

Rapid Prototyping für Mess- oder Regeltechnische Aufgaben

Einen Prototyp schnell und wirtschaftlich aufzubauen, gewinnt zunehmend an Bedeutung, denn die «Time-to-Market» ist häufig sehr entscheidend für den Erfolg des Produktes am Markt. Mit einem modularen Hardware-System und einer selbst dokumentierenden grafischen Programmierumgebung lassen sich mess- oder regeltechnische Aufgaben einfach und schnell implementieren und testen.

Die Entwicklung eines Produktes von der Idee bis zur Serie lässt sich grundsätzlich in drei Stufen einteilen: Machbarkeitsprüfung, Prototypenentwicklung und Realisierung der Serie (Bild 1). Gleichzeitig sollten alle Stufen transparent sein – auch für «Aussenstehende», um diese mit

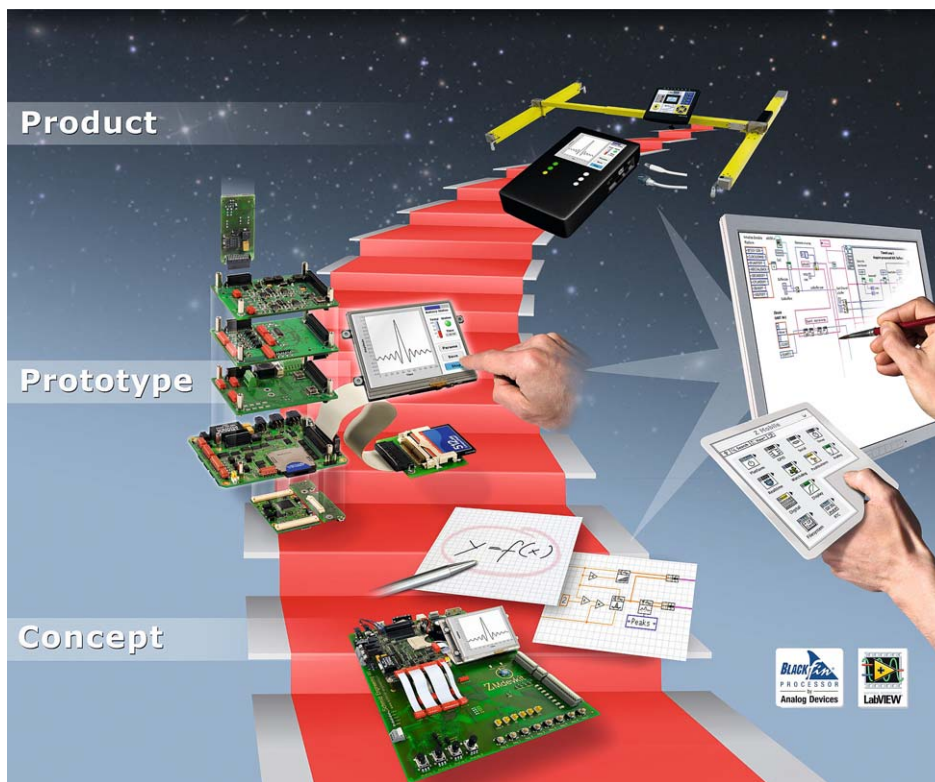
einzubeziehnen, zum Beispiel bei Code Reviews. Denn spätestens bei der Benutzeroberfläche möchte jeder mitreden.

