Methoden und Werkzeuge der Schaltungssynthese

Dipl.-Inform. Peter Blinzer, TU-Braunschweig, Abteilung Entwurf integrierter Schaltungen (E.I.S.)

Zusammenfassung

Beim Entwurf einer Schaltung ist die Auswahl einer geeigneten Synthesemethode aus RTL-Synthese, Controllersynthese oder High-Level-Synthese eine wichtige Voraussetzung für das Erreichen guter Ergebnisse. In diesem Beitrag werden mehrere Kriterien zur Einordnung von Entwürfen vorgestellt, welche die Auswahl von geeigneten Synthesemethoden und -werkzeugen vereinfachen. Diese Kriterien basieren auf einer Untersuchung charakteristischer Eigenschaften der verschiedenen Methoden und Werkzeuge und wurden durch eine Vielzahl von Entwurfsexperimenten verifiziert und verfeinert. Hierbei zeigte sich neben der elementaren Bedeutung der RTL-Synthese für alle Synthesemethoden auch der starke Einfluß der Optimierungsziele, insbesondere der Laufzeit. Die charakteristischen Eigenschaften, die Ergebnisse repräsentativ ausgewählter Entwurfsexperimente und die Auswahlkriterien bilden den Kern dieses Beitrages.

1 Einleitung

Verschiedene Methoden und Werkzeuge zur Schaltungssynthese ermöglichen die Nutzung der sich aus den beständig wachsenden Integrationsdichten ergebenden Möglichkeiten des Schaltungsentwurfs.

Zu den verbreiteten Methoden gehören neben der Register-Transfer-Level-Synthese (RTL-Synthese) die Controllersynthese und zu einem noch kleinen, wachsenden Anteil die Synthese algorithmischer Beschreibungen, die auch als Verhaltens- oder High-Level-Synthese bezeichnet wird. Diese Synthesemethoden und die sie umsetzenden Software-Werkzeuge unterscheiden sich erheblich in den synthetisierbaren Beschreibungsformen, den verwendeten Synthesealgorithmen und in der Größe des Ergebnisspielraums. Die Auswahl einer geeigneten Methode und entsprechender Werkzeuge erfordert ein Abwägen verschiedener Eigenschaften und oftmals verwirrender Aussagen. So vermitteln beispielsweise die Werbebehauptungen der Softwarehersteller den Eindruck, jedes ihrer Werkzeuge und die darin umgesetzte Synthesemethode wäre universell für alle Entwurfsaufgaben geeignet und garantiere optimale Lösungen bei kürzester Entwurfszeit. Die seit langem verkündete, bevorstehende Ablösung des „veralteten“ RTL-Entwurfsstils durch den algorithmischen Entwurf, als Parallele zur Verdrängung des Entwurfs auf Logikebene durch den RTL-Entwurf, läßt nicht nur weiter auf sich warten, sondern wirkt in laufenden Entscheidungsprozessen verunsichernd.

Um die Auswahl einer Synthesemethode zu vereinfachen und Unsicherheiten zu beseitigen, wurden die Möglichkeiten und Beschränkungen der RTL-Synthese, Controllersynthese und der High-Level-Synthese untersucht und verglichen. Hierzu wurden charakteristische Eigenschaften der Methoden betrachtet und mit diesen Methoden arbeitende Werkzeuge auf mehrere, unterschiedliche Entwurfsaufgaben angewendet. Aus den hierbei gewonnenen Ergebnissen wurden Entscheidungskriterien für die Auswahl von Methoden und Werkzeugen in Abhängigkeit von den Eigenschaften der Entwürfe abgeleitet.

In den folgenden Abschnitten werden zunächst die Synthesemethoden und die bei den Entwurfsexperimenten verwendeten Werkzeuge vorgestellt, wobei die Methoden jedoch nur in Hinsicht auf ihre charakteristischen, für die Auswahlkriterien bedeutenden Eigenschaften betrachtet werden. Für eine allgemeinere Einführung in die Synthesemethoden bzw. die verwendeten Werkzeuge wird auf [5], [8], [9], und [10] verwiesen. Danach werden zwei der in den Syntheseexperimenten untersuchten Entwürfe und einige der mit ihnen gesammelten Ergebnisse betrachtet, wobei die Synthesemethoden einander gegenüber gestellt werden. Die beiden Entwürfe wurden wegen ihrer Einfachheit und der Deutlichkeit ihrer Ergebnisse ausgewählt und erlauben im Rahmen dieses Beitrages die Darstellung elementarer Eigenschaften und Abhängigkeiten, welche die Grundlage der genannten Auswahlkriterien bilden. In diesem Zusammenhang werden auch die Auswirkungen der verwendeten Optimierungsziele betrachtet, insbesondere der Laufzeit- und Flächenoptimierung. Abschließend werden die Auswahlkriterien für die einzelnen Synthesemethoden zusammengefaßt und ein Ausblick auf weiterführende Arbeiten gegeben.

