Ein Branch&Bound-Ansatz zur Verdrahtung von Field Programmable Gate-Arrays

Möhrke, U.; Herrmann, P.; Steffen, M.; Spruth, W. G.

Universität Leipzig
Augustusplatz 10-11
04103 Leipzig

April 1998

Inhaltsverzeichnis

Abstrakt

Zur Verdrahtung der meisten FPGA-Architekturen können die aus dem ASIC-Entwurf stammenden Werkzeuge wie z.B. Kanalverdrahter nicht eingesetzt werden. Eine vollautomatische Verdrahtung mit optimalen Signallaufzeiten kann nur dann erreicht werden, wenn bei gegebener Plazierung die Leitungführung den technologischen Gegebenheiten angepaßt wird. Diese unterscheiden sich deutlich von denen in ASICs.
Im Rahmen des von der Deutschen Forschungsgemeinschaft (DFG) geförderten Gemeinschafts-Projekts „FPGA Entwurfssystem“, an dem die Universität Leipzig, die Universität Tübingen und die Technischen Universität München beteiligt sind, wurden am Lehrstuhl für Computersysteme (Prof. W.G. Spruth) des Instituts für Informatik der Universität Leipzig Verfahren zur effizienten und qualitativ hochwertigen Verdrahtung von FPGA-Bausteinen entwickelt.
Es wird eine Beschreibung des Verdrahtungsproblems für FPGAs gegeben und ein Lösungsansatz mit Hilfe des Branch&Bound – Verfahrens vorgestellt. Die Ergebnisse in Form von Programmlaufzeiten, Länge des kritischen Pfades und Anzahl der betrachteten Suchknoten in Abhängigkeit von einer Vielzahl von Schaltungsvarianten sind tabellarisch dargestellt und dokumentieren eine deutliche Verkürzung der längsten Pfade gegenüber dem Plazier- und Verdrahtungswerkzeug von Xilinx. Abschließend werden Probleme und weiterführende Arbeiten diskutiert.

1. Einleitung

FPGAs (Field Programmable Gate Arrays) stellen programmierbare Bausteine dar, die entweder einmalig (Antifuse) oder mehrmalig (z.B. SRAM) vom Nutzer programmiert werden können. Unter Verdrahtung versteht man die elektrische Verbindung von logischen Blöcken, Gattern, Ein-/Ausgabeblöcken oder Pads durch Leitungen auf dem Chip.

Die verfügbaren FPGAs unterscheiden sich stark in den Programmiertechnologien (SRAM, Antifuse, EPROM, EEPROM), und - für die Verdrahtung besonders wichtig - in der Architektur. Einen Überblick über die verschiedenen FPGA-Bausteinfamilien findet man in [ROGS93]. Weitere Arbeiten beschäftigen sich mit dem Einfluß der Architektur auf die Verdrahtbarkeit und die Komplexität des Verdrahtungsproblems ([TRIM95], [WUTM96], [BERO96], [WCMT97]).

Für die Verdrahtung von FPGAs ergibt sich eine wesentliche Gemeinsamkeit: Die Verdrahtung erfolgt über vorgefertigte Leiterbahnsegmente, die durch programmierbare Schalter (PIP: Programmable Interconnection Point) miteinander verbunden werden können. Aufgrund dieser Struktur unterscheidet sich das Verdrahtungsproblem für FPGAs stark von demjenigen für Makrozellen und Standardzellen, denn man kann die Leitungen nicht innerhalb vorgegebener Kanäle oder Flächen frei wählen. Das führt dazu, daß vorhandene Verdrahtungsprogramme nicht direkt auf dieses spezielle Verdrahtungsproblem angewendet werden können bzw. die Ergebnisse nicht zufriedenstellend sind.

In der Literatur werden üblicherweise Architekturen betrachtet, die aus Gittern programmierbarer logischer Blöcke bestehen, zwischen denen senkrecht und waagerecht Kanäle von vorgefertigten Leiterbahnen verlaufen. Die Leiterbahnen können einerseits in Verbindungsboxen mit den Anschlußpins der logische Blöcke und andererseits in Switchboxen untereinander verbunden werden.
Viele Ansätze aus der Literatur nutzen ein Verfahren aus zwei Schritten, einem Global- und einem Endverdrahtungsschritt. Beim Globalverdrahten werden den Netzen Kanäle zugeordnet, aus denen im Endverdrahtungsschritt die Leiterbahnsegmente ausgewählt werden müssen.
Bekannte Ansätze zur Endverdrahtung sind CGE [BRRV92] und SEGA [LEBR93].
Während es bei der Endverdrahtung darum geht, die endgültige Zuordnung zu den Leiterbahnsegmenten vorzunehmen, kommt es beim Globalverdrahtungsschritt darauf an, den Netzen Kanäle so zuzuordnen, daß das Zeitverhalten der Verdrahtung möglichst gut ist und eine Endverdrahtung existiert. Das zweite kann erreicht werden, indem die Kanäle möglichst gleichmäßig ausgenutzt werden. Neuere Arbeiten beschäftigen sich nicht nur mit der gleichmäßigen Ausnutzung der Kanäle, sondern auch mit den Möglichkeiten der Switchboxen (z.B. [TCWM97]).
Zur Lösung des Globalverdrahtungsproblems (bei Einschrittverfahren desVerdrahtungsproblems überhaupt) werden verschiedene Heuristiken zur Konstruktion von Steinerbäumen herangezogen. Bekannt sind die Heuristiken von Kou, Markowski und Berman ([KOMB81]) und Zelitkovsky ([ZELI93]). Auf diesen bauen verschiedene andere Ansätze auf. Iterative Anwendungen/Verbesserungen findet man z.B. in [ALRO96]. Die genannten Heuristiken können dazu genutzt werden, ganze Netze auf einmal zu verdrahten. Eine andere Möglichkeit besteht darin, die Senken der Netze einzeln zu betrachten ([RALJ96]). Das bietet die Möglichkeit, die zeitkritischen Anschlußpins einzeln zu betrachten.
Weite Verbreitung in CAD-Programmen finden simulated-evolution basierte Techniken. Auch für das Verdrahten von FPGAs gibt es entsprechende Ansätze. Entscheidend ist dabei, nach welchen Kostenfunktionen entschieden wird, welche Netze verdrahtet bzw. aufgerissen werden. Ein Beispiel findet man in [LEWU95].

