Effiziente Implementierung von Datenpfaden auf FPGAs

Andreas Koch
Abteilung Entwurf integrierter Schaltungen (E.I.S.), TU Braunschweig
koch@eis.cs.tu-bs.de

SDI ist eine Strategie zur Erstellung von breiten regulären Datenpfaden mit fester Topologie auf FPGAs. Dabei kommen parametrisierbare Modulgeneratoren, ein Floorplanner auf Basis eines genetischen Algorithmus und Kompaktierung durch plazierungsorientiertes Technology-Mapping zum Einsatz. Erste Ergebnisse zeigen eine Verbesserung gegenüber den bisher verwendeten Verfahren.

Einleitung

Die Implementierung von effizienten Datenpfadstrukturen auf FPGAs wird durch existierende Werkzeuge nur unbefriedigend bearbeitet. Gründe dafür sind unter anderem die Optimierung auf minimale Fläche, die bisher wegen der immer noch vergleichsweise kleinen Chip-Größen bevorzugt durchgeführt wurde, und das Fehlen bzw. der Verlust von Regularitätsinformationen.

Um die Leistungsfähigkeit der an der Abteilung E.I.S. entwickelten SPARXIL-Architektur [Koc94] eines konfigurierbaren Coprozessors auf Xilinx-FPGA-Basis zu steigern, sollen Verfahren untersucht werden, die die Optimierung von Schaltungen mit 16 bis 32 Bit breiten Datenpfaden verfolgen. Im folgenden wird ein Überblick über den in Entwicklung befindlichen Ansatz "Structured Design Implementation" (SDI), der auf der Ausnutzung regulärer Strukturen in den Schaltungen und einem von der Plazierung beeinflußten Technology-Mapping basiert, gegeben.

Vergleich mit bisherigen Ansätzen

Bisherige, nicht FPGA-spezifische Verfahren, die reguläre Strukturen verarbeiten, fallen häufig in eine von zwei Kategorien: Auf der einen Seite finden sich Ansätze, die aus einer flachen oder hierarchischen Netzliste nach deren Erstellung Regularitäten extrahieren. Die so erkannten *Makros* werden dann bei der Plazierung gesondert behandelt. Solche Extraktionsverfahren werden beispielsweise

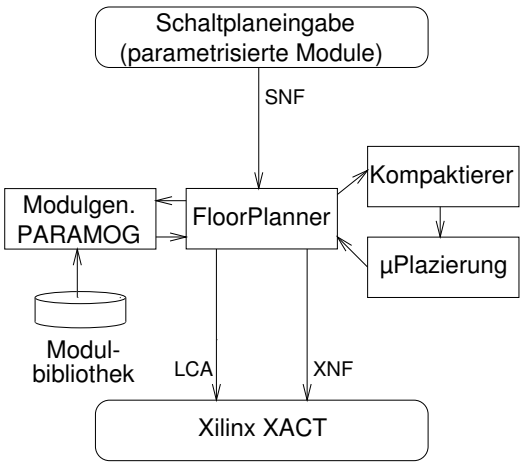

Abb. 1: Systemüberblick

in [Oda87], [Hir88] und [Yuw93] beschrieben. Nach der Strukturextraktion werden die einzelnen Makros dann durch Force-Directed- ([Oda87], [Chi93]) oder Simulated-Annealing-Verfahren ([Yuw93]) plaziert.

Der schon länger verwendete zweite Ansatz baut bereits bei der Schaltungserstellung auf der Verwendung von regulären *Modulen* auf, die dann geeignet plaziert werden. Stellvertretend für viele Verfahren seien hier [Cai90] und [Ben93] genannt. Dabei werden sowohl komplexe (Addierer, Registerbänke und Shifter) als auch primitive Module (AND, OR, INV, MUX, etc.) nach Vorgabe von Parametern wie Bitbreite und Anordnung von Pins automatisch generiert und anschließend linear plaziert. In [Cai90] geschieht dies beispielsweise durch ein A -Verfahren, das auf einem modifizierten Branch-and-Bound-Algorithmus aufbaut.