2 Eigenschaften der RTL-Synthese

Die RTL-Synthese, die derzeit den Anwendungsschwerpunkt von Synthesemethoden und -werkzeugen in der Praxis bildet, zeichnet sich durch mehrere Eigenschaften aus, die sie von den anderen Methoden unterscheidet.

Ähnlich vielen anderen, nicht synthesespezifischen Schaltungsmodellierungen oder Varianten der High-Level-Synthese erfolgt die Eingabe der Entwürfe zwar auch mit verbreiteten, textuellen Hardwarebeschreibungssprachen (*hardware description languages*,

HDLs), wie Verilog oder VHDL, jedoch mit einigen Besonderheiten. So sind die genormten HDLs Verilog und VHDL für synthesegeeignete Entwürfe nur unter Beachtung vieler Einschränkungen und genauer Vorgaben für die Modellierung von Registern und kombinatorischer Logik nutzbar. Besonders zu berücksichtigen sind die Synchronisation von Anweisungsfolgen auf äußere Ereignisse, die einer Flipflop-Taktung oder einer kombinatorischen Verarbeitung entsprechen muß, der Ausschluß von konkurrierenden Zuweisungen auf Variablen aus parallelen Anweisungsprozessen und die je nach beabsichtigter Logik korrekte Verwendung von Zuweisungsoperationen und Verzögerungen. Dies bringt zwangsläufig eine detaillierte Beschreibung von sequentieller und kombinatorischer Logik bzw. synchroner und asynchroner Verarbeitung mit sich.
Neben der flachen Strukturierung durch Anweisungsprozesse bieten solche RTL-Beschreibungen mittels Verilog-Modulen bzw. VHDL-Architekturen und deren Instanzierung auch die hierarchische Zerlegung als Mittel der Strukturierung und Kapselung von Entwurfsteilen. Die Kommunikation der Module bzw. Architekturen über Signalschnittstellen erlaubt neben der Funktionsbeschreibung durch Anweisungsprozesse auch die Möglichkeit zur Beschreibung von Entwürfen in Netzlistenform. Zudem sind für technologieabhängige Entwürfe gezielt Bezüge auf Elemente der Zieltechnologie über Instanzen und ihre Verdrahtung möglich. Durch die detaillierte Beschreibung von Schnittstellen, Verdrahtungen und Signalabhängigkeiten können mit RTL-Beschreibungen alle mit Schematic-Entry erfaßbaren Konstruktionsmöglichkeiten nachempfunden werden, wie z.B. mehrere Taktnetze, pegelgesteuerte Speicherelemente und bidirektionale, asynchrone Verbindungen.
Die RTL-Synthese bildet die Verhaltens- und Strukturaspekte von HDL-Beschreibungen in Form von Gatternetzlisten auf die Logikebene ab, wobei entsprechend der Optimierungsvorgaben Redundanzen entfernt und Zerlegungsstrukturen ausgewählt werden. Als Ergebnis wird der Entwurf vollständig durch Elemente der Zieltechnologie beschrieben, beispielsweise durch NAND-Gatter oder FPGA-Logikblöcke.
Für die anderen beiden hier betrachteten Synthesemethoden, die Controllersynthese und die High-Level-Synthese, bildet die RTL-Synthese als Übergang von Register-Transfer-Beschreibungen zu Gatternetzlisten einen zentralen Bestandteil der Entwurfsabläufe. Daher wurde als Werkzeug für die RTL-Synthese der Design-Compiler von Synopsys [9] ausgewählt, der im Zusammenspiel mit den übrigen verwendeten Programmen bereits mehrfach erfolgreich eingesetzt wurde. Es existieren aber auch andere, vergleichbare Werkzeuge für die RTL-Synthese, mit denen die RTL-Syntheseexperimente ebenso ausführbar gewesen wären, wie Synplify (Synplicity) oder Leonardo (Exemplar Logic).