Innerhalb des DFG-Gemeinschaftsprojekts „FPGA Entwurfssystem“ der Universitäten Leipzig, Tübingen und München werden insbesondere Verdrahtungsalgorithmen behandelt, die sich leicht auf verschiedene Typen von FPGAs anpassen lassen. Dabei sollen diese Algorithmen an existierenden Architekturen entwickelt und getestet werden. Die dafür notwendigen Architekturinformationen werden von den Herstellern aus verständlichen Gründen nur unvollständig veröffentlicht, so daß umfangreiche Arbeiten diesbezüglich notwendig sind. Aus diesem Grund beschränkten sich die Untersuchungen auf FPGAs der 3000 Familie von Xilinx. Über das LCA-Netzlistenformat dieser Familie ([MSH96]) ist die Integration der entwickelten Programme in den Xilinx-Entwurfsablauf und ein direkter Vergleich der Ergebnisse mit denen der Firma Xilinx möglich.

Das Ziel der Betrachtung von Algorithmen, die für die verschiedensten Typen von FPGAs anwendbar sind, führt zu gewissen Einschränkungen in der Wahl der Algorithmen. So erscheint es nicht sinnvoll, einen Versuch mit Global- und Endverdratung zu unternehmen, da aufgrund der sehr verschiedenen Architekturen keine allgemeingültige Unterteilung in Global- und Endverdrahtungsschritt möglich ist.

Es wird hier ein Ansatz mit Hilfe von Branch&Bound-Algorithmen vorgestellt. Die Verdrahtungsressourcen werden als Graph von Leiterbahnsegmenten (die Knoten des Graphen) und programmierbaren Schaltern (die Ecken des Graphen) betrachtet. Diese Darstellung ist unabhängig vom speziellen Typ des FPGAs und somit geeignet, um Algorithmen und Programme zu entwickeln, die sich leicht an unterschiedliche FPGAs anpassen lassen.

Der Vorteil von Branch&Bound-Algorithmen gegenüber anderen liegt darin, daß man mit Sicherheit das Optimum findet. Da das Verdrahtungsproblem für FPGAs NP-vollständig ist, mußte jedoch von vornherein damit gerechnet werden, daß die Komplexität der Aufgabenstellung Probleme bereitet. Das Ziel bestand darin, einen Algorithmus zu implementieren und zunächst festzustellen, ob Branch & Bound-Algorithmen grundsätzlich zur Lösung des FPGA-Verdrahtungsproblems geeignet sind.

2. Formulierung der Aufgabenstellung

Gegeben ist eine Menge von Leiterbahnsegmenten L und eine Menge von programmierbaren Schaltern $P \subset L \times L$. Diese bilden einen gerichteten Graphen G =(L, P). Die Menge der Leitungen L bilden die Knoten, die programmierbaren Schalter stellen die Kanten dar.
Ein Netz n = (q, S) sei gegeben durch seine Quelle $q \in L$ und die Menge der Senken $S \subset L$.
Eine Realisierung eines Netzes ist ein Baum B auf dem Graphen G, wobei q die Wurzel und S die Menge der Blätter bildet: Von q führt zu jeder Senke $s \in S$ ein Weg $\{p(1), \ldots , p(i)\} \subset P$, d.h. es gibt Leitungen e(0), ... , e(i) mit e(0)=q, e(i)=s und p(k) = (e(k-1), e(k)). Die Vereinigung der Wege bildet den Baum B.
Gegeben sei eine Menge von Netzen N = {n(1), ... , n(k)}
Die Verdrahtungsaufgabe besteht darin, zu jedem Netz eine Realisierung zu finden, so daß diese paarweise disjunkt sind (keine gemeinsamen Leitungen haben).

Zu dieser Formulierung der Aufgabenstellung kommen noch teilweise wesentliche Nebenbedingungen, wie z.B. Bedingungen über die maximale Signallaufzeit auf dem kritischen Pfad. Die Nebenbedingungen können sehr von den speziellen Wünschen des Anwenders oder auch von der speziellen Architektur des FPGAs abhängig sein.

Bei manchen FPGAs (z.B. Xilinx XC3000er Familie) müssen die Leiterbahnsegmente für Quellen und Senken nicht genau vorgeschrieben sein. Die Änderungen der Aufgabenstellung sieht dann wie folgt aus: Quellen und Senken der Netze sind dann Mengen von Leiterbahnsegmenten. N = (q, S), $q \subset L$, S={s(1), ... s(m)}, $s(j) \subset L$. Bei der Beschreibung der Wege ist e(0)=q, e(i)=s durch $e(0) \in q$, $e(i) \in s$ zu ersetzen.

