Aufgabenblatt 7 Ausgabe: 23.11., Abgabe 03.12.2025 24:00

Gruppe	
Name(n)	Matrikelnummer(n)

Aufgabe 7.1 (Punkte 5 + 5 + 5 + 5)

Schaltfunktionen:

(a) Wieviele Boole'sche Funktionen $f(x_3, x_2, x_1, x_0)$ von 4 Variablen gibt es?

(b) Wieviele Boole'sche Funktionen $f(x_3, x_2, x_1, x_0)$ von 4 Variablen gibt es, für die gilt $f(x_3, x_2, 0, x_0) = f(x_3, x_2, 1, x_0)$?

(c) Wieviele Boole'sche Funktionen $f(x_3, x_2, x_1, x_0)$ von 4 Variablen gibt es, deren Funktionstabelle genau 12 Einsen enthält (die also für 12 Kombinationen von (x_3, x_2, x_1, x_0) den Wert $f(x_3, x_2, x_1, x_0) = 1$ annimmt)?

(d) Wieviele Boole'sche Funktionen $f(x_3, x_2, x_1, x_0)$ von 4 Variablen gibt es, wenn als Funktionswert neben 0 und 1 auch noch * (Don't Care) zugelassen ist? Wie lautet die allgemeine Formel für n Variablen?

Aufgabe 7.2 (Punkte 5+10+5+10)

Golay(24,12,8)-Code: Wenn die Anzahl der Codewörter nicht zu groß ist, kann man fehlerkorrigierende Codes auch ohne komplizierte Mathematik einfach durch Ausprobieren aufbauen.

Ein Kandidat ist der Golay(24,12,8)-Binärcode, der aus insgesamt $2^{12} = 4096$ Codewörtern der Länge $n = 24$ Bit mit minimaler Hamming-Distanz $d = 8$ besteht.

Ein offensichtliches Verfahren zur Konstruktion besteht darin, einfach alle Bitmuster der gegebenen Wortlänge (hier $n = 24$ Bit) als Kandidaten für Codewörter auszuprobieren, zum Beispiel in der Reihenfolge der Dualzahlen (also `0x000000`, `0x000001`, `0x000002`, ... `0xFFFFFF`). Ein solches Bitmuster wird als neues Codewort akzeptiert, wenn es mindestens die geforderte Hammingdistanz (hier also $d = 8$) zu allen bisher gefundenen Codewörtern aufweist.

(a) Wieviele Bitfehler in einem bei der Übertragung gestörten Codewort können mit diesem Code erkannt bzw. korrigiert werden?

(b) Verwenden Sie das oben angegebene Verfahren um ausgehend vom ersten Codewort $a_0 =$`0x000000` die nächsten drei Codewörter des Golay(24,12,8)-Codes zu berechnen.

(c) Schreiben Sie eine Java-Methode (oder C-Funktion), um die Hamming-Distanz von zwei Codewörtern (gegeben als 32-bit `int`) zu berechnen:

```
int hamming( int a, int b ) {
  ...
}
```

Wichtig: Erklären Sie, wie Ihr Algorithmus funktioniert.

(d) Skizzieren Sie ein Programm (z.B. Java oder C), das mit dem angegebenen Verfahren alle 4096 Codewörter des Golay(24,12,8)-Codes berechnet und ausgibt. Als Datenstruktur eignet sich zum Beispiel ein Array (der Größe 4096) oder eine `ArrayList<Integer>`.

Aufgabe 7.3 (Punkte 15+15)

Kanonische Formen: Stellen Sie für die beiden folgenden Funktionen der drei Variablen x_3, x_2, x_1 zunächst die Funktionstabelle auf, und ermitteln Sie dann die kanonische DNF, die kanonische KNF, sowie die Reed-Muller-Form der Funktion:

(a) $f(x) = (x_3 \vee \overline{x_2}) \wedge (x_2 \vee \overline{x_1})$

(b) $g(x) = \overline{x_3} \oplus \overline{x_1}$

Aufgabe 7.4 (Punkte 5+5+5+5)

NAND als vollständige Basis: Geben Sie an, wie die folgenden Boole'schen Funktionen durch geeignete Schaltungen nur aus NAND-Gattern gebildet werden könnenr:

(a) $f_a(x_1, x_0) = x_1 \wedge x_0$ **and**

(b) $f_b(x_1, x_0) = x_1 \vee x_0$ **or**

(c) $f_c(x_1, x_0) = \overline{x_0}$ **not**

Die Realisierung der 3 Grundfunktionen der Boole'schen Algebra liefert den Nachweis, dass die NAND-Funktion eine vollständige Basismenge bildet, aus der sich beliebige Schaltungen aufbauen lassen.

(d) Formen Sie jetzt die folgende Schaltfunktion so um, dass Sie ausschließlich mit NAND-Schaltgliedern realisiert werden kann:

$$f_d(x_2, x_1, x_0) = (\overline{x_2} \wedge (\overline{x_1} \vee x_0)) \vee (x_0 \wedge (\overline{x_1} \vee x_0))$$