Estimating and Optimizing Power Consumption of Integrated Macro Blocks at the Behavioral Level

Lars Kruse

lars.kruse@epost.de

Die stetig steigende Komplexität integrierter Schaltungen führt zu einer zunehmenden Bedeutung eines Verlustleistung reduzierenden Designprozesses. In dieser Arbeit sind neue Techniken zur Leistungsabschätzung und zur Reduktion der Leistungsaufnahme integrierter Schaltungen entwickelt worden, die bereits in der Spezifikationsphase eingesetzt werden können. Es kommen dabei Algorithmen aus dem Bereich des Operations Research zum Einsatz. Die Abweichungen der Verfahren liegen bei den betrachteten und für die Praxis relevanten Fälle im Bereich von wenigen Prozent.

1 Einleitung

Die hier vorgestellte Arbeit verknüpft den abstrakten Hardware-Entwurf auf algorithmischer Ebene mit Analyse- und Optimierungsmethoden der Verlustleistung. Dieser Ansatz wird durch zwei zentrale Problemstellungen motiviert. Zum einen reduziert sich in vielen Bereichen die Lebensdauer von Produkten mit integrierten Schaltungen. Dies hat zur Konsequenz, dass die Entwicklungszyklen der Schaltungen bei steigender Komplexität ebenfalls kürzer werden müssen. Eine Lösung dieses Problems ist eine stärkere Automatisierung des Entwurfes unter Einsatz abstrakterer Designbeschreibungen. Bei den hier betrachteten reinen algorithmischen Beschreibungen ist lediglich die Funktionalität des Chips spezifiziert. Das zeitliche Verhalten wird vom Designer durch Randbedingungen vorgegeben. Synthesewerkzeuge transformieren die Beschreibung weitestgehend automatisch unter Einhaltung möglichst aller Randbedingungen und Minimierung von Kostenfunktionen wie die Chipfläche in ein detailliertes Modell, welches zur Fertigung der Chips verwendet werden kann. Die wesentlichen Aufgaben, die die Synthese bei der Umsetzung einer algorithmischen Beschreibung zu bewältigen hat, sind das Scheduling (wann werden Operationen ausgeführt), das Binden oder Abbilden (auf welcher Ressource wird eine Operation ausgeführt) sowie das Allokieren von Ressourcen (wieviele Multiplizierer werden beispielsweise verwendet).

Zum anderen ist durch die zunehmende Integrationsdichte und Komplexität integrierter Schaltkreise die Verlustleistung bzw. die Energieaufnahme eine bedeutende Dimension neben der Chip-Fläche und den Zeitanforderungen im Entwurfsraum integrierter Schaltungen geworden. So verbraucht beispielweise eine moderne Variante des PowerPC Prozessors 112W bei einer Versorgungsspannung von nur 1,87V. Das bedeutet, dass ein Strom

von ungefähr 60A zugeführt werden muss. Eine hohe Leistungsaufnahme bringt erhebliche Nachteile mit sich. So muss die aufgenommene Leistung vom Chip in Form von Wärme wieder abgeführt werden. Für Chips mit einem hohen Verbrauch bedeutet dies den Einsatz von teuren Gehäusen bzw. bei sehr hohen Leistungsaufnahmen sogar von passiven oder gar aktiven Kühlsystemen, die die Gesamtsystemkosten stark erhöhen. Eine hohe Leistungsaufnahme bedeutet auch, dass hohe Ströme im Chip fliessen. Diese reduzieren die Lebensdauer des Chips durch Effekte wie Elektromigration (Atomtransport in den Leiterbahnen bei hohen Stromdichten) oder beeinflussen die Zuverlässigkeit negativ beispielsweise durch *Übersprechen* von benachbarten Signalleitungen. Die Minimierung der Energieaufnahme ist eines der primären Ziele beim Entwurf von mobilen, batteriebetriebenen Applikationen. Eine geringere Energieaufnahme bedeutet eine höhere Batterielebensdauer der Applikation. Da nicht zu erwarten ist, dass die Energiedichte von Batterien in dem Maße in Zukunft steigen wird, wie die Komplexität und der damit einhergehenden Energieaufnahme von integrierten Systemen, wird die Energieaufnahme einen noch weiter steigenden Stellenwert im Entwurfsprozess moderner Systeme erhalten.

