Test und Verlässlichkeit
Grosse Übung 5

Prof. G. Kemnitz

Institut für Informatik, Technische Universität Clausthal

25. Juni 2015

WB und Rundungsfehler

Drainstromberechnung MOS-Transistor

Die Gleichung für die Berechnung des Drainstroms eines MOS-Transistors im aktiven Bereich lautet:

$$I_\mathrm{D} = k \cdot \left((U_\mathrm{GS} - U_\mathrm{T}) \cdot U_\mathrm{DS} - \frac{U_\mathrm{DS}^2}{2} \right)$$

soll mit folgenden Festkommaformaten programmiert werden:

```
int ID;  // Drainstrom in mA, 20 Nachkommabits
int k;   // Steilheit in mA/V^2, 24 Nachkommabits
int UGS; // Gate-Source-Spannung in V, 24 Nachkommabits
int UT;  // Einschaltspannung in V, 24 Nachkommabits
int UGS; // Gate-Source-Spannung in V, 24 Nachkommabits
int UDS; //Drain-Source-Spannungin V, 24 Nachkommabits
```

Beispielimplementierung der Gleichung in C:

```
int ID=(((k>>6)*(UDS>>16))>>12)*((UGS-UT-UDS/2)>>16)>>2;
```

1. Bestimmen Sie die Darstellungsbereiche und Quantisierungsfehler für k, ID und die Spannungen.
2. Bestimmen Sie die zulässigen Wertebereiche für k, wenn der Wertebereich der Spannungen auf $\pm 16\,\mathrm{V}$ begrenzt ist?
3. Bestimmen Sie für die nachfolgenden Zahlenbeispiele den Istwert ID_ist, den Sollwert ID_soll und den relativen Fehler (ID_ist − ID_soll)/ID_soll in Prozent:

k	UGS	UT	UDS	ID_ist	ID_soll	Fehler
$3{,}3\ \frac{\mathrm{A}}{\mathrm{V}^2}$	1,7 V	1 V	0,4 V			
$289\ \frac{\mathrm{A}}{\mathrm{V}^2}$	2,4 V	1 V	3,8 V			
$140\ \frac{\mathrm{mA}}{\mathrm{V}^2}$	4,5 V	1 V	3,8 V			

4. Erhöhen Sie die Rechengenauigkeit durch Berechnung der Produkte mit 64 Bit.
5. Lösen Sie dafür erneut Aufgabenteil 2 und 3.

Lösung Teilaufgabe 1

Bestimmen der Darstellungsbereiche und Quantisierungsfehler für k, ID und die Spannungen.

- Strom ID:
  - 12 Vorkomma- und 20 Nachkommabits
  - Darstellungsbereich: -0x800 bis 7FF,FFFFF ($\pm 2048\,\mathrm{mA}$)
  - Quantisierungsfehler: $\pm 2^{-21}\mathrm{mA}$
- Die Spannungen UT, UGS und UDS:
  - 8 Vorkomma- und 24 Nachkommabits
  - Darstellungsbereich: -0x80 bis 7F,FFFFFF ($\pm 128\,\mathrm{V}$)
  - Quantisierungsfehler: $\pm 2^{-25}\mathrm{V}$ ($\approx \pm 30\,\mathrm{nV}$)
- Steilheit k: 8 Vorkomma- und 24 Nachkommabits
  - Darstellungsbereich: -0x80 bis 7F,FFFFFF ($\pm 128\,\mathrm{mA/V^2}$)
  - Quantisierungsfehler: $\pm 2^{-25}\mathrm{pA/V^2}$ ($\approx \pm 30\,\mathrm{pA/V^2}$)

Lösung Teilaufgabe b

Zulässigen Wertebereiche für k, wenn der Wertebereich der Spannungen auf $\pm 16\,\mathrm{V}$ begrenzt ist:

```
int ID=(((k>>6)*(UDS>>16))>>12)*((UGS-UT-UDS/2)>>16)>>2;
```

- Keine WB-Überschreitung:

  `(tmp>>12)*(UGS-UT-UDS/2)>>16)`

  $$\begin{aligned} \mathrm{tmp} \cdot 2^{-12} \cdot 2,5 \cdot 16\,\mathrm{V} \cdot 2^{24-16} &< 2^{31} \\ \mathrm{tmp} &< 8{,}6 \cdot 10^8 \end{aligned}$$

- Keine WB-Überschreitung:

