Ein Überblick über Placement und Routing in integrierten Schaltungen und Systemen

Jean-Pierre Schwickerath
Technische Universität Darmstadt
tud@schwicky.net

Abstract

Es wird auf die spezifischen Probleme des Placement und des Routings im Entwurf von integrierten Schaltungen eingegangen. Genetische Algorithmen (GA) und stochastische Verfahren bieten eine effiziente Lösung zu diesen meist NP-harten Problemen. Die Nachteile der GA und der stochastischen Verfahren können umgangen werden, indem man beiden Ansätze miteinander kombiniert. Diese Idee wird erläutert und anhand eines Beispiels wird ein genetischer Placement und ein Routing Algorithmus erklärt.

1. Einleitung

Die meisten Probleme, die beim Entwurf von integrierten Schaltungen auftreten sind sehr schwierige kombinatorische Optimierungsprobleme. Mehrere, voneinander abhängige Kriterien müssen unter Rücksichtnahme einer grossen Menge nicht trivialer Bedingungen optimiert werden. Die Vorgänge bestehen aus Unterproblemen, die selbst *NP*-hart, gross und gegenseitig voneinander abhängig sind [8]. Meistens beruhen die Berechnungen eines solchen Unterproblems auf weiteren, noch nicht berechneten Schritten. So müssen zur Lösung der Probleme Abschätzungen zur Hilfe genommen werden.

Auch Placement und Routing gehören zum Entwurf von Schaltung (siehe Abbildung 1) und zu dieser Kategorie von Problemen. Man kann sie sowohl auf der Ebene einer Platine betrachten (Abbildung 2 zeigt beispielhaft den Ablauf des Entwurfs eines Platine, die aus vorgefertigten Funktionsblöcken aufgebaut wird), wo fertige Schaltungen miteinander verbunden werden müssen, wie etwa ein zentraler Prozessor, Speicherchips und Ein- und Ausgabemodule, als auch innerhalb eines solchen Chips (auch „Die" genannt), in dem diverse Funktionsblöcke (Rechenwerk, Steuerwerk, Register) oder eine Grosszahl einzelner Transistoren miteinander verbunden werden müssen. Die zu lösenden Probleme sind die gleichen und die dazu verwendeten Algorithmen ähnlich.

Diese Ausarbeitung soll einen Überblick über die Probleme geben, die beim Placement und beim Routing auftreten. Es wird ein genetisches und ein stochastisches Verfahren vorgestellt und besprochen wie die Verfahren den speziellen Problemen im Entwurf von integrierten Schaltungen angepasst werden können.

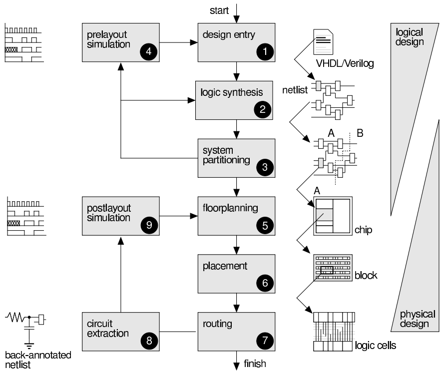


Abbildung 1. Entwurf eines Application-specific integrated circuit (ASIC). [13]

2. Zwei getrennte Probleme mit Gemeinsamkeiten

Beim Placement geht es darum die Komponenten der Schaltung auf dem Substrat zu platzieren und dabei verschiedene Vorgaben einzuhalten. Diese Vorgaben sind unter anderem: Minimierung der Fläche des benötigten Substrats, der Gesamtlänge der Verbindungen, die gleichmässige Verteilung der Hitze und Einhaltung gewisser Laufzeiten. Sie hängen einerseits vom Design ab, andererseits auch vom gewählten Substrat [7]. Es muss ausserdem genügend Platz für das Routing gelassen werden. Der Platz, der nach dem Placement auf dem Substrat übrig bleibt wird in rechteckige Flächen, so genannten Routingregionen, aufgeteilt.

