

Von der Spezifikation schnell zu Gattern

Mit C auf der Überholspur

Was tun, wenn die existierende Tool-Kette für das Chipdesign noch schneller arbeiten muss? Schneller gilt hier in beiden Bedeutungen: zum einen, um mit möglichst kurzer Designzeit von der Produktspezifikation über die Entwicklung zur Produktion zu gelangen – und das bei immer mehr Features und größerer Designkomplexität. Oft gilt schneller auch in Bezug auf die Ausführungsgeschwindigkeit bestehender Funktionen, um beim Ablauf Platz für neue Features zu schaffen. Hält man die Größe des Entwicklungsteams konstant, führen nur neue Werkzeuge zu besserer Designproduktivität.

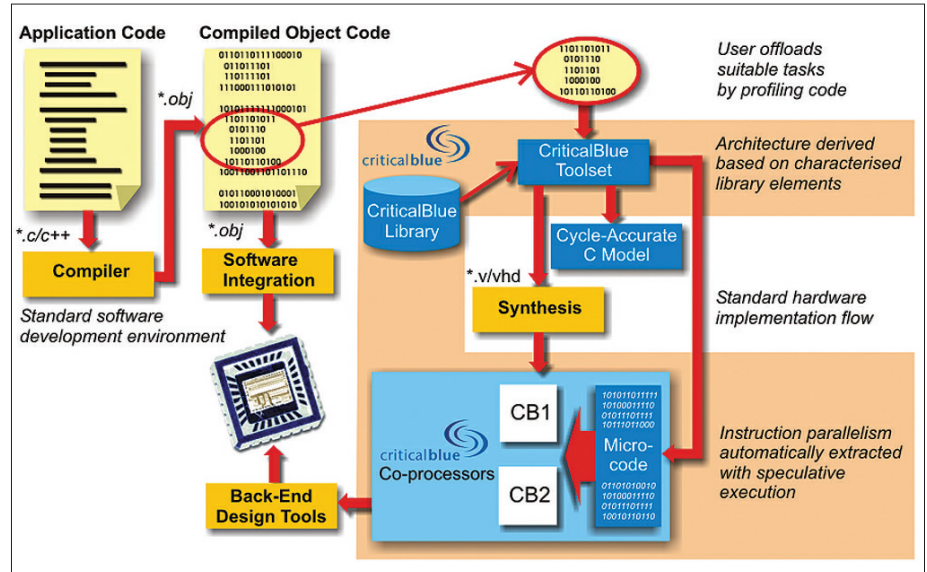


Bild 1. Cascade generiert aus kompiliertem C-Code den entsprechenden VHDL-Code für den Coprozessor (Bild: Critical Blue)

Das Beispiel des Rennwagens macht es klar: Auf das richtige Getriebe kommt es an, um die gewünschte Geschwindigkeit zu erreichen, denn die Drehzahl des Motors ist begrenzt – unabhängig von der PS-Zahl. Diese neuen Tools müssen auch direkt am Beginn des Zyklus greifen, um von Anfang an »Leerlauf« oder »zuviel Gewicht« zu vermeiden. Vom Systementwickler über die Softwareabteilung bis zum Hardwaredesigner müssen alle zusammen in den Entwicklungsprozess eingebunden werden: von der Spezifikation über Simulation und Verifikation bis zur Integration von Soft- und Hardware.

Der wichtigste Punkt besteht sicherlich darin, die Zeit bis zum ersten Prototypen auf ein Minimum zu reduzieren. Der erste Test einer neuen Software, die auf neuer Hardware läuft, ist der kritischste Zeitpunkt, denn jetzt kommen die Entwicklungen der meist unabhängigen Software- und Hardwaregruppe zusammen, und übergreifende komplexe Fehler tauchen auf. Da dies meist erst spät im Entwicklungszyklus geschieht, ist dann die Korrekturschleife lang und teuer. Eine Alternative besteht darin, möglichst schnell von der ausführbaren C-Simulation in Software zu einer schnellen Aufteilung in Hard- und Software zu gelangen und das System ohne Umcodierung ganz oder teilweise auf einem FPGA ablaufen zu lassen. Damit bleibt alles in C, Korrekturen sind in einer kurzen Schleife schnell gemacht und das Design ist rasch neu verifiziert.

