Sharing and Binding for General Circuits

Benedikt Lipinski
Interaktionstechnik und Design
Hochschule Hamm Lippstadt
Lippstadt, Germany
benedikt.lipinski@stud.hshl.de

Abstract

CONTENTS

I. Introduction

Durch das menschliche Verlangen, Problemstellungen zugunsten schwindender Komplexität und erhöhten Komforts zu automatisieren und zu brechen, wurden genau diese Anforderungen auf diejenigen elektrischer Schaltungen umgelegt.

A. Problemstellung

Nicht erst durch moderne Entwicklungen wie digitale Vernetzung mit dem Internet der Dinge oder das autonome Fahren wird die Betrachtung zeitlicher und finanzieller Aspekte für die Entwicklung multidimensionaler Systeme notwendig. Allerdings lassen genau diese Entwicklungen die Komplexität sprunghaft ansteigen.

B. Logische Bausteine

Realisierbar werden elektrische Verschaltungen in der Digitaltechnik erst durch den Einsatz sogenannter Logikbausteine, die miteinander kombiniert ein vorher genau bestimmtes Verhalten der geplanten Schaltung verursachen. Hierbei übernehmen verschiedene Bausteine verschiedene Aufgaben, die mal weniger und mal mehr komplizierte Operationen beinhalten.

1) AND-Gatter: Eine der Grundoperationen, die in fast jeder Schaltung zu finden ist, ist die AND-Operation, zu deutsch die UND-Operation. Hierbei wird der Eingang erst auf logisch 1 geschaltet, wenn alle Eingänge dies ebenfalls sind.

x	y	Out
0	0	0
0	1	0
1	0	0
1	1	1

2) OR-Gatter: Das Ergebnis des OR-Gatters ist in dem Fall wahr, wenn einer seiner Eingänge wahr ist.

x	y	Out
0	0	0
0	1	1
1	0	1
1	1	1

Simple Schaltungen lassen sich durch das gezielte Aneinanderreihen von AND- und OR-Gattern gut realisieren, sollte aber

3) Multiplexer: Multiplexer werden genutzt, wenn eine parallele Verarbeitung aus unterschiedlichsten Gründen nicht möglich ist. Das kann zum einen eine nicht ausreichende Verfügbarkeit an Kanälen sein, aber auch die bewusste Entscheidung, weniger Kanäle aus Kostengründen zu benutzen, sein. Dieser Schritt kann gewählt werden, wenn Verläufe zwar parallel laufen, aber nur exklusiv ausgeführt werden. Das bedeutet, dass nach einer elementaren Entscheidung nur einer der beiden Stränge ausgeführt werden kann. Beeindruckend allerdings ist, dass Multiplexer trotz ihrer fortgeschrittenen Eigenschaften gegenüber der Grundbausteine doch recht simpel aus genau diesem erzeugt werden können. So ergibt sich für einen einfachen Multiplexer das Schaltbild aus zwei AND-Gattern, einem NOT-Gatter und einem OR-Gatter. .....

4) FPGA: Der Einsatz von diesen Grundelementen ist für das Erreichen des Ziels völlig hinreichend, doch benötigen sie eine sehr hohe Fläche, auf der sie verbaut werden müssen. Ein zu erreichendes Ziel ist, die Fläche auf der die Schaltung realisiert werden muss, zu verringern. Dies ist durch die Verwendung von Halbleitern und die Entwicklung von integrierten Schaltkreisen auch erfolgreich gelungen, allerdings besitzen diese sogenannten ICs den Nachteil, dass ihre Schaltung- wie die einer Realisierung mit Bausteinen- auch fest gebunden ist. Ein Nachteil, den FPGA (Field Programmable Gate ARRAY) Bauteile nicht besitzen, ihr Vorteil ist die Zusammensetzung aus zusammengeschalteten Baugruppen aus Logikbausteinen, FlipFlops und deren Verbindung mit Multiplexern. In FPGAs kann das Verhalten der Eingänge zu den Ausgängen über..... ”programmiert” werden.

