Kapitel 1

Einleitung

Diese Arbeit befasst sich mit der Berechnung von Implikationen in boolsenen Netzwerken mit programmierbarer Logik - sogenannten Field Programmable Gate Arrays oder kurz FPGAs.

Es wird ein Verfahren vorgestellt, das mit dynamisch programmierbaren FPGAs einen Spezialrechner implementiert, der die Implikationsberechnung wesentlich schneller durchführt kann als ein konventioneller Mikroprozessor.

Das Besondere an diesem Verfahren ist, dass die dynamische Konfigurierbarkeit der Bausteine ausgenutzt wird, um die Schaltung des Spezialrechners an die jeweilige Eingabe anzupassen. Das heißt, es wird für jedes zu untersuchende boolsenen Netzwerk eine neue FPGA-Konfiguration generiert. Dieses Vorgehen wird als Rekonfigurierende Computing genannt und mit RC abgekürzt.


Diese Arbeit gliedert sich in die folgenden Kapitel:

1. Einleitung
2. Rekonfigurierende Computing
3. Implikationsberechnung durch Logisches Schließen
   In diesem Kapitel wird das verwendete Algorithmus zur Berechnung von indirekten Implikationen in boolsenen Netzen und seine Anwendungen vorgestellt. Grundlage ist das Buch von Professoren Kunz. [4]
4. Übertragung des Algorithmus auf FPGAs
   Dynamische Rekonfigurierung, Parallelisierung des Algorithmus, Details des Steuerungsaufmaßes, Mogliche Erweiterung durch Rekonfiguration während der Laufzeit.
5. Implementierung
   Hier wird detailliert die Schaltung des Implikationsbeschleunigers vor-
gestellt. Es wird auch auf die verwendeten Entwurfsmethoden und Software eingegangen.

6. Auswertung

7. Fazit
Bewertung der Ergebnisse im Hinblick auf die Fragen: ist das Verfahren praktikabel? Was kann die RC Forschung daraus lernen? Welche Anforderungen an FPGA Architekturen ergeben sich aus dieser Arbeit?

GLOSSAR
RC, Reconfigurable Computing
FPGA
Implikationen
Index: Fehlersimulation, DNS Sequenzen, Reconfigurable Computing, dynamisches Rekonfiguration, Implikationsberechnung, Simulation, ISCAS, FPGA Architektur.
Kapitel 2
Rekonfigurierbare Architekturen


Mikroprozessor Im ersten Fall handelt es sich um Standardhardware die im allgemeinen ohne Bezug zu dem zu implementierenden Algorithmus entwickelt wurde. Durch eine Folge von Befehlen die sequentiell abgearbeitet werden lassen sich beliebige Algorithmen implementieren. Es wird eine relativ kleine, zur Chipertigung festgelegte Hardware mit wenigen Standardoperatoren sehr oft und sehr schnell umkonfiguriert.

Spezialhardware Im zweiten Fall wird der Algorithmus durch eine Schaltung realisiert die speziell für diese Anwendung entwickelt wurde. Die Schaltung kann nur für eine eng begrenzte Klasse von Algorithmen verwendet werden. Die Konfigurierbarkeit beschränkt sich auf wenige Parameter die nur selten geändert werden. Die Realisierung kann als

Anwendungsspezifischer Schaltkreis erfolgen oder als FPGA mit nicht dynamisch veränderbarer Konfiguration.

Die zweite Variante ist in der Regel deutlich Leistungsfähiger, dafür aber unflexibler und teurer sofern nicht sehr hohe Stückzahlen erreicht werden. In erster Näherung kann man sagen, dass Spezialhardware meistens nur dann verwendet wird, wenn die Leistungsfähigkeit von Mikroprozessoren nicht ausreicht. ¹


Mikroprozessoren und Spezialhardware stellen in gewisser Hinsicht zwei Extreme dar. Auf der einen Seite der universelle Rechenapparat dessen Funktion mit jedem Takt neu bestimmt wird, auf der anderen Seite die starke Speziallösung die eventuell unbrauchbar wird und sich die Anforderungen ändern.