Die Aufgabe des Routings ist, die Verbindungen zwischen den Komponenten (Pins) herzustellen. Auch hier gelten Vorgaben wie die Minimierung der Länge der Verbindungen, der Anzahl an verwendeten Layern (Ebenen) und an Verbindungen zwischen den Layern (Vias), der Störungsquellen (Crosstalk) und des Rauschen. Routing bei VLSI[1]-Entwurf wird meistens in globales Routing (Zuteilung von Verbindungsnetzen zu gewissen Routingregionen) und detailliertes Routing (genaue Positionierung der Verbindungsnetze innerhalb einer Routingregion) aufgeteilt [11].

Die Placement- und Routingvorgänge sind voneinander abhängig (während des Placements muss abgeschätzt werden wie viel Platz das Routing für die notwendigen Verbindungen benötigen wird) aber ihre Komplexität verlangt, dass sie getrennt voneinander behandelt werden. Jede Entscheidung, die getroffen wird hat Auswirkungen auf den Rest des Prozesses insofern, dass die Platzierung einer Komponente oder das Routing einer Verbindung weniger Platz für die anderen Komponenten bzw. Verbindungen lässt. Folglich sollten Platzierungen oder Verbindungen, die einen grossen Einfluss auf die Optimierung haben zuerst festgelegt werden.

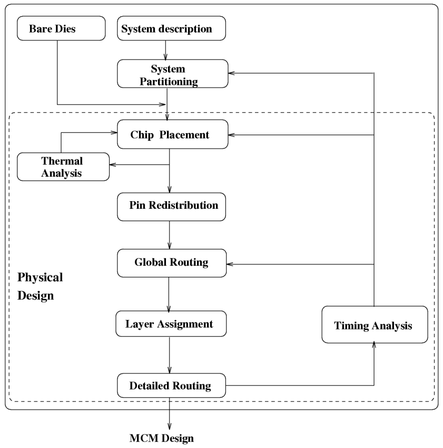


Abbildung 2. Ablauf des Designs eines Multi-Chip Moduls. [7]

[1] Very Large Scale Integrated

2.1. Das Placement-Problem

Das Placement-Problem kann folgendermassen definiert werden [9]: Unter Verwendung

1. einer Menge rechteckiger Komponenten (Zellen), jede mit einer Anzahl an Pins an vorgegebenen Positionen entlang dem Rand der Zelle positioniert,
2. einer Netlist, die die Verbindungen zwischen allen Pins spezifiziert, und
3. einer ungefähren horizontalen Länge W des zu entwerfenden Chips.

Berechne

1. die absolute Position jeder Zelle,
2. die Orientierung und Spiegelung jeder Zelle,
3. ein Rechteck B, dass die Form des Chips definiert.

Das Ziel ist es, die Fläche von B zu minimieren unter den Vorgaben, dass keine zwei Zellen sich überlappen, dass das Rechteck B alle Zellen umfasst und ungefähr die Länge W hat, und, dass die Flächen innerhalb von B, die nicht von Zellen belegt werden, gross genug sind um das Routing der Verbindungen zu beinhalten. Ein idealer Placementalgorithmus sollte folglich das flächenmässig kleinstmögliche Layout generieren, dass den elektrischen und thermischen Regeln entspricht.

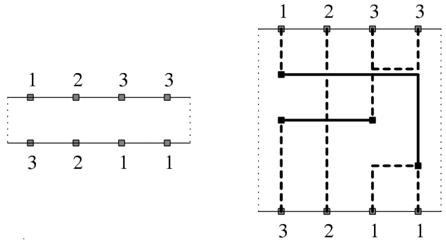
2.2. Das Routing-Problem

Das Routing-Problem kann als solches definiert werden [7]: Gegeben sei eine freie Fläche, eine Menge von Pins und eine Netlist. Berechne die Position und den Verlauf aller Verbindungen zwischen den Pins, so dass die minimale Anzahl an Layern benutzt wird, Laufzeiten und Rauschen minimiert werden, sowie Herstellungseinschränkungen eingehalten werden. Diese Einschränkungen beinhalten, dass verschiedene Nets[2] sich nicht auf einem Layer kreuzen können und, dass sie einen Minimalabstand voneinander haben müssen. Unter Umständen darf nur die Manhattan Geometrie[3] benutzt werden.