  `tmp=(k>>6) * (UDS>>16)`

  $$\begin{aligned} k \cdot 2^{24-6} \cdot 16\,\mathrm{V} \cdot 2^{24-16} &< 8{,}6 \cdot 10^8 \\ k &< 8{,}6 \cdot 10^8 \cdot 2^{-30} \approx 0,8 \end{aligned}$$

Programm zur Lösung von Teil c

```
float f_k, f_UGS, f_UT, f_UDS;

void do_calc_and_print_result(){
 int   k       = (int)(f_k*(1<<24));
 int   UGS     = (int)(f_UGS*(1<<24));
 int   UT      = (int)(f_UT*(1<<24));
 int   UDS     = (int)(f_UDS*(1<<24));
 int ID=(((k>>6)*(UDS>>16))>>12)*((UGS-UT-UDS/2)>>16)>>2;
 float ID_soll= f_k * f_UDS*(f_UGS-f_UT-f_UDS/2.0);
 float Fehler = (ID_ist-ID_soll)/ID_soll * 100;
 printf("ID_ist=%5.3fmA ID_soll=%5.3fmA rel. Fehler="
        "%2.3f%%\n",ID_ist*1000, ID_soll*1000, Fehler);
}
```

Ist- und Sollwerte zu Teil c

```
int main(){
 f_k=3.3E-3; f_UGS=1.7; f_UT=1; f_UDS=0.4;
 do_calc_and_print_result();
 f_k=289E-3; f_UGS=2.4; f_UT=1; f_UDS=1.2;
 do_calc_and_print_result();
 f_k=7.8E-5; f_UGS=4.5; f_UT=1; f_UDS=3.0;
 do_calc_and_print_result();
}
```

k	UGS	UT	UDS	ID_ist	ID_soll	Fehler
$3{,}3\ \frac{\mathrm{A}}{\mathrm{V}^2}$	1,7 V	1 V	0,4 V	0,641 mA	0,660 mA	-2,899%
$289\ \frac{\mathrm{A}}{\mathrm{V}^2}$	2,4 V	1 V	3,8 V	276,163 mA	277,440 mA	-0,460%
$140\ \frac{\mathrm{mA}}{\mathrm{V}^2}$	4,5 V	1 V	3,8 V	0,366 mA	0,468 mA	-21,750%

Programm zur Lösung von Teil d (+ WB(k) Teil e)

$$I_\mathrm{D} = k \cdot \left((U_\mathrm{GS} - U_\mathrm{T}) \cdot U_\mathrm{DS} - \frac{U_\mathrm{DS}^2}{2} \right)$$

```
int ID;  // Drainstrom in mA, 20 Nachkommabits
int k;   // Steilheit in mA/V^2, 24 Nachkommabits
int U...;// ...-Spannung in V, 24 Nachkommabits
```

Es ändert sich nur die Berechnung[1]:

```
long long int t1 = (long long int)k*UDS;
long long int t2 =(t1>>32)*(UGS-UT-UDS/2);
int       ID = t2 >> 20;
```

[1] Das Produkt von 3 Zahlen mit je 24 Nachkommabits hat $3 \cdot 24$ Nachkommabits. Die Verschiebungen im 32 und 20 Bit nach rechts führen auf die für den Strom vereinbarten 20 Nachkommabits.

Wertebereich von k zu Teil e

```
long long int t1 = (long long int)k*UDS;
long long int t2 =(t1>>32)*(UGS-UT-UDS/2);
int       ID = t2 >> 20;
```

Zulässiger Wertebereiche für k:

- Keine WB-Überschreitung für ID = t2 >> 20:

$$|t_2| < 2^{51}$$

- Keine WB-Ü. für t2 =(t1>>32)*(UGS-UT-UDS/2):

$$\begin{aligned} |t_1| \cdot 2^{-32} \cdot 2{,}5 \cdot 16\,\mathrm{V} \cdot 2^{24} &< 2^{51} \\ |t_1| &< 1{,}4 \cdot 2^{16} \end{aligned}$$

- Keine WB-Überschreitung für t1 =k*UDS:

$$\begin{aligned} |k| \cdot 2^{24} \cdot 16\,\mathrm{V} \cdot 2^{24} &< 1{,}4 \cdot 2^{16} \\ |k| &< 3{,}2 \end{aligned}$$

Ist- und Sollwerte und Fehler (Teil e)