3 Eigenschaften der Controllersynthese

Im Vergleich zur RTL-Synthese zeichnet sich die Controllersynthese durch die Spezialisierung auf kontrollorientierte Probleme und eine höhere Abstraktion sowohl der Entwurfselemente als auch der Zeitspezifikation aus. Aufgrund der Vielfalt an herstellerabhängigen Sprachen und Werkzeugen, wie Statemate von i-Logix, Renoir von Mentor Graphics, Visual-HDL von Summit oder Protocol-Compiler von Synopsys, sind die allgemeingültigen, charakteristischen Eigenschaften der Controllersynthese nur eine vergleichsweise kleine Schnittmenge über alle Eigenschaften von Werkzeugen und Sprachen.
Die Eingabe erfolgt mittels graphischer und textueller Entwurfselemente je nach Werkzeug durch Zustandsgraphen, Ablaufdiagramme, Wertetabellen, Produktionen formaler Sprachen [6] oder einer der Backus-Naur-Form (BNF) ähnlichen Notation, die der Protocol-Compiler von Synopsys verwendet [10]. Die Entwurfselemente sind in ihrer Ausführung fest mit einem globalen Takt des Controllers synchronisiert, so daß sich Spezifikation und Abstraktion der Zeit auf Taktung und Register-Transfers beschränken. Zeitlich sequentielle Abläufe von Entwurfselementen werden mit automatisch generierten endliche Automaten (*finite state machines*, FSMs) implementiert. Dies befreit den Entwickler von der langwierigen und fehleranfälligen Aufgabe, alle FSM-Register, Zustandscodierungen und -übergänge und von Hand zu erstellen. Stattdessen bleiben Register, Übergänge und Codierungen unter der Oberfläche der Entwurfselemente verborgen und werden entsprechend der Zielvorgaben automatisch erzeugt [7]. Ein Controllerentwurf setzt sich meist aus mehreren, parallel arbeitenden und kommunizierenden FSMs zusammen.
Die beim Entwurf auf Register-Transfer-Ebene möglichen Schwierigkeiten durch Kombination von synchroner und asynchroner Verarbeitung oder Abhängigkeiten von Signallaufzeiten sind beim Controllerentwurf aufgrund der vollständig synchronen Verarbeitung zu einem von außen angelegten Takt ausgeschlossen. Dies kann insofern als Nachteil angesehen werden, als daß es die Kommunikation mit der Umgebung des Controllers auf ein taktgesteuertes Schema beschränkt. Jegliche asynchrone Verarbeitung muß hierbei in einen separaten Teilentwurf für die RTL-Synthese ausgelagert werden, der entsprechend der synchronen Verarbeitung des Controllers an diesen angebunden wird.
Die RTL-Synthese wird aber auch aus einem anderen Grund nicht durch die Controllersynthese ersetzt werden. Die Eingabesprachen der Controllersynthese sind je nach Werkzeug unterschiedlich, wodurch die Entwurfseingaben an die Werkzeuge gebunden sind. Die Entwürfe werden außerdem von den Controllerwerkzeugen nicht direkt auf die Gatterebene abgebildet, sondern als synthesegeeignete RTL-Beschreibun-

gen der Sprachen Verilog oder VHDL aus den Werkzeugen exportiert, welche im Anschluß mittels RTL-Synthese in Gatternetzlisten überführt werden können. Die RTL-Synthese ist damit ein unentbehrlicher Bestandteil im Entwurfsablauf der Controllersynthese. Die zweistufige Überführung eines Controllerentwurfs in eine Gatternetzliste über eine RTL-Beschreibung erhöht die Anzahl der möglichen Optimierungsvorgaben für den Syntheseverlauf deutlich, da neben den Vorgaben für die Gatterabbildung und -optimierung verschiedene Möglichkeiten zur HDL-Abbildung des Controllers ausgewählt werden können. Die sorgfältige Belegung der einzelnen Vorgaben und Optionen gewinnt hier also mit wachsender Entwurfsabstraktion an Bedeutung [1].
Für das in diesem Beitrag vorgestellte Syntheseexperiment eines Controllers wurde der Protocol-Compiler von Synopsys verwendet, der mit seiner auf Produktionen formaler Sprachen basierenden Erfassung strukturierter Datenströme am besten für diese Entwurfsaufgabe geeignet war. Sind keine solchen Datenströme zu verarbeiten, so wären für die Controllereingabe und -synthese Werkzeuge mit Zustands- oder Ablaufdiagrammen als Eingabeform zu bevorzugen.

