Analyse von Synthese-Algorithmen für Pipeline-Strukturen

Andreas Wassatsch, Frank Grassert, Michael Grothmann, Dirk Timmermann

Universität Rostock
Fachbereich Elektrotechnik und Informationstechnik
Institut für Angewandte Mikroelektronik und Datentechnik
Richard-Wagner-Str. 31, 18119 Rostock, Germany
wassatsch@e-technik.uni-rostock.de

Der Einsatz dynamischer Schaltungstechnik ermöglicht die Realisierung geschwindigkeitskritischer Anwendungen im Bereich des Mikroprozessor-, DSP- und dedizierten ASSP-Designs. Die True-Single-Phase-Clock Logik (TSPC) ist dabei auf Grund ihrer Eigenschaften der geeigneteste Kandidat für auf Datendurchsatz optimierte IC-Designs. Mit der Verschmelzung der kombinatorischen Logik und des für eine Pipeline-Implementation notwendigen sequentiellen Elements zu einem Register mit eingebetteter kombinatorischer Logik ist durch die Verringerung der Setup-Zeit eine Steigerung der maximalen Taktfrequenz um 50% und durch die Reduktion der Zell- und Routingfläche eine Verkleinerung der Chip-Größe um 30% erreichbar.
Auf Grund der bisher noch fehlenden Unterstützung der dynamischen CMOS-Logik durch Standard-CAD-Software ist der Entwurfsprozeß solcher Hochgeschwindigkeitsschaltungen durch einen hohen manuellen Anteil geprägt. Mit der Integration zusätzlicher Syntheseschritte in den Design-Flow ist dieser Anteil jedoch eliminierbar. Das zentrale Element dieser Design-Flow Erweiterung wird dabei durch die Pipelinesynthese gebildet. Ausgehend von der Analyse bereits verfügbarer Implementationen, wurden mehrere neue Verfahren zur Pipeline-Synthese entwickelt. Neben der eigentlichen Pipeline-Generierung wurde dabei auch die Layout-orientierte Optimierung der Pipeline-Struktur berücksichtigt. Anhand von Design-Beispielen wurden die Eigenschaften der unterschiedlichen Verfahren untersucht und die Funktionsweise durch ein Equivalence-Checking zu einem Referenzmodell verifiziert.
In diesem Artikel werden wir weiterhin die Ergebnisse der Untersuchung zur generellen Eignung verschiedener Design-Architekturen für die Pipeline-Implementierung vorstellen.

1 Einführung

Für die Implementation hochfrequent getakteter Funktionsblöcke zur Verarbeitung von Datenströmen bietet sich der Einsatz dynamischer Schaltungstechnik an. Dabei erfolgt die Informationsverarbeitung meist in Pipeline-artigen Strukturen. Diese lassen sich auf Grund der in jeder dynamischen Zelle enthaltenen Register-Funktionalität sehr fein granular realisieren. Der Entwurf solcher Strukturen wird jedoch nur im geringen Umfang von kommerziell verfügbaren Synthese-Applikationen unterstützt und erfolgt daher hauptsächlich manuell [Sun97]. Als mögliche Lösung für dieses Design-Problem kann die Software-Umgebung DYNAMIC [WT00a] eingesetzt werden. Hauptbestandteil dieses Toolsets ist ein Micro-Pipeline-Reorganizer (MPR), der für die Generierung der Pipeline-Struktur verantwortlich ist. Nachdem in [WT00b] die strukturelle Eignung verschiedener Design-Grundelemente untersucht und bewertet wurden, sollen an dieser Stelle nun verschiedene Algorithmen für die Pipeline-Strukturierung eingehender betrachtet werden. Zur Erleichterung des Einstiges in dieses Materie sollen mit einer kurzen Einführung in die Grundlagen der Ziel-Technologie in Abschnitt 1.1 die charakteristischen Eigenschaften hervorgehoben werden. Dem schließt sich in Abschnitt 2 die Beschreibung der untersuchten Algorithmen zur Pipeline-Synthese an. In Abschnitt 3 werden die durchgeführten Untersuchungen der Algorithmen dokumentiert und im Anschluß die Ergebnisse bewertet und zusammengefaßt.

