Embedded COMPUTING DESIGN

JAN/FEB 2017
VOLUME 15 | 1
EMBEDDED-COMPUTING.COM

TRACKING TRENDS
2016 takeaways, 2017 trends to watch
PG 5

IoT INSIDER
Bluetooth and market trends in mesh networking
PG 6

DEVELOPMENT KIT SELECTOR
Digi-Key ELECTRONICS
www.embedded-computing.com/designs/iot_dev_kits

2017 EMBEDDED PROCESSOR REPORT:
AT THE EDGE OF MOORE'S LAW AND IoT
PG 10

REAL-TIME LINUX: ORIGINS AND IMPACTS
PG 18

I3C: AN UPGRADED INTERFACE FOR A WORLD OF SENSORS
PG 14

AD LIST

PAGE	ADVERTISER
27	**COMMELL Systems Corporation** – Embedded Computing Modules
1	**Digikey** – Development Kit Selector
13	**dSPACE** – dSPACE MicroLabBox: Compact Power in the Lab
23	**Embedded World** – March 14-16, 2017 Nuremburg, Germany
7	**Micro Digital, Inc.** – SMX RTOS: Ideal for Your Project
32	**Supermicro** – Connecting the Intelligent World from Devices to the Cloud
17	**Wind River Systems, Inc.** – Dive into the Industrial IoT with Confidence
3	**WinSystems, Inc.** – Power to Perform

PAGE	EMBEDDED WORLD PROFILES
30	**Acromag** – Military
31	**Dolphin Interconnect Solutions Inc.** – Networking

SOCIAL

Facebook.com/Embedded.Computing.Design

@Embedded_comp

LinkedIn.com/in/EmbeddedComputing

Pinterest.com/Embedded_Design/

Instagram.com/Embedded Computing

youtube.com/user/VideoOpenSystems

EMBEDDED COMPUTING DESIGN ADVISORY BOARD

Ian Ferguson, ARM
Jack Ganssle, Ganssle Group
Bill Gatliff, Independent Consultant
Andrew Girson, Barr Group
David Kleidermacher, BlackBerry
Jean LaBrosse, Silicon Labs
Scot Morrison, Mentor Graphics
Rob Oshana, NXP
Jim Ready, Independent Consultant
Kamran Shah, Silicon Labs

Embedded COMPUTING DESIGN

EMBEDDED COMPUTING BRAND DIRECTOR Rich Nass rnass@opensystemsmedia.com
EMBEDDED COMPUTING EDITORIAL DIRECTOR Curt Schwaderer cschwaderer@opensystemsmedia.com
TECHNOLOGY EDITOR Brandon Lewis blewis@opensystemsmedia.com
CONTENT ASSISTANT Jamie Leland jleland@opensystemsmedia.com
AUTOMOTIVE CONTRIBUTOR Majeed Ahmed
AUTOMOTIVE CONTRIBUTING EDITOR Jeremy S. Cook
DIRECTOR OF E-CAST LEAD GENERATION AND AUDIENCE ENGAGEMENT Joy Gilmore jgilmore@opensystemsmedia.com
CREATIVE DIRECTOR Steph Sweet ssweet@opensystemsmedia.com
SENIOR WEB DEVELOPER Konrad Witte kwitte@opensystemsmedia.com
WEB DEVELOPER Paul Nelson pnelson@opensystemsmedia.com
DIGITAL MEDIA MANAGER Rachel Wallace rwallace@opensystemsmedia.com
CONTRIBUTING DESIGNER Joann Toth jtoth@opensystemsmedia.com

SALES/MARKETING

SALES MANAGER Tom Varcie tvarcie@opensystemsmedia.com (586) 415-6500
MARKETING MANAGER Eric Henry ehenry@opensystemsmedia.com (541) 760-5361
STRATEGIC ACCOUNT MANAGER Rebecca Barker rbarker@opensystemsmedia.com (281) 724-8021
STRATEGIC ACCOUNT MANAGER Bill Barron bbarron@opensystemsmedia.com (516) 376-9838
STRATEGIC ACCOUNT MANAGER Kathleen Wackowski kwackowski@opensystemsmedia.com (978) 888-7367
SOUTHERN CAL REGIONAL SALES MANAGER Len Pettek lpettek@opensystemsmedia.com (805) 231-9582
SOUTHWEST REGIONAL SALES MANAGER Barbara Quinlan bquinlan@opensystemsmedia.com (480) 236-8818
NORTHERN CAL STRATEGIC ACCOUNT MANAGER Sean Raman sraman@opensystemsmedia.com (510) 378-8288
ASIA-PACIFIC SALES ACCOUNT MANAGER Elvi Lee elvi@aceforum.com.tw
BUSINESS DEVELOPMENT EUROPE Rory Dear rdear@opensystemsmedia.com +44 (0)7921337498

OpenSystems media®

WWW.OPENSYSTEMSMEDIA.COM

PUBLISHER Patrick Hopper phopper@opensystemsmedia.com
PRESIDENT Rosemary Kristoff rkristoff@opensystemsmedia.com
EXECUTIVE VICE PRESIDENT John McHale jmchale@opensystemsmedia.com
EXECUTIVE VICE PRESIDENT Rich Nass rnass@opensystemsmedia.com
CHIEF TECHNICAL OFFICER Wayne Kristoff
GROUP EDITORIAL DIRECTOR John McHale jmchale@opensystemsmedia.com
VITA EDITORIAL DIRECTOR Jerry Gipper jgipper@opensystemsmedia.com
INDUSTRY EDITOR Jessica Isquith jisquith@opensystemsmedia.com
MANAGING EDITOR Jennifer Hesse jhesse@opensystemsmedia.com
ASSISTANT MANAGING EDITOR Lisa Daigle ldaigle@opensystemsmedia.com
SENIOR EDITOR Sally Cole scole@opensystemsmedia.com
ASSOCIATE EDITOR Mariana Iriarte miriarte@opensystemsmedia.com
CREATIVE PROJECTS Chris Rassiccia crassiccia@opensystemsmedia.com
FINANCIAL ASSISTANT Emily Verhoeks everhoeks@opensystemsmedia.com
SUBSCRIPTION MANAGER subscriptions@opensystemsmedia.com

CORPORATE OFFICE
16626 E. Avenue of the Fountains, Ste. 201 • Fountain Hills, AZ 85268 • Tel: (480) 967-5581

SALES AND MARKETING OFFICE
30233 Jefferson • St. Clair Shores, MI 48082

REPRINTS

WRIGHT'S MEDIA REPRINT COORDINATOR Wyndell Hamilton whamilton@wrightsmedia.com (281) 419-5725

POWER TO PERFORM

Rugged, reliable and resilient embedded computing solutions

WinSystems' embedded single board computers are designed to support a broad range of industry applications and challenging operational environments. From energy and transportation management, to industrial IoT and automation—our systems enable the collection, processing and transmission of real-time data requirements at the heart of your overall system.

Our full line of embedded computers, I/O cards, and accessories help you design smarter projects offering faster time to market, improved reliability, durability and longer product life cycles. From standard components to full custom solutions, WinSystems delivers world-class engineering, quality and unrivaled technical support.

Embed success in every application with ***The Embedded Systems Authority!***

EBC-C413
EBX-compatible SBC with Intel® Bay Trail E3845/E3825 Processor

SBC35-C398Q
Quad-Core Freescale i.MX 6Q Cortex A9 Industrial ARM® SBC

PX1-C415
PC/104 form factor SBC with PCIe/104™ OneBank™ expansion and latest generation Intel® Apollo Lake-I SOC processor

SCADA ENERGY IOT AUTOMATION TRANSPORTATION

Single Board Computers | COM Express Solutions | Power Supplies | I/O Modules | Panel PCs

WinSystems®
The Embedded Systems Authority

817-274-7553 | www.winsystems.com
ASK ABOUT OUR PRODUCT EVALUATION!
715 Stadium Drive, Arlington, Texas 76011

intel IoT Solutions Alliance

QUALITY ISO 9001:2008 CERTIFIED

CONTENTS

Embedded COMPUTING DESIGN

Jan/Feb 2017 | Volume 15 | Number 1

opsy.st/ECDLinkedIn @embedded_comp

COVER

Trends in the electronics industry have had a profound impact on the development of embedded microprocessors, from the slowing in Moore's Law to the advent of the Internet of Things (IoT). The result is an emphasis on low cost and power consumption rather than feeds and speeds, as well as development tools and methodologies geared towards rapid innovation.

WEB EXTRAS

> **Floored: CES 2017 recap**
By Rich Nass, Embedded Computing Brand Director and Brandon Lewis, Technology Editor
http://bit.ly/FLooredCES2017

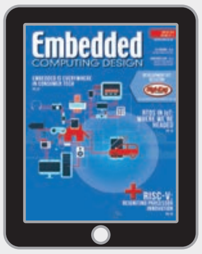
> **Discovering the technologies behind the autonomous vehicle**
By Rudy Ramos, Mouser Electronics
http://bit.ly/AutonomousDriveTech

> **Security and the Cortex-M MPU, part 4**
By Ralph Moore, Micro Digital
http://bit.ly/Cortex-MSecurity4

DOWNLOAD THE APP

Download the Embedded Computing Design app

iTunes: itun.es/iS67MQ
Magzter: opsy.st/ecd-magzter



Published by:

OpenSystems media

2017 OpenSystems Media®
© 2017 Embedded Computing Design

All registered brands and trademarks within *Embedded Computing Design* magazine are the property of their respective owners.

iPad is a trademark of Apple Inc., registered in the U.S. and other countries. App Store is a service mark of Apple Inc.

ISSN: Print 1542-6408
Online: 1542-6459

ENVIROINK

rdear@opensystemsmedia.com

2016 TAKEAWAYS, 2017 TRENDS TO WATCH

By Curt Schwaderer, Editorial Director

2016 was a year that saw the Internet of Things (IoT) trend hit critical mass. This critical mass was formed from previous attempts that resulted in a realization that IoT is a complex mash-up requiring embedded, enterprise, and network systems expertise in order to be successful. Such cross-functional teams rarely exist in one company. Some companies with compelling IoT business cases built a cross-functional team by using a combination of in-house technical expertise and augmenting with consulting firms that have skills in the other areas. 2016 saw the emergence of some of these consulting companies taking the experience gained from these early adopter IoT projects to launch IoT platforms aimed at solving the cross-functional skillset problem. These new IoT platforms set the stage for an increase in IoT systems participation.

New breed of engineers emerging

One interesting outgrowth of 2016 was the emergence of the "IoT engineer" version 1.0. This is an embedded engineer, cloud software developer, mobile app developer, or networking specialist that has gained knowledge and skills in one or more of the other disciplines within IoT. This phenomenon has historical precedence – the advent of the CPU saw the emergence of the "computer engineer" – one with a mix of skills in traditional electrical engineering, digital electronics, and software development.

LTE proliferation

The transition of mobile 3G technology to 4G/LTE caught stride in 2016. Mobile operators either started or continued deploying LTE networks in order to keep up with the growing speed and capacity demands of their subscribers. LTE deployments aren't something that only global operators are doing – a number of rural mobile carriers and cooperatives also spent significant time and resources moving their systems to or toward LTE. I attended a Rural Independent Network Alliance (RINA) wireless event in late summer, and many of the RINA members already had LTE migration activities underway.

Wi-Fi

At the end of 2015, many nationwide cable operators made announcements for aggressive plans to deploy Wi-Fi hot spots to extend customer reach and lay the foundation for additional subscribers and use cases. Wi-Fi deployments were significant throughout 2016 and found their way into increased localized analytics applications in the retail and hospitality industries. Perhaps the most important indicator for Wi-Fi is in the automotive industry, where the models released in 2016 were largely Wi-Fi hotspot enabled.

These three drivers in 2016 had significant impact in the medical, industrial, automotive, and consumer marketplaces. Sensors have dropped in price. Compute resources have also. Embedded devices are becoming connected through a variety of means. LTE and Wi-Fi availability provides cost-effective and geographic connectivity needed to develop and deploy IoT solutions. And cloud applications and services provide the application and analytics back end that extract the business insights to increase profitability, gain business insights, lower operational costs, and attract new customers.

2017 emerging embedded trends

Critical mass has gathered – will 2017 bring the massive wave of applications that begin to interconnect and automate every aspect of our lives? Likely not in a single year, but expect the trend to accelerate.

5G emerging

One significant driver is Japan's pledge to become completely 5G by the 2020 Olympics. This caused many mobile infrastructure vendors to announce LTE+ systems that add some features, capacity, and data rates of 5G. For those that don't know, 5G is the next generation of LTE and its primary driver is the capacity and speed characteristics to support massive machine-to-machine (M2M) and IoT systems. Announcements and press around 5G will pick up and some LTE+ trials may happen in 2017. The march to 5G has begun.

Maturation of IoT platforms

IoT platforms will continue to mature in 2017. Most platforms introduced last year had a single IoT application under their belt. The current value of adopting an IoT platform is a framework and access to IoT expertise. Having talked with and interviewed a number of IoT platform vendors last year, I heard two contrasting opinions forming:

1. It is possible to create a framework and work toward standardized interfaces that result in a common IoT platform with interoperable sensors, gateways, mobile apps, and cloud applications.
2. It is not possible to create a standardized framework, but it is possible to have a flexible architecture that can be used as a baseline, then customized to include the right sensors, connectivity, cloud services, and mobile apps.

This year will likely see one of these opinions emerge, as platforms start to be used for a larger variety of IoT applications they weren't specifically designed for. ECD

blewis@opensystemsmedia.com

BLUETOOTH AND MARKET TRENDS IN MESH NETWORKING

By Brandon Lewis, Technology Editor

Of all the design decisions associated with an Internet of Things (IoT) deployment, the most fundamental is the choice of network architecture. Particularly when dealing with wireless machine-to-machine (M2M) communications, the selection of an appropriate network topology has significant ramifications on the cost, power consumption, and bandwidth requirements of the devices and infrastructure that comprise these networks, and thus contributes considerably to the success or failure of connected products.

On the low data rate end of the wireless sensor network (WSN) spectrum, mesh topologies have been adopted as the architecture of choice for systems such as lighting and beacons in the retail, home, and building automation sectors. Mesh architectures are ideally suited for such applications because they support a relay approach to low-bandwidth wireless data transmission in which the range of low-cost, low-power devices can be "extended" as signals hop from node to node across a network. Mesh networks also have the added benefit of being ad hoc so that nodes can be added or removed with little to no network reconfiguration, and, when appropriate routing techniques are applied, can also be more resilient than alternatives as they reduce single points of failure (though many still employ central hubs or gateways). Indeed, many mesh networks are characterized as having "self-healing" properties, so the more nodes on a mesh, the better.

Traditionally, many of the systems and technologies used in the deployment of mesh networks have relied on the IEEE 802.15.4 standard, which defines Layers 1 and 2 of the Open Systems Interconnection (OSI) model (the physical and data link layers, respectively). While IEEE 802.15.4 lays the groundwork for the creation of cheap, often battery-powered wireless devices, the upper layers of the OSI stack (Layers 3 through 7) have been specified by protocols such as ZigBee, Z-Wave, more recently Thread, and a host of other proprietary technologies. For more than a decade, these 802.15.4-based specifications have dominated their target markets.

But a shakeup could be in store for this landscape – the Bluetooth Special Interest Group (SIG) will reportedly launch its Bluetooth Mesh specification in the first half of 2017.

