Verlässliche Echtzeitsysteme

Übungen zur Vorlesung

Florian Franzmann Martin Hoffmann Isabelle Stilkerich

Friedrich-Alexander-Universität Erlangen-Nürnberg
Lehrstuhl Informatik 4 (Verteilte Systeme und Betriebssysteme)
http://www4.cs.fau.de

20. Juni 2013

Überblick

Nächster Übungstermin

- nächster *regulärer Übungstermin* erst am 11. Juli
- bis dahin sind noch der 27.6. und der 4.7. Donnerstage
- wir stehen auf jeden Fall für Rechnerübungen zur Verfügung
- könnten aber auch Themen noch einmal besprechen
- oder demonstrieren
- git/Gerrit wäre ein heisser Kandidat ☺

Themenwünsche?

Table of Contents

Annahmen

- C99
- x86 bzw. x86-64, d. h.
  - vorzeichenbehaftete Integer als Zweierkomplement implementiert
  - `char` hat 8 Bit
  - `short` hat 16 Bit
  - `int` hat 32 Bit
  - `long` hat 32 Bit auf x86 und 64 Bit auf x86-64

Frage 9

Angenommen `x` hat Typ `int`. Ist `x << 0` ...

1. definiert für alle Werte
2. definiert für manche Werte
3. definiert für keinen Wert

von `x`?

Erklärung

- negative Werte können nicht nach links verschoben werden
- noch nicht einmal um `0` Bit

Frage 10

Angenommen `x` hat Typ `int` und ist positiv. Ist `x << 1` ...

1. definiert für alle Werte
2. definiert für manche Werte
3. definiert für keinen Wert

von `x`?

Erklärung

- Es darf nicht in das Vorzeichenbit hineinverschoben werden

$\Rightarrow$ nicht definiert für große Werte von `x`

Frage 11

Angenommen `x` hat Typ `int`. Ist `x << 31` ...

1. definiert für alle Werte
2. definiert für manche Werte
3. definiert für keinen Wert

von `x`?

Erklärung

- Es darf nicht in das Vorzeichenbit hineinverschoben werden

$\Rightarrow$ funktioniert hier nur mit `x == 0`

Frage 12

Angenommen `x` hat Typ `int`. Ist `x << 32` ...

1. definiert für alle Werte
2. definiert für manche Werte
3. definiert für keinen Wert

von `x`?

Erklärung

- Verschiebung um Bitbreite eines Datentyps nicht zulässig

Frage 13

Angenommen `x` hat Typ `short`. Ist `x << 29` ...

1. definiert für alle Werte
2. definiert für manche Werte
3. definiert für keinen Wert

von `x`?

Erklärung

- Vor der Verschiebeoperation wird nach `int` umgewandelt
- Verschiebung um mehr als die Bitbreite ist also kein Problem

Frage 14

Angenommen `x` hat Typ `unsigned`. Ist `x << 31` ...

1. definiert für alle Werte
2. definiert für manche Werte
3. definiert für keinen Wert

von `x`?

Erklärung

- jeder Wert, dessen *promoted type* `unsigned` ist kann um nichtnegativen Wert verschoben werden
- solange die Bitbreite nicht erreicht wird

Frage 15

Angenommen `x` hat Typ `unsigned sort`. Ist `x << 31` ...

1. definiert für alle Werte
2. definiert für manche Werte
3. definiert für keinen Wert

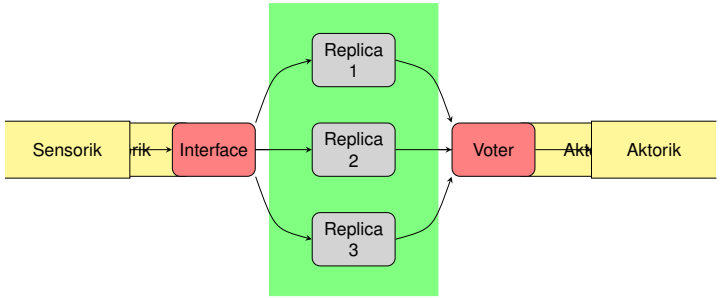
von `x`?

