

# Verifikation und Performance-Analyse

## Verschiedene MHP-Konfigurationen unter Verwendung virtueller Prototypen

Dieser Artikel präsentiert einen auf virtuellen Prototypen basierenden Ansatz zur Verifikation und Performance-Analyse. Dieser erlaubt eine frühe Untersuchung von Netzwerkprotokollen ohne die Existenz realer Hardware. Verschiedene Netzwerkkonfigurationen lassen sich so prüfen und durch automatisierte Simulationsläufe analysieren und testen.

Von Andreas Braun, Oliver Bringmann und Wolfgang Rosenstiel

Die Performance-Analyse verschiedener MOST-Geräte und -Netzwerkkonfigurationen sowie die Verifikation des verwendeten Protokoll-Stacks innerhalb eines Knotens sind wichtige, aber zeitraubende Aufgaben. Nach dem Stand der Technik können Ergebnisse erst erzielt werden, nachdem ein echter Hardware-Prototyp verfügbar ist. Zusätzlich ist

der Aufwand für Tests und für die Performance-Analyse verschiedener, auf echten Hardware-Prototypen basierender Netzkonfigurationen zeitaufwendig und kann zu langen Design-Iterationen führen. Dieser Artikel zeigt einen Ansatz für die Verifikation und Performance-Analyse von verteilten eingebetteten Systemen – basierend auf virtuellen Prototypen –, der eine frühe und flexible Integration der zu Grunde liegenden Hardware-Plattform erlaubt.

Das MOST High Protocol (MHP) dient der Paket-Datenübertragung in MOST-Netzwerken. Es ist ein verbindungsorientiertes Netzwerkprotokoll, welches einige Mechanismen des Übertragungs-Kontrollprotokolls (TCP) verwendet. Das MHP-Protokoll liegt zwischen den Netzwerkdiensten und der Applikationsschicht eines MOST-Gerätes. Der Kommunikationsprozess zwischen Geräten schließt Verbindungseinrichtung, Datenübertragung und Verbindungsbeendigung ein. Nach der Datenübertragung bestätigt der Empfänger dem Sender den korrekten Empfang; andernfalls werden die Daten erneut angefordert.

Für das Testen und die Performance-Analyse des MHP-Protokolls wurde eine auf virtuellen Prototypen basierende Simulationsumgebung entwickelt. Der virtuelle Prototyp kapselt den MHP-Quellcode in einem virtuellen Gerät

und stellt eine Verbindung zu anderen virtuellen MHP-Geräten her. Der virtuelle Prototyp wird unter Verwendung von SystemC implementiert, um eine genaue Simulation des zeitlichen und des funktionellen Verhaltens darstellen zu können. SystemC ist eine auf C++ basierende Modellierungs- und Simulationssprache, die Nebenläufigkeiten und Zeitaspekte während der Simulation von eingebetteter Software und der zu Grunde liegenden Hardware-Plattform abdeckt. Die virtuellen Geräte werden aus verschiedenen Modulen zusammengesetzt, die über Software-Schnittstellen miteinander verbunden werden. Diese Module implementieren den gekapselten MHP-Protokoll-Stack, die Kommunikationsmodule und die Anwendung.

Jedes virtuelle Gerät lässt sich aus unterschiedlichen Modulen zusammensetzen, um z.B. die Verbindung zwischen verschiedenen MHP-Versionen während eines Simulationslaufes zu verifizieren. Die Verbindung der verschiedenen Geräte erfolgt unter Verwendung von „Transaction Level Modelling“-Techniken (TLM-2.0). TLM ist ein High-Level-Ansatz für die abstrakte Kommunikationsmodellbildung zwischen Modulen. Die Konfiguration der Geräte und des Netzwerkmodells erfolgt mit Hilfe von XML-Dateien. Die Verifikation und die Performance-Analyse der verschiedenen Geräte- und Netzwerkkonfigurationen werden mittels einer Automatic Test Pattern Generation (ATPG) unterstützt, indem automatisch verschiedene XML-Konfigurationen erzeugt werden. Nach der Simulation werden die Ergebnisse aufgezeichnet, um eine umfassende Analyse des kompletten Systems zu ermöglichen.

