

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. Generische 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 Ansätzen miteinander kombiniert. Diese Idee wird erläutert und anhand eines Beispiels wird ein genetischer Placement und ein Routing Algorithmus erklärt.

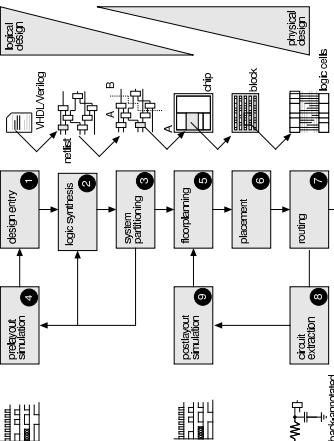


Abbildung 1. Entwurf eines Application-specific integrated circuit (ASIC) [13]

Die meisten Probleme, die beim Entwurf von integrierten Schaltungen auftreten sind sehr schwierige kombinatorische Optimierungssprobleme. 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 [18]. 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 Schaltungen. Man kann sie sowohl auf der Ebene einer Platine betrachten (siehe Abbildung 1) und zu dieser Kategorie von Problemen. Man kann sie ebenfalls auf der Ebene einer Platinenbauteile aufgebaut 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 ähneln.

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-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 benötigt wird), aber ihre Komplexität verlangt, dass sie getrennt voneinander behandelt werden. Jede Entscheidung, die getroffen wird hat Auswirkungen auf den Rest des Prozesses; insoforn, 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.

- ### 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 ungefährten 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 flächenminig Kleinstmögliche Layout generieren, dass den elektrischen und thermischen Regelns 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 Nets² sich nicht auf einem Layer kreuzen können und, dass sie einen Minimalabstand voneinander haben müssen. Unter Umständen darf nur die Manhattan Geometrie³ benutzt werden.
- Abbildung 3 illustriert dieses Problem. Dort gilt es die Nets zwischen dem Pins mit der gleichen Nummer zu richten. 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

²Ein Net ist eine Menge von Punkten (Pins), die zu verschiedenen Blöcken gehören und die miteinander verbunden werden müssen.

³Nur horizontale und vertikale Segmente dürfen benutzt werden.

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

Very Large Scale Integrated

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.

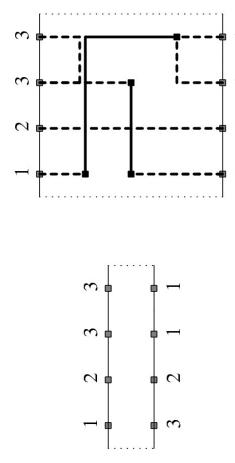


Abbildung 3. Beispiel für ein Channel-Routing-Problem (a) und eine mögliche Lösung (b), [11]

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 es 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 bei nahe im thermodynamischen Gleichgewicht befindet (Abbildung 4). Das

Ziel ist eine möglichst gleichmässige Kristallisierung zu erhalten.

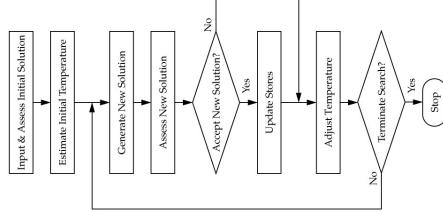


Abbildung 4. Globale Struktur des Simulated Annealing Algorithmus, [12]

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 Chromosone eine abstrakte 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 geschieht anhand der Fitness eines Individuums, die aussagt wie „gut“ dieses Individuum (also die Lösung) ist.
- Diese ausgewählten Individuen werden der *Kreuzung (crossover oder recombination)* unterzogen. Es entstehen neue Individuen mit den besten Eigenschaften der selektierten Vorfahren.
- Während der *Mutation* werden die neu erzeugten Individuen verändert um die Vielfalt der neuen 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-Architekturen⁴ erreicht. 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 hochwertigeren 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.

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 evalute () Routine ist die Kostenfunktion, die bestimmt wie gute eine Lösung ist. Wenn für R Generationen keine messbaren Verbesserungen aufgetreten sind, dann wird die Bevölkerungsgröße M reduziert und ihre Mutationsrate p_{mut} wird erhöht. Das ist dann der Übergang zu SA. Experimente und Benchmarks (Abbildung 7) belegen, dass der stochasticische 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, den 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

⁴Multiple Instruction Multiple Data-Architekturen oder vernetzte Rechner [10].

```

generate( $\Pi_0$ );
 $\forall s \in \Pi_0 : T_s = L_s$ ;
evaluate( $\Pi_0$ );
 $q = \text{bestOff}(\Pi_0)$ ;
 $cr = 0$ ;  $R = 77$ ;
repeat until stopCriteria( $Q$ );
  if no improvement for  $R$  generations do :
     $cr = cr + 1$ ;
     $M = \max(\text{round}(\beta * M), 1)$ ;
     $I_e = \text{reduce}(\Pi_0, M, 1)$ ;
     $p_{mut} = \min(7p_{mut}, 1.0)$ ;
    end;
     $\Pi_n = \emptyset$ ;
    repeat  $M$  times:
      select  $t \in \Pi_0$ ,  $t \in \Pi_1$ ;
       $v = \text{crossover}(s, t)$ ;
       $T_v = \frac{I_e}{2}$ ;
       $I_n = I_n \cup \{v\}$ ;
    end;
    evaluate( $\Pi_n \cup I_n$ );
     $I_c = \text{reduce}(\Pi_n \cup I_n, M)$ ;
     $\forall s \in \Pi_n : s = \text{SAmutate}(s)$ ;
     $\forall s \in \Pi_n : \text{with prob. } p_{inv} \text{ do} :$ 
       $s = \text{invert}(s)$ ;
      evaluate( $\Pi_n$ );
       $q = \text{bestOff}(\Pi_n \cup \{q\})$ ;
    end;
     $\forall t \in \Pi_n : q : t = \text{optimize}(t)$ ;
     $r = \text{bestOff}(\Pi_n \cup \{q\})$ ;
  end;
end;