Erklärung

- `unsigned short` wird nach `int` umgewandelt
- eine `1` darf nicht in das Vorzeichenbit hineinverschoben werden
- Verschiebung um bis zu 15 wäre immer in Ordnung

Table of Contents

Klassische “Triple Modular Redundancy” (TMR)

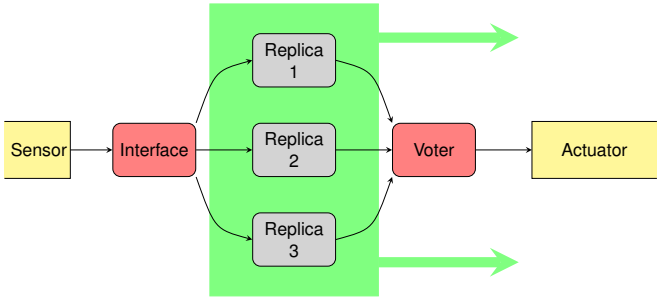


- Schnittstelle sammelt Eingangsdaten (Replikationsdeterminismus)
- Verteilt Daten und aktiviert Replikate
- Mehrheitsentscheider (Voter) wählt Ergebnis
- Ergebnis wird an Aktuator versendet

Redundanzbereich

Ausschließlich Replikatausführung.

Erweiterung I – kodierte Ausgangswerte



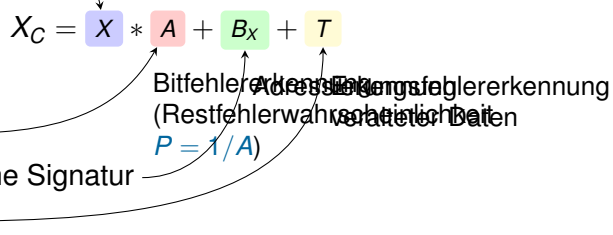
- Erweiterung der Ausgangsseite mit Informationsredundanz
- Mehrheitsentscheid über kodierte Prüfsumme

Table of Contents

Erweiterte arithmetische Kodierung

nach Forin 1989: “Vital coded microprocessor principles and application for various transit systems” [1]

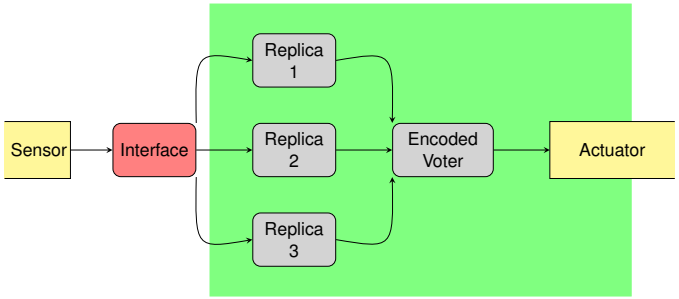
- Arithmetisch kodierter Wert X_C
- Ausgangswert
- (Prim-)Zahl
- Variablenspezifische Signatur
- Zeitstempel



