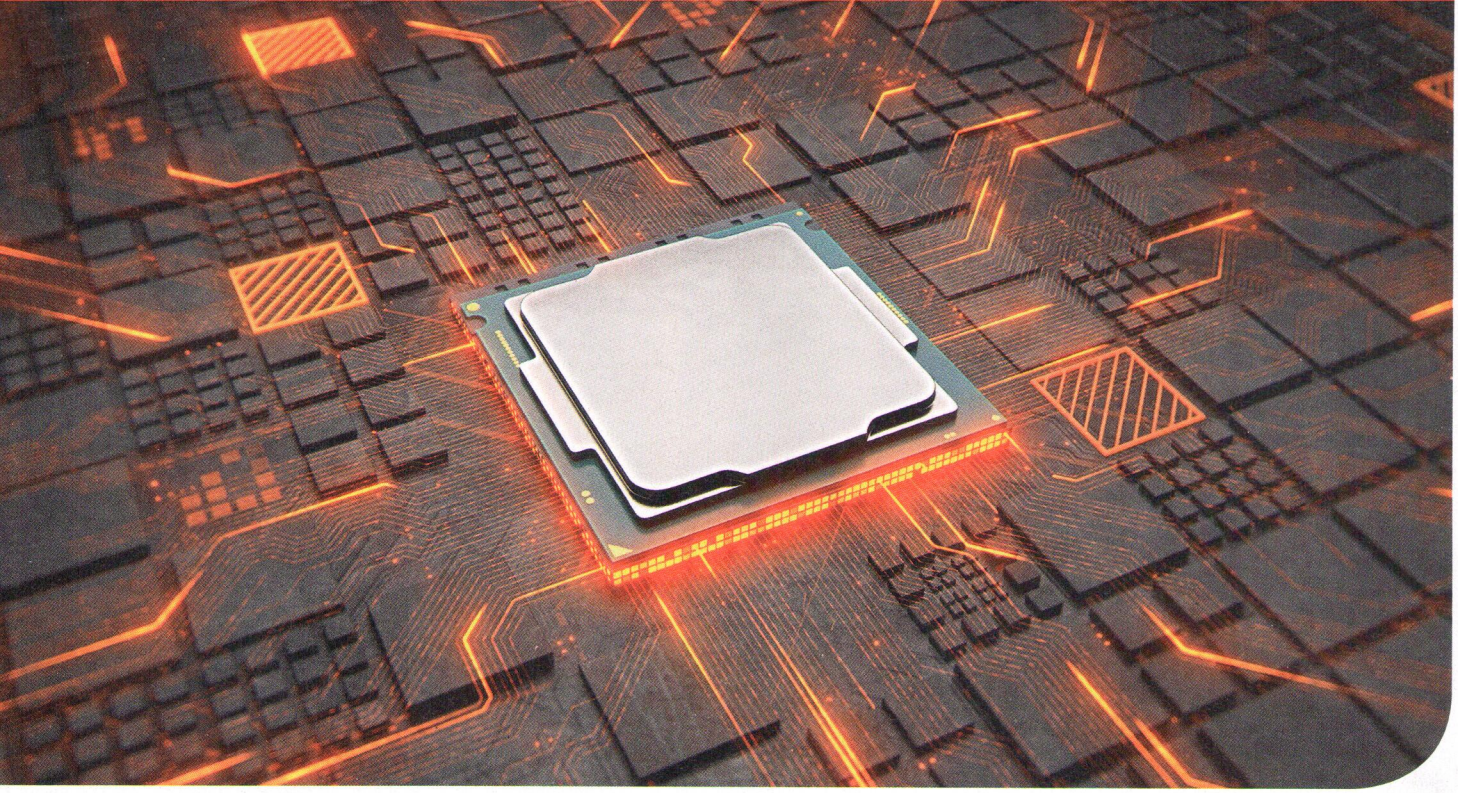


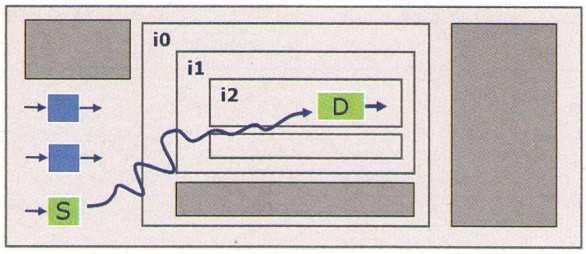
Q.E.D.