Von der Idee zum validierten Konzept

Eine Produktentwicklung beginnt mit der Machbarkeitsprüfung einer

Idee, einer komplexen Aufgabe oder mit den funktionellen Anforderungen in einem Pflichtenheft. Wichtigste Voraussetzung ist hier ein einfaches und komfortables Entwicklungswerkzeug, das Raum für Kreativität, Neugier oder Trial-and-Error bietet und innerhalb weniger Tage reale Ergebnisse liefert. Dazu gehört auch, bereits in der Konzeptphase Algorithmen unterschiedlicher Ziel-Hardware zu testen und Benchmarks zu fahren. Eine Entwicklungsumgebung, aus modularer, praxisbewährter Hardware und der grafischen Programmieroberfläche LabVIEW ermöglicht es Wissenschaftlern und Systemingenieuren, sich ohne aufwändige Vorentwicklungen von Hardware, Software und Low-Level-Treibern direkt auf die Aufgaben zu konzentrieren. Also das in Gedanken bestehende Modell als grafische Darstellung auf dem Bildschirm abzubilden und anschliessend mit der unterlegten modularen Hardware zu testen. Die datenfluss- und funktionsorientierte Programmierung erfolgt per Drag-and-drop standardisierter Funktionsblöcke und führt so ohne grossen Programmieraufwand schnell zu verwertbaren Ergebnissen. Gleichzeitig bietet sie ideale Voraussetzungen für das Verfolgen von Fehlern und Erstellen von Vari-

Bild 1: Rapid Prototyping – in drei Stufen zum Endprodukt. Mit modularer Hardware und grafischer Systemprogrammierung.



Autoren

Marco Schmid, Schmid Engineering AG
Gerhard Schlicht, CC&I Computer Communication & Interface GmbH



Infos

Schmid Engineering AG
9542 Münchwilen
marco@schmid-engineering.ch
Tel. 071 969 35 90
www.schmid-engineering.ch



Bild 2: Das ZBrain-Entwicklungssystem ist eine grafisch programmierbare Entwicklungs- und Testumgebung mit skalierbarem Prozess I/O.

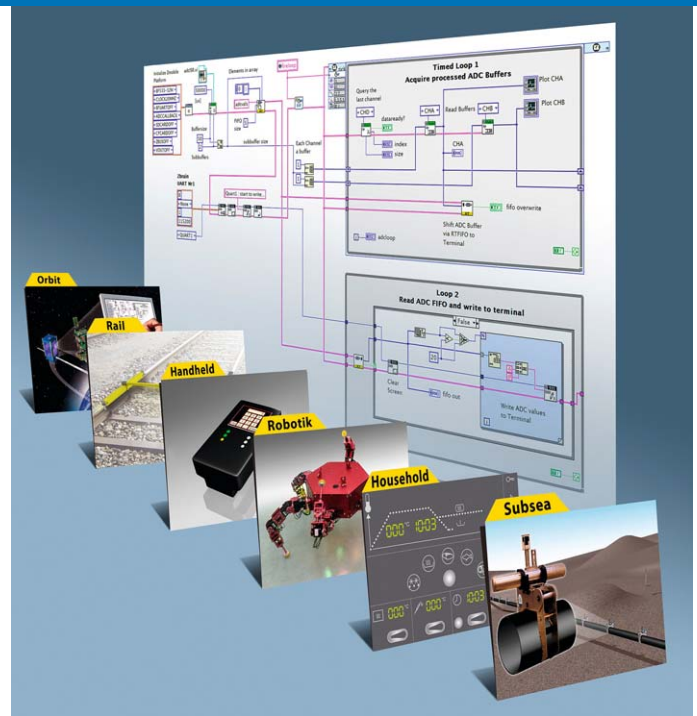


Bild 3: Software Frameworks sind vorentwickelte und getestete Vorlagen für unterschiedliche Anwendungsfelder.