4 Synthese eines Digital-Audio-Receivers

Eines der zahlreichen Syntheseexperimente, die in [1] sowohl mit RTL- als auch mit Controllersynthese durchgeführt wurden, beschäftigt sich mit einer Konvertierungsschaltung für digitale Audiodatenströme, die beispielsweise von CD-Playern oder DAT-Rekordern ausgegeben werden [3].
Bei der Synthese einer verhaltensorientierten Verilog-RTL-Beschreibung zeigten sich trotz Variation aller verfügbaren Optionen für die Abbildung auf Gatter und die Logikoptimierung keine sichtbaren Veränderungen in der resultierenden Gatternetzliste. Die in **Bild 1** sichtbaren Laufzeitschwankungen wurden ausschließlich durch die zufallsgesteuerte Plazierung und Verdrahtung hervorgerufen und sind unabhängig von den Optionen der RTL-Synthese. Als Zieltechnologie wurde hierbei ein Xilinx XC3042A FPGA verwendet, weswegen die Größe dieser Schaltung in FPGA-Logikblöcken (*combinatorial logic blocks*, CLBs) angegeben wird. Dieses Größenmaß ist trotz seiner höheren Granularität mit dem bekannten Gate-Count qualitativ vergleichbar, wie ein im Rahmen der Experimente ebenfalls ausgeführter Vergleich mittels linearer Korrelation zeigte [1].
Die Ergebnisse zeigen deutlich, daß die Register-Transfer-Struktur der Verilog-Beschreibung der Gatterabbildung und Logikoptimierung keinen ausreichenden Spielraum ließ, um die Ergebnisse merklich zu beeinflussen. Die Ursache hierfür liegt in der expliziten Konstruktion und Codierung der verschiedenen, entwurfsbestimmenden FSMs, die nur kleine Logikfunktionen zwischen Registern benutzten. Die Gatternetzliste war also wegen der Register-Transfer-Struktur und der geringen kombinatorischen Funktionalität schon zu genau festgelegt, um noch Optimierungen zu ermöglichen.

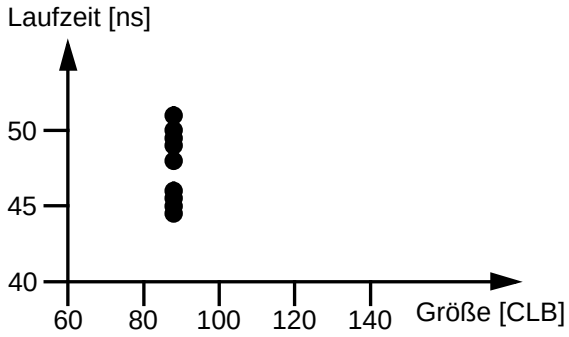

Bild 1 RTL-Synthese des Digital-Audio-Receivers

Die Verwendung der Controllersynthese für dasselbe Entwurfsproblem erlaubte erkennbar bessere Möglichkeiten zur Auslotung des Ergebnisraumes, wobei verschiedene Möglichkeiten zur Konstruktion und Codierung der FSMs angewendet wurden (**Bild 2**). So wurden als Konstruktionsvarianten neben einer monolithischen FSM für den gesamten Controller auch verschiedene, automatisch erstellte FSM-Partitionierungen betrachtet. In jeder dieser Konstruktionen konnten die FSM-Codierungen unabhängig voneinander auf mehrere Arten erfolgen, z.B. als Binärcode mit minimaler Breite, als Binärcode mit minimalen Bitänderungen bei Transitionen oder als One-Hot-Code. Allerdings waren die Ergebnisse nicht genauso kompakt und schnell, wie bei der handgeschriebenen RTL-Beschreibung. Dafür betrug die Entwurfszeit für die Eingangsbeschreibung der Controllersynthese nur etwa die Hälfte der für die direkte Entwicklung einer RTL-Beschreibung benötigten Zeit [2].

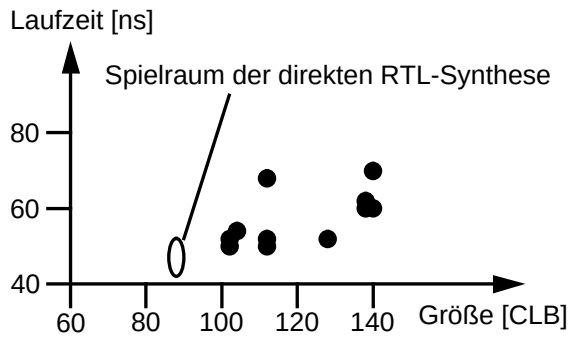

Bild 2 Controllersynthese des Audio-Receivers

Der direkte Vergleich von RTL-Synthese und Controllersynthese zeigt deutlich den eingeschränkten Spielraum der RTL-Synthese, der durch die detaillierte

Beschreibung von Register- und Logikstrukturen bedingt ist. Diese detaillierte Kontrolle bringt außerdem eine höhere Entwurfszeit mit sich. Mit der Controllersynthese ist eine zügigere Erfassung und Umsetzung synchroner, kontrollflußorientierter Entwürfe möglich, wobei die Ergebnisse mittels verschiedener Optionen und Zielvorgaben in Fläche und Arbeitsgeschwindigkeit an das Entwurfsumfeld angepaßt werden können. Die komfortable Eingabe der Entwürfe und einfache Auslotung des Ergebnisspielraums fordert allerdings ihren Tribut im Bereich der erreichbaren Schaltungsgeschwindigkeit und -größe.