### ■ MHP-Netzwerksimulation

Die MOST-Netzwerksimulation lässt sich aus verschiedenen Geräten mit

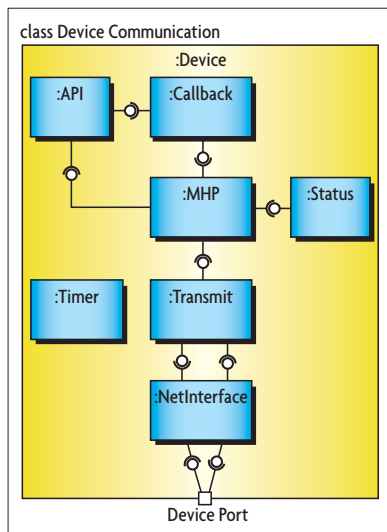
unterschiedlichster Konfiguration aufbauen. Die Geräteverbindung wird mittels TLM durchgeführt, um die Simulation zu beschleunigen. Die Kommunikation zwischen den Geräten und andere Status-Parameter werden für die Analyse und die automatische Überprüfung von Eigenschaften während der Simulation verfolgt. Zur Verifikation des Protokoll-Stacks wurde zusätzlich ein Host mit speziellem Funktionsumfang entwickelt. Der Host kann instanziierte Geräte stimulieren, indem er MHP-Nachrichten über das Netzwerk sendet. Erhaltene Nachrichten werden gespeichert und mit vordefinierten Kommunikationssequenzen verglichen.

### ■ Kapselung des MHP-Quellcodes

Der bereitgestellte Quellcode des MHP-Protokoll-Stacks liegt in C vor. Um eine mehrfache Instanzierung für die Netzwerksimulation zu ermöglichen, wurde der C-Quellcode des Protokoll-Stacks in der so genannten MHP-Klasse gekapselt. Die MHP-Klasse ist das zentrale Element in jedem virtuellen Gerät. Um den MHP-Code mit z.B. Statusdaten oder Zeitfunktionen zu versorgen, wurden zusätzliche Klassen entwickelt und mit der MHP-Klasse verbunden. Das Hauptziel der Kapselung war, die Anforderungen für die mehrfache Instanzierung mit minimaler Modifizierung des gelieferten Quellcodes zu erfüllen. Das stellt sicher, dass sich neue MHP-Versionen ohne viel Aufwand einbinden lassen. Dies wurde erreicht, indem lediglich zwei Blöcke, welche die Klassendefinitionen einschließen und über Makros C-Funktionsaufrufe mit C++-Member-Funktionsaufrufen ersetzen, innerhalb des Codes eingefügt wurden.

### ■ MHP-Geräte-Modellierung

Für die Simulation ist es nicht ausreichend, den gegebenen C-Code des MHP-Protokoll-Stacks zu kapseln. Die MHP-Klasse muss den Status der Knoten kennen, und eine Applikation wird benötigt, um spezifische Verifikations- und Evaluierungsszenarien zu integrieren. Außerdem ist eine Kompo-



■ Bild 1. Struktur eines virtuellen Gerätes.

nente zur Integration der erforderlichen Zeitfunktionen notwendig.

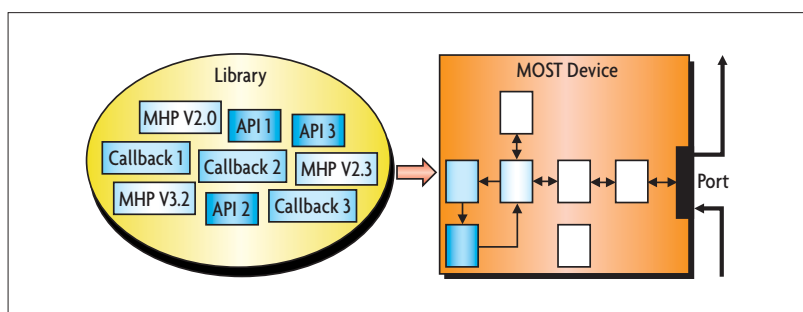
Um den Knotenstatus an die MHP-Klasse übergeben zu können, wird die so genannte Status-Klasse verwendet. Diese Klasse stellt Funktionen zur Verfügung, die sich über definierte Schnittstellen der MHP-Klasse aufrufen lassen. Die Applikation wird in eine so genannte API-Klasse integriert, die eine Schnittstelle zur MHP-Klasse bereitstellt, um z.B. Daten versenden zu können. Die integrierte Callback-Klasse verbindet die MHP-Klasse mit der Applikation, so dass z.B. empfangene Pakete an die API weitergereicht

zwischen dem elektronischen Host-Controller (EHC) und dem Intelligent Network Interface Controller (INIC) zu simulieren. Diese werden Transmit- und NetInterface-Klasse genannt. Alle erwähnten Klassen werden in der Device-Klasse zusammengefasst. Um die Verbindung mit anderen Geräten zu modellieren, schließt die Device-Klasse einen TLM-2.0-Port ein. Bild 1 zeigt die Struktur eines virtuellen Gerätes einschließlich aller erforderlichen Klassen und Schnittstellen.

