

Automatische Hardware-Implementierung digitaler Filter für einen Audio-Codec

Von MediaTek Inc.

DIE KONSTRUKTION EINER FILTERKETTE FÜR EINEN AUDIO-Codec erfordert eine sorgfältige Balance zwischen Leistung, Energieverbrauch und Chipgröße. Ingenieure von MediaTek sollten eine Lösung entwickeln, die nicht nur strikte Vorgaben bezüglich des Signal/Rausch-Verhältnisses (Signal-to-Noise Ratio, SNR) und des Klirrfaktors erfüllt, sondern auch möglichst wenig Strom verbraucht und eine geringe Chipfläche einnimmt.

Bei MediaTek wurden Systementwürfe bislang durch handgeschriebenen Register Transfer Level (RTL) Code implementiert. Mit dieser Methode erzielt man zwar relativ kleine Chipflächen, sie ist aber auch langwierig und bei geänderten Anforderungen muss die Implementierung unter Umständen aufwendig überarbeitet werden. Dieser Ansatz birgt zudem auch wirtschaftliche Risiken: Oft wissen die Entwickler bis zum Schluss des Entwicklungsprozesses nicht, wie schwer sich das Place-and-Route [die Anordnung und Verbindung der Komponenten auf dem Chip] gestalten wird. In dieser Phase ist es aber praktisch unmöglich, noch etwas zu verändern und trotzdem das Freigabedatum einzuhalten.

Die Ingenieure entschieden sich darum für eine neue Methode, bei der die Entwicklung in MATLAB® erfolgt und mit dem Filter Design HDL Coder™ synthetisierbarer RTL-Code generiert wird. Durch

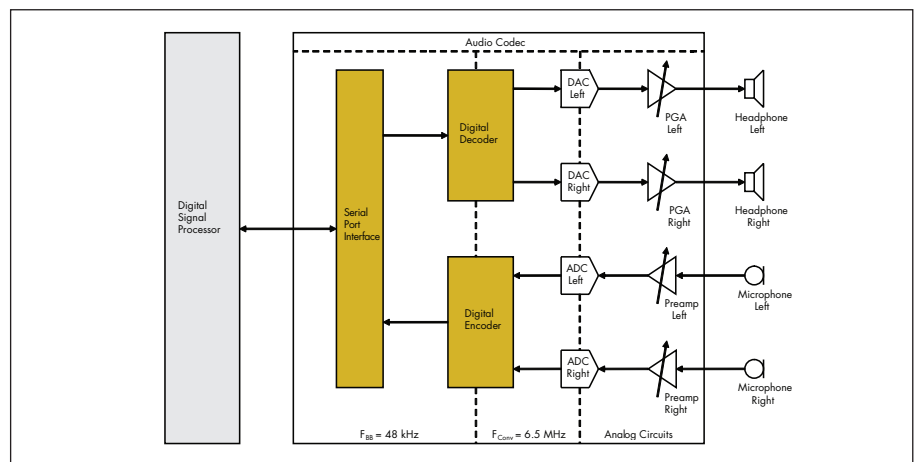


ABB. 1: High-Level-Blockdiagramm des Audio-Codecs.

diese direkte Verbindung von Systementwurf und Halbleiterentwicklung lassen sich in kurzer Zeit sowohl verschiedene Filterarchitekturen vergleichen als auch die Chipgröße optimieren. Die Entwick-

lung des RTL-Codes konnte so von drei Monaten auf weniger als zwei Wochen verkürzt werden. Modifikationen am System, die früher fast einen Monat dauerten, sind jetzt in nur drei Tagen möglich.