Automatisierte Konnektivitätsprüfung. Die Verbindungen zwischen den Designblöcken und I/O-Zellen komplexer Chips lassen sich nicht mittels Inspektion verifizieren. Simulation und Emulation können zwar Bugs aufspüren, bleiben aber unvollständig. Formale Tools dagegen finden nicht nur alle Fehler – sie können das auch beweisen.

Die Konnektivitätsüberprüfung gehört zu den häufigsten Anwendungen formaler Techniken (mehr dazu im **Wissenskasten**). Der Zweck dieser Verifikationsaufgabe lässt sich sehr einfach formulieren: Es soll sichergestellt werden, dass zwischen den Designblöcken und I/O-Zellen die richtigen Verbindungen existieren. Dies klingt simpel, stellt aber in Wirklichkeit eine enorme Herausforderung dar, denn ein modernes SoC enthält komplexe Subsysteme, die aus Tausenden Instanzen hochgradig konfigurierbarer Module und IP-Blöcke zusammengesetzt sind. Programmierbare Elemente sorgen für Variabilität und Anpassungsfähigkeit, während gemultiplexte I/O-Pads den Anwendern die Kontrolle darüber geben, auf welchen Pins welche Protokolle laufen. Es können einige Hunderttausend Verbindungswege vorhanden sein, von denen jeder einzelne wichtig für die korrekte Funktion des Chips ist.

Signale, die es zu verbinden gilt, können mehrere Blöcke und Hierarchieebenen durchlaufen (**Bild 1**). Da der Signalweg auch Inverter enthalten kann, gilt es, die Polarität der Signale im Auge zu behalten. Ebenso kommen Zustandselemente wie Register oder Flipflops vor, die zwischen Start- und Endpunkt Verzögerungen von mehreren Zyklen bewirken. Einige globale Signale wie Takte, Reset-Signale und Scan-Enables werden an Tausenden oder gar Millionen von Zustandselementen geführt, und auch die Korrektheit dieser Verbindungen sollte verifiziert werden. All diese Aspekte sowie die schiere Anzahl der zu prüfenden Verbindungen sorgen dafür, dass die Konnektivitätsverifikation per Inspektion nicht praktikabel ist.

Die Simulation oder Emulation der Konnektivität ist praktikabler, aber prinzipbedingt unvollständig. Eine Testsuite, die alle erforderlichen Pfade überdeckt, wäre mühsam zu schreiben, schwierig zu pflegen, wenn sich das Design weiterentwickelt, und Zeit raubend in der Ausführung. Das Debugging aufgedeckter Fehler ist nicht trivial, denn die Symptome eines Bugs müssen zurückverfolgt werden, um die unkorrekte oder fehlende Verbindung zu finden. Weder die Simulation noch die Emulation kann irgendeinen Korrektheitsnachweis liefern, auch wenn die Testsuite noch so gut ist. Genau hierin liegt der Grund für die große Verbreitung der formalen Technik: Ein formales Tool kann potenziell sämtliche Verbindungsfehler finden und nach Beseitigung sämtlicher



1 | Signalweg: Zwischen Quelle und Ziel einer Verbindung können verschiedene Hierarchieebenen liegen

Bilder: temp-64GTX - stock.adobe.com, OneSpin

Bugs nachweisen, dass die Konnektivität vollständig ist.

