Algorithmen für Hardware-Entwurfswerkzeuge

TECHNISCHE UNIVERSITÄT DARMSTADT

Aufgabe 4 — Verbesserungen und Dokumentation

Abgabe bis zum 2019-03-06, 23:59 MET
Florian Stock

Die von Ihnen bisher entwickelten Programme sollen in *allen* Bereichen Designanalyse, Placement und Routing verfeinert und in der endgültigen Version umfassend dokumentiert werden.

Inhaltsverzeichnis

2 Einleitung

In dieser Phase der praktischen Arbeiten haben Sie Gelegenheit, Ihre Ideen zur Verbesserung der bisherigen Algorithmen umzusetzen. Da nach jeder der Phasen 1 bis 3 von allen Teilnehmern dazu ein breites Ideenspektrum geäußert wurde, sollte es an konkreten Ansatzpunkten nicht mangeln.

3 Mögliche Problemstellungen

Zu diesen Ansatzpunkten gehören unter anderem:

- Das Beseitigen von bekannten Programmierfehlern.
- Das Ersetzen von vorläufigen Routinenprototypen durch die ursprünglich geplanten aber noch unrealisierten Endversionen.

- Die Untersuchung und Beseitigung von Ineffizienzen sowohl in Bezug auf Laufzeit als auch auf Speicherbedarf. Beides unterstützt durch ein Profiling des Programmes.
- Der Einbau von neuen Kriterien in die Optimierung. Dazu gehören beispielsweise ein Timing-Driven Mode (via Criticality) beim Routing oder eine verdrahtungsorientierte Platzierung. Für letztere wurden verschiedene Abschätzungsverfahren in der Vorlesung vorgestellt.

4 Abgabe

4.1 Programmquellen

Gemäß den Anforderungen des Leitfadens. Dabei muss der *gesamte* Funktionsumfang der bisherigen Phasen 1 bis 3 enthalten sein. Die Aufrufe der Unterfunktionen Designanalyse, Platzierung und Verdrahtung erfolgen analog zu den früheren Aufgabenstellungen. Alle Teile des Codes *müssen* weitgehend die im Leitfaden genannten Programmierrichtlinien einhalten.

Dabei sind auch die Blockkommentare am Anfang jeder Datei sowie genaue Angaben über den Autor der Klasse erforderlich. Falls mehrere Autoren an einer Datei oder an unterschiedlichen Methoden beteiligt waren, so ist diese Tatsache sowie die Art des jeweiligen Beitrags auch durch Kommentare zu beschreiben.

4.2 Dokumentation

In einer 20-30 Seiten langen Dokumentation ist das *gesamte* Projekt umfassend zu beschreiben. Zu den zu betrachtenden Aspekten gehören:

- Benutzerhandbuch, u.a. mit Beschreibung von Aufrufparametern
- Algorithmische Grundlagen
- Datenstrukturen und ihre Zusammenhänge untereinander
- Konkrete Realisierung in Java
- Profiling-Daten für die verschiedenen Betriebsmodi (sowohl in Bezug auf Speicher als auch Ausführungszeit)
- Messergebnisse analog zu Phase 1 bis 3 für die verschiedenen Betriebsmodi bei Anwendung auf die Beispielschaltungen s27 bis clma
- Die Parameterkurven des Simulated Annealing (mindestens mit den Daten Gesamtkosten, Akzeptanzrate und Bewegungsradius bei jedem Schritt) nur für die Schaltung clma
- Vergleich der Messergebnisse nach der Verfeinerung in Aufgabe 4 mit den Ergebnissen der Vorversionen aus den Phasen 1 bis 3

Zur Darstellung der teilweise komplexen Zusammenhänge ist es unerlässlich, dass Sie diese durch Zeichnungen veranschaulichen. Dabei ist es Ihnen freigestellt, ob Sie eine lesbare und konsistente eigene Notation oder eine Standardnotation wie UML verwenden. Seitenweise Zeichnungen reichen aber alleine *nicht* aus. Sie müssen durch entsprechende Beschreibungen im Fliesstext erläutert und in den Kontext der Gesamtdokumentation (Algorithmenbeschreibung etc.) eingebettet werden.

