1 Einleitung

Im Bereich des elektronischen CAD, dem rechnergestützten Entwurf von VLSI-Schaltungen, entwickelt sich Floorplanning zunehmend zu einer zentralen und anspruchsvollen Aufgabe.

Der Komplexitätsanstieg und der Wunsch nach ständig steigender Performanz der zu realisierenden integrierten Digitalschaltungen bedingen entsprechend höhere Anforderungen auf der Seite der Entwurfswerkzeuge (Tools) und -methodik. Durch den hierarchischen Entwurfsstil mit Top-down-Planungsphase ([Sur88]) und der Möglichkeit der Drei-Phasen-Planung ([ASZ92]) ist das PLAYOUT-Entwurfssystem, entwickelt in der Arbeitsgruppe von Professor Zimmermann am Fachbereich Informatik der Universität Kaiserslautern, von der methodischen Seite her dafür ausgelegt.

Für die Tools war zunächst die Minimierung der Chipfläche, bzw. das Einhalten diesbezüglicher Spezifikationen, oberste Prämisse. So wurden im Bereich Flächenabschätzung auch wichtige Forschungsergebnisse ([Zim90], [Zim88]) erzielt. In jüngerer Zeit richtete sich das zweite Hauptziel auf die Optimierung der Performanz, also die Minimierung der maximalen Verzögerungszeit, bzw. das Einhalten von Verzögerungszeitspezifikationen. In [Heb95] werden erste Ergebnisse für den Bereich der Repartitionierung, also für die Transformation von Struktur nach Struktur, dokumentiert. Das Hauptziel meiner Arbeit ist es, Methoden des timing-driven Floorplanning zu erforschen und ihre Anwendbarkeit im Kontext des PLAYOUT-Systems nachzuweisen.

Im folgenden werden der Aufbau und die Teilziele dieser Arbeit näher erläutert. Nach einem Überblick über den hierarchischen VLSI-Entwurf und den Stand einiger kommerzieller Systeme auf diesem Gebiet werden in **Kapitel 2** wichtige Teilprobleme des timing-driven Floorplanning formalisiert.

Die Timinganalyse ([Sol91]) benötigt mit dem Timinggraphen eine aus der Strukturbeschreibung abgeleitete Datenstruktur. Sie berechnet geforderte und (abgeschätzte) tatsächliche Signalankunftszeiten an den Knoten des Timinggraphen und stellt mit deren Differenz, dem Schlupf (Slack), ein weiteres wichtiges Kriterium zur Beurteilung des Zeitverhaltens und zur Steuerung von Algorithmen, die es optimieren, zur Verfügung.

Kernpunkt des Plazierungsproblems, so wie es in der Regel im VLSI-Entwurf anzutreffen ist, ist die überschneidungsfreie Anordnung rechteckiger Flächen (für die Subzellen) innerhalb eines umschließenden Rechtecks (auch CuD-Rahmen genannt). Das Floorplanningproblem (im engeren Sinne) verallgemeinert das Plazierungsproblem dahingehend, daß die Subzellen

eine Reihe von Formalternativen, also Rechtecke unterschiedlicher Breite und Höhe und auch unterschiedlicher Fläche, besitzen. Die Einschränkung auf binäre Slicingstrukturen erleichtert u. a. den Algorithmenentwurf, d. h. das Finden effizienter Heuristiken zur näherungsweise optimalen Lösung np-vollständiger Probleme, und erlaubt eine effiziente Flächenberechnung bzw. -abschätzung (Sizing).

Setzt man rekursive Bipartitionierung als Plazierungstechnik ein, so sind binäre Slicingbäume ein adäquates Beschreibungsmittel für die entstehende Aufteilung der Schaltung. Das Partitionierungsproblem besteht darin, einen Hypergraphen in eine vorgegebene Anzahl von Teilgraphen (Partitionen) zu zerlegen mit dem Ziel der Minimierung der Schnittkosten und unter Einhaltung von Randbedingungen (Ausgewogenheit der Flächenverhältnisse).

Das Verdrahtungsproblem besteht bei der Flächenplanung in der globalen Verdrahtung (Global Routing), dem Zuweisen von Netzverläufen zu Verdrahtungsflächen, z. B. Kanälen zwischen den Zellen oder Überzellflächen, unter den Gesichtspunkten Netzlängen- oder Flächenminimierung und Optimierung des Zeitverhaltens. Infolgedessen wird Detailverdrahtung hier auch nicht näher untersucht.