Der optimale Designflow, der zum Beispiel mit C beginnt und für den gesamten Prozess gleich bleibt, besteht damit theoretisch aus nur einem Tool, sowohl für Software als auch für Hardware. Da die unterschiedlichen Anforderungen der Applikationen dies leider unmöglich machen, bleibt als bester Ansatz ein paralleler Weg zum schnellen Prototypen, um dem gesamten Team Fehler sofort auf-

zuzeigen, die dann als Teil des bewährten Design-Flows korrigiert werden. Es wird in schnellen Iterationsschritten entwickelt und damit werden die Korrekturkosten minimiert.

Aus dem breiten Angebot werden hier drei Werkzeuge herausgegriffen, die von Code in C ausgehen, aber für verschiedene Bereiche optimiert sind: Handel-C von Celoxica, Bach von Sharp (bald integriert in die Tools von Mentor Graphics) und als drittes Critical Blues Cascade, das kompilierten Code als Startpunkt nimmt, mit dem Ziel, einen separaten Coprozessor als Applikationsbeschleuniger zu generieren. Alle drei Tools gehen von C aus und generieren daraus Gatterbeschreibungen, die entweder direkt in ein FPGA geladen werden können oder VHDL erzeugen, das dann auch wieder für FPGAs, ASICs oder SoCs verwendet wird. Als offensichtlicher Vorteil führt dieser Ansatz mit den bestehenden Entwicklungsressourcen zu schneller getesteter/korrigierter Code. Da die Entwicklungsumgebung so lange wie möglich in C bleibt, sind schnelle Iterationen möglich. C muss nicht manuell in VHDL

übersetzt werden, um Gatter für den Test im FPGA zu erhalten, und vermeidet Fehlerquellen. Ein wichtiger Aspekt für die entstandene Hardware ist sicher die resultierende Gattereffizienz, aber das gilt nicht bei der Prototypenphase im FPGA. Hier sind nur zwei Punkte wichtig: Passt der Funktionsumfang in das FPGA und ist der Ablauf schnell genug.

Für ein ASIC geht die Gatterzahl in die Produktionskosten ein, aber auch hier gilt: Wieviel schneller ist das Endprodukt bei nicht optimaler Gatterzahl im ASIC am Markt; Verbesserungen sparen erst dann Geld, wenn die Volumenproduktion anläuft und eventuell auch aus anderen Gründen Korrekturen erforderlich sind. Für einen Coprozessoranatz gelten für Gatterzahl und Geschwindigkeit andere Anforderungen: Hier ist der »Flaschenhals« die Zeit beim Datentransfer über den Bus und nicht die Verarbeitung im Coprozessor selbst.

Bei vielen Projekten wird die Spezifikation zuerst, zur Simulation und zum Test (Bild 1) als C-Programm implementiert. Ein schneller Weg zur Implementierung der C-Funktion im FPGA ergibt eine

par	definiert den parallelen Ablauf der folgenden Funktionen
delay	verzögert die Daten um einen Taktzyklus
chan x	definiert Kanäle zur Kommunikation zwischen Blöcken
?	liest die Daten des entsprechenden Kanals
!	schreibt Daten an den jeweiligen Kanal
seq	definiert den Code für sequenzielle Ausführung
ram/rom	definiert verschiedene Speicherarten in Breite und Länge
interface	Verbindungen festlegen zu externer HW wie RAM/ROM/IO
width	Datenbreite festlegen von 1=Dracht zum n-Bit Bus
pragma	Festlegen eines bestimmten Zeitverhaltens