```

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 können arbeiten Placement-Werkzeuge mit so genannten delay oder performance-oriented Placement [7]. Dort beschränken Timingbedingungen die obere Schranken der Verbindungsstrecken, die dann beim Placement berücksichtigt werden. Wenn man mehr als zwei Pins auf einem Net hat, dann bestimmt die Topologie des Netzwerks dessen Verzögerung. Es ist schwierig während des Platzes 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 Senkten.

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

Die konstruktiven nehmen ein unvollständiges Placement als Eingaben 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 Placmentfläche 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 (mm ²)
Apté	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 ⁺	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 Hälfte 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.

- (f) Nimm die genaue Festlegung der Verbindungsnetze. Alle Bedingungen, die beim Placement eine Rolle spielen, 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.
- (g) 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 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, Nächtfahren zu produzieren, die nur die besten Eigenschaften ihrer Vorfahren erbten. Diese besten Individuen

jeder Iteration wird eine Komponente gedreht oder verschoben und wenn die Konfiguration besser ist als die vorherige, dann wird 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 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, Nächtfahren zu produzieren, die nur die besten Eigenschaften ihrer Vorfahren erbten. Diese besten Individuen

sind genau die besten Routingsolutionen, die den gewünschten Kriterien entsprechen.
Die Anzahl der Individuen $|P_c|$ 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 ( $P_e$ )
 $P_{best} = \text{best.individual} (P_e)$ 
for generation = 1 until max-generation
   $P_e = \emptyset$ 
  for offspring = 1 until max.offspring
     $p_o = \text{selection} (P_e)$ 
     $p_g = \text{selection} (P_e)$ 
     $p_a = p_o \cup \text{crossover} (p_o, p_g)$ 
    fitness calculation ( $P_e$ )
  endfor
   $P_e = P_e \cup \text{fitness calculation} (p_a)$ 
   $P_{best} = \text{best.individual} (P_{best} \cup P_e)$ 
  mutation ( $P_e$ )
   $P_e = \text{selection} (P_e)$ 
endfor
optimize ( $P_{best}$ )

```

Abbildung 9. Umriss eines genetischen Routingalgoritmus. [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 Routierichtung 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 Routierichtung 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 Punkte der schon zum Netz von t_j gehört (wie in Abbildung 10 (c)) oder umgekehrt.

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

Sollte die Erzeugung der Erweiterungslinien nicht innerhalb von Iterationen zum Ende kommen, dann werden alle Erweiterungslinien gelöscht, die Routingregion zufällig um eine Gitterlinie erweitert und die Linien neu erzeugt, wenn noch keine Verbindung hergestellt werden konnte, dann wird diese Bevölkerung komplett ausgetölt und zufällig neu erzeugt.

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

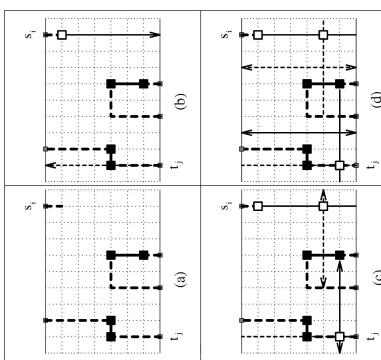


Abbildung 10. Beispiel für ein zufälliges Routing zwischen s_i und t_j . [11]

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 $|p_c| = 50$ Individuen ausgeführt. Dabei konnte er sogar den Joo6_16 Benchmark ausführen, an dem der Greedy Algorithmus gescheitert ist.

Bei der Erzeugung der Erweiterungslinien nicht innerhalb von Iterationen zum Ende kommen, dann werden alle Erweiterungslinien gelöscht, die Routingregion zufällig um eine Gitterlinie erweitert und die Linien neu erzeugt, wenn noch keine Verbindung hergestellt werden konnte, dann wird diese Bevölkerung komplett ausgetölt und zufällig neu erzeugt.

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

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

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

Literatur

- [1] M. Breuer. Min-cut placement. *Journal of Design Automation and Fault Tolerant Computing*, 1(4):343–352, 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] I. Y. Cho and J. D. Cho. Improving performance and routability estimation in mem placement. [5] J. Cong, L. He, C. Koh, and P. Madden. Performance optimization of vlsi interconnect layout. 1996.
- [6] E. C. S. D. I. 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. Lieng 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.

zeuge (Dragon, Capi, mPL, mPG und QPlace) miteinander verglichen. Sie haben die Werkzeuge auf Probleme angesetzt, deren optimale Verbindungsnetzpläne 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 man folgern, dass Werkzeuge zum Entwurf von Schaltungen immer noch verbessert werden müssen. Sollte dies gelingen, so kommt es dem Fortschritt mehrerer Generationen an Herstellungsverfahren gleich.

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 stochastiche 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 Rechnern sehr gut skalierten.

Was Benchmarks angeht so stellt Smith in [13] fest, dass man CAD-Algorithmen die auf Zufallsstößen 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 Werk-