Holodeck für die Entwicklung eingebetteter Systeme

Gründe für die Virtualisierung eingebetteter Systeme

Virtuelle Maschinen sind heute Stand der Technik und helfen, Rechenzentren effizient auszulasten. Auch auf dem PC hat sich die Virtualisierung breitgemacht – teilweise, ohne dass es dem Anwender wirklich bewusst wird. Und selbst bei eingebetteten Systemen hat Virtualisierung mittlerweile ihren Platz. Ist gerade kein Smartphone zur Hand, kommt man zum Beispiel über den BlueStacks App Player zu seiner täglichen Dosis Angry Birds. Eine Spielerei? Auch, aber eben nicht nur. Für den Entwickler eingebetteter Software kann eine virtualisierte Plattform ein wertvolles Test- und Debugging-Werkzeug sein.

Warum eingebettete Systeme virtualisieren?

Auf eingebetteten Systemen ist niemand ernsthaft versucht, Software entwickeln zu wollen. PCs bieten eine weit leistungsfähigere Entwicklungsumgebung mit Cross-Compiler. Um Binärcode ausführen zu können, ist er allerdings umständlich auf das eingebettete System zu überspielen, und ab hier kann das Debuggen echte Verzweiflung hervorrufen. Denn „eingebettet“ bedeutet, dass diese Systeme ein Teil einer möglicherweise komplexen Umgebung sind. Es kann bei der Testautomatisierung unmöglich sein, Umgebungsparameter derart zu kontrollieren, als dass Ergebnisse reproduzierbar wären.

Tatsächlich ist die Emulation verbreiteter eingebetteter Systeme schon heute Stand der Technik. Das Android SDK enthält einen „Mobile Device Emulator“, der auf dem Entwicklungssystem Android-Applikationen testweise ausführt, ohne dass ein physisches Android-Gerät vorhanden sein müsste.

Für realistische Funktionstests fehlt der Emulation die Anbindung an eine Umgebung, mit der das eingebettete System interagieren soll. Die Fachwelt nennt eine solche Anbindung „in the Loop“, oder kurz IL. Die echte oder emulierte Hardware kommuniziert über IL mit der echten oder ebenfalls simulierten Umgebung. Auf diese Weise sind verschiedenste IL-Kombinationen denkbar.

Ein „in-the-Loop“ für alle Fälle

Ein System wird für Konstrukteure und Kunden früh erfahrbar, wenn ein Prototyp als Simulation vorliegt. Erwartete und insbesondere unerwartete Effekte werden so sichtbar. Entwickler bekommen ein Gefühl für das System und können den Ressourcenbedarf abschätzen. Denn die Integration von Elektrik, Elektronik, Mechanik, Fluidik, Chemie und Biologie am Laborarbeitsplatz ist zeitintensiv und dabei teuer und platzraubend. Virtuelle Komponenten lassen sich flexibel austauschen und benötigen keinen anderen Arbeitsplatz außer einem leistungsfähigen Rechner.

Simulation ist auf unterschiedlichen Ebenen der Abstraktion möglich: abstrakt und schnell oder genau und langsamer. Beim modellbasierten Softwareentwurf wird der Weg vom Abstrakten über das Hinzufügen von immer mehr Detaillierungsgrad zum Konkreten verfolgt. So entsteht vom Beginn bis zum Ende eines Projekts eine Evolution über diverse „X in the Loops“ bis schließlich zum Prototypen.