anten. Grundlage hierzu ist eine grosse Zahl unterschiedlichster Funktionsblöcke (Virtuelle Instrumente/VIs), die lediglich mit Verbindungslinien am Bildschirm miteinander «verdrahtet» werden. Diese VIs reichen von einfachen Ein- und Ausgangsfunktionen bis zu komplexen mathematischen Analyse- und Signalverarbeitungsblöcken. Auch eigene Algorithmen, sogar C-Bibliotheken, können jederzeit eingebunden werden. Auf einfache und schnelle Art werden so unterschiedliche Mess- oder Regeltechnikideen und Lösungswege miteinander verknüpft. Mit Modellierung und Simulation lassen sich ein System und seine Komponenten auf hohem Abstraktionsniveau beschreiben und analysieren. Das gibt ein Gefühl für Zusammenhänge, wichtige Systemparameter und Verhalten. Auch mit wenig Erfahrung kann man sich dank einfacher, interaktiver Tools an den Entwurf digitaler Filter heranwagen, verschiedene Topologien (Hoch-, Tief-, Bandpass) ausprobieren und das Verhalten im Zeit- und Frequenzbereich beurteilen. Dasselbe gilt auch für die Identifikation von Regelstrecken sowie das Auslegen und Dimensionieren offener und geschlossener Regelkreise.

Nach Erstellen des Funktionsdiagramms wird der (noch) abstrakte Algorithmus mit realen Prozesssignalen verbunden und auf der mög-

lichen Ziel-Hardware live getestet (Bild 2). Mit Varianten lassen sich verschiedene Strategien durchspielen. Ein nahtloser Übergang von Floating-Point- zu Fixed-Point-Arithmetik spart enormen Aufwand und ermöglicht schon früh eine Verhaltensanalyse des Algo-

rithmus auf dem Zielsystem. Auf diese Weise entsteht auf der Entwicklungsplattform mit der Machbarkeitsprüfung Schritt für Schritt eine Struktur, ein grafisches Konzept.

Natürlich kann eine Machbarkeit auch auf andere Weise, ➔

Systemvergleich

Teure Hardware, günstige Entwicklung

Schneller zum Ziel

Die vorgestellte Entwicklungsmethode zusammen mit der modularen Hard- und Software führt zu einer wesentlich höheren Produktivität aller Beteiligten. Die Entwicklungszeiten und damit die Entwicklungskosten reduzieren sich um Faktoren, was die Time-to-Market deutlich verkürzt. Software-Anpassungen im fertigen Produkt an besondere Gegebenheiten eines speziellen Kundenprozesses, sind einfach und auch vor Ort durchführbar.

Realtime-Anwendungen

Hier gilt es, rechtzeitig den konzeptionellen Aufbau und die Engpässe zu bestimmen. Allgemein lässt eine komplizierte Applikationslogik mit vielen Benutzerinteraktionen weniger Spielraum für komplexe DSP-Algorithmen. Eine Blackbox ohne jegliche Benutzereingriffe schafft hingegen eine Datenerfassung im MHz-Bereich. Benchmarking und kompetente Unterstützung schon während der Machbarkeitsprüfung steigern die Projektsicherheit erheblich und können vor Überraschungen schützen.

Hardware-Kosten der Serie

Der hohe Abstraktionsgrad der grafischen Software und der Wunsch nach Portierbarkeit verlangen genügend Hardware-Ressourcen und verteuern Ziel-Hardware. Diese besteht meistens aus einem DSP mit mindestens 32 MByte RAM und 8 MByte Flash. Die Mehrkosten durch etwas höhere Prozessor- und Speicherkosten sind jedoch im Hinblick auf die Gesamtkosten in den meisten Fällen vernachlässigbar.

Return of Investment

Die zum Start notwendigen Investitionen für die Werkzeuge sind nicht unbedeutend. Betrachtet man sie aber in Relation zu den Einsparungen bei den Entwicklungs-, Änderungs- und Wartungskosten über den Produktlebenszyklus, ist es eine Investition mit hoher Rendite. Mit dem Gewinn an Flexibilität hat der Anwender ein mächtiges Instrument in der Hand, um einfach, schnell und erfolgreich neue Marktanforderungen zu meistern.