Im folgenden werden die entwickelten Techniken zur Verlustleistungsanalyse und zur Reduktion der Leistungsaufnahme der Hardware-Ressourcen (auch Makroblöcke oder Makros genannt), wie bspw. Addierer, Multiplizierer oder Register, während des algorithmischen Entwurfsprozesses vorgestellt. Diese Verfahren sind aufgrund der Fokussierung auf Makroblöcke insbesondere für Schaltungen der digitalen Signalverarbeitung geeignet, da in diesen Schaltungen die Verlustleistung überwiegend in den Rechenwerken und Registern entsteht.

2 Verlustleistungsanalyse von Algorithmen

2.1 Quellen der Verlustleistung in integrierten Schaltkreisen

Die Verlustleistung einer Schaltung wird bei der überwiegend eingesetzten CMOS Technologie bestimmt durch eine dynamische und eine permanente Komponente. Die permanente Komponente, welche im wesentlichen unabhängig vom Betriebszustand des Systems ist, ergibt sich aus sogenannten Leckströmen im Halbleiter. Dieser Anteil lässt sich durch Fertigungstechniken reduzieren und ist für die meisten Schaltungen vernachlässigbar gering. Der überwiegende dynamische Anteil der Verlustleistung entsteht bei Umladevorgängen von Kapazitäten auf dem Chip. Es fliessen dabei sowohl Lade-/Entladeströme als auch Kurzschlussströme. Das heisst, dass jeder Wechsel des logischen Wertes eines Signals einen Stromfluss zur Folge hat. Die Leistungsaufnahme lässt sich somit durch eine Reduktion der Schaltaktivität und der Versorgungsspannung, mit der die internen Kapazitäten umgeladen werden, minimieren. Letzteres ist ein sehr effizientes Mittel, da die Spannung V_{dd} quadratisch in die Leistungsaufnahme P eingeht:

$$P = \frac{1}{2} \cdot \alpha \cdot C \cdot {V_{dd}}^2 \tag{1}$$

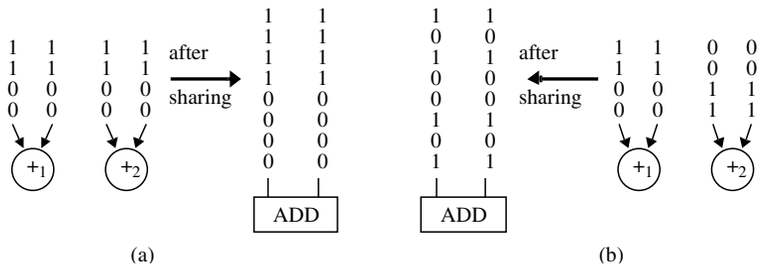

Abbildung 1: Einfluss von gemeinsamer Ressourcennutzung. (a) Reduktion der Schaltaktivität. (b) Erhöhung der Aktivität.

α ist hier die Schaltaktivität pro Zeiteinheit und C die Kapazität. Der Minimalwert der Spannung wird jedoch durch die eingesetzte Halbleitertechnologie begrenzt, um Signalqualität und maximale Verzögerungszeiten zu garantieren. Daher liegt der Fokus dieser Arbeit auf einer Abschätzung bzw. Minimierung der Schaltaktivität.

