Einsatz von IP für FIR-Filter-Design

Effektive PLD-Implementierung in Kommunikations-Testgerät

JEFF FOX, CHARLIE EVANS

Indem Designs immer komplexer und größer werden, nimmt auch der Einsatz von wiederverwendbaren IP-Funktionsblöcken (Intellectual Property) zu. Dieser Beitrag beschreibt ein Design, das mit parametrisierten, wiederverwendbaren IPs realisiert wurde. Dabei war die Evaluierung des entsprechenden IP-Cores noch vor der Kaufentscheidung wichtig, da so das Design-Risiko und die Kosten reduziert werden konnten. Das Design-Beispiel ist ein digitaler FIR-Filter als Teil einer TIMS-Linecard (Transmission Impairment Measurement Set), die in einem Test-Gerät zur Überwachung von Kommunikationsdiensten eingesetzt wird. Das Handheld-Gerät von Hewlett Packard ist speziell für den Test von Kommunikationsgeräten in den Bereichen ATM, SONET und xDSL ausgelegt.

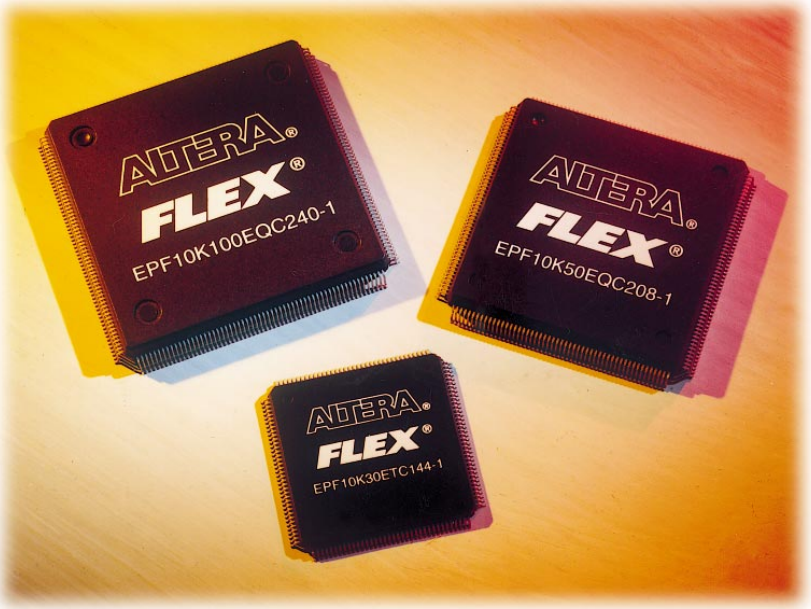

Bild 1: Unter Nutzung parametrisierbarer IP-Cores lassen sich in PLDs wie den hier gezeigten FLEX-Bausteinen Filter effektiv implementieren

In vielen digitalen Systemen werden Filter zur Reduzierung von unerwünschtem Rauschen, zur Dämpfung oder Verstärkung bestimmter Frequenzkomponenten, zur spektralen Aufteilung oder zur Signalerfassung bzw.-analyse eingesetzt. Dabei gibt es prinzipiell zwei Arten von Filtern: FIR-Filter (Finite Impuls Response) und IIR-Filter (Infinite Impuls Response). FIR-Filter werden in Systemen eingesetzt, die eine exakte lineare Phasenabhängigkeit erfordern. Dies kann – abgesehen von einigen speziellen Ausnahmen – mit IIR-Filtern nicht erreicht werden. Im Gegensatz zu IIR-Filter haben FIR-Filter auch den Vorteil, dass sie eine stabile Frequenzabhängigkeit haben – allerdings bei möglichen zusätzlichen Hardware-Kosten gegenüber einem IIR-Filter. IIR-Filter, die nicht inherent stabil sind, können für Systeme in Betracht gezogen werden, die ihre Phasenverzögerung tolerieren können.
Die Struktur eines FIR-Filters basiert auf einer gewichteten, stufenförmigen Verzögerung. Das Filter-Design erfordert die Auswahl von Komponenten, die die Frequenzanforderungen des Systems erfüllen. Die Koeffizienten bestimmen den Frequenzgang des Filters. Durch Änderung der Koeffizientenwerte kann bestimmt werden, welche Signalfrequenzen das Filter passieren. Der erste Schritt bei der Entwicklung eines FIR-Filters ist die Definition des gewünschten Frequenzganges, der Abtastrate und der erforderlichen Genauigkeit, abhängig von den Systemanforderungen. Wenn dies definiert ist, müssen die Koeffizienten berechnet werden, die in das FIR-Filter implementiert die Filter-Charakteristik bestimmen.