Bluetooth Mesh: Potential "big" benefits for WSNs
"Bluetooth Mesh is big," says Lee Ratliff, Principal Analyst for Connectivity and IoT at IHS (www.ihs.com). "Mesh is important because it addresses some of the weaknesses of Bluetooth. You can [theoretically] extend the range by a km or more just by hopping from node to node. Obviously that goes a long way towards correcting what has always been a weakness of Bluetooth, which is range."

The Bluetooth Mesh specification will be an extension of Bluetooth Low Energy (BLE), which provides not only a comparable energy profile to competing wireless mesh solutions, but the per-unit cost synergies of a technology already shipping in high volumes and compatibility with existing BLE devices back to Bluetooth version 4.0 as well. Aside from increased range, cost synergies, low power consumption, and interoperability, though, perhaps the biggest advantage of Bluetooth Mesh is also a drawback of standards like ZigBee, Z-Wave, and Thread – the requirement of a central hub or gateway.

A hard truth for hubs
As noted previously, mesh networks allow signals to be repeated from one node to another across a network. However, all mesh architectures are not created equal. In the cases of the aforementioned Bluetooth competitors, for example, individual nodes primarily act as repeaters for source signals originating from a central hub. While this does not imply that nodes are incapable of communicating independently of the hub once a source signal has been received, the hub itself is a requirement of such topologies. From an engineering standpoint, this architectural requirement incurs additional cost, complexity, as well as a point of failure.

From the buyer's perspective, particularly in the home automation and other consumer markets, hubs also add cost and complexity that consumers may be unwilling or unable to deal with. As Ratliff puts it, "That's a device that most consumers don't want to buy in the first place, and once they've bought it, they're looking for a way to get rid of it."

Bluetooth Mesh, on the other hand, will not require a hub for systems and applications where one is not needed, which facilitates cost reductions and improves network reliability since data can always be routed through nodes within 30 meters of the transmitting device in the event that an adjacent node goes down. Furthermore, the power of Bluetooth in devices such as smartphones, smart speakers (i.e., Amazon Echo, Google Home), and other consumer devices "cannot be overstated," according to Ratliff, due to existing consumer confidence, ease of use, installed base, and economies of scale.

"They've got these big tech behemoths like Amazon and Google and Apple who are coming from a world where Bluetooth and Wi-Fi are the connectivity standards of choice," he says. "They're coming into the smart home where ZigBee and Z-Wave have always been the choice, and they're bringing Wi-Fi and Bluetooth with them.

"I think that's how the smart home is going to happen successfully," Ratliff continues. "Unfortunately, it's not going to be the early adopter route where these guys love fiddling with the home network and configuring devices and playing with their router and all that kind of stuff. That's not going to fly."

Beyond consumer home automation, the primary target for Bluetooth Mesh technology is the smart lighting industry. This has been a key segment for ZigBee devices for many years, Ratliff says, but despite the fact that "All the first tier lighting manufacturers use Zigbee … The whole Bluetooth Mesh initiative is all about lighting."

"It's to get that market," says Ratliff. "Anything after that is just gravy. Everyone is gunning for ZigBee's position in lighting."

The success or failure of Bluetooth Mesh in the lighting sector could have larger implications in the commercial and industrial spaces, however, as although Ratliff acknowledges that 802.15.4 standards are well established there and centralized network gateways are commonplace, "Mesh allows Bluetooth to get into a lot of applications that it couldn't before." For instance, Bluetooth Mesh will likely only accelerate the adoption of BLE technology in retail beacons, and as bring your own device (BYOD) and mobile usage expand outside of the consumer world, such trends could continue.

"The jury is still out on low-power, short-range wireless," Ratliff says. "We've got a lot of competitors between Bluetooth, Wi-Fi trying to compete in that space, ZigBee, Z-Wave, Thread, and a host of others. We don't know what's going to happen, but I wouldn't bet against Bluetooth. They've got a lot of advantages."

Market numbers and more M&A

Today, BLE technology, which will soon incorporate Bluetooth Mesh, deploys in the hundreds of millions of units per year. Though the current figures still represent "chicken feed for Bluetooth because they're used to smartphone numbers – billions of units," Ratliff expects volumes to ramp up quickly with BLE shipments crossing a billion units in the next few years.

Ratliff identifies the major players in the BLE market as top-tier vendors Nordic Semiconductor, Texas Instruments, Qualcomm (via the CSR acquisition of 2015), and Dialog Semiconductor, with companies like Cypress Semiconductor (via the acquisition of Broadcom's short-range wireless portfolio in 2016) and Microchip (via the Atmel acquisition of 2016) poised to enter that upper echelon. Aside from these, several boutique shops round out the ecosystem.

Despite this competitive field of no less than 22 different participants by Ratliff's count, his recent analysis also shows that "only three or four of them are getting 95 percent of the market share" of BLE. Combined with the technology's rapid growth and large organizations with remaining available capital, this fact signals to the IHS analyst that more consolidation is on the horizon.

"It's going to shake out, and five years from now there will be two or three companies that dominate [BLE] and there won't be anybody else," says Ratliff. "That whole area is attracting a lot of attention."

In the meantime, the continued competition should help breed quality in BLE technology, and, subsequently, Bluetooth Mesh. ECD

SMX® RTOS

Ideal for Your Project.

- Integrated platforms for 150+ boards
- TCP/IPv4/6, mDNS, SNMPv3, SNTP, SSH, TLS/SSL, HTTPS, SMTPS
- WiFi 802.11n, P2P, SoftAP, WPA2
- USB host and device
- Flash file systems
- Drivers, BSPs, Bootloader, GUI, IEEE754 Floating Point
- Progressive MPU security **smxrtos.com/mpu**
- Advanced RTOS kernel **smxrtos.com/special**
- Broad ARM & Cortex support **smxrtos.com/processors**
- Full source code. No royalty.
- Custom source eval and free trial

μd Micro Digital
YOUR RTOS PARTNER

www.smxrtos.com

FREE MAKERPRO STUFF: WHAT'S THE REAL COST?

By Jeremy S. Cook, Contributing Editor

Perhaps I should start by talking about my "free" dog, Evie. My wife and I adopted her at no cost when she was a few weeks old, and she's provided an enormous amount of love and enjoyment. On the other hand, she has cost us thousands of dollars in food, veterinarian fees, a ruined pair of glasses, and many other things damaged or ruined.

In other words, getting Evie was one of the best things we've ever done, but there was certainly a cost to it.

I've found this to be analogous to getting free stuff in the MakerPro world. If part of or all of your business is showing off what you made to the world, as you gain some semblance of influence, companies may offer you free items to use in videos or to otherwise promote. Sometimes it's great, sometimes it's not, and sometimes, even though it's a good product, you probably would have been better off just buying it rather than figuring out how to get it for free.

Your time is valuable

The most obvious cost to something like this is your time. Using round (and not necessarily representative of my income or anyone else's) numbers, let's say you can make $100 per hour producing videos, writing, or developing new products. You'd like to have a certain widget that costs $10, so you spend time researching the companies' press department, send off a few emails or tweets, and wait for them to respond. They say yes, which you think is great. Then, after having spent an hour of your time in total, and delaying whatever you were trying to do slightly, you get the product.

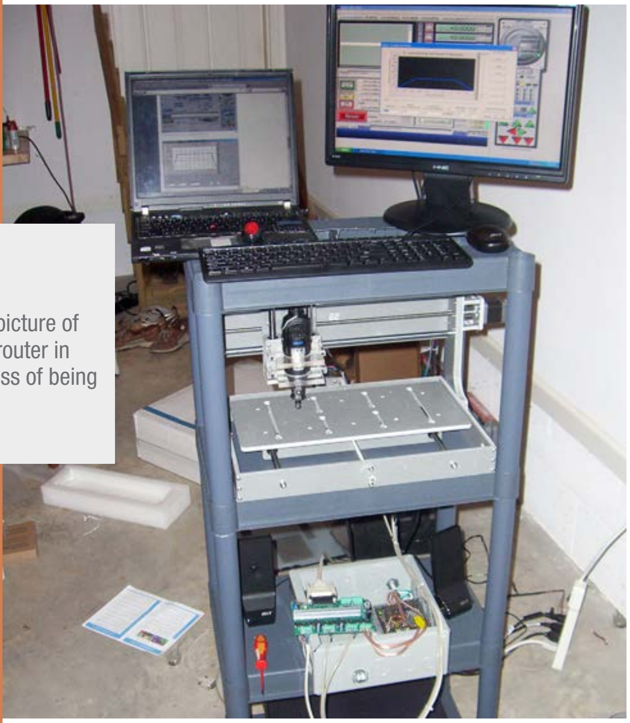


FIGURE 1
An early picture of my ZTW router in the process of being set up.

So you spend $100 of your time to get a product that would have cost you $10. Plus, you now feel obligated to use the item in your endeavor, whether it's good or not. You should also probably disclose your relationship with them so everything is on the up-and-up. That is a pretty high price to save $10.

Of course, if you're talking about something that costs $1000, the math becomes different. The value of your time, however, is something to consider before you start worrying about saving a few dollars.

The potential benefit

Like a free dog, though it might not make immediate financial sense, sometimes a free product can lead to a very valuable relationship that's beneficial for everyone. From my experience, as I was just starting to get into making stuff and blogging about it, a company called Zen Toolworks (ZTW; www.zencnc.com) gave me a very significant discount to start using their ZTW CNC router and blog about it (Figure 1). This computerized tool allowed me to make all kinds of interesting things, but what benefited both of us was that as I set this router up, I wrote many articles about the product – in fact, my blog's CNC archive is mostly about this router. This gave me a ton of content for my site, and lots of MakerPro writing experience. ZTW got the benefit of having their product featured many times, hopefully helping business along the way. It was a win-win scenario.

On the other hand, there have been times where I've regretted taking a free or discounted product, either because it didn't perform as I would have liked, or it just wasn't worth the effort to do what I'd promised in exchange (usually a video or review).

When evaluating which "freebies" you should take, really consider the cost beyond just dollars. Will it help you build valuable relationships? Is it a product you want to use and can honestly recommend? Will it restrict or expand how you can pursue future projects? It's a lot to think about. Just make sure that what you get out of a free product, however you personally measure it, is worth it for you. ECD

IVI: REVISITING RADIO TUNERS IN CAR HEAD UNITS

By Majeed Ahmad, Automotive Contributor

Advanced driver assistance system (ADAS) features are taking over the head unit in cars, and that inevitably requires more space and power. So how can car OEMs take things out of the head unit and move in more ADAS functions?

Automotive chipmaker Maxim Integrated has an idea. It entails replacing multiple radio tuner circuits inside the head unit with a single tuner located near the antenna, and then linking the tuner to the main host processor in the head unit to perform baseband tasks.

In a traditional architecture, multiple radio tuner front ends are connected to the head unit via analog cables, which are inherently susceptible to noise and heat dissipation while transporting signals from antenna to head unit. Additionally, each tuner requires its own baseband hardware.

What Maxim brings to the table is a remote tuner solution that is positioned near the antenna to free up space in the head unit (Figure 1). It digitizes tuner outputs for transmission to the head unit over a single coaxial cable that carries both radio signals and power. The result is lower thermal overhead, less noise, and reduced cable weight and cost.

digital hybrid radio receiver integrated circuit (IC) that unifies hardware for multiple radio broadcast standards like AM, FM, DMB, etc. It serializes the signals using Maxim's 14-bit GMSL SerDes pair and sends them to the head unit over a low-cost coaxial cable.

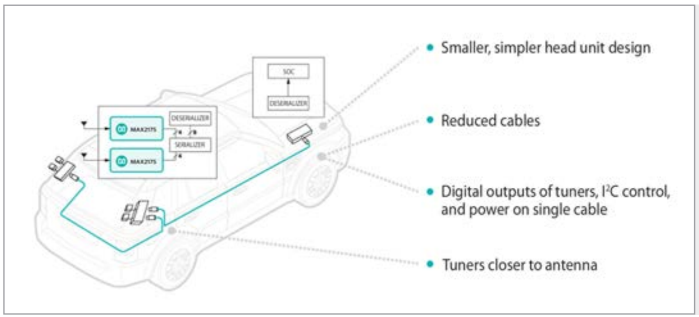

Smaller, simpler head unit design
Reduced cables
Digital outputs of tuners, I²C control, and power on single cable
Tuners closer to antenna
SOC
DESERIALIZER
MAX2175
SERIALIZER

FIGURE 1
A smaller and simpler head unit design saves space, minimizes heat dissipation, and reduces cable weight and cost.

Digital, as opposed to analog, signals are less prone to noise and heat dissipation. Maxim's RF to Bits tuner also allows baseband processing to be carried out in software within the main head unit system on chip (SoC), which eliminates the need for a separate baseband processor (Figure 2).

William Chu, managing director of automotive at Maxim, says that a single piece of hardware for all radio standards allows car manufacturers to simplify head unit design and lower power consumption. "Just one digital cable instead of four analog cables also lowers the cost and weight and improves radio performance," Chu says.

Moreover, to facilitate a software-defined radio (SDR) approach to baseband processing, Maxim, in collaboration with Munich-based digital audio specialist Fraunhofer IIS, is making core development kits (CDKs) available to automotive engineers. For its part, Fraunhofer IIS claims to have developed a complete SDR solution for Maxim's infotainment hardware. **ECD**