Abbildung 3 illustriert dieses Problem. Dort gilt es die Nets zwischen dem Pins mit der gleichen Nummer zu routen. Bei dem Problem handelt es sich um ein so genanntes Channel-Routing-Problem weil sich die Pins ausschliesslich auf zwei gegenüberliegenden Seiten eines für Routing zur Verfügung stehenden Rechtecks, befinden. Dabei stellen die gestrichelten Linien die Verbindungen auf einem ersten

[2] Ein Net ist eine Menge von Punkten (Pins), die zu verschiedenen Blöcken gehören und die miteinander verbunden werden müssen.
[3] Nur horizontale und vertikale Segmente dürfen benutzt werden.

Layer, und die durchgezogenen Linien auf einem 2. Layer dar. Die gefüllten Rechtecke sind Vias, also Verbindungen von einem Layer zum anderen. Die Lösung genügt den eben genannten Bedingungen.



(a) (b)

Abbildung 3. Beispiel für ein Channel-Routing-Problem (a) und eine mögliche Lösung (b). [11]

3. Grundsätzliche algorithmische Überlegungen zu den Problemen

Sowohl Routing als auch Placement sind *NP*-harte Probleme. Selbst das einfache Berechnen der kürzesten Verbindung zwischen einer Menge von Transistoren ist gleich schwer wie das Lösen des Steiner Problems in einem Graphen (SPG), welches *NP*-hart ist. Als die Anzahl der Transistoren in einer Schaltung immer grösser wurde und man die Placement- und Routing-Probleme nicht mehr effizient lösen konnte, ist man auf die Suche nach neuen Algorithmen gegangen. Mittlerweile werden ist fast allen CAD-Programmen zum Entwurf von Schaltungen genetische Algorithmen und stochastische Verfahren verwendet, die im folgenden kurz vorgestellt werden.

3.1. Simulated Annealing (SA)

Das in der CAD-Community am beliebtesten stochastische Verfahren ist das "Simulated Annealing".

Der Algorithmus ist dafür bekannt, dass er auf Kosten übertrieben langer Laufzeit qualitativ hochwertige Lösungen berechnet. Der Algorithmus ist eine Generalisierung der Monte-Carlo-Methode, in der das Einfrieren einer Flüssigkeit oder die Kristallisierung eines Metalls simuliert wird. Eine ursprünglich geschmolzene Substanz wird so langsam abgekühlt, dass sie sich immer beinahe im thermodynamischen Gleichgewicht befindet (Abbildung 4). Das Ziel ist eine möglichst gleichmässige Kristallisierung zu erhalten.

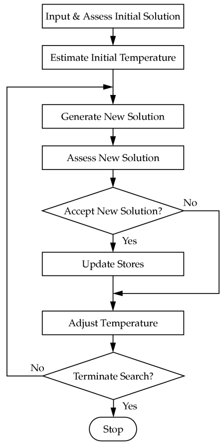


Abbildung 4. Globale Struktur des Simulated Annealing Algorithmus. [12]

3.2. Genetische Algorithmen (GA)

Genetische Algorithmen, die mit probabilistischen Übergängen nach vorteilhaften Anpassungen in einer sich immer ändernden Umgebung suchen (Abbildung 5) werden im Entwurf von integrierten Schaltungen oft verwendet.

Dort bilden Chromosome eine abtrakte Darstellung des Problems. Individuen sind potentielle Lösungen dieses Problems. Diese Individuen durchlaufen einen Evolutionsprozess, der aus drei Hauptteilen besteht:

- *Selektion* beschreibt die Phase, in der mehrere Individuen für den weiteren Evolutionsprozess ausgewählt werden. Dies geschiet anhand der Fitness eines Individuums, die aussagt wie „gut" dieses Individuum (also die Lösung) ist.
- Diese ausgewählen Individuen werden der *Kreuzung (crossover oder recombination)* unterzogen. Es entstehen neue Individuen mit den besten Eigenschaften der selektierten Vorfahren.
- Während der *Mutation* werden die neuerzeugten Individuen verändert um die Vielfalt der neue Bevölkerung zu garantieren.