Bei der Kalkulation der Koeffizienten sollte man schrittweise vorgehen:

1. Spezifikation des gewünschten Frequenzganges.
2. Spezifische Implementierungsbedingungen für die Realisierung des Filters in Hardware. Beispielsweise soll das Filter eine begrenzte Anzahl von Taps, eine begrenzte Genauigkeit (Word-Breite der Daten und Koeffizienten) und reale (nicht komplexe) Koeffizienten haben.
3. Bildung der Inversen Diskreten Fourier Transformation (IDFT), um die Filter Impulscharakterstik zu erhalten. Die Werte für die Impuls-Antwort sind die FIR-Filter-Koeffizienten.
4. Gewichtung der Koeffizienten, indem man die Impulsanwort mit einer „Windowing"-Funktion multipliziert, um die Störungen zu reduzieren.

Nach diesen Schritten kann man an die Implementierung des Filters gehen und verifizieren, ob die erwünschte Frequenzcharaktersitik erreicht wurde.
Interpolation und Dezimierung sind übliche Verfahren bei der Implementierung von FIR-Filtern. Bei der Interpolation werden zusätzliche Abtastpunkte hinzugefügt, um die Abtastrate zu erhöhen, während bei der Dezimierung unnötige Abtastpunkte entfernt werden.
Das in diesem Beitrag beschriebene System erforderte ein Low-Pass-FIR-Filter mit Dezimierung. Die Dezimierung ermöglicht es, einen einfacheren analo-