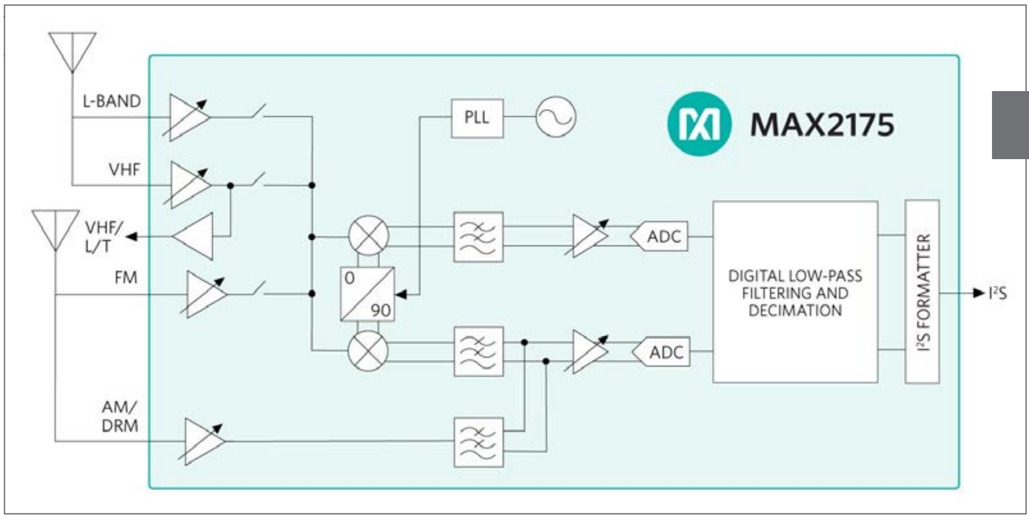
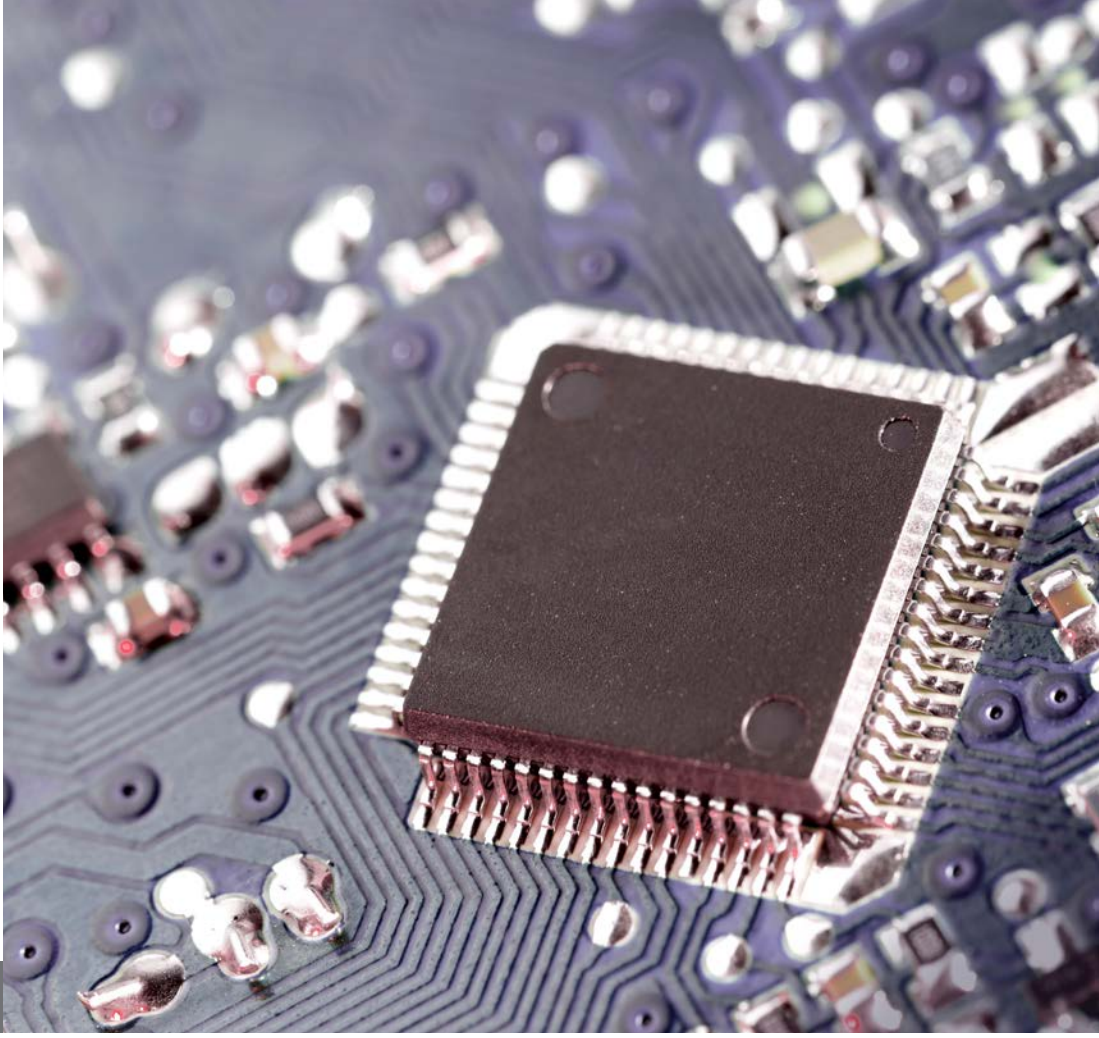


FIGURE 2

The hybrid radio receiver IC replaces multiple tuners with a deserializer that is connected to the SoC in the head unit.

2017 EMBEDDED PROCESSOR REPORT: AT THE EDGE OF MOORE'S LAW AND IoT

By Brandon Lewis, Technology Editor



With the benefits of Moore's Law waning and the Internet of Things (IoT) targeting an untold number of lower end devices, embedded processor vendors are now tailoring solutions to the specific needs of end customers and applications more than ever before. The result? An emphasis on power efficiency, security, development tools, and cost.

Forces at work in the electronics industry have reshaped the embedded processor landscape in recent years, among them, the slowing of Moore's Law and the realization that most IoT devices will emphasize price and power rather than feeds and speeds.

To be sure, embedded processors have traditionally been slow to adopt advanced process nodes, as longer lifecycles and the applications in question typically haven't required top-of-the-line performance. However, with the cost of developing chips that push the limits of semiconductor lithography on the rise, Jag Bolaria, Principal Analyst for Embedded and Servers at The Linley Group, expects that power efficiency and consumption will continue to displace performance as the key driver of embedded and IoT processing solutions.

"There is enough left in Moore's Law to take us out to 2022, but the rate of new process technology will slow down and the costs for development will increase," Bolaria says. "Specifically, the next nodes to come out are 10 nm in 2017, 7 nm around 2019, and 5 nm around 2022. The latter, however, is likely to need new fabrication technologies such as extreme ultraviolet (EUV) lithography, and at 5 nm, vendors may look at using exotic materials.

"The IoT will focus on low cost, low power, and power efficiency," Bolaria continues. "There is enough performance in existing products to satisfy most IoT applications. Most embedded applications do not need leading-edge performance, but they do want the best power efficiency and the lowest power consumption. Embedded products from leading vendor NXP do not offer leading performance, but are instead right sized – in terms of absolute power, level of integration, and power efficiency – for the target application."

Bolaria's observations serve as a proof point for the continued success of the 8-bit microcontroller (MCU) market, where despite years of speculation to the contrary, vendors such as Microchip have seen sustained success in their PIC and (recently acquired) AVR product lines. But as opposed to innovations in

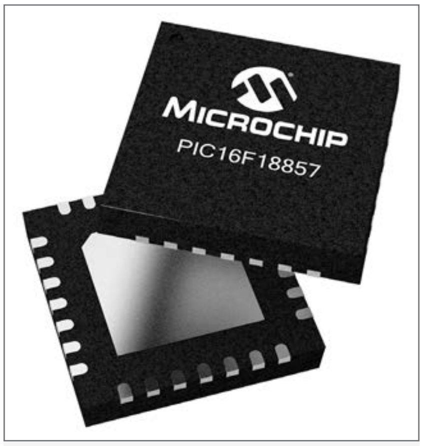


Figure 1 | The 8-bit PIC16F18857 MCU features Core-Independent Peripherals (CIPs), up to 56 KB Flash memory, and a 10-bit analog-to-digital converter (ADC) for a variety of general-purpose and low-power applications.

the CPU itself, the endurance of these 8-bit technologies can largely be attributed to intelligent hardware blocks like core-independent peripherals (CIPs) that "allow designers to implement the most time-sensitive parts of their application in fixed-function, low-power, and always-on hardware," says Greg Robinson, Senior Director of 8-bit MCU Marketing at Microchip Technology Inc. (Figure 1).

"CIPs have the ability to communicate directly with other peripherals to create configurable hardware blocks," says Robinson. "These "core-independent" blocks consume very little power and are much smaller than the RAM and Flash needed to implement the same function within the core. They also provide a quicker and more reliable response than a software-driven routine, which provides greater performance. CIPs require little to no code, eliminating the time and cost for validation and ultimately resulting in faster time to market.

"This is a great alternative to the practice of utilizing the CPU's sleep mode to lower power consumption when it's not needed and then wake it up with a hardware interrupt in the event that it's needed again, which brings in potential issues with variable startup time and interrupt latency," Robinson adds.

The effects of this trend can also be seen at the home of Moore's Law, Intel, where delivering features tailored to embedded systems is beginning to supersede Dhrystone Million Instructions Per Second (DMIPS) targets. While Jonathan Luse, General Manager of IoT Planning and Product Line Management at Intel asserts that the recently released Intel Atom Processor E3900 series based on the 14 nm Goldmont microarchitecture does provide a 70 percent improvement in CPU performance and a 2.9x increase in graphics capability over previous generations in the same 6-10 W power envelope, he also acknowledges that although the semiconductor giant is still "expected to deliver an increase in CPU performance per watt generation over generation or a graphics improvement generation over generation at the same power threshold," customers are continually "asking for more than that nowadays."

This is evident in perhaps the most significant enhancement of the Atom Processor E3900, Intel Time Coordinated Computing Technology (Intel TCC Technology), a switched fabric with global time awareness that connects internal digital signal processors (DSPs), image signal processors (ISPs), and sensor subsystems, and can be extended to time-sensitive peripheral devices via PCI Express (PCIe). "All about deterministic control loops or systems that require real-time computing down to the microsecond level," technologies like TCC are an example of the chipmaker's concerted effort to address application-specific needs in the key segments of automotive, industrial, and video (Figure 2).

"With Intel's approach from the edge to the cloud and the fog network in between, we have to think about it from a vertical solution first and work our way back to the technologies needed to solve the business problems that we've got," Luse says.

Security, tools trend upwards

Like power efficiency, security is another aspect of embedded processing that is moving to the forefront of design requirements. While processor-based security

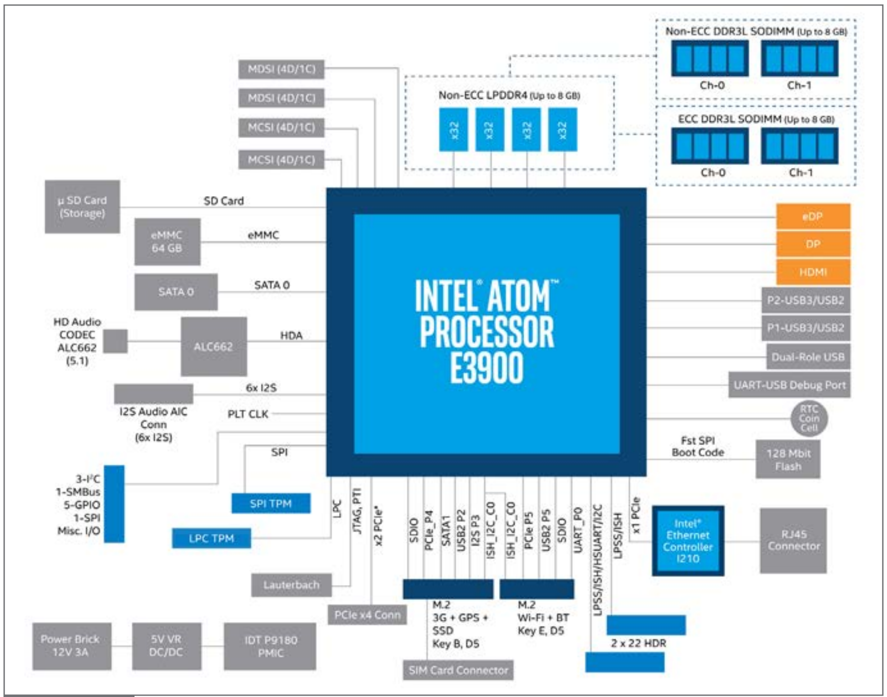


FIGURE 2 Intel Time Coordinated Computing Technology (Intel TCC Technology) in the Atom Processor E3900 series allows the processor and multiple interfaced devices to function in a time-coordinated fashion with microsecond latencies.

has historically been seen as prohibitive due to the cost and development effort associated with implementation, "ARM's introduction of TrustZone into Cortex-M-class processors over time is going to make the [security] hardware essentially not be a cost factor," says Steve Hoffenberg, Director of the IoT and Embedded Technology Practice at VDC Research.

"ARM TrustZone in particular has been around a long time, but its usage is growing considerably," says Hoffenberg. "The hardware has been there, but in the majority of devices that have used those processors, [developers] haven't actually used the security features of the hardware.

"With TrustZone now in Cortex-M, there will be little to no incremental hardware cost," Hoffenberg continues. "There will be more development time, which will be non-recurring engineering (NRE) on the software side to actually utilize the hardware, and there's is going to be an initial hurdle for a lot of engineers who start using that hardware in their lower end devices. But once they do, it should be relatively little, even incremental software development time, over the long run."

Development tools have also evolved to facilitate the utilization of security capabilities inherent in modern processors, with current versions of the ARM Keil Microcontroller Development Kit (MDK) and Cortex Microcontroller Software Interface Standard (CMSIS) both supporting TrustZone-enabled MCUs. Per Reinhard Keil, Senior Director of MCU Tools at ARM and Founder of Keil Elektronik GmbH, as "more and more devices include pre-programmed firmware in ROM" these tools will accelerate the development curve for "future Cortex-M23 and -M33 applications split into secure and non-secure parts."

"MDK and CMSIS version 5 already include full support for TrustZone-enabled Cortex-M devices," Keil says. "Several program examples and a full-featured real-time operating system (RTOS) help you get started quickly, and we will deploy more template applications that exemplify the setup of secure systems. Consistent APIs are fundamental for such software environments, and ARM enables this with CMSIS and the application binary interface (ABI) procedure call standard for compilers."

Keil also notes that Fixed Virtual Platforms (FVPs) available in the MDK-Professional Edition can be leveraged to simulate complete Cortex processor systems, considering memory and peripherals in addition to the core itself. An FVP for the Cortex-M33 provides simulation speeds "similar to a Cortex-M running at 100 MHz," Keil says, permitting engineers to "develop and validate software for next-generation MCUs today."

Elsewhere in the tools market, MCU suppliers themselves have started to update their offerings in anticipation of application development based on ARMv8-M architectures equipped with TrustZone technology, and here the Renesas Synergy Platform is an indicator. According to Semir Haddad, Director of Marketing for MCU and MPU Product Solutions at Renesas Electronics America, Synergy will support "the Cortex-M22 and Cortex-M33 in a future generation with integrated software and tools that abstract many low-level details from the developer for ease of use and fast time to market."

"One of the focus areas for our next-generation Synergy Platform will be to further reduce complexity for developers creating secure connected devices with advanced security technologies," says Haddad. "The use of ARM TrustZone for Cortex-M as a hardware virtualization capability will be integrated with the other elements of the Synergy Platform, including the security scheme, offering developers improved ease of use and leading to a higher adoption rate of TrustZone features in ARMv8-M-based systems."

M&A: Causes and consequences

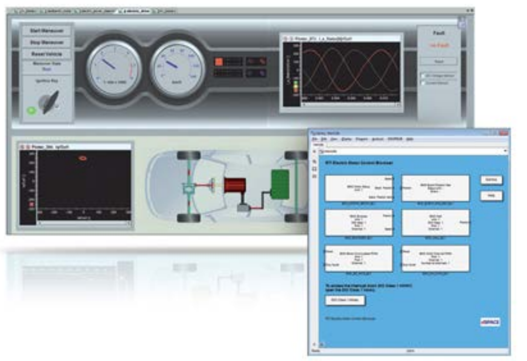
The most undeniable trend in the processor industry over the last 18 months has been the appetite for acquisition, which Bolaria attributes to the fact that "process technology is no longer bringing the same benefits in lower silicon and transistor costs" on the one hand, and "cheap money" on the other.

"If a company can get loans at 3 percent and they buy a company that has 9 percent margins, the acquiring company is making 6 percent for doing nothing," explains Bolaria. "Then they can further increase the margin by cutting costs and thus make more money. Chip vendors need to make enough margin on the current generation of products to fund the next generation, and with increasing costs, they need to figure out how to increase the return."

In addition to the factors mentioned, consolidation "in and of itself is an indication that the margins and revenue opportunity in a lot of traditional embedded processor markets that these suppliers have been chasing are starting to wane with more competition, which is stemming from across the world," says Dan Mandell, Senior Analyst in the IoT and Embedded Technology Practice at VDC Research. "The aggressive investing of the Chinese Government in trying to acquire as much semiconductor manufacturing capacity as they could over the past couple of years is just one con. The other is that it's going to take a couple of years for the integration of these mergers to really materialize in terms of the management, the personnel, the products, and the market strategies.