Laut [8] sind GA für CAD-Probleme bestens geeignet, denn:

- Die relative Performanz von GA verbessert sich mit der Komplexität, und die Probleme mit denen hier gehandhabt wird, sind sehr komplex.
- GA sind von Natur aus parallelisierbar und beinahe lineare Skalierung wurde auf MIMD-Architekturen[4] erzielt. Der Grossteil der Rechenzeit fällt bei der Berechnung der Fitness (Kostenfunktion) an. Die Selektion und die Mutation sind nicht rechenintensiv. Folglich kann das Berechnen einzelner Kostenfunktionen auf Clusterknoten ausgelagert werden, da zwischen den Kostenfunktionen kein Datenaustausch stattfinden braucht. In Entwicklungseinrichtungen von integrierten Schaltungen sind oft viele vernetzte Rechner vorhanden auf denen viele parallele GA ausgeführt werden können. Im Gegensatz dazu sind SA basierte Algorithmen an sich sequenziell und dadurch sind viel schlechter parallelisierbar.
- Bei der Entwicklung einer Schaltung werden gewisse Schritte sehr oft iterativ wiederholt. Ein ideales CAD Werkzeug sollte die Möglichkeit bieten zwischen 1) einer überwiegend guten aber sehr schnell berechneten Lösung und 2) einer qualitativ hochwertigen Lösung für die man allerdings mehr Rechenzeit opfern muss, zu wählen. GA berechnen eine sehr gute Lösung, wenn man sie lange laufen lässt, aber dank der schnellen Konvergenz des Algorithmus in den ersten Generationen kann man innerhalb kürzester Zeit eine relativ gute Lösung berechnen.

3.3. Die Fusion: SAGA

Ein Problem der GA ist ihr Konvergenzverhalten. Am Anfang verbessert sich die Kostenfunktion sehr schnell aber dann wird es sehr schwer weitere Verbesserungen zu erzielen. Ein Grossteil der Rechenzeit geht im späteren Ablauf des Algorithmus verloren, wo nur sehr kleine Verbesserungen sehr langsam erzielt werden. Ausserdem ist ein GA immer der Gefahr ausgesetzt sich in einem lokalen Optimum zu verfangen. Esbensen und Mazumder beschreiben in [9] wie man mit sehr einfachen Mitteln dem entgegenkommen kann. Die Idee ist SA zu verwenden, der auch im späteren Teil der Berechnung immer noch respektable Verbesserungen erzielt aber am Anfang nicht so schnell konvergiert wie ein GA. Dieses SAGA-Verfahren beginnt mit der Ausführung eines reinen GA. Wenn dieser dann beinahe zur Stagnation kommt, wechselt SAGA immer mehr zur Ausführung von SA.

[4] Multiple Instruction Multiple Data-Architekturen, wie Shared-Memory-Multiprozessorsysteme oder vernetzte Rechner [10].

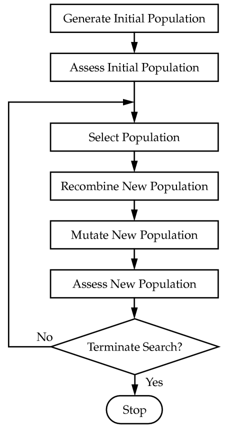


Abbildung 5. Globale Struktur eines genetischen Algorithmus. [12]

Abbildung 6 zeigt, wie man die zwei Algorithmen miteinander kombinieren kann. Π_c ist die Bevölkerung, die betrachtet wird. Die `evaluate()` Routine ist die Kostenfunktion, die bestimmt wie gut eine Lösung ist. Wenn für R Generationen keine messbaren Verbesserungen aufgetreten sind, dann wird die Bevölkerungszahl M reduziert und ihre Mutationsrate p_{mut} wird erhöht. Das ist dann der Übergang zu SA.

Experimente und Benchmarks (Abbildung 7) belegen, dass der stochastische Optimierungsalgorithmus SAGA mindestens genauso gut ist wie andere Systeme und in vielen Fällen sogar die Fläche der Schaltung weiter reduzieren kann und dabei sogar eine kürzere Laufzeit aufweist. Die verwendeten Benchmarks aus Abbildung 7 sind die des MCNC International Workshop on Placement and Routing von 1992.