Auch SDI verwendet diesen zweite Ansatz: Sowohl die Schaltungserstellung als auch das Floorplanning finden auf Modulebene statt. Im Gegensatz zu den bisher angesprochenen Werkzeugen kann innerhalb von SDI aber die Modulstruktur aufgebrochen werden, um primitive nebeneinanderliegende Module ef-

fizient in FPGA-Logikblöcke abzubilden. Da SDI für den SPARXIL-Prozessor entwickelt wurde, wird die Abbildung derzeit auf Xilinx XC4000 CLBs vorgenommen. Die Strategie selbst könnte aber auch leicht auf andere FPGAs mit Matrix-Struktur (z.B. AT&T ORCA) angepaßt werden. Die Gesamtfunktion der verschmolzenen Module wird nach einem lokalen Technology-Mapping unter Beibehaltung der Datenpfadstruktur *mikroplaziert*. Dieses Vorgehen unterscheidet SDI von bisherigen plazierungsorientierten Technology-Mapping-Verfahren ([Mur91], [Cha93]), die Regularitäten unberücksichtigt lassen.

Der von SDI verwendete Modulgenerator PARAMOG ist nach klassischem Muster aufgebaut ([Shu89], [Ben93]). Anfragen können durch Angaben über Bitbreite, Datentypen und Platzbedarf parametrisiert werden, und PARAMOG liefert Vorschläge zur Realisierung der Funktion. Anders als das komplexere System LORTGEN [Bra94] hat PARAMOG keine interne Wissensbasis und nimmt keine Bewertung von Designalternativen vor. Im Rahmen von SDI werden diese Aufgaben begrenzt vom FLOORPLANNER übernommen. PARAMOG liefert aber im Gegensatz zu LORTGEN bereits plazierte und verdrahtete Module (*Hardmacros*), die FPGA-spezifische Eigenheiten wie die HardCarry-Logik der Xilinx XC4000-Bausteine ausnutzen.

SDI besteht aus mehr als einem Werkzeug. Die Strategie kombiniert einen Floorplanner, Modulgeneratoren, Werkzeuge zur Plazierung und globalen Verdrahtung mit Minimierungs- und Technology-Mapping-Verfahren. Sie kann daher nicht mit spezialisierten Einzelwerkzeugen verglichen werden, die nur einen Teilbereich abdecken (beispielsweise dem ASYL Synthese- und Mapping-Werkzeug [Bab92]). Diese können i.d.R. aber mit geringem Aufwand in SDI integriert werden.

Strukturierte Schaltplaneingabe

Schaltungen werden in SDI in Form von parametrisierten Modulen eingegeben. Dadurch können Informationen über die logische Struktur der zu implementierenden Schaltung von der Eingabe (beispielsweise auf Schaltplanebene) an die Plazierungs- und Verdrahtungswerkzeuge (P&R) weitergegeben werden. Die reguläre Struktur des Datenpfades kann dann sowohl zur Optimierung der Schaltung als auch der CAD-Algorithmen ausgenutzt werden.

