Algorithmen für Chip-Entwurfswerkzeuge

Aufgabe 2 — Schaltungsplatzierung

TECHNISCHE UNIVERSITÄT DARMSTADT

Embedded Systems & Applications

Abgabe bis zum 2015-01-21
Prof. Koch

Es soll ein Programm zur Platzierung von Schaltungen auf der im Leitfaden beschriebenen fiktiven FPGA-Architektur entwickelt werden. Optimierungsziel ist das Bestimmen einer Lösung mit minimaler Verdrahtungslänge, für Gruppen mit 3 Teilnehmern ist dies nur sekundäres Optimierungsziel. Primär soll dort die Verzögerungszeit minimiert werden.

Inhaltsverzeichnis

2 Einleitung

In dieser Phase des Praktikums sollen Sie das Gelernte über Graph-Algorithmen, Heuristiken und Platzierungsverfahren anwenden.

3 Problemstellung

Ihr Programm soll als Eingabe eine Netzliste und eine Architekturbeschreibung bekommen. Weiterhin soll es mindestens die in Abschnitt 12.5.1 des Leitfadens genannten Parameter zur Modifikation der FPGA-Architektur berücksichtigen.

Mit diesen Angaben soll folgende Operation durchgeführt werden: Ordnen Sie alle Elemente (also Logik- *und* Ein-/Ausgabenblöcke) überlappungsfrei auf dem durch die Architekturparameter beschriebenen FPGA an.

Benutzen Sie hierfür den in der Vorlesungen vorgestellten, VPR nachempfunden, Simulated Annealing Algorithmus, mit folgenden Parametern:

Längenmetrik: Benutzen Sie die $Star+$-Metrik. Als Implementierungstipp: Benutzen Sie das Verfahren zur schnellen Aktualisierung der Kosten nach einem Zug. Benutzen Sie hierfür als Parameter die in der Vorlesung vorgeschlagenen Werte von $\gamma = 1.59$ und $\phi = 1.0$.

Startplatzierung: Eine zufällige Lösung.

Starttemperatur: Benutzten Sie hierfür das VPR-Verfahren mit zufälligen Zügen und der Standardabweichung.

Cooling Schedule, Zugreichweite: Benutzen Sie hier ebenfalls die VPR-Ansätze mit adaptiven Cooling Schedule und Veränderung der Zugreichweite zur Steuerung der Akzeptanzrate.

Temperaturstufe: Bieten sie, analog zu VPR, verschiedene Schritte pro Temperaturstufe an. Mindestens sollten sie die Variante mit $10N^{\frac{4}{3}}$ Schritten und die 10-fach schnellere Variante anbieten.

Kostenfunktion: Benutzen Sie eine selbstnormalisierende Kostenfunktion. Für 2-er Gruppen soll diese nur die $Star+$-Metrik für die Verdrahtungslänge berücksichtigen. 3-er Gruppen (und 2-er die sich Bonuspunkte verdienen wollen) sollen die minimale Verzögerung $D_{\max}$ optimieren. Die Schaltung soll also ein Minimum an kombinatorischer Verzögerung bzw. Verzögerung zwischen zwei Flip-Flops haben. Ein Vorschlag zur Abschätzung der Verzögerung, neben den in der Architekturdatei angegebenen Blocklaufzeiten, ist folgendes Modell: Dabei wird die Verzögerung durch die Verdrahtung als die *minimale* Anzahl von programmierbaren Schaltern (T_{switch}) angenommen, die zwischen Quell- und Zielblock einer Verbindung liegen. Bitte beachten Sie, dass T_{switch} sowohl beim "Aufspringen" als auch beim "Abspringen" auf ein Leitungssegment auftritt. Die kürzeste mögliche Verbindung zwischen zwei aneinandergrenzenden Blöcken mit optimal angenommener Pin-Zuweisung ist also $2T_{\text{switch}}$. Beispiel: Zielblock liegt rechts neben Quellblock, als Ausgangspin wird die rechte Variante von Pin 4 benutzt. Am Zielblock wird der nächstgelegene Eingangspin 1 angenommen. Dieses Netz kann also ohne eine Verbindungsmatrix realisiert werden.

4 Abgabe

Gemäß den Anforderungen im Leitfaden. Die Hauptklasse (mit der Funktion `main`) soll `Placer` heissen. Damit soll ein Programmaufruf ähnlich zu

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

möglich sein. Gute begründete Abweichungen davon (CLASSPATH etc.) dokumentieren Sie bitte auch im `README`.

Weiterhin legen Sie bitte für die Schaltungen aus Tabelle 1 folgendes Ihrer Abgabe bei (als zusätzliche Dateien in der abzugebenden `.jar` Datei):

- Die erzeugten `.p` Dateien.