4. Placement

Beim Placement gibt es ein Aspekt, der immer wichtiger wird: die Länge der Verbindungen. Mit steigenden Frequenzen muss immer mehr darauf geachtet werden, dass das Signal lange genug anliegt um von der Quelle (Driver) zur Senke (Terminal, Sink) zu kommen. Sollte dies nicht der Fall sein, müssen Buffer eingebaut werden. Diese kosten wiederum Zeit und Platz. Bei langen Verbindungsleitungen tritt zudem das Problem des Widerstandes und der Kapazität auf. Die Leitungen müssen nahe beim Driver dicker sein als

```
generate(Πc);
∀ s ∈ Πc : Ts = ⊥;
evaluate(Πc);
q = bestOf(Πc);
cR = 0;
repeat until stopCriteria():
   if no improvement for R generations do :
      cR = cR + 1;
      M = max(round(β^cR M), 1);
      Πc = reduce(Πc, M);
      pmut = min(γ pmut, 1.0);
   end;
   Πn = ∅;
   repeat M times:
      select s ∈ Πc, t ∈ Πc;
      v = crossover(s, t);
      Tv = ⊥;
      Πn = Πn ∪ {v};
   end;
   evaluate(Πc ∪ Πn);
   Πc = reduce(Πc ∪ Πn, M);
   ∀ s ∈ Πc : s = SAmutate(s);
   ∀ s ∈ Πc : with prob. pinv do :
      s = invert(s);
   evaluate(Πc);
   q = bestOf(Πc ∪ {q})
end;
∀ t ∈ Πc ∪ {q} : t = optimize(t);
r = bestOf(Πc ∪ {q});
```

Abbildung 6. Der Überblick über SAGA. [9]

beim Terminal. Das beeinflusst beim Placement den Platz, der für das Routing reserviert werden muss.

Auf Grund der Widerstände, die auftreten könne arbeiten Placement-Werkzeuge mit so genannten delay oder performance-oriented Placement [7]. Dort beschränken Timingbedingungen die obere Schranke der Verbindungslängen, die dann beim Placement berücksichtigt werden. Wenn man mehr als zwei Pins auf einem Net hat, dann bestimmt die Topologie des Netzes dessen Verzögerung. Es ist schwer während des Placements die Verzögerung zu approximieren. Zu diesem Zweck wurde vorgeschlagen das globale Routing und das Placement gleichzeitig zu bearbeiten [2]. Weitere Ansätze betrachten stattdessen den Widerstand der Leitung und die Kapazität der Senken.

Bei den traditionellen Placement-Ansätzen gibt es zwei Gruppen: konstruktive und iterative.

Die konstruktiven nehmen ein unvollständiges Placement als Eingabe an und produzieren dank bestimmten Regeln ein fertiges Placement. Dafür werden oft Min-Cut basierte Algorithmen verwendet, die sukzessive Anwendungen von Partitioning [1] benutzen. Grundsätzlich funktioniert der Algorithmus wie folgt:

1. Teile die Placementfläche in zwei.
2. Tausche die Logikzellen um die Schnittkosten zu minimieren.
3. Wiederhole ab Schritt 1. Teile immer kleinere Stücke bis alle Logikzellen platziert sind.

Benchmark	System	Area (mm²)
Apte	SAGA	53.58
	BB [5]	54.05
	Seattle Silicon [6]	54.77
Xerox	Seattle Silicon [6]	25.79
	BB [5]	26.17
	SAGA	27.15
	BEAR [3]	28.47
	MOSAICO[1]	29.01
	VITAL[1]	31.71
Hp	SAGA	11.81
	Seattle Silicon [6]	11.85
	BB [5]	12.15

Abbildung 7. Vergleich der Flächen der Ergebnisse von SAGA und anderen Algorithmen. [9]

Abbildung 8 zeigt wie man das Teilen durchführt:

(a) Teile den Chip mit einem Gitter in Behälter auf.

(b) Führe alle Verbindungen zum Mittelpunkt des jeweiligen Behälters.

(c) Mache einen Schnitt und tausche die Logikzellen um die Kosten des Schnitts zu minimieren.

(d) Nimm die geteilten Hälften und entferne alle Kanten, die nicht innerhalb einer Hälfte sind.

(e) Wiederhole den Prozess mit einem neuen Schnitt und fahre fort bis nur noch einzelne Behälter übrig sind.