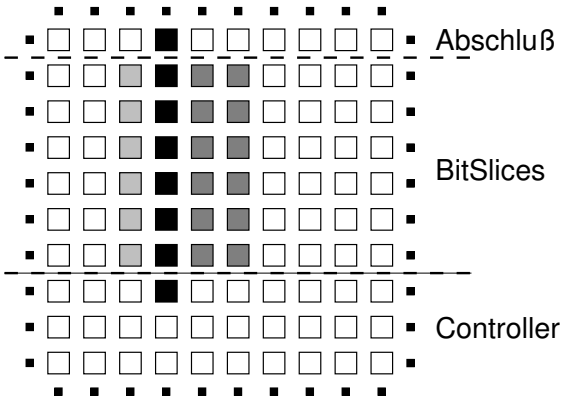

Abb. 2: Chiptopologie

Zieltopologie der Datenpfade

Die Zieltopologie von SDI basiert auf einer Dreiteilung der zur Verfügung stehenden Chipfläche (Abb.2). Der weitaus größte mittlere Teil ist dabei für den Aufbau der regulären Sektion des Datenpfades bestimmt. Dieser besteht aus einer horizontalen Anordnung von Modulen, die wiederum aus vertikal gestapelten Schichten zur Bearbeitung der einzelnen Bits bestehen. Unterhalb des Datenpfades befindet sich ein reservierter Bereich für den Controller, dessen irreguläre Logik nicht von SDI erfaßt wird. Zur Aufnahme von Unregelmäßigkeiten in den Modulen, die nicht für jedes Bit repliziert werden, steht oberhalb der Bitslices noch ein schmaler Bereich zur Verfügung. Nach Implementierung des Datenpfades kann die gesamte noch freie Fläche des Chips zur Implementierung des Controllers verwendet werden.

Die Verdrahtung wird nach dem klassischen Schema von horizontal verlaufenden Daten- und vertikalen Steuersignalen aufgebaut. Die Steuersignale können durch die Verwendung von Langverbindungen (vertical long lines) effizient an die Bitslices herangeführt werden.

Modulgenerierung durch PARAMOG

Die SDI-Modulbibliothek stellt neben einfachen logischen und arithmetischen Basisoperationen auch komplexere Funktionen wie RAM, ROM und Multiplizierer bereit. Die Modulgeneratoren erzeugen aus den vom

Anwender vorgegebenen Anforderungen dann Layout-Vorschläge für konkrete Realisierungen.

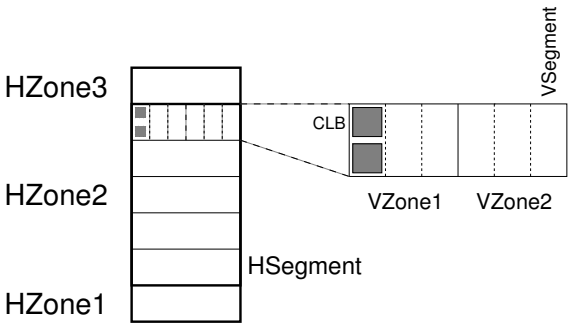

Abb. 3: Regulärer Aufbau von Modulen

Dabei sind die Module selbst auch auf regulären Strukturen aufgebaut. Ein Modul besteht aus Segmenten gleicher Logik und Verdrahtung, die horizontal und vertikal repliziert werden (Abb.3). Durch diese Gliederung können z.B. auch gefaltete U-förmige Module beschrieben werden.

Die Modulbibliothek ist technologiespezifisch, da Eigenheiten der Zielarchitektur ausgenutzt werden. Im Fall der bisher vorliegenden XC4000-Bibliothek werden beispielsweise die HardCarry-Logik der FPGAs und die Möglichkeit, Logikblöcke als RAMs und ROMs zu konfigurieren, ausgenutzt.

Die von PARAMOG erzeugten Module sind bereits fertig plaziert und intern verdrahtet. Die Lage von Pins kann aber während der Verdrahtung der Gesamtschaltung durch Vertauschen von logisch äquivalenten Pins eines CLBs begrenzt geändert werden.

Modulselektion und -plazierung

Der im Rahmen von SDI entwickelte FLOORPLANNER [Put95] ruft nach Einlesen der Netzliste von den beteiligten Modulgeneratoren ab, in was für Topologien sie die konkreten Instanziierungen der Module mit den angegebenen Parametern bereitstellen können.

Nun beginnt der FLOORPLANNER durch einen fuzzy-gesteuerten genetischen Algorithmus mit der Selektion von konkreten Modulimplementierungen bei gleichzeitiger linearer Plazierung. Da der FLOORPLANNER die Regularität der Schaltung ausnutzt, indem er nicht auf einzelnen Gattern, sondern kompletten Modulen arbeitet, ist das Problem besser handhabbar als die Arbeit auf einer Netzliste aus Basisgattern.