Verschiedene API- und Callback-Komponenten können instanziiert und jeweils aus einer Standard-API- oder Callback-Klasse abgeleitet werden, um Bauteile verschiedenen Verhaltens zu simulieren. Die Auswahl einer der entwickelten API- oder Callback-Komponenten aus einer Bibliothek wird in der Konfigurationsdatei für jedes Gerät definiert (Bild 2). Bei Verwendung dieser Strategie ist es auch möglich, verschiedene Versionen des MHP-Protokoll-Stacks zusammen im virtuellen Netz zu integrieren und zu testen.

### ■ Geräteverbindung

Um mit anderen Geräten zu kommunizieren, schließt jedes Gerät einen TLM-2.0-Port ein. Die Datenübertragung über die TLM-Ports wird abstra-

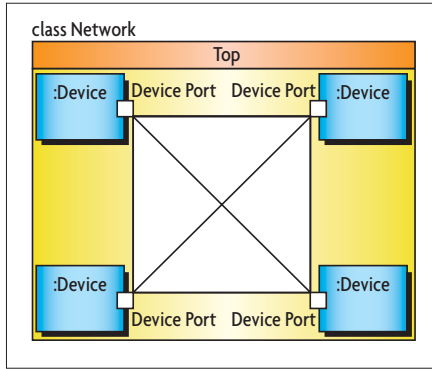


■ Bild 2. Konfiguration des virtuellen Gerätes.

werden können. Das Zeitverhalten eines kompletten Knotens wird durch die Timer-Klasse gesteuert. Der MHP- und der API-Klasse ist es erlaubt, Timer zu registrieren und zu bedienen. Tritt ein Timer-Interrupt auf, wird eine definierte Funktion aufgerufen.

Zwei zusätzliche Klassen, die zwischen der MHP-Klasse und den Verbindungs-Ports liegen, wurden entwickelt, um einen möglichen Engpass

hiert. Das bedeutet, dass nicht jedes Bit eines Frame simuliert wird, es werden gleich ganze MHP-Frames zwischen verschiedenen Geräten übertragen. Zusätzlich wird die Übertragungszeit von MHP-Nachrichten zwischen Geräten zu Null abstrahiert. Die Ports der Bauteile werden durch die so genannte „many to many binding“-Technik miteinander verbunden (Bild 3), um die Simulation zu be-



**Bild 3. Vereinfachte Verbindungsstruktur zwischen verschiedenen Geräten.**

schleunigen. Diese Abstraktionen sollten aufgrund der hohen Bandbreite von MOST eine minimale Wirkung auf die Genauigkeit während der Performance-Analyse haben. Der Hauptengpass wird durch die Datenübertragung zwischen EHC und INIC verursacht.

## Netzwerkconfiguration

Das Simulationsmodell wird durch XML-Dateien konfiguriert, die vor Beginn der Simulation geladen werden. Die Konfigurationsdatei schließt Simulation, Host und Gerätekonfiguration ein. Die Simulationskonfiguration definiert die Simulationszeit und instanziiert einen optionalen Host und bis zu 64 Geräte. Die Host-Konfiguration enthält Send- und Empfangsnachrichten zusammen mit Adressen, Timing- und Sequenz-Beschränkungen zur Überprüfung. Die Gerätekonfiguration wählt die Version der Klassen API, Callback und MHP aus einer Bibliothek und konfiguriert die Klassen selbst. Die MHP-Klasse wird z.B. mit der Anzahl von Send- und Empfangsverbindungen und der Paketgröße für die Datenübertragungen durch das MHP-Protokoll konfiguriert. Weitere Geräteparameter werden für den Eingangs- und den Ausgangspuffer auf die API-Konfiguration verwendet.

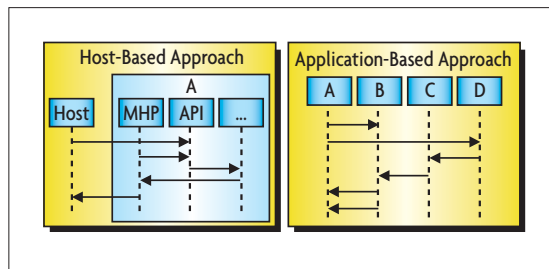
## Aufzeichnung und Überprüfung

Jeder über die TLM-Kanäle ausgetauschte Frame wird zusammen mit einem Zeitstempel in einer Protokolldatei gespeichert. Zusätzlich werden entdeckte Paketverluste, die analysierte

Kommunikationsbandbreite, die Zahl von Sendewiederholungen und andere Parameter für die spätere Analyse gespeichert. Im Vergleich zur realen Hardware bietet die Simulation bessere Möglichkeiten zur Beobachtung. Mit einem Debugger ist es möglich, das Modell Schritt für Schritt auszuführen. Zudem lassen sich so Variablen beobachten und auch verändern. Für die Verifikation wird zusätzlich zu den Geräten ein optionaler Host instanziiert, der die Überprüfung während der Simulation durchführt. Der Host stimuliert instanziierte Geräte, indem er Nachrichten sendet. Die empfangenen Nachrichten werden mit den zu erwartenden verglichen.