1.1 Schaltungstechnik

Bei der Realisierung eines Signalverarbeitungsalgorithmus durch eine Pipeline-Implementation wird auch die dafür notwendige Chip-Fläche zu einem Design-Kriterium. Bei einem Vergleich zwischen der Standard-CMOS-Logik und Vertretern aus dem Bereich der dynamischen Logik ist bezüglich der zur Realisierung der Funktionalität notwendigen Transistoren im Anwendungsgebiet Pipeline-Designs ein Vorteil der dynamischen Technik zu erkennen. So benötigt man für ein n-Eingangs-Gatter inklusive anschließendem Register in Standard-CMOS $2n+14$ Transistoren. Eine entsprechende Implementation in dynamischer Logik begnügt sich mit $n+11$ Transistoren.

1.1.1 Dynamische Schaltungstechnik

Ein typischer Vertreter aus der Reihe der dynamischen Schaltungstechniken ist die True-Phase-Single-Clock Logik (TSPC) [YKS87]. Wie aus Abb. 1 anhand der unterlegten Bereiche ersichtlich, erfolgt, im Gegensatz zur Standard-CMOS-Technik mit der dort angewandten komplementären Ausführung der Transistorbäume, die Implementation der Logikfunktion immer nur durch einen einfachen Transistorbaum. Prinzipiell entspricht der Aufbau einer TSPC-Zelle der Aneinander-Reihung eines dynamischen Elementes für die Realisierung der logischen Funktion und eines reduzierten Latches zur Speicherung des aktuellen Zustandes.

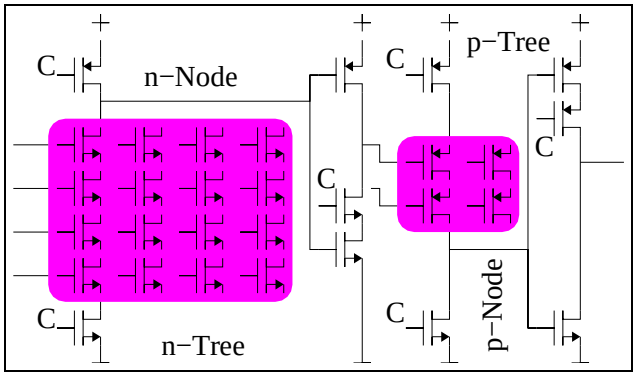

Abbildung 1: Aufbau eines TSPC-Gatters

Durch die Möglichkeit der Zusammenfassung der logischen Funktion mit einem durch zwei aufeinanderfolgende Latch-Strukturen gebildetes Register in einer Zelle kann ein sequentielles Element mit eingebetteter Logik implementiert werden, das vor allem für Pipeline-Strukturen auf Grund seiner geringen Größe und seiner hohen Takt-Geschwindigkeit benötigt wird. Die Funktionsweise einer TSPC-Zelle läßt sich, wie bei dynamischen Schaltungstechniken üblich, in zwei Arbeitsphasen unterteilen. In der in Abb. 2 jeweils rot gekennzeichneten Precharge-Phase wird eine interne Kapazität aufgeladen, welche meist nur durch die parasitären Anteile der Verdrahtung und der Gatekapazitäten implementiert ist. Während der anschließenden Evaluate-Phase, sie entspricht den grün schraffierten Bereichen, wird durch die im jeweiligen Transistor-Baum abgebildete Logik bestimmt, ob die so aufgebrachte Ladung wieder abgeleitet werden muß. Durch die alternierende Anordnung eines N- und eines P- bzw. N2-Blockes in der Schaltung kann bei der TSPC-Technik, trotz der Beschränkung auf nur ein globales Taktsignal, eine abwechselnde Tätigkeit benachbarter Teilblöcke erreicht werden.