- Ergebnisse Aufgabenteil e (verbessertes Programm):

k	UGS	UT	UDS	ID_ist	ID_soll	Fehler
3,3 $\frac{A}{V^2}$	1,7 V	1 V	0,4 V	0,641 mA	0,660 mA	-0.587%
289 $\frac{A}{V^2}$	2,4 V	1 V	3,8 V	277,429 mA	277,440 mA	-0,004%
140 $\frac{mA}{V^2}$	4,5 V	1 V	3,8 V	0,458 mA	0,468 mA	-2,187%

- Ergebnisse Aufgabenteil c (ursprüngliches Programm):

k	UGS	UT	UDS	ID_ist	ID_soll	Fehler
3,3 $\frac{A}{V^2}$	1,7 V	1 V	0,4 V	0,656 mA	0,660 mA	-2,899%
289 $\frac{A}{V^2}$	2,4 V	1 V	3,8 V	276,163 mA	277,440 mA	-0,460%
140 $\frac{mA}{V^2}$	4,5 V	1 V	3,8 V	0,366 mA	0,468 mA	-21,750%

Diskussion

- Mikrorechnerlösungen nutzen oft Festkommazahlen. Festlegung der Kommaposition, Überläufe, Rundungsfehler sind bereits bei einfachen Berechungen anspruchsvolle Aufgaben.
- Es gibt Programmgeneratoren zur Konvertierung von Gleitkomma- in Festkommaberechnungen, die den Programmierer von der Festlegung der Verschiebungen, Wertebereiche etc. entlasten.
- Diversitärer berechnete Ergebnisse sollten abweichende numerische Fehler haben. Erfordert einen Fenstervergleich. Die Größe des Vergleichsfensters sollte aus der Spezifikation hervorgehen.

Typ-Kontrollen

- Welche Fehler enthält das Programmskelett?
- Welche werden beim Übersetzen und welche werden zur Laufzeit erkannt?

```
float f_k, f_UGS, f_UT, f_UDS;

void do_calc_and_print_result(){
 int   k       = (f_k*(1<<24));
 ...
 long long int t1 = (k*UDS;
 long long int t2 =(t1>>32)*(UGS-UT-UDS/2);
 int            ID = t2 >> 20;
 float ID_soll= f_k * f_UDS*(f_UGS-f_UT-f_UDS/2.0);
 ...
 printf("ID_ist=%5.3dA ID_soll=%5.3fA\n",ID_ist, ID_soll);
}
```

Zeile 3: Typecast des Gesamtergebnisses nach int. Warnung.
Zeile 4: ein Typecast nach 64-Bit-Integer (long long int).
Zeile 5: Este Formatangabe »dezimal«, zugeordnet »float«. Warnung.

Diversität

Bereich der Anzahl der maskierten FF, viele Maskierungen

Für eine Fehlererkennung durch Verdopplung und Vergleich sei angegeben, dass die Diversität 90% betrage.

1. In welchem Bereich liegt unter dieser Annahme die Anzahl der nicht erkannten Fehlfunktionen für ξ =1.000 Fehlfunktionen? (Irrtumswahrscheinlichkeit $\alpha < 2\%$)
2. Was bedeutet hier Irrtumswahrscheinlichkeit <2%.
3. Wie groß ist die Anzahl ξ_{NErk} der maskierten Fehlfunktionen mit einer Irrtumswahrscheinlichkeit $\alpha < 2\%$ maximal?
4. Was bedeutet jetzt Irrtumswahrscheinlichkeit $\alpha < 2\%$?

Lösung Aufgabenteil 1 und 2

Diversität 90%, ξ =1.000 Fehlfunktionen, $\alpha < 2\%$ beiderseitig.
Gesucht: Bereich der Anzahl der nicht erkannten FF.

- Erwartungswert: $E\left(\xi_{\mathrm{NErk}}\right) = (1 - Div) \cdot \xi = 100$
- Standardabweichung:[2]: $\sqrt{D^2\left(\xi_{\mathrm{NErk}}\right)} \approx \sqrt{E\left(\xi_{\mathrm{NErk}}\right)} = 10$
- Poissonverteilung mit $E\left(\xi_{\mathrm{NErk}}\right) = 10 \cdot \sqrt{D^2\left(\xi_{\mathrm{NErk}}\right)}$ ist in guter Näherung normalverteilt. Irrtumswahrscheinlichkeit $\alpha < 2\%$ verlangt $\varepsilon_\sigma > 2{,}33$.
- Bereich der Anzahl der nicht erkannten FF: 100 ± 24

Was bedeutet hier Irrtumswahrscheinlichkeit <2%.

