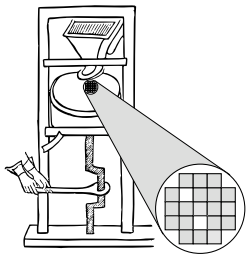
Technische Universität Darmstadt
FG Eingebettete Systeme und
ihre Anwendungen (ESA)

Prof. Dr. Andreas Koch



"Algorithmen im Chip-Entwurf"

Aufgabe 2: Schaltungsplatzierung

Abgabe bis zum 13.12.2005, 12:00 MET Mittags

Es soll ein Programm zur Platzierung von Schaltungen auf der im Leitfaden beschriebenen fiktiven FPGA-Architektur entwickelt werden. Optimierungsziel ist primär das Bestimmen einer Lösung mit minimaler Verzögerungszeit, sekundär soll auch noch die gesamte Verdrahtungslänge minimiert werden.

1 Einleitung

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

2 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. Dabei soll in erster Linie eine minimale Verzögerung $D_{\max}$ erreicht werden. Die Schaltung soll also ein Minimum an kombinatorischer Verzögerung bzw. Verzögerung zwischen zwei Flip-Flops haben.

Die spätere Verdrahtbarkeit hängt desweiteren auch von der durch die Platzierung induzierten Gesamtlänge aller Netze ab. Daher soll diese als sekundäres Kriterium minimiert werden.

Zur Abschätzung der Verzögerung verwenden Sie bitte, neben den in der Architekturdatei angegebenen Blocklaufzeiten, 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 (Abbildung 1). Bitte beachten Sie, dass T_{switch} sowohl beim "Aufspringen" als auch beim "Abspringen" auf ein Leitungssegment auftritt. Die kürzeste mögliche Ver-

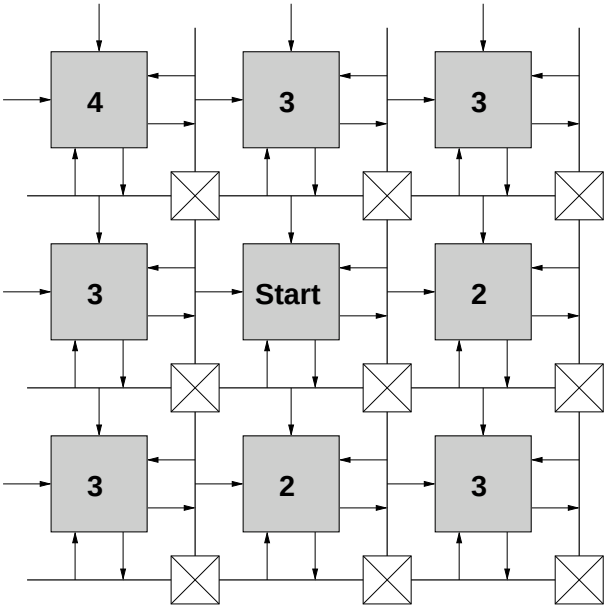


Abbildung 1: Minimale Block-Block Verzögerungen in T_{switch}

bindung 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.

Die geschätzte Verdrahtungslänge geben Sie bitte als Anzahl der benutzten Leitungen (also der Segmente zwischen den Verbindungsblöcken an). Sie sollten dazu die übliche Annäherung mittels des halben Netzumfangs verwenden, korrigiert mit $q(i)$ für eine hohe Anzahl von beteiligten Pins i. Werte von $q(i)$ für $i = 1 \ldots 50$ finden Sie in der Datei **`qi.txt`** auf der ACE05 Web-Seite. Für $i > 50$ verwenden Sie bitte die Abschätzung

$$q(i) = 2.79 + 0.02616 \cdot (i - 50)$$

Zur Überprüfung ihrer Ergebnisse erweitern Sie bitte das in Aufgabe 1 erstellte Programm um die Timing-Analyse (falls noch nicht implementiert) und Verdrahtungslängenberechnung. Dies soll mittels der eben beschriebenen Abschätzungen (bei fehlender **`.r`** Datei) bzw. in Form exakten Summe der benutzten Metallsegmente (bei angegebener **`.r`** Datei) geschehen.

3 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 prak05.arch s27.p -X 8 -Y 8
```

möglich sein. Abweichungen dazu (CLASSPATH etc.) dokumentieren Sie bitte auch im **`README`**. Die Abgaben sind bis spätestens Dienstag, dem 13.12.2005, 12:00 MET mittags, abzuschicken.

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

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

- Die erzeugten `.p` Dateien.
- Das jeweils geschätzte $D_{\max}$ (in ns).
- Die jeweils gesamte geschätzte Verdrahtungslänge (in Segmenten).
- 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 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 P3-1000/512MB Rechner unter Linux wird die Schaltung `clma` durch ein in C geschriebenes Programm in knapp 60 Minuten den Anforderungen entsprechend platziert.

4 Kolloquium und Vortrag

Am Donnerstag, dem 15.12.2005, findet mit jeder Gruppe ein ca. 30-minütiges Einzelkolloquium statt. Ihr Zeit-Slot wurde Ihnen bereits für die erste Abgabe mitgeteilt.

In der Vorlesungszeit am Freitag, dem 16.12.2005, findet dann eine zentrale Besprechung statt. Hier werden alle Gruppen in 10-minütigen Vorträgen über ihre Lösung der Aufgabe referieren (auch hier: Anforderungen siehe Leitfaden).

5 Hilfestellung

5.1 Gruppenarbeit

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

1. Annealing-Heuristik (Starttemperatur, Züge, Abkühlen etc.)
2. Kostenfunktionen (Zeit- und Verdrahtungslängenabschätzung)

3. 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 Gruppenarbeit ausgelegt.

5.2 Tipps

- Verwenden Sie als Kernalgorithmus Simulated Annealing (SA), die Tabu-Suche oder Partitionierung. Alle anderen Möglichkeiten sind entweder zu kompliziert (z.B. genetischer Algorithmus) oder nicht zielführend (z.B. Nachbarsuche). Bei der Partitionierung ist das Minimieren der Verzögerung nicht direkt möglich und muss, z.B. durch entsprechende Kantengewichte (unvollständig), nachgebildet werden.
- 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.