Datenflussbasierte Modellierung, Analyse und Synthese leistungsfähiger Hardware-Software-Systeme zur Verarbeitung von Bildströmen

Joachim Keinert

Fraunhofer-Institut für Integrierte Schaltungen IIS
D-91058 Erlangen,
joachim.keinert@iis.fraunhofer.de

Abstract: Hohe Datendurchsätze eingebetteter Systeme zur Bildverarbeitung erschweren eine automatisierte Implementierung. Daher wird im Folgenden eine neue Entwurfsmethodik aufgezeigt, welche die Aspekte Modellierung, Analyse und Synthese abdeckt. Im Gegensatz zu zahlreichen existierenden Ansätzen kommt dabei das Konzept des mehrdimensionalen Datenflusses zum Einsatz, welches durch eine neue Modellierungstechnik speziell auf die Bildverarbeitung angepasst wird. Dadurch ergeben sich vielfältige Vorteile. Diese umfassen die Möglichkeit, mehr- und eindimensionale, statische und dynamische Algorithmenteile miteinander zu verbinden. Eigens entwickelte Analysekonzepte zur Berechnung des Speicherbedarfs erschließen neue Entwurfsalternativen bezüglich benötigter Hardwareressourcen und gestatten die akkurate Modellierung von Datenumsortierungen. Neue Hardwarekommunikationsprimitive ermöglichen einen hohen Datendurchsatz, verschiedene Implementierungsvarianten und bessere Effizienz im Vergleich zu eindimensionalen Beschreibungstechniken.

1 Einleitung

Anwendungen der Bildverarbeitung zeichnen sich durch hohe Anforderungen an den erreichbaren Datendurchsatz aus. Zusammen mit steigender Komplexität geraten daher die Kosten für den Systementwurf zur größten Gefahr für technische Entwicklungen. Daher müssen laut [Des09] dringend neue Entwurfsmethodiken gefunden werden, welche den Entwickler besser unterstützen. Dies umfasst eine Modellbildung auf hoher Abstraktionsebene, sowie eine automatische Analyse und Synthese auf massiv parallelen Architekturen. Deshalb wird im Folgenden eine neue Entwurfsmethodik vorgestellt, welche besonders Bildverarbeitungsanwendungen mit ihren hohen Geschwindigkeitsanforderungen adressiert.

Problemstellung Echtzeitfähige Bildverarbeitungsanwendungen werden häufig in Form von *sogenannten Blockdiagrammen* spezifiziert, da eine solche Darstellung einerseits die komplexe Gesamtanwendung in mehrere einfachere Module zerlegt — auch *Aktoren* oder *Prozesse* genannt —, und andererseits die inhärente Parallelität sichtbar macht. Abbildung 1a zeigt ein entsprechendes Beispiel in Form eines JPEG2000 Kodierers. Dieser

überführt eine Sequenz unkomprimierter Bilder in eine komprimierte Darstellung. Der erste Schritt dazu besteht aus einer sukzessiven Anwendung mehrerer *Wavelettransformationen (WT)*. Diese gehören zur Klasse der *lokalen Algorithmen*, welche das Eingangsbild mit einem Fenster abtasten und für jede Fensterposition eine konstante Anzahl an Ausgabebildpunkten erzeugen. In Abbildung 1b wurde beispielsweise eine Fenstergröße von 3×3 Pixel angenommen bei einer Eingangsbildgröße von 3×6 Pixel. Dabei soll sich das Fenster in horizontaler und vertikaler Richtung um zwei Pixel weiterbewegen, sodass sich die einzelnen Fenster überlappen. Außerdem sind lokale Algorithmen oft so gehalten, dass das Fenster an den Rändern über das eigentliche Bild hinausragt. Für die überstehenden Fensterpositionen müssen dementsprechend Pixelwerte angenommen oder berechnet werden. Diese sind in Abbildung 1b einem *erweiterten Rand* zugeordnet.

Für die Wavelettransformation entstehen pro Fensterposition vier Pixel, die jeweils einem separaten Teilbild zugeordnet werden. Die Berechnung wird typischerweise in eine horizontale und eine vertikale Filterung unterteilt, wobei die Fenstergröße je nach verwendetem Waveletfilter variiert. Die Teilbilder werden schließlich mittels sogenannter *Blockgeneratoren (BG)* in Codeblöcke zerlegt, die unabhängig voneinander einer *Entropiekodierung (EC)* unterzogen werden.