In die Bewertungsfunktion des FLOORPLANNERs fließt neben dem "Zusammenpassen" der Modultopologien (ein Datenpfad mit homogener Anzahl von CLBs/Bit ist besser als eine starke Variation) auch das Zeitverhalten von Leitungen und der Platzbedarf der Schaltung ein. Der Abschätzung des Zeitverhaltens liegen Timing-Modelle der verschiedenen auf dem FPGA vorhandenen Verdrahtungsresourcen zu Grunde.

Kompaktierung

Zur weiteren Optimierung werden durch den FLOORPLANNER nebeneinander plazierte primitive Module kompaktiert. Da die dabei verwendeten allgemeinen Verfahren in der Regel die Besonderheiten einer FPGA-Architektur nicht ausnutzen, werden nur Basisgatter und andere Logik, die nicht speziell optimiert wurde, der Kompaktierung unterzogen.

Dabei werden ihre Modulzugehörigkeiten aufgehoben und die Funktionen der Bitslices durch klassische Minimierungs- und Mapping-Methoden bearbeitet. Zu diesem Zweck können Standardwerkzeuge eingesetzt werden: In der ursprünglichen Version von SDI wurde beispielsweise SIS 1.3 [Sen92] verwendet. Wie bereits in der Einleitung beschrieben, können aber leicht andere Werkzeuge in die Strategie integriert werden. So bindet die aktuelle Version von SDI stattdessen das leistungsfähigere Verfahren FLOWMAP [Con94] in die Strategie ein.

Auch bei diesem rechenaufwendigen Vorgang leistet die Ausnutzung der Regularität gute Dienste, da schon berechnete Teilergebnisse repliziert werden können.

Als Ergebnis der Kompaktierung liegt für jeden gepackten Bereich eine Netzliste von N-LUTs (im Fall des XC4000 sind es N=4 Look-up-Tables) vor, die ggf. noch ein Flip-Flop vor dem Ausgang haben. Die LUT mit dem eventuell vorhandenen FF bildet für die folgende Verarbeitung eine Einheit. Im Fall der XC4000-FPGAs passen zwei solcher Einheiten in einen CLB, sie werden daher als *CLB/2* bezeichnet.

Mikroplazierung

Nach der Kompaktierung müssen die CLB/2 eines Bereiches jetzt neu plaziert werden. Dabei

ist es nicht sinnvoll, ausschließlich auf probabilistische Verfahren wie Simulated Annealing [Sec85] o.ä. aufzubauen, da diese i.d.R. keine Rücksicht auf die reguläre Struktur des Gesamtlayouts nehmen.

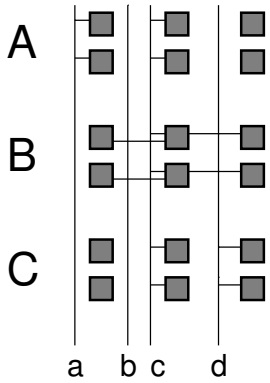

Abb. 4: Mikroplazierung der HZonen A, B, C mit Verdrahtung der Steuersignale a, b, c, d

Bei der Mikroplazierung gilt es, die CLB/2 innerhalb einer HZone (siehe Abb.4) so zu plazieren, daß neben einer möglichst gute Verdrahtung der Steuerleitungen auch noch die Gesamtverzögerung durch die HZone berücksichtigt wird. Dieses Problem schließt sowohl die Plazierung der CLB/2 als auch das globale Verdrahten der Steuerleitungen ein. Dabei können ggf. auch Steuerleitungen repliziert werden (in mehreren vertikalen Kanälen auftreten).

Pin-Assignment und Verdrahtung

Für diese Schritte ist zumindest vorerst geplant, die herstellereigenen Werkzeuge, im Fall von Xilinx das XACT-Paket, zu verwenden. Dabei werden die von SDI implementierten regulären Teile eines Entwurfes dem Werkzeug PPR als “guide design” eingegeben. PPR implementiert damit den gesamten Chip, bestehend aus regulärem Datenpfad und irregulärem Controller. Der reguläre Teil wird entsprechend den Vorgaben von SDI angelegt. Die Plazierung des Controllers erfolgt unter Ausnutzung der noch freien Chipfläche mittels Simulated Annealing. Alle noch nicht verlegten Leitungen werden durch den PPR Maze-Router mit Rip-up-and-retry Erweiterung verdrahtet. In Berücksichtigung der Kanalauslastung wird hier ggf. auch eine Änderung des Pin-Assignments vorgenommen.