Wertebereichseinschränkungen

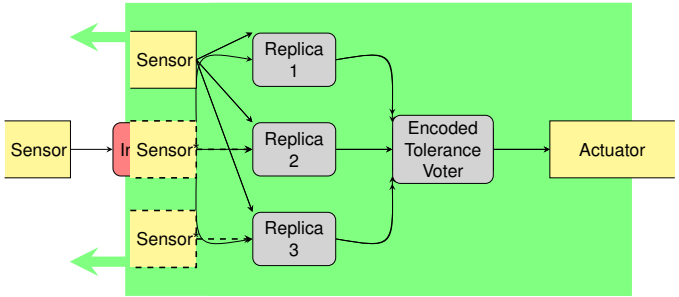
- (Prim-)Zahl A sollte so groß wie möglich sein:
  - ⇝ Möglichst geringe Restfehlerwahrscheinlichkeit ($P = 1/A$)
- Wertebereich des dynamischen Zeitstempels
  - $T = \{x | x \in \mathbb{N}_0 \wedge x \leq D_{max}\}$
  - Zeitstempel darf überlaufen: $D_{max} + 1 = 0$
- Für jede Signatur B_* muss dann gelten
  - $B_* + D_{max} < A$
  - Die minimale Distanz zwischen jeweils zwei Signaturen im System muss kleiner D_{max} sein: $\forall i, j : |B_i - B_j| < D_{max}$

Erweiterung I – kodierte Ausgangswerte



- Replikate liefern arithmetisch kodierte Ergebnisse
- Mehrheitsentscheid auf kodierten Prüfsummen
- Übertragung kodierter Ergebnisse

Erweiterung II – Datendiversität



- Replikate ermitteln Eingangsdaten selbständig
- Diversitäre Eingangsdaten
  - Unterschiedliche Messzeitpunkte (zeitliche Redundanz)
  - Redundante Sensoren (physikalische Redundanz)
- Mehrheitsentscheid mittels Toleranzbereich (Tolerance Voter)

Vereinfachung für diese Übung

Für diese Übungsaufgabe:

- Keine Datendiversität am Eingang
- Nur Absicherung der Ausgangsseite!

EAN Vergleichsoperator

- Voting basiert auf kodierter Vergleichsoperation:

$\rightsquigarrow X_C = Y_C \Rightarrow X * A + B_X + T_X = Y * A + B_Y + T_Y$

- Im fehlerfreien Fall gilt:

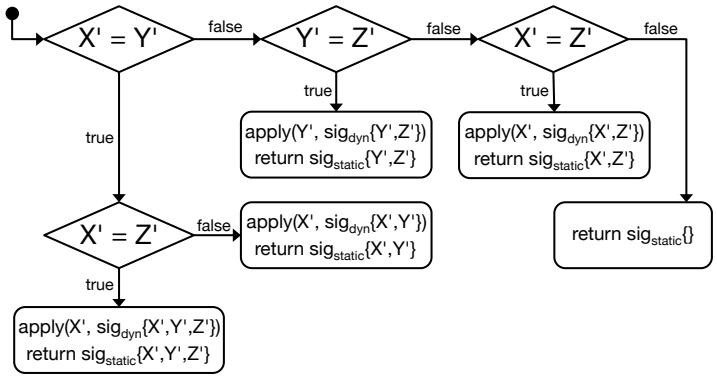
$X = Y$, $T_X = T_Y$, $A = A$ aber $B_X \neq B_Y$!

- *Rohwerte* sind identisch
- Ergebnisse sind aktuell
- (Prim-)Zahl ist per Definition identisch
- Signaturen sind unterschiedlich (aber konstant!)

Bestimmung der Gleichheit durch Differenzbildung:

$\rightsquigarrow X_C - Y_C = B_X - B_Y = const.$

Kodierter Mehrheitsentscheid



- Bestimmung von dynamischer und statischer Signatur:

$\rightsquigarrow sig_{dyn}(X', Y') : X' = Y' \Rightarrow X' - Y'$

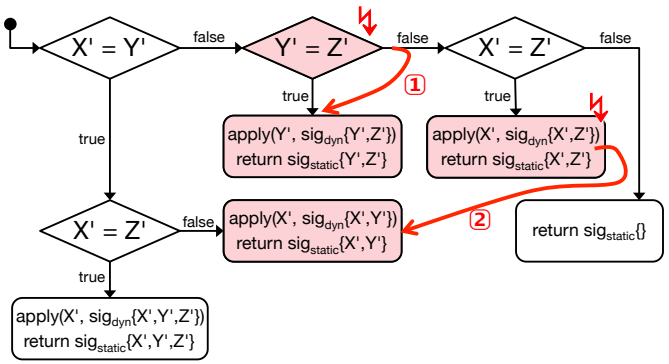
$\rightsquigarrow sig_{static}(X', Y') : X' = Y' \Rightarrow B_X - B_Y$

Kodierter Mehrheitsentscheid (Forts.)