Sowohl die Wavelettransformation wie auch die Blockgenerierung erfordert die Bereitstellung eines Pufferspeichers, dessen Größe mit der Bildbreite wächst. Aus diesem Grund gestattet der JPEG2000 Standard, große Eingangsbilder mittels eines *Bildzerlegers (BZ)* in unabhängige Teilbilder zu unterteilen. Abbildung 1c stellt diese Operation exemplarisch für ein Eingangsbild der Größe 6×6 dar, welches in zwei Teilbilder der Größe 3×6 zerlegt wird. Die Bildquelle generiert das Eingangsbild dabei Zeile für Zeile, wie mittels entsprechender Zahlen gekennzeichnet ist. Die Lesereihenfolge dagegen weicht von der Schreibreihenfolge dahingehend ab, dass zuerst das linke Teilbild durch den Bildzerleger ausgegeben wird, bevor das rechte Teilbild an der Reihe ist. Anders ausgedrückt findet eine Umsortierung der Pixel statt. Die beiden Teilbilder werden dann unabhängig voneinander mittels der Waveletzerlegung transformiert wie in Abbildung 1b angedeutet.

Im Folgenden wird nun eine Entwurfsmethodik vorgestellt, welche es erlaubt, Anwendungen wie die JPEG2000 Kompression auf hohem Abstraktionsniveau zu beschreiben. Automatische Systemanalysen unterstützen den Entwickler bei Optimierungsfragen, ohne dabei jedoch datenabhängigen Kontrollfluss gänzlich auszuschließen. Außerdem werden Techniken zur automatischen Hardwaresynthese vorgestellt, um potentielle Fehlerquellen zu eliminieren und gleichzeitig hohe Geschwindigkeitsanforderungen erfüllen zu können.

Stand der Technik Um die Komplexität des Entwurfs eingebetteter Systeme zu reduzieren, sind bereits zahlreiche Werkzeuge und Methoden entworfen worden. Dabei setzen sich *aktororiente* Ansätze immer mehr durch, bei denen das Gesamtsystem in kleinere Prozesse zerlegt wird, welche durch Kanäle miteinander kommunizieren (siehe auch Abbildung 1a). Entsprechende Ansätze finden sich beispielsweise in *SystemC* oder *Simulink*. Aufgrund deren Allgemeinheit sind solche Spezifikationen allerdings extrem komplex zu analysieren. Dadurch sind automatische Systemanalysen erschwert und oft auf simulative Techniken beschränkt.

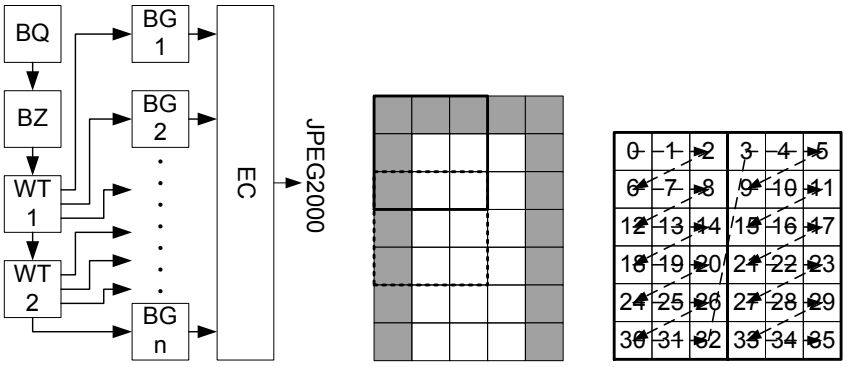


Abbildung 1: **(a)** Datenflussgraph zur JPEG2000 Kodierung, bestehend aus einer Bildquelle (BQ), einem Bildzerleger (BZ), Wavelettransformationen (WT), Blockgeneratoren (BG), und dem Entropiekodierer (EC). **(b)** Prinzip lokaler Bildverarbeitungsalgorithmen mit überlappenden Fenstern. Weiße Pixel repräsentieren das Eingangsbild, graue Rechtecke den erweiterten Rand. **(c)** Zerlegung des Eingangsbildes in zwei unabhängige Teilbilder. Nummern repräsentieren die Pixelproduktionsreihenfolge seitens der BQ, die Lesesequenz durch den BZ ist mittels Pfeilen angedeutet.