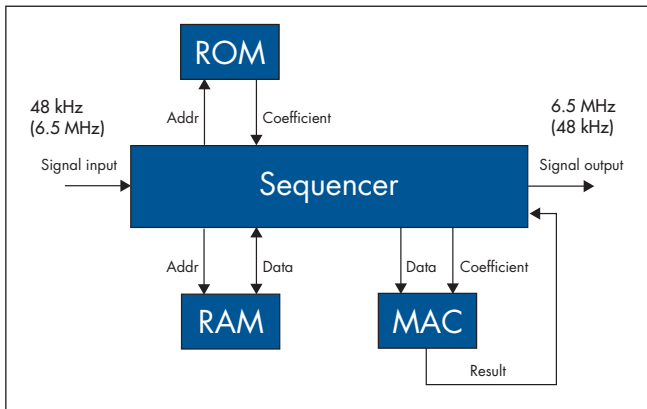


ABB. 2: Blockdiagramm eines konventionellen Audio-Codec.

Entwurf einer Codec-Architektur

Ein Audio-Codec besteht aus zwei separaten Verarbeitungsketten (Abb. 1). Der Audio-Encoder bildet die Schnittstelle vom Mikrofon zum digitalen Signal-Prozessor (DSP). Der Audio-Decoder arbeitet in der Gegenrichtung und wandelt vom DSP erzeugte Signale in Daten um, die an den Lautsprecher geschickt werden.

Eine zentrale technische Herausforderung der bei MediaTek entwickelten Systeme besteht darin, dass der hinter dem Mikrofon-Vorverstärker angesiedelte A/D-Wandler (ADC) sowie der vor den Lautsprecher-Verstärker geschaltete D/A-Wandler (DAC) mit der relativ hohen Frequenz von 6,5 MHz arbeiten müssen. Der DSP dagegen verarbeitet Daten mit einer einheitlichen Rate von 48 kHz. Der digitale Abschnitt des Audio-Codec wandelt diese beiden Frequenzen über eine Abfolge digitaler Filter wechselseitig ineinander um. Im Stereo-Encoderkanal beispielsweise dezimieren acht Filter das Eingangssignal, der Stereo-Decoder dagegen interpoliert sein Eingangssignal mit Hilfe von neun Filtern.

Während der Architektur-Entwicklung werden diese Kanäle mit MATLAB modelliert. Die Grundlage für die Modellierung des D/A- und A/D-Wandlers sowie zum Aufbau der verketteten digitalen Multiratenfilter bilden von den Analog-Designern gelieferte Parameter.

Die vom Modell erzeugten Ergebnisse werden mit MATLAB, der Signal Processing

Toolbox™ und der Filter Design Toolbox™ durch Berechnung von FFTs und die Schätzung von SNR und Klirrfaktor überprüft. Die Ingenieure können so die Architektur optimieren, bevor sie mit der Implementierung beginnen.

Implementierung auf Basis eines FSM-Sequencers

Beim vorhergehenden Entwicklungsprozess wurde die Signalverarbeitungs-Kette von einem einzelnen Multiplikator-Akkumulator (MAC) und einem als endlicher Zustandsautomat (Finite State Machine, FSM) implementierten Sequencer gebildet (Abb. 2). Der Sequencer ist verantwortlich für die Adressierung von RAM und ROM und die Steuerung ihrer Operationen. Außerdem legt er Eingabeargumente für den MAC fest. Das ROM speichert die Filterkoeffizienten, das RAM die Zwischenergebnisse der Multiplikation/Akkumulation und der MAC führt Berechnungen auf der Basis der Koeffizienten und Datensamples durch.

Ursprünglich wurde dieser Sequencer mit Stift und Papier entwickelt, was kompliziert und zeitaufwendig war. Der handgeschriebene RTL-Code war so unflexibel, dass sich auch kleine Veränderungen schwierig gestalteten. In einem Fall erwies sich das Place-and-Route des RTL-Codes für den Zustandsautomaten des Sequencers sogar als fast unmöglich. Dieser Zustandsautomat enthielt etwa 2000 Zustände mit über 40 Variablen. Das erste Place-and-Route der Gate-Level-Netlist ergab eine Flächen-

