Sebastian Huhn, huhn@informatik.uni-bremen.de, MZH 4280
Jil Tietjen, Jil.Tietjen@dfki.de, MZH 4208

Programmieraufgabe

Test von Schaltungen und Systemen

Aufgabe 1 (Zulassung zum Fachgespräch)

Als Zulassungsvoraussetzung zum Fachgespräch muss die folgende Aufgabe erfolgreich bearbeitet werden. Die Bearbeitung kann einzeln oder zu zweit erfolgen. Im Falle einer gemeinsamen Bearbeitung muss der Beitrag jedes Einzelnen klar erkennbar sein. Zusätzlich ist eine Präsentation (inkl. Foliensatz) im Rahmen des Tutoriums obligatorisch, welche eine Dauer von 10 Minuten nicht überschreiten soll. Hierbei sollen nicht nur die berechneten Kennzahlen, sondern ebenfalls die abstrakte Modellierung und der daraus resultierende Implementierungsansatz vorgestellt werden.

Die vollständige Abgabe (inkl. Quellcode und Dokumentation) muss spätestens am Sonntag, 12. Januar 2020 um 23:59 Uhr per Email an huhn@informatik.uni-bremen.de erfolgen.
Der primäre Präsentationstermin ist auf das Tutorium am Dienstag, 29.01.2020 und der Ausweichtermin auf das Tutorium am Dienstag, 21.01.2020 datiert. Die Präsentation ist obligatorisch; ohne erfolgte Präsentation ist keine Zulassung zum Fachgespräch möglich.

Die Programmieraufgabe ist in drei Teile gegliedert. Das Resultat ist ein Schaltkreis-Simulator für einfache *Stuck-at* (SA)-Fehler. Hierbei sollen *Stuck-at-0* (SA-0) sowie *Stuck-at-1* (SA-1) Fehler unterstützt werden.

a) Es ist ein Parser für das `bench` Format sowie eine passende Datenstruktur zur Repräsentation des Schaltkreises zu entwerfen. Des Weiteren muss ein Parser für die Testmuster Datei im `vec` Format und eine adäquate Datenstruktur für die Verarbeitung dieser Testmustermenge realisiert werden.

b) Für den eingelesenen Schaltkreis ist eine vollständige Fehlerliste bzgl. des SA-Fehlermodells zu erstellen und diese in einer geeigneten Datenstruktur abzuspeichern. Hierbei ist die Implementierung weiterer Optimierungstechniken optional, die bspw. Techniken wie die Fehlerkollabierung realisieren. (Für die Berechnung der u.g. Metriken muss jedoch von der vollständigen Fehlerliste ausgegangen werden.)

c) Zusätzlich muss ein Simulationsmodell generiert werden, sodass eine Simulation jedes Testmusters aus der gegebenen Testmustermenge möglich ist und die überdeckten SA-Fehler (aus der erstellten Fehlerliste) bestimmt werden können. Abschließend soll die Berechnung der Fehlerüberdeckung erfolgen.
Hinweis: Es muss lediglich ein Zeittakt pro Testmuster betrachtet werden.

Das Programm erhält als Parameter den Pfad zu einer `bench` Datei, die den Schaltkreis beschreibt. Zusätzlich wird dem Programm einen weiteren Pfad zu einer `vec` Datei übergeben. Diese `vec` Datei enthält alle Testmuster für **einen einzelnen** Schaltkreis.

Als Rückgabe liefert das Programm für **jeden** Schaltkreis über die Standardausgabe `stdout` ein Tupel aus Gleitkommazahlen zurück, welches die folgenden Werte beinhaltet:

- Gesamtzahl an SA-Fehlern in der erzeugten Fehlerliste,
- Anzahl an detektierten SA-Fehlern durch die gegebene Testmustermenge und
- Prozentuale Fehlerüberdeckung bzgl. SA-Fehlermodells.

Rahmenbedingungen: Das Programm soll in C++ implementiert und kompakt, jedoch verständlich, dokumentiert werden. Das Programm muss auf den X-Rechnern kompilierbar sein, wobei als Referenz ein *Fedora 30* Betriebssystem anzunehmen ist.

Ein Build-System mittels *Make* (besser: *CMake*) ist optional, jedoch wünschenswert. Andernfalls sind die exakten Compiler- bzw. Linker-Aufrufe zur Erzeugung der Binärdatei anzugeben. Ausgewählte Beispiel-Schaltkreise im `bench` Format und die dazugehörigen `vec` Dateien befinden sich auf der LV-Webseite der AG Rechnerarchitektur und im Stud.IP und müssen durch das Programm erfolgreich verarbeitet werden können.

Hinweis: Sequentielle Elemente (Flip-Flops) werden im Bench Format wie folgt repräsentiert: `out = DFF(in)`. Im eingelesenen Schaltkreis können diese Flip-Flips für die Simulation wie folgt behandelt werden: Eingehende Signale des Flip-Flops als (pseudo) primäre Ausgänge und ausgehende Signale des Flip-Flops als (pseudo) primäre Eingänge.