WHITE PAPER

Vikas Varshney,
Engineering Manager,
Emulation Debug Tools

Debbie Greenstreet,
Marketing Director,
Wireless Base Station Infrastructure

Texas Instruments

KeyStone Multicore SoC Tool Suite: One platform for all needs

Abstract

Historically, a significant challenge in implementing and deploying multicore System-on-Chip (SoC) devices has been the availability of proper tools to program and debug these platforms. For developers to take full advantage of the performance of multiple cores, efficiently partitioned and high quality software running on those cores is crucial. The debugging of complex multicore systems comes with additional complexity from their concurrent processing paradigm and the limited accessibility of sub-system interfaces. The strength of a tool suite is measured by the time it takes to develop and debug multicore SoCs, and the ability to achieve optimal SoC performance.

TI's Code Composer Studio™ (CCStudio) Integrated Development Environment with KeyStone extensions includes best-in-class multicore data visualization technology for debug, verification and trace capabilities. TI recently introduced new KeyStone multicore SoCs, with a heterogeneous mix of ARM® RISC processors and TMS320C66x DSP cores. The CCStudio tool suite with KeyStone extensions, developed in conjunction with the KeyStone devices and software, provides a single, system-level view into the SoC providing visibility to the ARM and DSP cores, accelerators and peripherals.

Introduction

As processors have become more capable they have also become more complex. This makes the developer's ability to debug, troubleshoot and maintain software more challenging. With the introduction of multicore processors the rigors of implementation and analysis increase exponentially.

Consider the following scenario: a team of talented engineers ponders the prospect of developing a new wireless base station with an exciting new SoC. This new device holds the promise that the engineers' company can launch a competitive new product to the market quickly and efficiently. While the selected SoC is a heterogeneous multicore device with a mix of RISC and DSP cores, the engineering teams are functionally organized as separate ARM and DSP development teams. They understand that they will need to work together to troubleshoot combined system software on a single platform. The question is, are there tools available to navigate the complexity of the SoC?

The manager of the engineering team has similar reservations. While the new SoC promises a quantum leap in performance, significantly reduced power and lower costs over existing solutions, the manager knows that poor debug and analysis tools have been to blame for indefinite delays and non-deterministic schedules during the integration, debug and system test process with alternative SoCs. One potential benefit of this SoC platform is the promise of software reuse across of multiple products, but the manager is concerned whether the tools to support this SoC will truly allow for reuse and retest in a timely fashion.

Finally, the business manager of the product line is also wary. With her experience deploying increasingly complex products, she is well aware that despite the wonderful new features and benefits of the latest generation SoC, it is up to the business team to take advantage of this and specify features that the development teams can implement quickly and successfully. Time to market is a critical factor on the business side of product deployment and any glitches in debugging; integrating, testing and trialing the product will have negative impacts on revenue and profit.

So what specific capabilities are engineers looking for in development tools for multicore SoC devices?

- First, developers need a **global view of processing elements**. Not only do they need to see the events on a particular processing core, but they need to see all of the processing on all of the cores simultaneously.
- Second, they need to be able to **visualize the communication between processing elements**. To identify challenging failures such as the source of excessive delay, a view into all of the interactions, as each of the processing elements are executing, is essential.
- A key capability required of a multicore tool is the ability to **measure the utilization of each element of the SoC** to determine if the processing cores are appropriately utilized. Developers need to determine if a processing element is tasked to a level approaching overload and, if so, how to rebalance the SoC.
- Lastly, the ability to **identify blocked processes and to determine processing deadlocks and system inefficiencies** is required. The engineering team developing a software solution on a multicore SoC must have the confidence that the entire system works in real time and is designed and partitioned to optimize the silicon architecture. To be assured of this, software developers must have a strong real-time view of their solution at work. The result is a well integrated product that is ready for field trial and ultimately deployment. Figure 1 exemplifies an example of TI's KeyStone architecture in the debug paradigm.