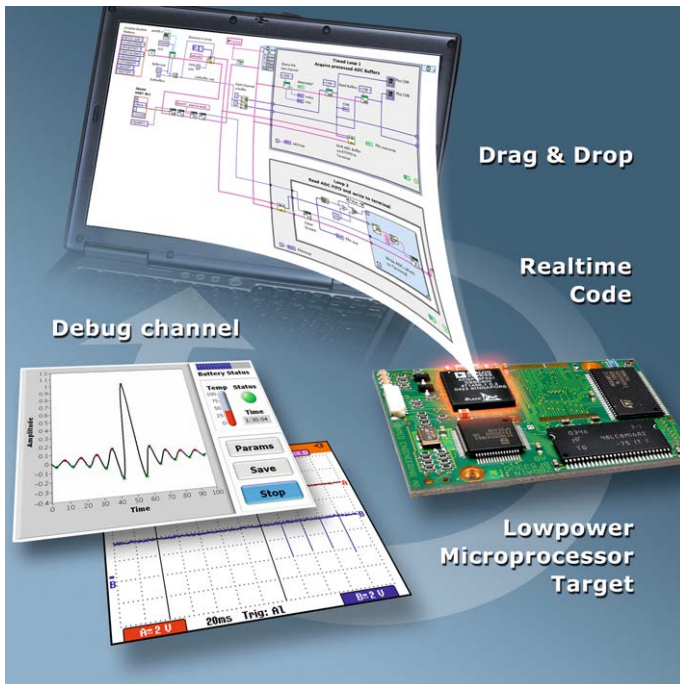


Bild 4: LabView-Blockdiagramme werden in Echtzeitcode übersetzt, als Firmware ins Zielsystem geladen und dort live getestet.

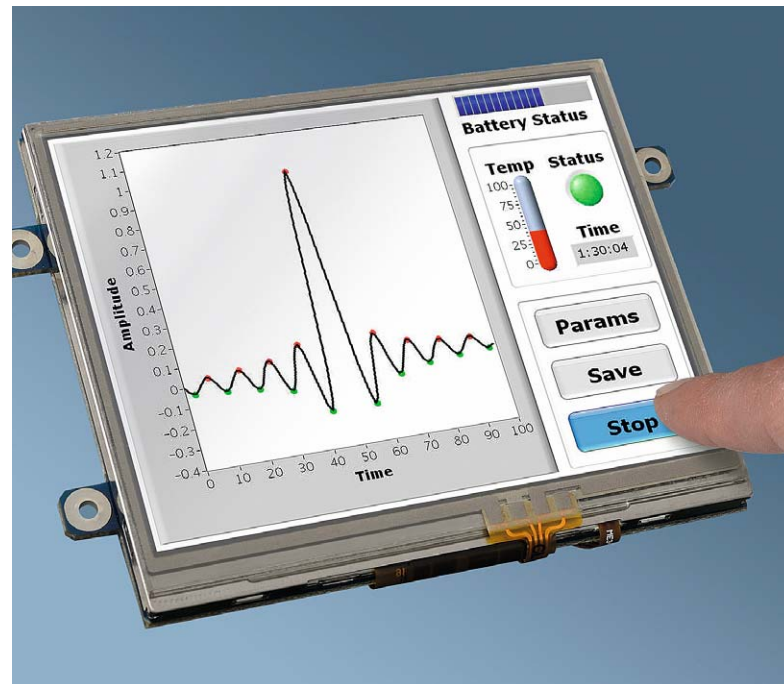


Bild 5: Interaktive grafische Bedienerschnittstellen (GUIs) werden am PC gestaltet und ins Embedded-Zielsystem geladen.

zum Beispiel auf PC- oder PXI-basierten Systemen, überprüft werden. Die Verwendung der produktiven, grafischen Programmierumgebung ist aber auch hier empfehlenswert. Stimmen die Ergebnisse des Konzepts, ist die Machbarkeit geprüft, dann ist der Weg frei für die eigentliche Prototypenentwicklung.