3. Branch&Bound-Verfahren

Aufgabenstellung: (Extremwertaufgabe)

Gegeben sei eine Menge M und eine Wertefunktion w: $M \rightarrow R \cup \{\infty\}$ (R sei die Menge der reellen Zahlen.)
Man finde ein $m \in M$, so daß $w(m) = \min\{ w(x) : x \in M \}$ gilt.

Lösungsalgorithmus:

Es wird ein Baum von Knoten betrachtet, wobei jeder Knoten k eine gewisse Teilmenge $M_k \subseteq M$ repräsentiert. Der Wurzel k(0) entspricht die gesamte Menge M: $M_{k(0)} = M$. Sei s ein bestimmter Knoten und s(1), ... , s(n) seien die Nachfolgeknoten zu s. Dann soll $M_s = M_{s(1)} \cup \ldots \cup M_{s(n)}$ gelten. Jedem Knoten k wird ein Wert o(k) und ein Wert u(k) zugeordnet. o(k) stellt eine obere Grenze für das Minimum $\min\{ w(x) : x \in M \}$ dar. Das kann ein Wert zu einem Element e aus M_k sein, muß es aber nicht. Es kann auch der Wert w(e) zu einem beliebigen Element aus e genommen werden, z.B. der beste Wert, der bisher für ein Element aus M ermittelt wurde. u(k) stellt eine untere Schranke für die Werte über M_k dar: $u(k) \leq \min\{ w(x) : x \in M_k \}$. Ist für einen Knoten k der Wert u(k) größer als der Wert o(k), so wird das Minimum der Funktion w nicht in M_k angenommen. Die dort beginnenden Teilbäume brauchen nicht betrachtet werden.
Der Algorithmus besteht darin, nach einer bestimmten Vorschrift, wiederholt einen existierenden Knoten k auszusuchen und Nachfolgeknoten zu diesem zu bestimmen. Hierbei kommt es auf eine geeignete Auswahl des Knotens k, einer Zerlegung der Menge M_k und der Bestimmung von o(k) und u(k) an. Z.B. könnte als nächster Knoten k stets derjenige mit der kleinsten unteren Schranke u(k) gewählt werden. Das ist ein optimistisches Vorgehen, möglichst schnell einen Wert in der Nähe des Minimums zu finden.

3.1 Anpassung an das Verdrahtungsproblem

Auf den ersten Blick ähnelt der benutzte Ansatz einem, der in der Literatur oft zitiert wird: Das Rip-up-and-Reroute-Verfahren. Dabei werden die Netze einzeln nacheinander oder auch unabhängig voneinander verdrahtet. Um Überschneidungen in der Verdrahtung oder Probleme mit den Pfadlaufzeiten auf den Netzen zu lösen, wird nach bestimmten Vorschriften eine Menge von Netzen ausgewählt, für die man neue Realisierungen sucht. Dies wird solange fortgesetzt, bis eine gültige Verdrahtung gefunden ist.

Auch der hier verwendete erste Schritt ist eine Verdrahtung der Netze, wobei diese unabhängig voneinander untersucht werden. Dabei beanspruchen mehrere Netze eine große Anzahl von Leiterbahnsegmenten. Ebenfalls geht es bei dem Algorithmus darum, vorhandene Überschneidungen bei der existierenden Verdrahtung aufzulösen. Bei den Rip-up-and-Reroute-Verfahren werden gezielt Netze ausgewählt und neu verdrahtet. Dabei geht die Information über den vorhergehenden Zustand verloren. Über Heuristiken nähert man sich dem Optimum an. Es ist jedoch nicht sicher, daß dies erreicht wird.
Darin liegt der wesentliche Unterschied zu dem hier verwendeten Ansatz. Während der Abarbeitung des Algorithmus geht keine Information verloren, die zu einer besseren Lösung führen kann.

Branch&Bound-Verfahren zur Lösung des Verdrahtungsproblems für FPGAs:

Sei L die Menge der Leiterbahnsegmente, N die Menge der zu verdrahtenden Netze. Der Lösungsraum F besteht aus den Funktionen der Menge der Leiterbahnsegmente in die Menge der zu verdrahtenden Netze: $F = \{f: L \rightarrow N\}$. Jedem Element aus dem Lösungsraum, d.h. jeder solchen Funktion f, wird ein Wert w(f) wie folgt zugeordnet: Existiert keine Verdrahtung, so daß jedes Netz nur ihm zugeordnete Leiterbahnsegmente nutzt, so ist der Wert Unendlich. Gibt es eine solche Verdrahtung, so werden die längsten Pfade aller derartigen Verdrahtungen betrachtet. Der kleinste Wert eines längsten Pfades ist der Wert der Funktion.
Jeder Knoten k unseres Verzweigungsbaumes wird durch eine Teilmenge $F(k) \subseteq F$ der Menge der eben beschriebenen Funktionen repräsentiert.
Eine untere Schranke u(k) für den Wert der Elemente aus F(k) kann man wie folgt bestimmen. Für jedes Netz n betrachtet man die Menge der bei einer der Funktionen aus F(k) diesem Netz zugeordneten Leiterbahnsegmente: $L(k, n) = \{ l \in L : \exists f \in F(k): n = f(l) \}$. Danach wird für jede Quelle-Senke-Verbindung die kürzeste Realisierung betrachtet, die man erhält, indem nur Leitungen aus der entsprechenden Menge L(k,n) genutzt werden. Der längste Pfad der (möglicherweise nicht zulässigen) erhaltenen Realisierung der Netzliste ist eine untere Schranke für die Werte der Funktionen aus F(k).
Eine obere Schranke kann durch einen vorgegebenen Wert oder durch den längsten Pfad einer schon gefundenen Verdrahtung gegeben sein.