Die Schaltaktivität läßt sich nicht ohne weiteres bestimmen, selbst wenn wie hier angenommen eine Ausführung der Designbeschreibung durchgeführt ist und alle Resultate und Zwischenergebnisse des Algorithmus bekannt sind [CP95]. Das Problem liegt darin begründet, dass erst nach der Synthese die konkreten Datenströme in der Hardware berechnet werden können. Die gemeinsame Nutzung von Hardware-Ressourcen bestimmt die Daten, die die Ressourcen verarbeiten, und somit die Schaltaktivität an deren Ein- und Ausgängen. Abbildungg 1 macht dies an zwei extremen Beispielen deutlich. Links sind zwei Additionen gezeigt, die die gleichen Operandenwerte verknüpfen. Werden diese Additionen jeweils auf getrennte Addierer abgebildet, so finden insgesamt vier Schaltvorgänge (Bitwechsel) statt. Eine Hintereinanderausführung auf einem Addierer (Komponente *ADD*) reduziert die Anzahl der Bitwechsel auf zwei. Rechts in der Abbildung ist der entgegengesetzte Fall gezeigt: die zweite Addition verknüpft die invertierten Daten der ersten Addition. Mit zwei Addierern ergeben sich wieder vier Bitwechsel. Bei einer gemeinsamen Ressource erhöht sich die Schaltaktivität auf zwölf. Zwischen diesen beiden Extremen sind natürlich alle Varianten möglich.

2.2 Modellierung der Verlustleistung

In der Literatur sind zahlreiche Modelle für die Leistungsaufnahme von Makroblöcken vorgestellt worden [RJD98]. Gemeinsames Ziel dieser Modelle ist, die Leistungsaufnahme eines Makros für gegebene Daten oder Datenstatistiken an seinen Ein- und Ausgängen zu berechnen. In den meisten Fällen ist dazu eine einmalige Charakterisierung der Makros notwendig. Hierbei werden durch elektrische Simulationen der Makros für bestimmte, vorgegebene Daten die Verlustleistung berechnet. Aus den Werten werden dann die Modelle generiert. Das in dieser Arbeit eingesetzte Modell verwendet die Hamming-Distanz sowie die Anzahl der konstant auf logisch Null bleibenden Bits aufeinanderfolgender Eingangs-

werte pro Eingang (Operand) [JKSN99]. Dies bedeutet beispielsweise, dass das Modell für einen Addierer vier Parameter hat. Eine längere Datenfolge ist für die Verlustleistungsberechnung eines neu am Eingang anliegenden Datenwortes für die hier betrachteten Makros nicht notwendig.

Eine Normierung der Modellparameter auf die Bitbreite des jeweiligen Eingangs erlaubt zudem, dass das Modell für alle Eingangsbitbreiten skalierbar ist. Die Skalierungsfunktion wird während der Charakterisierung mittels Regression über verschieden grosse Instanzen der Makros realisiert.

Um die Auswirkungen aller möglichen Abbildungen von Operationen und Variablen auf Makros bestimmen zu können, werden alle Operations- bzw. Variablenpaare betrachtet. Für jedes Paar wird die Leitungsaufnahme (im folgenden auch als *Kosten* bezeichnet) für den Fall berechnet, dass die beiden Operationen bzw. Variablen direkt zeitlich hintereinander auf einer Ressource ausgeführt bzw. gespeichert werden. Nur die direkte Nachfolgerbeziehung ist von Interesse, da die Modelle nur die direkt aufeinanderfolgenden Daten berücksichtigen. Zwei Werte müssen für jedes Paar und einer gegebenen Ausführungsreihenfolge berechnet werden, wie im folgenden verdeutlicht wird. Angenommen, Addition $+_1$ wird vor der Addition $+_2$ im Algorithmus bspw. aufgrund von Datenabhängigkeiten berechnet (in der Hardware *ausgeführt*). Das Verlustleistungsmodell berechnet zum einen die Kosten, wenn die Eingangsdaten von $+_2$ denen von $+_1$ folgen, wobei die Daten jeweils aus der gleichen Ausführung des Algorithmus stammen. Diese Kosten werden *Intra-Iterationskosten* genannt. Da in der Regel ein Algorithmus in Hardware, z.B. ein digitialer Filter, wiederholt ausgeführt wird, müssen die Eingangsdaten der Operation $+_1$ aus Iteration $i+1$ mit denen von $+_2$ aus Iteration i ebenfalls verknüpft werden (*Inter-Iterationskosten*). Die Kosten werden in einer Matrix K abgelegt, wobei die untere Dreiecksmatrix die Intra-Iterationskosten speichert:

$$K(i,j) = \begin{cases} \infty, op_i, op_j \text{sind nicht kompatibel} \\ \frac{1}{T}\sum_{t=1}^{T} h(op_i(t), op_j(t)), i < j \\ \frac{1}{T}\sum_{t=1}^{T} h(op_i(t), op_j(t+1)), i \geq j \end{cases}$$

wobei

- $h(p,q)$ das entsprechende Verlustleistungsmodell bezeichnet, welches die Kosten für die Eingangsdaten p und q berechnet,
- $op_i(t)$ bezeichnet die Eingangsdaten der Operation op_i in Iteration t, und
- T gibt die Anzahl der betrachteten Iterationen an.

Zwei Operationen werden als *inkompatibel* bezeichnet, wenn sie gleichzeitig in Hardware ausgeführt werden müssen und sich deshalb keine Ressource teilen können. Die zeitliche Abfolge der Operationen wird durch das Scheduling während der Synthese bestimmt. Ist kein Schedule gegeben und stehen zwei Operationen nicht in einer Vorgängerrelation aufgrund von Datenabhängigkeiten, so müssen zwei Kostenmatrizen aufgestellt werden (eine pro Reihenfolge). Im folgenden wird davon ausgegangen, dass ein Schedule gegeben ist. Wie der andere Fall behandelt wird, ist in der Arbeit nachzulesen.

Die Diagonaleinträge $K(i,i)$ stellen einen Spezialfall dar. Sie speichern die Kosten, wenn die Operation op_i eine dedizierte Ressource erhält, also keine Ressourcenteilung stattfindet. Die Kostenmatrix kann bspw. wie folgt angewendet werden. Die Kosten für die Ausführung von drei Additionen $+_1$, $+_2$ und $+_3$ auf einem Addierer in der angegebenen Reihenfolge ergeben sich zu: $K_+(1,2) + K_+(2,3) + K_+(3,1)$, wenn K_+ die Kostenmatrix für die eingesetzten Addierer ist. Der letzte Term $K_+(3,1)$ ist notwendig, um die Inter-Iterationskosten zu berücksichtigen.

2.3 Problemformulierungen

Im folgenden werden die beiden zentralen Problemstellungen dieser Arbeit formuliert. Es wird dabei davon ausgegangen, dass ein Schedule sowie entsprechende Kostenmatrizen vorhanden sind. In beiden Fällen ist nach einer Abbildung von Operationen auf Makros gefragt, die bei einer Synthese zu berechnen ist. Das erste Problem betrachtet die Variante ohne eine Begrenzung von Ressourcen, während das zweite Problem eine Ressourcenanzahl (Allokation) vorgibt. Eine Allokation ist sinnvoll, um den Flächenverbrauch des Chips zu begrenzen.

Problem P1 Gegeben ist eine Kostenmatrix $K(i,j), i,j \in \{1,\ldots,n\}$ für n Operationen $op_1,\ldots,op_n$ eines Typs. Wie groß ist die minimale [maximale] Verlustleistung der zu verwendenden Makros, wenn die n Operationen auf eine nicht eingeschränkte Anzahl von diesen (von 1 bis n) abgebildet werden?

Problem P2 Gegeben ist eine Kostenmatrix $K(i,j), i,j \in \{1,\ldots,n\}$ für n Operationen $op_1,\ldots,op_n$ eines Typs sowie m allokierte Makros (Ressourcenbeschränkung). Wie groß ist die minimale [maximale] Verlustleistung der m Makros, wenn diese n Operationen auf sie abgebildet werden?

Es wird sowohl nach einer minimalen als auch nach einer maximalen Leistungsaufnahme gefragt. Beide Wert zusammen erlauben eine Aussage über die Größe des sogenannten *Designraums*, d.h. welche Kosten sind im besten Fall realisierbar bzw. welche Kosten können im schlechtesten Fall entstehen, wenn die Leistungsaufnahme nicht bei der Synthese berücksichtigt wird.