Schritt für Schritt zur Lösung

Die grundsätzlichen Anforderungen an ein Prototyp-Entwicklungssystem sind den Anforderungen für die Machbarkeitsprüfung ähnlich. Im Prototyping gehören rollende Spezifikationen, Sonderwünsche, Projektänderungen und Varianten-erstellung zum täglichen Brot. Also muss auch die Entwicklungsmethode einerseits änderungsfreundlich und einfach zu korrigieren sein. Andererseits sind für alle Beteiligten Nachvollziehbarkeit und schnelle Überprüfung der einzelnen Programmfortschritte besonders wichtig, das heißt in kleinen und sicheren Schritten zur Lösung zu gelangen. Die einzelnen Schritte sollten immer wieder an der Prototypen-Hardware überprüft werden. Ein weiterer Punkt ist eine perma-

nente und transparente Dokumentation, die das Problem eines Stellenwechsels mitten in einem Projekt drastisch entschärft. Auf einer klar strukturierten und selbst dokumentierenden Programmieroberfläche kann jeder neue Projekt-ingenieur in kurzer Zeit produktiv werden. Mit einer modularen Hardware und einer für jeden verständlichen, grafischen Programmieroberfläche wird der Übergang vom grafischen Konzept aus der Machbarkeitsprüfung zur Prototypen-erstellung fließend, denn Algorithmen aus der Machbarkeitsprüfung lassen sich zum großen Teil 1:1 übernehmen. Die Entwicklung von Software, Benutzeroberfläche und Hardware sind voneinander entkoppelt und können von Anfang an parallel betrieben werden.

Echtzeit-Software grafisch entwickeln

Die Entwicklung von Software für Embedded-Systeme ist meistens eine schwierig abzuschätzende Aufgabe, weil Multitasking, Echtzeit, Parallelität, Synchronität und restriktive Timings auch für Spezialisten Herausforderungen sind. Systemingenieuren, Prozessspezialis-

ten und Wissenschaftlern ist dieses Know-how häufig weniger geläufig. Trotzdem sollen und müssen auf der Embedded-Plattform schnell Fortschritte und überprüfbare Ergebnisse erzielt werden. Hier bieten sich Softwareframeworks als Starthilfe an (Bild 3). Dies sind lösungsneutrale Software-Vorlagen, in denen Multitasking, Echtzeitverhalten, Fehlerbehandlung, Kommunikation und Prozess-I/O bereits implementiert sind. Zu den zwei Letzteren gehören nach Bedarf Dienste wie Analog-, Digital- und Serial I/O, USB, Ethernet-Anbindung, CAN, Parameter- und Massenspeicher, grafische Bedienerschnittstellen, Audio und Video. Die Softwareframeworks sind weitgehend plattformunabhängig und deshalb die ideale Arbeitsumgebung für sich ändernde Situationen. Sie bieten Blockschaltbilder mit System-sicht auf Hard- und Software.

Was dem Rohbau noch fehlt, sind die Algorithmen, welche nun Schritt für Schritt eingebettet und verifiziert werden können. Bei Bedarf lassen sich auch «Super-VIs» erstellen, die mit mächtigen C++ Klassen vergleichbar sind, womit

sich eine noch höhere Abstraktionsstufe erreichen lässt.

Grafische Debugging-Hilfsmittel erleichtern und verkürzen die Fehlersuche. Sequenzieller und paralleler Datenfluss wird im gesamten Diagramm auf Knopfdruck visualisiert. Vom Anwender gesetzte Testpunkte zeigen die aktuellen Werte von Variablen an und Hardware-in-the-Loop (HIL) bezieht die Embedded Hardware mit ein.