Tabelle 1: die wichtigsten Handel-C-Konstrukte

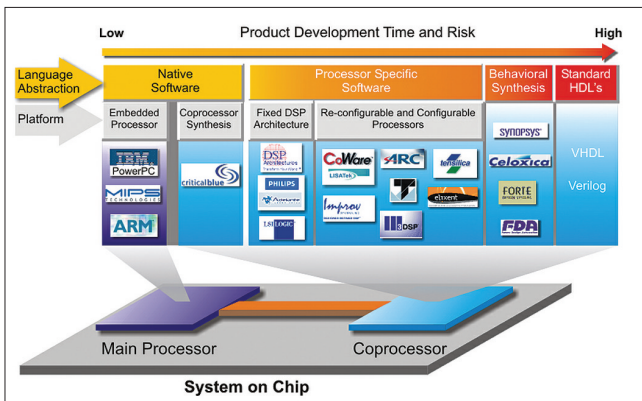


Bild 2. Critical Blues Sicht des Markts von der reinen Softwarelösung bis zu VHDL/Verilog (Bild: Critical Blue)

mehr als 1000-fache Simulationsgeschwindigkeit und findet damit Fehler vielfach schneller, erhöht die Iterationsrate im Vergleich zu reinen Softwarelösungen dramatisch und zeigt Gatteraufwand und Ablaufgeschwindigkeit. Die Algorithmenverifikation ist ein weiteres Beispiel, besonders wenn die Verarbeitung von Multimediafunktionen wie Ton (zum Beispiel MP3) und Bild (zum Beispiel Kompression) mitspielen. Die Ablaufzeit im FPGA ist um ein Vielfaches verkürzt oder erreicht eventuell sogar Echtzeit. Ist die gesamte Applikation simuliert und verifiziert, muss in Hard- und Software partitioniert werden. Es sollte soviel wie möglich auf dem Prozessor ablaufen, um die vorhandenen Ressourcen Prozessor, RAM/ROM zu möglichst 100 Prozent auszunutzen und zusätzliche Hardware als FPGA oder ASIC zu minimieren. Da viele Funktionen bei diesem Ablauf zusammen von Soft- und Hardwaredesignern verifiziert wurden, sind die erforderlichen Ressourcen in Hard- oder Software bereits bekannt und der Integrationsprozess in den endgültigen Prototypen gestaltet sich problemloser. Celoxicas Handel-C ist sicherlich die bekannteste Möglichkeit, um C-Code zum Ablauf in Hardware für FPGAs/ASICs zu übersetzen beziehungsweise für die Cosimulation und Coverifikation von Hardware und Software einzusetzen. Das Entwicklungs-Kit DK ist seit 2001 auf dem Markt, eine kostengünstige FPGA-Version gibt es seit kurzem. Aber wie funktioniert das Ganze? Die Grundlage ist CSP (communicating sequential processes). Das hört sich kompliziert an, bedeutet aber lediglich eine bestimmte Art, unabhängige Programmteile problemlos miteinander kommunizieren zu lassen. Implementiert damals in OCCAM für Multiprozessor-Transputersysteme, konnten auf verschiedenen Prozessoren implementierte Programmteile automatisch zusammenarbeiten. Es gab bei jedem Programmteil »Eingänge«, »Ausgänge« und »Verbindungen«. Derselbe Ansatz gilt auch für Hardware. Handel-C basiert auf demselben Protokoll und das Design-Kit DK wandelt im Prinzip C-Funktionen in Logikblöcke um, die über Register verbunden sind. Es optimiert das resultierende Netzwerk und erzeugt daraus ein EDIF-File für die FPGA-Tools von Altera/Xilinx. Neben EDIF kann auch Code in RTL

(Register Transfer Logic) in VHDL und Verilog für Standard-Tool-Ketten beim FPGA/ASIC-Design generiert werden.

Aus dem Denkmodell Logik/Register folgt, dass Handel-C eine vollkommen synchrone Implementierung der Logik erzeugt. Es gibt also nur einen Takt, der alles steuert. Änderungen wandern mit der positiven Taktflanke von Block zu Block. Zwischen zwei Taktzyklen stabilisiert sich der neue Logikstatus und wird mit dem nächsten Takt weitergegeben. Das gesamte Netzwerk ist per Definition voll synchron. Bei einer direkten Übersetzung von C in Gatter ergibt sich damit ein sequenzieller Ablauf wie bei Software, nur schneller. Die richtige Beschleunigung kommt mit dem Einsatz des PAR-Statements, wobei die davon eingeschlossenen Funktionen als Gatter implementiert jetzt paral-