Beide Problembeschreibungen lassen sich als entsprechende graphentheoretische Probleme formulieren. Gegeben ist ein gerichteter, kantenbeschrifteter und vollständiger Graph $G(V,A)$ mit $V = \{op_1 \ldots, op_n\}$ und $A = V \times V$ die Menge der Kanten. Jede Kante $(op_i, op_j) \in A$ ist mit den Kosten $w_{ij} = K(i,j)$ beschriftet. Das Optimierungsproblem für P1 lautet dann, alle Knoten aus V mit knotendisjunkten Kreisen zu überdecken, so dass die Gesamtkosten minimal [maximal] sind und die Vorgängerrelation der Operationen erhalten bleibt. Die Gesamtkosten ergeben sich aus der Summe der Kanten, die zu den Kreisen gehören. Jeder Kreis einer Lösung repräsentiert eine Ressource, wobei die

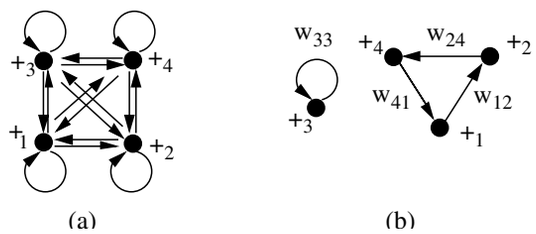

Abbildung 2: Problemformulierung mit einem Graphen (ohne Kantenbeschriftung). (a) Definition. (b) Eine mögliche Lösung.

Knoten eines Kreises die Operationen angeben, die auf die Ressource abgebildet sind. Für Problem P2 muss lediglich ergänzt werden, dass die Anzahl der Kreise auf m beschränkt ist.

Abbildung 2 zeigt ein Beispiel mit vier Additionen. Links ist der vollständige Graph dargestellt, während rechts eine Lösung hervorgehoben ist. Schleifen, wie in (b) an Operation $+_3$ dargestellt, repräsentieren Ressourcen, die genau einer Operation zugeteilt sind.

In der Arbeit konnte bewiesen werden, dass das Problem P2 NP-vollständig ist. Der Beweis gelingt durch Reduktion des Problems *Partitionierung von tripartiten Graphen in Dreiecke* [GJ79]. Ein entsprechender Beweis oder Gegenbeweis konnte bislang für Problem P1 nicht gefunden werden.

2.4 Untere und obere Schranken der Verlustleistung

Da zumindest für Problem P2 aufgrund der NP- Vollständigkeit relativ sicher ist, dass keine optimale Lösung in polynomieller Zeit gefunden werden kann, wird der Ansatz zur schnellen Berechnung von guten unteren bzw. oberen Schranken verfolgt. Das Optimum wird dabei von *unten* angenähert, während die schlechteste Lösung von *oben* approximiert wird. Dies erlaubt eine gute Abschätzung der Größe des Designraums.

Das folgende Integer Lineare Programm (ILP) gibt die optimale Lösung für P1 an (die schlechteste Lösung wird berechnet, indem $\min$ durch $\max$ ersetzt wird):

$$z = \min \sum_{i,j=1}^{n} w_{ij} x_{ij} \quad \text{mit folgenden Randbedingungen:} \tag{2}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad i = 1, \ldots, n \tag{3}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad j = 1, \ldots, n \tag{4}$$

$$x_{ij} \in \{0, 1\} \tag{5}$$

$$\text{Menge von Gleichungen, die die Vorgängerrelation modellieren} \tag{6}$$

$x_{ij} = 1$ bedeutet, dass die Kante (op_i, op_j) zur Lösung gehört und somit die Operationen