Formale Konnektivitätsprüfung der traditionellen Art

Die formale Verifikation benötigt Eigenschaften, anhand derer sie das Design prüfen kann. Man kann sich leicht vorstellen, mithilfe von SystemVerilog Assertions eine Reihe von Eigenschaften zu formulieren und damit die Signale zu spezifizieren, die es zu verbinden gilt. Bei kleinen Designs kann dies tatsächlich ein gangbarer Weg sein. Tausende solcher Eigenschaften zu formulieren, wäre dagegen eindeutig zu viel, und genau hier kann eine formale App zur Konnektivitätsprüfung helfen. Da die Struktur von Konnektivitätseigenschaften ziemlich regelmäßig ist, lassen sich diese automatisch generieren, wenn eine Spezifikation der vorgesehenen Konnektivität vorgelegt wird. Dem formalen Tool wird diese Spezifikation meist in Gestalt eines traditionellen Spreadsheets zur Verfügung gestellt (**Bild 2**).

Das Ausfüllen dieses Spreadsheets ist für die Anwender wesentlich einfacher als das Schreiben von Assertions. Die Felder enthalten die Quelle und das Ziel eines jeden Verbindungswegs und geben an, um wie viele Zyklen das Signal auf seinem Weg verzögert wird und unter welchen Bedingungen der Pfad freigegeben werden sollte. Schließlich ist auch der relevante Takt aufgeführt. Die Freigabebedingung ist insbesondere für jene Pfade wichtig, die Multiplexer enthalten. Dies können beispielsweise I/O-Pfade sein, die unter verschiedenen Bedingungen mehrere Verbindungen unterstützen. Mithilfe der Angaben im Spreadsheet kann ein formales Tool ohne jede manuelle Spezifikation selbstständig alle erforderlichen Eigenschaften generieren. Eine Kombination aus struktureller Analyse und formalen Proof Engines deckt sämtliche Bugs im Design beziehungsweise alle Fehler im Spreadsheet auf und weist anschließend die vollständige Konformität mit der Spezifikation nach.

```
# Source          # Destination     # Delay # Condition          # clock
top.a.b.c,        top.x.y.z
top.i1_m_s_bit.x, top.i1_m_d_bit.y, 1,       top.debug_en==1'b0, top.clk
top.i2_m_s_bit.x, top.i2_m_d_bit.y, 2:3,     top.debug_en==1'b0, top.clk
top.i1_m_s_bot.x, top.i1_m_d_bot.y, 5,       top.debug_en==1'b0, top.clk
top.i2_m_s_bot.x, top.i2_m_d_bot.y, 2,       top.debug_en==1'b0, top.clk
```

2 | Spreadsheet I: Die beabsichtigte Konnektivität lässt sich auf diese Weise spezifizieren

```
# Source          # Destination
i:top.a.b, sig_c,  m:m_x,     sig_y
m:m_a,     sig_b,  i:top.x,   sig_y
m:m_s_b?p, sig_a,  m:m_d_b?p, sig_x
m:m_s_b?t, sig_b,  m:m_d_b?t, sig_y
m:m_s_b?o, sig_c,  m:m_d_b?o, sig_z
m:m_s_*t,  sig_d,  m:m_d_*t,  sig_x
```

3 | Spreadsheet II: Die abstrakte Konnektivitäts-Spezifikation ist wesentlich knapper

Wie bei verschiedenen anderen formalen Apps, wird auch die Konnektivitätsüberprüfung routinemäßig an kompletten Chipdesigns vorgenommen. Dies ist notwendig, weil die Gesamtheit der zu verifizierenden Verbindungen nur auf der obersten Ebene sichtbar ist. Während formale Tools in ihrer Kapazität beschränkt sind, lassen sich Konnektivitätsprüfungen auch an großen Chips durchführen, da für das jeweils anstehende Problem immer nur ein kleiner Teil des Designs relevant ist. Um die Analyse zu beschleunigen, wird Logik, die nicht mit dem Problem in Zusammenhang steht, beim Erstellen des formalen Modells vernachlässigt. Allerdings reizen die sehr umfangreichen, heterogenen Computingplattformen von heute und ähnliche SoC die Kapazität traditioneller formaler Tools in hohem Maße aus. Hinzu kommt, dass das Befüllen eines Spreadsheets unrealistisch wird, wenn es nicht mehr um Tausende, sondern um Hunderttausende Zellen geht. Somit steht außer Frage, dass die traditionelle Art der formalen Konnektivitätsüberprüfung weiterentwickelt werden muss.

