Structured Design Implementation – Eine Implementierungsstrategie für Datenpfade auf FPGAs

Andreas Koch
Abt. 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 plazierungsorientiertes Technology-Mapping zum Einsatz.

Einleitung

Plazierungs- und Verdrahtungsverfahren für FPGAs haben die Regularität von Datenpfad-Schaltungen (RTL) bisher nur unvollständig ausgenutzt. Dies führt gerade bei FPGAs, bei denen Signalverzögerungen besonders durch die Verbindungen zwischen den Logikelementen verursacht werden, zu einem signifikanten Leistungsverlust. Auch könnten durch die Ausnutzung regulärer Strukturen für die Bearbeitung der einzelnen Komponenten leistungsfähigere Algorithmen zur Anwendung kommen, die bei der Bearbeitung einer flachen, unstrukturierten Netzliste an ihre Grenzen stoßen (NP-hart).

Um die Leistungsfähigkeit der an der Abt. 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

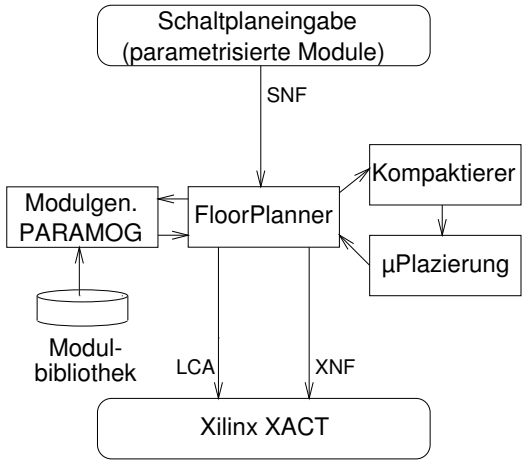

Abb. 1: Systemüberblick

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 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 automa-

tisch 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 effizient 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.

Strukturierte Schaltplaneingabe

Eine wesentliche Voraussetzung für SDI ist die Weitergabe von Informationen über die logische Struktur der zu implementierenden Schaltung von der Eingabe (beispielsweise auf Schaltplanebene) an die Plazierungs- und Verdrahtungswerkzeuge (P&R). SDI basiert auf der Spezifikation der Schaltung durch parametrisierte Module, die mittels entsprechender Generatoren erst in eine Layoutrepräsentation instanziiert werden. Durch den Verzicht auf ein "flattening" eines Entwurfes in Basisgatter kann die reguläre Struktur des Datenpfades auf allen Ebenen der Designimplementierung ausgenutzt werden.

Als Eingabe in SDI wird ein einfaches Netzlistenformat (SNF) verwendet. Durch einen SNF-Export aus der Designdatenbank des Cadence OPUS EDA-Systems können aber auch andere Formate wie EDIF, Verilog und VHDL (nur Strukturbeschreibungen) nach SNF übernommen werden.

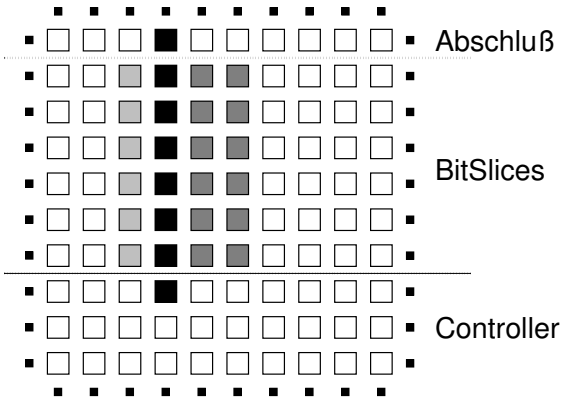

Abb. 2: Chiptopologie

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. Dabei ist die Anordnung von unten nach oben LSB nach MSB. 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. Dieser kann Logik aufnehmen, wie sie beispielsweise beim Auslesen von Carry- und Overflow-Signalen

aus einem sonst regulären Addierer verwendet wird. Irregularitäten am unteren Ende der Module, beispielsweise die Initialisierung einer CarryChain, können in den Controller-Bereich hineinragen. Nach Plazierung des Datenpfades steht die gesamte noch freie Fläche des Chips zur Implementierung des Controllers bereit.

