler erkannt oder sogar korrigiert werden. Um Fehler erkennen zu können, wird jedes Datenwort um ein "Parity Bit" ergänzt, so dass die Anzahl der "1" pro Wort immer geradzahlig ("even parity") oder immer ungeradzahlig ("odd parity") ist.
Um Fehler nicht nur erkennen, sondern auch korrigieren zu können, ist zusätzliche Redundanz notwendig. Bei der sogenannten Blockkorrektur wird pro Datenwort ein Parity-Bit hinzugefügt und nach einer gewissen Anzahl Datenwörter ein zusätzliches Prüfwort eingefügt. Damit kann ein *Einzelfehler* in einem Datenblock erkannt und korrigiert werden (siehe Beispiel unten).

In dieser Aufgabe sollen über eine Zweidrahtleitung 4 Bit grosse Datenwörter seriell übertragen werden. Zur Übertragung soll folgendes Format verwendet werden:

- Das MSB (Most Significant Bit = Bit mit dem grössten Stellenwert) soll zuerst übertragen werden.
- Nach der Übertragung eines 4-Bit-Datenwortes wird ein Parity-Bit gesendet, das die Ergänzung auf Odd-Parity (ungerade Anzahl Einer) bildet.
- Nach 4 Datenwörtern (inklusive Parity-Bits) wird ein Prüfwort bestehend aus 5 Bit übertragen. Es hat folgenden Aufbau: Die Bits mit dem gleichen Stellenwert und die gesendeten Parity-Bits werden auf Even-Parity ergänzt.

Der vom Empfänger erhaltene Bitstrom strukturiert dargestellt (Bsp.):



a) Die folgenden vier hexadezimalen Ziffern sollen übertragen werden:
$A, $D, $2,$9

	8	4	2	1	P
10	1	0	1	0	0
13	1	1	0	1	1
2	0	0	1	0	1
9	1	0	0	1	0

Bestimmen Sie das Parity-Bit für jede der vier Ziffern und anschliessend das 5-Bit Prüfwort, welches zusätzlich übertragen werden muss. P wort — 11000

b) Die folgende Bitkombination wird vom Empfänger aufgenommen (Die Zeichen werden in der Reihenfolge von links nach rechts empfangen):
0100|00111|11001|01010|11000|...

Welche hexadezimalen Ziffern sind vom Sender geschickt worden, unter der Voraussetzung, dass höchstens ein Bit falsch empfangen wurde?

c) Der Parity-Generator für das Prüfwort soll nun in Hardware realisiert werden. Entwerfen Sie eine kombinatorische Logik, die aus den 4 Bits das Even-Parity-Bit des Prüfwortes bildet.

TODO : Lernphase

8	4	2	1	P
0	1	0	0	0
0	0	1	1	1
1	1	0	0	1
0	1	~~0~~ 1	1	0
1	1	0	0	0

even parity

odd