5 Eigenschaften der High-Level-Synthese

Die High-Level-Synthese erhöht im Vergleich zur RTL- und Controllersynthese die Abstraktion der Entwurfselemente und der Zeitspezifikation von der Register-Transfer-Ebene mit ihren Laufzeiten und einzelnen Takten auf die Ebene von Algorithmen und Verarbeitungsschritten. Auch hier existieren unterschiedliche Werkzeuge, Eingabesprachen und Spezialisierungen auf Anwendungsgebiete, z.B. für Datenpfade digitaler Signalprozessoren (DSPs), Bildverarbeitungsalgorithmen oder programmierbare Prozessoren mit anwendungsspezifischen Befehlen (ASIPs). Aufgrund der Streuung von spezialisierten High-Level-Syntheseverfahren sind allgemeingültige Aussagen über charakteristische Eigenschaften des gesamten Synthesespektrums nicht möglich. Im folgenden wird daher nur eine relativ universelle Form der High-Level-Synthese betrachtet, wie sie die Werkzeuge Monet von Mentor Graphics und Behavior-Compiler von Synopsys durchführen.

Die Spezifikation von Zeiten erfolgt hierbei durch Takte und Taktintervalle und wirkt sich sowohl auf die zeitliche Bindung einzelner Operationen als auch auf die Umsetzung in Hardwarebaugruppen aus. Während bei der RTL- und Controllersynthese Operationen fest an Takte gebunden sind und sich die Struktur der Beschreibung in der resultierenden Gatternetzliste wiederfindet, ist dies bei der High-Level-Synthese anders. Der genaue Ausführungszeitpunkt oder -raum einer Operation im Taktschema wird nicht durch den Entwickler festgelegt, sondern durch das Scheduling der Synthese innerhalb eines bestimmbaren Zeitrahmens eingeordnet. In Kombination mit der Allokation von Hardwareressourcen, die eng mit dem Scheduling zusammenarbeitet, entsteht damit ein optimiertes Verhältnis zwischen Verarbeitungsleistung und Hardwarekosten für die geforderten Zielvorgaben.

Die Entwurfseingabe erfolgt wie bei der RTL-Synthese mit einer der textuellen HDLs Verilog oder VHDL, wobei jedoch die verwendbaren Sprachkonstrukte für die Spezifikation der Zeit und des reaktiven Verhaltens noch stärker eingeschränkt sind. Die algorithmischen Beschreibungen sind auf ein einziges Verilog-Modul bzw. eine VHDL-Architektur mit darin enthaltenen Anweisungsprozessen beschränkt. Die Zeitkontrollen dieser Prozesse synchronisieren die Anweisungen mit einem Taktsignal, wobei jede Synchronisation den Optimierungsspielraum der High-Level-Synthese verkleinert. Sie ist damit auf synchrone Schaltungen beschränkt und benötigt außerdem zeitliche Freiheiten für Operationen, um die High-Level-Optimierungen wirksam werden zu lassen. Durch Instanzierung von RTL-Entwürfen wird die Verbindung zu Register-Transfer-Beschreibungen geschaffen, wobei diese von der High-Level-Synthese jedoch nur angebunden und in keiner Weise optimiert werden.

Die High-Level-Optimierungen Scheduling und Allokation berücksichtigen im wesentlichen die arithmetischen Operationen der HDLs, können aber auch andere Hardwarebaugruppen optimiert in das Verarbeitungsschema der resultierenden Schaltung einpassen. Die bestmögliche Nutzung von Speicherelementen wird durch eine Lebenszeitanalyse der benutzten Variablen und die entsprechende Aufteilung der Variablen auf Hardwareregister erreicht.

Aus demselben Grund wie bei der Controllersynthese kann auch die High-Level-Synthese die RTL-Synthese nicht ersetzen. Die Implementierung asynchroner Kommunikationsschnittstellen und die Zerlegung der durch die High-Level-Synthese generierten RTL-Beschreibungen in Gatternetzlisten machen die RTL-Synthese und die sie ausführenden Werkzeuge auch hier unentbehrlich.

Die High-Level-Synthese bietet ähnlich der Controllersynthese eine große Anzahl von Möglichkeiten zur Beeinflussung der resultierenden Gatternetzliste, da auch sie ein zweistufiges Verfahren mit abschließender RTL-Synthese ist und verschiedene Vorgaben und Einstellungen für die High-Level-Optimierungen erlaubt. Die richtige Abstimmung der Optionen und Vorgaben auf den jeweiligen Entwurf und das angestrebte Ergebnis sind in der High-Level-Synthese wegen der großen Distanz der Entwurfsbeschreibung zur Gatternetzliste besonders wichtig [1].