Die Beschreibung der Mengen F(k):

Sicher ist es nicht möglich, jedes Element f aus der Menge F(k) einzeln anzugeben. Wir haben uns für folgende Variante entschieden. Zu jedem Knoten k gibt es zwei Mengen $R_1(k)$, $R_2(k) \subseteq L \times N$, geordneter Paare von Leiterbahnsegmenten und Netzen. $R_1(k)$ beschreibt dabei vorgeschriebene Zuordnungen von Leiterbahnsegmenten zu Netzen und $R_2(k)$ verbotene Zuordnungen. Insbesondere gibt es zu jedem Leiterbahnsegment höchstens einen Eintrag in $R_1(k)$, und die Mengen $R_1(k)$ und $R_2(k)$ haben keine gemeinsamen Elemente. F(k) wird dann wie folgt beschrieben: $F(k) = \{ f \in F : \forall (l,n) \in R_1(k): f(l) = n, \forall (l,n) \in R_2(k): f(l) \neq n \}$.

Für die Zerlegung der Menge F_k zu einem Knoten k in Teilmengen und die Auswahl eines Knotens k, an dem die Suche fortgesetzt wird, sind bisher zwei Varianten untersucht worden.

3.1.1 Variante 1

Bei der ersten Variante lag das Hauptaugenmerk auf dem behutsamen Umgang mit Speicherplatz. Die Entscheidung fiel dabei auf eine Tiefensuche in einem binären Verzweigungsbaum. Zu jedem Knoten k werden ein Leiterbahnsegment l(k) und ein Netz n(k) ausgesucht. Für die Verzweigungsknoten k_1 und k_2 soll dann gelten: $R_1(k_1) = R_1(k) \cup \{ (l(k), n(k)) \}$, $R_2(k_1) = R_2(k)$ und $R_1(k_2) = R_1(k)$, $R_2(k_2) = R_2(k) \cup \{ (l(k), n(k)) \}$, d.h. im ersten Fall wird vorgeschrieben, daß das Leitungselement l(k) nur für das Netz n(k) genutzt werden darf, im zweiten Fall darf das Leitungselement l(k) nicht für das Netz n(k) genutzt werden. Es wird zuerst der Knoten k_1 und der sich dort ergebende Teilbaum untersucht. Erst nachdem dieser Teilbaum betrachtet wurde, wird der Knoten k_2 erzeugt und die Suche dort fortgesetzt.

3.1.2 Variante 2

Die Arbeit an der ersten Variante hat zur Idee für die Implementierung geführt, die ein vorsichtiges Umgehen mit Speicherplatz auch bei einem breiteren Verzweigungsbaum ermöglichen.
Auch hier wird zu jedem Knoten k ein Leiterbahnsegment l(k) betrachtet. Weiter wird aber nicht nur ein Netz betrachtet, sondern eine Menge von Netzen N(k) = { $n_1(k)$, $n_2(k)$, ... $n_j(k)$ }. Die Menge der Netze ergibt sich wie folgt: Unter Beachtung der beim Knoten k getroffenen Einschränkungen werden die kürzesten Wege für alle Quelle-Senke-Verbindungen bestimmt. Jede Quelle-Senke-Verbindung gehört zu einem Netz n. Nutzt der kürzeste Weg das Leiterbahnsegment l(k), so gehört n zur Menge N(k). Es gibt nun j+1 Nachfolgeknoten. Für k_r ($1 \le r \le j$) wird verlangt, daß l(k) höchstens von Netz $n_r(k)$ genutzt wird: $R_1(k_r) = R_1(k) \cup \{ (l(k), n_r(k))\}$, $R_2(k_r) = R_2(k)$. Für den letzten Knoten k_{j+1} wird die Nutzung des Leiterbahnsegmentes l(k) für alle Netze aus N(k) verboten: $R_1(k_{j+1}) = R_1(k)$, $R_2(k_{j+1}) = R_2(k) \cup \{(l(k), n_i(k)): 1<=i<=j\}$ Die Nachfolgeknoten werden alle gleichzeitig erzeugt.

Für die Auswahl eines Knotens, bei dem die Suche fortgesetzt wird, spielt folgende Kostenfunktion für die Knoten eine wichtige Rolle: Es werden dafür wieder die kürzesten Quelle-Senke-Verbindungen betrachtet, die bei dem entsprechenden Knoten möglich sind. Dadurch ergibt sich für jedes Leitungselement eine Menge von Netzen, die dieses Leiterbahnsegment nutzen möchten. Es werden alle Leiterbahnsegmente betrachtet, für die diese Menge mehr als ein Netz enthält. Die Summe der jeweils um eins verminderten Mächtigkeiten der zugehörigen Mengen von Netzen ergibt die Anzahl der Überschneidungen der betrachteten Verdrahtung und die Kosten für den Knoten. Ein weiteres Kriterium für die Auswahl eines Knotens ist die Stufe im Verzweigungsbaum. Dazu kommen spezifische aufgaben- und architekturabhängige Kosten. Das kann zum Beispiel die erwartete Länge des längsten Pfades sein.

4. Implementierung

Die Implementierung des verwendeten Algorithmus erfolgte in C++. Diese besteht im wesentlichen aus vier Teilen:
- Netzliste,
- Architekturinformation,
- Kürzester Weg-Algorithmus und Signallaufzeitmodell
- Branch&Bound-Algorithmus.

Die Netzliste enthält Angaben über die zu verdrahtenden Netze. Das sind im wesentlichen die Zuordnung der Quellen und Senken.