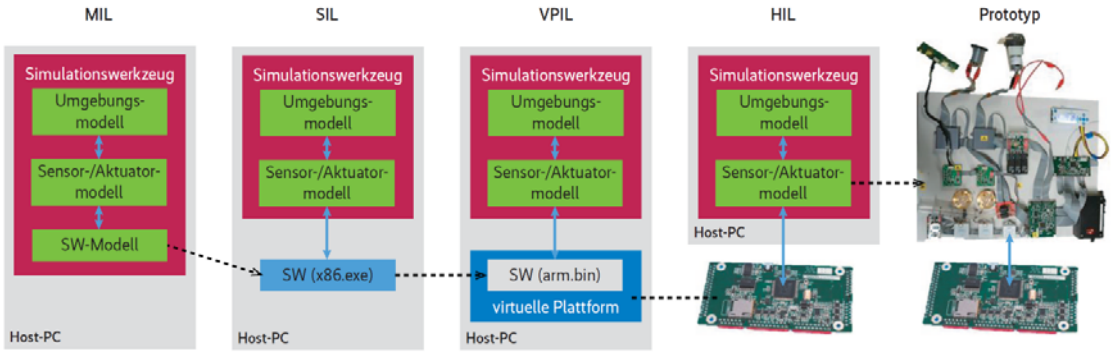

Abb. 1: Schritt für Schritt vom Modell zum Prototyp

Die Abb. 1 zeigt gängige Konzepte in der Übersicht mit von links nach rechts fallendem Abstraktionsgrad. Während das eingebettete System über den Prozess weiterentwickelt wird, ist die Testumgebung idealerweise vom Anfang bis zum Ende dieselbe und wiederverwertbar. Da es sich bei diesem Dokument um eine gekürzte Version handelt, kass aus platzgründen nur kurz auf VPIL (Virtual-Platform-in-the-Loop) eingegangen werden. Weiteres zum ungekürzten Artikel findet sich am Ende dieses Textes.
Sowohl beim MIL (Model-in-the-Loop) als auch bei der SIL (Software-in-the-Loop) lassen sich nur bedingt Aussagen über den Ressourcenverbrauch und die Echtzeitfähigkeit der eingebetteten Software machen. Zu verschieden ist die Hardware in Eigenschaften wie Performanz, Bitbreite, Fließkomma-Arithmetik in Hardware oder Software, Little versus Big Endian und Alignment. Wenn beim Erstellen der SIL die Portabilität der Codebasis berücksichtigt wird, lässt sich mit Cross-Kompilierung der Binärcode für die Zielplattform unter Berücksichtigung dieser Besonderheiten erstellen.
Zwar könnte man ab diesem Punkt bereits mit Evaluierungs-Boards auf HIL (Hardware-in-the-Loop) aufsetzen. Doch dadurch beginnt man den Kampf an gleich zwei Fronten, nämlich dem Debuggen der Software und der Inbetriebnahme von schwer einsehbarer Hardware. Eine Alternative bilden virtuelle Plattformen wie SoCLib [2] oder OVP [3].
Es handelt sich hierbei um Bibliotheken von Instruktionssatzsimulatoren sowie Speicher- und Peripheriemodellen. Gleich einem Bausatz kann der Entwickler aus diesen Komponenten seine eigene virtuelle Hardwareplattform zusammenstecken und darauf seine crosscompilierte Software ausführen. Dabei sind Laufzeit- und Echtzeitverhalten sowie der Ressourcenverbrauch relativ genau messbar – in Abhängigkeit von der Genauigkeit der Umsetzung der virtuellen Plattform.

Beispiel aus der Medizintechnik

Im folgenden Beispiel aus der Medizintechnik wurden die genannten Methoden angewendet, um die Umsetzung eines WAKD (Wearable Artefical Kidney Device) Abb. 2 zu erforschen. Es gilt, Elektrik, Elektronik, Mechanik, Fluidik, Chemie und Biologie eng miteinander zu verzahnen und ein für den Patienten lebensrettendes und insbesondere sicheres Gerät entstehen zu lassen. Im Vergleich zum typischen Dialysegerät, das circa 100˘kg wiegt, wäre eine tragbare Variante einer dramatischen Miniaturisierung zu unterwerfen. Allerdings muss sie dank Dauerbetrieb auch nur circa 7˘Prozent der Leistung eines herkömmlichen Geräts erbringen.

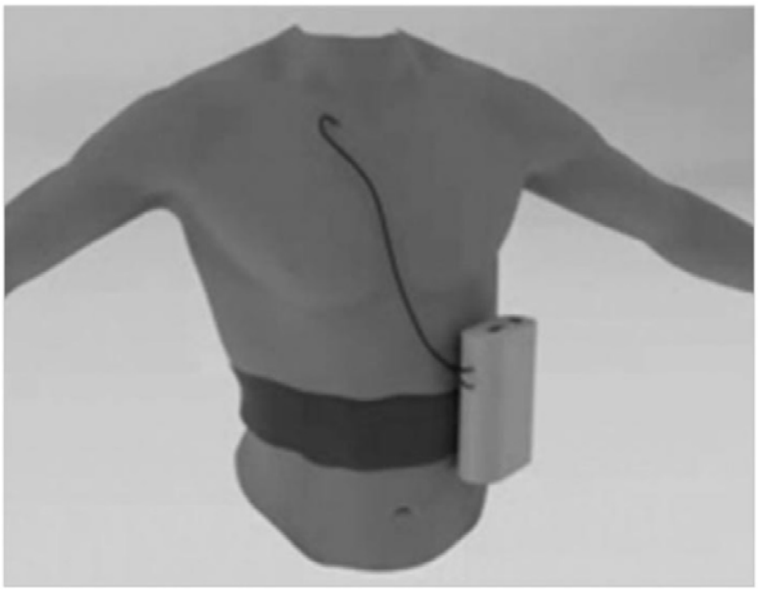

Abb. 2: WAKD (Wearable Artefical Kidney Device)