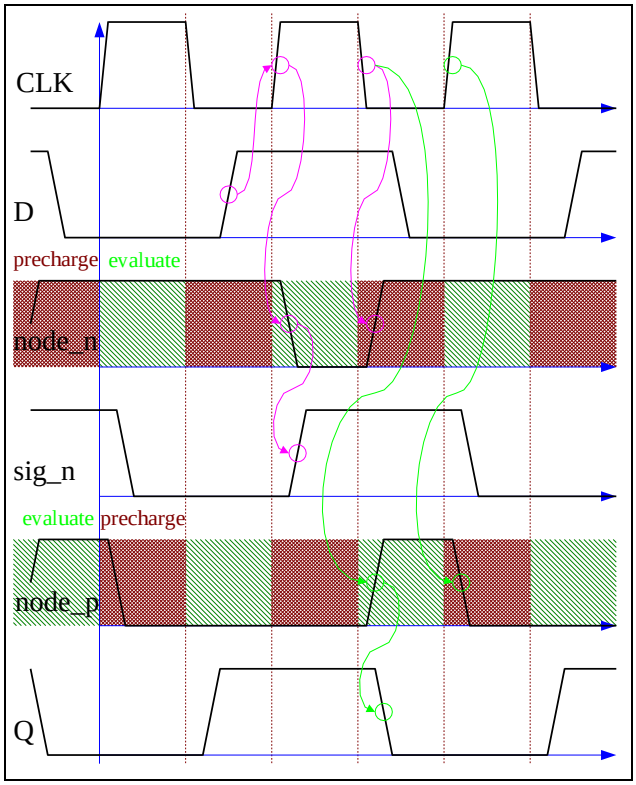

Abbildung 2: Funktion eines TSPC Flipflop

1.2 Design-Flow und Toolset

Durch die Erweiterung eines konventionellen Design-Flows um die in Abb. 3 gezeigten Elemente des DYNAMIC-Toolsets, wird die Integration dynamischer Schaltungstechnik in einen automatisierten Entwicklungsablauf ermöglicht.

Nach einer Abbildung des Designs auf eine dem Funktionsumfang der dynamischen Ziel-Bibliothek entsprechende statische CMOS-Logiksynthesebibliothek erfolgt der Aufbau und die Optimierung der Pipeline. Die verwendeten Algorithmen werden in Abschnitt 2 erläutert. Im Anschluß erfolgt die Abbildung der modifizierten Netzliste auf die Zielbibliothek. Nach der Generierung der Strukturen für die Taktverteilung innerhalb der Pipeline wird im normalen CMOS-Design-Flow fortgefahren.

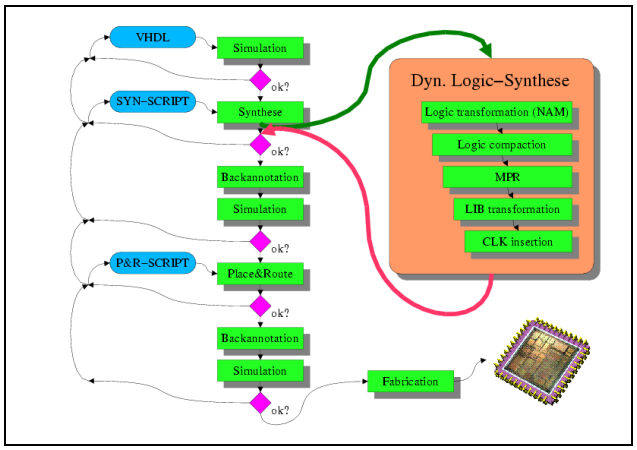

Abbildung 3: DYNAMIC Design-Flow

Der Design-Flow ist für Pipeline-bare Signalverarbeitungsverfahren mit nicht rekursiven Datenströmen, z.B. aus dem Bereich der Digitalen Filter oder der räumlich entwickelten iterativen Algorithmen wie dem CORDIC Verfahren, geeignet. Durch die Synthese-abhängige Anzahl der Pipeline-Stufen ist eine Adaptierbarkeit der Pipelineumgebung bezüglich der Latenzeit von Vorteil. Zur Vereinfachung einiger Pipelinesynthesealgorithmen ist die Zielbibliothek auf Zellen mit nur einem Ausgang beschränkt.

1.3 Verifikation

Da innerhalb des verwendeten Design-Flows einige Veränderungen bzw. Umformungen der Datenbasis erfolgen, ist nach einigen Teilschritten eine Verifikation des erzeugten Designs gegenüber den Referenzquellen notwendig. Der Vergleich des Verhaltens eines Testdesigns gegenüber seiner Referenz ist zum Beispiel bei geeigneter Stimulation innerhalb einer HDL-Simulation möglich. Ein allgemeinerer Ansatz ist jedoch die Verwendung von Methoden aus dem Gebiet der Formalen Verifikation [Mee00] [DMN87]. Insbesondere das Verfahren zum Equivalence-Checking erlaubt einerseits die designunabhängige Einbindung der Verifikation in den Design-Flow und andererseits die effiziente und umfassende Durchführung der Verifikationsschritte im Vergleich zu einer Stimuli-getriebenen Vergleich-Simulation. Wie in Abb. 4 dargestellt, erfolgt nach dem Ausgleich der Signalpfade ein Vergleich auf Äquivalenz zum Referenz-Designs der noch rein kombinatorischen Netzliste. Ein weitere Equivalence-Check wird nach der Library-Transformation, also der Umwandlung des kombinatorischen Designs in eine Verarbeitungspipeline, durchgeführt. Diese Netzliste sollte identisch zu dem idealen Ergebnis eines Pipeline-Retimings, angewendet auf die Kombination des Referenzdesigns und eines entsprechend tiefen FIFO-Registerblockes, sein.