Sollte dieser Prozeß erfolgreich abgeschlossen werden, steht als Endprodukt ein FPGA-Bitstream bereit, der reguläre und irreguläre Elemente in der angestrebten Weise vereint.

Beispiel

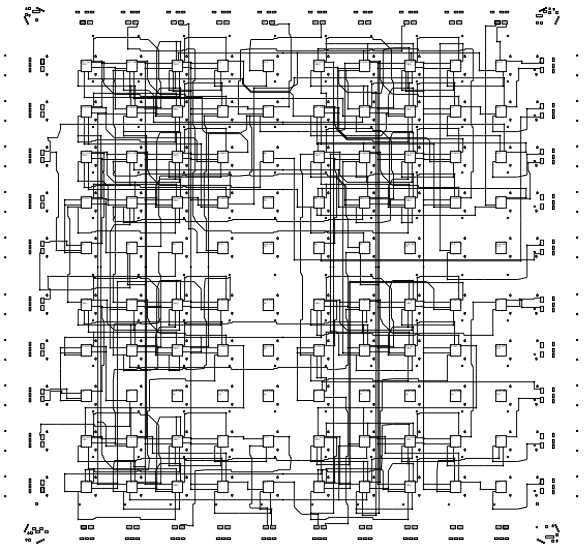

Abb. 5: PPR Plazierung und Verdrahtung

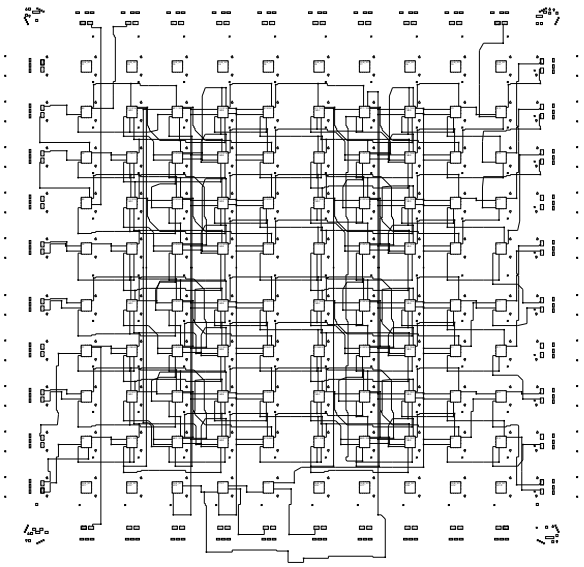

Abb. 6: SDI Plazierung mit PPR Verdrahtung

Die Abbildungen 5 und 6 vergleichen die jeweils von PPR und SDI erzeugten Layouts eines 16-Bit Datenpfades (zwei Instanzen

eines komplexen kombinatorischen Modules mit je vier HZonen, je vier Bit pro HZone, 16 4-LUTs pro HZone). Dabei wurde in beiden Fällen das technology-mapping ausgeschaltet, um den direkten Vergleich der Plazierungsverfahren zu ermöglichen. PPR wurde in seiner maximalen Optimierungsstufe (placer_effort = 5) performance-driven (dp2p) ohne Einschränkungen betrieben. Das abgedruckte Layout ist die beste Lösung, die PPR nach 15 Durchläufen fand. Das Routing beider Layouts durch PPR wurde ebenfalls mit maximaler Optimierung (router_effort = 4) vorgenommen.

Eine oberflächliche Betrachtung zeigt, daß die SDI-Plazierung deutlich regulärer ist als die PPR-Lösung, da die natürliche Struktur des Datenpfades ausgenutzt wurde. Weiterhin ist die SDI-Lösung weniger congested als die von PPR (insbesondere im ersten Quadranten). Auf dem kritischen Pfad ist die Verdrahtung der SDI-Lösung 13% schneller als die der maximal optimierten PPR-Lösung.