Um diese Problematik zu lösen, existieren wie in Abbildung 2 dargestellt zwei grundsätzlich unterschiedliche Ansätze. Auf der einen Seite gibt es eine große Anzahl *eindimensionaler Datenflussmodelle*. Diese transportieren einen eindimensionalen Strom von *Datenelementen* zwischen den Aktoren. Dazu werden meist FIFOs verwendet, was eine parallele (Hardware-) Implementierung stark vereinfacht. In statischen Modellen wie *synchronem Datenfluss (SDF)* wird die Anzahl der konsumierten und produzierten Datenelemente zum Zeitpunkt der Kompilierung festgelegt. Auf der anderen Seite gestatten dynamische Modelle [Buc93] datenabhängige Entscheidungen zur Laufzeit, welche in *Zustandsmaschinen* festgehalten werden können [KSS+09]. Allerdings bedeutet die Einschränkung auf eindimensionale Datenströme, dass lokale Algorithmen mit überlappenden Fenstern oder Datenumsortierungen wie oben diskutiert, nur sehr umständlich beschrieben werden können [KHT06]. Dadurch sind viele wichtige Merkmale, wie beispielsweise Datenabhängigkeiten oder paralleler Datenzugriff nicht richtig sichtbar. Infolgedessen ist die automatische Generierung möglichst leistungsstarker Hardware stark erschwert.

Auf der anderen Seite existieren wie in Abbildung 2 verdeutlicht zahlreiche Ansätze, welche kommunizierende Schleifen betrachten [BBS09, NTS+08, Fea06]. Der Datentransport geschieht mittels mehrdimensionaler Arrays, welche in dedizierte Speicher abgebildet werden. Damit können Bildverarbeitungsalgorithmen einfach und präzise dargestellt werden. Um jedoch solche Anwendungen in parallele Systeme zu übersetzen, sind relativ komplizierte Analysen notwendig, so dass meist nur statische Algorithmen oder stark eingeschränkte datenabhängige Entscheidungen unterstützt werden. Auf der anderen Seite erlaubt dies aber auch eine sogenannte *polyhedrale Analyse [BKV+08]*, welche selbst bei großen Bildabmessungen effiziente Analysen gestattet. Außerdem können die einzelnen

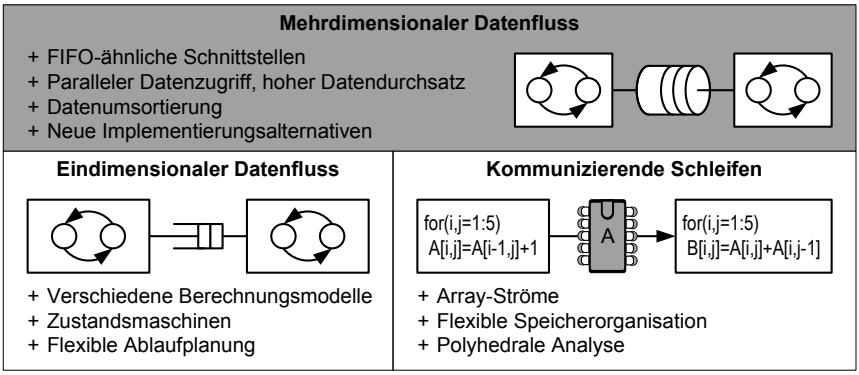


Abbildung 2: Vorteile zweier unterschiedlicher Entwurfskonzepte und ihre Vereinigung mittels mehrdimensionalem Datenfluss. Der Schwerpunkt der vorliegenden Dissertation und daraus resultierende Vorteile sind grau hinterlegt.

Datenelemente in unterschiedlicher Form im Speicher abgelegt werden, um eine möglichst effiziente Implementierung zu erhalten.

Beiträge dieser Dissertation Wie oben ausgeführt, bieten sowohl Datenfluss als auch kommunizierende Schleifen Vorteile für die Systemmodellierung, Analyse und Synthese. Dies gilt ganz besonders, da Anwendungen wie der in Abbildung 1a gezeigte JPEG2000 Kodierer sowohl Bilder in Form mehrdimensionaler Arrays als auch eindimensionale Datenströme verarbeiten. Allerdings gibt es nur sehr wenige Arbeiten, welche sich mit der Zusammenführung dieser beiden Ansätze befassen. Und diese beschränken sich im Wesentlichen darauf, kommunizierende Schleifen in eindimensionale Prozessgraphen zu übersetzen [BBS09, NTS+08]. Dies löst jedoch nicht die Schwierigkeiten, mehrdimensionale datenabhängige Algorithmen beschreiben zu können. Außerdem erfordert die Spezifikation kompletter Systeme immer noch die Verwendung zweier vollständig unterschiedlicher Beschreibungstechniken.