"There are a lot of synergies to be had as far as the different mergers taking place between the different semiconductor suppliers," Mandell says. "The semis will maintain their merger and acquisition appetite through the next year and continue to invest heavily in both hardware and software solutions so that they can best position themselves from a competitive standpoint for some of these burgeoning opportunities in advanced driver assistance systems (ADAS) or 5G – areas where the revenues aren't currently but that demand a lot of up-front R&D and product development and coordination among various partners in order to be able to reap the rewards in five years' time." ECD

dSPACE MicroLabBox – Compact power in the lab



Are you looking for a powerful development system for all kinds of laboratory tasks that takes up little space and comes at an attractive price? dSPACE MicroLabBox provides a Simulink®-programmable real-time processor with high computing power, combined with an FPGA and over 100 I/O interfaces. All this with a desk space no larger than a conventional laptop computer. MicroLabBox is the ideal solution for conveniently creating, optimizing and testing controllers in drive technology, robotics, medical engineering, and many other areas. MicroLabBox - your new all-in-one system for research and development.

Embedded Success dSPACE

I3C: AN UPGRADED INTERFACE FOR A WORLD OF SENSORS

Interview with Ken Foust, MIPI Alliance

KEN FOUST
Chair of the MIPI Alliance Sensor Working Group

Shortly after the 10 year anniversary of the iPhone, Ken Foust, Chair of the MIPI Alliance Sensor Working Group reflects on how that platform catapulted the use of sensors in smartphones to new heights, but also the challenges that increased sensor integration has wrought for legacy interfaces like SPI, UART, and I2C in terms of cost, power, and performance.

Enter the Improved Inter-Integrated Circuit (I3C), a next-generation chip-to-chip interconnect capable of supporting not only mobile devices, but Internet of Things (IoT), wearables, and automotive sensor subsystems as well.

What can you tell us about the history of the I3C interface specification?

FOUST: I3C by now goes back more than a few years. If you were to rewind three or four years, there were several companies that were dealing with some of the same pain points when it came to adding sensors to mobile handsets. One of those issues really stemmed from the proliferation of sensors.

At that time I was at a startup MEMS company trying to convince people that things like accelerometers, gyroscopes, and magnetometers were going to unlock a whole lot of potential for consumer devices. The iPhone, with its wide touchscreen that needed auto rotation and various input types for gaming, which sensors allowed, opened up opportunities but also created a lot of problems. Quickly we went from very few sensors in phones to today where a phone could have up to ten sensors. You could have accelerometers, magnetometers, gyroscopes, ambient light sensors, proximity, pressure, temperature, humidity, infrared, RGB sensors – there are so many sensors that connecting them up to an applications processor or the primary compute was becoming expensive. No longer could the I2C bus – at least a single iteration of it – handle all of the sensors. You had to have multiple instances of the I2C bus, which created more GPIO demand. All of the sensors were getting smarter so that you could configure them to sit in a low power state, look at their data, and interrupt by transitioning a dedicated pin from low to high, for example, when something occurred. But if you had 10 sensors, that's 10 GPIO pins.

Since the sensor subsystem in these handsets was getting expensive, several of us got together and put down all of our gripes with respect to I2C, SPI, and UART, whether we were sensor vendors or people making reference designs based on various processors. We put all of these gripes and our wishes out on the table and then spent a good year discussing them, turning it into a problem statement with a proposed solution.

With I3C, we now have a lasting, low-speed interface that is derived from requirements for sensors, but will find home anywhere that I2C, SPI, and in many instances, UART, have been found.

How are IoT and embedded developers going to benefit from I3C?

FOUST: What people will like about I3C are things like the dynamic address allocation procedures, where we standardized a lightweight, simple way to discover, enumerate, and assign addresses to devices.

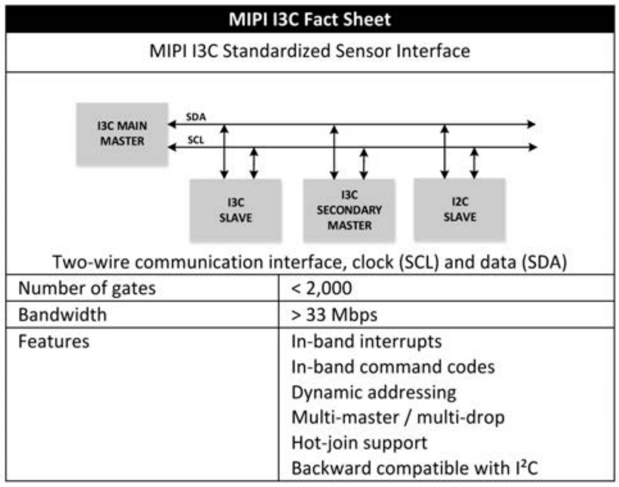
In something like I2C, where you have fixed addresses, there's a chance that there could be conflicts. You don't have that with I3C. Where SPI is considered fast for a low-speed interface, each SPI device requires a dedicated chip select (CS) line to address it. You don't have to do that with I3C. We implemented an in-band interrupt capability that lets you take all of those out-of-band dedicated GPIOs that allow sensors to interrupt the host and lets each of them simply do it over the two wires of I3C, reducing cost (Figure 1).

Then we looked at things like power. I2C is open drain, so every time you make a 0 you're fighting a pull-up resistor and it's rather slow, with 400 kHz or 1 MHz being the popular speeds. This means that applications processors have to stay up for a long time as they unload FIFO buffers full of data. With I3C we ramped up the clock and added support for push/pull capability so we're not fighting the pull-up resistors, which allows applications processors to get data from the sensors faster.

In addition to faster single data rate operation, we added the concept of high data rate modes – double data rate and ternary symbol transcoding – that can optionally be supported. With these high data rate modes, for the same amount of energy that you spend clocking at 12.5 MHz, you can double or nearly triple the effective throughput, so your millijoules per bit energy starts to drop as you adopt those modes (Figure 2).

The interface is also fast enough that it has optional timing control built in so there are ways for the host to synchronize sensor data collection in time, which can be important for sensors or applications that are time of flight-based, for example. You wouldn't do that natively with legacy interfaces like I2C or SPI.

These are all features that can lend themselves to not only sensors in handsets, but all of these other areas where low-speed interfaces are used, both for chip-to-chip connectivity as well as other applications or industries like wearables, IoT, automotive, etc.


MIPI I3C Fact Sheet

MIPI I3C Standardized Sensor Interface

I3C MAIN MASTER
SDA
SCL
I3C SLAVE
I3C SECONDARY MASTER
I2C SLAVE

Two-wire communication interface, clock (SCL) and data (SDA)

Number of gates	< 2,000
Bandwidth	> 33 Mbps
Features	In-band interrupts In-band command codes Dynamic addressing Multi-master / multi-drop Hot-join support Backward compatible with I²C

FIGURE 1 A snapshot of the MIPI I3C interface's features.

Is any additional complexity associated with programming an I3C interface?

FOUST: Your basic I3C looks a lot like I2C, except with defined areas where you drive the clock with push/pull and use a little bit of logic to do in-band interrupts.

From a slave side we really wanted to limit logic complexity and gate count, so we set a target of less than 2,000 gates for a slave-side implementation because, in our opinion, we want this interface to be appropriate for extremely inexpensive sensors or ICs – single-cent-type things. We really tried to push complexity to the host side if there was going to be complexity.

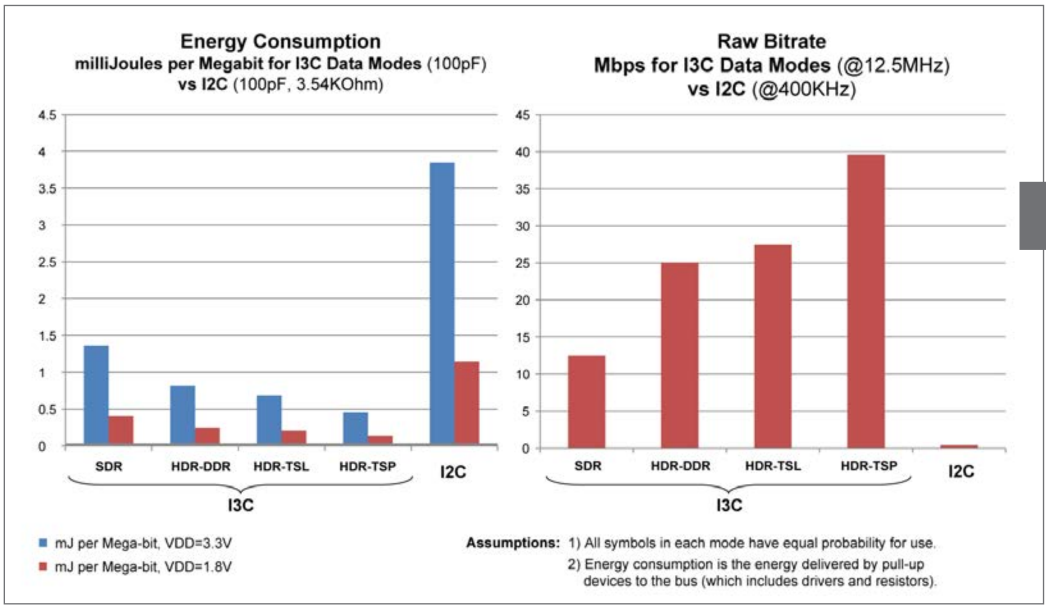


FIGURE 2

Shown here is a comparison of the energy consumption and raw bitrate of I2C versus MIPI's I3C interface.

It does remain backwards compatibility with I2C, correct?

FOUST: Something from the early days of our charter was to maintain backwards compatibility as best as we can. What we found in our analysis was that not all I2C devices are created the same, and not all of them follow exactly the published I2C specification, but where you do have those compatible devices, yes. That mostly revolves around whether they correctly implemented the 50 nanosecond spike filter. If they did not, there are ways for the bus to continue to function, there's just a degradation in performance due to the existence of that device. For instance, you have to talk to the device at the speed of the device, and the bus has to run at that speed.

However, we found that's not the majority of I2C devices, and that the majority were implemented correctly. So coexistence? Yes. Compatibility? Yes. We anticipate host controllers out there that are I2C/I3C host controllers and slave devices that are I2C/I3C devices. Legacy I2C devices can coexist, and in the spec we point out things in legacy devices that may be impacting performance and how to adjust how the bus operates in the event that that happens.

Do you see I3C completely replacing the lower-speed interconnects in certain segments?

FOUST: If you look at MIPI's broad influence in mobile, for instance, that's where it has the best chance of happening. However, in certain segments, there are some things about I2C and SPI and UART that are very favorable. Using them in mobile was a hindrance, but in some applications they can be advantageous. For instance, I2C is very slow and requires very strong drivers, so its power hungry. But in doing that it can drive very long lines – longer lines than we would ever consider a mobile interface driving. Do we necessarily view I3C as an interface that would be routed through, around, and all over a tube television like I2C was developed for? Probably not. I think I2C is still going to be the interface for that, but that's not what I3C was developed for.

What's next for MIPI and I3C?

FOUST: When we came up with the charter for our working group, we knew that the world was changing a bit. We'd seen things like IoT and drones and AR/VR and autonomous vehicles coming, and anticipated that those industries would continue to drive future capabilities into I3C. We'll continue to work on I3C and keep it relevant in those segments.

All the major IP providers in the world have this IP available for people to start synthesizing into their designs, so I have pretty high confidence that by the end of this year we will see devices that support I3C out there. ECD

OpenSystems Media E-cast

Embedded University: Device Management 101

Looking at the "Things" in the Internet of Things – nodes and devices – we investigate how to effectively commission, monitor, and detect problems throughout their lifecycle. Methods for adding device management capabilities in the design stage are considered, as well as ways to implement these capabilities in legacy devices. This three-part course will take developers end-to-end in the deployment of manageable IoT devices.

March 21-23, 2017

Sponsored by Wind River

Embedded COMPUTING DESIGN

ecast.opensystemsmedia.com/720

DIVE INTO THE INDUSTRIAL IoT WITH CONFIDENCE

By Mychal McCabe, Vice President, Corporate Marketing, Wind River

We're still in the early days of the Industrial Internet of Things (IIoT) or Industry 4.0, and there's a fair amount of trust that's required for OEMs looking to adopt this technology. If you're making changes to your infrastructure, or potentially even swapping out or building an entirely new infrastructure, you're making a huge commitment in terms of CapEx.

There are numerous sectors that can be positively impacted by implementing a software-defined infrastructure approach. These segments cover the gamut from greenfield to brownfield. Regardless of where they reside, they can all gain and grow significantly by making proper use of the IIoT. Wind River customers comprise many of these critical infrastructure segments, including communications, defense, energy, healthcare, manufacturing, and transportation.

There are generally two camps: there are the providers of high-end, high-production manufacturing equipment that need to migrate downward to bring in the communications technology and cloud-based analytics. And there are the traditional embedded providers that are trying to migrate upwards into larger-scale platforms. It's that gray area in the middle that scares people. They often look at it as an unknown, and they look at themselves as pioneers.

It's inevitable. Change is coming.

Change is needed, and change is coming. The business drivers that are motivating customers to change include: the obsolescence cycle; capital cost reduction pressure; current systems limit or lag innovation; poor component interoperability; high integration and maintenance costs; and insufficient system security models. The technology enablers that will make this a reality include: the IoT; virtualization; cloud computing; open platforms; data analytics; and proof points from adjacent industries.

When our customers and potential customers look at the amount of intelligence and processing that's needed, both at the edge and in the cloud, it becomes compelling for them to migrate to an enterprise-style virtualization approach that can reap the benefits and knowledge garnered by the embedded domain, which has mastered low-overhead communication, real-time capability, small memory footprint, and high levels of security.

Some customers, like those in the oil production industry, want to be sure that they can leave their systems intact for a minimum of five to six years and be guaranteed continuous operation. One way to do that is by designing in triple redundancy at the edge, where you have an online system, a backup system, and a backup to the backup. While such an architecture ensures the desired operation, it also comes at significant CapEx cost. Adding some level of "smarts," i.e., IIoT, can reduce the CapEx cost while providing even better service than before.

That's where the trust comes in. While OEMs may be required to make a substantial reduction in proprietary hardware at the edge, they need to know that all of the reliability, safety, and security capabilities that they had before and need going forward will be intact, even though they're migrating to more generic hardware.

That's where Wind River's technology is differentiated. With our VxWorks real-time operating system, we have a strong history of delivering solutions for deployment in critical infrastructure applications where failure is not an option, and we continue to evolve and expand our embedded portfolio to include innovative and highly reliable IoT solutions. Wind River Titanium Control exemplifies Wind River's extensive technology heritage and strategic leadership by delivering a solution to power the Industrial IoT. It is a commercially deployable virtualization software platform that reliably delivers the performance needed for critical infrastructure applications and control services at any scale, providing both the ultra efficiency and ultra reliability to keep costs down and systems running. And with Wind River Helix Device Cloud, a cloud-based IoT device management platform, companies can build device management capabilities right into their IoT infrastructure to meet the challenges associated with device lifecycle management and reduce the complexities of building and operating large-scale device deployments.