To satisfy these requirements, a tool suite must **understand and report on the performance of each of the processing elements and interfaces** of the SoC. The tools must be able to show synchronization and timing relationships of all processing elements. Showing each standalone core or programmable entity is not enough.

Developers of products using TI's KeyStone multicore SoC devices enjoy best-in-class multicore development and debug tools based on TI's CCStudio tool suite. Developers can identify and resolve deep system-

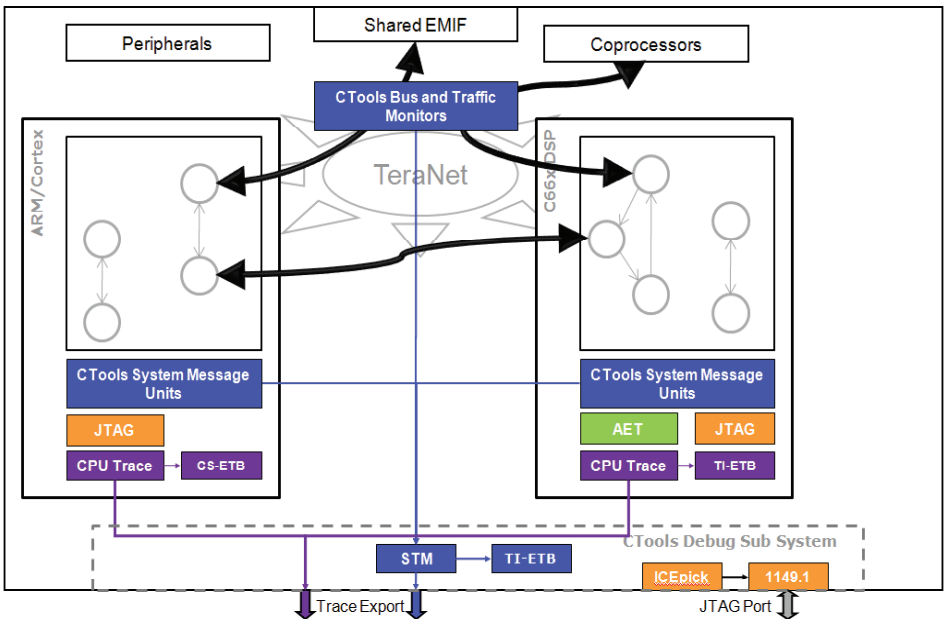

Figure 1. KeyStone heterogeneous multicore SoC architecture with debug and trace perspective.

level issues swiftly and efficiently ensuring optimized integration and test cycles. They can also leverage this detailed insight once products have been deployed with remote trace capability. This enables fast response to field issues and an ability to excel in supporting their end customers, the service providers and operators. As TI introduces its first set of heterogeneous multicore devices, adding an ARM® processor to the existing DSP multicore mix, it now extends the same level of coverage and analysis to ARM developers via the CCStudio tool suite with KeyStone extensions.

The debug process begins

The Analyzer suite is another critical aspect of the tool chain. This includes trace capability, both at a core and system level, and logic analyzer functionality leveraging on-chip hardware event snoopers instrumentation and software instrumentation. The suite also includes Multicore System Analyzer, a KeyStone extension providing software instrumentation. Each tool is designed to support specific use cases leveraging various technologies that are synchronized with each other for a more complete system solution. Combined, the suite offers data correlation across cores and tools to a common global timeline, and supports a synchronized scrolling of views across tools.

CCStudio not only provides tools for developing code for DSP and ARM but also has Analyzer suite of tools to help with performance optimization phase of development as shown in Figure 2. CCStudio has been extended to incorporate Linux™ support in addition to the DSP support that CCStudio has long featured, Linux-based ARM developers can debug using the familiar Linux GDB debugger within the CCStudio environment. CCStudio provides capabilities to simultaneously debug ARM application running on Linux OS and the kernel in stop-mode enabling end-to-end tracking of execution flows.