Pin Assignment und Intervallfestlegung sind zwei Aufgaben, die im Überschneidungsbereich von Formauswahl und globaler Verdrahtung liegen. Deshalb sind sie bei der Problembeschreibung weder dem einen noch dem anderen untergeordnet, sondern werden eigenständig behandelt.

Die vorgenannten Aufgaben werden zu den drei großen Themenkomplexen Plazierung, globale Verdrahtung und Abschätzung zusammengefaßt, für die **Kapitel 3** dann den Stand der Technik aufarbeitet.

Die vorgenommene Klassifizierung der gebräuchlichen Plazierungsverfahren orientiert sich an [ShM91]: Plazierung mittels Simulated Annealing, mit Kräftemodellen, durch Partitionierung, mit Methoden der numerischen Optimierung und mit genetischen Algorithmen. Anschließend werden die drei verbreiteten Methoden der Berücksichtigung von Zeitverhalten (Netzgewichtung, Netzrestriktionen wie z. B. Netzlängenvorgaben, pfadbasierte Verfahren) dargestellt und kombiniert mit den besprochenen Plazierungsalgorithmen.

Die Ausführungen zur globalen Verdrahtung beginnen mit der Konstruktion verschiedener Arten von Routinggraphen (z. B. Channel Intersection-Graph und Floor Connection-Graph) und schildern detailliert sequentielle und parallele Routingverfahren. Auch hier schließen sich Betrachtungen über Verfahren, die auch die Verzögerungszeiten berücksichtigen, an. In erster Linie sind dieses modifizierte Steinerbaumheuristiken, die zwischen zeitkritischen und unkritischen Senken unterscheiden (also auf Ergebnisse der Timinganalyse angewiesen sind) und

mehr Wert auf eine Minimierung der Verzögerungszeit zu den zeitkritischen Senken als auf eine minimale Netzlänge legen.

Chipfläche und maximale Verzögerungszeit sind die beiden Hauptkriterien zur Bewertung der Güte von Entwürfen (Planungsergebnissen). Eine gute Abschätzung der benötigten Verdrahtungsfläche ist Kernstück des hierarchischen Top-down-Entwurfs. Für eine akkurate Abschätzung der Verzögerungszeiten der Netze werden RC Delay-Modelle (Elmore Delay-Modell, $t_{0,7}$-Modell) herangezogen. Die Summe der Verzögerungszeiten der Zellen und Netze auf einem Pfad ergibt die Pfadverzögerungszeit. Die maximale Pfadverzögerungszeit zu minimieren, ist Optimierungsziel der zeitgesteuerten Algorithmen. Bestimmt wird sie mittels der bereits angesprochenen Timinganalyse auf dem Timinggraphen zur Schaltung. Angemerkt werden soll an dieser Stelle, daß mit *Zeit* im Kontext dieser Arbeit stets die Verzögerungszeit (Delay) gemeint ist (nicht etwa die Laufzeit von Algorithmen).

Über Partitionierung wurde in den letzten Jahren sehr viel publiziert, z. B. in Form von Übersichtsartikeln ([Joh96]) und ausführlichen Zusammenstellungen ([AlK95]), so daß es angemessen erscheint, ihr in diesem Kapitel nur unterproportional Platz (lediglich als Teilkapitel zur Plazierung) einzuräumen. Dafür werden am Ende des Kapitels noch einige interessante Floorplanningansätze aus der Literatur (u. a. Mason von [LaD86], PEPPER von [NLG95] und EXPLORER von [EsK96]) diskutiert.

Kapitel 4 erläutert, wie diese drei großen Aufgaben im Chip Planner des PLAYOUT-Entwurfssystems angegangen werden. Der Schwerpunkt der Beschreibung liegt dabei auf der aktuellen Version, dem *planner.v6*, mit dem zum ersten Mal in PLAYOUT ein Floorplanning-Tool zur Verfügung steht, das die Entwurfsziele Chipfläche und maximale Verzögerungszeit gleichberechtigt betrachtet und zu optimieren versucht.

Zunächst werden der Chip Planner in das PLAYOUT-Entwurfssystem eingeordnet, seine Eingabe- und Ausgabe-Beziehungen mit und seine Abhängigkeiten von den anderen Tools beschrieben, sowie deren Funktionalität skizziert.