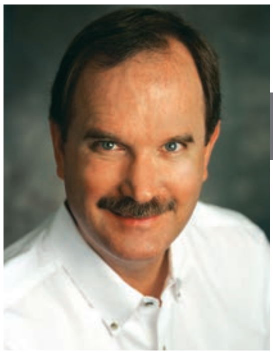
When we first started talking about IoT, service providers were daunted thinking about the massive rip and replace that was required. Now, we know for certain that it *can* come albeit with a slow roll, and we've seen customers — some Fortune 100 manufacturing customers — execute in that gradual fashion. In reality, it occurs at a pace that the customer is comfortable with.

Aging automation systems are a major concern for critical infrastructure companies. They are seeking to lower operational costs, respond quickly to changing market demands, and improve data security and worker safety. While their systems may be reliable and stable, they are typically custom-built, proprietary, and closed, so adding new features is a lengthy and extremely costly process, which ultimately stifles innovation. Moreover, it is difficult if not impossible to extract meaningful data from these systems that might help drive greater efficiency. In mature companies, legacy system expertise is dwindling as workers retire, to be replaced by candidates accustomed to more modern technologies.

At the end of the day, it's all about helping our customers achieve success whether in the form of business transformation, optimized productivity, increased efficiencies, and/or cost savings. And the Industrial IoT is providing that opportunity for success.

Wind River • www.windriver.com/markets/industrial

PATCHING UP LINUX FOR REAL-TIME APPLICATIONS: ORIGINS AND IMPACTS ON IoT


JIM READY

Interview with Jim Ready, Independent Consultant

A pioneer of embedded operating systems (OSs), Jim Ready is not only credited with the creation of one of the first commercially available real-time operating systems (RTOSs), the Virtual Real-Time Executive, under his guidance MontaVista helped pave the way for the use of Linux in embedded devices in the early 2000s. Now an independent consultant, Ready reflects on how early work in embedded Linux that prompted modern mobile OSs like Android also branched into more deeply embedded applications through the advent of capabilities such as the Realtime Preemption (RT-Preempt) patch, and how that evolution could ultimately impact the software hierarchy in the Internet of Things (IoT).

How do you describe "real time" today in the OS sense?

READY: There are two ways to think about this. One is, in academic circles you can find what people have defined as real-time computing, and I don't think that's changed any: The classic phrase is, "The right results, on time," and that's always whatever your requirements are. It has nothing to do with time necessarily; it has to do with meeting the requirements of the time-related aspects of your application, which could be very soft or very stringent.

The second is some of the things that have changed how people look at real time. If you look at your smartphone, it's doing all sorts of stuff that is obviously real time, from the radio to picture processing. There is a range of ways that's solved: Number one is you just run what you need in hardware and achieve whatever requirements you have; or it can be very dedicated software on that hardware, so classic hard-real-time software running in the bowels of the hardware; or, even with the software that moves up into the application area and into the main processor, because the underlying power of an ARMv8-A in an iPhone allows for incredible processing in a very, very short period of time.

The system design aspect of this has settled down. In other words, people know how to do this now and there are no giant arguments about what Linux can or can't do or what dedicated software can or can't do or what an RTOS can or can't do. It's settled into, "Choose the right solution for the right aspect of the problem," and people are reasonably sophisticated about that now. 15 years ago we might have had some of these arguments where people were going berserk about Linux not being able to do real time, but that's all gone. Whatever deficiencies there were have been fixed, solved, and are good enough; people have 15 more years of experience designing these systems; and the hardware has gotten 15 years better. It's a non-issue that's been solved effectively, to my mind.

What was done to make the Linux kernel real time?

READY: All of the classic things you did when you built an embedded system, Linux had not been geared that way because Linus [Torvalds] was working on his desktop. Linux at the very beginning had something called Big Kernel Lock (BKL), which meant that relatively large portions of the kernel were made non-preemptive just to make things simpler, and once you got into those you had to execute to completion.

The whole game of adding real-time capabilities to Linux was to reduce the size of that BKL, which then would increase responsiveness to be able to do other things. That was what the RT-Preempt patch work focused on, and the point was basically to get the kernel out of the way. Once you did that you had the same instructions everybody else had in terms of handling interrupts, hardware latencies, and everything else, so it removed an artificial restriction and still protected the kernel, just in a much more fine-grained way. So there could be mountains of code that execute when they execute, but not at the expense of whatever needs to be done in real time. This was done through a kind of partition.

At the Linux Foundation there is a working group that continues to maintain and mainstream the RT-Preempt patch. It's lead by Thomas Gleixner, and he was one of the original guys who worked on Linux to make it more real time. So there's a long, long continuum of really good embedded guys who have put their efforts into continuing to improve the capabilities of Linux.

Given work on the RT-Preempt patch and other deterministic enhancements, how does real time Linux now compare to an RTOS?

READY: Whether you're running an RTOS or you're running Linux, the processor doesn't go any faster – the guys who write the RTOS effectively use the same instructions as the guys who write Linux. It's really how you organize everything, and for all intents and purposes, 10 or more years ago we showed that the latencies you could achieve with Linux were on the same order of what you could achieve on the same processor with an RTOS. It wasn't like 100 to one or even 10 to one – by careful design, you could utilize the underlying hardware and get the results you needed.

At MontaVista we had something called the Bare Metal Engine that produced an environment within Linux that allowed access to the bare instruction sets that the hardware provided so you were at the same level of latencies that you would be at with an RTOS. Say you had a multicore network processor like a Cavium Octeon with dedicated hardware engines in it doing packet processing. If you wanted to have an environment that was Linux but you didn't want to have any of the latencies of Linux get in the way, the Bare Metal Engine allowed partitioning and the use of the internal capabilities of Linux such as processor affinity to tailor an environment so that the application could have full access and control over the underlying packet processing hardware, but still in a Linux context. It was a real Linux program, just with a very refined, very high-performance, very dedicated environment. If you wanted to put an RTOS down, you could, but you didn't necessarily have to. It could be a pure application, for example, that was driving these hardware accelerators.

We were doing this in 2009, 2010, and 2011, so a relatively long time ago. We were taking capabilities that found their way into the Linux kernel like the RT-Preempt patch and exploiting them, and doing very low latency, very high-performance environments that were as capable as if there was no OS running on the chip. It's not like you were chopping out big chunks of stuff at all. There was no removal of code whatsoever. It was just functions and capabilities of Linux itself, and if you use them in a particular way and configure them in a particular way, you can achieve this. It wasn't a trick where there was one line of Linux left and you snuck in an RTOS or you had an RTOS and you ran Linux as all of the tasks. It's Linux. We weren't gutting it. We weren't ripping it apart. We were using features that were there and adding some in the classic open source way, but it was absolutely in the mainstream of Linux.

If you put together a full RTOS you need 1 MB or 2 MB, and it's not that hard to get Linux configured to run in that range. Now, it still wouldn't make any sense to use Linux where you need a five-line executive or a little tiny kernel. But the whole point of early work on the RT-Preempt patch was to make Linux capable enough to be able to use it where it makes sense to use it. What it took out of the equation was "I'd really like to use Linux here but it's not quite real time enough" so that it could be symmetric with your requirements.

What implications does embedded Linux have on the future of OSs given ongoing convergence in the IoT?

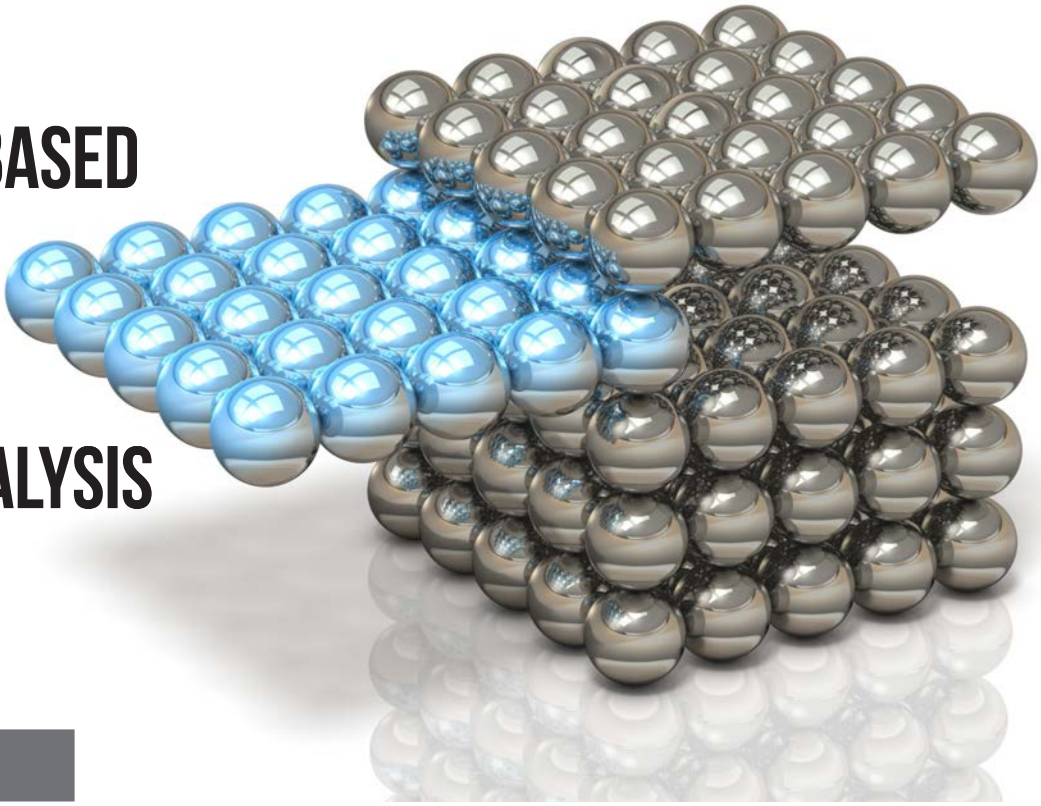
READY: If you meet your requirements, mainstream is the place to be. Fundamentally that's why I got interested in Linux for embedded systems in the first place many years ago, because it was just so expensive to be on the outside. If you had an RTOS you had to go find a TCP stack, etc. Once I saw Linux come out, I said, "Hey, this solves a huge number of problems if we can make it work for embedded systems." Of course we did, and the rest is history. The more mainstream you are, the less work you've got to do.

The way that's happening in the IoT is through the consortiums that have been formed. And it's not even so much standards based as it is implementations, so these consortiums are cutting right to the chase and providing open source versions of middleware and protocol stacks through reference implementations that give people a place to start. The vast majority of the software and intelligence will be shifted off the devices into standard computing environments designed for IoT, so you'll want to be in the normal world as much as possible. People are taking a system approach to this, secondly there are all of these consortia, and it's very hard for a proprietary technology to carve out any kind of control; I think that's impossible now.

In the actual physical systems there is a hierarchy of underlying hardware capability and then there will follow a hierarchy of software. There is a role for RTOS in there, and at some point where you leave off with RTOS, the next stop is Linux, which can then carry you quite a ways since that's what's also on the cloud side of things. It will span both sides, as it will obviously have a home even in the hardware hierarchy that's supplying the cloud.

If you look at it economically, the pressure is probably going to be on the smallest number of people to adapt so the vast majority of folks can continue doing what they're used to doing, just for efficiency reasons. You're not going to retrain a gazillion IT guys. ECD

A SOURCE-ANNOTATION-BASED FRAMEWORK FOR STRUCTURAL COVERAGE ANALYSIS TOOL TESTING



By Olivier Hainque

Automated testing of software tools always requires some way of comparing what the tool does against what we expect it to do. Testing compilers, for example, usually entails verifying the behavior of compiled programs, checking compilation error messages, or analyzing the generated machine code. For static or dynamic analysis tools, this typically involves checking the tool outputs for well-defined sets of inputs.

The following presents a framework developed for in-house testing of a structural coverage analysis tool, where expected coverage results are expressed as annotations in source comments. This framework was used to qualify the tool for use in several safety-critical software projects with stringent certification constraints in the avionics domain.

We first summarize the coverage criteria and tool output formats that need to be supported, then introduce the main principles of our scheme for describing the tool's expected results and explain the advantages over performing comparisons against baseline outputs.

Coverage criteria and output report formats

The code coverage analysis tool we need to test supports the three coverage criteria defined by the DO-178C certification standard for airborne software[1]: Statement Coverage, Decision Coverage, and Modified Condition/Decision Coverage commonly referred to as MC/DC[2]. It produces two kinds of output report formats:

- *Annotated sources* generated from the sources to analyze, with lines prefixed by a line number and a coverage result indication;
- and a *text report* listing violations against the coverage criteria for which an assessment was required.

The excerpt in Figure 1 shows an example piece of an annotated source result for a Decision Coverage assessment performed by the tool on an Ada function that was called only once, with a value of X greater than the Max parameter.

The information at the start of each line is the tool's output showing the coverage results. The "-- #" texts are special comments within the original source (comments start with "--" in Ada) recognized by the framework as introducing *tags* that allow users to denote the lines in expectations of the coverage results, which will be described later on.

In DO-178C parlance, Boolean expressions such as the one controlling the if-statement are called *decisions*, and achieving Decision Coverage requires tests evaluating each decision True least once and False at least once, in addition to executing each source statement.

In the example at hand, the decision controlling the if-statement was only evaluated once to the Boolean value False for X > Max. The decision is thus only partially covered and the return statement on line 4 is never executed. This is conveyed by the "!" and "-" characters next to the line numbers on lines 3 and 4, together with the "+" annotation on line 6 indicating proper coverage of the return statement there.

The alternate kind of output format for this assessment, a text report listing coverage violations with respect to the criteria, would include messages such as those in Figure 2, where the first part is a *filename:line-number:column-number* source location consistent with the annotated source result.

Stating basic expected coverage results

What the actual coverage results looked like for the tool were illustrated in the previous section. How test-writing personnel specify the expected results for a given test scenario will be described next.

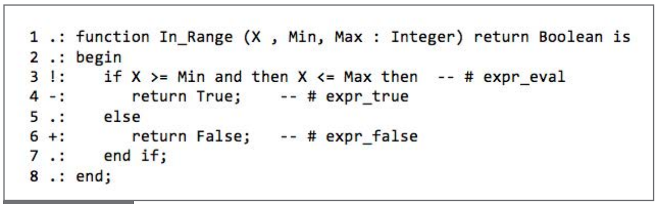
The main goal of the tool is to let testers state expectations efficiently using the source code being tested while abstracting away the report format details. It also encourages active thinking about what the expected results for a test should be.

The tool defines a test as a combination of three categories of source files:

- *Functional sources*, which is the code whose coverage the tester wants to assess and check if the results conform to the coverage tool requirements;
- *driver sources*, which call into the functional code in a specific manner with precise coverage objectives;
- and *helper sources*, which are simply needed for completeness and do not require coverage analysis.

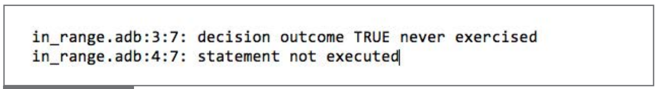
Expected coverage results are then stated as specially formatted comments in the driver sources, referring to lines in the functional sources also tagged by specially formatted comments.

The In_Range example function presented previously shows instances of special comments introducing tags. The "expr_eval" tag, for example, allows denoting the line where the decision expression gets evaluated. A given tag may appear on several lines.