Infolgedessen wurde in dieser Dissertation ein Konzept erarbeitet, wie die beiden Ansätze mittels *mehrdimensionalem Datenfluss* vereint werden können. Dazu wurde ein entsprechendes mehrdimensionales Berechnungsmodell in Form des *Windowed Data Flow* erarbeitet [KHT06, KFHT07]. Dieses beschreibt die Abtastung mehrdimensionaler Arrays mittels möglicherweise überlappender Fenster einschließlich Randbehandlung und Datenumsortierung. Die Erarbeitung eines sogenannten *mehrdimensionalen FIFOs* gestattet, ein- und mehrdimensionale Algorithmen direkt zu koppeln und in ein und derselben Entwurfsumgebung mit ähnlichen Konzepten zu beschreiben [KFHT07]. Dadurch kann das Berechnungsmodell direkt in Systementwurfswerkzeuge wie dem *SystemCoDesigner* [KSS+09] integriert werden, welche bereits mit eindimensionalen Datenflussmodellen umgehen können. Dank Kompatibilität mit polyhedralen Analysetechniken können Anwendungen, welche auf sehr großen Bildern arbeiten, effizient analysiert werden, um beispielsweise den benötigen Pufferspeicher automatisch zu berechnen [KDH+09]. Ins-

besondere konnten durch die Kombination von Datenfluss und mehrdimensionalen Arrays Konzepte zur Analyse von Datenumsortierungen bereitgestellt werden, um Ablaufplanungen mit optimiertem Durchsatz zu ermitteln und neue Entwurfsalternativen aufzudecken. Und schließlich erlauben neue Methoden der Kommunikationssynthese die Übersetzung von mehrdimensionalen FIFOs in Hardwaremodule [KHT08], welche Bilder in Kinoauflösung in Echtzeit verarbeiten können und dabei verschiedene Abstufungen zwischen benötigten Hardwareressourcen und erreichbarem Durchsatz anbieten. Verglichen mit Ergebnissen, welche durch klassische Verhaltenssynthese eindimensionaler Datenflussmodelle erhalten wurden, bietet das mehrdimensionale FIFO einen höheren Datendurchsatz bei gleichzeitig kleinerem Ressourcenbedarf.

Trotz dieser Vorteile sind mehrdimensionale Datenflussmodelle noch erstaunlich wenig verbreitet, bzw. weißen noch erhebliche Einschränkungen auf. So gestattet *IMEM* [LOT07] nur die Modellierung einfacher lokaler Filter ohne Unterabtastung oder Datenumsortierung. Und *Gaspard2* [BMD07] bietet weder eine Unterstützung der Randbehandlung, noch eine enge Kopplung mit eindimensionalen Datenflussmodellen. Außerdem erreichen die Ergebnisse der Kommunikationssynthese nicht die Effizienz manueller Lösungen.

Daher möchte diese Dissertation zu neuen Lösungen im Bereich des mehrdimensionalen Datenflusses beitragen. Die vorliegende Zusammenfassung adressiert hierzu die Aspekte Applikationsmodellierung (Kapitel 2), Kommunikationssynthese (Kapitel 3) sowie die Speicheranalyse (Kapitel 4).

2 Applikationsmodellierung

Zur Beschreibung von Anwendungen mit mehr- und eindimensionalen Algorithmen wurde in dieser Dissertation ein mehrdimensionales Datenflussmodell mit dem Namen *Windowed Data Flow* [KHT06, KFHT07] entworfen. Dieses lässt sich wie in Abbildung 3 skizziert nahtlos mit bestehenden eindimensionalen Berechnungsmodellen kombinieren. Dazu wird die Anwendung in *Prozesse* oder *Aktoren* zerlegt, welche sowohl eindimensionale wie auch mehrdimensionale *Anschlusspunkte* besitzen können. *Eindimensionale Anschlusspunkte* werden mit eindimensionalen FIFOs verbunden, *mehrdimensionale Anschlusspunkte* mit neuartigen *mehrdimensionalen FIFOs*. Diese beschreiben die Abtastung von Arrays mittels möglicherweise überlappender Fenster durch die Senke, und die inkrementelle Produktion durch die Quelle. D.h., die Senke muss nicht warten, bis ein komplettes Array produziert ist, sondern kann mit dem Lesen beginnen, sobald alle Datenelemente für die erste Fensterposition verfügbar sind. Die Reihenfolge, mit welcher die verschiedenen Schreib- und Lesefensterpositionen abgearbeitet werden, kann durch entsprechende Parameter bestimmt werden, so dass Datenumsortierungen intuitiv beschreibbar sind.