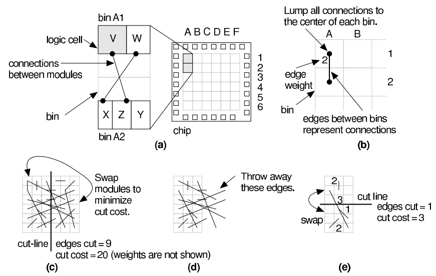


Abbildung 8. Min-Cut Placement. [13]

Bei den iterativen Methoden wird ein Annahme für das Placement gemacht und dann wird diese verfeinert indem man Bedingungen für ein besseres Placement berücksichtigt. Die GA- und SA-Ansätze fallen in diese Kategorie. In jeder Iteration wird eine Komponente gedreht oder verschoben und wenn die Konfiguration besser ist als die vorherige, dann wir diese als neue Grundlage genommen. Bei den iterativen Verfahren gibt es grundsätzlich zwei Kriterien:

- Das Selektionskriterium, das entscheidet welche Zelle verschoben werden soll.
- Das Messkriterium, das entscheidet ob die ausgewählte Zelle nun verschoben wird.

Wenn man das Simulated Annealing Verfahren auf das Placement anwendet, sieht der Algorithmus folgendermassen aus [13]:

1. Wähle eine Logikzelle für einen potentiellen Tausch, meistens zufällig ausgewählt.
2. Berechne die objektive Funktion E für das neue Placement.
3. Wenn ΔE negativ oder Null ist, dann werden die Logikzellen getauscht. Wenn ΔE positiv ist, dann werden die Zellen mit einer Wahrscheinlichkeit von $e^{-\Delta E/T}$ getauscht (Messkriterium).
4. Wiederhole ab Schritt 1 für eine feste Anzahl an Durchläufen und senke die Temperatur T nach einem Abkühlungsschema. Zum Beispiel $T_{n+1} = 0.9T_n$.

5. Routing

Wenn das Placement abgeschlossen ist geht es beim Routing um die genaue Festlegung der Verbindungsnetze. Alle Bedingungen, die beim Placement eine Rolle spielten, gelten auch hier und müssen zusammen mit weiteren Bedingungen eingehalten werden. Alle diese Bedingungen werden eine nach der anderen auf Erfüllung getestet, wenn es darum geht die Qualität einer generierten Lösung zu bewerten. Es gibt aber auch Algorithmen, die von ihrem Ablauf her bei der Generierung neuer Lösungen gewissen Bedingungen nachkommen.

Anhand des Beispiels aus Abbildung 9 soll erläutert werden, wie Channel-Routing mit Hilfe von genetischen Algorithmen funktionieren kann. Im Channel-Routing-Problem kann ein Individuum als das Ergebnis eines Routings betrachtet werden. Die Qualität dieses Routings kann anhand der Fitness des Individuums festgestellt werden. Am Anfang wird eine initiale Bevölkerung zufällig erzeugt. Diese Bevölkerung wird einem Evolutionsprozess (wie in 3.2 beschrieben) ausgesetzt. Wenn die Simulation funktioniert, dann werden immer bessere Individuen in der Bevölkerung vorherrschen, weil sie eine höhere Wahrscheinlichkeit haben, Nachfahren zu produzieren, die nur die besten Eigenschaften ihrer Vorfahren erben. Diese besten Individuen sind genau die besten Routinglösungen, die den gewünschten Kriterien entsprechen.

Die Anzahl der Individuen $|\mathcal{P}_c|$ bleibt konstant über alle Generationen hinweg. Am Ende des Algorithmus wird p_{best} einer Optimierung unterzogen und bildet dann die endgültige Lösung des Routings.

```
create initial population (Pc)
fitness_calculation (Pc)
pbest = best_individual (Pc)
for generation = 1 until max_generation
   Pn = ∅
   for offspring = 1 until max_descendant
      pα = selection (Pc)
      pβ = selection (Pc)
      Pn = Pn ∪ crossover (pα, pβ)
   endfor
   fitness_calculation (Pn)
   Pc = reduction (Pc ∪ Pn)
   pbest = best_individual (pbest ∪ Pc)
   mutation (Pc)
   fitness_calculation (Pc)
endfor
optimize (pbest)
```

Abbildung 9. Umriss eines genetischen Routingalgorithmus. [11]

Der Erzeugungsprozess der initialen Bevölkerung läuft wie folgt ab. Am Anfang wird eine Bevölkerung aus zufälligen Individuen erzeugt. Nun wünscht man sich die Verbindung zwischen den Pins s_i und t_j. Dazu wird von beiden Pins aus eine senkrechte Linie erzeugt bis diese auf ein Hindernis trifft, zum Beispiel den Rand der Routingregion oder ein anderes, schon geroutetes Netz (Abbildung 10 (a, b)). Zwischen den Endpunkten der gerade erzeugten Erweiterungslinien wird jeweils zufällig ein Punkt bestimmt. Von dort aus werden waagerechte Linien in beide Richtungen erzeugt (Abbildung 10 (c)). Die Suche geht weiter indem man auf den waagerechten Linien zwei zufällige Punkte auswählt und dort senkrechte Erweiterungslinien verankert (Abbildung 10 (d)).

