Algorithmen für Chip-Entwurfswerkzeuge Aufgabe 1 — Schaltungsanalyse

TECHNISCHE UNIVERSITÄT DARMSTADT

Embedded Systems & Applications

Abgabe bis zum 2012-11-12

Florian Stock, Prof. Koch

Es soll eine Java-Klassenbibliothek zur Bearbeitung von Netzlisten und Layouts auf einer parametrisierten FPGA-Architektur erstellt werden. In einer ersten Anwendung sollen auf Basis dieser Bibliothek bestehende Layouts in Bezug auf Korrektheit und Zeitverhalten analysiert werden.

Inhaltsverzeichnis

1.1 Einleitung

Um den Umgang mit den verschiedenen Dateiformaten, einfachen Graphalgorithmen sowie der im Praktikum verwendeten FPGA-Architektur zu erlernen, sollen Sie ein Programm zur Analyse und Prüfung von bestehenden platzierten und verdrahteten Schaltungen schreiben.

Dieses Programm wird Ihnen in späteren Praktikumsphasen noch gute Dienste leisten, da Sie dann *selber* Platzierungs- und Verdrahtungswerkzeuge realisieren werden. Da die manuelle Begutachtung der Ergebnisse dieser Werkzeuge nur bei sehr kleinen Schaltungen erfolgversprechend ist, werden Sie auf die zuverlässige automatische Prüfung angewiesen sein.

1.2 Problemstellung

Ihr Programm soll als Eingabe eine Netzliste, eine Architekturbeschreibung, eine Platzierungsdatei und optional auch noch eine Verdrahtungsdatei bekommen. Weiterhin soll es mindestens die in Abschnitt 12.5.1 des Leitfadens genannten Parameter zur Modifikation der FPGA-Architektur bearbeiten. Mit diesen Angaben sollen folgende Operationen durchgeführt werden:

Überprüfung der Platzierung Alle Blöcke (Logik- und Ein-/Ausgabeblöcke) aus der Netzlistendatei müssen tatsächlich in der Platzierungsdatei platziert worden sein. Jede Platzierungsposition darf dabei maximal nur einmal verwendet werden (keine Überlappung von Blöcken). Fehler sind durch geeignete Meldungen auszugeben.

Überprüfung der Verdrahtung Falls eine Verdrahtungsdatei angegeben wurde, ist diese auf folgende Bedingungen zu prüfen:

- Genau eine Quelle pro Netz.
- Mindestens eine Senke pro Netz.
- Alle Teile eines Netzes sind tatsächlich miteinander verbunden (es gib einen Weg durch die angegebenen **`IPIN`**, **`CHANX`**, etc. von der Quelle zu allen Senken).
- Verdrahtungsressourcen wurden nicht mehrfach benutzt.

Konsistenzprüfung der Verdrahtung Falls eine Verdrahtungsdatei angeben wurde, muss die dort beschriebene Verdrahtung mit der in der Netzliste spezifizierten Konnektivität verglichen werden. Dazu müssen aus den gestückelten **`CHANX`** etc. Angaben in der Verdrahtungsdatei wieder ganze Netze extrahiert und diese mit dem in der Netzliste gegebenen Verlauf verglichen werden. In der Verdrahtung dürfen nur genau die Blockein- und ausgänge miteinander verbunden worden sein, die in der Netzliste gefordert waren. Jede Abweichung (nicht mehr bestehende Verbindungen oder Kurzschlüsse zwischen Netzen) muss als Fehlermeldung ausgegeben werden. Bitte beachten Sie, dass bei unserer FPGA-Architektur die Eingangspins der Logikblöcke äquivalent sind. Das heisst, lediglich die Tatsache, *dass* bestimmte Netze an die Eingänge des Logikblocks angeschlossen sind, ist relevant. Nicht aber, an welche speziellen *Pins* sie angeschlossen sind.

Timing-Analyse In den nächsten Phasen werden Sie eine Timing-Analyse zur Beurteilung der Verzögerungszeit Ihrer bearbeiteten Schaltungen benötigen. Deren Grundlagen sollen schon hier implementiert werden: Falls eine Verdrahtungsdatei vom Benutzer angegeben wird, soll eine einfache Timing-Analyse gemäß dem Modell aus Abschnitt 2.3 des Leitfadens durchgeführt werden. Dabei soll nur einer der kritischen Pfade ausgegeben werden. Dies kann z.B. in dem im Abschnitt 3.4 des Leitfadens vorgeschlagenen Format geschehen. Verwenden Sie zur Berechnung das in der Vorlesung vorgestellte Verfahren (Annotation von T_a und T_r, kritische Verbindungen haben slack $= 0$).