In dem von der Europäischen Union geförderten Projekt NEPHRON+ˇ[4] wurde bei der Entwicklung eines solchen WAKD der Virtualisierung eine gewichtige Rolle zugeschrieben. Die WAKD-Testumgebung wurde in zwei Teilmodelle zerlegt: das Patientenmodell und ein Modell der Systemfluidik.
Das Strukturmodell von einem Patienten ist mit zwei Behältnissen recht einfach. Ein Behältnis modelliert das Volumen der Flüssigkeiten mit ihren Inhaltsstoffen im Zellgewebe des menschlichen Körpers. Das andere Behältnis steht stellvertretend für das Blutvolumen im menschlichen Körper. Der Austausch von Stoffen und Flüssigkeiten zwischen beiden Behältnissen erfolgt zeitlich verzögert über Diffusion durch die Zellmembranen.
Das Modell aus Abb. 3 zeigt das Fluidik-System im WAKD mit zwei Flüssigkeitskreisläufen für Blut und Plasma. Die Modellierung von Drücken und Durchflüssen durch Schlauchleitungen ist relativ einfach, auch weil es für Hydrauliksysteme umfangreiche mathematische Modelle gibt. Die Leistung der Blutplasmaseparation und der Toxinseparation wurde empirisch im Labor bestimmt.

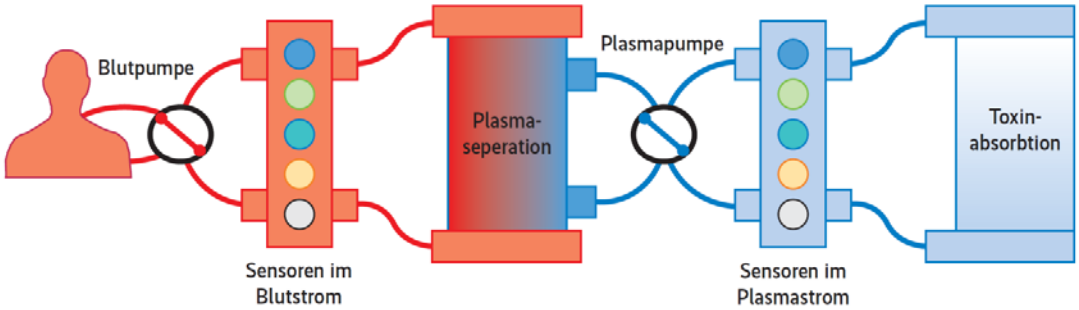

Abb. 3: Sensor-/Aktuator Modell: Virtuelle Sensoren messen Konzentrationen, Drücke, Volumenflüsse und Temperaturen

Der Toxinfilter nimmt Einfluss auf die Inhaltsstoffe, die sich auf die Behältnisse im Patientenmodell als veränderte Volumen, Temperatur und Konzentrationen widerspiegeln. Die Werte werden im Fluidik- Modell von den simulierten Sensoren ausgelesen. Das Überschreiten von Grenzwerten löst Alarme und Zustandswechsel im Softwaremodell aus – oder eben nicht, falls ein Fehler gemacht wurde. Die Entwicklungsarbeit und das Debugging konnten auf hohem Abstraktionsgrad beginnen.
Die Abb. 4 zeigt schematisch die virtuelle Plattform, des WAKD im Nephron+-Projekt. Patienten- und Fluidik-Modell laufen unter Simulink in einer Co-Simulation mit einem ARM-Instruktionssatzsimulator. Die Anbindung erfolgt über eine am OFFIS-Institut für die Co-Simulation entwickelte OVP-Peripherie [5]. Über die Verbindung einer virtuellen seriellen Schnittstelle mit einem realen Bluetooth Port wird mit der WAKD-App auf einem Android-Smartphone kommuniziert. Simulierte Pumpen in Simulink lassen sich so über das reale Benutzungsinterface steuern. Benutzer erhalten einen realistischen Eindruck von der App, und Entwickler können testen, ob Protokolle und Synchronisation zwischen Smartphone und WAKD funktionieren.

