13. Der Autorouter

Der Autorouter kann die Platine automatisch entflechten. Dafür benötigt der Computer eine Strategie, diese besteht aus einer Menge an Regeln. Gespeichert sind diese Regeln in einer *.ctl -Datei. Problem: Der Autorouter ist "blind". Trotzdem soll er die Platine optimal entflechten. Dazu werden zwei Pins per Lee-Algorithmus mit einander verbunden.

Der Lee-Algorithmus findet immer die kürzeste Verbindung zwischen zwei Pins.

Eigentlich geht Routing nur auf dem Leiterbahnraster. Beim Lee-Algorithmus ist das Raster die Mitte der Kästchen (s. Arbeitsblatt). Wellen werden gebildet von Pin 1 zu Pin 2. Dann folgt eine Rückverfolgung von Pin 2 zu Pin 1. Dabei kann man zwischen Trace Hugging (immer nah an anderen Leiterbahnen) oder wenigen Ecken auswählen. Oft sind diese Verfahren allerdings vermischt, sodass es einen Weg gibt, der wesentlich mehr Sinn ergibt als ein anderer.

Regeln des Autorouters

Die vom Autorouter gefundenen Lösungen werden mit virtuellen Kosten bewertet und zwar unter anderem mit folgenden Kostenparametern:

- Länge (Länger ist teurer)
- Ecken
- Trace Hugging (preiswerter, wenn ja)
- Vorzugsrichtungs-Verletzung (teurer, wenn ja)
- Vias

Die insgesamt preiswerteste Lösung wird ausgegeben.

Phasen des Autorouters

Busses: Zuerst werden Busleitungen verlegt.

Route: Danach versucht der Autorouter auf jeden Fall (fast) alle Leitungen zu entflechten, egal wie teuer die Platine wird. (Allerdings darf der Autorouter nie den Designstil ANY verwenden!!)

Optimizer: Zum Schluss werden mehrere Optimierungsläufe versucht, um eine "kostengünstige" optimale Entflechtung zu finden.

Statistiken des Autorouters

Werden vom Autorouter in eine spezielle Datei geschrieben (*.pro)

- Wie viel Zeit benötigte das Autorouting? (Rechenzeit)
- Wie viele Rip ups gab es?
- Wie viele Vias gibt es?
- Wie hoch ist die Auflösung (sind alle Bahnen verlegt worden --> Prozentwert)?