1.3 Abgabe

Gemäß den Anforderungen im Leitfaden. Die Hauptklasse (mit der Funktion **`main`**) *muss* **`DesignAnalyzer`** heissen. Damit *muss* ein Programmaufruf der Form

```
java DesignAnalyzer s27.net prak10.arch s27.p s27.r -X 8 -Y 8
```

möglich sein. Weichen Sie nicht von diesen Vorgaben ab, da unsere automatischen Testskripte Ihre Abgabe sonst als fehlerhaft einstufen. Erweiterungen dazu (CLASSPATH etc.) dokumentieren Sie bitte auch im **`README`**.

1.4 Kolloquium und Vortrag

Am Mittwoch, dem 13.11.2012, findet mit den Gruppen ein jeweils 30-minütiges Einzelkolloquium statt. Den genauen Zeitpunkte werden in der Vorlesung vergeben.

In der Vorlesungszeit am Freitag, dem 15.11.2010, findet dann in der Vorlesungszeit eine zentrale Besprechung statt. Hier werden alle Gruppen in 10-15-minütigen Vorträgen über ihre Lösung der Aufgabe referieren (auch hier: Anforderungen siehe Leitfaden).

1.5 Hilfestellung

1.5.1 Gruppenarbeit

Durch die Komplexität der Aufgaben und die vorgesehen Abgabetermine ist eine echte Gruppenarbeit unerläßlich. Diese erste Aufgabe läßt sich beispielsweise aufteilen in (in aufsteigender Schwierigkeit angegeben):

a) Dateioperationen: Einlesen der Netzlisten, Platzierungs- und Verdrahtungsdaten

b) Timing-Analyse: Bestimmen der Verzögerungszeiten.

c) Prüfoperationen: Prüfen von Platzierung und Verdrahtung, Vergleich mit Netzliste.

Voraussetzung für eine solche Aufteilung ist, dass sich die Gruppe *vorher* auf eine gemeinsame Datenstruktur geeinigt hat, die dann von allen Mitgliedern benutzt wird.

Andere Aufteilungen sind natürlich auch denkbar. Dazu gehören auch Varianten wie

a) Programmierung

b) Erstellen von Testdaten und Durchführen der Tests

c) Dokumentation und Vortrag

Gleich welche Arbeitsteilung Sie auch verwenden: Die Aufgaben sind vom Umfang und Bearbeitungszeitraum auf Gruppenarbeit ausgelegt. Bitte sprechen Sie daher mit dem Dozenten Probleme, verursacht beispielsweise durch Trittbrettfahrer oder davonpreschende Arbeitstiere, *frühzeitig* an!

1.5.2 Implementierung und Test

Bei der Implementierung ist die Verwendung der Java 2 Collection Classes sehr hilfreich. Auf diese Weise müssen Basisdatenstrukturen wie Listen, Mengen etc. nicht mehr neu erstellt werden. Konstrukte wie Iteratoren erlauben die Bearbeitung dieser Typen in übersichtlicher und konsistenter Weise.

Auf der Web-Seite der Veranstaltung finden Sie in zwei TGZ-Archivdateien Beispielschaltungen (siehe auch Leitfaden, Abschnitt 7). Beachten Sie für diese erste Aufgabe: Diese Testdaten sind fehlerfrei! Zum Test Ihres ersten Programmes müssen Sie also **selber** sinnvolle Fehlerfälle erzeugen. Erfahrungsgemäß wird die Endnote für diese Aufgabe eng mit der Qualität Ihrer Testfälle korrelieren. Stecken Sie daher bitte ausreichend Mühe in die methodische Entwicklung einer geeigneten Testsuite von Fehlerfällen!

1.6 Hinweise zum Thema Plagiarismus

Im Rahmen dieser Veranstaltung wird eine vorher festgelegte Arbeitsgruppe bewertet. Fremde Code-Bibliotheken dürfen Sie nur für die Realisierung nebensächlicher Funktionen verwenden (z.B. log4j für Logging, ANTLR für Lexer/Parser, etc.), wobei Sie alle diese externen Quellen korrekt zitieren müssen. Bitte fragen Sie in Zweifelsfällen bei Ihrem Betreuer nach! Zusammenarbeit über Gruppengrenzen hinweg ist in Form der wechselseitigen Vorträge und durch Diskussion (auch im Forum) von Lösungs*ideen* erlaubt. Es dürfen aber keine Artefakte wie Programm-Code, Dokumentationsteile (Text, Zeichnungen) oder ähnliches ausgetauscht werden.

Der Fachbereich Informatik misst der Einhaltung der Grundregeln der wissenschaftlichen Ethik großen Wert bei. Mit der Abgabe einer Lösung (Hausaufgabe, Programmierprojekt, etc.) bestätigen Sie, dass Ihre Gruppe die alleinigen Autoren des gesamten Materials sind. Falls Ihnen die Verwendung von Fremdmaterial gestattet war, so müssen Sie dessen Quellen deutlich zitiert haben. Weiterführende Informationen zu diesem Thema finden Sie unter **http://www.informatik.tu-darmstadt.de/Plagiarism**.