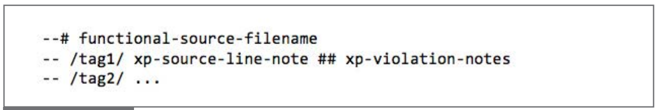
```
1 .: function In_Range (X , Min, Max : Integer) return Boolean is
2 .: begin
3 !:    if X >= Min and then X <= Max then  -- # expr_eval
4 -:       return True;    -- # expr_true
5 .:    else
6 +:       return False;   -- # expr_false
7 .:    end if;
8 .: end;
```

FIGURE 1 Shown here is an example annotated source result for a Decision Coverage assessment performed by the structural coverage analysis tool on an Ada function.



```
in_range.adb:3:7: decision outcome TRUE never exercised
in_range.adb:4:7: statement not executed
```

FIGURE 2 Messages generated in a text report of the Decision Coverage assessment performed by the structural coverage analysis tool on an Ada function are shown here.



```
--# functional-source-filename
-- /tag1/ xp-source-line-note ## xp-violation-notes
-- /tag2/ ...
```

FIGURE 3 The special comments describing expected coverage results in driver sources for the structural coverage analysis assessment performed on the example Ada function are shown here.

The special comments describing expected coverage results in driver sources are sequences of comments as shown in Figure 3, where "xp" stands for "expected." The first line marks the start of expected results for the functional source named *functional-source-filename*. The */tag1/* line states expectations for all the source lines tagged with *tag1* in this source. *xp-source-line-note* conveys the coverage indication character expected in the annotated source output format for these lines (same annotation for all the lines), and *xp-violation-notes* conveys the set of violation messages expected in the text report format for these lines (same set for all the lines).

A driver source may contain several */tag/* lines for a given functional source and expectations for several functional sources.

On */tag/* lines, short identifiers may be used to denote the various possible coverage indications in a compact fashion. "l+", "l-" or "l!", for example, are available for *xp-source-line-note* to denote an expected "+", "-", or "!" coverage indication on the annotated source lines, respectively. For *xp-violation-notes* we have, for example, among all the possibilities, "s-" to denote an expected "statement never executed" violation, or "dF-" for a "decision outcome False never exercised" violation.

Figure 4 (page 22) shows a sketch of a driver source to illustrate a Decision Coverage test over the source file providing the In_Range function named in_range.adb. This driver implements the execution scenario previously used to illustrate the output report formats, calling In_Range function once with X > Max.

The /expr_eval/ line states expected coverage results for the set of lines tagged "expr_eval" within in_range.adb. In the example, this is the single line where the decision gets evaluated (only once to False by this specific driver), so a partial coverage indication on the annotated line (l!) and a "decision outcome True not exercised" violation diagnostic in the text report (dT-) should be expected.

The /expr_true/ and /expr_false/ lines state expected coverage results for the source lines tagged "expr_true" and "expr_false", with those tags chosen to denote lines with statements executed when the decision evaluates to True or False. The "0" used as *xp-violations-notes* for "expr_false" denotes an empty set, meaning that no violation for these lines in the text report is expected. This is consistent with the expectation of a '+' in the annotated source format (l+ as the *xp-source-line-note*), corresponding to full coverage of all the items on the lines (in the example, a single return statement on a single line) as enforced by the execution scenario.

These expectations correspond exactly to the actual results shown in the initial example; this test would "pass" using the structural coverage analysis tool testing framework.

Advanced expectations
The previous section showed examples of basic formulations of expected tool behavior, unconditional and referring to entire lines. Allowing a complete test suite for the set of criteria that was targeted, however, required the development of a number of advanced capabilities.

The most immediate need was for precise source locations in *xp-violation-notes* to allow referencing a particular section of a line when distinct diagnostics could legitimately be expected for different items on this line.

When assessing MC/DC, for example, the tool diagnostics refer to specific operands within a Boolean expression (*conditions* within *decisions* in DO-178C terms), and most coding standards allow Boolean expressions with multiple operands on the same line. Testers must be able to specify the particular condition on a line for which a coverage diagnostic is expected. Similar needs occur with other criteria as well, for example, with Statement Coverage when multiple statements share the same source line or with Decision Coverage when nested decisions are involved in an expression.

This is supported through extensions with the form :"*line-excerpt*" at the end of violation designators, as in the sample expectation line for a hypothetical test of an MC/DC assessment over the example In_Range function in Figure 5. c!: "X >= Min" states that we expect an incomplete condition coverage diagnostic (c!) designating the "X >= Min" section of the line, which is just the first operand of the decision.

A few other facilities were introduced to support, for example, cases where a single statement spans multiple lines, where expectations vary for different versions of the tool or the compilation toolchain, or the use of a common driver for the assessment of several coverage criteria. The exact details are beyond the scope of this article.

```
with In_Range, Support;

procedure Test_GTmax is
begin
   Assert (not In_Range (X => 4, Min => 2, Max => 3));
end;

--# in_range.adb
--  /expr_eval/  l! ## dT-
--  /expr_true/  l- ## s-
--  /expr_false/ l+ ## 0
```

FIGURE 4 This sketch of driver source illustrates a Decision Coverage test over the source file providing the In_Range function (in_range.adb), where the function is called once using X > Max as in the previous assessment example.

```
--  /expr_eval/  l! ## c!:"X >= Min"
```

FIGURE 5 The sample expectation line here depicts a hypothetical test over the In_Range function example used previously, where c!: "X >= Min" states the expectation of an incomplete condition coverage diagnostic over the "X >+ Min" section of the line.

Execution model overview
The general execution model underlying the tool testing framework consists of reasoning about sets of coverage result indications, referred to as coverage notes. Four kinds of coverage note objects are handled by combining two independent aspects: the note origin and the kind of output format a note applies to.

Regarding the note origin, the following are distinguished:

- *Expected notes*, coming from a statement of an expected result, and
- *emitted notes*, found in reports produced by the tool.

Regarding the kind of output format, the following are defined:

- *Line notes*, for coverage indication characters in an annotated source, and
- *violation notes*, for violation messages found in a coverage text report.

The *xp-source-line-note* in a */tag/* line spec is then internally modeled as an *expected line note* object. The *xp-violation-notes* are modeled as *expected violation note* objects, and *emitted line* or *emitted violation* note objects are extracted from the coverage reports produced by the tool.

Essentially, the test suite engine performs the following steps for each test:

1. Parse the test sources to construct sets of *expected* line and violation notes, one set of each per functional source. The engine matches */tag/* specifications in driver sources with tagged lines in the functional sources and instantiates

individual note objects with a specific kind and source location information.
2. Build the executable from the sources, execute it, and run the coverage analysis tool for the desired criterion, producing coverage reports.
3. Parse the reports to construct sets of *emitted* line and violation notes.
4. Match the expected line/violation notes against the emitted ones and report differences. A test passes when the tool has reported all the expected coverage indications against the assessed criterion *and* there is an expectation for all the coverage deviations reported by the tool.

Main characteristics of the scheme

An important characteristic of the scheme is to place the coverage result expectations for a test literally within the sources that drive how the functional code is exercised, and hence which pieces are covered and to what degree. This makes it convenient to verify consistency between what the test code does and the corresponding expected coverage results are, and provides a straightforward mechanism to document the connections between the two through comments in the source.

Another key aspect is the development of a specialized syntax to describe expectations, encouraging test writers to actively think about the expected results. This departs from methods using comparisons with baseline outputs, where baselines are typically obtained by generating outputs with the tool and verifying that the outputs are correct. There is no way to generate the specifications of expected results within this framework.

Similar ideas are used in some variants of test suites based on the DejaGNU framework (www.gnu.org/software/dejagnu), such as the one used by the GCC project (gcc.gnu.org).

The approach is also interesting for long-term maintenance of test suites. First, any changes in the report formats are handled by adjustments to the test suite execution engine, which are very localized and well controlled. This differs from baseline-oriented frameworks where changes to the report formats usually incur adjustments to the full test baseline, which becomes tedious and error prone when the test base grows large. Second, test source maintenance is also easier since the coverage expectations are entirely disconnected from the designated lines' relative positions in sources. Comments may be added or subprograms reordered, for example, without needing to update the expected results.

The main shortcoming of the framework is specialization. It is currently tailored for coverage analysis tools, and the code supports just the tool for which the environment was originally developed. Nevertheless, generalization is possible in a number of directions. Support has been developed for the C language, for example, and could be added for other languages based on customer demand. The framework could also

Get your free one-day ticket today!
Simply enter the following code at embedded-world.de/voucher:
2ew17P

Nuremberg, Germany
14 – 16.3.2017

embeddedworld
Exhibition&Conference
... it´s a smarter world

Get up to speed with the latest developments in your industry!

embedded world is THE meeting place for the international embedded community – secure your crucial knowledge advantage now!

Trade fair organizer
NürnbergMesse GmbH
T +49 9 11 86 06-49 12
visitorservice@nuernbergmesse.de

Conference organizer
WEKA FACHMEDIEN GmbH
T +49 89 2 55 56-13 49
info@embedded-world.eu

embedded-world.de

Media partners
elektroniknet.de
computer-automation.de
ENERGIE & TECHNIK Solutions for a Smarter World
DESIGN & ELEKTRONIK KNOW-HOW FÜR ENTWICKLER
MEDIZIN-und-elektronik.DE
Elektronik
Elektronik automotive
Markt&Technik
Computer & AUTOMATION
MEDIZIN+elektronik

NÜRNBERG MESSE

be adapted to other coverage analysis tools when the tool features a command line interface. There is no fundamental limitation in this area.

Abstraction capabilities

The tags scheme, which allows designating sets of lines, provides greater abstraction than a mere individual line naming facility where each specific line would need to be matched mechanically by an expectation. Actually, the tag in */tag/* lines is interpreted as a regular expression, so there are powerful ways to construct elaborate line set patterns and a careful selection of tags can help simplify the expression of expected results significantly. In a way, devising a set of tags for a test can be perceived as defining a very basic micro language for designating source line sets, and from this perspective the tags scheme offers a kind of meta language that gets instantiated for every test.

Another level of factorization is allowed by the capability to share sources between tests, and in particular to have a common set of driver sources for different implementations of a functional idiom.

As an example, consider the goal of testing proper behavior of the tool on a Boolean expression like "A and then B" in Ada. A natural starting point is the simplest case where A and B are simple Boolean variables, with functional code such as that shown in Figure 6.

To exercise the code in Figure 6 a few drivers could be written that call the Eval_Andthen procedure directly in different manners, once or multiple times, passing different values to A and B and stating the expected results accordingly.

It is realized that additional tests would be of interest for functional code with operands more complex than elementary Boolean variables. If such tests are written as independent entities, starting from the basic case as a model, it is almost immediately apparent that the set of driver sources needed is extremely similar to the one that was first written; just call in the functional code differently and have identical coverage expectations.

Instead, an environment can be set up where each set of tests for a kind of operand provides a helper API that the driver code can always use in the same fashion, regardless of the actual operand kinds. The driver code in Figure 7 provides an example, where FUAND stands for "Functional And". The helper package is expected to provide an "Eval_TT_T" subprogram that calls into the functional code, arranging for both operands to evaluate True (_TT_) so the decision evaluates True as well (trailing _T).

Adding tests for a new combination of operand kinds then just requires providing the functional code and helper package, and the addition of a driver source automatically benefits all the operand kind variants already in place. This is a pretty powerful mechanism, which can even be generalized further to support coverage assessments for decisions in general contexts, not only as controlling expressions in if-statements.

```
procedure Eval_Andthen (A, B : Boolean; Result : out Boolean) is
begin
  if A and then B then  -- # expr_eval
    Result := True;     -- # expr_true
  else
    Result := False;    -- # expr_false
  end if;
end;
```

FIGURE 6 Shown here is an example test of the structural coverage analysis tool's behavior using the simple Boolean expression "A and then B" in the Ada language.

```
procedure Test_FUAND_T is
begin
   FUAND_Helper.Eval_TT_T;
end;

--# fuand.adb
--  /expr_eval/   l! ## dF-
--  /expr_true/   l+ ## 0
--  /expr_false/  l- ## s-
```

FIGURE 7 The example driver code displayed here utilizes a helper API that can always be used in the same way regardless of the type of operand, where the "Eval_TT_T" subprogram calls into functional code to initiate evaluation of True (_TT_) and (_T) by both operands.

Summary and perspectives

As part of the development of a framework for in-house testing of a coverage analysis tool, we have devised an approach where the expectations on coverage results are expressed as special comments within the test sources. A few important aspects of these scheme have been outlined here. The framework described encourages active thinking about what the expected results should be for each test, and offers abstraction facilities that allow the factorization of development and maintenance efforts.

The approach described served as the basis of our GNAT coverage tool qualification for several industrial projects that used the tool as part of DO-178B and DO-178C certifications in the avionics domain, up to the most stringent certification level, which requires MC/DC. Based on this work we are evaluating possible ways to formalize aspects of our testing strategies for coverage analysis issues, in particular regarding the implications of proper MC/DC testing with respect to expression topologies, expression context, and the kind and complexity of operands. ECD

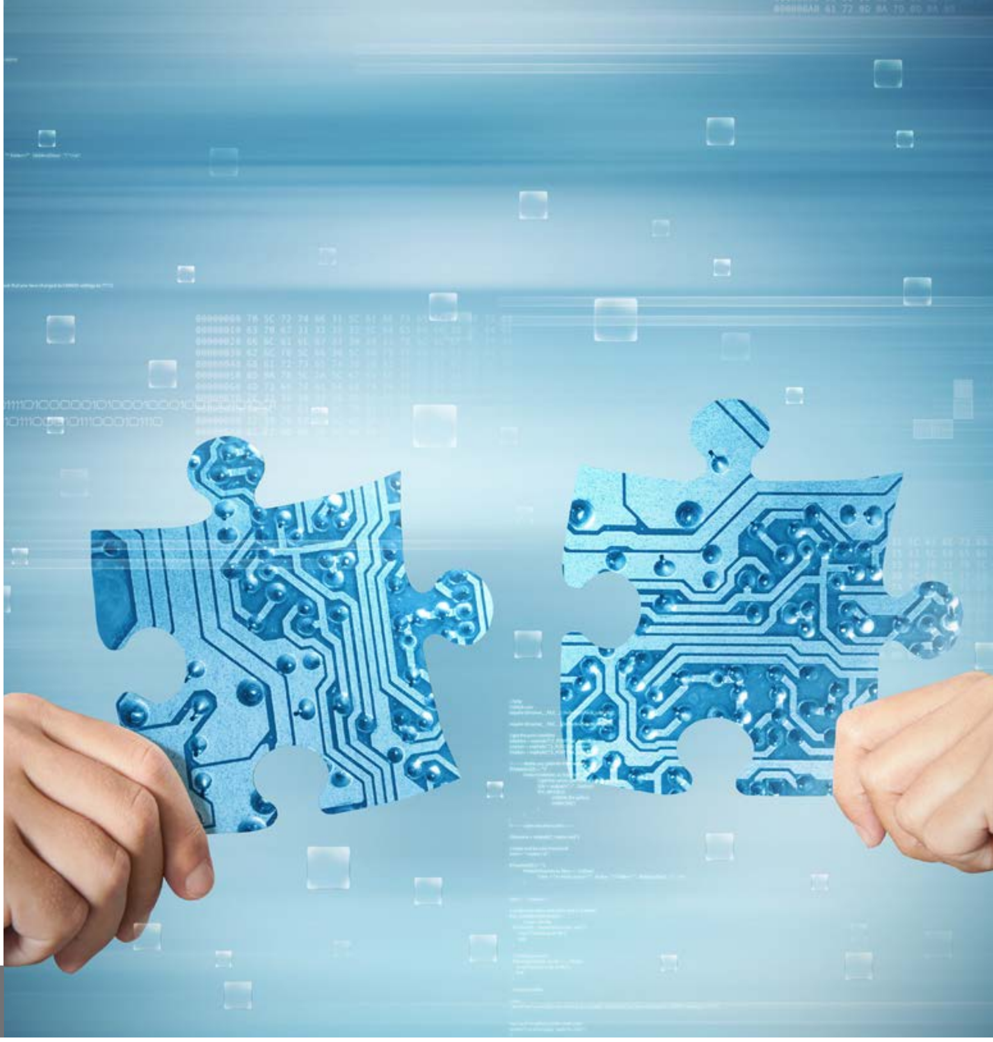
Olivier Hainque *is a Senior Software Engineer at AdaCore. His current role includes overseeing the GCC compiler suite for AdaCore's coverage analysis solution, as well as team and projects coordination. He received a Ph.D. in Computer Science in 2000 for work in the area of Synchronous Languages – the separate compilation and distributed execution of Esterel applications, in cooperation with a French aircraft manufacturer.*

References:

1. RTCA. (2011, December 13). RTCA DO-178C/EUROCAE ED-12C, Software Considerations in Airborne Systems and Equipment Certification. RTCA.
2. Kelly J. Hayhurst, D. S. (2001, May 1). A Practical Tutorial on Modified Condition / Decision Coverage. NASA.