Die Architekturinformation verfügt über Angaben zur Struktur des zu verdrahtenden FPGAs, zur Lage der Leitungen und ihrer Beziehungen zu Ein-/Ausgabeblöcken, Gattern, logischen Blöcken usw. Aufgrund des hohen Zeitaufwandes wurden bisher nur Informationen zur Architektur der FPGAs der Xilinx XC3000er Familie implementiert.

Als Algorithmus zum Auffinden kürzester Quelle-Senke-Verbindungen wurde eine Variante von Dijkstras Algorithmus verwendet, wobei jedoch die Suche von Quelle und Senke aus gestartet wird. Zur Zeit ist keine architekturspezifische Bestimmung der Signallaufzeiten implementiert. Es wird einfach die Anzahl der programmierbaren Schalter (PIPs) gezählt. Sollte später bei einer architekturspezifischen Bestimmung die Abschätzung bei der Suche von den Senken aus problematisch sein, so kann man sich auf eine Vorwärtssuche beschränken. Der Algorithmus kann jederzeit durch einen anderen ersetzt werden, z.B. wenn sich eine andere Suche durch die Architektur anbietet.

Die Implementierung des Branch&Bound-Algorithmus

Beim Branch&Bound-Algorithmus entsteht ein Baum von Suchknoten. Bei der Suche werden nach bestimmten Entscheidungskriterien stets ein Blatt und ein zu betrachtendes Leiterbahnsegment ausgewählt. Dazu werden die nachfolgenden Knoten erzeugt, solange keine Lösung der Aufgabe gefunden wurde. Für die nachfolgenden Knoten wird die Erfüllung der Bedingungen der Aufgabenstellung überprüft. Daraufhin werden sie in die Liste der vorhandenen Knoten aufgenommen oder verworfen.

Die Entscheidungskriterien zur Auswahl des Blattes sind unterschiedlich einstellbar. Es sind Kombinationen der folgenden Kriterien möglich:

- Überschneidungen bei der Verdrahtung mit den zugehörigen kürzesten Quelle-Senke-Verbindungen.
- Die Stufe des Blattes im Baum. Dadurch kann eine Tiefensuche erreicht werden.
- Architekturabhängige Kriterien.

Für die Auswahl des Leiterbahnsegmentes können ebenfalls verschiedene Kriterien herangezogen werden. Für die verwendeten Beispiele ist folgende Variante gewählt worden: Jedem Leiterbahnsegment wird eine Priorität zugeordnet. Am Anfang werden alle Prioritäten auf 1 gesetzt. Wurde bei einem Knoten ein bestimmtes Leiterbahnsegment gewählt und ist die Suche von diesem Knoten ausgehend erfolglos oder verschlechtern sich die betrachteten Zielfunktionen für alle direkten Nachfolgerknoten, so ist zu erwarten, daß die Wahl dieses Leiterbahnsegmentes solche Probleme auch bei anderen Knoten erkennen läßt. Entscheidungen bezüglich dieses Leiterbahnsegmentes sollten daher auf möglichst geringer Stufe im Suchbaum getroffen werden und seine Priorität für die Auswahl wird erhöht. Ein weiteres Entscheidungskriterium für die Auswahl ist die Anzahl der Überschneidungen von Netzen an den Leiterbahnsegmenten. Leiterbahnsegmente mit vielen Überschneidungen werden möglichst früh gewählt. Außerdem werden solche Leiterbahnsegmente möglichst früh gewählt, die auf einem kürzesten Weg besonders nah an Quelle oder Senke liegen. Für diese Leiterbahnsegmente erwartet man nicht so viele Freiheiten.

Bei der Betrachtung eines Knotens und der Erzeugung der nachfolgenden Knoten werden die kürzesten Quelle-Senke-Verbindungen benötigt, die zu diesem Knoten gehören. Der Aufwand, diese jedesmal bei der Betrachtung eines Blattes des Suchbaumes neu zu berechnen oder zu jedem Blatt sämtliche Quelle-Senke-Verbingungen zu speichern, erscheint unverhältnismäßig hoch. Faßt man die benötigten Quelle-Senke-Verbindungen in einer Menge zusammen und beseitigt Duplikate, (die bei der Erzeugung neuer Knoten und der dort nötigen Suche nach kürzesten Quelle-Senke-Verbindungen entstehen,) ergibt sich eine erstaunlich geringe Zahl benötigter Wege.
Aber auch dann erscheint es nicht sinnvoll, an jedem Blatt des Suchbaumes für jede Quelle-Senke-Verbindung einen Verweis auf den entsprechenden kürzesten Weg zu benötigen. Das Problem wurde dadurch gelöst, indem für jeden Knoten des Suchbaumes die Änderungen der kürzesten Wege vermerkt werden, d.h. für alle Quelle-Senke-Verbindungen, für die der kürzeste Weg des vorhergehenden Knotens aufgrund der Einschränkungen des Suchraumes nicht mehr in Frage kommt, wird der nun kürzeste Weg vermerkt. Änderungen zu kürzesten Wegen, die an keinem Blatt mehr benötigt werden, können entfernt werden.
Mit der Beschreibung der den Knoten zugeordneten Teilmengen F(k) des Suchraumes (siehe Abschnitt Branch&Bound-Verfahren zur Lösung des Verdrahtungsproblems für FPGAs) ist genauso verfahren worden. Zu jedem Knoten wird das betrachtete Leiterbahnsegment vermerkt und das Netz, für das es reserviert werden soll, bzw. die Netze, für die die Nutzung des Leiterbahnsegmentes verboten wird.
Beim Übergang von der Betrachtung eines Blattes des Suchbaumes zu einem neuen erhält man die Angaben für das neue Blatt aus den Angaben für das alte, indem man alle Vorgängerknoten der beiden Blätter untersucht.