C. Strategien zur Architektur-Optimierung

Mit steigender Komplexität der Verschaltungen steigt zwingend auch die Anzahl der zu nutzenden Bauteile. Zudem steigt auch die Verwendung performaterer Bausteine stetig, um die Abarbeitung bezüglich Leistung und aller zeitlichen Deadlines einhalten zu können. Nachteilig ist vor dem Hintergrund ebenfalls, dass Bausteine mit einer höheren Leistung oftmals auch einen erhöhten Platzbedarf einfordern [2, S.327].
Ziel ist es, Strategien umzusetzen, die die Chipfläche verringern, ohne dabei die Leistungsfähigkeit der Schaltung zu beeinflussen. Leider ist dieses Problem nicht mit der Einhaltung beider Anforderungen zu vereinbaren, wodurch zum Beispiel die Verminderung der Kosten durch Senkung der Chip-Anzahl und Fläche immer auch zum Nachteil der Performance agiert. Der Zusammenhang zwischen Fläche und der erzeugten Leistungsverminderung ergibt sich aus der Fläche a in der Einheit Gatteräquivalent $(GateEquivalentGE)$, das die Größe des simpelsten Gatters widerspiegelt [2, S.326] und der Performance, die den Kehrwert der Ausführungszeit T_{ges} widerspiegelt $\frac{1}{T_{ges}}$, die sich wieder aus der Taktzeit T_c multipliziert mit der Latenz λ zusammensetzt [2, S.325-326]. Eine mögliche Realisierung der Schaltung mit einer bestimmten Performance bei einer bestimmten Fläche lässt sich als Punkt in einem Diagramm darstellen, dessen Achsen sowohl die Chipfläche, die Latenz und den Takt abbilden [2, S.326]. Dieses Vorgehen [**?**] ist in diesem Fall notwendig, da mehr als eine Variable optimiert werden muss, und es dementsprechend mehr als eine gültige Lösung dieses Problems gibt. Als mögliches Hilfsmittel kann hier die Pareto-Optimierung verwendet werden. Die Pareto-Optimierung erzeugt eine Kompromisslösung durch die Gewichtung jeder Einzelfunktion. Die Idee dahinter ist, dass die Gewichtung die Wichtigkeit der Einzelfunktion repräsentiert und ihr bei steigendem Gewicht mehr Aufmerksamkeit verleiht [4, S.45].

D. Allocation

Nicht nur, aber vor allem aus Kosten und Platzgründen stehen der Entwicklung selten unbegrenzte resourcen zur verfügung. Aus diesem Grund kann die Betrachtung der zur verfügung stehenden resourcen zwingend notwendig werden.

Somit ist das ergebnis der allokierung nach Teich [1] , eine funktion $\alpha(r_k)$ die angibt wie viele verfügbare Bauteile es zu einer Jeweiligen Resourcentyp gibt.
Damit würde, wenn vom Resourcentyp Mulitplizier: r_1,Beispielweise 1 Einheiten zur verfügung stehen die Allokierungsfunkion $\alpha(r_1) = 2$ lauten. Die Funktion der Allokation zeigt deshalb ziemlich genau welche probleme beziehungsweise zeitliche veränderungen entstehen durch die grenzen, die durch die begrenzte verfügbarkeit an Resourcen gegeben werden. Dies kann ein nützlicher zwischenschritt sein um den überblick zu behalten,den Teich mittels eines Ressourcen graphen anschaulich darstellt.In vielen Literaturen allerdings wird dieser schritt auch in die Bindung integriert und nicht extra behaldelt.