The ability to control cores synchronously via the debugger is a very useful multicore debug feature enabling simultaneous program state inspection. This feature is very valuable when developers are trying to debug concurrent programs running across multiple cores for deadlocks and race conditions. Synchronous “locked” running and stepping is another critical debug feature in the symmetric multi-processing (SMP)

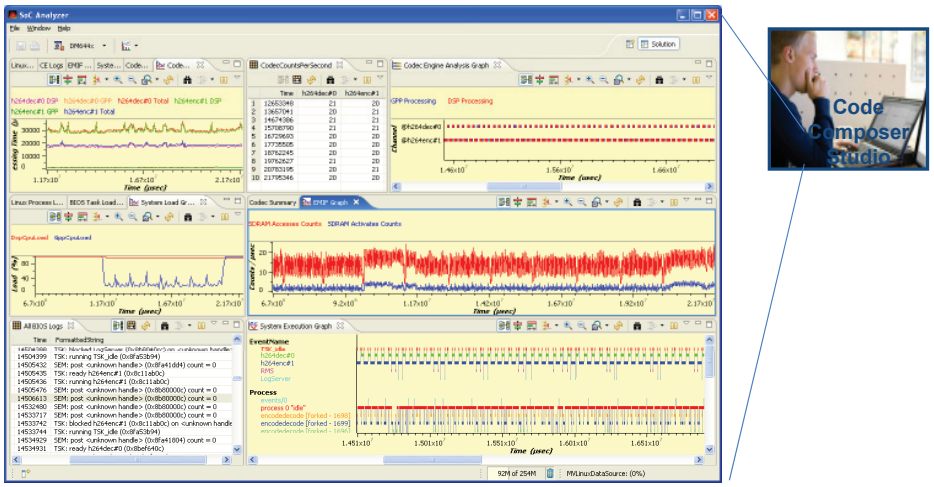

Figure 2. Multicore performance with single-core simplicity enables viewing and debugging DSP and ARM cores with a single integrated tool.

environment where interaction visibility needs to be understood at a granular level. CCStudio's multicore triggering feature is another valuable feature that is used to investigate cross-processor dependencies and behaviors. In a DSP and ARM® multicore triggering scenario, a processor or set of processors can be configured to trigger or respond to external processor events. For instance, a breakpoint hit on processor one can send a signal to processor two to halt or perform some other debug operation. This technique is helpful for identifying intermittent glitches, crashes, runaway code and bogus interrupts.

Core trace

Trace Analyzer runs at the at the core level, where developers have the ability to analyze and visualize CPU program and data traces. CCStudio provides different ways to setup and trigger trace collection at the point of interest. Data is captured using the on-chip Embedded Trace Buffer (ETB), or XDS trace receiver hardware, and is post-processed in the Trace Analyzer while taking advantage of KeyStone architecture hardware-level instrumentation and non-intrusive trace analysis. This enables software designers to optimize system performance using trace data to gain insight into inclusive and exclusive function CPU cycle stall profiling and cache profiling. CCStudio provides core trace support both for DSP and ARM cores. The CCStudio inclusion of traditional ETM trace tools allows ARM Linux™ developers to work in their preferred debug environment.

The Trace Analyzer enables debug of difficult real-time problems via several key features. First, it maintains a function call graph that allows the user to view the steps taken to arrive at the condition under evaluation. It also provides a log view for detailed PC trace data. Source code correlation features for DSP and ARM cores assist in identifying the relationship of each line of code to the current state of the executing system. The Analyzer provides advanced data navigation features including find, search and filtering controls, zooming and measurement markers, and scrolling for synchronous views. All of this is supported on the CCStudio tool base. In addition, the Trace Analyzer results can be exported to 'csv' format for viewing and analyzing elsewhere. Core trace has been one of the widely used technologies to provide instruction-level visibility into program execution sequences. Core trace provides core level execution visibility without code instrumentation, and is supported for both DSP and ARM core elements in TI's KeyStone architecture.