Die besonderen Stärken der High-Level-Synthese zeigen sich bei Entwurfsproblemen, die mit algorithmischen Beschreibungen einfacher zu lösen sind, als mit einzelnen Register-Transfers und die Operationen mit hohen Hardwarekosten benötigen. Ebenso wie die Controllersynthese ergänzt die High-Level-Synthese die RTL-Synthese durch eine einfachere, problemorientierte Entwurfseingabe. Zudem eröffnet sie einen weiten Spielraum für Verarbeitungszeit und Komponentennutzung einer Schaltung.

Die Entwurfsexperimente für die High-Level-Synthese wurden mit dem Behavior-Compiler von Synopsys [8] durchgeführt, der mit dem Design-Compiler in einer Entwicklungsumgebung integriert ist und reibungslos mit diesem zusammenarbeitet.

6 Synthese eines Kryptographie-Datenpfades für das IDEA-Verfahren

Eines der sowohl mit RTL-Synthese als auch High-Level-Synthese ausgeführten Syntheseexperimente war die Implementierung eines Kryptographie-Datenpfades für das Verschlüsselungsverfahren IDEA [4]. Die Realisierung des IDEA-Algorithmus als Datenpfad basiert auf acht identischen Stufen, die regulär zu einer Pipeline verschaltet sind. Innerhalb jeder Stufe werden mehrere arithmetische Operationen, wie Additionen und Multiplikationen, ausgeführt. Die Syntheseexperimente beschränkten sich auf eine der acht Stufen, die mit Verilog für die RTL-Synthese und die High-Level-Synthese beschrieben wurde. Als Zieltechnologie wurde ein Xilinx-FPGA des Typs XC4062XL verwendet.

Die RTL-Synthese der IDEA-Stufe ergab in Abhängigkeit der benutzten Optimierungsoptionen und Zielvorgaben ein weites Spektrum an Lösungen. Die Menge an kombinatorischer Logik und ihre nur grobe Strukturierung in der Verilog-Beschreibung gaben den Algorithmen der RTL-Synthese genug Freiraum, um sichtbare Variationen im Verhältnis von Verarbeitungsgeschwindigkeit und Schaltungsgröße zu produzieren (**Bild 3**).

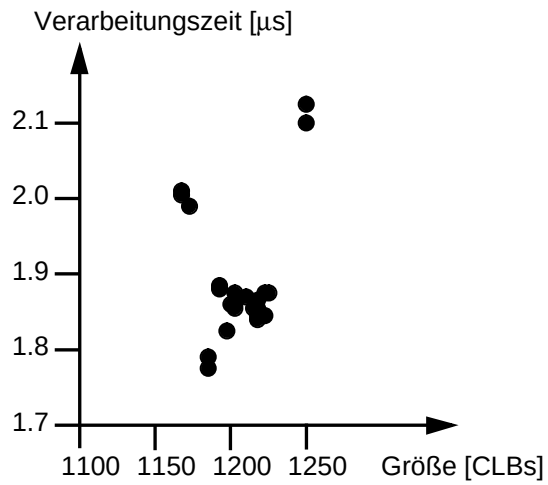

Bild 3 RTL-Synthese einer IDEA-Stufe

Die bei der RTL-Synthese benutzten Optimierungen beinhalteten die Neustrukturierung boolescher Funktionen und die Selektion verschiedener Operatortypen, wie z.B. Ripple-Carry-, Carry-Look-Ahead- oder Carry-Select-Addierer. Die Grundstruktur der Verilog-Beschreibung und der Bezug von Operationen zu Takten wurde trotz dieser Optimierungen vollständig in der Gatternetzliste beibehalten.

Eine herausragende Bedeutung bei der Steuerung der Ergebnisstreuung nahm die Optimierungsvorgabe der maximalen Signallaufzeiten ein. Die kürzesten Laufzeiten wurden mit Vorgaben erreicht, die bereits dicht an diesem Optimum lagen. Mit kleineren oder größeren Zeitvorgaben wurden schlechtere Laufzeiten erzielt und die übrigen Vorgaben und Optionen gewannen an Bedeutung, insbesondere die Vorgabe der maximalen Schaltungsgröße, bei der jedoch nur die Variation zwischen minimaler Fläche und irrelevanter Fläche Wirkung zeigte. Die schlechtesten Ergebnisse in Laufzeit und Fläche wurden durch keinerlei Zeitvorgaben hervorgerufen.