HARDWARE EMULATION FOR MULTI-LEVEL DEBUGGING METHODOLOGY

By Lauro Rizzatti



Chip design debug is a difficult discipline, and system on chip (SoC) design has made it more so. It's like the proverbial needle in the haystack. For SoC designs it's two haystacks, one for software, the other hardware. Software development groups often point a collective finger at the hardware group claiming it's a hardware bug, while the hardware group snaps back, claiming it is a software bug. It's hard to know who's right without effective verification tools to pinpoint the problem. That's where hardware emulation comes in.

Hardware emulation can be invaluable for debugging hardware and for testing the integration of hardware and software within SoC designs well ahead of first silicon. When two disparate parts of the engineering group – hardware designers and software developers – use emulation, they're able to share the same system and design representations. Combined software and hardware views of the SoC design enable them to work together to debug hardware and software interactions.

As the foundation of most SoC verification flows, hardware emulation allows engineering groups to plan more strategically and implement a debugging approach based on multiple abstraction levels. Engineering groups don't have to dive into two haystacks independently of one another. Instead, they can track a bug across the

Rizzatti, LLC
www.rizzatti.com

E-MAIL
Lauro@rizzatti.com

LINKEDIN
www.llinkedin.com/in/lauro-rizzatti-2165a5

boundary between the embedded software and the underlying hardware to determine whether the problem lies in the software or in the hardware.

Implementing a debugging methodology based on multiple abstraction levels starts with embedded software at the highest level, and moves down in abstraction levels to trace the behavior of individual hardware elements. In fact, starting from a database comprising multi-billion clock cycles, a software debugger can localize a problem to within a few million clock cycles. At this level, either the software developers can identify the source in the software code or their hardware design counterparts can use a software-aware hardware debugging approach to focus on a lower level of abstraction. The methodology calls for monitors, checkers, and assertions implemented through hardware transactors that avoid speed degradation and help narrow down the problem to a few thousand cycles.

Once the collected data at those two levels has been reviewed, hardware emulation allows the engineering group to move down to the signal level. It can analyze the information via the register transfer level (RTL) waveforms of the identified period of time and trace its likely origin. Either a hardware bug is unearthed or the hardware is cleared of failure. If it's the latter, it forces the decision to move back to the software environment.

Navigating multiple levels of debug abstraction

Navigating between different levels of abstraction – from software through hardware and back – avoids long simulation runs and wading through massive amounts of detailed data (Figure 1).

The multi-level debugging methodology would not be possible with software simulators because they are too slow to effectively execute embedded software. Indeed, they would run for many months to process billions of cycles on designs whose sizes reach into several hundreds of millions of application specific integrated circuit (ASIC) equivalent gates. This is an unacceptable time constraint for suppliers of consumer electronics devices, or any other electronics devices for that matter.

"WHILE STILL WIDELY USED, THE IN-CIRCUIT-EMULATION MODE, THE ORIGINAL STYLE OF EMULATION THAT DROVE ITS SUCCESS IN THE VERIFICATION SCENE, NOW FACES A VIABLE ALTERNATIVE IN TRANSACTION-BASED VERIFICATION."

While still widely used, the in-circuit-emulation (ICE) mode, the original style of emulation that drove its success in the verification scene, now faces a viable alternative in transaction-based verification. Conceptually, the idea is simple. Tests are written at high levels of abstraction and the conversion from high-level commands to bit-level signals is moved from the testbench into a dedicated entity called a transactor. By mapping the transactor onto a hardware emulator, acceleration of five or six orders of magnitude compared to simulation-based verification can be easily achieved.

Engineering groups use transactors to build a virtual test environment instead of the ICE physical target system by replacing a set of I/O protocol-based speed adapters with an equivalent set of transactors (Figure 2).

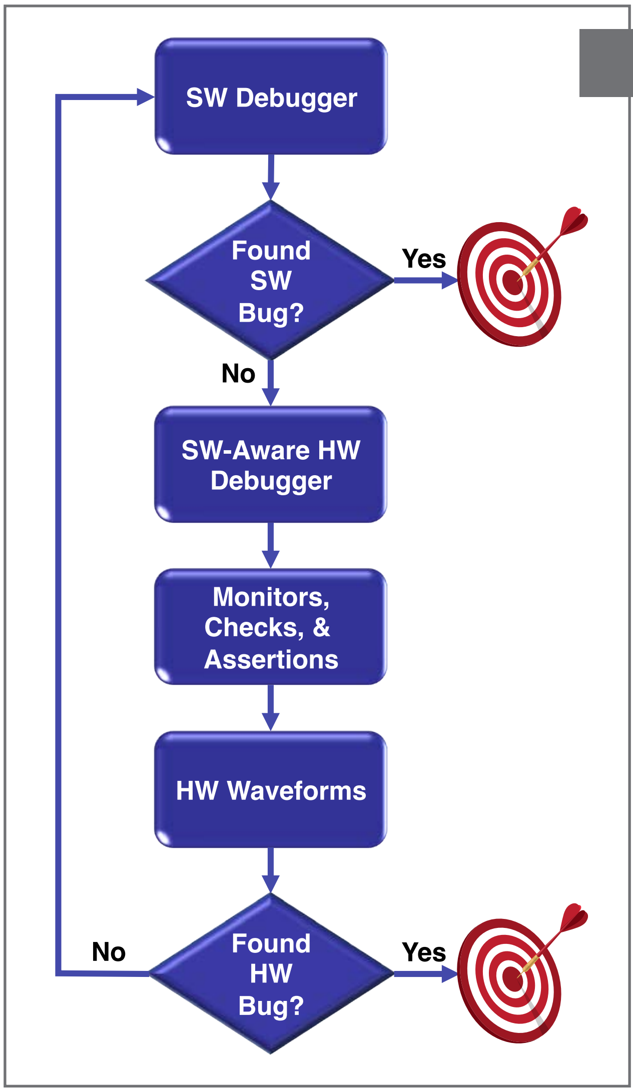


FIGURE 1

Hardware emulation provides an ecosystem for software and hardware debugging.

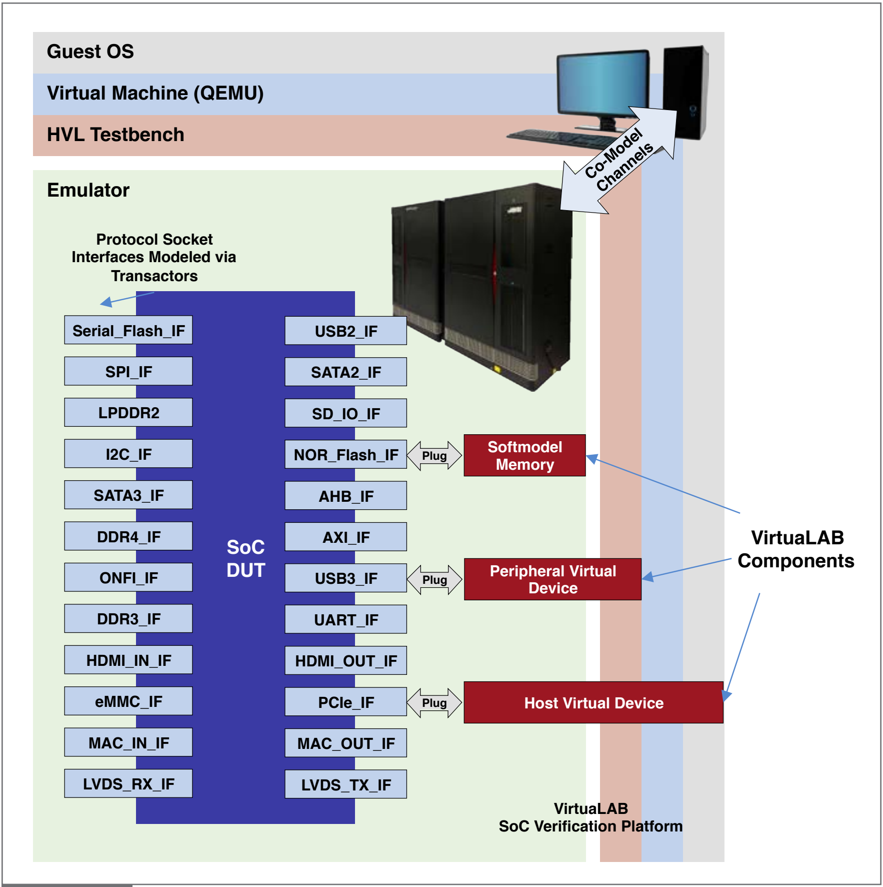


FIGURE 2 A complete virtual test environment includes all SoC peripheral interfaces modeled via transactors.

Design debugging is simplified in transaction-based acceleration. By gaining full control of the design clocks that aren't sourced by the hardware test bed, debugging becomes easier and more efficient. By controlling the clock frequency, it's possible to stop the emulated design-under-test (DUT) model, read its memory content, force some registers, or dump the waveforms.

Traditionally, debugging in an ICE environment required hardware logic analyzers driven by the uncontrollable clocks coming from the target system. The setup caused nondeterministic behavior and compromised the ability to debug the DUT. Hardware emulation vendors recently addressed the random behavior of the ICE periphery with methods to convert it into a deterministic behavior.

A multi-level perspective of co-verification

Once software designers and hardware developers experience transaction-based verification using hardware emulation, their entire verification perspective changes. The ability to quickly setup a powerful test environment free of cumbersome ICE hardware means easier and more effective debugging. The goal may be the same – better designs in less time – but the experience now can be something far less challenging.

Engineering groups are finding that modern hardware emulators are requisite to test the hardware and integrate hardware and software in SoC designs. It allows them to plan more strategically and to successfully implement hardware/software co-verification. ECD

Dr. Lauro Rizzatti is a verification consultant and industry expert on hardware emulation. Previously, Dr. Rizzatti held positions in management, product marketing, technical marketing, and engineering.

COMMELL

PICMG 1.3 fanless barebone New
based on i7-6820EQ Skylake-H
CMB-B72

expansion capability
performance
high-end media
broad I/O
fanless

6th Gen Intel® Core™ i7/i5/i3(Skylake)
Skylake-U series New

Mini-ITX,	Pico-ITX,	3.5" Miniboard
LV-67T,	LP-175,	LE-37G

New
Skylake-H series & Skylake-S series

Mini-ITX, 3.5" SBC	ATX MB, Mini-ITX
LV-67R, LE-37I	AS-C7A, LV-67S

Intel® Braswell fanless Embedded New

Mini-ITX,	Pico-ITX,	3.5" SBC
LV-67P,	LP-176,	LE-37F

Qseven 2.0,	Pico-ITX,	3.5" Miniboard
QE-E71,	LP-174,	LE-37E

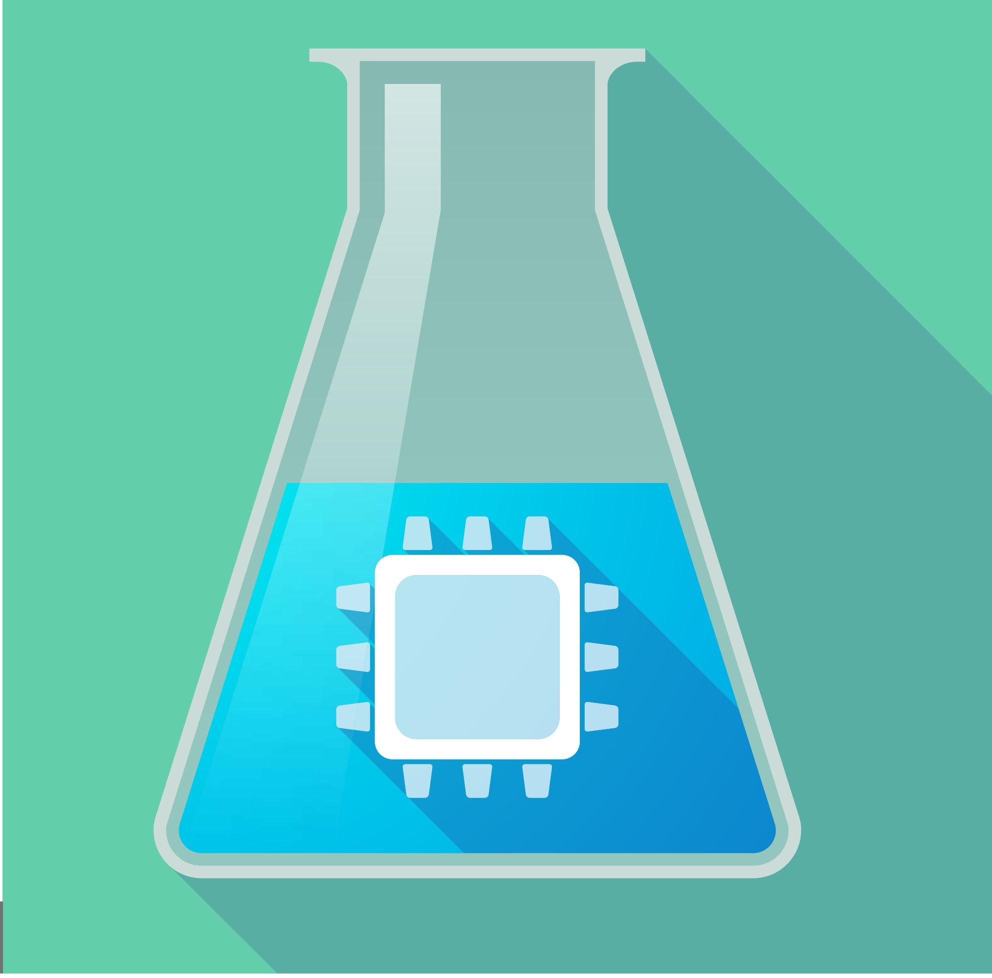
5th Gen Intel® Core™ i7/i5/i3

PICMG 1.3 Half-size, Full-size
HE-B72, HE-B71, FS-A78
Skylake-H, 4th Gen Intel® Core™ i7/i5/i3
New

www.commell.com.tw
Taiwan Commate Computer Inc.
info@commell.com.tw

"PORTABLE STIMULUS": SYSTEM-LEVEL VERIFICATION TRENDS FOR 2017 AND BEYOND

By Adnan Hamid



The functional verification space has had more innovation than any other part of the front-end design flow, and yet the amount of time and effort spent in verification continues to grow. The problem stems from rising complexity and the fact that simulation as a technology has failed to scale ever since single processors stopped becoming more powerful. It is compounded by an increasing number of tasks that verification is expected to perform, such as power verification.