KONTAKT

OneSpin Solutions GmbH,
Nymphenburger Straße 20a,
80339 München,
Tel. 089 990130,
E-Mail info@onespin-solutions.com,
www.onespin-solutions.com

Die Lösung: Connectivity XL

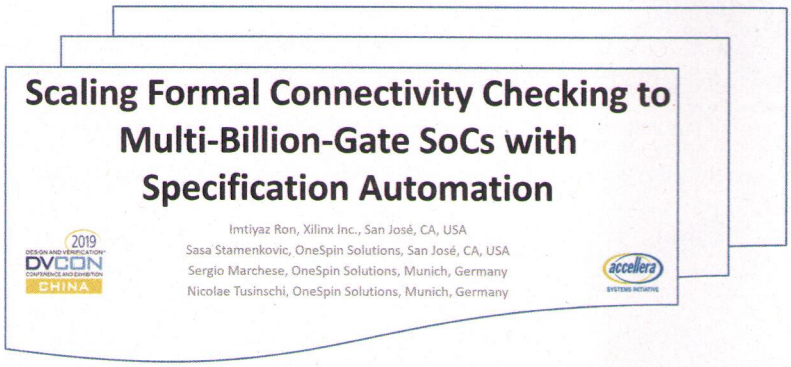
Mit Connectivity XL steht eine neue Methodik für die Konnektivitätsverifikation zur Verfügung, die auf die Herausforderungen sehr umfangreicher SoC-Designs ausgerichtet ist. Zu den entscheidenden Neuerungen gehört, dass das Spezifizieren der beabsichtigen Konnektivität abstrahiert wird. Wie man in **Bild 3** sieht, bleibt das Spreadsheet als Eingabemedium erhalten, jedoch lässt sich die Spezifikation dank der Verwendung von Wildcards wesentlich knapper fassen. Schließlich ist es an der Tagesordnung, dass bestimmte Blöcke mehrfach vorkommen und nach einem einheitlichen Schema benannt werden. Deshalb lässt sich die Zahl der benötigten Spreadsheet-Zeilen mithilfe von Wildcards entscheidend verringern, was wiederum den Zeitaufwand zum Spezifizieren der angestrebten Konnektivität von Monaten auf Tage reduzieren kann.

Ein formales Tool kann die abstrakte Verifikation lesen, gemeinsam mit dem Design kompilieren und die Wildcards so erweitern, dass ein traditionelles Konnektivitäts-Spreadsheet mit einer Verbindung pro Zeile entsteht. Allerdings umfasst diese Spezifikation möglicherweise einige Hunderttausend Zeilen, sodass ein traditionelles Konnektivitätsprüfungstool an

FAZIT

Für künftige Komplexität gerüstet. Die zunehmende Größe und Komplexität integrierter Schaltungen machen formale Apps immer nützlicher – vor allem für die Konnektivitätsüberprüfung. Simulation, Emulation oder manuelle Techniken können hier keinesfalls mithalten, und selbst traditionelle formale Tools sind nicht hinreichend skalierbar. Connectivity XL ist dieser Aufgabe dank mehr Kapazität und Automatisierung gewachsen. Die Tauglichkeit der App wurde an einem realen SoC-Design mit mehreren Milliarden Gattern nachgewiesen. Auch wenn die Designs weiter an Umfang zunehmen, ist diese neue Kategorie formaler Tools dafür gerüstet, auf Jahre hinaus eine gangbare Lösung zu bieten.