Das Layer dieser Linien wird wie folgt bestimmt. Jedes Layer hat eine bevorzugte Routingrichtung und sei r_n eine zufällige Zahl zwischen 0 und 1. Wenn $r_n \leq 2/3$ dann wird die Linie auf das Layer gesetzt, dass ihre Routingrichtung bevorzugt. Andernfalls wird die Linie auf das andere Layer gesetzt.

Der Erweiterungslinienprozess wird beendet wenn

- Erweiterungslinien beider Punkte treffen sich auf dem gleichen Layer
- Die Erweiterungslinie von s_i trifft auf eine Punkt, der schon zum Netz von t_j gehört (wie in Abbildung 10 (e)) oder umgekehrt.

Sollte die Erzeugung der Erweiterungslinien nicht innerhalb von *i* Iterationen zum Ende kommen, dann werden alle Erweiterungslinien gelöscht, die Routingregion zufällig um eine Gitterlinie erweitert und die Linien neu erzeugt.

Wenn nach zehn Erweiterungen der Routingregion immer noch keine Verbindung hergestellt werden konnte, dann wird diese Bevölkerung komplett ausgelöscht und zufällig neu erzeugt.

Der Routingprozess wird abgeschlossen indem der kürzeste Weg auf den Erweiterungslinien gewählt wird.

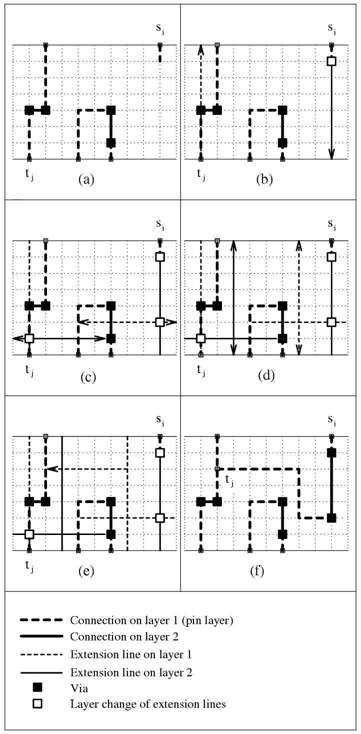


Abbildung 10. Beispiel für ein zufälliges Routing zwischen s_i und t_j. [11]

Dieser Channel-Routing Algorithmus liefert sehr gute Ergebnisse, wie der Vergleich bekannter Channel-Router in Abbildung 11 zeigt. Der Algorithmus wurde nach 150 Generationen unterbrochen und mit jeweils $|\mathcal{P}_c| = 50$ Individuen ausgeführt. Dabei konnte er sogar den Joo6_16 Benchmark ausführen, an dem der Greedy Algorithmus gescheitert ist.

Benchmark	System	Col.	Rows	Netlength	Vias
Yoshimura-Kuh channel	Yosh.-Kuh [34]	12	5	75	21
	Weaver [19]	12	4	67	12
	Monreale [11]	12	4	72	11
	Our work	12	4	70	11
Joo6_12	Weaver [19]	12	4	79	14
	Packer [12]	12	4	82	18
	Monreale [11]	12	4	84	13
	Our work	12	4	79	14
Joo6_13	Greedy [28]	18	8	194	38
	Weaver [19]	18	7	169	29
	Silk [23]	18	6	171	28
	Packer [12]	18	6	167	25
	Our work	18	6	165	25
Joo6_16	Weaver [19]	11	8	131	23
	Weaver[a] [19]	11	7	121	21
	Monreale [11]	11	7	120	19
	Our work	11	6	116	15
Burstein's difficult channel	Mighty [32]	13[b]	4	83	8
	Packer [12]	12	4	82	10
	Monreale [11]	12	4	82	10
	Our work	12	4	82	8