Zwischen diesen beiden Extremen - Programmierbare Hardware und Hardware mit starker Funktionalität - gibt es einen großen Raum von architekturellen Alternativen die sich bis jetzt kaum in der Praxis durchsetzen.

konnten. Besonders interessant sind in diesem Bereich die Rekonfigurierbaren Architekturen.

Rekonfigurierbare Architekturen stellen eine Alternative für die extensive Verwendung von Spezialhardware in Standardrechnern dar. Sie ermöglichen es eine komplette Palette spezialisierter VLSI-Lösungen die alle gleichzeitig zur Verfügung stehen zu emulieren durch eine kleine Menge Hardware die jeweils nach Bedarf die gerade geforderte Funktionalität implementiert.

2.1 Charakterisierung rekonfigurierbarer Architekturen

Im allgemeinen nutzen rekonfigurierbare Architekturen einen Großteil ihrer Chipfläche für ein im Vergleich zu Mikroprozessoren große Zahl an Ausführungseinheiten, die über ein konfigurierbares Verbindungsnetzwerk miteinander verbunden sind. Die Operation jedes Rechenelementes kann ebenso programmiert werden wie die Verbindungen. Verschiedene Teile eines Algorithmus werden über die Chipfläche verteilt und Zwischenergebnisse können direkt von Element zu Element fließen ohne in einem zentralen Speicher abgelegt zu werden. Im einfachsten Fall wird der gesamte Algorithmus in einen Datenflußgraphen überführt und auf die rekonfigurierbare Zielarchitektur abgebildet. Rekonfigurierbare Architekturen tendieren dazu, Aufgaben räumlich zu verteilen, im Gegensatz zu programmierbaren Architekturen - wie Mikroprozessoren - die die Aufgaben zeitlich verteilen.

Die Hauptunterschiede zwischen rekonfigurierbaren Architekturen und konventionellen Prozessoren sind:

Befehlsverteilung Die Konfigurationen werden nicht Takt für Takt aus einem zentralen Befehlsstrom an die Ausführungseinheiten verteilt, sondern werden lokal gespeichert. Dadurch können wesentlich größere und komplexere Konfigurationen verwendet werden. Da die externe Bandbreite natürlich weiterhin begrenzt bleibt, folgt daraus, daß Rekonfigurierbare Architekturen nur selten neue Konfigurationen aus externen Speicher lä-

den. Intern können aber mehrere Konfigurationen gespeichert sein. Siehe z.B.[14]

Datenfluß für Zwischenergebnisse Zwischenergebnisse werden parallel von der Quelle zur Senke geleitet anstatt sämtliche Kommunikationen über einen zentralen Ressource durchzuführen, wie zum Beispiel eine Registerbank.

Mehr Ausführungseinheiten, feinere Granularität Es gibt viele Ausführungseinheiten die unabhängig voneinander konfiguriert werden können. Dadurch können mehr und flexiblere Operationen ausgeführt werden als bei einem konventionellen Prozessor.


Viele dieser Eigenschaften wurden erst durch die Verfügbarkeit sehr hoch integrierter Schaltkreise ermöglicht. Dies erklärt, wie hohe diese Architekturen erst seit relativ kurzer Zeit systematisch erforscht werden.

Die oben angegebene unformale Charakterisierung deckt einen recht großen Entwurfsspaum ab. Deshalb unterscheiden sich die veröffentlichten Ansätze sehr stark:


Prozessoren mit konfigurierbaren Ausführungseinheiten Ein konventioneller Mikroprozessor bei dem die Funktionalität der Proze-
sorbefehle über FPGA ähnliche Strukturen frei konfiguriert werden kann. Beispiele: CARP[15].


Zelluläre Automaten Dies ist das älteste Beispiel einer konfigurierbaren Architektur. Sie wurden erstmals von von Neumann 1966 beschrieben und ähneln schon sehr stark den FPGAs.[18].