System trace

For conditions requiring debugging of system level issues, the KeyStone architecture has an on-chip System Trace Module. This module provides hardware-accelerated software instrumentation and hardware bus monitoring to "see" the transactions for each master to selected slave interfaces through tracing of key transaction points. System trace capabilities allow developers to monitor system transactions, capture data non-intrusively and post-process data using the CCStudio logic analyzer. For system-level visibility, the technology provides indispensable capabilities via instrumentation of traces from the cores, and combines them with hardware monitoring events outside the processor.

The Logic Analyzer tool offers a graphical event timeline view of the system trace events and messages, illustrating data throughput and analysis of use cases, as shown in Figure 3 on the following page. It manages a large number of event sources using either a hierarchical or a flat representation and offers advanced features such as zooming, measurement markers, bookmarks and sorting.

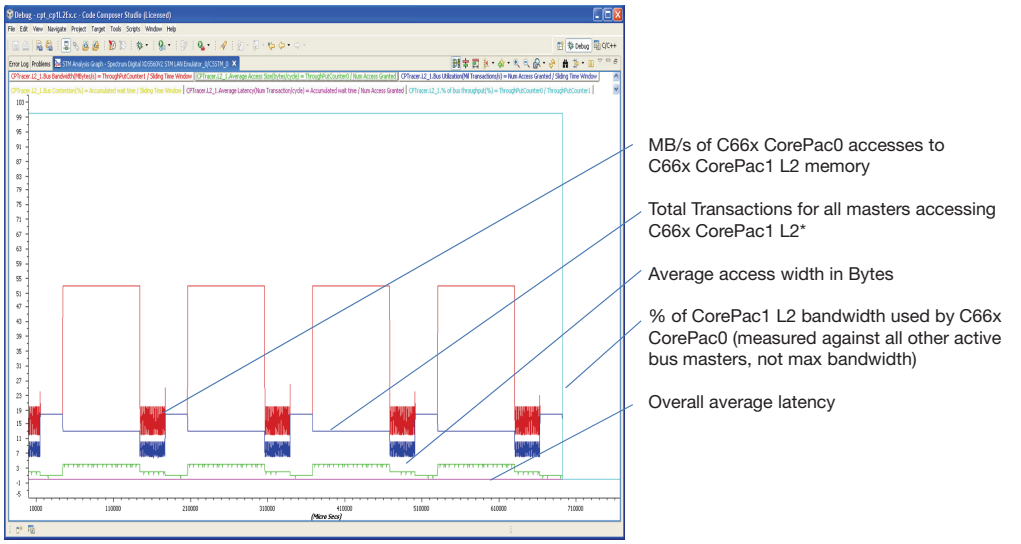

Figure 3: The Logic Analyzer correlates System Trace Module events.

Linux™ software developers can quickly take advantage of System Trace technology by using a loadable Linux System Trace Module (STM) character mode device driver. This allows the debug data and instrumentation logs to be routed to a STM port without any change in the application code, and with all of the comforts of the traditional "printf" debug function. For example, ARM® Linux developers can use standard C lib print functions and direct the output to the STM device without changing any code. This sends all of the instrumentation output to the System Trace port which automatically timestamps the message and allows correlation with messages coming from other applications or tasks.

Multicore System Analyzer

Hardware instrumentation-based core and system trace capabilities are extended using software instrumentation and a full view of the KeyStone elements as they process the integrated application software. The KeyStone Multicore System Analyzer (MCSA) is based on a Unified Instrumentation Architecture (UIA) which defines a set of APIs, interfaces and guidelines to collect data in real time using the software instrumentation. This allows instrumented components from different parts of the SoC to work together.