Die Dokumentation soll als *eine* (1) konsistente Datei (fortlaufende Seitenzahlen, Inhaltsverzeichnis, etc.) im Adobe Portable Document Format (PDF) abgegeben werden. Bei Fragen zu der Erstellung von PDF wenden Sie sich ggf. an Kommilitonen oder den Betreuer.

> Fehlende, offensichtlich fehlerhafte oder unvollständige Dokumentation wird die Verweigerung der Abnahme und die Forderung von Nachbesserungen zur Folge haben.

4.3 Ergebnisse

Weiterhin legen Sie bitte für die Beispielschaltungen s27 bis clma folgendes Ihrer Abgabe bei:

- Die von Ihrem Placer erzeugten `.p` Dateien.
- Die von Ihrem Router erzeugten `.r` Dateien sowohl für die von Ihrem Programm erzeugten `.p` Dateien als auch für die Musterlösungen (Dateien im Archiv auf dem Web-Site).
- Die von Ihrer Designanalyse für jede `.p`/`.r` Kombination *geschätzte* $D_{\max}$ (in ns) und Verdrahtungslänge (als Anzahl von Segmenten).
- Das nach dem Verdrahten nun *exakt* berechnete $D_{\max}$ (in ns) und die exakte Verdrahtungslänge (als Anzahl von Segmenten).
- Die minimale Track-Anzahl pro Kanal, mit der Ihr Programm die Schaltung erfolgreich verdrahten konnte.
- Die Laufzeit Ihrer drei Programme für die Beispiele (in s). Dabei geben Sie für den Router die Zeit für die komplette binäre Suche zum Bestimmen der minimalen Track-Anzahl an, nicht nur die eines einzelnen Routing-Laufes mit vorgegebener Track-Anzahl.
- Die Leistungsdaten des verwendeten Testrechners (Prozessor, Takt, Speicher, Betriebssystem).

Mit Ausnahme der eigentlichen `.p`/`.r` Dateien binden Sie die anderen Informationen in das Dokumentationsdokument als Tabellen und Graphiken ein.

Dabei nehmen Sie als Platzierungsfläche (Abmessungen des fiktiven FPGAs) bitte die kleinste quadratische Größe an, in die die jeweilige Schaltung passt (also bei 6 Blöcken ein 3x3 Feld).

Falls die Laufzeiten Ihres Programmes bei einer aufeinanderfolgender Platzierung und Verdrahtung einer Schaltung 10 Stunden überschreiten sollten, so brechen Sie den Lauf bitte ab und machen einen entsprechenden Vermerk in Ihren Messergebnissen. Nach der Verfeinerung sollte dieser Fall in dieser Phase alerdings nicht mehr auftreten.

5 Tipps

- Unterschätzen Sie den Aufwand für eine akzeptable Dokumentation nicht! Stellen Sie ggf. eines Ihrer Gruppenmitglieder ausschliesslich für diese Aufgabe ab. In der Industrie wird das auch ähnlich praktiziert, solche Spezialisten nennt man Technical Writer.
- Sie können den aktuellen Speicherbedarf Ihres Programmes abschätzen durch einen externen Profiler, oder durch Ermitteln des maximalen Wertes über die gesamte Programmlaufzeit von:

```
tempMemory = Runtime.getRuntime().totalMemory() - Runtime.getRuntime().freeMemo
```

Der VisualVM-Profiler ist seit einiger Zeit Teil der Standard-Java-Distribution und kann gestartet werden als `jvisualvm`. Weiterführende Informationen und neuere Versionen finden sich unter **https://visualvm.dev.java.net/**.

- Zur Verwendung von UML- oder ähnlichen Diagrammen: Die Tatsache, dass es z.B. eine 1:N Relation gibt, gehört zur Beschreibung der Datenstruktur. Das Detail, dass diese mittels eines HashSets realisiert wurde, gehört zur Beschreibung der Java-Implementierung.

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 von Lösung*sideen* 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**.