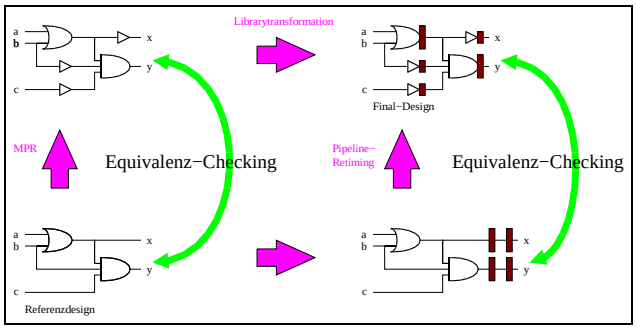

Abbildung 4: Verifikationsschritte

2 Algorithmen

Da die Anwendung des Pipeline-Retiming auf ein Design, bestehend aus einem kombinatorischen Block und einem FIFO-Registerblock, mit den bisher verfügbaren Werkzeugen nicht die gewünschten feingranularen Pipeline-Designs erzielt, war eine Suche nach alternativen Lösungsansätzen für das Design-Problem erforderlich. Als Startpunkt wurde dabei die Annahme gewählt, daß durch die vorangegangene Logik-Synthese unter Verwendung einer auf die rein kombinatorische Funktion abstrahierten Synthese-Bibliothek bereits eine optimale Abbildung des Designs erfolgt ist. Die Aufgabe des Algorithmus zur Pipeline-Erzeugung reduziert sich hierdurch auf die Ausbalancierung der Datenpfade im Design, um so den maximalen Durchsatz zu erreichen. Die Latenzzeit des resultierenden Designs bleibt durch die Reorganisation unbeeinflusst, was die Vorhersage des Timingverhaltens der resultierenden Schaltung ermöglicht. Sie ist bereits nach der Logik-Synthese durch die maximal auftretende Anzahl von Zellen in den Signalpfaden fixiert.