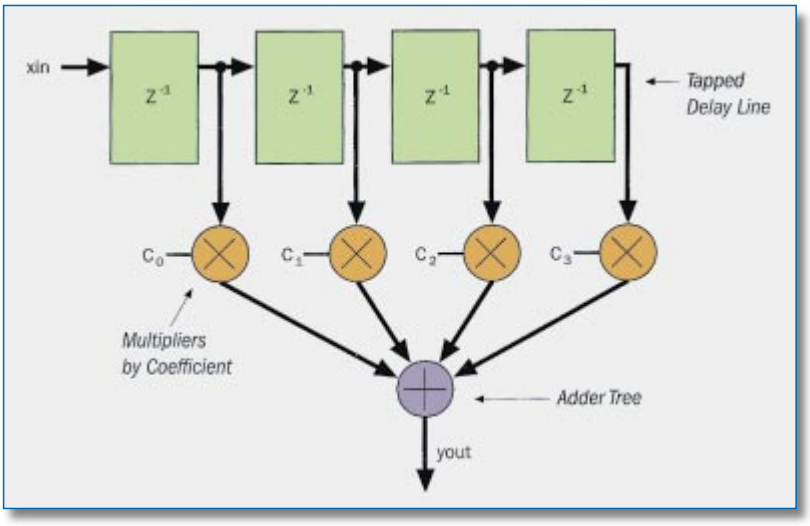

Bild 2: Grundschema eines FIR-Filters

gen Anti-Aliasing-Filter (Low-Pass) einzusetzen. Durch die Dezimierung kann die Abtastrate erhöht werden, wobei anschließend die zusätzlichen Abtastungen wieder entfernt werden, um eine dem System entsprechende Rate zu realisieren.
Es gibt verschiedene Architekturen, um FIR-Filter zu implementieren – jede mit eigener Charakteristik. Prinzipiell kann man zwischen parallelen und seriellen Strukturen unterscheiden. Bei einer parallelen Implementierung führt der Filter eine Berechnung in einem einzigen Taktzyklus aus. Die Breite eines jeden Speicherelementes in einer stufenförmigen Verzögerung entspricht der Datenbreite (W_{data}). Es gibt also N W_{data}-breite Speicherelemente, wobei N die Anzahl der Stufen ist. Parallele Filter bieten die höchste Leistungsfähigkeit, benötigen aber auch die größte Fläche, da sie N Multiplizierer erfordern, wobei jeder Multiplizierer W_{data} x W_{coeff} ist.
Eine serielle Implementierung ist kleiner, aber langsamer als ein paralleler FIR-Filter. Er erfordert zwar die gleiche Anzahl an Speicherelementen und Multiplizierer, aber jeder Multiplizierer ist nur 1 bit x W_{coeff} groß. Für jeden Ausgangswert sind W_{data} Taktzyklen erforderlich.

System-Aufbau

Der FIR-Filter wurde in diesem Design in einem FLEX 10K100 realisiert. Dieser programmierbare Logikbaustein befindet sich in einem BGA mit 356 Anschlüssen. Das PLD hat Interfaces zu einem OnBoard-DSP, einem AD-Wandler und einem Systemprozessor. Die TIMS-Module in dem Service-Advisor-Test-Tablett nutzen verschieden Abtastraten mit entsprechenden anlogen Anti-Aliasing-Filtern, um Frequenzen in dem Bereich von 50 Hz bis 2 MHz genau testen und analysieren zu können. Das PLD ist das Herz des Messsystems – es generiert die Abtast-Taktrate, wählt die entsprechenden Analogfilter aus und erledigt auch noch andere Aufgaben.
Ein wichtiger Grund für die Implementierung des FIR-Filters in ein PLD war der Fakt, dass die neuesten Testfunktionen es erforderten, dass der DSP in den TIMS-Modulen eine Abtastrate verwendet, die achtmal langsamer als die langsamste verfügbare Abtastrate war. Es war eine Lösung erforderlich, die es HP erlaubte, existierende analoge Anti-Aliasing-Filter einzusetzen, um Zeit zu sparen und das Design-Risiko zu verringern. Es wurde ein Low-Pass-FIR-Filter (mit Dezimierung) als beste Lösung ausgewählt. Allerdings kann ein DSP diese Funktion nicht schnell genug ausführen und andere kritische Messfunktionen gleichzeitig durchführen. Da der FLEX 10K-Baustein von Altera bereits für das Interface und die Steuerung des AD-Wandlers eingesetzt wurde, entschloss man sich, auch das FIR-Filter direkt in den FLEX10K-Baustein zu integieren.
An dieser Stelle soll angemerkt werden, dass DSPs eine begrenzte Zahl von MAC-Einheiten (Multiplizierer-Akkumulator) haben, die viele Taktzyklen für die Bearbeitung eines Wertes benötigen. Dadurch wird die Leistungsfähigkeit des Filters beschränkt. Implementiert man dagegen ein paralleles FIR-Filter mit Pipeline in einem PLD, so kann eine Datenrate von über 100 MS/s realisiert werden - im Vergleich zu 3 MS/s für einen typischen DSP.

Design-Prozess

Die traditionelle Design-Methode für einen Funktionsblock in einem System besteht aus der Spezifizierungs- und der Hardware-Implementierungs-Phase.