Um Datenabhängigkeiten ausdrücken zu können, besitzt jeder Aktor eine Zustandsmaschine. Deren Zustandsübergänge sind mit Bedingungen annotiert, welche für eine Aktivierung erfüllt sein müssen. So bedeutet $i_1(1)$ in Abbildung 3 beispielsweise, dass am eindimensionalen Anschlusspunkt ein Datenelement verfügbar sein muss. $i_1[0] == 3$ prüft, ob dessen Wert drei beträgt. Im Falle mehrdimensionaler FIFOs fordert $i_2(1)$ entsprechend

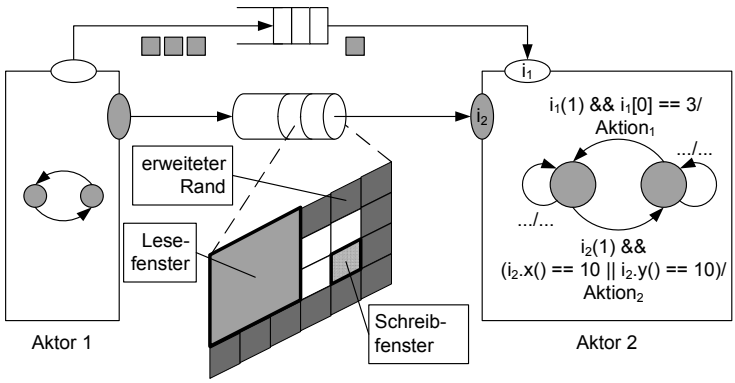


Abbildung 3: Prinzip der Applikationsmodellierung

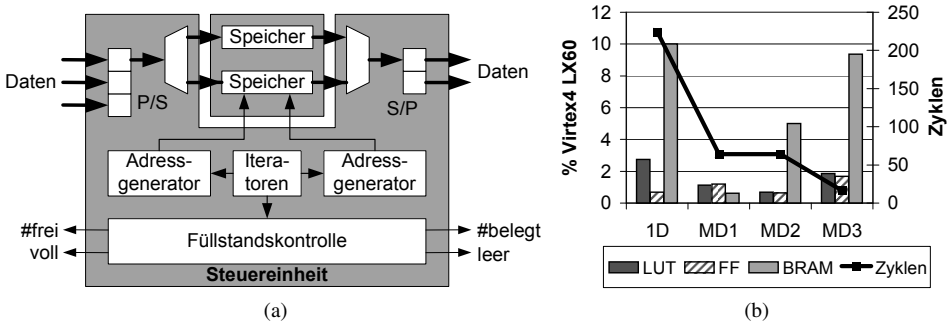
die Verfügbarkeit des kompletten nächsten Lesefensters, und $i_2.x()$ bzw. $i_2.y()$ repräsentiert die aktuelle Fensterkoordinate innerhalb des Arrays.

Sind alle Bedingungen eines Zustandsübergangs erfüllt, kann die dazugehörige *Aktion* ausgeführt werden. Auf diese Weise können datenabhängige Algorithmen beschrieben werden, welche pro Aufruf eine unterschiedliche Anzahl an Datenelementen lesen oder schreiben. Dadurch lässt sich Abbildung 1a beispielsweise dahingehend modifizieren, dass die beiden Wavelettransformationen *WT1* und *WT2* zu einem Aktor zusammengefasst werden und sich dadurch die benötigten Multiplizierer teilen können. Des Weiteren konnte gezeigt werden, dass sich auch Algorithmen mit lifting-basierter Speicherallokation oder nicht rechteckförmigen Fenstern beschreiben lassen.

3 Hardwaresynthese Mehrdimensionaler Kommunikationsprimitive

Im vorherigen Kapitel wurde eine neue Methode vorgestellt, um Anwendungen der Bildverarbeitung auf hohem Abstraktionsniveau zu modellieren. Solch ein Systemmodell kann seine Vorteile allerdings nur dann voll ausspielen, wenn es automatisch in Hardware- und Softwarelösungen für höchste Durchsatzanforderungen umgesetzt werden kann. Dazu gibt es in der Literatur eine große Anzahl an Vorschlägen, welche sich beispielsweise mit der optimierten Verhaltenssynthese von Aktoren beschäftigen [KSS+09], oder Strategien zur Vermeidung von Engpässen bei der Speicheranbindung entwickeln. Dabei werden aber notwendige Datenumsortierungen wie in Abbildung 1c außer Acht gelassen.

Aus diesem Grund wurde in dieser Dissertation eine Technik entwickelt, um mehrdimensionale FIFOs automatisch in eine Hardwarearchitektur wie in Abbildung 4a dargestellt zu übersetzen. Besonderes Augenmerk lag dabei auf der Datenumsortierung, wobei pro Takt mehr als ein Datenelement gelesen und geschrieben werden kann. Solche Szenarien treten beispielsweise bei einer diskreten Kosinustransformation auf, bei der immer acht Pixel auf einmal gelesen und geschrieben werden müssen.