Per Drag-and-drop zur Benutzeroberfläche

Es gibt Embedded-Systeme, die ohne Bedieneringriffe auskommen und deshalb keine besonderen Ansprüche an die Benutzeroberfläche stellen. In den meisten Fällen ist jedoch eine grafisch gestaltete Benutzeroberfläche oder eine Bedienerführung erwünscht oder notwendig. In manchen Fällen ist die Bedienerergonomie des GUIs (Graphical User Interfaces) sogar für den Erfolg eines Gerätes am Markt mit entscheidend. Jeder, der schon einmal eine Bedienoberfläche für ein System gestalten durfte, ist sich des grossen Aufwandes dieser Aufgabe bewusst. Sie umfasst nicht nur einen hohen Programmieraufwand, sondern auch Abstimmungen mit Designern, Marketing, Kunden und Endanwendern. Alle möchten mitreden, haben Änderungswünsche und Abstimmungsbedarf. Nach einer kurzen Einführung kann jede der beteiligten Gruppen oder Personen durch Verbinden grafischer Elemente ihre Vorschläge selbst entwerfen, realisieren und prüfen. Das LabView-Frontpanel-Konzept für Embedded-Anwendungen unterstützt diesen Prozess durch einfach zu handhabende, sogenannte Faceplates. Unterschiedlichste Varianten und Strategien vom einfachen Prozessmonitor bis hin zu komplexen, mehrschichtigen Benutzerführungen können per Drag & Drop aufgebaut werden. Steuerungselemente, Eingabefeld, Taster, Schieberegler und Drehknöpfe, Anzeigen, Linienplots, Tachos oder Bargraphen lassen sich zu Benutzeroberflächen kombinieren. Sehr gute Effekte können mit

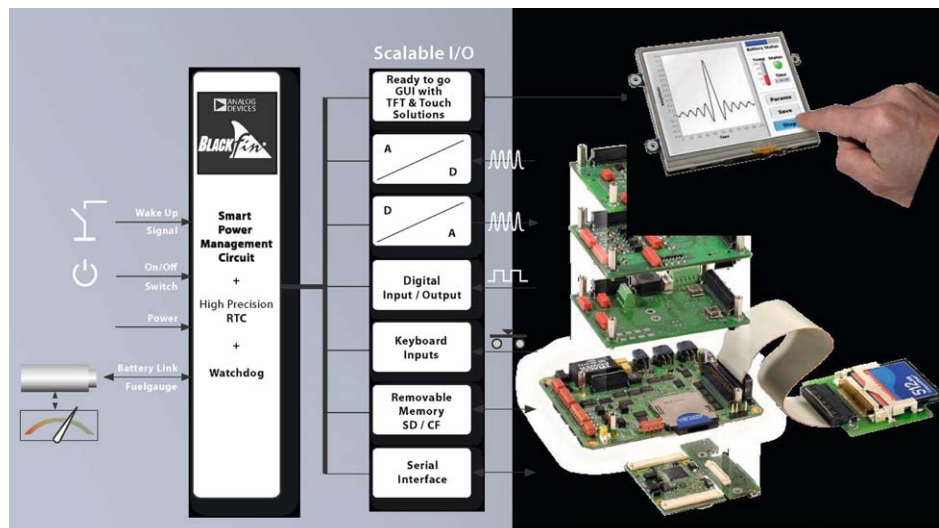


Bild 6: Die standardisierte, modulare ZBrain Hardware ist skalierbar, real-time-fähig und hat einen sehr geringen Stromverbrauch.

unterschiedlich grossen JPG-Bildern erzielt werden. Sie werden über VIs sichtbar gemacht und positioniert. So lassen sich GUI-Prototypen oder ergonomische Grafikelemente mit gängigen Zeichnungsprogrammen entwerfen und ins Zielsystem laden. Die Benutzer-eingaben werden in VIs ausgewertet und als Events von der Programmlogik verarbeitet. Anschliessend wird der Code generiert, welcher der Embedded-CPU die gesamte Kontrolle über das GUI übergibt. Als Interface dienen Farb-TFTs mit Touch-Funktionen (Bild 5). Die Bedienoberfläche kann parallel zur Hardware und zur Prozess-Software entwickelt werden.