Dies läßt vermuten, daß bei dem verwendeten Synthesewerkzeug die Laufzeitvorgabe das dominante Optimierungsziel ist und alle übrigen Vorgaben im Rahmen der Zeitoptimierung berücksichtigt werden. Außerdem verwendet die Optimierung der Zeit offenbar eine Heuristik für die Erreichbarkeit der Zielvorgabe zur Abbruchsentscheidung. Die Auswahl der Optimierungsvorgaben und Optionen war hier also von entscheidender Bedeutung.

Die Konvertierung der RTL-Beschreibung in eine für die High-Level-Synthese verwendbare Form erforderte lediglich geringe Änderungen an der Synchronisation der Anweisungsprozesse und damit wenig zusätzlichen Entwurfsaufwand. Ohne eine solche Vorlage hätte die Entwicklung der Eingangsbeschreibung für die High-Level-Synthese jedoch ähnlich viel Zeit erfordert wie der RTL-Entwurf. Die Ergebnisse der Synthese zeigen jedoch drastische Unterschiede, bedingt durch die High-Level-Optimierungen des Datenflusses und der Operatoren sowie der Abbildung der Anweisungsfolge auf eine FSMD-Architektur (*finite state machine with datapath*). Das Spektrum der aus einer Beschreibung generierbaren Ergebnisse ist bei der High-Level-Synthese nicht nur wesentlich größer als bei der RTL-Synthese, sondern entspricht im Verhältnis von Verarbeitungsgeschwindigkeit und Schaltungsgröße grob dem Verlauf einer typischen Trade-Off-Kurve (**Bild 4**).

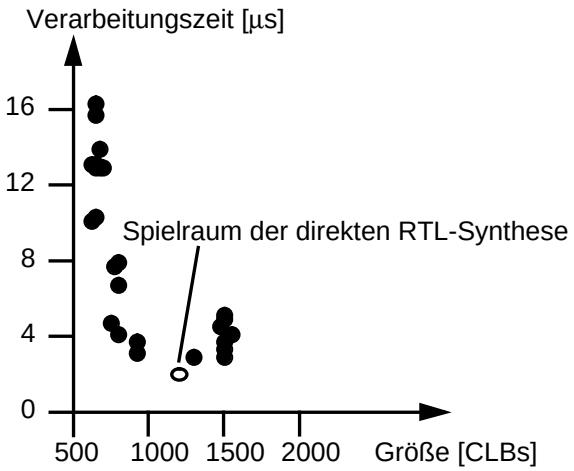

Bild 4 High-Level-Synthese einer IDEA-Stufe

Die Vorgabe von Zeitwerten für die Taktperiodenlänge und die maximale Verarbeitungszeit in Takten war als

Optimierungsziel sowohl bei den High-Level-Optimierungen als auch bei der nachfolgenden RTL-Synthese bestimmend für die Ergebnisstreuung. Da zwischen diesen Werten trotz getrennter Spezifikation ein Zusammenhang im Bezug auf die Verarbeitungsleistung der resultierenden Schaltung besteht, sollten sie sorgfältig untereinander und mit den übrigen Vorgaben und Optionen abgestimmt werden.
Die Ergebnisse der High-Level-Synthese blieben im Bereich der Verarbeitungsleistung zwar hinter denen der RTL-Synthese zurück, bieten aber sehr interessante Kompromißlösungen zwischen maximaler Verarbeitungsleistung und kleinstmöglicher Fläche. Ist eine minimale Größe der Schaltung das wichtigste Entwurfsziel, so kann die High-Level-Synthese in diesem Fall mit besseren Ergebnissen aufwarten, als die RTL-Synthese.

7 Zusammenfassung und Ausblick

Aus dem Vergleich charakteristischer Eigenschaften der drei betrachteten Synthesemethoden und den bei Entwurfsexperimenten gesammelten Erfahrungen [1] konnten verschiedene Kriterien zur Auswahl der je nach Entwurfsanforderungen zu bevorzugenden Synthesemethode abgeleitet werden.
So sollte der Entwurf mit Register-Transfer-Beschreibungen und RTL-Synthese implementiert werden, wenn das Entwurfsproblem eine der folgenden Anforderungen stellt:

- asynchrone Aktivitäten
- mehr als ein Taktsignal
- genaue Kontrolle der Schaltungsstruktur
- genaue Zeitvorgaben für Berechnungen
- genaue Bestimmung und Ansteuerung von Speicherelementen
- pegelgesteuerte Speicherelemente
- bidirektionale Kommunikation
- direkte Bezüge zu einer Zieltechnologie bzw. ihrer Baugruppen

