

Algorithmen für Chip-Entwurfswerkzeuge



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Aufgabe 3 – Schaltungsverdrahtung



Embedded Systems & Applications

Abgabe bis zum 2013-01-21

Florian Stock, Prof. Koch

Es soll ein Programm zur Verdrahtung von Schaltungen auf der im Leitfaden beschriebenen fiktiven FPGA-Architektur entwickelt werden. Optimierungsziel ist primär das Bestimmen einer Lösung mit minimaler Anzahl von Verdrahtungsspuren (Tracks) pro Kanal, sekundär soll auch noch die längste Verzögerungszeit minimiert werden.

Inhaltsverzeichnis

2	Einleitung	1
3	Problemstellung	1
4	Abgabe	2
5	Kolloquium und Vortrag	3
6	Hilfestellung	3
6.1	Gruppenarbeit	4
6.2	Tipps	4
7	Hinweise zum Thema Plagiarismus	5
2	Einleitung	

In dieser Phase des Praktikums sollen Sie das Gelernte über FPGA-spezifische Verdrahtungsverfahren anwenden.

3 Problemstellung

Ihr Programm soll als Eingabe eine Netzliste, eine Platzierungsdatei und eine Architekturbeschreibung bekommen. Weiterhin soll es mindestens die in Abschnitt 12.5 des Leitfadens genannten Parameter zur Modifikation der FPGA-Architektur berücksichtigen.

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

Mit diesen Angaben soll folgende Operation durchgeführt werden: Verdrahten Sie die in der Netzliste angegebenen Verbindungen zwischen den durch die Platzierungsdatei positionierten Elementen. Dabei soll in erster Linie eine erfolgreiche Verdrahtung mit der minimalen Anzahl von Tracks pro Kanal erreicht werden. Nehmen Sie dazu an, dass in horizontalen und vertikalen Kanälen die gleiche Anzahl von Tracks bereitsteht (**W**, in der Architekturdatei definiert). Sekundär sollten Sie die Verzögerungszeit D_{\max} des kritischen Pfades minimieren.

Mit der vorgegebenen Platzierung und den von Ihrem Programm zu berechnenden Verdrahtungsdaten lassen sich nun die exakten Signallaufzeiten bestimmen. Dies ist eine Verfeinerung gegenüber dem in der Vorphase erstellten Platzierer: Dieser musste noch eine Abschätzung verwenden (i.d.R. korrigierter halber Umfang des umschliessenden Rechtecks des Netzes), jetzt haben Sie die exakten Entfernungen in Metallsegmenten zur Verfügung. Falls noch nicht geschehen (war bereits Bestandteil von Aufgabe 2), erweitern Sie auch noch Ihr Überprüfungsprogramm (aus Aufgabe 1) um die Ausgabe der Summe dieser exakten Verdrahtungslängen.

4 Abgabe

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

```
java Router s27.net prak10.arch s27.p s27.r -X 8 -Y 8 -Wh 4 -Wv 4
```

möglich sein. Weitere Optionen (z.B. Suchtiefen, verschiedene Optimierungsmodi, etc.) können Sie selbst festlegen. Abweichungen zu dem Standardaufruf oben (CLASSPATH, zusätzlich Optionen, etc.) dokumentieren Sie bitte auch im **README**. Die Java-Quellen *müssen* gut dokumentiert sein (aussagekräftiges JavaDoc, auf Wunsch alternativ auch Prosa). Das **README** muss einen kurzen Überblick über den Programmaufbau und die verwendeten Algorithmen geben. In allen Dateien muss Ihre Gruppennummer stehen!

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

Weiterhin legen Sie bitte für die Schaltungen aus Tabelle 1 folgendes Ihrer Abgabe bei:

- Die als Eingabe verwendeten `.p` Dateien (sowohl von Ihrem Placer aus der letzten Abgabe erzeugt als auch die Musterlösungen). Dabei stellen Sie bitte sicher, dass innerhalb von belegten I/O-Pads immer der Sub-Block 0 zuerst verwendet wird.
- Die erzeugten `.r` Dateien.
- Das nun exakt berechnete D_{\max} (in ns).
- Die minimale Track-Anzahl pro Kanal, mit der Ihr Programm die Schaltung erfolgreich verdrahten konnte (siehe dazu Abschnitt 6.2).
- Die gesamte verwendete Verdrahtungslänge (exakt, als Anzahl von Segmenten).
- Die Laufzeit Ihres Programmes (in s). Dabei geben Sie für den Router die Zeit für die *komplette* binäre Suche (siehe unten) 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).

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

Falls die Laufzeiten Ihres Programmes bei einer Schaltung drei Stunden überschreiten, so brechen Sie den Lauf bitte ab und machen einen entsprechenden Vermerk in Ihren Messergebnissen. Zum Vergleich: Auf einem P3-1000/512MB Rechner unter Linux wird die Schaltung `c1ma` durch ein in C geschriebenes Programm in knapp 40 Minuten den Anforderungen entsprechend verdrahtet. Auf einem Rechner mit AMD Phenom X6 (3,2 GHz) / 4GB gelingt dies, nach wie vor Single-Threaded, in gut 7,5 Minuten. Um Ihnen eine Vorstellung vom Aufwand zu geben: Dabei werden auf einer 92x92 Matrix 14 Tracks pro Kanal und insgesamt 137643 Segmente benötigt. Das längste Netz besteht aus 2915 Segmenten und der kritische Pfad hat eine Länge von 197ns. Für die Schaltung `pdc` werden auf dem P3-1000 in knapp 60 Minuten auf einer 68x68 Matrix insgesamt 104067 Segmente auf maximal 20 Tracks pro Kanal verdrahtet, mit einem kritischen Pfad von 154ns.