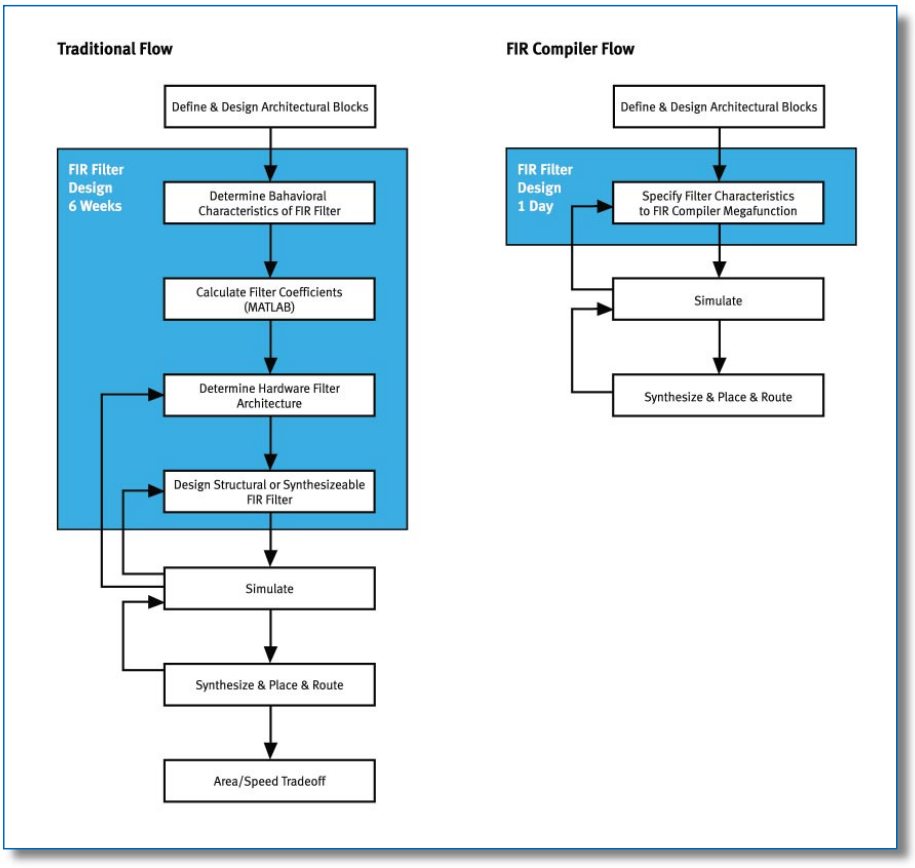

Bild 3: Vergleich traditioneller und FIR-Compiler-Designfluss

Beide Phasen erfordern mehrmalige Änderungen und Anpassungen und dementsprechend viel Zeit. Man entschloss sich daher, den FIR-Compiler von Altera einzusetzen, um schneller an bessere Resultate zu kommen. Das OpenCore-Evaluierungssystem von Altera reduzierte die Design-Risiken und beschleunigte den Entwicklungsprozess, indem das komplette Filter-Design noch vor der Lizenzierung des IP-Kernes evaluiert werden konnte.
Bei dem neuen Entwicklungsmodell werden funktionale Blöcke (IP-Cores) mit Hilfe des MegaWizard-Systems von Altera kundenspezifisch konfiguriert und dann im System evaluiert, bevor der Core lizenziert wird.
Die MegaWizards-Plug-Ins sind GUI-basierende Tools, die helfen die IP-Cores (Megafunktionen) zu parametrisieren und in die Designs zu integrieren. Die DSP MegaWizard Plug-Ins arbeiten zusammen mit der Matlab- und Simulink-Software von *The Math Works*, um das komplette System-Level-Design bzw. die System-Implementierung zu ermöglichen.
Mit dem FIR-MegaWizard wird automatisch eine optimierte Implementierung eines spezifizierten Filters erzeugt. Dabei werden verschiedene Alternativen unterstützt. Generell erzeugt MegaWizard Matlab-Modelle für die System-Level-Verifizierung und VHDL- sowie Verilog-Modelle für die RTL-Simulation.
Mit OpenCore konnte man das Verhalten und die Leistung des FIR-Filters vollständig evaluieren. Durch Einsatz der existierenden Tools konnte man bei HP das System vollständig nachbilden und testen – einschließlich der Compilierung, Simulation und Verifizierung – noch vor dem IP-Kauf. Durch die Evaluierung wurden alle erforderlichen Funktions-, Timing- und Platzbedarfs-Informationen bereitgestellt, um die Lizenzierungs-Entscheidung zu erleichtern. Wenn eine Funktion lizenziert ist, dann können mit Hilfe der MAX+PLUS II oder Quartus-Software Programmier-Files sowie EDIF-, VHDL- oder Verilog HDL-Netzlisten-Files für die Post-Layout-Simulation in EDA-Tools erzeugt werden.