Die Controllersynthese mit den speziell dafür ausgelegten Eingabewerkzeugen bietet sich für alle Entwürfe an, die keine derartigen Anforderungen stellen. Sie kann sehr effektiv für Entwürfe mit diesen Eigenschaften bzw. Erfordernissen angewendet werden:

- synchrone Logik mit nur einem Taktsignal
- kontrollflußdominierte Arbeitsweise
- komplexe oder hierarchische Zustandsverarbeitung
- kompakte und übersichtliche Entwurfsdarstellung
- komfortable und zügige Entwurfseingabe
- bestmögliche Codierung von FSMs

Die High-Level-Synthese von algorithmischen Beschreibungen ist den beiden vorgenannten Methoden deutlich überlegen, wenn die zu erstellende Schaltung über folgende Eigenschaften verfügen soll oder diese zuläßt:

- synchrone Verarbeitung auf der Basis nur eines Taktsignals
- datenflußdominierte Arbeitsweise
- mehrere arithmetische Operationen, insbesondere gleichartige
- einige Freiheiten in der zeitlichen Ausführung von Operationen
- die Beschreibung der Funktion in sequentiellen Anweisungsfolge
- großes Spektrum an Lösungsvarianten, ohne die Beschreibung zu verändern
- minimale Schaltungsgröße durch Mehrfachnutzung von Hardwarebaugruppen

Diese Kriterien stellen zwar keine vollständige Menge von Entscheidungsregeln für alle charakteristischen Eigenschaften von Synthesemethoden, Werkzeugen und Entwürfen dar, dürften aber die Entscheidung für eine geeignete Synthesemethode zur Lösung eines Entwurfsproblems wesentlich erleichtern. Sollte eine klare Entscheidung dennoch nicht möglich sein, so kann eine Partitionierung des Entwurfs und die Anwendung mehrerer Synthesemethoden in Kombination hilfreich sein.
Der vorliegende Beitrag beleuchtete nur einen kleines, aber bedeutendes Teilstück in dem stark verwobenen Netz der Entwurfsentscheidungen des synthesebasierten Schaltungsentwurfs. Die Wahl geeigneter Optimierungsoptionen und Zielvorgaben für jeden Syntheseschritt und die Partitionierung von Entwürfen für die verschiedenen Synthesemethoden sind eng verknüpfte Problemstellungen, die für das Erreichen bestmöglicher Syntheseergebnisse ebenfalls zu lösen sind [1]. Bei den präsentierten Syntheseexperimenten wurde dies daran deutlich, daß die RTL-Beschreibungen der Entwürfe die schnellsten Schaltungen hervorbrachten. Die Erfahrung des Designers führte hier zur Auswahl geeigneter Architekturen, die derartige Ergebnisse ermöglichten. Einem unerfahrenen Designer wären vergleichbare Lösungsansätze und damit ähnlich gute Ergebnisse vielleicht verborgen geblieben.
Weitere Untersuchungen der Möglichkeiten und Grenzen von Entwurfsbeschreibungen, Synthesemethoden und -optimierungen werden zum Aufbau einer breiten Entscheidungsgrundlage noch benötigt. Dies schließt neben weiteren Syntheseexperimenten auch die Einbeziehung anderer Synthesewerkzeuge ein.

8 Literatur

[1] Blinzer, P.; Möglichkeiten und Methoden der Schaltungssynthese; Dissertation, Technische Universität Braunschweig, voraussichtlich Herbst 1999.

[2] Holtmann, U. und Blinzer, P.; Design of a SPDIF Receiver using Protocol Compiler; 35th Design Automation Conference, Juni 1998.

[3] International Electrotechnical Commission; IEC 958, digital audio interface, First edition 1989-03; IEC, 1989.

[4] International Standardisation Organisation; ISO standard 9979 idea-tm(2); ISO, 1998.

[5] Michel, P., Lauther, U. und Duzy, P.; The Synthesis approach to digital system design; Kluwer Academic Publishers, 1992.

[6] Seawright, A. und Brewer, F.; Synthesis from Production-Based Specifications; 29th Design Automation Conference, Juni 1992.

[7] Seawright, A., Holtmann, U., Meyer, W., Pangrle, B., Verbrugghe, R. und Buck, J.; A System for Compiling and Debugging Structured Data Processing Controllers; Euro-DAC 1996.

[8] Synopsys; Behavior Compiler Guide; Version 1997.08; Synopsys Inc., 1997.

[9] Synopsys; Design Compiler Reference, Version 1997.08; Synopsys Inc., 1997.

[10] Synopsys; Protocol Compiler Reference, Version 1998.08; Synopsys Inc., 1998.