Bilder: OneSpin



4 | DVCon China 2019: Auf der Konferenz wurde eine Konnektivitäts-Fallstudie präsentiert

seine Grenzen stoßen dürfte. Fortlaufende Verbesserungen in den zugrundeliegenden formalen Algorithmen von Connectivity XL unterstützen jedoch immer längere Konnektivitätsspezifikationen für immer umfangreichere SoC-Designs. Maschinelles Lernen auf der Basis jahrelanger Erfahrung in der formalen Technik wird ins Spiel gebracht, um die beste Proof Engine für die anstehende Aufgabe zu finden.

Automatische Abstraktionen reduzieren das formale Modell auf die minimal benötigte Logik, was die Laufzeit verkürzt und den Speicherbedarf reduziert. Eine weitere Neuheit von Connectivity XL ist das Zusammenfassen der strukturellen und der formalen Analyse im Interesse größtmöglicher Wirksamkeit. Im Rahmen der Erzeugung detaillierter Spezifikationen detektiert diese Analyse automatisch Verzögerungen und Inverter in den Verbindungspfaden und leitet Multiplexing-Bedingungen her. Insgesamt bietet Connectivity XL einen stärker automatisierten Ablauf als traditionelle Konzepte, kommt mit größeren Designs zurecht und produziert umfassende Nachweise – auch bei äußerst komplexen Chips.

Reale Verifikationsresultate

Auf der DVCon China 2019 (Design and Verification Conference) präsentierten Xilinx und OneSpin eine Fallstudie (**Bild 4**) zur Anwendung der Connectivity-XL-App von OneSpin an einem SoC mit mehreren Milliarden Gattern. Der auf 7-nm-Technologie basierende Chip enthielt 60 Millionen Instanzen von 35 000 Modulen, 90 Millionen Flipflops und 80 000 Zustandsautomaten. Als eines der umfangreichsten Designs der Welt brachte dieses Projekt viele Tools in den Design- und Verifikationsabläufen an ihre Grenzen – auch die Konnektivitätsüberprüfung, denn es galt, mehr als eine Million Verbindungen zu spezifizieren, über die Designiterationen hinweg zu pflegen und zu verifizieren.

Das Verifikationsteam probierte mehrere traditionelle Konnektivitäts-Apps – darunter auch die von OneSpin – aus, doch keine ließ sich auf diesen großen Chip skalieren. Der Arbeitsaufwand zum Spezifizieren und Pflegen von mehr als einer Million Verbindungen war schlicht inakzeptabel. Die Laufzeiten der formalen Tools waren viel zu lang, und allzu häufig wurden keine konklusiven Nachweise produziert. Angesichts des eng gesteckten Design-Zeitplans waren Abstriche an der Qualität ausgeschlossen, und eine gründliche Verifikation wurde als kritisch erachtet. Connectivity XL erwies sich als der Aufgabe gewachsen. Dank des abstrakten Spreadsheet-Formats reduzierte sich dessen Umfang auf weniger als ein Hundertstel, und gleichzeitig vereinfachte sich die Pflege der Verbindungsliste.

WISSENSWERT

Die formale Verifikation galt stets als eine anspruchsvolle Technik, mit der Experten einzelne Logikblöcke oder allenfalls deren kleinere Cluster gründlich verifizieren können. Der Reiz formaler Techniken besteht darin, dass sich damit alle möglichen Verhaltensweisen des zu verifizierenden Designs umfassend analysieren lassen. Hierin unterscheiden sich formale Verifikationen grundlegend von Simulationen, die nur einen Bruchteil des möglichen Verhaltens aktivieren, indem sie bestimmte Tests ausführen. Wird ein Fehler im Design von keinem der Tests angestoßen, wird er auch nicht gefunden. Unentdeckt bleibt der Bug ebenfalls, wenn er zwar angestoßen wird, aber keine Änderung der Ergebnisse bewirkt. Sofern ihnen ein hinreichend robuster Bestand an Eigenschaften zur Verfügung steht, die das gewünschte Verhalten beschreiben, können formale Tools nicht nur sämtliche Bugs finden, sondern auch nachweisen, dass es keine weiteren unentdeckten Fehler gibt.