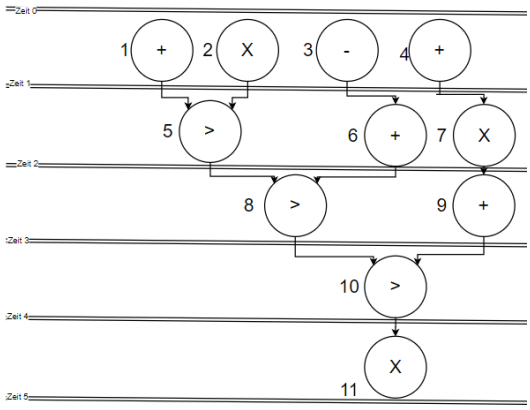

Fig. 1. Besipiel zur Bindung

E. Binding

Im unterschied zur Allokierung wird in der Bindung schließlich festgelegt welche Resource welche operation zugeteilt bekommt,also die Zuweisung von Operationen zu bestimmten hardware Komponenten [3]. in einer Utopischen vorstellung in der Kosten und Fläche keine rolle spielen würden, könnte für jeden Rechnenschritt, jede Operation, also zum beispiel für jede Multiplikation oder jeden vergleich eine eigene Komponete Benutzt werden. Leider ist diese vorstellung in der Praxis nicht durchführbar, da jedes gesparte bauteil auf eine Große stückzahl gerechnet den Kostendruck lindert.Das ist zudem auch nicht zwingend nötig, da Beilspielsweise ein Vergleich nach eine Addition, welche beide durch eine Alu realisiert werden können, nicht unbedingt durch 2 verschiedene Hardware einheiten umgesetzt werden müssen, da der vergleich erst stattfinden kann wenn das ergebnis der Addition vorliegt. genannt wird dies in diesem fall ein Optimal Resource sharing oder zu Deutsch ein Optimales ausnutzen einer Resource [2].
Die Bindung bietet ein Tool, das veranschaulicht wie optimal die vorhandene schaltung auf und mit den gegebenen Bauteilen realisiert wird. Da das engültige Ziel eine Optimale ausnutzung der vorhandenen resourcen ist müssen nun schrittweise die benötigten Bauteile minimiert werden, dies wird erreicht indem anfangs jede Operation einem bauteil zugeordnet wird, es wird ein sogenannte dedizierte Ressource erzeugt, ein spezialfall [3, S. 150]

Dem Bild ist zu entnehmen das in diesem beispiel für die zuordnung jeder Aufgabe zu einer Resource:11 Bauteile benötigt werden. Die sich aus 3 Multiplikationen = r_1 und aus 8 operationen für eine Arithmetisch logisch einheit r_2 ergibt.

v_i	$\beta(v_i)$	$\gamma(v_i)$
v_1	r_2	1
v_2	r_1	1
v_3	r_2	2
v_4	r_2	3
v_5	r_2	4
v_6	r_2	5
v_7	r_1	2
v_8	r_2	6
v_9	r_2	7
v_10	r_2	8
v_11	r_1	3

Da es sich, um ein recht einfach zu überblickendes beispiel handelt, lässt sich durch einen geübten blick bereits erkennen wie eine Optimierung aussehen könnte. Im ersten Zeitslot ist abzulesen, das 3 Arithmetische operationen und eine Multiplikation durchgeführt werden, somit werden 3 ALU Einheiten und ein Multiplizierer benötigt.Im weiteren verlauf lässt sich unteranderem erkennen das in keinem weiteren zeitzlot mehr als 3 ALU einheiten gebraucht werden, deswegen ist dies auch unsere minimal benötigte anzahl an Resourcen dieses typs. Ganz genau deutlich wird dies durch eine veranschaulichung in einer tabelle.

Zeitslot	ALU	Multiplizierer
1	3	1
2	2	1
3	2	0
4	1	0
5	0	0

Aus der Tabelle kann nun abgelesen werden, das 3 Resourcen des Typs ALU benötigt werden und 1 des Typs Mulitplizierer. Übertragen auf die obere Tabelle ensteht nun folgendes Binding.

v_i	$\beta(v_i)$	$\gamma(v_i)$
v_1	r_2	1
v_2	r_1	1
v_3	r_2	2
v_4	r_2	3
v_5	r_2	1
v_6	r_2	2
v_7	r_1	1
v_8	r_2	1
v_9	r_2	2
v_10	r_2	1
v_11	r_1	1

Somit wurden in diesem beispiel 5 Alu einheiten die einen anteil von $62,5\%$ ausmachen eingespart. Bei den Multiplizierern wurde die benötigten Resourcen auf $\frac{1}{3}$ reduziert.

F. Sharing

das sharing beschreibt zusammen mit dem Binding, die effektive mehrfachnutzung der Ressourcen, welches der minimierung der verwendeten hardware dient und somit zu reduzierung von Flächen und kosten problemen beiträgt. wobei das ziel des Sharing die frage nach dem Alghoritmus zur optimerung ist und anschließend mit einem Binding endet. In dem Kapitel Binding wurde ein einfach im Kopf zu lösendes Beispiel gezeigt, leider bleibt es in der Realität nicht immer bei so einfach zu lösenden systemen, so dass da schnell der überblick verloren gehen kann und zudem ein system unter umständen nicht zu lösen wäre. Aus diesem grund befasst sich dass sharing mit der frage wie ein vorhandenes system effektiv optimiert werden kann.Zu den Verwendeten Alghoritmen zählen unter anderem aus dem breich der REsource Dominated Circuits das Register sharing, Multiport Memory Binding und das Bus sharing and Binding und für den bereich der general circuits, sind es Unconstrained minimum - Area Binding, Performance Constrained Binding und zu letzt Performance Directed Binding.