schnell genug sein, wird kritischer Code in Assembler optimiert, entsprechend dem weiteren Parallelisieren in Handel-C. Das Ziel ist nicht unbedingt, von Anfang an die schnellste Implementierung zu erreichen: den Prototyp so schnell wie möglich und fehlerfrei zu erstellen ist gefragt. Optimiert wird soweit wie notwendig, um eine akzeptable Gatterzahl zu erreichen. Der Hauptvorteil ist die schnelle Top-down-Entwicklung in nur einer Designumgebung, wobei die Funktionsübertragung in ein FPGA von Anfang an einfach möglich ist (Bild 1). Es wird von Celoxica nicht nur die DK-Software für Handel-C angeboten, sondern auch FPGA-Boards und die Verbindungssoftware, um fast transparent Software eingebunden im FPGA ablaufen zu lassen. Der Entwickler kann also von Anfang an einfach abschätzen, welche Ablaufgeschwindigkeit sich in Software oder Hardware ergibt. Die zusätzlichen »Nebenwirkungen« sollten nicht unterschätzt werden: Da mit der transparenten Implementierung als Soft- oder Hardware von Anfang an »gespielt« werden kann, ist bei der Partitionierung bereits viel Know-how über die Applikation vorhanden und die »Hot-Spots« sind bekannt. Wichtig ist auch, dass alles in Handel-C

Fast alle ANSI-C-Konstrukte werden unterstützt, auch while, if, do, for, static, truct, typedef.

Simulation/Synthese für bis zu 128 Bit Breite, auch für Busse, Floatingpoint kann in Test-Benches eingesetzt werden.

par	zeitunabhängige parallele Ausführung
chan	Kommunikationskanal zu parallelen Subsystemen
int#nn	Definition der Signalbreite von 1 bis n
unsigned#xx	Definition eines unsigned-Datentyps der Breite x
[x..y]	Operator, um nur die Bits x bis y auszuwählen
han	Definition eines synchronen Kommunikationskanals
achan	Definition eines asynchronen Kommunikationskanals

Tabelle 2: Einige Bach C-Features

lel schalten. Zuerst werden vom Entwickler die offensichtlichen Funktionen parallelisiert und dann solange weiter optimiert, bis die Applikation schnell genug ist. Sollte die erforderliche Geschwindigkeit durch die Limitierung auf nur einen Taktbereich nicht erreichbar sein, können Teile definiert werden, die von unabhängigen Clocks gespeist werden. Hier triggert dann ein Taktbereich den nächsten zur sofortigen Ausführung. Ein solcher Ansatz ist zum Beispiel auch notwendig, wenn Peripheriefunktionen von externen Takten abhängen.

Der Weg der progressiven Parallelisierung kann vielleicht am besten mit einer Softwareentwicklung in C und Assembler verglichen werden: Zuerst wird die Applikation/Algorithmus entwickelt und verifiziert. Sollte die Ausführung in C nicht

(Programm oder Testbench) bleibt, egal ob es als Soft- oder Hardware ausgeführt wird. Das EDIF für die Altera/Xilinx-Tools kann ohne Umweg über VHDL erzeugt werden, Umkodierungsfehler in VHDL werden vermieden – alles bleibt bis zum Schluss in Handel-C.

Im CSP-Ansatz scheint irgendwie Musik zu sein, denn neben Handel-C gibt es auch Bach. Sharp ist

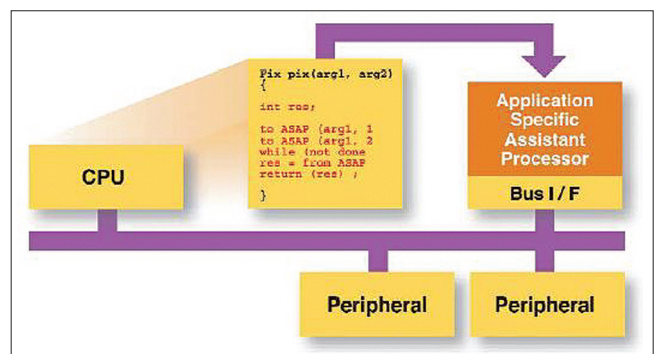


Bild 3. Mentor-ASAP-Ansatz integriert Sharps Bach (Bild: Mentor)