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.

Diese Zieltopologie wurde von Anfang an beim Entwurf der SPARXIL-Architektur berücksichtigt, sie kann aber allgemein für den Aufbau breiter Datenpfade genutzt werden.

Modulgenerierung durch PARAMOG

Die PARAMOG-Modulbibliothek stellt neben einfachen logischen und arithmetischen Basisoperationen auch komplexere Funktionen wie RAM, ROM und Multiplizierer bereit. PARAMOG erzeugt aus den vom Anwender vorgegebenen Anforderungen (z.B. Bitbreiten der Eingänge, Optimierung Platz ./. Zeit etc.) dann Layout-Vorschläge für konkrete Realisierungen. So kann ein 16 Bit-Addierer beispielsweise als 8x1, 16x1 oder 4x2 CLBs angeboten werden.

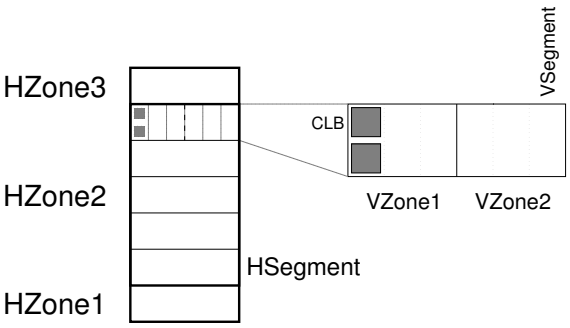

Abb. 3: Regulärer Aufbau von Modulen

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 befragt nach Einlesen der Netzliste PARAMOG, in was für Topologien die konkreten Instanziierungen der Module mit den angegebenen Parametern bereitgestellt werden können.

Nun beginnt der FLOORPLANNER durch einen fuzzy-gesteurten 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 derzeitigen Version von SDI wird beispielsweise SIS 1.3 [Sen92] verwendet. Soweit sie das EQN-Format beherrschen, können aber auch beliebige andere Werkzeuge integriert werden. Dadurch kann SDI direkt von aktuellen Ergebnissen auf den Gebieten der Logikminimierung und des Technology-Mappings profitieren.

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.

Bei der Mikroplazierung gilt es, die CLB/2 innerhalb einer HZone (siehe Abb.4) so zu plazieren, daß sich eine möglichst gute Verdrahtung der Steuerleitungen ergibt. Die Gesamtverzögerung durch die HZone ist dabei nur ein sekundäres Kriterium. 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).

Zur Lösung werden Verfahren der Integer-Programmierung verwendet. Abhängig davon, ob Bitslice-übergreifende Steuerleitungen existieren, kommen dabei zwei unterschiedliche Modelle zum Einsatz.

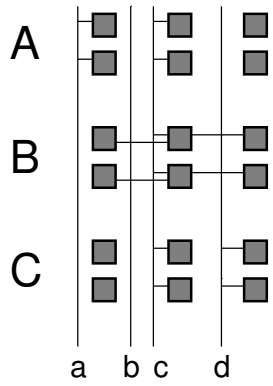

Abb. 4: Mikroplazierung der HZonen A, B und C mit Verdrahtung der Steuersignale a, b, c und d

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.

Projektstatus

Die Implementierung der Modulgeneratoren ist bereits abgeschlossen. Auch ein Großteil der Bibliothek ist erstellt. Der FLOORPLANNER ist ebenfalls bereits implementiert, die Bewertungsfunktion bedarf jedoch noch der Abstimmung. Im Bereich der Kompaktierung und Mikroplazierung existieren Algorithmen und mathematische Modelle, diese sind aber bisher nur außerhalb von SDI an abstrakten Beispielen getestet worden. Die konkrete Im-

plementierung dieser Ansätze und die Integration aller Komponenten zu einem Gesamtsystem sind derzeit die vordringlichsten Aufgaben.

Literatur

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

[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

[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

[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