2.1 R2L-Algorithmus

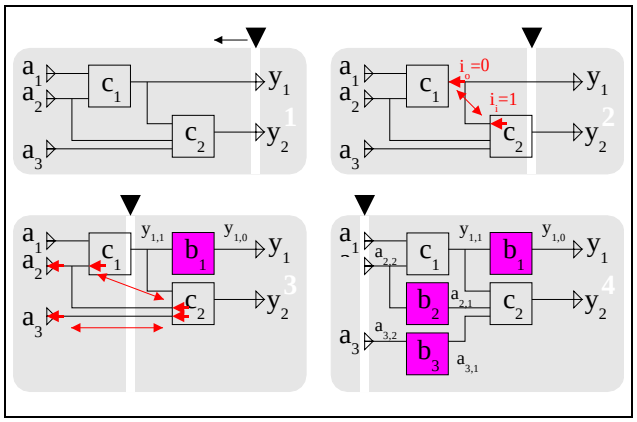

Abbildung 5: R2L Algorithmus

Dieses iterative Verfahren wird mit dem Satz der Ausgangssignale als aktuelle Parameter gestartet, um, wie in Abb. 5 gezeigt, eventuell vorhandene Signalbypässe zu eliminieren. Das wird durch das bedingte Einfügen zusätzlicher Puffer-Zellen erreicht. Im ersten Schritt der Iteration wird zuerst ein Indexwert für alle Signale des aktuellen Levels initialisiert. Anschließend wird für alle Zellen, die eines dieser indizierten Netze treiben, der Index der mit den Eingängen verbundenen Netze überprüft. Auf der Grundlage des Netzindexes eines Eingangspins kann dann die weitere Behandlung dieses Verbindungspunktes bestimmt werden. Ein nicht initialisierter Index kennzeichnet ein noch nicht bearbeitetes Netz und damit die Zugehörigkeit zum nachfolgenden Netzlevel. Ein Index mit dem gleichen Wert wie das zugehörige Netz am Zellausgang weist auf einen Netzbypaß in diesem Level hin. Diese Nebenleitung wird durch das Einfügen einer Puffer-Zelle, gekennzeichnet durch den Namen b in Abb. 5, eliminiert. Zur Begrenzung der Anzahl der eingefügten Puffer und damit der in die Pipeline eingebrachten zusätzlichen Register wird je Netzlevel und Netz jeweils nur eine Puffer-Zelle instantiiert. Wird ein Netzindex mit einem kleineren als dem Level des Ausgangsnetzes gefunden, zeigt dies eine nicht erlaubte Rückkoppelung innerhalb der zu erstellenden Pipeline an, welche durch den Entwickler manuell zu korrigieren ist. Netzlevel größer als der des betrachteten Ausgangspins sind durch das Verfahren bedingt, nicht möglich. Nach Abschluß der Berechnungen für den aktuellen Netzlevel wird solange mit dem nächsten Netzlevel fortgefahren, bis die Eingangspins der Netzliste erreicht sind. Der R2L-Algorithmus in der beschriebenen Form eignet sich nur für Netzlisten mit Single-Ausgangszellen.

2.2 L2R-Algorithmus

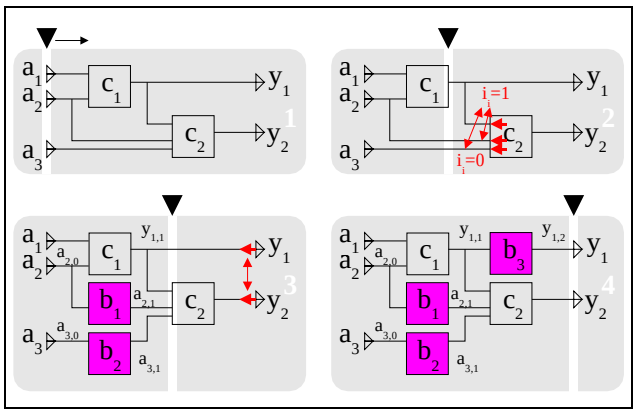

Abbildung 6: L2R Algorithmus

Im Gegensatz zum R2L Verfahren startet der L2R Algorithmus die Bearbeitung der Netzliste mit den Netzen der Eingangsports des Designs. Zu Beginn einer Iteration werden alle aktuellen Netze mit dem derzeitigen Netzlevel initialisiert. Anschließend wird in den Ausgangsnetzen der von dem aktuellen Netz gespeisten Zellen der nachfolgende Netzlevel propagiert. Durch eine Überprüfung der Netzlevel an den Eingangspins der Zellen, welche aus Elementen der List der aktuellen Netze gespeist werden, lassen sich nun Signalbypässe in dieser Pipeline-Stufe erkennen und beheben. Tritt bei wenigstens einem Eingangspin der nachfolgende Netzlevel auf, gehört diese Zelle nicht in die aktuelle Pipeline-Stufe. Ein nicht initialisierter Eingangspinlevel zeigt eine unerlaubte Rückkoppelung im Design an. Die Iteration wird bis zum vollständigen Erreichen der Ausgangsports der Pipeline weitergeführt. Der L2R-Algorithmus ist auch für Designs, die Zellen mit mehreren Ausgängen enthalten, geeignet.

2.3 Shift-by-Simulated-Annealing

Die Ergebnisse der beiden vorangegangenen Algorithmen zeigen eine direkte Abhängigkeit vom Ausgangsdesign. Die geometrische Form der Pipeline, d.h. insbesondere der Verlauf der Zellreihenbreite über die Pipeline-Stufen, wird durch die interne Struktur des Designs bestimmt. Es existiert jedoch bei der Transformation kein Freiheitsgrad, der eine Anpassung des Synthese-Ergebnisses an spezielle Anforderungen ermöglicht. So kann die Aufgabe der Optimierung der resultierenden Pipeline bezüglich seiner geometrischen Form nur in den vor-