II. Grundlegendes

A. Kompatiblitäts- und Konfliktgraphen

Als vorrausetzung, um überhaupt eine sinnvolle Bindung beziehungsweise Allokierung erzeugen zu können muss sich im 1. schritt einmal mit dem eigentlichen problem befasst werden. Um das Problem darstellen zu können wird ein Graph aufgestellt, der die eigenschaften der elemente beschreibt. Dieser Schritt ermöglicht die betrachtung der einzelnen operationen mit einander und stellt auf welche mit einander Kompatibel sind und welche ein kritisches verhalten zu einander aufweisen.
Zwei oder mehr Operationen sind mit einander Kompatibel und können an die Gleiche Resource gebunden werden wenn sie nicht mit einander Konkurrieren und wenn sie dem gleichen Resourcen-typen angehören [3, S.231]
Das bedeutet konkreit, dass aufgaben mit einander kompatibel sind, wenn sie nicht im gleichen Zeitslot neben einander anliegen und wenn sie des gleichen typs sind zum Beispiel Addierer. Die entwicklung der ges graphensystems kann grundsätzlich über 2 ansätze erfolgen, zum einen kann ein Kompatiblitätsgraph oder Verträglichkeitsgraph entwickelt werden, dieser graph gibt an welche Operationen verträglich zu einander sind und somit keine probleme darstellen komplemäntar dazu lässt sich durch den Konflikt graphen angeben welche Operationen im konflik zu einander stehen und eine genauere betrachtung bedüfen.
Ein Ressourcen Verträglichkeitsgraf ist definiert durch $G_+(V,E)$ in denen die Knoten durch $V = \{v_i; i = 1,2,...,n\}$ definiert werden und die Kanten durch $E = \{\{v_i, v_j\} i,j = 1,2,...,n\}$ Die Knoten stellen die Operationen an sich dar, wohingegen die Kanten die Kompatiblen Operationspaare angeben [3, S.231].
Zusammen gefasst nach Teich [1, S. 181-182] lassen sich folgende verträglichkeiten für den kompatiblitätsgraphen aufführen:

demnach ist eine **Schwache verträglichkeit** gegeben, wenn zwei knoten des Graphen eines Grahen mit einander verbunden sind und des gleichen Types angehören also zum beispiel eine Addition und ein Vergleich, da beide durch eine Arithmetisch logische einheit Verarbeitet werden können. [**?**]

Ablaufplan verträglich ist ein graph hingegen, wenn seine Knoten als erste bedingung vom gleichen typ sind, also wie bei der schwachen verträglichkeit zum Beispiel aus 2 operationen vom typ Multiplizierer. Zum zweoten muss die bediingung erfüllt sein, das die starzeit des ersten knoten vor der des zweiten knotens liegen muss. Das bedeutet, dass Knoten die Ablaufverträglich sind nicht in dem selben zeitslot ausgeführt werden dürfen, andernfalls wären sie nicht Ablaufplan vertträglich [**?**].

Als letztes definiert sich die **Starke verträglichkeit**, durch zwei Knoten die wieder vom gleichen typ sind und zudem von einandern abhängig sind. Das bedeutet die beiden knoten sind in einem logischen strang durch einen pfad im ablaufplan mit einander verbunden. [**?**]

Die Graphische umsetzunge des Verträglichkeitsgraphes mit dem Beispiel aus den vorherigen kapiteln sieht dann wie folgt aus:

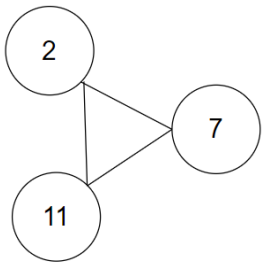

Fig. 2. Verträglichkeitsgraph Mulitplizierer

Aus den vorherigen kapiteln, wissen wir bereits, dass für die Optimierung der verschaltung nur ein Multiplizierer benötigt wird und genau das können wir in diesem beispil nun auch

grafisch betrachten. Zu sehen ist, dass jeder knoten mit einer kante verbunden ist und somit sind alle Operationen mit einander Ablaufplan verträglich, weswegen der einsatz von nur einem Multiplizierer nun auch Optisch besser nachzuvollziehen ist.