- Symmetrischer Bereich, d.h. ξ_{NErk} ist mit einer Wahrscheinlichkeit <1% kleiner und mit einer Wahrscheinlichkeit <1% größer als der angenommene Bereich.

[2] Zählprozess unabhängiger seltener Maskierungen führt auf eine Poisson-Verteilung.

Lösung Aufgabenteil 3 und 4

Wie groß ist die Anzahl ξ_{NErk} der maskierten Fehlfunktionen mit einer Irrtumswahrscheinlichkeit $\alpha < 2\%$ maximal?

- Einseitige Bereichschätzung mit $\alpha < 2\%$ verlangt $\varepsilon_\sigma > 2{,}02$.
- Erwartungswert und Standardabweichung sind wie in Aufgabenteil 1 $E\left(\xi_{\mathrm{NErk}}\right) = 100$ und $\sqrt{D^2\left(\xi_{\mathrm{NErk}}\right)} \approx 10$.
- Maximale Anzahl der nicht erkannten FF: 121

Was bedeutet jetzt Irrtumswahrscheinlichkeit $\alpha < 2\%$?

- Im Mittel treten Werte größer 121 nicht öfter als bei einem von 50 Versuchen auf. Die unter Bereichgrenze ist bei einseitiger Bereichsschätzung null, so dass zu kleine Werte ausgeschlussen sind.

Bereich der Anzahl der maskierten FF, seltene Maskierungen

Die Diversität sei $Div = 95\%$.

1. Von bis zu wie vielen Fehlfunktionen wird mit einer Irrtumswahrscheinlichkeit $\alpha < 10\%$ maximal eine Fehlfunktion maskiert?
2. Um welchen Faktor ist die Schranke größer als der Erwartungswert?

Lösung Teilaufgabe 1

Für geringe Zählwerte ist die Anzahl der Maskierungen poison-verteilt. Für »mit $\alpha < 10\%$ mehr als eine Masierung« muss gelten:

$$e^{-\xi\cdot(1-Div)} \cdot (1 + \xi \cdot (1 - Div)) > 1 - 10\%$$

Numerische Lösungssuche:

```
for xi=1:20
  alpha=1-exp(-0.05*xi)*(1+0.05*xi);
  printf('xi=%2d alpha=%4.1f%%\n', xi, 100*alpha);
  if alpha>0.1 break end
end
```

Ausgabe:

```
xi= 9 alpha= 7.5%
xi=10 alpha= 9.0% <= Lösung
xi=11 alpha=10.6%
```

Lösung Teilaufgabe 2

Um welchen Faktor ist die Schranke größer als der Erwartungswert?

- Erwartungswert $E(\xi_{\text{NErk}}) = \xi \cdot (1 - Div) = 0{,}5$ Maskierungen.
- Garantierbare $\alpha < 10\%$ garantierbare Schranke: 1 Maskierung, d.h. doppelt so groß.

Schätzen der Diversität

Eine Fehlererkennung durch Verdopplung und Vergleich hat von $\xi = 3000$ Fehlfunktionen $\xi_{\text{NErk}} = 289$ nicht erkannt.

1. Wie groß ist die zu erwartende Anzahl der nicht erkannten FF bei Irrtumswahrscheinlichkeit von $\alpha < 2\%$ maximal?
2. Wie groß ist die Diversität mindestens?

Lösung:

1. Die Differenz zwischen Erwartungswert und einer Realisierung normalverteilter ZG ist auch normalverteilt und hat dieselbe Standardabweichung wie die Realisierungen, im Beispiel abschätzungsweise $\approx \sqrt{\xi_{\text{NErk}}} = 17$. $\alpha < 2\%$ verlangt $\varepsilon_\sigma > 2{,}02$.
   Obergrenze: $\xi_{\text{NErk}} < 289 + 17 \cdot 2{,}02 = 323,34$
2. Mindestdiversität: $Div > 1 - \frac{323{,}34}{3000} = 89{,}2\%$