hergehenden Syntheseschritt verlagert werden. Die adressierten Optimierungsziele sind jedoch auch bei der Logik-Synthese nicht erreichbar. Eine Annäherung an die gestellten Designanforderungen ist jedoch durch die Verschiebung beweglicher Logikzellen innerhalb der Pipeline-Struktur möglich.
Als wichtigstes Bewertungskriterium für die Auswahl der Translationskandidaten wählten wir mit Blick auf eine möglichst kleine, Rechteck-förmige Pipeline-Struktur die Anzahl der Zellen in einer Pipeline-Stufe. Ziel der Optimierung ist es, bei möglichst geringer Vergrößerung der Gesamt-Fläche nur eine geringe Abweichung der Pipelinenstufehöhe vom Durchschnitt zu erreichen. Auch eine Optimierung der Zellanordnung innerhalb der Pipeline-Stufen zur Minimierung der notwendigen Routingkanäle zwischen den die Pipeline-Stufen repräsentierenden Zellreihen ist möglich.

3 Analyse

Auf Grund der verwendeten TSPC Schaltungstechnik und des damit verbundenen Verhältnisses zwischen dem Implementationsaufwand für die TSPC-Grundzelle und dem eingebetteten Logikblock in Form des N-Kanal Transistorbaums kann zur Vereinfachung für die nachfolgenden Untersuchungen die Zellgröße aller TSPC-Bibliothekselemente einheitlich auf den abstrakten Wert 1 festgelegt werden. Ausgehend von einem gemeinsamen Referenzdesign wurden sowohl die Ergebnisse für den einzelnen Lauf des R2L- und L2R-Algorithmus ermittelt als auch die Werte bei nachfolgender Optimierung durch das eingesetzte Shift by Simulated Annealing Verfahren.

3.1 Testdesigns

Bei der Untersuchung des Verhaltens der Pipeline-Algorithmen wurden arithmetische Operatoren aus der vorhandenen Designware als Testbench eingesetzt. Diese finden auch bei der normalen Schaltungssynthese eines Signalverarbeitungsdesigns Verwendung und bieten eine größere Menge verschiebbarer Zellen als zum Beispiel Addiererstrukturen.

4 Ergebnisse

In Tabelle. 1 ist ein Auszug der ermittelten Resultate wiedergegeben. Die Werte in den mit A gekennzeichneten Spalten entsprechen den normierten Zellflächen. In den Spalten b ist die maximale Breite der Pipeline angegeben. Durch die Einführung der Pipeline ist eine Vergrößerung der benötigten Fläche zu beobachten. Dieses

n	ungepipelined	Pipeline-	L2R		L2R opt.				R2L		R2L opt.			
	A	stufen	A	b	A	%	b	%	A	b	A	%	b	%
2	12	5	23	6	20	87	4	67	24	6	20	83	4	67
4	130	17	259	28	241	93	21	75	283	24	241	85	21	87
6	346	31	909	64	767	84	35	55	957	41	768	80	35	85
8	646	43	2029	111	1510	74	50	45	1911	57	1477	77	45	79
16	2893	97	13526	359	10004	74	112	31	10304	128	7860	76	103	80

Tabelle 1: Optimierungsergebnisse dw-mul

entspricht in etwa dem Mehrbedarf an Chip-Fläche, der durch das Hinzufügen eines Pipeline-Register je kombinatorischer Zelle entsteht. Die Optimierung eines R2L- und eines L2R-Pipeline-Designs bringt mit geringen Abweichungen identische Resultate. Die Streuung wächst Verfahrens-bedingt mit zunehmendem Freiheitsgrad bzw. Bitweite. Der leichte Vorteil der R2L-Designs, der mit wachsender Bitbreite erkennbar wird, kann durch die Architektur des für die Logik-Synthese verwendeten Designware-Makro und der damit verbundenen Bevorzugung des R2L-Algorithmus erklärt werden. Die verhältnismäßig hohe Anzahl der erzeugten Pipeline-Stufen ist durch die im Umfang stark auf die logischen Grund-Funktionalitäten eingeschränkte Synthese-Bibliothek zu begründen. Diese Beschränkung war hilfreich bei der Verdeutlichung der erreichten Ergebnisse.