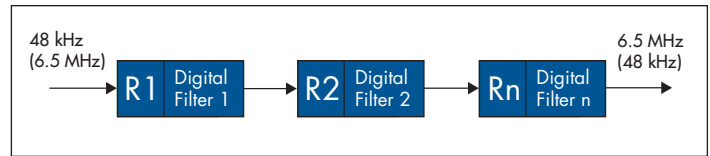


ABB. 3: Die neue Entwurfsarchitektur. Jeder Filter wurde eigenständig mit dem Filter Design HDL Coder implementiert. R1, R2 und Rn stehen für die Wandlung der Datenrate (Dezimation/Interpolation) des ersten, zweiten und n-ten Filters der Kette.

nutzung von lediglich 10%. Die benötigte Siliziumfläche war also zehnmal so groß wie veranschlagt. Mit großem Aufwand wurde die Sequencer-Logik in ein Custom ROM synthetisiert. Auch danach hatte der Sequencer noch einen zwei Mal höheren Flächenbedarf als zuvor für die gesamte kombinatorische Logik veranschlagt worden war (ein Custom ROM benötigt in der Regel mehr Chipfläche).

Automatisierung der Implementierung

Bei der neuen Entwicklungsmethode kann auf den komplexen Sequencer verzichtet werden. An seine Stelle tritt eine Reihe digitaler Filter, die mit MATLAB und der Filter Design Toolbox entwickelt und dann mit dem Filter Design HDL Coder implementiert werden (Abb. 3). Jeder digitale Filterblock arbeitet eigenständig, es können also problemlos Blöcke verändert, entfernt oder neu in die Gesamtkette eingefügt werden.

Die Decoder-Kette enthält vier Halbband-FIR-Filter und einen Abtastratenwandler. Jeder Filter interpoliert um den Faktor zwei, wodurch das Ausgangssignal von 48 KHz auf 96 kHz, 192 kHz, 384 kHz und schließlich 768 kHz gebracht wird. Ein einziger Abtastratenwandler sorgt dann für die letzte Umwandlung in die Zielfrequenz von 6,5 MHz.

Die Encoder-Kette besteht aus zwei Cascaded-Integrator-Comb (CIC) Dezimationsfiltern sowie zwei Halbband-FIR-Filtern. Der erste CIC-Dezimationsfilter reduziert die Eingangsrate von 6,5 MHz um den Faktor 25 auf 260 kHz. Auf ihn folgt ein einfacher, handgeschriebener Zero-Stuffer, der das Signal durch

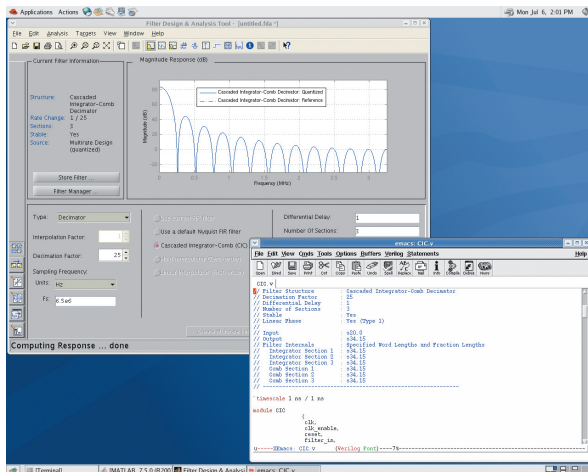


ABB. 4: Das GUI des Filter Design HDL Coder mit einem Teil der Filterarchitektur sowie einem Abschnitt des generierten Codes.

Einfügen von Nullen um den Faktor 48 interpoliert. Die interpolierten Daten haben eine Frequenz von 12,48 MHz und werden von einem zweiten CIC um den Faktor 65 dezimiert. Im ersten Halbband-FIR-Dezimationsfilter werden sie mit der resultierenden Datenrate von 192 kHz verarbeitet und um zwei dezimiert. Der zweite Halbband-Dezimationsfilter halbiert die dabei erzeugten 96 kHz auf die endgültige Datenrate von 48 kHz. Der Entwurf des CIC für den Encoderkanal wurde durch die Filter Design Toolbox erleichtert, mit der eine ganze Reihe von Optionen, etwa die Zahl der verwendeten Bits, in sehr kurzer Zeit miteinander verglichen werden konnten.