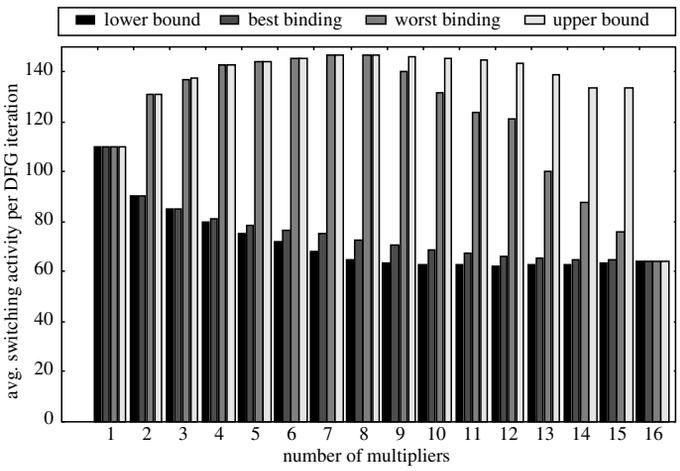

Abbildung 3: Vergleich der unteren und oberen Schranken mit der besten/schlechtesten Lösung (Multiplikationen der Cosinus Transformation).

op_i und op_j direkt hintereinander auf einer Ressource ausgeführt werden. Gleichungen (3) und (4) beschreiben die Kreisüberdeckung. Die Gleichungsmenge (5) ist hier nicht angegeben, da diese zur Berechnung der Schranken entfernt werden. Das verbleibene ILP mit den Gleichungen (2) bis (5) beschreibt das sogenannte *Zuweisungsproblem*, welches in $O(n^3)$ mit der *Ungarischen Methode* gelöst werden kann [PS82].

Ein ILP für P2 kann durch Erweiterung des oberen Programms mit der Gleichung:

$$\sum_{i \geq j} x_{ij} = m \tag{7}$$

gewonnen werden. Diese Gleichung beschränkt die Anzahl der Kreise der Lösung auf exakt m, da genau m Kanten von Operationen mit höherem Index zu Operationen mit niedrigerem Index zugelassen werden. Es wird vorgeschlagen, dieses Gleichungssystem ((2)-(5), (7)) mittels einer Lagrange Relaxation zu approximieren [Fis85]. Dies ist ein iteratives Verfahren, wobei in jeder Iteration das Zuweisungsproblem gelöst wird. Dazu wird die Gleichung (7) aufgelöst und mit einem multiplikativen Faktor in der Zielfunktion (2) berücksichtigt.

Die Abbildung 3 zeigt Abschätzungsergebnisse für P2 im Vergleich zu der optimalen bzw. schlechtesten Lösung. Als Beispiel dient hier eine Cosinus Transformation, die 16 Multiplikationen enthält. Auf der X-Achse sind für alle möglichen Allokationen die durchschnittliche Hamming-Distanz als Kostenfunktion pro Iteration der Transformation aufgetragen. Ein Bild einer Gebäudereihe wurde transformiert und lieferte dabei die Datenströme. Es zeigt sich, dass in allen Fällen die untere Schranke das Optimum sehr gut approximiert. Die Berechnung der oberen Schranke weist für grössere m Abweichungen auf. Hier wird der Einfluss der Relaxation der Vorgängerrelation sichtbar. Diese Fälle sind aller-

dings von geringerem Interesse, da zum einen der maximale Wert nur selten Anwendung findet und zum anderen kleine m bevorzugt werden (Flächeneinsparung). Die Abbildung verdeutlicht auch den Einfluss des Abbildens auf die Schaltaktivität. Das Maximum für $m = 8$ ist doppelt so gross wie das entsprechende Minimum. Dies zeigt, dass eine Berücksichtigung der Leistungsaufnahme während der Architektursynthese wirkungsvoll ist.

3 Optimierung der Verlustleistung in der Architektursynthese

Die Optimierungstechniken, die in diesem Abschnitt beschrieben werden, zielen darauf ab, die Verlustleistung durch geschicktes Abbilden der Operationen auf die Ressourcen während der Synthese zu reduzieren. Diese Optimierungen gehen dabei von einer gegebenen Ausführung der Designbeschreibung aus, d.h. die Operandenwerte aller Operationen sind bekannt. Es findet somit eine Optimierung für einen bestimmten Datenstrom statt. Untersuchungen haben aber gezeigt, dass ein bestimmtes Optimum auch eine gute Architektur für Datenströme mit vergleichbarem Korrelationsgrad ist.