FPGA basierende Architekturen Mit FPGAs lassen sich weitgehend beliebige Schaltungen realisieren, sie lassen sich also auch als Rekonfigurierbare Rechner nutzen. Ihr Nachteil ist der hohe Flächenbedarf für die sehr detaillierte Konfiguration, sie sind jedoch die flexibelste aller hier vorgestellten Architekturen und sind als als Standardprodukte kommerziell verfügbar.

2.2 FPGA basierende Rechner

Die meisten Forschungs- projekte zu rekonfigurierbaren Architekturen beschaftigen sich mit FPGAs. Die relativ große Zahl der FPGA basierenden Projekte erklärt sich hauptsächlich daraus, dass FPGAs die einzige rekonfigurierbaren Architekturen sind, die als Standardbaustein im Handel erhältlich sind. Einige RA Gruppen beschäftigen sich aber auch gezielt mit FPGAs, da sie die meisten Freiheitsgrade und damit den größten Raum für Experimente. Zum Beispiel lassen sich alle anderen Architekturen - im allgemeinen und Flächen und Geschwindigkeitsbemäßen - auf FPGAs abgebildet werden. Weiterhin sind FPGAs mit ihrer extrem feinen Granularität gerade in den Anwendungen gut, in denen die heutigen grob granularen Hochleistungsprozessoren besonders schlecht abschneiden.

2.2.1 Erfolge FPGA basierender RA

André DeHon führt in [2] Beispiele für die Rechenleistung der bekanntesten FPGA Rechner auf:


- Multiplikation großer Zahlen 16x schneller als Cray-II
- Mit 600 kbit/s, 512-Bit RSA Dekodierung. Schnelle Implementierung zum Zeitpunkt der Veröffentlichung
- Suche in Zeichenketten mit ähnlicher Leistung wie eine Speziallösung aus 28 VLSI Schaltkreisen.
- Diskrete Cosinustransformation mit 15 GOPS.

SPLASH SPLASH von SRC besteht aus 32 XC3090. SPLASH führt laut [20] DNS Sequenzierung 400 mal schneller aus als ein CRAY-II

Fehlersimulation Der Rechenzeitaufwand und die gute Parallelisierbarkeit einiger Fehlersimulationen hat Fehlersimulation in einer weiteren kommerziellen Anwendung für FPGA basierende Rechner werden lassen.

Es hat sich gezeigt dass FPGAs überall dort gut sind wo auf kleinen Datenwörtern operiert wird, wie zum Beispiel beim DNS-sequenzieren, Fehlersimulation und Logikemulation und bei Anwendungen die sich nicht gut auf die Operationen eines Standardprozessors Abbilden lassen, wie beispielweise RSA Dekodierung. Der Hauptvorteil der Rekonfigurierbare Architekturen erfolgreich mit VLSI Spezialschaltungen konkurrieren lässt, ist dass sich die Schaltungen dynamisch an die Eingabedaten anpassen lassen. Dies bringt bei der Mustererkenntung (DNS Sequenzierung und Suche in Zeichenketten) und beim RSA Dekodierungen den eigentlichen Leistungsprung. Dynamische Optimierungstechniken werden auch für Mikroprozessoren untersucht. Die erste kommerzielle Anwendung dürfte SUNs Hot Spot Compiler sein [24].

Die Eignung von FPGAs ist bisher überwiegend für systolische Algorithmen untersucht worden. Diese Arbeit soll an einem Beispiel untersuchen, wie gut FPGAs für einen etwas komplizierteren, rekursiven Algorithmus auf einer zuflächigen Graphen abschneiden.

2.2.2 Neue FPGA Architekturen

Die auf dem Markt verfügbaren FPGAs wurden nicht im Hinblick auf Rekonfigurierbare Architekturen entwickelt. Deshalb stellt sich die Frage, wie soll ein FPGA aussehen der für RC geeignet ist? Unter anderem ergeben sich die folgenden Anforderungen:

Einfaches Timingmodell (BRASS, TSFPFA, DFG) Synchroner Arbeitsweise mit einer garantierten Taktfrequenz unabhängig vom Pausch eines Signals und der detailierten Plazierung sind erforderlich, um die Algorithmen für die dynamische Schaltkreissynthese einfach und schnell zu halten. Außerdem würde sonst die Taktfrequenz zu stark von den Eingabedaten abhängen.