(a) (b)

Abbildung 4: **(a)** Architektur der generierten Kommunikationsprimitive. **(b)** Syntheseergebnisse für das zeilenweise Lesen von 8×8 Blöcken und der spaltenweise Ausgabe. Das Ergebnis *1D* wurde durch traditionelle Verhaltenssynthese eines eindimensionalen Datenflussmodells ermittelt. *MD1-3* definieren unterschiedliche Syntheseparameter für ein und dasselbe mehrdimensionale FIFO. Die Zyklenangabe beschreibt die Verarbeitungsdauer eines 8×8 Blockes.

Die erzeugte Hardware besteht aus zwei separaten Einheiten, nämlich einem Speichersystem und einer Steuereinheit. Ersteres besteht aus mehreren Speichermodulen, um mehr als ein Datenelement pro Takt lesen und schreiben zu können. Spezielle Abbildungstechniken stellen sicher, dass möglichst wenig Speichermodule benötigt werden, wobei aktuell die Existenz separater Lese- und Schreibschnittstellen angenommen wird. Dies ist typischerweise bei chipinternem Speicher sowie bei QDR Speicher der Fall.

Die Steuereinheit wiederum teilt sich in verschiedene Komponenten auf, welche alle automatisch generiert werden. Iteratoren dienen der Mitverfolgung, welche Array-Positionen aktuell gelesen und geschrieben werden sollen. Anhand dieser Information können Adressgeneratoren die benötigten Lese- und Schreibadressen berechnen. Die Füllstandskontrolle schließlich bestimmt, wie viele Datenelemente noch von der Quelle geschrieben werden können, ohne dabei Daten im Speicher zu zerstören, welche später noch benötigt werden. Außerdem wird ermittelt, wie viele Fenster zum Lesen bereitstehen, ohne dass es weiterer Schreibvorgänge seitens der Quelle Bedarf. Dadurch kann trotz der Fähigkeit Daten umzusortieren eine FIFO-ähnliche Schnittstelle bereitgestellt werden, welche sich zum einfachen Datenaustausch zwischen Modulen eignet. Dies erhöht insbesondere die Möglichkeit der Modulwiederverwendung, da Module mit unterschiedlicher Lese- und Schreibreihenfolge einfach gekoppelt werden können.

Die Seriell-Parallel (S/P) und Parallel-Seriell (P/S) Konverter schließlich dienen dazu, verschiedene Entwurfsalternativen bereitstellen zu können. Je mehr Datenelemente pro Takt geschrieben und gelesen werden müssen, desto teurer wird die resultierende Hardwareimplementierung. Folglich können je nach benötigten Durchsatzanforderungen unterschiedliche Modulvarianten entworfen werden, wie in Abbildung 4b dargestellt. Diese zeigt die Syntheseergebnisse für das zeilenweise Lesen eines 8×8 Blocks und dessen spaltenweise Ausgabe. Eine solche Operation wird beispielsweise für die Implementierung einer zwei-

dimensionalen inversen diskreten Kosinustransformation benötigt. Wie aus Abbildung 4b ersichtlich, gestattet die entwickelte Methodik, aus ein und derselben Kommunikationsspezifikation verschiedene Hardwareimplementierungen zu erzeugen, welche sich im Ressourcenbedarf und unterstütztem Durchsatz unterscheiden. Im Vergleich zu einer Lösung, welche durch klassische Verhaltenssynthese eines eindimensionalen Aktors generiert wurde, offeriert das vorgeschlagene mehrdimensionale FIFO einen höheren Durchsatz bei gleichzeitig kleinerem Ressourcenbedarf. Kombiniert mit Taktfrequenzen von mehr als 300 MHz auf Xilinx Virtex4 FPGAs erlauben diese Fähigkeiten, Bildverarbeitungsalgorithmen mit sehr hohen Durchsatzanforderungen zu realisieren.

4 Automatische Speicherbedarfsanalyse

Wie in Kapitel 3 dargestellt, lassen sich mehrdimensionale FIFOs automatisch in Hardwareimplementierungen für höchste Durchsatzanforderungen übersetzen. Dazu ist es allerdings notwendig, dass bereits im Systemmodell der Applikation die entsprechenden FIFO-Größen korrekt spezifiziert wurden. Obwohl dies grundsätzlich manuell geschehen kann, stellt dies einen erheblichen Zeitaufwand dar, weil der benötigte Speicherbedarf nicht nur von Parametern wie der Fenstergröße abhängt, sondern auch von Durchsatzanforderungen sowie der Topologie des Datenflussgraphs.