Heutzutage können wesentlich mehr Anwender, von denen viele keine Experten sind, von den Fähigkeiten der formalen Verifikation profitieren. Diese erheblich gewachsene Verbreitung der formalen Verifikation hat mehrere Gründe. So hat der umfangreiche Einsatz standardisierter Formate, vor allem der SystemVerilog Assertions (SVA), dazu geführt, dass für das Schreiben formaler Eigenschaften deutlich weniger Fachwissen erforderlich ist. Die modellbasierte Mutationsüberdeckung kann die nicht von Assertions überdeckten Teile des Designs identifizieren und den Anwendern damit wertvolle Hinweise liefern. Formale Tools bringen heute eine vermehrte Automatisierung und mehr simulationsähnliche Debug-Features mit, wodurch sie einfacher anzuwenden sind. Regelmäßig sorgen Durchbrüche hinsichtlich der Leistungsfähigkeit formaler Algorithmen dafür, dass sie für große Blöcke und Cluster geeignet sind, die noch vor wenigen Jahren ausgeschlossen waren.

Der wichtigste Grund für den vermehrten Einsatz der formalen Verifikation ist jedoch, dass die Mehrzahl der Anwender mit Applikationen (Apps) arbeitet, die für bestimmte Verifikationsaufgaben ausgelegt sind. Apps generieren in der Regel die meisten oder gar alle Eigenschaften, die für die formale Analyse benötigt werden, wobei die Features der Algorithmen und Tools genau auf die Zielapplikation ausgerichtet sind. Das Resultat ist eine Push-Button-Lösung, die auch Anwender ohne Erfahrung in der formalen Verifikation mit wenig Einarbeitung verwenden können. Die Apps sind darüberhinaus so effizient, dass viele selbst bei sehr umfangreichen SoC-Designs auf den kompletten Chip angewendet werden. Unverändert ist es das Ziel, sämtliche Bugs zu finden und den Nachweis zu führen, dass alle Fehler gefunden wurden – allerdings nur jene Fehler, die mit der jeweils vorliegenden Aufgabe in Zusammenhang stehen.

Bild: OneSpin

Connectivity XL fand eine ganze Reihe komplexer Designfehler, die sich mit anderen Tools oder Methoden nur sehr schwer hätten aufdecken lassen: unter anderem fehlerhafte Blockintegrationen, die Freigabe mehrerer Treiber an einem Pfad und rekonvergente Pfade. Die zur Verfügung gestellten Debuginformationen ermöglichten eine einfache Ursachenermittlung selbst bei Pfaden mit mehr als zweitausend Signalen zwischen Quelle und Ziel. Sobald die gefundenen Probleme behoben waren, ließen sich alle der mehr als eine Million Verbindungen in wenigen Tagen überprüfen, wobei jeweils mehrere Jobs parallel abgearbeitet werden konnten. Für keine der Verbindungen ergaben sich nicht konklusive Ergebnisse. ml

Autor
Tom Anderson ist Technical Marketing Consultant bei OneSpin Solutions.

Online-Service
Info: Connectivity XL und weitere spezialisierte Apps dieses Anbieters

www.elektronik-informationen.de/84049

Neues Software-Release verbessert Versionsverwaltung und Codeverteilung

National Instruments hat die neuen Versionen seiner Softwareplattformen Labview und Labview NXG herausgebracht. **Labview 2019** hilft dem Anwender, Versionen mit fragmentierten, nicht standardisierten Methoden und deren Abhängigkeiten zu verwalten. Die neue Verteilungsoption beinhaltet eine standardisierte Methode mit integrierter Versionsverwaltung sowie Abhängigkeitsmanagement und lässt sich nutzen, um Systemsoftware zu replizieren und zu teilen. Außerdem hat NI neue G-Datentypen eingeführt sowie den Überblick in der IDE und die Fehlerbehandlung verbessert.