5. Ergebnisse

Es wurden 2 Varianten eines Branch&Bound-Algorithmus für die Verdrahtung von FPGAs implementiert. Die erste Variante soll hier nicht weiter betrachtet werden, da sich ihr Verhalten mit der zweiten Variante weitgehend nachbilden läßt.
Mit der zweiten Variante wurden verschiedene einfache Beispiele durchgerechnet und damit die prinzipielle Anwendbarkeit von Branch&Bound-Algorithmen auf das Verdrahtungsproblem für FPGAs gezeigt. Es wurde von Plazierungen und Verdrahtungen ausgegangen, die mit dem Xilinx Plazierer und Verdrahter PPR erzeugt wurden.

5.1 Länge des kritischen Pfades

Bezüglich des von uns verwendeten Optimierungskriteriums konnte dabei eine deutliche Verkürzung der längsten Pfade gegenüber einer Berechnung mit dem Xilinx Plazierer und Verdrahter (PPR) erreicht werden (Tabelle 1).

Schaltung	Verwendeter FPGA	Anzahl logischer Blöcke	Anzahl I/O-Blöcke	Länge des kritischen Pfades	
				Vorgabe	Ergebnis
5xp1	3120	9	17	27.25	23.25
9sym	3120	7	10	29.25	28.25
alu2	3120	41	16	49.25	40.25
b12	3120	16	24	28.25	28.25
b9	3120	31	62	35.25	30.25
c8	3120	21	46	36.25	34.25
cc	3120	13	41	26.25	22.25
clip	3120	12	14	30.25	27.25
cm138a	3120	9	14	24.25	21.25
cm150a	3120	10	22	44.25	41.25
cm151a	3120	6	14	33.25	30.25
cm152a	3120	5	12	28.25	27.25
cm162a	3120	9	19	35.25	32.25
cm163a	3120	8	21	23.25	21.25
cm85a	3120	7	14	25.25	25.25
cmb	3120	8	20	32.25	30.25
con1	3120	3	9	19.25	18.25
cordic	3120	12	25	38.25	32.25
count	3120	25	51	61.25	50.25
cu	3120	13	25	37.25	34.25
f51m	3120	8	16	27.25	25.25
frg1	3120	34	31	83.25	72.25
i1	3120	11	41	26.25	24.25
inc	3120	18	16	28.25	24.25
lal	3120	19	45	32.25	28.25
misex1	3120	10	15	24.25	22.25
mux	3120	10	22	46.25	38.25
pcle	3120	14	28	45.25	40.25
pcler8	3120	27	44	50.25	40.25
pm1	3120	12	29	23.25	21.25
rd73	3120	6	10	23.25	21.25
rd84	3120	8	12	25.25	22.25
sao2	3120	17	14	39.25	37.25
sct	3120	15	34	28.25	25.25
t481	3120	5	17	24.25	23.25
term1	3120	19	44	49.25	42.25
ttt2	3120	26	45	38.25	33.25
unreg	3120	17	52	26.25	23.25
vg2	3120	27	33	49.25	40.25
x2	3120	8	17	24.25	22.25
z4ml	3120	4	11	21.25	20.25
apex7	3142	43	86	48.25	41.25

Tabelle 1: Untersuchte Schaltungen und Länge des kritischen Pfades nach dem verwendeten Signallaufzeitmodell

Demgegenüber zeigen die mit dem Programm xdelay von Xilinx berechneten Längen der längsten Pfade in den meisten Fällen eine Verschlechterung (Tabelle 2). Das liegt an den verschiedenen Berechnungen des längsten Pfades. Leider liegen nicht genügend Informationen vor, um mit den gleichen Daten und dem gleichen Modell wie Xilinx arbeiten zu können.Wäre das von Xilinx verwendete Modell bekannt, so könnte auch bezüglich dieses Modells eine Verdrahtung mit kleinstmöglichem längsten Pfad ermittelt werden.

Schaltung	Länge des kritischen Pfades (xdelay) [ns]		Schaltung	Länge des kritischen Pfades (xdelay) [ns]		Schaltung	Länge des kritischen Pfades (xdelay) [ns]	
	Vorgabe	Ergebnis		Vorgabe	Ergebnis		Vorgabe	Ergebnis
5xp1	20.3	21.7	cm85a	16.5	18.3	pcler8	33.2	33.8
9sym	18.2	21.0	cmb	21.8	25.7	pm1	17.8	19.4
alu2	35.2	35.3	con1	15.3	14.4	rd73	15.4	16.0
b12	20.9	20.6	cordic	25.4	30.6	rd84	17.2	22.8
b9	25.8	28.2	count	42.8	47.1	sao2	26.6	32.7
c8	25.2	26.5	cu	24.6	24.0	sct	23.1	23.7
cc	21.8	25.2	f51m	17.9	19.4	t481	17.3	16.8
clip	22.2	25.1	frg1	55.0	56.0	term1	34.4	38.6
cm138a	16.5	19.2	i1	16.8	20.9	ttt2	27.1	30.0
cm150a	30.0	36.8	inc	20.9	22.8	unreg	18.9	20.8
cm151a	22.4	23.1	lal	23.2	23.9	vg2	34.4	34.9
cm152a	18.2	19.1	misex1	16.8	20.3	x2	17.0	19.7
cm162a	23.2	25.2	mux	30.0	33.9	z4ml	13.3	13.1
cm163a	16.4	18.2	pcle	33.0	35.4	apex7	41.1	41.7