Zwei Optimierungsarten werden hier vorgestellt. Der nächste Abschnitt beschreibt zunächst Heuristiken, die mit geringem Berechnungsaufwand gute Architekturen liefern. Danach werden Techniken vorgestellt, das Optimum finden. Beide Arten haben gemeinsam, dass die entwickelten Algorithmen die Berechnung der oben vorgestellten unteren Schranken einsetzen.

3.1 Heuristiken

Die Lösung einer Berechnung von unteren Schranken stellt in der Regel keine legale Lösung des ursprünglichen Problems dar, weil nicht alle Randbedingungen (hier die Vorgängerrelation) berücksichtigt werden. Aus einer unteren Schranke lässt sich aber eine legale Abbildung der Operationen berechnen, indem verletzte Randbedingungen aufgelöst werden. Die sogenannte *Verschmelzungsheuristik* ist eine einfache Variante, die zunächst alle Kreise einer unteren Schrankenlösung an den Stellen aufteilt, an denen die Vorgängerrelation verletzt ist. Beispielsweise wird aus dem Kreis $(+_1, +_3, +_4, +_2, +_5, +_1)$ die beiden Kreise $(+_1, +_3, +_4, +_1)$ und $(+_2, +_5, +_2)$, wenn der Index die Ausführungsreihefolge der Operationen angibt. $+_2$ darf nicht zwischen $+_4$ und $+_5$ ausgeführt werden. Falls nach diesem Schritt für P2 die Anzahl der Kreise m übersteigt, so müssen Kreise verschmolzen werden. Es werden solange jeweils die beiden Kreise verschmolzen, die die Gesamtkosten am geringsten erhöhen, bis keine Verbesserung gefunden wird (lokales Minimum). Zwei Kreise können dabei immer nur auf eine bestimmte Weise verschmolzen werden.

Das Ergebnis der Verschmelzungsheuristik kann im allgemeinen weiter verbessert werden. Die *3-optimale Heuristik* testet alle sinnvollen Permutationen von drei Operationen. Sinnvoll sind nur Permutationen, bei denen die Operationen aus drei verschiedenen Kreisen stammen. Es wird jeweils die Permutation gewählt, die die Gesamtkosten am stärksten reduziert. Die Heuristik iteriert bis zum lokalen Minimum. Diese Heuristik kann erweitert

werden, in dem grössere Gruppen von Operationen permutiert werden. In der Arbeit wurde noch eine 2x2 Variante untersucht, bei der zwei Operationspaare vertauscht werden. Noch grössere Gruppen sind nicht sinnvoll, da der Berechnungsaufwand deutlich steigt, aber kaum bessere Ergebnisse erzielt werden wie in der Tabelle unten zu sehen. Sie zeigt einen Vergleich der verschiendenen Heuristiken am Beispiel der Cosinus Transformation. Die mittlere Abweichung für die 2x2-optimalen Variante liegt im Durchschnitt unter 1%.

	HVZ	H3Z	H2x2Z	HPL	H3L	H2x2L
natürlich	5,32	0,89	0,04	2,53	0,26	0,06
künstlich	15,09	3,02	0,39	7,19	1,6	0,18

Tabelle 1: Vergleich der Heuristiken für Cosinus Transformation mit natürlichen bzw. künstlich generierten Bildern. HV: Verschmelzung, H3: 3-optimal, H2x2: 2x2-optimal. Untere Schranke Z: Zuweisungsproblem, L: Lagrange.