Der FIR-Filter-Entwurf

Mit dem FIR-Compiler konnten die Filter-Parameter kundenspezifisch verändert werden, um der Applikation bestmöglich zu entsprechen. Der Compiler erzeugte eine kundenspezifische Ausprägung der Megafunktion, die dann direkt in das Design eingebracht wurde.
Zuerst wurde der FIR-Filter erzeugt, indem der MegaWizard Plug-in-Manager gestartet wurde. Über das GUI-Interface wurde eine neue kundenspezifische Megafunktion spezifiziert. Von der DSP Megacore Liste wurde dann die FIR-Compiler-Funktion ausgewählt und mit einem entsprechend File-Namen versehen.
Alle Spezifikationen auf System-Level beginnen mit der Auswahl des gewünschten Frequenzganges für eine gegebene Abtastrate. Für FIR-Filter-Designs wird allgemein eine lineare Phase ausgewählt. Nach der Erstellung dieser zwei Parameter generiert der FIR-Compiler automatisch einen Koeffizienten-Satz, um die gewünschten Ergebnisse zu erreichen. Um ein besseres Filter zu erzeugen, lässt man normalerweise die Koeffizienten über eine mathematische Funktion laufen. Jedes dieser mathematischen Fenster (Windows) hat eine spezielle Charakteristik. Um ein Window zu erzeugen, ist nur ein Knopfdruck notwendig. Neben dem Windows-Typus wird auch die Abtastrate, die Anzahl der Stufen (Taps), die Eingangsdaten, etc. spezifiziert. Wenn mehr Koeffizienten (oder Taps) erforderlich sind, können sie einfach per Knopfdruck hinzugefügt werden.
Sind alle Design-Paramater spezifiziert, gibt der MegaWizard automatisch den Frequenzgang und die Koeffizientenwerte aus.
Die Koeffizienten konnten von einen System-Level-Tool wie Matlab direkt in MegaWizard importiert werden. Das Wizard-Tool kann Festkomma- oder Fließkomma-Koeffizienten aus einem einfachen ASCII-Text-File lesen. In diesem Fall wurden die Koeffizienten direkt von einem DSP-Tool importiert.
Mit MegaWizard konnten die Koeffizienten so wie sie waren oder auch skaliert genutzt werden. Die Koeffizienten werden in der „Scalable and Rounded Box“ ▶

Baustein	Parameter	Genutzte LEs	Genutzte EABs	f_{max}/MHz
Flex 10KE-1	17 TAPs, parallel	879	0	82
	19 TAPs, parallel	1260	0	101
	79 TAPs, parallel	761	5	69