Als nächsten schritt werden nun sogenannte Cliquen eingeze-

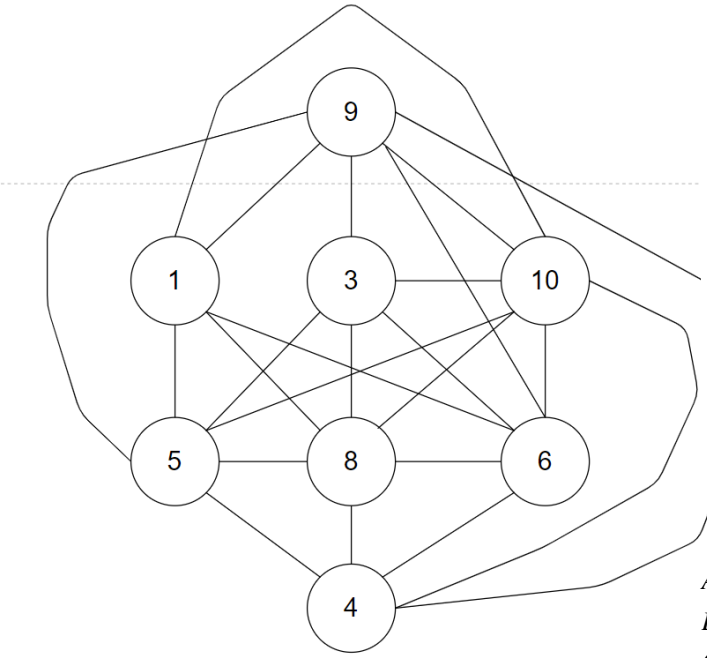

Fig. 3. Verträglichkeitsgraph Arithmetisch logische einheit mit Cliquen

ichnet, sie repräsentieren die zusammenfassung Mehrerer Operationen und zeigen ihre zusammenführung auf eine Komponente, so das für jede clique genau eine Komponente eingesetzt wird. Zu sehen ist, das Bei dem Verträglichkeitsgraphen der Mulitplizierer genau eine Clique eingezeichnet werden muss. Da mit steigender Komplexität der aufgabe, auch die Komplexität für das einzeichen der Cliquen steigt, bietet sich dieses verfahren leider nur für die anwendung an simple verschaltungen an, der Grund hierfür ist die klassifizierung dieses problems als NP-Hartes Problem. [2, S.361]

B. Resource Dominated circuits

da probleme leider ein wenig komplizierter

behamndelt wir die grundlegende optimierung mit dem grundlagen wissen aus Konflikt und Optimum graph

z.b mittels register sharing -G.micheli S 156

III. General Circuits

A. Allgm.

ganz allgemein wird jetzt auch die verzögerung durch steuerlogik und die verkabeluing betrachtet, also wie man so schön sagt ein reale verschaltung

B. Baugruppen

1) Register:

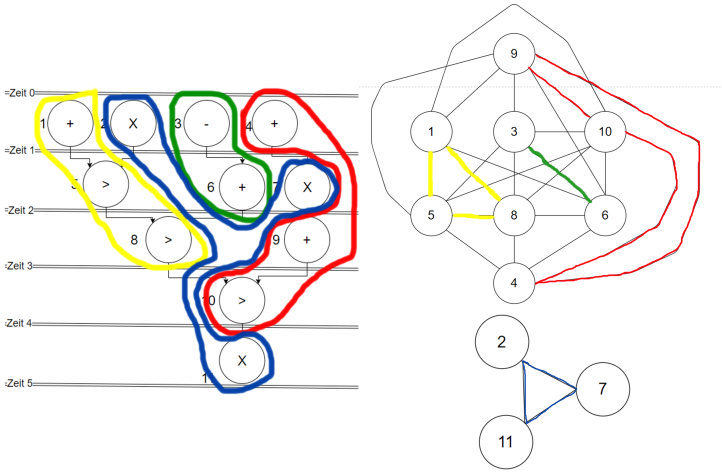

Fig. 4. Ablaufplan und verträglichkeitsgraphen mit Cliquenbildung

2) Steuerlogik:

3) Verkabelung:

4) kontroll einheiten:

IV. Sharing and Binding for General Circuits

A. Unconstrained minimum - Area Binding

B. Performance Constrained Binding

C. Performance Directed Binding

V. Gleichnis,vorgestellter Algorithmen

VI. Ausblick

References

[1] Jürgen Teich, Christian Haubelt, Digitale Hardware/Software-Systeme, Synthese und Optimierung, 2. Auflage,Springer,2007

[2] Dr.Walter Lange, Prof.Dr.Martin Bogdan, Entwurf und Syntese von Eingebetteten Systhemen, Ein Lehrbuch,Oldenbourg Verlag München, 2013

[3] DEMICH

[4] Martin Pieper, Mathematische Optimierung, Eine Einführung in die Kontinuierliche Optimierung mit Beispielen,Springer Spektrum,2017