Bei der Wahl von rekursiver, FM-Mincut-basierter Bipartitionierung (FM-Mincut ist die auf Fiduccia und Mattheyses, [FiM82], zurückgehende Variante des Mincut-Algorithmus) als Grundlage des Plazierungsverfahrens unterscheiden sich *planner.v6* und *planner.v5* nicht. Jedoch sind die aufgesetzten Inplace-Verfahren (auch bekannt unter dem Namen Terminal-Propagierung bzw. Terminal Propagation) in wesentlichen Aspekten verschieden.

Im *planner.v5* bestand die Plazierung noch aus einem dreistufigen Verfahren: Rekursive Partitionierung, h/v-Orientierung (h/v = horizontal/vertikal) mit dem Verfahren der optimalen Überlagerung und l/r-Ordnung (l/r = links oder unten / rechts oder oben). Da die Partitionie-

rung bis auf die Vorgabe der Orientierung der obersten drei Slicinglinien keine weitere Information über die Orientierung von Slicinglinien hatte, erschwerte sich die Auswertung externer Netzverbindungen, die nur auf der Basis sogenannter “Nachbarschaftsbeziehungen” ([PaZ90]) beruhte. Mit jedem Partitionierungsschritt verringerte sich der auswertbare Bereich, und es wurden dadurch (auf den unteren Ebenen des Slicingbaums) viel weniger externe Pins bei der Terminal-Propagierung berücksichtigt als beim ursprünglichen Verfahren von [DuK85]. Zudem hatte der Entwerfer wenig Anhaltspunkte zur günstigen Festlegung der Orientierung der obersten drei Slicinglinien, so daß diese meist willkürlich geschah.

Der *planner.v6* arbeitet dagegen mit einem einstufigen Plazierungsverfahren, das bei jedem Bipartitionierungsschritt die Kosten für beide Orientierungsmöglichkeiten der Slicinglinie berechnet und vergleicht (optional sogar mit einer Look-ahead-Strategie). Der günstigere Schnitt wird ausgewählt und eine Aufteilung der (abgeschätzten) Flächen vorgenommen, d. h. es wird bei der Terminal-Propagierung stets eine “Pseudotopographie” ([Fri96]) zugrunde gelegt. Dieses Verfahren bildet die Grundlage für verschiedene Erweiterungen zur Berücksichtigung des Zeitverhaltens, die detailliert in den Unterkapiteln 4.2.2 (TPmK: Terminal-Propagierung mit Klassifikation) und 4.2.3 (TPmB: Terminal-Propagierung mit Budgetierung) beschrieben sind.

Für die globale Verdrahtung wurde ein Typ von Routinggraph gesucht und gefunden, der eine durchgängige Behandlung von Kanalverdrahtung (Zwischenzellverdrahtung, typischerweise repräsentiert in einem Channel Intersection-Graph) und Überzellverdrahtung (Floor Connection-Graph) erlaubt. Die naheliegende Kombination dieser beiden Graphen, wie sie z. B. [Len90] vorschlägt, wurde jedoch verworfen zugunsten einer uniformen, erweiterten B-Kanal-Aufteilung auf der Basis des Channel Intersection-Graphen. (Der in PLAYOUT verwendeten Klassifizierung der Kanaltypen in A-, B- und C-Kanäle, [Anh89], ist das Unterkapitel 4.3.1 gewidmet.)

Das performance-driven Global Routing im *planner.v6* ist gekennzeichnet durch die folgenden Punkte ([Fri98]):

- sequentielles Routing der Netze, wobei die zeitkritischen Netze zuerst (sortiert nach der Kritikalität) verlegt werden und anschließend die unkritischen Netze
- SERT-basierte Steinerbaumheuristik für die zeitkritischen Netze (SERT = Steiner Elmore Routing Tree)
- kostenminimale Verdrahtung der unkritischen Netze (SMT-Heuristik, SMT = Steiner Minimal Tree), wobei die abgeschätzte Chipfläche mit in die Kostenberechnung eingeht, dadurch daß ein Overload-Faktor in Abhängigkeit vom Belegungszustand der Kanäle die Kantengewichtung des Routinggraphen dynamisch beeinflußt.

Die Abb. 1.1 zeigt übersichtsartig, in einem vereinfachten Klassifikationsschema, die Einordnung und die bisher angesprochenen Aufgaben und Bestandteile des Floorplanning im hierarchischen VLSI-Entwurfssystem PLAYOUT.