High-Level-Language-Lösung für die schnelle Entwicklung von Multi-Million-FPGAs /SopCs

basiert auf ISO/ANSI-C	Möglichkeit zur direkten Hardwareimplementierung beim Prototyping
einfaches Timing-Modell	einfach für Softwaredesigner da komplett synchrones Verhalten
auch komplexe C-Funktionen	flache Lernkurve für Softwerker, damit schnelle Umsetzung in HW
mit Bit-Manipulation	schnelle Umsetzung von DSP-Algorithmen in effiziente Hardware
Statemachines direkt in C	vereinfacht die Entwicklung komplexer Kontroll-Flows
RAM/ROM/IO Konstrukte mit Clock/Enable/Transfer	komplette Systemdefinition möglich
mehrere Taktbereiche	schneller/effizienter Einsatz der Hardware über einfache Syntax
	basiert auf einfachem Taktmodell, erlaubt aber zusätzliche Taktbereiche

Tabelle 3: Handel-C Kurzübersicht der Hauptvorteile

von demselben C-Startpunkt ausgegangen, und zwar SharpLaboratories of Europe in Oxford und Sharp Corporation in Japan. Bisher wurde diese Lösung nur intern zur Entwicklung von etwa 20 ASICs mit extrem reduzierter Designzeit eingesetzt. Bald ist diese Lösung über Mentor Graphics auch extern verfügbar: Die Bach-Technologie wurde in die Mentor-ASAP-Tools integriert. Im Gegensatz zu Handel-C ist bei Bach das Ziel die Generierung eines möglichst effizienten VHDL-Codes, der dann von Standard-ASIC/FPGA-Tools weiterverarbeitet wird. Bach ist im Gegensatz zum voll synchronen Han-

del-C »untimed«, also nicht an einen übergeordneten Takt gebunden. Die Funktionsbeschreibung bei Bach-C ist nicht zeitbezogen (untimed, nicht asynchron), damit hat der Bach-Synthesizer mehr Freiheitsgrade zur Erzeugung von effizientem VHDL-Code im Vergleich zu einer synchronen Beschreibung. Leider geht damit auch die direkte Beziehung zwischen Code und Ausführungsablauf verloren. Daten können jedoch auch durch komplexe Zeitbeziehungen nicht verlorengehen.

Im Gegensatz zu VHDL und SystemC ist die Parallelität nicht auf den Top-Level beschränkt.

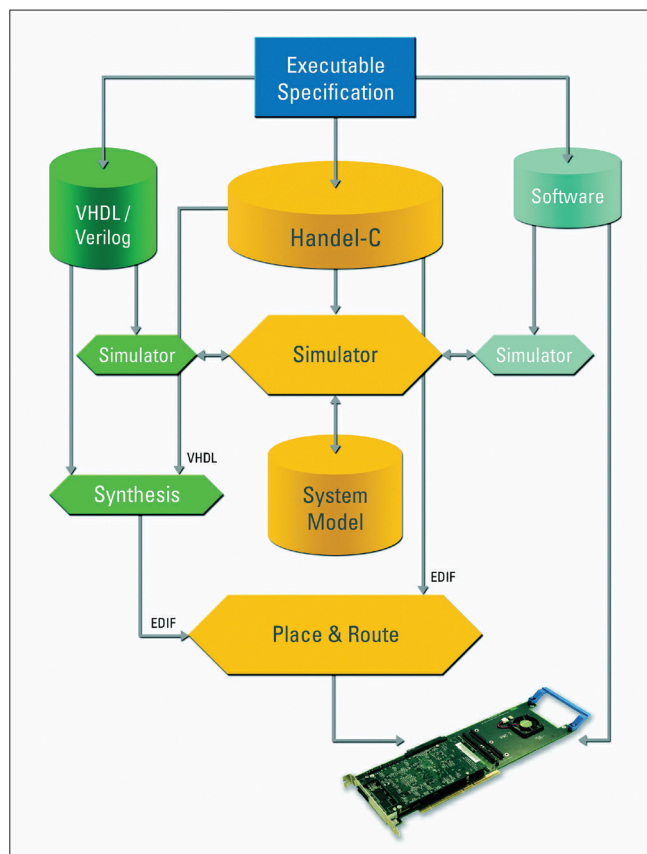


Bild 4. Handel-C-System-Design-Flow

(Bild: Celoxica)