3.2 Optimale Verfahren

Es existieren verschiedene Strategien, um kombinatorische Probleme wie P1 und P2 zu lösen. Drei Ansätze wurden untersucht. Der einfachste Ansatz (*ER*) ist eine erschöpfende Suche, bei der alle legalen Lösungen berechnet werden. Eine Verbesserung besteht in der sogenannten *Branch-and-Bound* Methode darin, dass durch die Berechnung von unteren Schranken Teile des sogenannten Lösungsbaumes ausgeschlossen werden können, falls die untere Schranke des Teilbaumes höher als eine bislang gefundene Lösung ist. Als initiale Lösung dienten die oben beschriebenen Heuristiken. Der Algorithmus *BBZ* verwendet die untere Schranke aus dem Zuweisungsproblem, während *BBL* die Lagrange Schranke einsetzt. Der letzte untersuchte Ansatz verwendet einen allgemeinen ILP Löser, der iterativ komplexer werdende ILPs verarbeitet. Zwei verschiedene Ansätze wurden untersucht. Startpunkt ist in beiden Fällen das ILP des Zuweisungsproblems für P1 bzw. ein um Gleichung (7)erweitertes ILP für P2. Nach jeder Iteration wird die erhaltene Lösung nach verletzten Randbedingungen an die Vorgängerrelation untersucht. Wird eine Bedingung gefunden, so stellt das Ergebnis nur eine untere Schranke dar und eine weitere Iteration muss durchgeführt werden. In der ersten Variante (*ILP1*) wird eine Gleichung dem ILP hinzugefügt, so dass die Bedingung immer erfüllt ist. Bei der zweiten Variante (*ILP2*) werden für alle verletzten Randbedingungen jeweils eine Gleichung ergänzt. Diese Variante hat den Vorteil, dass mit großer Wahrscheinlichkeit weniger Iterationen durchgeführt werden müssen. Jedoch werden die zu lösenden ILPs u.U. schnell größer, so dass die einzelnen Berechnungen länger dauern.

Die folgende Tabelle zeigt einen Vergleich der fünf Varianten. Die größten Probleme können mit dem Branch-and-Bound Algorithmus berechnet werden, der die untere Schranke des Zuweisungsproblems einsetzt.

	ER	BBZ	BBL	ILP1	ILP2
n	16	22	10	14	18

Tabelle 2: Maximale Problemgröße n, die innerhalb von 24h berechnet werden konnte (440MHz Ultra Sparc CPU).

Literaturverzeichnis

[CP95] J.-M. Chang and M. Pedram. Register Allocation and Binding for Low Power. *Proc. of IEEE/ACM Design Automation Conf.*, pages 29–35, 1995.

[Fis85] M.L. Fisher. An Applications Oriented Guide to Lagrangian Relaxation. *Interfaces*, 15(2):10–21, March-April 1985.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: Freeman, 1979.

[JKSN99] G. Jochens, L. Kruse, E. Schmidt, and W. Nebel. A New Parameterizable Power Macro-Model for Datapath Components. *Proc. of Design, Automation and Test in Europe*, pages 29–36, 1999.

[PS82] C.H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization, Algorithms and Complexity*. Prentice Hall, Inc., 1982.

[RJD98] A. Raghunathan, N.K. Jha, and S. Dey. *High-Level Power Analysis and Optimization*. Kluwer Academic Publishers, 1998.

Lars Kruse studierte von 1991 bis 1996 Informatik an der Carl von Ossietzky Universität Oldenburg. Von 1996 bis 2000 war er wissenschaftlicher Mitarbeiter am Oldenburger Forschungsinstitut für Informatik-Werkzeuge und -Systeme (OFFIS). Dort beschäftigte er sich im Rahmen von internationalen Projekten mit dem Thema der Verlustleistungsanalyse und -optimierung auf der Gatter-, Register-Transfer- und der algorithmischen Abstraktionsebene. Von 2000 bis 2001 übernahm er die Position des Gruppenleiters in der Forschungsgruppe 'Low Power Design'. 2001 schloss Herr Kruse seine Promotion mit dem Thema *Estimating and Optimizing Power Consumption of Integrated Macro Blocks at the Behavioral Level* ab. Die entwickelten Verfahren werden zur Zeit von der Firma Chipvision AG kommerzialisiert. Seit Januar 2001 arbeitet Herr Kruse bei der Firma Thales Electronic Engineering als Consultant im Bereich Embedded Software und Hardware Design.