Tabelle 2: Berechnete Längen der kritischen Pfade nach dem Programm xdelay (Xilinx)

5.2 Programmlaufzeiten

Für jedes Beispiel wurden Programmläufe mit verschiedenen Einstellungen untersucht (Tabelle 3). In den ersten drei Fällen (A-C) wurde von vorhandenen Verdrahtungen ausgegangen und Verbesserungen für diese gesucht. Dabei wurden die Suchknoten wie folgt ausgewählt:

A) Es wird ein Suchknoten betrachtet, der die größte Tiefe im Suchbaum hat und dabei möglichst wenig Überschneidungen der ausgewählten minimalen Wege.
B) Es werden die Knoten mit der größten Tiefe im Suchbaum betrachtet, daraus diejenigen mit möglichst wenig Überschneidungen, daraus wiederum diejenigen mit dem kürzesten kritischen Pfad. Diese Knoten werden alle betrachtet, maximal jedoch 10 von ihnen, und ihre Nachfolgeknoten erzeugt, bevor eine neue Auswahl getroffen wird.
C) Das Vorgehen hier entspricht einer optimistischen Strategie. Wir betrachten zuerst die Knoten mit wenig Überschneidungen, danach die Tiefe im Suchbaum und dann die Länge des kritischen Pfades.
D) Es wird wie bei C) vorgegangen, jedoch wird nicht von einer Startverdrahtung ausgegangen. Es war keine maximale Länge des kritischen Pfades vorgegeben.

Schaltung	Programmlaufzeiten [s]				Anzahl der betrachteten Suchknoten				Maximale Anzahl minimaler Wege			
	A	B	C	D	A	B	C	D	A	B	C	D
5xp1	3	13	3	2	90	331	97	89	175	190	189	162
9sym	1	5	1	1	52	303	52	53	92	93	92	91
alu2	564	2481	1686	1652	5288	18224	8863	8845	1297	1254	1192	1218
b12	1	1	1	10				162				310
b9	89	822	135	156	950	8216	1082	1384	749	803	789	750
c8	23	88	22	18	446	1255	419	247	442	451	441	453
cc	8	57	8	8	218	1371	219	219	216	255	216	211
clip	12	50	17	16	278	1023	381	290	255	271	257	250
cm138a	13	57	35	16	367	1303	895	459	215	217	215	220
cm150a	3	20	3	4	198	1137	198	206	139	148	139	141
cm151a	1	2	1	1	86	203	86	65	84	87	84	82
cm152a	0	2	0	0	37	171	37	40	50	57	50	62
cm162a	5	40	5	6	165	1258	161	218	162	177	159	164
cm163a	1	7	2	2	83	332	74	72	122	138	122	118
cm85a	1	2	1	1	41	89	41	100	64	71	64	112
cmb	1	12	2	1	133	764	137	134	122	131	124	116
con1	0	0	0	0	27	92	27	27	34	37	35	37
cordic	6	40	6	6	234	1135	232	236	176	186	176	177
count	52	303	85	69	896	5185	850	924	528	535	488	461
cu	13	26	17	9	254	585	286	142	260	277	269	260
f51m	2	10	2	2	70	538	91	80	130	148	133	132
frg1	148	782	277	307	1983	8937	2515	3112	949	945	927	965
i1	3	30	3	3	206	1786	207	206	149	168	149	152
inc	182	288	125	63	1281	2859	1040	430	472	474	474	441
lal	9	48	9	12	189	1001	186	296	354	348	347	332
misex1	8	45	9	6	189	1087	201	214	173	185	174	164
mux	3	16	3	3	118	534	118	126	152	170	152	148
pcle	18	65	26	22	321	982	477	344	309	326	306	292
pcler8	121	940	613	976	1395	8226	2560	4923	754	812	815	826
pm1	8	33	7	8	127	551	122	120	242	243	226	219
rd73	1	5	1	1	49	251	49	50	89	98	89	91
rd84	2	16	2	3	73	700	73	125	145	163	145	132
sao2	15	71	52	23	283	1488	484	395	345	365	368	351
sct	20	130	22	23	322	2492	328	419	305	316	305	292
t481	0	3	0	0	43	386	43	43	53	59	54	58
term1	13	68	14	26	199	836	203	501	380	400	380	350
ttt2	133	1274	305	266	1450	11632	2678	2369	681	695	670	677
unreg	41	302	41	40	245	1562	248	257	290	321	291	296
vg2	100	281	72	97	1593	3334	818	952	677	708	694	742
x2	2	11	2	2	73	567	77	76	121	130	120	116
z4ml	0	2	0	0	39	277	40	41	60	65	63	68
apex7	335	1512	983	1362	2666	6785	3055	2620	1041	1154	1100	1053

Tabelle 3: Programmlaufzeiten

Beim Vergleich der Berechnungen A) und B) ist zu erkennen, daß die Anzahl der maximalen Zahl auf einmal benötigter Wege meist nur gering angestiegen ist, obwohl deutlich mehr Knoten betrachtet wurden (und auch gleichzeitig im Suchbaum vorhanden waren). Die maximale Anzahl der gleichzeitig betrachteten Wege liegt bei der Berechnung B) im Durchschnitt um 6,8 Prozent über der Anzahl bei A). Demgegenüber stieg die Anzahl der betrachteten Suchknoten auf das 5,2-fache an.