The electronic design automation (EDA) industry dealt with the transition from a single-point tool (simulation) at the heart of the verification effort, as well as from languages (SystemVerilog) and methodologies (universal verification methodology (UVM)) developed to verify and validate an accelerator approach to system design. Today, verification is a flow that has to accommodate several execution platforms in addition to subsystems that likely contain one or more processors that cannot easily be taken out of the design for the purpose of verification. When such subsystems are themselves integrated, this becomes an impossible approach.

Another problem is that existing verification methodologies focus on stimulus generation and cannot take into account the intended purpose of the design. This has resulted in a verification methodology that has become increasingly inefficient and ineffective.

Reaching "Portable Stimulus"

When looking at a suitable replacement verification methodology, several important requirements have to be considered:

› First, it has to be able to treat verification as a flow, meaning that it has to be able to work at multiple levels of abstraction and target various execution engines, ranging from virtual prototypes through simulation, emulation, and physical prototypes to actual silicon.
› The second major requirement is that it has to be composable. If a verification model is developed for a subsystem, it must be able to be incorporated easily into a higher level model without requiring extensive rework.
› A number of other requirements address issues such as readability and coverage, in addition to those that define the basic capabilities of such a system, such as the ability to define sequential behaviors and constraints.

"THE NEW VERIFICATION INTENT MODEL WILL CONTINUE TO ENCAPSULATE NOTIONS OF RANDOMIZATION, PROVE TO FIND ISSUES THAT NOBODY THOUGHT TO WRITE A TEST FOR, AS WELL AS ACCEPT THAT SOFTWARE PLAYS AN INCREASINGLY IMPORTANT ROLE IN SYSTEM FUNCTIONALITY."

As industry demand for such a solution grew, the Accellera Systems Initiative initiated an effort to design such a language in February 2015. Companies such as Breker Verification Systems and Mentor Graphics had commercial offerings in this space for some time and were able to present their market experience to the committee. During the process, new user requirements and concerns were raised and solutions evolved to address those needs.

Two years later that effort is close to the first release of a new verification language that is internally being called "Portable Stimulus." What makes this different from previous verification languages is that this is not a model that helps with the generation of stimulus, but a definition of verification intent. Instead of concentrating on what legal combinations of inputs look like, it focuses on end-to-end functionality that should exist within a design.

The objective of Portable Stimulus is to be able to write your verification intent once and be able to use it at all stages of silicon realization (Figure 1). With a Portable Stimulus model it is possible to generate constrained random test cases, just as in UVM, but these generated tests can be self-checking instead of requiring a separate scoreboard implementation. It is also possible to produce metrics of design intent coverage directly from the model, which is different from developing a separate functional coverage model that only tells how much of the implementation has been exercised rather than how much of the intended design functionality has actually been tested. In addition, a single Portable Stimulus model can be used as an input to synthesize tests for a variety of target execution platforms, including UVM, simulation, emulation, post-silicon validation, etc.

The Portable Stimulus verification intent model will continue to encapsulate notions of randomization, prove to find issues that nobody thought to write a test for, as well as accept that software plays an increasingly important role in system functionality. Therefore, using Portable Stimulus, the processor can be considered a resource within the design to be exploited as opposed to part of the problem. Models for processors and their implementation tend to be some of the most thoroughly verified pieces of the design, and can be used as trusted agents in the verification of the blocks around them and the connectivity that connects them.

The Portable Stimulus verification solution

The Accellera committee responsible for the Portable Stimulus initiative decided to support two methods of developing a Portable Stimulus model. The first method is to write models using simple C++ constructs, and the other is a new domain-specific language with specialized declarative semantics. Both approaches have matching semantics so models can be freely passed between one form and the other.

However, three important aspects remain for a viable Portable Stimulus solution:

› First, Portable Stimulus models must be graph-based in order to naturally capture the intended flow of execution through the design. The graph being captured is equivalent to a Unified Modeling Language (UML) activity diagram, which is a flow chart that can be considered a graph.

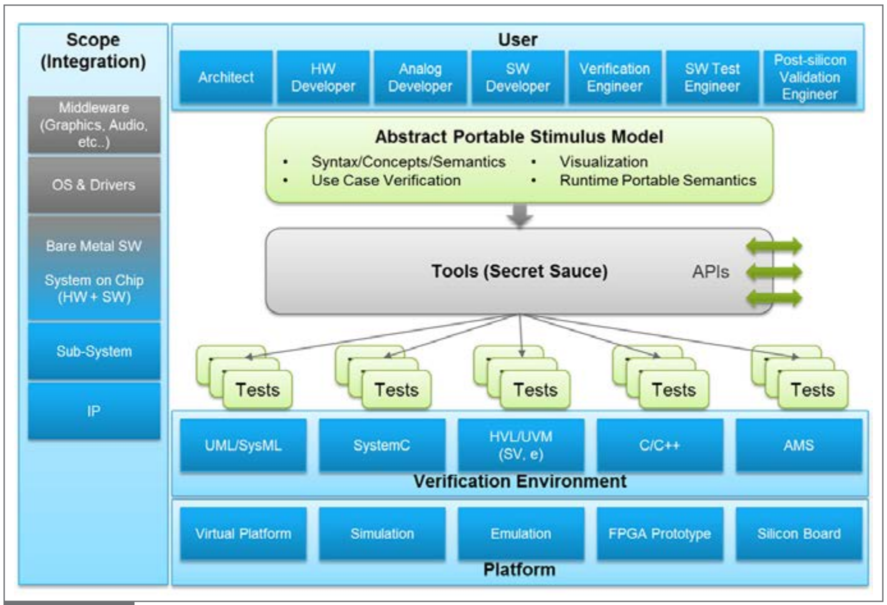


FIGURE 1 Accellera will soon release a language called "Portable Stimulus," a way to define verification intent.

- Second, Portable Stimulus must provide an abstraction of the hardware/software interface so that, for example, a tool can map a register write to either a UVM verification IP (VIP) or a software-driven test.
- Third, Portable Stimulus models must be composable so that lower level models can be seamlessly combined to define higher level use cases.

The general form of the resulting solution is a tool tasked with generating a self-checking test scenario. This includes finding a random, but legal, path through the graph that satisfies the constraints provided, which in turn defines the scenario that will be run. The tool may then generate software that will be compiled and executed on processor cores contained within the system on chip (SoC). In addition, there may be events that need to be injected into the SoC, and it may be necessary to time these with events happening from within the SoC. This requires that a testbench is constructed that is capable of performing such functions.

This approach leaves plenty of room for innovation in the capabilities of tools and the quality of test cases generated for various targets. It also opens up a whole new class of tools that would surround the verification intent model. What is described can be seen as a verification synthesis tool that can take a high-level model as input and generate specific test cases. Other tools could concentrate on coverage or prioritizing the progress of the verification effort, as well as debug tools that could help with identifying insecure paths through a system that could result in security vulnerabilities.

System-level verification, 2017 and beyond

The future is exciting and users have not waited for the completion of the Portable Stimulus standardization effort. Many advanced design houses have developed ways to inject this new methodology into their flows already, and the whole industry continues to learn more about the task of system-level verification. Thankfully, with the support of C++ models, it should be extensible well into the foreseeable future and able to address new challenges as they arise. It also has the advantage that plenty of people already know the language, so getting up to speed should be fast.

2017 could be a great year for verification. ECD

Adnan Hamid is Founder and Chief Executive Officer of Breker Verification Systems, and inventor of its core technology, Trek.

Embedded World Profiles

Military

Acromag®
THE LEADER IN INDUSTRIAL I/O

AcroPack® PCIe-Based I/O Modules

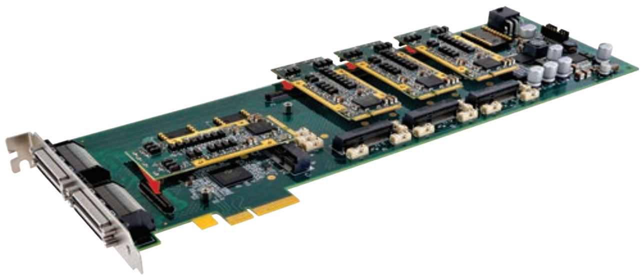
AcroPack® mezzanine modules deliver maximum I/O density when used with AcroPack carrier cards for PCIe or VPX-based systems. Combining different AcroPack module types on one carrier allows for a simplified modular approach to system assembly.

Designed for COTS applications these general purpose I/O modules deliver high-speed and high resolution A/D and D/A, digital I/O, serial communication, and re-configurable FPGA functions.

To further simplify the integration of AcroPack mezzanine modules into your host system, they maintain the same functionality and memory map of existing Industry Pack (IP) modules.

Acromag provides software support packages for embedded applications running on Linux®, Windows®, or VxWorks® operating systems. The libraries provide generic routines to handle reads, writes, interrupts, and other functions. Demonstration programs enable the developer to quickly exercise the I/O modules before attaching the routines to the application program. This diagnostic tool can save hours of troubleshooting and debugging.

See the AcroPack I/O modules at Embedded World 2017 in booth #2-408



FEATURES

- Mix and Match endless I/O combinations in a single slot
- VPX and PCIe carriers
- Linux, Windows, and VxWorks support
- Sample software and diagnostics
- Solid Down connector I/O interface
- -40 to 85°C operating temperature
- A/D, D/A, serial, digital I/O and FPGA

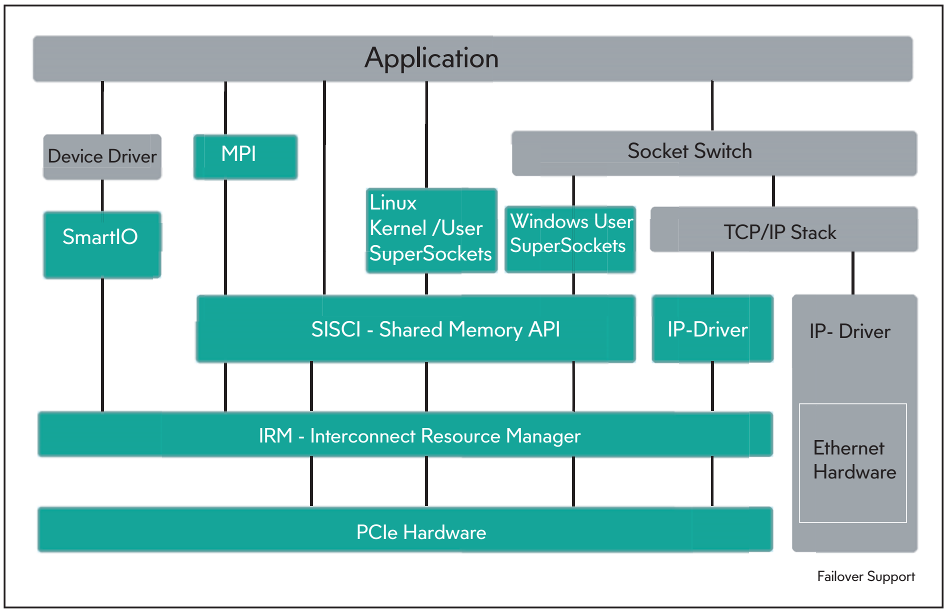
embedded-computing.com/p374017

Acromag
www.acromag.com/acropacks

sales@acromag.com
www.linkedin.com/company/acromag
248-295-0880
www.twitter.com/acromag

eXpressWare™ PCI Express Software

Dolphin's eXpressWare™ is a complete software suite for PCIe Networking that includes support for sockets, TCP/UDP, and shared memory applications. This comprehensive application development environment includes advanced PCIe features, such as reflective memory or multi-cast, peer to peer communication, and Dolphin smart I/O technology. For customers seeking a software solution for PCI Express, this suite is ideal as it can be licensed for custom designs and supports Dolphin's PCI Express hardware products. eXpressWare™ software includes three main components: a complete low level API for shared memory applications (SISCI), a robust low latency sockets API (SuperSockets), and a standard TCP/IP driver (IPoPCIe). These three components create a comprehensive development and deployment environment for applications using PCIe as a network between systems.



Networking with PCIe delivers performance and reliability. eXpressWare™ software is designed to easily enable PCIe networking. It maximizes the low latency and throughput advantages of PCI Express, enabling customers to build scalable networked systems with PCIe cabling or across a backplane. The software suite supports a variety of chipsets from major silicon providers such as IDT, PLX/Broadcom and Microsemi. eXpressWare™ software supports standard form factor boards and switches from Dolphin, but can also be customized to support third party boards and applications.

eXpressWare™ supports most common Linux distributions including 2.6 and 4.x releases, along with Windows, RTX, VxWorks, and Redhawk. eXpressWare™ customers can exploit the full capabilities of PCI Express while implementing standards based software.

FEATURES

› Fast FPGA and GPU transfers
› Shared Memory API
› Reflective Memory Solution
› Dolphin Smart I/O Software
› PCIe Peer to Peer Transfers
› Fast Sockets implementation
› TCP and UDP transfers

embedded-computing.com/p374016

Dolphin Interconnect Solutions
www.dolphinics.com/products/dolphin_pci_express_software.html

paraison@dolphinics.com   +1-603-747-4100
in www.linkedin.com/company/dolphin-interconnect-solutions

Embedded/IoT Solutions

Connecting the Intelligent World from Devices to the Cloud

Long Life Cycle · High-Efficiency · Compact Form Factor · High Performance · Global Services

IoT Gateway Solutions

E100-8Q, E200-9B

Compact Embedded Server Appliances

SYS-5028D-TN4T

Network, Security Appliances

SYS-5018D-FN8T (Front I/O)

High Performance / Short Depth Solutions

SYS-1028R-WMRT

Cold Storage

SYS-5018A-AR12L

High Performance / IPC Solutions

SYS-6018R-TD (Rear I/O)

- Low Power Intel® Quark™, Intel® Atom™ Intel®, Core™ processor families, and High Performance Intel® Xeon® processors
- Standard Form Factor and High Performance Motherboards
- Optimized Short-Depth Industrial Rackmount Platforms
- Energy Efficient Titanium - Gold Level Power Supplies
- Fully Optimized SuperServers Ready to Deploy Solutions
- Remote Management by IPMI or Intel® AMT
- Worldwide Service with Extended Product Life Cycle Support
- Optimized for Embedded/IoT Applications

Intel Inside®. Powerful Productivity Outside.

TITANIUM We Power the Cloud™

Learn more at www.supermicro.com/embedded

© Super Micro Computer, Inc. Specifications subject to change without notice.
Intel, the Intel logo, Intel Core, Intel Quark, Xeon, and Xeon Inside are trademarks or registered trademarks of Intel Corporation in the U.S. and/or other countries.
All other brands and names are the property of their respective owners.

03_Q3_SM_USP_070616_MasterFile