Pipelining (BRAS, TSFPFA) Die Ausführungseiten eines FGPA sind typischerweise sehr schnell im Vergleich zum Verdrahtungsnetzwerk. Da bei vielen Algorithmen der Datendurchsatz wichtig ist, als die Versorgungszahl lohnt es sich Flp-Flops in das Verbindungsnetzwerk einzufügen. Dazu dann im allgemeinen Verbindungsweg gibt die eine unterschiedlich viele Takte benötigen, sind dann zusätzlich retiming Register an den Eingängen der Ausführungseinheiten nötig. BRASS Trumpet kann auf diese Weise für beliebige Anwendungen Taktraten von 350 MHz garantieren, was in herkömmlichen FGPA nur in Ausnahmefällen möglich ist.

Schnelle Neukonfiguration (Xilinx 6200, BRASS, DFG) Wenn sich die Konfiguration des FPGA im Betrieb häufig ändert, sollten die Änderung der Konfiguration schnell gehen. Verschiedene Techniken können dabei helfen:

- Breiter, schneller Datenpfad zur Neukonfiguration (XC6200, BRASS)
- Änderung von Teilkonfiguration (XC6200, BRASS)
- (DRAM) Konfigurationsspeicher mit breitem Datenpfad zum Konfigurationsspeicher (neorAM [22], BRASS, DFG)
- Double Buffering, eine zweite Konfiguration läßt sich verändern während die erste Konfiguration benutzt wird. Die Umschaltung erfolgt in einem einzelnen Takt (DFG)

2.2.3 Vorteile von FPGA Rechnern

Die Realisierung einer Schaltung als FPGA ist relativ ineffizient. Für die Konfiguration, vor allem aber für die feste Verdrahtung wird sehr viel Fläche benötigt. Eine typische Aufteilung der Chipfläche eines FPGAs ist 90% für die Verdrahtung, 9% für den Konfigurationspeicher und 1% aktive Logik. Dies erhöht die Kosten und reduziert die Geschwindigkeit einer Schaltung gegenüber einer VLSI Realisierung.


2.3 RA im Vergleich zu Standardprozessoren


Die Architektur der Prozessoren ist dabei traditionell dadurch bestimmt, dass eine kleine Anzahl aktiver Resourcen ständig wieder verwendet wird. In den Anfängen der Computerentwicklung war dies notwendig, da Hardware teuer und fehleranfällig war. Schon seit langem werden jedoch komplette Prozessoren in einem einzigen Baustein realisiert, und die verfügbare Fläche wächst ständig weiter. Doch wie lässt sich diese Fläche nutzen?

2.3.1 Probleme hoher Instruktionsspeicherraten


Die Leistungsfähigkeit der Prozessoren wird schon lange nicht mehr von der für die Berechnung der Operationen benötigte Zeit bestimmt. So benötigt beispielsweise der in Hewlett-Packards PA-RISC 8200 Prozessor 2 sekunden 64-Bit Addierer, in aus heutiger Sicht schon fast altdöckerer 0,5 µm Technologie, nur 0,7 ns für eine Addition. Diese Zykluszeit des Prozessors beträgt jedoch nur 6,3. Das bedeutet, es hat wenig Sinn, die zusätzliche verfügbare Hardware zu verwenden, um die Ausführung der Standardoperationen zu beschleunigen.
zu beschleunigen.


Eine Möglichkeit die Leistung zu erhöhen ist, mehrere Befehle gleichzeitig auszuführen. Werden s Befehle gleichzeitig ausgeführt, so sind s Ausführungszeiten gleichzeitig aktiv. Es wird also absolut genommen mehr Hardware sinnvoll genutzt. Die Effizienz sinkt dadurch jedoch weiter.