Um den Entwickler bei dieser Aufgabe zu unterstützen, wurden deshalb in dieser Dissertation unterschiedliche Verfahren zur Speicheranalyse entwickelt und verglichen. Das erste basiert auf einer SystemC-Simulation, welche beliebige Ablaufplanungen unterstützt. Außerdem lassen sich zwei verschiedene Adressgenerierungsstrategien vergleichen, welche sich in unterschiedlichen Speichergrößen niederschlagen.

Während dieser Ansatz die größtmögliche Flexibilität bietet, führt dies im Zusammenhang mit großformatigen Bildern zu sehr langen Analysezeiten. Gerade bei statischen Anwendungsteilen, welche anhand besonderer Eigenschaften der Zustandsmaschine identifiziert werden können, bieten sich daher deutlich effizientere analytische Verfahren an. Deshalb wurde in dieser Dissertation gezeigt, wie ein mehrdimensionaler Datenflussgraph automatisch in ein polyhedrales Modell überführt werden kann. Dazu wird jeder Aktoraufruf in einem Punktgitter dargestellt, so dass die Ablaufplanung und Speicheranalyse auf das Verschieben von Punktgittern und auf Punktenumeration zurückgeführt werden kann. Eine neue Beschreibungstechnik für Datenumsortierungen wie in Abbildung 1c dargestellt führt dabei zu effizienteren Ablaufplänen, bei denen im Vergleich zu in der Literatur beschriebenen Verfahren eine bessere Parallelität bei der Ausführung der beteiligten Aktoren erreicht werden kann. Außerdem wurden Algorithmen vorgestellt, welche Datenflussgraphen mit mehreren Quellen und Rückkopplungsschleifen unterstützen. Anhand eines mehrstufigen Filters konnte dann gezeigt werden, dass solch ein Ansatz im Vergleich zur Simulation ein um 42% besseres Ergebnis liefert, während die Analyse nur 1% der Zeit bedarf. Des Weiteren konnte eine neue Technik entwickelt werden, welche durch unterschiedliche Ablaufplanung von Aktoren zur Unter- und Überabtastung Implementierungsalternativen aufzeigt, die zwar den gleichen Durchsatz aufweisen, sich aber trotzdem in den benötigten Ressourcen unterscheiden. So kann ein mehrstufiger Rauschfilter beispielsweise in zwei

Varianten implementiert werden, wobei die eine im Vergleich zur anderen zwar 28,6% mehr Speicher benötigt, dafür aber 16,1% weniger Multiplizierer. Dadurch kann die Anwendung automatisch optimal auf eine Zielarchitektur angepasst werden, in Abhängigkeit davon, welche Ressource gerade in ungenügender Anzahl vorhanden ist.

5 Zusammenfassung und Ausblick

In dieser Dissertation wurde eine neue Methodik zum Entwurf von Systemen entwickelt, welche Bildströme verarbeiten. Untersuchte Beispiele umfassen die morphologische Rekonstruktion, die Kodierung mittels JPEG und JPEG2000 einschließlich lifting-basierter Waveletstufe, sowie eine mehrstufige Rauschunterdrückung. Dabei wurden die Vorteile zweier unterschiedlicher Konzepte in Form von Datenfluss und von polyhedraler Analyse kombiniert. Insbesondere lassen sich dadurch Anwendungen beschreiben, welche statische und dynamische, eindimensionale und mehrdimensionale Algorithmen beinhalten. Ein neuartiges mehrdimensionales FIFO erlaubt die Integration der Konzepte in bestehende Entwurfswerkzeuge wie beispielsweise *SystemCoDesigner* [KSS+09]. Außerdem können flexible Hardwarekommunikationsprimitive automatisch synthetisiert werden, welche sehr hohen Durchsatzanforderungen gerecht werden, und welche unterschiedliche Implementierungsalternativen unterstützen. Die polyhedrale Analyse kann vorteilhaft zur effizient Ablaufplanungen und zur Bestimmung notwendiger Speichergrößen eingesetzt werden. Dabei können unterschiedliche Implementierungsalternativen untersucht werden. Gerade der letzte Aspekt wurde in der traditionellen polyhedralen Analyse bisher nicht berücksichtigt.

Zusammenfassend kann also gesagt werden, dass mehrdimensionale Datenflussmodelle trotz ihrer noch relativ geringen Verbreitung eine vielversprechende Technologie darstellen und ein großes Potential für weitere Entwicklungen und Forschung bieten. Dazu gehört insbesondere die Softwaresynthese von mehrdimensionalen Datenflussgraphen. Ebenso vielversprechend sind weitere Verbesserungen in der Hardwaresynthese zur Unterstützung zusätzlicher Speichertypen und Optimierungen für datenabhängige Entscheidungen.