Generierung von RTL-Code und Optimierung der Chipfläche

Nach dem Entwurf der einzelnen Filter der Encoder- und Decoderketten wurde für jeden Filter Verilog®-Code mit dem Filter Design HDL Coder generiert (Abb. 4). In diesem Stadium konnten die Ingenieure mit der Optimierung der Implementierung für eine möglichst kleine Chipfläche beginnen.

Für die RTL-Generierung experimentierte das Team mit verschiedenen Optimierungs- und Architektur-Optionen des Filter Design HDL Coders. Zur Optimierung der Halbband-FIR-Filter wurde beispielsweise zuerst eine Option mit vollkommen paralleler Architektur

getestet, bei der Filter- und Datenrate identisch sind. Mit dem Synopsys® Design Compiler wurde der Code für diese Variante synthetisiert und die erforderliche Fläche ermittelt. Die Distributed Arithmetic-Option des Filter Design HDL Coder, bei der die Filterratenrate 16 bis 20-mal so hoch ist wie die Datenrate, erzeugte im Vergleich einen Entwurf, der nur etwa ein Viertel der Fläche des vollständig parallelen Entwurfs benötigte. Da auf dem Chip bereits eine Clock mit ausreichender Frequenz vorhanden war, war dies die beste Wahl.

Verifikation der Implementierung

Die RTL-Implementierung wurde nun mithilfe von MATLAB-Skripten, RTL-Testbenches und Verilog-Simulationen verifiziert. Viele der in der Architekturphase entwickelten MATLAB-Skripte wurden dabei wiederverwendet und zur Erzeugung von Anregungssignalen sowie zur Nachbearbeitung der Ergebnisse durch FFTs und die Schätzung von SNR und Klirrfaktor eingesetzt.

Nach dieser ersten Verifikationsphase wurde der RTL-Code zur Synthese an das nächste Team übergeben. Während des Place-and-Route-Schritts einer Synthese werden Timing-Daten in Form einer SDF- (Standard Delay Format) Datei erzeugt. Diese Timing-Daten wurden in einer weiteren Runde von Gate-Level-Simulationen eingesetzt um

sicherzustellen, dass es in der Logik keine Wettlaufsituation gab. Der Entwurf wurde dann an die Halbleiterfertigung übergeben.

Tests des ersten fertigen Chips deckten keinerlei Fehler im Digitalteil des Audio-Coders auf. Die Ingenieure hatten dadurch jetzt die Zeit, die Analogentwickler bei deren Arbeit und den verbleibenden Testaktivitäten zu unterstützen.

Einfluss aktueller Herstellungsprozesse

Durch veränderte Herstellungsverfahren verschieben sich auf aktuellen Chips die Flächenanteile von analogen und digitalen Komponenten. Während analoge Transistoren eine bestimmte Größe haben müssen, um die nötige Leistung zu erzielen, schrumpfen die digitalen Transistoren mit jedem neuen Fabrikationsprozess. War bei der 250 nm-Fertigung noch jeweils die Hälfte der Chipfläche mit digitalen und analogen Komponenten belegt, so sind es beim aktuellen 45 nm-Prozess noch 25% Digitalanteil und dafür 75% Analoganteil.

Der neu eingeführte Entwicklungsprozess produziert zwar größere Chipflächen als die konventionelle Methode, aber dieser Effekt wird durch die fortschreitende Miniaturisierung digitaler Komponenten kompensiert und die Zeitersparnis ist erheblich.

In künftigen Projekten will MediaTek die Chipfläche durch Nutzung anderer vom Filter Design HDL Coder angebotener platzsparender Architektur-Optionen weiter optimieren, etwa solche mit teilweise serieller Struktur. Eine andere Variante ist, die Sequencer-Architektur mittels automatischer Codegenerierung zu implementieren. ■

Quellen

MEDIATEK
www.mediatek.com

MATHWORKS-PRODUKTE FÜR DIE SIGNALVERARBEITUNG
www.mathworks.de/nn9/dsp