Mit PAR werden Unterprozesse als gleichzeitig auszuführen definiert und können selbst weitere PARs enthalten. Bei der Ausführung laufen dann alle Subprozesse gleichzeitig ab, bis das PAR in der höchsten Ebene beendet ist. Der Datentransfer zwischen gleichzeitigen Prozessen wird über synchrone oder asynchrone Konstrukte definiert, jeweils »send« und »receive«. Sender und Empfänger müssen zum Transfer bereit sein. Jeder ACHAN (asynchrone Kanal) arbeitet wie eine globale Variable, daher kann der genaue Ablauf bei der Simulation nicht definiert werden. Zusätzlich zur internen Kommunikation sind beide Konstrukte auch für I/O und andere externe Transfers vorgesehen. Bach enthält einen Behavioural-Synthesizer, der den parallelen Bach-Code in mehrere Kommunikationsmodule kompiliert und auch die

dazugehörigen I/O-Interfaces generiert. Das VHDL-RTL kann direkt als Input für Standard-VHDL-Tools verwendet werden. Bach wurde in Mentors Tool ASAP integriert und ist bereits im Betatest bei Kunden im Einsatz.

Newcomer Critical Blue deckt einen etwas anderen Bereich ab: Hier wird nicht von C, sondern von bereits kompiliertem Applikationscode ausgegangen. Der wird von Cascade analysiert, automatisch wird parallel ausführbarer Code erkannt und daraus Standard-VHDL erzeugt. Als Hauptziel soll nicht der gesamte Code in Hardware ausgeführt werden, sondern nur Teile, die zum Beispiel besonders viel Prozessorzeit erfordern oder schneller ablaufen müssen. Als Ergebnis entsteht das VHDL für einen Coprozessor in Hardware, der über den Bus mit den anderen Blöcken im System kommuniziert. Sollte das existierende Prozessorsystem nicht mehr genug sein, muss zum Beispiel die Taktfrequenz nicht erhöht werden; alles bleibt wie es ist, es kommt sozusagen nur ein Spezialist als FPGA oder ASIC dazu, der den »Überlauf« abarbeitet. Es wird dabei von bereits getestetem kompiliertem Applikationscode ausgegangen, die erste Implementierung von Cascade verarbeitet ARM-Code. Dabei nimmt das Cascade-Tool den kompilierten Code, der mit den normalen Prozessor-Tools erzeugt wurde. Die Analyse des Objektcodes ergibt einen schnellen Überblick bezüglich der möglichen Aufteilung in Prozessorcode und Coprozessorfunktionen. Die automatische Erstellung des zyklusgenauen C-Modells für den ausgewählten Code erlaubt die schnelle Analyse der resultierenden Coprozessorleistung. Als nächster Schritt wird daraus automatisch VHDL-RTL erzeugt und mit Standardtools die Daten für FPGAs und ASICs generiert.

Die Dominanz der Software in Projekten ist bekannt. Mit diesen Tools erweitert sich der Einfluss der Software noch mehr in den Bereich Hardware. Immer schnellere Produktzyklen erfordern beschleunigte Codegenerierung, von daher muss auf einer höheren Abstraktionsebene gearbeitet werden, bei der schneller mehr getesteter Code entsteht. Handel-C nimmt als Startpunkt C, erlaubt sofort ohne Umweg das Prototyping im FPGA und hilft bei der Partitionierung und Verifikation. Bach als Teil von Mentors ASAP unterstützt den schnellen Weg über VHDL zum ASIC oder FPGA. Critical Blues Cascade-Lösung liegt irgendwo dazwischen, zum Beispiel wenn das Prozessorsystem nach einem »Turbo« ruft und der Rest unverändert bleiben soll. Die größte Hürde beim Einsatz dieser Tools ist damit auch die Entwicklungsabteilung: den bewährten Designflow zu ändern, ist immer ein Risiko. Vielleicht kann man diesen Übergang am besten mit der Einführung von VHDL vergleichen, als das Schaltbild plötzlich durch viele Zeilen Code beschrieben wurde, was heute Standard ist.

(Jürgen Pintaske, ExMark/pa)

Info: www.sle.sharp.co.uk/research/scd
www.mentor.com/ASAP, www.celoxica.com
www.criticalblue.com