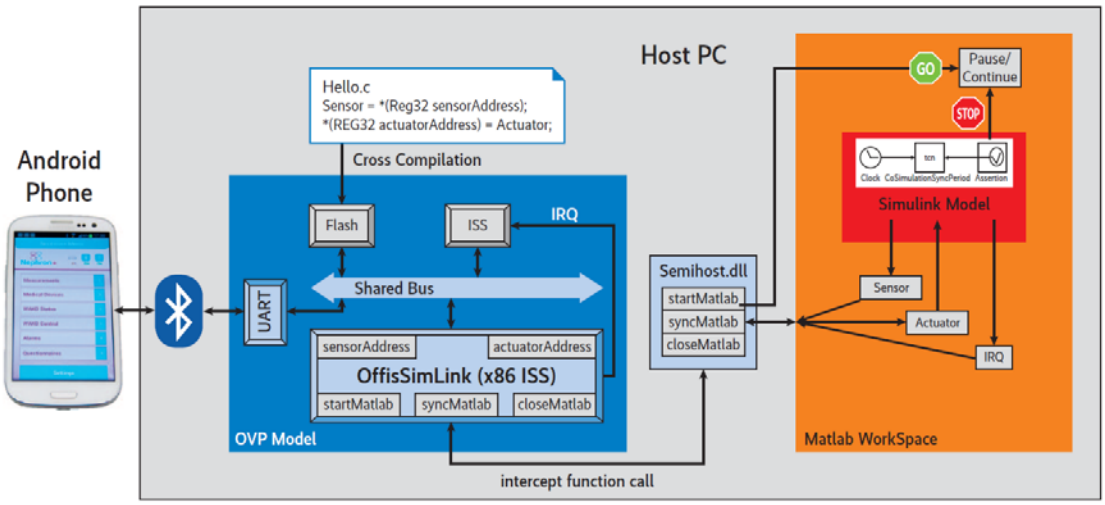

Abb. 4: Virtuelle Plattform für ein Dialysegerät.

Die Steuerungsalgorithmen zur Flusskontrolle laufen als Tasks unter dem Betriebssystems FreeRTOS auf dem Instruktionssatzsimulator. Die Emulation kann dabei je nach Detaillierungsgrad sogar schneller sein als das Original. Im Projekt wurde sechsfache Echtzeit mit einem Core-i5-Notebook erreicht. Ein Vorteil, wenn Tests des WAKD längere Therapiesituationen umfassen sollen, bei denen der simulierte Patient Nahrung oder Getränke zu sich nimmt.

Zusammenfassung
Der Aufwand von MIL über SIL nach VPIL und HIL zum Prototypen mag zunächst ungerechtfertigt erscheinen, und für kleinere Projekte stimmt das vermutlich sogar. Aber die Erfahrung hat gezeigt, dass bereits bei mittlerer Komplexität mit erheblichen Einsparungen bei den Integrationstests zu rechnen ist. Die graduelle Verfeinerung der Emulation bis weiter zum Prototyp erhöht den Detailierungsgrad in nur kleinen Schritten und damit auch die zu handhabende Komplexität. Die Simulationen lassen tiefe Einblicke in

jeden Winkel des Systems zu. In der Folge schleichen sich weniger Fehler ein, und diejenigen, die es doch schaffen, können sich nicht lange verstecken und werden bereits auf dem Weg zum Prototyp aufgespürt.

Literatur- und Quellenverzeichnis

[1] Simulink Coder, http://www.mathworks.de/products/simulink-coder/index.html
[2] SoCLib, www.soclib.fr
[3] Open Virtual Platform, www.ovpworld.org
[4] NEPHRON+, www.nephronplus.eu
[5] OFFIS SimLink; Open Virtual Platform – Matlab/Simulink Co-Simulation, www.system-synthesis.org/offissimlink

Bei diesem Artikel handelt es sich um eine gekürzte Fassung aus dem Sonderheft iX Developer Embedded Software mit dem Titel „Gründe für die Virtualisierung eingebetteter Systeme, Holodeck auf der Spur“. Der Autor bedankt sich beim Verlag Heise zur Zweitverwertungsmöglichkeit in dieser Form.

Autor

Frank Poppen befasst sich am Oldenburger OFFIS Institut für Informatik seit 15 Jahren mit der Entwicklung und praktischen Erprobung von Hardware Design Methodiken. Neben medizinischen Anwendungsfeldern liegen Kooperationserfahrungen gleichermaßen in der Avionik wie Automotive mit einem Schwerpunkt im Entwurf energieoptimierter integrierter Schaltungen (ASICs) vor. Frank Poppen ist Technical Chair einer industriellen Electronic Design Konferenz, der Synopsys User Group (SNUG).

Kontakt

Internet: www.offis.de
Email: frank.poppen@offis.de