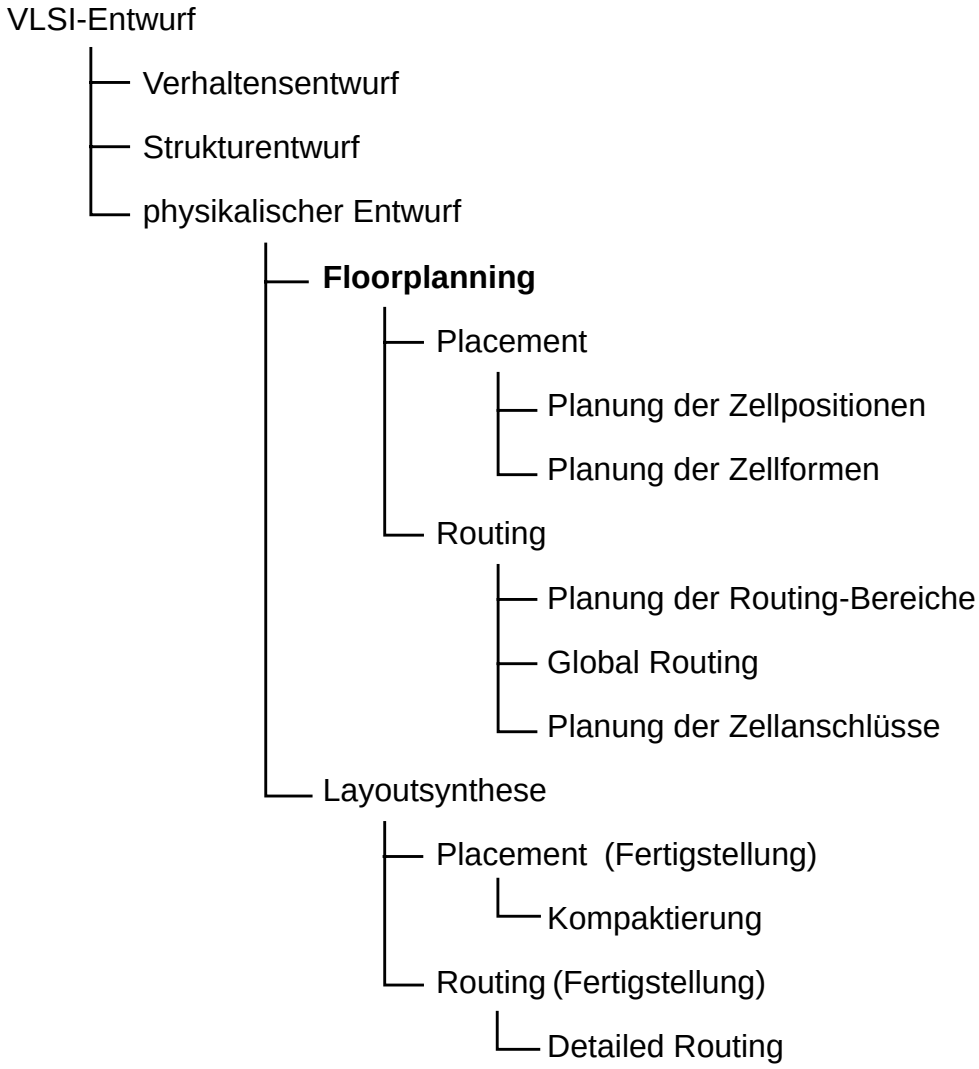

Abb. 1.1: Floorplanning in PLAYOUT

Die in jüngster Zeit zur Taktnetzverdrahtung vorgelegten Algorithmen (Zero-Skew Clock Routing, [Tsa93], [Eda94]) ähneln denen der zeitgesteuerten globalen Verdrahtung in vielen Aspekten. Dennoch werden ihnen eigene Unterkapitel (Kap. 3.4, bzw. Kap. 4.5 über die Realisierung im Chip Planner, [Kur96]) zugestanden, da sie mit der ausgewogenen Bipartitionierung der Taktnetzsenken eine weitere Aufgabe zu lösen haben. Da zu diesem Thema schon eine ausführliche Untersuchung ([HNZ93]) im Kontext des PLAYOUT-Projekts dokumentiert ist, beschränkt sich diese Arbeit auf die Herausstellung der wesentlichen Ideen.

Ein zentraler Punkt ist auch in Kapitel 4 die Abschätzung. Während für Flächenabschätzung in PLAYOUT eine Reihe detaillierter Studien ([Zim88], [Heb95], [Hes99]) existieren, ist die Implementierung im CHIP Planner die erste vollständig umgesetzte, hierarchische Modellierung des Zeitverhaltens. Es wird eine strukturelle Äquivalenz des definierten und verwendeten hierarchischen Timinggraphen mit dem Fortpflanzungsmodell von [Kel89] aufgezeigt, jedoch dessen Konzept der Realisierungssicherheit verworfen (und durch die Verwendung abgeschätzter Werte ersetzt). Der hierarchische Timinggraph unterscheidet sich auch grundlegend von den flachen Versionen, Verbindungsgraph und Registerverbindungsgraph, die während der Repartitionierungsphase ([Sol91], [Heb95]) Verwendung finden.