Name	Blöcke	Netze
s27	6	11
tcon	16	33
bbara	33	38
inc	64	71
bw	132	137
C2670	259	416
mm30a	514	548
tseng	1047	1099
s298	1931	1935
pdc	4575	4591
clma	8383	8445

Tabelle 1: Zu bearbeitende Schaltungen und ihre Charakteristika

- Für 3-er Gruppen: Das jeweils geschätzte $D_{\max}$ (in ns).
- Die jeweils gesamte geschätzte Verdrahtungslänge in Segmenten (Kosten der $Star+$-Metrik).
- PDF-Dateien mit den Parameterkurven des Simulated Annealing für jede Schaltung (mindestens mit den Daten Gesamtkosten, Akzeptanzrate und Bewegungsradius bei jedem Schritt) Dabei sollen Kurvenplots ähnlich denen der Vorlesung gezeigten verwendet werden.
- Die jeweilige Laufzeit Ihres Programmes (in s).
- Die Leistungsdaten des verwendeten Testrechners (Prozessor, Takt, Speicher, Betriebssystem).

Dabei nehmen Sie als Platzierungsfläche (Abmessungen des fiktiven FPGAs) bitte die kleinste quadratische Größe an, in die die jeweilige Schaltung passt (also bei z.B. 6 Blöcken ein 3x3 Feld). Falls die aktuellen Architekturparameter (Datei oder Kommandozeilenoptionen) einen größeren Chip fordern, legen Sie diese kleinste Platzierungsfläche in die Mitte des Chips.

Ihr Simulated Annealing muss reproduzierbar konvergieren. Das heisst, dass es auch über mehrere Programmläufe bei derselben Schaltung und denselben Parametern, aber *unterschiedlichen* Zufallsgeneratorstartwerten eine vergleichbare Lösungsqualität erreicht. Hier sind Schwankungen der endgültigen Best-So-Far-Kosten von +/- 10% akzeptabel. Beachten Sie: Die Lösungs*qualität* muss vergleichbar sein, es muss nicht immer die *gleiche* Platzierung herauskommen. Das ist unter den gegebenen Voraussetzungen (viele zufällige Züge) auch sehr unwahrscheinlich!

Falls die Laufzeiten Ihres Programmes bei einer Schaltung zwei Stunden überschreiten, so brechen Sie den Lauf bitte ab und machen einen entsprechenden Vermerk in Ihren Messergebnissen. Zum Vergleich: Auf einem Intel Pentium III (1 GHz) / 512MB Rechner unter Linux wird die Schaltung `clma` durch ein in C geschriebenes Programm in knapp 60 Minuten den Anforderungen entsprechend platziert. Auf einem Rechner mit AMD Phenom X6 (3,2 GHz) / 4GB gelingt dies, nach wie vor Single-Threaded, in gut 7 Minuten.

5 Vortrag

In der Vorlesungszeit am Dienstag, dem 23.01.2015, findet dann eine zentrale Besprechung statt. Hier werden alle Gruppen in ca. 20-minütigen Vorträgen über ihre Lösung der Aufgabe referieren (auch hier: Anforderungen siehe Leitfaden).

6 Hilfestellung

6.1 Gruppenarbeit

Durch die Komplexität der Aufgaben und die Lage der Abgabetermine ist eine echte Gruppenarbeit unerläßlich. Diese zweite Aufgabe läßt sich beispielsweise aufteilen in

a) Annealing-Heuristik (Starttemperatur und -lösung, Züge, Abkühlen etc.)

b) Kostenfunktionen (Zeit- und Verdrahtungslängenabschätzung)

c) Infrastruktur und Test (Wiederverwenden von Dateioperationen, Erweitern des Prüfprogrammes aus Aufgabe 1, Durchführen der Messungen)

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

Gleich welche Arbeitsteilung Sie auch verwenden: Die Aufgaben sind vom Umfang und Bearbeitungszeitraum auf zügige *Gruppenarbeit* ausgelegt.

6.2 Tipps

- Behalten Sie die Effizienz Ihres Programmes im Auge. Wo wird die meiste Rechenzeit gebraucht? Hier sind sogenannte *Profiler* hilfreich. Auf dem Web sind dabei eine ganze Reihe von Möglichkeiten verfügbar (siehe Google). Als Einstieg sei hier nur kurz der bereits ab Java 1.4 eingebaute Profiler (`java -Xrunhprof:help`) und ein passendes Auswerteprogramm (`PerfAnal.jar`) genannt. Bessere Lösungen (komfortabler zu bedienen, führen auch noch über Speicherverbrauch Buch) existieren aber!

- Falls Sie möchten, können Sie relativ leicht bei SA mit der Art der Züge experimentieren. Beispielsweise könnten hier neben dem klassischen Paartausch auch der Ringtausch über mehrere Elemente oder der Austausch ganzer Regionen erfolgen.

- Implementationstipp: Nichtdeterministisches Verhalten macht das debuggen sehr schwierig. Implementieren Sie einen optionalen Kommandozeilenschalter, mit dem Sie den RNG-Seed initialisieren können und geben Sie, falls nicht gesetzt diesen aus. Bei fehlerhaftem Verhalten lässt sich so ein Lauf ggf. reproduzieren (Achtung: Für ein komplett deterministisches Verhalten kann es notwendig sein, das Verhalten von HashSet/Map/Iteratoren deterministisch zu machen).

7 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 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**.