[a] interactively
[b] additional column in the middle of the channel

Abbildung 11. Benchmarkergebnisse. [11]

6. Zusammenfassung und Ausblick

Bei Placement und Routing hat man es mit *NP*-harten Problemen zu tun. Viele, voneinander abhängige Kriterien müssen unter Berücksichtigung nicht trivialer Bedingungen optimiert werden. Bei der heutigen Grösse und Komplexität der Schaltungen ist es nicht mehr möglich diese mit traditionellen kombinatorischen Verfahren zu bearbeiten. Seit Mitte der 1990er Jahre sind immer mehr genetische und stochastische Verfahren im Einsatz. Sie haben den Vorteil sehr schnell zu konvergieren und, wenn man ihnen genügend Laufzeit zur Verfügung stellt, liefern sie auch qualitativ exzellente Ergebnisse. Ein wichtiges Kriterium der Algorithmen ist ihre inhärente Parallelisierbarkeit. Nur so kann gewährleistet werden, dass sie in vertretbarer Zeit bearbeitet werden können, wie etwa die genetischen Algorithmen, die auf vernetzten Rechner sehr gut skalieren.

Was Benchmarks angeht so stellt Smith in [13] fest, dass man CAD-Algorithmen, die auf Zufallsdaten zurückgreifen sehr schlecht vergleichen kann. Die Ergebnisse sind nicht wiederholbar und Vergleiche aufzustellen ist gewagt, es sei denn man hat statistisch ausreichend viele Testläufe für statistisch ausreichend viele Chips durchgeführt.

Chang et al. haben in [3] Placement und Routing Werkzeuge (Dragon, Capi, mPL, mPG und QPlace) miteinander verglichen. Sie haben die Werkzeuge auf Probleme angesetzt, deren optimale Verbindungsnetzlängen bekannt sind. Sie sind zu dem Schluss gekommen, dass die Länge der Verbindungen in den Ergebnissen im Durchschnitt 1.62 bis 2.07 Mal länger waren als die optimalen Lösungen. Sie stellten ebenfalls fest, dass die Qualität der Lösungen um 9% bis 17% schlechter wurden, wenn das Problem um Faktor 10 vergrössert wurde. Daraus kann mal folgern, dass Werkzeuge zum Entwurf von Schaltungen immer noch verbesserungsfähig sind. Sollte dies gelingen, so kommt es dem Fortschritt mehrerer Generationen an Herstellungsverfahren gleich.

Literatur

[1] M. Breuer. Min-cut placement. *Journal of Design Automation and Fault Tolerant Computing*, 1(4):343–382, October 1977.

[2] M. Burstein. A non placement-routing approach to automation of VLSI layout design. *Proceedings of the International Symposium of Circuits*, pages 234–244, 1989.

[3] M. R. Chin-Chih Chang, Jason Cong and M. Xie. Optimality and scalability study of existing placement algorithms. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 23(4):537–549, 2004.

[4] J. Y. Cho and J. D. Cho. Improving performance and routability estimation in mcm placement.

[5] J. Cong, L. He, C. Koh, and P. Madden. Performance optimization of vlsi interconnect layout, 1996.

[6] E. C. S. D.J. Hathaway, R. R. Habra and S. J. Rothman. Circuit placement, chip optimization and wire routing for IBM IC technology. *IBM J. RES. Develop.*, 40(4):453–460, 1996.

[7] S. C. Donald. An overview of placement and routing algorithms for multi-chip modules.

[8] R. Drechsler, H. Esbensen, and B. Becker. Genetic algorithms in computer aided design of integrated circuits, 1994.

[9] H. Esbensen and P. Mazumder. SAGA: a unification of the genetic algorithm with simulated annealing and its application to macro-cell placement. *Proceedings of the Seventh International Conference on VLSI Design*, pages 211–214, 1994.

[10] R. Hoffmann. Einführungskapitel zur Vorlesung Rechnerarchitektur, 2003.

[11] J. Lienig and K. Thulasiraman. A genetic algorithm for channel routing in VLSI circuits. *Evolutionary Computation*, 1(4):293–311, 1993.

[12] C. S. E. Project. *Mathematical Optimization*. 1995.

[13] M. J. S. Smith. *Application-Specific Integrated Circuits*. Addison-Wesley Publishing Company, 1997.