In **Kapitel 5** werden die Ergebnisse der aktuellen Implementierung untersucht und u. a. mit denen der früheren Version (*planner.v5*) verglichen, die mit dem oben beschriebenen Verfahren der Quadropartitionierung ([PaZ90]) gute Resultate im Hinblick auf die Minimierung der Chipfläche erzielte. Dabei wurden die Messungen, die auch die Verzögerungszeiten mit in Betracht ziehen, hauptsächlich an drei Chipentwürfen durchgeführt, die aus dem Volumenvisualisierungsprojekt ([Lic98]) der AG Zimmermann stammen und deren Entwurf zuvor mit kommerziellen Tools geschah.

Diese drei Chipentwürfe verwenden die ES2_0.7-Technologie, können also dem Sub-Micron-Bereich zugeordnet werden. Um darüber hinaus in den Deep Sub-Micron-Bereich zu gelangen, wurde bei den Messungen mit verschiedenen Skalierungen der Technologieparameter experimentiert. Für eine kurze Zusammenfassung der Meßergebnisse sei auf Kapitel 6 verwiesen.

Die Verwendung von Benchmark-Daten, die für Arbeiten im Standardzellenbereich gängige Praxis ist, war leider nicht möglich, da für timing-driven Floorplanning (mit flexiblen Subzellen und Makrozellen) keine Benchmarks verfügbar sind. Da jedoch reale Chipentwürfe die Grundlage der Beispiele sind, ist die Aussagekraft der Ergebnisse (insbesondere in Bezug auf Zeitverhalten) mit Sicherheit höher als bei Verwendung von Pseudoentwürfen.

Ein Resümee zieht **Kapitel 6** und beschließt mit einem Ausblick auf interessante Fragestellungen und offene Forschungsthemen im weiteren Kontext dieser Arbeit.

Auf das **Literaturverzeichnis** folgt der **Index (mit Abkürzungsverzeichnis)**. Drei **Anhänge** diskutieren Aspekte zu der Umwandlung von Hypergraphen in Graphen, zu Unterschieden von topologischen und topographischen Nachbarschaften und zu maximalen Netzlängen für Verzögerungszeitvorgaben.

Ein paar Anmerkungen zu getroffenen **Konventionen** sollen diese Einleitung beschließen. Um die Lesbarkeit des Textes zu erhöhen, werden einige zentrale Begriffe als englische Wörter

wie deutsche behandelt, auch wenn sie (noch) nicht in die deutsche Sprache aufgenommen wurden. Dazu gehören u. a. *Software*, *Hardware*, *Floorplanning*, *Timing*, *Routing*, *Sizing* und *Slicing*. Englischer Begriff und deutsche Übersetzung werden häufig synonym verwendet, z. B. *Floorplanning* oder *Flächenplanung*, *Routing* oder *Verdrahtung*. Wenn die deutsche Übersetzung hingegen gebräuchlicher erscheint als das englische Original, wird sie durchgehend verwendet, z. B. *Plazierung* statt *Placement*, *Partitionierung* statt *Partitioning*, Ausnahmen bilden dann nur feste, rein englische Kombinationen wie *timing-driven Placement*.

Neu eingeführte Begriffe werden **fett** gekennzeichnet, weiterhin auch solche, die die Struktur des Textes verdeutlichen. Auf *Kursivschrift* wurde weitgehend verzichtet, Ausnahmen bilden Eigennamen (z. B. *planner.v6*) und Namen von Beispielen.

Definitionen, Formeln, Variablen und Werte dafür sind ebenso wie die Abbildungen grundsätzlich in Helvetica gesetzt. Auf eine Numerierung der Definitionen und Formeln wurde verzichtet, die interessanten werden über einen zugeordneten Namen identifiziert (z. B. *Elmore Delay-Modell*), der einprägsamer ist als eine Nummer.

Die Namen von Produkten, die in dieser Arbeit erwähnt werden, können urheberrechtlich geschützt sein und sind es in der Regel auch. Daß ein Name nicht als Trademark gekennzeichnet ist, berechtigt nicht zur Annahme, daß er kommerziell nutzbar sei.