Um auf diese Weise die s-fache Befehlsrate zu erreichen steigt nämlich der Flächenbedarf für Befehls cache mindestens proportional zu s. Auch der Aufwand für die anderen Abschnitte genannter Maßnahmen steigt.

Wenn dabei das sequentielle Prozessormodell beibehalten werden soll müssen zusätzlich Abhängigkeiten zwischen den gleichzeitig ausgeführten Befehlen aufgelöst werden. Insgesamt steigt der Aufwand für die Steuerlogik bei superskalaren Prozessoren schneller als linear mit der Zahl der parallel ausgeführten Befehle, so daß der Anteil der für die Ausführungszeiten verwendeten Chipfläche tendenziell sinkt.

Dieses Problem ist lange bekannt, und es gibt mehrere Ansätze den Befehlstrom zu entlasten indem einer Dateipfadkonfiguration mehrere Operationen ausgeführt werden.

**Vektorrechner** Vektorrechner führen die selbe Operation auf einem Vektor statt auf einem Skalar aus. Die Konfiguration des Datenpfades bleibt dabei über eine gewisse Anzahl von Takten bestehen.

**MMX** Die Multimedia-Erweiterungen verschiedener Chiphersteller sind ein Spezialfall der Vektorrechner bei dem durch minimale Eingriffe in die Architektur eines Skalar- oder Superskalprozessors den kleinen Eingriffen geringer Länge - typischerweise 2 bis 8 Zahlen - bearbeitet werden können.

**VLIW** (Very Large Instruction Word) Prozessoren mit einem sehr großen Befehlswort reduzieren zwar im allgemeinen nicht die Anforderungen an die Befehlsbandbreite, die Steuerlogik wird jedoch dadurch vermindert, daß es per definition keine Abhängigkeiten zwischen gleichzeitig ausgeführten Operationen gibt, und daß alle Operationen eines Befehls gleichzeitig ausgeführt werden können. Es können also bei gleichem Aufwand für die Steuerlogik mehr Operationen gleichzeitig ausgeführt werden.

Rekonfigurierbare Architekturen versuchen möglichst ganze Teilalgorithmen in einer Konfiguration unterbringen und die Rekonfiguration zu so weit zu reduzieren, daß der Nachteil der erheblich größeren Konfiguration ausgeglichen wird.

**2.3.2 Unangepaßte Wortbreite**

Aus den oben angegebenen Gründen kann die Wortbreite der Prozessoren erhöht werden ohne die Fläche oder die Geschwindigkeit des Prozessors wesentlich zu beeinflussen. Die Wortbreite wächst daher tendenziell in der Holung, mehr Arbeit mit einem Befehl erledigen zu können. Dies erhöht zwar die absolute Leistung, reduziert aber die Effizienz, da die Fläche der Datencache proportional zur Wortbreite des Prozessors steigt, die Leistung bei den meisten Anwendungen aber nur geringfügig steigt.

Prozessoren sind also bei Anwendungen die auf kleinen Datenwerten operieren sehr ineffizient (DNS Sequenzverarbeitung, RSA Dekodierung). Prozessoren können sich hingegen optimal der von der Anwendung benötigten Wortbreite anpassen. Trotzdem sind FPGAs bei Prozessortypischen Wortbreiten
inoeffizient, da die Fähigkeit jedes Bit einzeln zu konfigurieren für diese Anwendungen nicht genutzt wird. Es gibt daher eine Reihe Rekonfigurierbarer Architekturen die mit mittleren Wortbreiten von etwa 8 Bit arbeiten.

Der in dieser Arbeit vorgestellte Algorithmus arbeitet mit 4-Wertiger Logik. Bei einer Implementierung auf einem 64-Bit RISC Prozessor hat man die Wahl entweder immer nur 2 Bit - etwa 3% - der Datenpfade und des Datencaches zu benutzen, oder mehrere Signalgewichte in ein Wort zu packen, was Zusatzaufwand für die Dekodierung bedeutet.