***Tabelle 1*: Ressourcen-Bedarf für verschiedene Filter**

dargestellt. Der Wizard entdeckt jede Symmetrie automatisch und wählt die optimale Filter-Architektur aus.
Im nächsten Schritt musste festgelegt werden, ob man die ganze oder eine limitierte Genauigkeit für den Filter-Ausgang wählt. Der FIR-Compiler bestimmt automatisch die Ausgangs-Bitbreite auf Basis der aktuellen Koeffizientenwerte und der Eingangs-Bitbreite. Diese zwei Parameter bestimmen die maximalen positiven und negativen Ausgangswerte. Für die volle Genauigkeit müsste die Anzahl der Bit des Systems genommen werden. Für eine limitierte Genauigkeit bot der Wizard zwei Möglichkeiten, um entweder das MSB (Most Significant Bit) zu beschneiden bzw. zu saturieren oder das LSB (Least Significant Bit) zu runden bzw. zu beschneiden. Danach wurde die Architektur des Filters ausgewählt (ob parallel oder seriell, mit oder ohne Pipelining, Anzahl der Eingangskanäle). In **Tabelle 1** sind die Auswahlmöglichkeiten für verschiedene Architektur-Optionen dargestellt. In der „Resource Usage Box" wird die geschätze Größe angezeigt, entweder in EABs oder in Logikzellen, sowie die Anzahl der erforderlichen Taktzyklen, um eine FIR-Berechung durchzuführen.
Hier ist zu erwähnen, dass die Anzahl der Taps direkt die Ressource-Nutzung beeinflusst. Mit MegaWizard kann man die Anzahl der Taps reduzieren und danach den entsprechenden Filtergang dahingehend prüfen, ob er noch immer die Spezifikation erfüllt. Damit bietet MegaWizard eine iterative Möglichkeit, die Filter-Performance zu optimieren. Mit MegaWizard und der OpenCore-Evaluation konnte das Filter-Design mit sehr geringem Ressourcen-Bedarf realisiert werden.

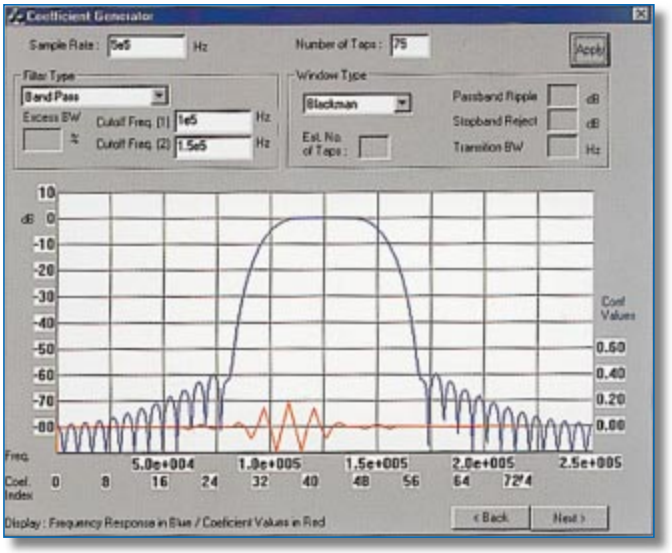

***Bild 4**: Einfache Filter-Parametrisierung mit dem Mega-Wizard-Tool*

System-Implementierung

Der FIR-Filter-Wizard generiert VH DL- und Verlog-Files für die RTL-Level-Simulation mit EDA-Tools von Cadence, Exemplar Logic, Mentor Graphics, Synopsys, Synplicity und Viewlogic. Die Altera-Entwicklungstools MAX+PLUS II und Quartus haben Interfaces zu allen gängigen EDA-Tools. Nach der System-Simulation mit den entsprechenden Ergebnissen konnte das System implementiert werden. Die Compilierung erfolgte einfach per Knopfdruck. Nach der Compilierung wurde das komplette Design mit Hilfe des MAX PLUS II Simulators bezüglich des Timings verifiziert. Nach der Lizenzierung der Megacore-Funktion wurde ein FLEX-Baustein programmiert und der korrekte Betrieb in der Hardware verifiziert. Die Hardware-Implementierung arbeitete entsprechend der Simulation korrekt. (la)

***Jeff Fox** ist Director MegaCore Engineering bei Altera, **Charlie Evans** ist Hardware Engineer bei der STD Division von Hewlett Packard.*