## MHP-Verifikation und Performance-Analyse

Für die Protokollüberprüfung lässt sich der zusätzliche Host instanziiert, um das Netzwerk zu stimulieren, indem er entweder MHP-Nachrichten über das



**Bild 4. Host-basierter und applikationsbasierter Simulationsansatz.**

Netz sendet oder beispielsweise die API eines Gerätes direkt stimuliert. Für die Performance-Analyse wird kein Host verwendet. Verschiedene Anwendungen übernehmen die Lastgenerierung (Bild 4).

## MHP-Verifikation

Für die Protokoll-Verifikation wird der host-basierte Ansatz verwendet. Die Stimuli und die Überprüfungseinschränkungen für den Host werden durch die XML-Konfigurationsdatei beschrieben. Die Überprüfung schließt die Korrektheit des Frames selbst, Sequenzbedingungen unter Berücksichtigung von bedingten Ausdrücken und

Zeitgrenzen für Netzwerknachrichten sowie Funktionsaufrufe innerhalb eines Bauteils ein. Wird durch den Host ein Fehler erkannt, wird die Simulation angehalten. Die Fehlersteuerung liefert eine Fehlermeldung zurück, und die Protokolldatei endet mit dem unerwarteten Frame. Die Testbeschreibung wird durch UML-Sequenz- und Strukturdiagramme modelliert. Das Strukturdiagramm beschreibt die Verbindungsstruktur der verwendeten Geräte und den Aufbau eines Gerätes selbst. Das Sequenzdiagramm beschreibt die Kommunikation zwischen den Geräten, die während der Simulation überprüft werden. Die Struktur- und die Sequenzdiagramme werden mit Hilfe von Enterprise Architect (EA) modelliert und in ein spezifisches XML-Format für den Datenaustausch, genannt XML Metadata Interchange (XMI), exportiert. Die Information innerhalb der XMI-Datei wird analysiert und eine Konfigurationsdatei für das Simulationsmodell erzeugt.

Um den Verifikationsprozess zu automatisieren, werden mittels einer ATPG viele verschiedene Testfälle erzeugt. Dabei ist es möglich, verschiedene Parameter aus vordefinierten Intervallen zufällig mit verschiedenen Zufallsverteilungen zu erzeugen. Diese Testfälle lassen sich nach der Erzeugung automatisch ausführen. Wird ein Fehler erkannt, kann der fehlerhafte Testfall für weitere Analyseschritte nach den Testläufen markiert werden.

## MHP-Performance-Analyse

Für die Performance-Analyse kommt der applikationsbasierte Ansatz zum Einsatz. Dafür wurden verschiedene Applikationen entwickelt, welche als Lastgenerator dienen – z.B. sendet eine Applikation Datenpakete von definierter Größe zu einem definierten Zeitpunkt, eine andere Anwendung sendet Datenpakete mit einer definierter Größe in einem definierten Zeitfenster. Die Send- und Empfangsbandbreite jedes Bauteils und andere

Parameter wie beispielsweise Paketverluste, Sendewiederholungen und Pufferüberläufe werden für die Analyse gespeichert.

Um eine ausreichende Parameterabdeckung der Analyse zu erhalten, müssen zahlreiche Simulationsläufe für die Performance-Analyse durchgeführt und analysiert werden. Verschiedene

Testscenarien, z.B. vier sendende und ein Empfangsgerät, müssen hierfür definiert werden. Für jedes dieser Testscenarien werden mehrere Konfigurationsdateien mit zufälligen Konfigurationsparametern durch eine ATPG erzeugt. Diese Konfigurationen lassen sich automatisch durch das SystemC-Modell ausführen. Die entsprechenden

Aufzeichnungen während der Simulation werden nach der Simulation gespeichert. Im nächsten Schritt werden die Aufzeichnungsdateien analysiert und Statistiken für jedes Szenario erzeugt. Die Worst-Cases werden gekennzeichnet und können innerhalb des SystemC-Modells für die weitere Analyse wieder ausgeführt werden. *bg*



**Dipl.-Ing. Andreas Braun**

ist wissenschaftlicher Mitarbeiter in der Abteilung Systementwurf in der Mikroelektronik (SiM) am FZI Forschungszentrum Informatik in Karlsruhe.



**Dr. rer. nat. Oliver Bringmann**

ist Bereichsleiter im Forschungsbereich Intelligent Systems and Production Engineering am FZI Forschungszentrum Informatik in Karlsruhe.



**Prof. Dr. Wolfgang Rosenstiel**

ist Professor am Wilhelm-Schickard-Institut der Universität Tübingen und Direktor am FZI Forschungszentrum Informatik in Karlsruhe.