Projektstatus

Die Arbeit an den Modulgeneratoren und der Bibliothek ist abgeschlossen. Der FLOORPLANNER ist ebenfalls bereits implementiert, die Bewertungsfunktion bedarf jedoch noch der Abstimmung. Die Bereiche Kompaktierung und Mikroplazierung sind weitgehend abgeschlossen. Es gilt nun die Komponenten zu einem Gesamtsystem zu integrieren und ihr Zusammenspiel zu automatisieren. Als schwierig erweist sich das Benchmarking der Strategie, da es für Datenpfadschaltungen keinen etablierten Korpus von Testfällen ähnlich dem für Logiksynthese oder SC-Layout gibt.

Literatur

[Ben93] Ben Ammar, L., Greiner, A., "A High Density Datapath Compiler Mixing Random Logic with Optimized Blocks", *Proc. EDAC 1993*, pp. 194-198

[Bab92] Babba, B., Crastes, M., Saucier, G., "Input driven synthesis on PLDs and PGAs", *Proc. EDAC 1992*, pp. 48-52

[Bra94] Brand, H.J., Müller, D., Rosenstiel, W., "Specification and Synthesis of Complex Arithmetic Operators for FPGAs", in *Field Programmable Logic – Architectures, Synthesis and Applications*, ed. by Hartenstein R.W., Servits, M.Z., Springer 1994, pp. 78-88

[Cai90] Cai, H., Note, S., Six, P., DeMan, H., "A Data Path Layout Assembler for High-Performance DSP Circuits", *Proc. 27th DAC 1990*, pp. 306-311

[Cha93] Chau-Shen, C., Yu-Wen, T., "Combining Technology Mapping and Placement for Delay-Optimization in FPGA Designs", *Proc. ICCAD 1993*, pp. 123-127

[Chi93] Chih-Liang, E.C., Chin-Yen, H., "SEFOP: A Novel Approach to Data Path Module Placement", *Proc. ICCAD 1993*, pp. 178-181

[Con94] Cong, J., Ding, Y., "FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs", *IEEE Trans. on CAD*, Vol. 13, No. 1, January 1994, pp. 1-12

[Hir88] Hirsch, M., Siewiorek, D., "Automatically Extracting Structure from a Logical Design", *Proc. ICCAD 1988*, pp. 456-459

[Koc94] Koch, A., Golze, U., "A Universal Co-Processor for Workstations" in *More FPGAs*, ed. by Moore, W., Luk,W., Oxford 1994, pp. 317-328

[Mur91] Murgai, R., Shenoy, N., Brayton, R.K., Sangiovanni-Vincentelli, A., "Performance Directed Synthesis for Table Look Up Programmable Gate Arrays", *Proc. ICCAD 1991*, pp. 572-575

[Oda87] Odawara, G., Hiraide, T., Nishina, O., "Partitioning and Placement Technique for CMOS Gate Arrays", *IEEE Trans. on CAD*, Vol. CAD-6, No. 3, May 1987, pp. 355-363

[Put95] Putzer, H., "Ein fuzzy-gesteuerter Genetischer Algorithmus mit Anwendungsmöglichkeiten auf das Plazierungsproblem bei FPGA-Chips", *7. E.I.S. Workshop 1995*

[Sec85] Sechen, C., Sangiovanni-Vincentelli, A. "The TimberWolf placement and routing package", *IEEE J. Solid-State Circuits*, SC-20(2), pp. 510-522, 1985

[Sen92] Sentovich, E.M. et al., "SIS: A System for Sequential Circuit Synthesis", *Electr. Res. Lab. Memo No. UCB/ERL M92/41*, Dept. of EE and CS, UC Berkeley 4 May 1992

[Shu89] Shung, C.S. et al., "An Integrated CAD System for Algorithm-Specific IC Design", *Proc. Intl. Conf. on System Design 1989*, Hawaii

[Yuw93] Yu-Wen, T., Wu, A.C.H, Youn-Long, L., "A Cell Placement Procedure That Utilizes Circuit Structural Properties", *Proc. EDAC 1993*, pp. 189-193