Programmieren mit Kontrollen

Aufgabe 3.1: Datencontainer mit Prüfsumme

Für folgende Datenstruktur

```
#define cLen 100
struct DFeld {
 unsigned char fill;// Füllstand des Feldes
 int feld[cLen];    // Feld
 int PSum;          // Prüfsumme
}
```

soll eine Funktion

```
int putVal(struct DFeld *DFptr, int val);
```

zum Ablegen einer Zahl zuzügl. Addition des Werts zur Prüfsumme und »fill+=1«, eine Funktion

```
int getVal(struct DFeld *DFptr, int val);
```

zur Entnahme einer Zahl zuzügl. Subtraktion des Werts von der Prüfsumme und »fill-=1«

und eine Kontrollfunktion

```
int checkDF(struct DFeld *DFptr);
```

mit einer Bereichkontrolle des Füllstands und einer Kontrolle der Prüfsumme programmiert werden. Die Rückgabewerte seien Fehlfunktionsnummern:

```
RC=0: keine Fehlfunktion
RC=1: Füllstand kleiner null
RC=2: Füllstand größer cLen
RC=3: abzuholender Wert nicht im Container
RC=4: Prüfsummenfehler
```

Implementierung des »Ablegens« und der »Entnahme«:

- »putVal()« soll den Wert immer an das Ende des Feldes schreiben,
- »getVal(...)« soll die erste Zahl mit dem abzuholenden Wert aus dem Feld löschen und die Lücke schließen.

Zahl ablegen und zur Prüfsumme addieren

```
#define cLen 100
struct DFeld {
 unsigned char fill;// Füllstand des Feldes
 int feld[cLen];    // Feld
 int PSum;          // Prüfsumme
}
```

```
int putVal(struct DFeld *DFptr, int val){
 if (DFptr->fill>=cLen) return 2; // Feld voll
 DFptr->feld[DFptr->fill] = val;
 DFptr->fill++;      // Füllstand erhöhen
 DFptr->PSum += val; // Prüfsumme aktualisieren
 return 0;
}
```

Die Prüfsummenaktualisierung kostet nur eine zusätzliche Programmzeile.

Zahl ablegen und von der Prüfsumme abziehen

```
int getVal(struct DFeld *DFptr, int val){
 unsigned char idx=0;
 while(1){              // Zahl suchen
  if (val==DFptr->feld[idx]) break;
  // wenn Zahl nicht vorhanden, dann RC=3
  if (idx>=DFptr->fill) return 3;
  idx++;
 };
 DFptr->fill--;         // Länge reduzieren
 DFptr->PSum -= val; // von der Prüfsumme abziehen
                        // alle Elemente danach eins vor
 while (idx<DFptr->fill){
  DFptr->feld[idx] = DFptr->feld[idx+1];
  idx++;
 }
 return 0;
}
```

Kontrollfunktion

```
int checkDF(struct DFeld *DFptr){
 unsigned char idx;
 if (DFptr->fill <0)    return 1; // negativer Füllstand
 if (DFptr->fill >cLen) return 2; // zu hoher Füllstand
 int sum=0;                      // Prüfsummenbildung
 for (idx=0; idx<DFptr->fill; idx++)
   sum += DFptr->feld[idx];
 if (sum != DFptr->PSum) return 4;// Prüfsummenfehler
 return 0;                       // Kontrolle bestanden
}
```

- Die Kontrolle sollte fast jede Verfälschung der Prüfsumme, des Füllstands und der Daten erkennen.
- Die Programmierung wirksamer Kontrollen erfordert einigen Aufwand.
- Dieser Aufwand wird durch einen geringeren Tests- und Debug-Aufwand kompensiert.

Fehlerbehandlung

Begriffsklärungen

Was bedeuten die Begriffe

- Fail Fast,
- Fail Save,
- Fail Slow,
- Fehlerisolation,
- statische Neuinitialisierung,
- dynamische Neuinitialisierung,
- Input-Workaround?

Aufgabe 3.14: Beispiele für die Fehlerbehandlung

Nennen Sie Beispiele (Ihnen bekannte Programme und Geräte) die folgende Techniken nutzen:

1. Zeitüberwachung mit Service-Abbruch bei Zeitüberschreitung.
2. Wiederholungsanforderung nach fehlerhaftem Datenempfang.
3. Systeme, bei denen sich Fehlverhalten durch andere Eingabereihenfolgen, Nutzung anderer Eingabemenüs etc. umgehen lassen.
4. Systeme, die beim Ausschalten automatisch ihre Bearbeitungszustand sichern.
5. Systeme, die nach einer Fehlfunktion vom letzten gesicherten Zustand starten.
6. Versenden von Fehlerinformationen an die Firma, die das System entwickelt hat.