1. Vergleichsoperation wird durchgeführt (z. B. $X' = Y' \wedge X' = Z'$)
   - Berechnung von sig_{dyn}
   - Vergleich mit sig_{static}
2. Verzweigunsentscheidung wird nachberechnet:
   - Wiederholte (redundante) Berechnung von sig_{dyn}
   - Addiere sig_{dyn} (*apply*) zum gewählten Ergebnis
3. Konstante Signatur des durchlaufenen Zweiges identifiziert Gewinner (Rückgabewert: sig_{static})
   - Aktor wählt entsprechendes Replikatergebnisse
   - führt inverse Operation zu *apply* durch

Im Voter wurde die *dynamisch berechnete Signatur der Verzweigungsentscheidung* hinzuaddiert. Im Aktor wird mit der entsprechenden *konstanten Signatur zurückgerechnet*.

Kodierter Mehrheitsentscheid - Fehlerfall



1. Falsche Verzweigungsentscheidung: ($Y' \neq Z'$)
   - Y' wird als korrekt angenommen, sig_{dyn} wird erneut berechnet
   - allerdings ist sig_{dyn} tatsächlich $\neq sig_{static}$
   - Fehler wird bei der inversen Operation zu *apply* erkannt
2. Falscher (plötzlicher) Sprung
   - X' wird als korrekt erkannt, sig_{dyn} wird erneut berechnet
   - Ein fehlerhafter Sprung in einen anderen Block führt zu einem inkonsistenten Rückgabewert $sig_{static}\{X', Z'\}$
   - $sig_{dyn}\{X', Y'\} \neq sig_{static}\{X', Z'\}$ wird beim dekodieren erkannt

Table of Contents

Aufgabenstellung

Aufgabe

Erweitern Sie Ihre Software-TMR Implementierung um einen EAN-kodierten Voter für die (Festkomma)-Filterimplementierung aus der vorherigen TMR-Aufgabe.

- Jeder Filterschritt entspricht dabei einer Zeiteinheit
  - ⇝ Nutzen Sie einen globalen Zähler als Zeitgeber für T
  - Begrenzen Sie dabei T auf den Wertebereich $0 \ldots D_{max}$
- Jedes Replikat hat genau einen Ausgabewert (integer)
  - ⇝ Legen Sie für jede der drei Ausgabewerte (X', Y', Z') jeweils *unterschiedliche* aber *konstante* Signaturen (B_X, B_Y, B_Z) fest
- Nutzen Sie für X' den nächstgrößeren Datentyp zu X
  - ⇝ Wählen Sie eine Zahl A mit möglichst großem Hamming-Abstand, *vermeiden Sie* dabei mögliche *Überläufe bei der Kodierung*

Hinweise

- In dieser Aufgabe betrachten wir nur die Ausgangsseite
- Nach dem Voting kann das Interface das Ergebnis gleich dekodieren/validieren
- Die Eingangsseite bleibt vorerst „ungeschützt“
- die besten Kandidaten für A bei 8 Bit-Zahlen sind 185 und 233
  ⇝ Hamming-Distanz vier
- die besten Kandidaten für 16 Bit sind 58659, 59665, 63157, 63859 und 63877
  ⇝ Hamming-Distanz sechs

Für jede Operation zwischen zwei codierten Werten

ist eine eigene Funktion mit konstanten Signaturwerten notwendig!

Literatur

[1] Forin.
Vital coded microprocessor principles and application for various transit systems.
IFA-GCCT, pages 79–84, 1989.

Fragen?