Labview NXG 2019 soll vor allem die Entwicklung automatisierter Mess- und Prüfsysteme erleichtern. Die Codeimplementierung und -verteilung ist schnell umsetzbar. Matlab-Dateien können importiert und exportiert werden. Schnittstellen zu Software weiterer Drittanbieter sind ebenfalls integriert. pat

www.elektronik-informationen.de/84023

Verbesserter Designer und kostenloser Viewer

Altium bringt eine verbesserte Version seiner Designsoftware Altium Designer 19 und den kostenlosen **Altium 365 Viewer** auf den Markt. **Altium Designer 19.1** bietet laut Anbieter eine optimierte Produktperformance und Stabilität. Aufgrund der Verbesserungen treten weniger Fehler auf, und es erhöht sich die Effizienz im Designprozess.

Der Altium 365 Viewer ist ein kostenloses Tool, um Dateien aus dem Designer anzusehen. Bislang war dafür eine Lizenz nötig. Der Viewer visualisiert die PCB-Designs in 2D sowie 3D und ermöglicht es, diese mit anderen zu teilen. Er verknüpft außerdem Materiallisten automatisch mit den dazugehörigen Informationen der Lieferkette. pat

www.elektronik-informationen.de/84020

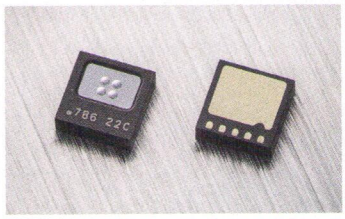
R-Car-Prozessor als Testinstrument

Göpel erweitert seine **Variotap**-Technologie zur universellen Prozessoremulation auf das Renesas-SoC R-Car V3M. Damit lässt sich dieser Bilderkennungsprozessor über den Debug-Port als designintegriertes Test- und Programmierinstrument verwenden. Das Variotap-Modell für R-Car V3M ist eine Softwareoption, welche die Nutzung der verschiedenen Controllermodule über die Debugschnittstelle ermöglicht. So kann angeschlossene Hardware getestet und validiert werden. Darüber hinaus lassen sich Flash-Speicher programmieren und Verbindungen zu externem RAM testen. dar

www.elektronik-informationen.de/84016

Kleiner Ferninfrarotsensor zur Körpertemperaturmessung

Melexis stellt den nach eigenen Angaben kleinsten FIR-Sensor **MLX90632** in einem SMD-SFN-Gehäuse mit Abmessungen von $3 \times 3 \times 1\,mm^3$ vor. Dieser misst die Körpertemperatur im medizinisch relevanten Temperaturbereich von 35 bis 42 °C mit einer maximalen Abweichung von ±0,2 °C. Somit eignet er sich beispielsweise für tragbare Diagnosegeräte oder klassische Stirn- und Ohrthermometer. Neben der Objekttemperatur wird auch die Umgebungstemperatur erfasst.



Auf der Oberseite des Sensors (links) befindet sich das FIR-Element mit einem Sichtfeld von 50°

Kompensationsalgorithmen gleichen die Empfindlichkeit gegenüber thermischem Rauschen aus und erzielen damit eine hohe Wärmestabilität. Eine Kalibrierung wird werkseitig vorgenommen, Anpassungen sind aber möglich. Der Wellenlängendurchlassbereich des optischen Filters liegt zwischen 2 und 14 µm. Dadurch beeinflusst sichtbares Licht der Umgebung nicht die Messung.

Die Versorgungsspannung des Bausteins beträgt 3,3 V. Ein externer Mikrocontroller kann den Sensor über eine I^2C-Schnittstelle mit entweder 3,3 oder 1,8 V Referenzspannung ansprechen, um auf die Messwerte im RAM oder auf Einstellungs- und Kalibrierungsparameter im EEPROM zuzugreifen. Die Aktualisierungsrate lässt sich dabei auf 16 ms bis 2 s festlegen, standardmäßig beträgt sie 0,5 s. pat

www.elektronik-informationen.de/83083

Bild: Melexis