Die Gegenüberstellung der Pipeline-Strukturen in Abb. 7 zeigt den schon an Hand der Flächen- und Pipelinebreiten-Werte aus Tabelle. 1 erkennbaren Optimierungseffekt. Der R2L-Algorithmus erzeugt Pipeline-Strukturen, in denen die aus der kombinatorischen Logik entstandenen Logik-Register bezüglich ihrer Position in der Pipeline zu den Ausgängen hin streben. Auf Grund der Architektur der Ausgangsdesigns tritt dieser Effekt jedoch nicht so stark wie für den L2R-Algorithmus in Erscheinung. Beim L2R-Algorithmus ist hingegen eine erhöhte Logikzellendichte im Bereich der Eingänge zu beobachten.

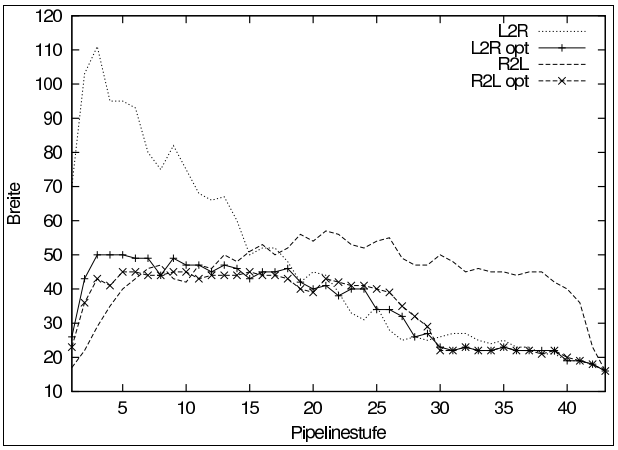

Abbildung 7: Pipelinebreitenverlauf für mul8

Die deutliche Verjüngung der Struktur bei den optimierten Designs im letzten Drittel der Pipeline läßt weiterhin den dort angeordneten Carry-Propagate Addierer des Carry-Save-Multiplizierer Designware-Makros erkennen. Durch die zusätzliche Regel in unserer Optimierungsstrategie wurde zugunsten einer kleineren Gesamt-Fläche die Generierung einer rechteckigen Pipeline-Struktur erfolgreich verhindert.

5 Zusammenfassung

Der Einsatz dynamischer Schaltungstechnik für die Implementation feingranularer Signalverarbeitungspipelines in einem automatisierten Design-Flow ist durch die Verwendung der untersuchten Algorithmen möglich. Die Architektur-bedingte Ausprägung der äußeren Pipeline-Form kann durch einen nachfolgenden Optimierungsschritt bei einer Vielzahl von Designs den bestehenden Integrationsanforderungen angepaßt werden. Zugleich wurde eine Reduzierung der Chip-Fläche auf Grund der Optimierung erreicht. Durch die optimierte Zellen-Anzahl für die Pipeline-Realisierung wird im gleichen Maße auch eine Verringerung der Verlustleistung möglich. Mittels Equivalence-Checking wurde die fehlerfreie Design-Umsetzung und Optimierung nachgewiesen.

Literatur

[DMN87] DEVADAS, S., H.-K.T. MA und A.R. NEWTON: *On Verification of Sequential Machines at Differing Levels of Abstraction*. In: *Proceedings of the 24th ACM/IEEE Design Automation Conference*, Seiten 271–276, 1987.

[Mee00] MEETH, S.: *Formal Methods in Functional Verification of Circuits*. In: *International Cadence User Group Conference (ICU'2000)*, Seite SIG IC, San Jose (CA) USA, September 2000.

[Sun97] SUNDSBØ, I.: *Analysis and VLSI design of synthesis filter bank for image subband coding*. Doktorarbeit, Norwegian University of Science and Technology, Dept. of Physical Electronics, 1997.

[WT00a] WASSATSCH, A. und D. TIMMERMANN: *DYNAMIC - A Java Based Toolset For Integrating Dynamic Logic Circuits Into A Standard VLSI Design FLow*. In: *International Cadence User Group Conference (ICU'2000)*, Seiten SIG IC – ic6, San Jose (CA) USA, September 2000.

[WT00b] WASSATSCH, A. und D. TIMMERMANN: *Untersuchung zum Einfluß der speziellen Anforderungen dynamischer Schaltungstechnik auf den Systementwurf*. In: *ITG/GI/GMM Workshop: Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen*, Seiten 278–287, Frankfurt/Main(Germany), 28.2.-1.3. 2000.

[YKS87] YUAN, J., I. KARLSSON und C. SVENSSON: *A true single phase clock dynamic CMOS circuit technique*. IEEE Journal of Solid State Circuits, SC-22:899–901, 1987.