AUFGABE 6: TMR UND EAN

In dieser Aufgabe implementieren Sie TMR-Varianten für einen gegebenen Filteralgorithmus. Die tatsächliche Funktionalität des Filters ist für die Bearbeitung der Aufgabe nicht relevant. Achten Sie nur auf die korrekte Verwendung (Aufruf, Eingabe/Ausgabewerte).

Die Effektivität Ihrer Implementierung wird mittels eines Fehlerinjektionsexperiments getestet. In der Vorgabe finden Sie eine vollständige, allerdings noch ungesicherte, Ausführung des Algorithmus.

Achten Sie bitte auf die Kommentare in den Vorgaben, welche Stellen angefasst werden sollen/dürfen und welche unangetastet bleiben sollen.

Setzen Sie zunächst die notwendigen Umgebungsvariablen:

```
cd 06_TMR; source ecosenv.sh
```

Aufgabenstellung

1. *Ungesichertes System:* Kompilieren Sie die Vorgabe. Sichern Sie die Ergebnisse unter dem Variantennamen *plain*:
`ccmake .` → `FAILVARIANT`
Führen Sie die Injektion, wie in der Übung vorgestellt durch und betrachten Sie die Ergebnisse. *An welchen Stellen entstehen die meisten Fehler?* Versuchen Sie anhand des Dumps die entsprechenden Stellen im C-Code zu identifizieren.

☞ `src/app_plain.c`

☞ `make fail-1-trace`
...
`make fail-5-browse`
☞ `tt_scope.dis`

2. *Fehlerhypothese:* Bevor Sie anfangen die Filterimplementierung gegen transiente Fehler zu schützen, benötigen Sie eine Fehlerhypothese. Analysieren Sie die Filterimplementierung mit Hilfe der *Fehlerbaummethode* und halten Sie das Analyseergebnis schriftlich fest. *An welchen Stellen können Fehler auftreten?*

3. *TMR-Ausführung:* Entwerfen Sie eine dreifach redundante Ausführung des Filteralgorithmus. Ändern sie entsprechend

☞ `src/app_tmr.c`

`src/app_plain.c` in `CMakeLists.txt` in `src/app_tmr.c` um, um die korrekte Datei zu übersetzen. Behalten Sie die Schnittstelle von `convolve()` bei, Rückgabewert und Parameter von `process()` können Sie nach belieben anpassen!
Replizieren Sie die Eingabedaten auf geeignete Weise und entwerfen Sie einen Votingalgorithmus `vote()` zum Vergleichen der Ergebnisse. Auch hier haben Sie freie Hand bei Parameter- oder Rückgabewahl.
Benennen Sie die Variante als *TMR_base* (`ccmake .` → `FAILVARIANT`) und führen Sie eine erneute Fehlerinjektion durch. ☞ `make fail-1-trace`, `make fail-2-import`, etc.
Konnten Sie die *NEGATIVE MARKER* verringern? An welchen Stellen finden sich noch unbemerkte Fehler? *Um welche Art von Test handelt es sich bei Ihrem Voter? Welche neuen möglichen transienten Fehler sind hinzugekommen? Welche Teile Ihres Fehlerbaums haben Sie durch die Replikation abgedeckt?*

4. *SoR:* Bestimmen Sie die durch Redundanz gesicherten Bereiche in ihrem Programm (→ Sphere of Redundancy) und markieren Sie diese durch Kommentare in ihrem Quellcode.
Können eventuell noch ungeschützte Stellen in ihrer Implementierung in den Redundanzbereich aufgenommen werden? Versuchen Sie *Silent Data Corruptions* weiter zu minimieren.
Implementieren Sie ihre optimierte Variante in und verwenden Sie den Fail*-Variantennamen *TMR_opt* für die Fehlerinjektion. ☞ `src/app_tmr_opt.c`

5. *CoRed-Voter:* Schützen Sie nun Ihren Voter mit Hilfe des in der Tafelübung vorgestellten Verfahrens der *Erweiterten Arithmetischen Codierung*. Kopieren sie dafür Ihre Datei `src/app_tmr_opt.c` nach `src/app_tmr_ean.c` und implementieren Sie hier ihren codierten Voter. Achten Sie auch auf die erneute Anpassung Ihrer `CMakeLists.txt`. *Welche Teile des Fehlerbaums sind nun abgedeckt? Was ist nun immer noch ungeschützt? Mit welchem Verfahren könnte man den verbleibenden Teil der Impementierung schützen?*

Die Fehlerinjektion kann durchaus einige Zeit in Anspruch nehmen. Das ungesicherte System benötigt auf acht Kernen ca. 10 Minuten. Nutzen Sie die Zeit, um sich bereits Gedanken über mögliche Verbesserungen zu machen.

Hinweise

- Bearbeitung: Gruppe mit je zwei bis drei Teilnehmern.
- Abgabezeit: 7.7.2015
- Fragen bitte an `i4ezs@lists.informatik.uni-erlangen.de`