5 Kolloquium und Vortrag

In der Vorlesungszeit am Freitag, dem 25.01.2013, findet dann eine zentrale Besprechung mit anschließendem Kolloquium statt. Hier wird die Gruppe in ca. 20-minütigen Vorträgen über ihre Lösung der Aufgabe referieren (auch hier: Anforderungen siehe Leitfaden). Stellen Sie sicher, dass Sie Ihre Präsentation vor diesem Termin gut vorbereitet haben und in mindestens einem Probevortrag (vielleicht vor den anderen Gruppenmitgliedern) eine rhetorisch gute Vortragsweise und das Einhalten des Zeit-Limits geübt haben!

6 Hilfestellung

6.1 Gruppenarbeit

Durch die Komplexität der Aufgaben und die Lage der Abgabetermine ist eine echte Gruppenarbeit unerlässlich. Diese dritte Aufgabe lässt sich beispielsweise aufteilen in

- a) Aufbau des Routing-Graphen aus der Architekturbeschreibung.
- b) Implementierung der Kostenfunktionen.
- c) Implementierung der Basisalgorithmen (Global Router und Signal Router).

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 Gruppenarbeit ausgelegt.

6.2 Tipps

- Es wird dringend empfohlen, sich bei der Implementierung grundsätzlich an den PathFinder/VPR-Ansatz zu halten. Ihrer Kreativität können Sie dann bei der Ausgestaltung der Kostenfunktionen und verschiedener Detailoptimierungen freien Lauf lassen.
- 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 in Java 1.4 eingebaute Profiler (`java -Xrunhprof:help`) und ein passendes Auswerteprogramm (`HPjmeter`) genannt.
- Vermeiden Sie das Anlegen von ständig neuen Objekten. Verwenden Sie stattdessen bestehende Objekte wieder und ändern lediglich deren Inhalt (Instanzvariablenwerte).
- Die minimale Track-Anzahl ermitteln Sie am einfachsten durch eine binäre Suche. Lassen Sie Ihr Programm dabei zunächst mit einer recht großen Track-Anzahl laufen. Als Richtlinie: VPR braucht maximal 20 Tracks, um eine durch VPR berechnete Platzierung der Schaltung `pd` zu verdrahten. Dies ist verdrahtungstechnisch die aufwendigste Beispielschaltung. Nach einer erfolgreichen Verdrahtung einer Schaltung legen Sie im Stil der binären Suche neue Werte für `Wh` und `Wv` fest und versuchen es erneut. Auf diese Weise tasten Sie sich durch eine Folge von erfolgreichen und erfolglosen Verdrahtungsversuchen an den Minimalwert von Tracks heran. Beispiel: Bei der Verdrahtung von `clma` könnten dabei folgende Schritte durchlaufen werden: 12 Tracks (reichen nicht), 24 Tracks (reichen), 18 Tracks (reichen), 15 Tracks (reichen), 13 Tracks (reichen nicht), 14 Tracks (reichen). Damit ist klar, daß für diese Platzierung von `clma` 14 Tracks die minimal erfolgreiche Anzahl ist. Einer dieser Durchgänge benötigt rund 9 Minuten. Nach 30 erfolglosen Iterationen wird der Versuch mit der gegebenen Track-Anzahl abgebrochen und ein neuer Versuch mit mehr Tracks gestartet.
- Durch Benutzung des `W`-Parameters wird eine feste Track-Anzahl eingestellt. Hier soll dann *keine* binäre Suche mehr stattfinden, sondern in einem Durchlauf mit diesen Werten die Schaltung verdrahtet werden bzw. das Fehlschlagen ausgegeben werden.

-
- Falls Sie zu Testzwecken die Möglichkeit realisieren möchten, die binäre Suche mit einer vorgegebenen Anzahl von Tracks zu starten, führen Sie dazu bitte einen neuen Kommandozeilenparameter namens **SB** ein, der als Argument die gewünschte Startanzahl von Tracks hat. Die Angabe wird dann identisch für beide Richtungen (vertikal und horizontal) verwendet.
 - Es kann nützlich sein, die Startanzahl der binären Suche aus den jeweiligen Schaltungscharakteristika abzuschätzen. Hier dürfen Sie nur allgemein anwendbare Abschätzungen verwenden, keine der Art “wenn Schaltung X aus der Test-Suite, fange mit n Tracks an”. Darunter fallen auch Versuche, individuelle Schaltungen durch gezielte Abfrage von Schaltungsgrößen etc. zu identifizieren!
 - Die in der Vorlesung gezeigte Wiederverwendung der Wellenfront klappt *nur* bei reiner Verdrahtungsorientierung, nicht aber im Timing-Mode. Da im letzteren Fall jede Senke einen eigenen Criticality-Wert haben kann, können die Kosten in der Wellenfront nicht von Senke zu Senke weiterverwendet werden. Die PriorityQueue (die ja die Wellenfront enthält) muß also zwischen den Senken auf den eigentlichen Routing-Baum zurückgesetzt werden, alle anderen besuchten, aber nicht für den Anschluss verwendeten Knoten müssen entfernt werden.

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 Lösungsideen 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>.