Die Hardware wächst flexibel mit der Aufgabe

Neben der Software ist das Prozess-I/O während der Entwicklung

die zweite grosse Variable. Entweder verlangen Algorithmen nach neuen Inputs oder die Prozessanforderungen verändern sich. Mit felderprobten Prozess-I/O-Modulen, die sich bedarfsabhängig als «Turm» aufeinanderstecken lassen, lässt sich das ZBrain-System jederzeit modifizieren und erweitern (Bild 6). Die Embedded-Hardware lässt sich so schnell und einfach den unterschiedlichsten Anforderungen anpassen. Die Wahl der CPU-Plattform lässt sich ohne Einfluss auf die Software hinauszuzögern, da die Systemsprache LabView weitgehend plattformunabhängig ist und die Hardware-Funktionen über High-Level-Funktionsblöcke angesprochen werden. Sobald alle Varianten geprüft und damit die Anforderungen an die Rechenleistung definiert sind, wird die passende CPU aus der →

Modulare Plattform

Das ZBrain-Hardware-System

Das ZBrain-System ist eine Kombination aus modernen, einfach zu handhabenden Werkzeugen, universeller Testumgebung und kompakten, skalierbaren Hardware-Modulen. Es eignet sich für mess- oder regeltechnische Aufgaben und ermöglicht eine schnelle Machbarkeitsprüfung, effiziente Erstellung und Test der Software, der Bedienoberfläche, der Prototypen- und der Ziel-Hardware. Zusätzliche Funktionen lassen sich mit kundenspezifischen I/O-Boards realisieren.

Auf der ZBrain-Ziel-Hardware läuft kein übliches Betriebssystem, sondern ein echtzeitfähiger, kooperativer Multitaskingkernel von Analog Devices. Es handelt sich dabei um ein rudimentäres RTOS für Blackfin-Prozessoren. Das Zbrain Board Support Package (BSP) bietet über 200 typische Embedded-Funktionen an, die mit LabView per Drag-and-Drop in die Software integriert werden können.

Blackfin Prozessorfamilie gewählt. Der Blackfin ist ein Mikroprozessor, dessen Core unter anderem aus einem 32-Bit-RISC-Prozessor und einem 16-Bit Fixed Point DSP besteht. Zusammen mit grafischer Programmierung ist er eine ideale CPU-Plattform mit hohen Leistungsreserven. Der DSP eignet sich zur schnellen Abarbeitung von Algorithmen. Trotzdem sind keine speziellen DSP-Kenntnisse erforderlich, da die überlagerte System-Software diese Aufgaben übernimmt. Für einfache Systeme, wie zum Beispiel mobile Handhelds, genügen meist stromsparende Singlecore-Prozessoren mit entsprechender Eingabeperipherie. Komplexe, besonders rechenintensive Aufgaben können bei Bedarf auf

mehrere Kerne (Multicore-Prozessoren) verteilt werden. Bei umfassendem, mehrkanaligem Prozess-I/O mit paralleler Signalverarbeitung und Datastreaming bieten sich Lösungen an, die zusätzlich ein FPGA (Field Programmable Gatearray) verwenden.

Vom Prototyp zum Endprodukt

Wie beim Übergang von der Machbarkeitsprüfung zum Prototypen ist auch der Schritt zum Endprodukt fließend. Durch den modularen Aufbau der Hardware ist für das Endgerät häufig nur ein anderes Gehäuse notwendig. Unterscheiden sich jedoch der mechanische Aufbau und damit der Formfaktor oder die zulässigen Kosten

des Serienproduktes von denen der Prototypen, dann ist Optimieren angesagt. Die felderprobten I/O-Schaltungen des Prototyps werden dann per Drag-and-drop den Serien-Anforderungen entsprechend zusammengefasst und gegebenenfalls modifiziert. Die Verbindung mit dem Prozessor erfolgt dabei über definierte Hardware-Ports, womit die Low-Level-Treiber wiederverwendet werden können. Das Risiko aufwändiger Neuentwicklung der Serien-Hardware wird auf ein Minimum reduziert. Die für den Prototyp entwickelte Software kann weitgehend eins zu eins auf das endgültige Zielsystem übernommen werden. Damit entsteht in kurzer Zeit ein aufgabenspezifisches Serienprodukt. (pm) ■