Die geringste Anzahl von Suchknoten mußte für die meisten Beispiele bei der Berechnung A), d.h. bei konsequenter Tiefensuche, betrachtet werden. Das läßt sich sicherlich damit erklären, daß problematische Leiterbahnsegmente oft erst bei Suchknoten mit großen Tiefen erkannt werden können. Bei der Tiefensuche werden sie dann innerhalb anderer Zweige des Baumes früher betrachtet.

Der Fall D) zeigt, daß das Programm auch ohne vorgegebene Verdrahtung genutzt werden kann.

6. Ausblick

Für die angegebenen Beispielschaltungen konnte die Anwendbarkeit von Branch&Bound-Algorithmen auf das Verdrahtungsproblem für FPGAs gezeigt werden. Entsprechende Ergebnisse für sehr komplexe Schaltungen liegen momentan noch nicht vor. Untersuchungen dazu werden im Rahmen von Graduierungsarbeiten vorgenommen. Im Zusammenhang damit werden im Interesse einer Verbesserung des Algorithmus verschiedene Ansätze verfolgt.
Dazu gehört die Aufteilung eines Problems in Teilprobleme, die dann zusammengefaßt werden. Es können z.B. zwei Teilmengen der Netze betrachtet werden. Wenn dort scheinbar eine neue Verdrahtung möglich ist, können Kombinationen mit anderen Teilmengen gebildet werden.
Es ist zu untersuchen, welche Konstellationen zum Abbruch der Suche an einem Suchknoten führen. Allgemeine Eigenschaften solcher Konstellationen können zu einer Entwicklung besserer Suchstrategien führen. Zusätzlich kann die Untersuchung solcher Konstellationen während des Programmlaufes genutzt werden, um Schlußfolgerungen für das Vorgehen bei den noch nicht betrachteten Suchknoten abzuleiten. Für diesen Ansatz spricht, daß bei der Tiefensuche die besten Ergebnisse erreicht wurden. Diese sind auf die Markierung der als schwierig erkannten Leiterbahnsegmente zurückzuführen, die dann mit erhöhter Priorität zur Verzweigung genutzt wurden.
Die Abschätzung, welche Suchknoten zum Ziel führen können, muß verbessert werden (Kombination mit Heuristiken).

8. Danksagung

Für die vielen helfenden Diskussionen und Anregungen bei der Anfertigung der vorliegenden Arbeit sowie für die technische Unterstützung sei den Herren Prof. K. Antreich, Prof. F. Johannes und Dipl.-Inf. M. Senn des Lehrstuhls „Rechnergestütztes Entwerfen“ der Technischen Universität München gedankt.

Literatur

[ALRO96] Michael J. Alexander, Gabriel Robins
New Performance-Driven FPGA Routing Algorithms.
IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems, Vol. 15, Nr. 12, Dezember 1996

[BERO96] Vaughn Betz und Jonathan Rose
Directional Bias and Non-Uniformity in FPGA Global Routing Architectures
International Conference on Computer Aided Design, November 1996

[BRRV92] Stephen Brown, Jonathan Rose und Zvonko G. Vranesic
A Detailed Router for Field-Programmable Gate Arrays
IEEE Transactions on Computer-aided Design, Vol. 11, Nr. 5, Mai 1992

[KOMB81] L. Kou, G. Markowsky und L. Berman
A fast algorithm for steiner trees
Acta Informatica, vol. 15, pp.141-145, 1981

[LEBR93] Guy G. Lemieux und Stephen D. Brown
A Detailed Routing Algorithm for Allocating Wire Segments in Field-Programmable Gate Arrays
Proceedings of ACM Physical Design Workshop, April 1993

[LEE61] C. Lee
An algorithmus for path connections and its applications
IEEE Transactions on Electronic Computers, September 1961

[LEWU95] Yuh-Sheng Lee und Allen C.-H. Wu
A Performance and Routability Driven Router for FPGAs considering Path Delays

[MHS96] Ulrich Möhrke, Paul Herrmann, Marco Schmidt
Das Xilinx_LCA-Format
Institut für Informatik, Universität Leipzig, Report Nr.2, April 1996

[RALJ96] Srilata Raman, C.L. Liu und L.G. Jones
A Timing Constrained Incremental Routing Algorithm for Symmetrical FPGAs
in Proceedings of European Design and Test Conference, 1996

[ROGS93] Jonathan Rose, Abbas el Gamal und Alberto Sangiovanni-Vincentelli
Architecture of Field-Programmable Gate Arrays
Proceedings of the IEEE, Vol. 81, Nr. 7, Juli 1993, Seiten 1013-1029

[TCWM97] Shashidhar Thakur, Yao-Wen Chang, D.F. Wong und S. Muthukrishnan
Algorithms for an FPGA Switch Module Routing Problem with Application to Global Routing
IEEE Transactions on Computer-aided Design of integrated Circuits and Systems, Vol. 16, Nr. 1, Januar 1997

[TRIM95] Stephen Trimberger
Effects of FPGA Architecture on FPGA Routing
ACM 32nd Design Automation Conference, Juni 1995

[WCMT97] Yu-Liang Wu, Douglas Chang, Malgorzata Marek-Sadowska und Shuji Tsukiyama
Not Necessarily More Switches More Routability
ASP-DAC, 1997

[WUTM96] Yu-Liang Wu, Shuji Tsukiyama und Malgorzata Marek-Sadowska
Graph Based Analysis of 2-D FPGA Routing
IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems, Vol. 15, Nr. 1, Januar 1996

[ZELI93] A. Z. Zelitkovski
An 11/6 approximation algorithm for the network steiner problem
Algorithmica, vol. 9, pp. 463-470, 1993