Literatur

[BBS09] T. Bijlsma, M.J.G. Bekooij und G.J.M. Smit. Inter-task communication via overlapping read and write windows for deadlock-free execution of cyclic task graphs. In *Proceedings of SAMOS*, Seiten 140–148, July 2009.

[BKV+08] F. Balasa, P. G. Kjeldsberg, A. Vandecappelle, M. Palkovic, Q. Hu, H. Zhu und F. Catthoor. Storage Estimation and Design Space Exploration Methodologies for the Memory Management of Signal Processing Applications. *J. Signal Process. Syst.*, 53(1-2):51–71, 2008.

[BMD07] Sébastien Le Beux, Philippe Marquet und Jean-Luc Dekeyser. A Design Flow to Map Parallel Applications onto FPGAs. In *Proceedings of FPL*, Seiten 605–608, 2007.

[Buc93] Joseph Tobin Buck. *Scheduling dynamic dataflow graphs with bounded memory using the token flow model*. Dissertation, University of California at Berkeley, 1993.

[Des09] Design International Technology Working Group (ITWG). International Technology Roadmap for Semiconductors — Design. Bericht, ITRS, 2009.

[Fea06] Paul Feautrier. Scalable and structured scheduling. *Int. J. Parallel Program.*, 34(5):459–487, 2006.

[KDH+09] Joachim Keinert, Hritam Dutta, Frank Hannig, Christian Haubelt und Jürgen Teich. Model-Based Synthesis and Optimization of Static Multi-Rate Image Processing Algorithms. In *Proceedings of DATE*, Seiten 135–140, 2009.

[KFHT07] Joachim Keinert, Joachim Falk, Christian Haubelt und Jürgen Teich. Actor-Oriented Modeling and Simulation of Sliding Window Image Processing Algorithms. In *Proceedings of ESTIMEDIA*, Seiten 113–118, 2007.

[KHT06] Joachim Keinert, Christian Haubelt und Jürgen Teich. Modeling and Analysis of Windowed Synchronous Algorithms. *ICASSP2006*, III:892–895, 2006.

[KHT08] Joachim Keinert, Christian Haubelt und Jürgen Teich. Automatic Synthesis of Design Alternatives for Fast Stream-Based Out-of-Order Communication. In *Proceedings of VLSI-SoC*, Seiten 265–270, Rhodes Island, Greece, October 2008.

[KSS+09] Joachim Keinert, Martin Streubühr, Thomas Schlichter, Joachim Falk, Jens Gladigau, Christian Haubelt, Jürgen Teich und Michael Meredith. SystemCoDesigner—an automatic ESL synthesis approach by design space exploration and behavioral synthesis for streaming applications. *ACM Trans. Des. Autom. Electron. Syst.*, 14(1):1–23, 2009.

[LOT07] Najeem Lawal, Mattias O'Nils und Benny Thörnberg. C++ based System Synthesis of Real-Time Video Processing Systems targeting FPGA Implementation. In *IPDPS*, Seiten 1–7, 2007.

[NTS+08] H. Nikolov, M. Thompson, T. Stefanov, A. Pimentel, S. Polstra, R. Bose, C. Zissulescu und E. Deprettere. Daedalus: toward composable multimedia MP-SoC design. In *Proceedings of DAC*, Seiten 574–579, 2008.

Joachim Keinert studierte Elektro- und Informationstechnik an der Universität Stuttgart sowie an der École Nationale Supérieure des Télécommunications (Télécom ParisTech) in Paris. Das Studium schloss er im Jahr 2004 mit dem Erwerb des französischen und deutschen Diploms (mit Auszeichnung) ab. Im Anschluss bearbeitete er am Fraunhofer Institut für Integrierte Schaltungen verschiedene Entwicklungs- und Forschungsprojekte in der Bildverarbeitung für digitale Kinoanwendungen. Dies umfasste auch die Mitarbeit in ISO/IEC Standardisierungsgremien zum Thema JPEG2000. Parallel dazu forschte Joachim Keinert an der Universität Erlangen-Nürnberg unter Leitung von Prof. Dr.-Ing. Jürgen Teich an neuen Methodiken zum automatisierten Entwurf von hochleistungsfähigen Bildverarbeitungsanwendungen. Eine entsprechende Dissertation schloss er im Oktober 2009 mit Auszeichnung (summa cum laude) ab. Die aktuellen Forschungsinteressen von Joachim Keinert umfassen die effiziente Kompression von Bildern auf Mehrprozessorarchitekturen, sowie dazu notwendige Entwurfsmethodiken.