2.3.3 Schlechte Ausnutzung von Konstanten


Es gibt in der Compilerforschung Ansätze, selbstmodifizierende Software zu generieren, die diesen Effekt zur Softwareoptimierung nutzt, Rekonfigurierbare Architekturen profitieren jedoch noch weitaus stärker davon, da sie nicht nur die Zahl der Operationen sondern auch die Komplexität der Operatoren reduzieren kann. Beim DNS Sequenzieren Beispielweise reduziert die Fläche durch die Spezialisierung auf eine zu suchende Sequenz um den Faktor 2. Konstantenmultiplikationen mit einer n-Bit Konstanten erfordern im schlimmsten Fall n/2, erwartet etwa $\sqrt{n}$ Additionen. Wenn, wie bei FIR Filtern, die Quantisierung der Koeffizienten beeinflusst werden kann, kann die Zahl der Additionen noch weiter reduziert werden.

2.3.4 Schlechte Verwaltung von Zwischenwerten

Um ein Zwischenresultat, das nicht in den Registern abgelegt werden kann, zu verwenden, muss ein Prozessor einen hohen Aufwand betreiben. Effektive

Addressen müssen berechnet werden, Cache-tags für mehrere Caches verglichen werden, virtuelle Addressen müssen in physikalische Addressen umgerechnet werden, und so weiter. Die dafür verwendete Hardware beträgt immer den gleichen Aufwand, auch wenn die Zugriffsfolge vorhersehbar ist.

In Rekonfigurierbaren Architekturen können die Speicherzugriffe wesentlich besser optimiert werden, da Zwischenwerte müssen oft gar nicht gespeichert werden, sondern können gleich an die nächste Ausführungseinheit weitergegeben werden. Außerdem kann die Speicherarchitektur in der Regel an die Anwendung angepasst werden. Die in dieser Arbeit vorgestellte Schaltung kann zum Beispiel in einem Takt etwa 250 Byte Registerinhalte mit Daten aus einem Stack austauschen. Bei 10 MHz entspricht das einer Bandbreite von 5 GByte/s. Parallel dazu kann etwa die doppelte Datennenge zwischen die Ausführungseinheiten bewegt werden. Durch die direkte Weiterleitung der Zwischenwerte an die nächste Ausführungseinheit entfallen außerdem die sonst für Graph aufgaben erforderlichen Zeigerreferenzerung.

2.4 RA's fordern hohen Entwicklungsaufwand


Das führt dazu, dass man Zeit für die Implementierung eines Algorithmus auf einer RA noch fast so hoch ist wie für die Entwicklung von Spezialhardware. Dadurch können RA's trotz ihrer theoretischen Eignung für allgemeine Aufgaben in der Praxis heute nur für Spezialanwendungen
eingesetzt werden.

Die Entwicklung von Compilern für diesen Bereich ist sehr schwierig. Deshalb versucht das BRASS Projekt der Uni Berkeley die FPGA-Architektur so zu optimieren, daß die Leistung nicht ganz so stark von der Qualität der Synthesesoftware abhängt. Dazu dienen die in Kapitel 2.2.2 vorgestellten Techniken.

BELEG: Chipfläche für ALU vs RESE
BELEG: Prozentanteil der Anwendungen mit PPU, MUL, MMX, etc.
INDEX: Mikroprozessor, Prozessor, superskalar, Sprungvoraussetzung, spekulative Ausführung
Kapitel 3

Implikationsberechnung in booleschen Netzwerken
TODO: Andre mit Accent
Befehl korrigiert: s/& c statt & (active), statt & (processor), Klein/groß,
Anglizismen, fehlende Buchstaben am Wortende (besonders e)
Literaturverzeichnis


[8] Peixing Zhong, Margeret Martonosi, Pranav Asgar, Sharad Malik, "Accelerating Boolean Satisfiability with Configurable Hardware" in FPGA '98

[9] HP, Addiererdesign für den HP 8200

[10] HP, Strength Reduction


[13] Hartenstein?

[14] DPGA

[15] GARP

[16] Matrix

[17] PADDI

[18] von Neumann cellular


[23] Leigerson, hierarchical routing on PAT trees

26
[24] SUN, HotSpot