As with the Core and System Trace Analyzers, the MCSA supports live data analysis as well as capture and post-process modality, facilitating the software developers' test schedule and process. Runtime logging can be enabled and disabled. The MCSA makes analysis visualization easy, and provides functions such as execution graphs, duration analysis, context aware profiles, load analysis and statistics analysis. One of the key benefits of the MCSA is that it supports system analysis locally via an Ethernet or JTAG port. It can also virtually extend the Embedded Trace Buffer via Ethernet for remote debug and analysis. This allows remote software developers or testers to assist and participate in the application software integration and test process. These features facilitate remote access to deployed systems, greatly improving responsiveness to field reported bugs and issues.

Armed with this KeyStone SoC level analyzer, software team members can efficiently attack their design verification and debug of their system-level application software. System-level correlated "execution flow" with hardware-assisted monitoring events outside the cores is very helpful in identifying system-wide interaction issues. Developers achieve device level global time-stamping information embedded in the core, and system trace information can establish a common global time-base, which is a powerful way to time correlate system-wide events and understand cross dependencies.

Interconnect bus traffic snoopers, trap hardware events and transactions in a continuous system monitoring mode, providing significant visibility into the bus address and data monitoring. This visibility allows the team to analyze the SoC behavior and diagnose spurious transactions or events. The information collected by these snoopers can be further processed and correlated with core trace information, providing DSP and ARM® core bus level visibility from a unified perspective.

In addition, developers leverage bus transfer profilers and performance monitors to augment overall bus and traffic visibility. An example is DMA transfer profiling for channels, read, write and burst size. Several critical sets of performance optimization information are available. Throughput, channels interleaving, and transfer durations are examples of the performance data provided. The performance monitors provide non-intrusive visibility into the complex SoC interconnection network to help understand sustained data bandwidth and latency characteristics. This is crucial if real time performance goals are to be achieved.

CToolsLib

A suite of enablers called CTools are the primary on-chip debug and trace technologies behind the CCStudio tool suite and its KeyStone extensions. They provide tools at several layers; SoC system level, sub-systems and core support, including IEEE 1149.1 (JTAG). With CTools, the KeyStone tool suite supports the traditional JTAG external emulator-based debug, and also provides in-field (JTAG-less) debug and trace capability without the need for external debuggers or a trace collector. Through the use of the CToolsLib portfolio, software developers leverage a collection of embedded target APIs that enable easy access to the CTools debug and tracing capabilities. The CToolsLib APIs are provided in source code, making it easy for engineers to embed them in their application for access during debug and in the field; if the need for field debugging arises. The APIs are written in C and include HTML documentation and data structures. The library suite includes functions such as advanced event triggering (AET), embedded trace buffer, system trace, DSP trace, ARM trace and instrumentation for specific KeyStone SoC features such as Multicore Navigator and TeraNet. CCStudio also provides utilities to import and analyze in-field debug and trace information with the Analyzer Suite.

CCStudio Eclipse ecosystem

CCStudio is part of the Eclipse IDE platform. This extends the debug and test options even further for the KeyStone software development team, with options to leverage other Eclipse elements or plug-ins to this robust system. CCStudio with KeyStone extensions provides a tremendous amount of system-level and SoC element level analysis, and offers flexibility both with and without external debuggers. Integration with Eclipse provides developers the flexibility to incorporate Eclipse functionality to supplement CCStudio or because Eclipse elements are more familiar to an individual developer.

Conclusion

The CCStudio tool suite with KeyStone extensions provides superior system-level visibility and analysis capabilities without requiring code changes. This is critical for developers seeking visibility to better understand multicore and multi-operating system-related issues. In the absence of such sophisticated visibility developers would have to rely on custom solutions and their own resources to understand complex multicore issues.This could add days or even weeks to problem solving. With TI's tool suite, non-intrusive debug and analysis capabilities with an adequate level of information and visibility is available in minutes resulting in shortened development time and better scheduling, as shown in Figure 4. As a result, customers will maintain the confidence in timely, high performance product delivery.

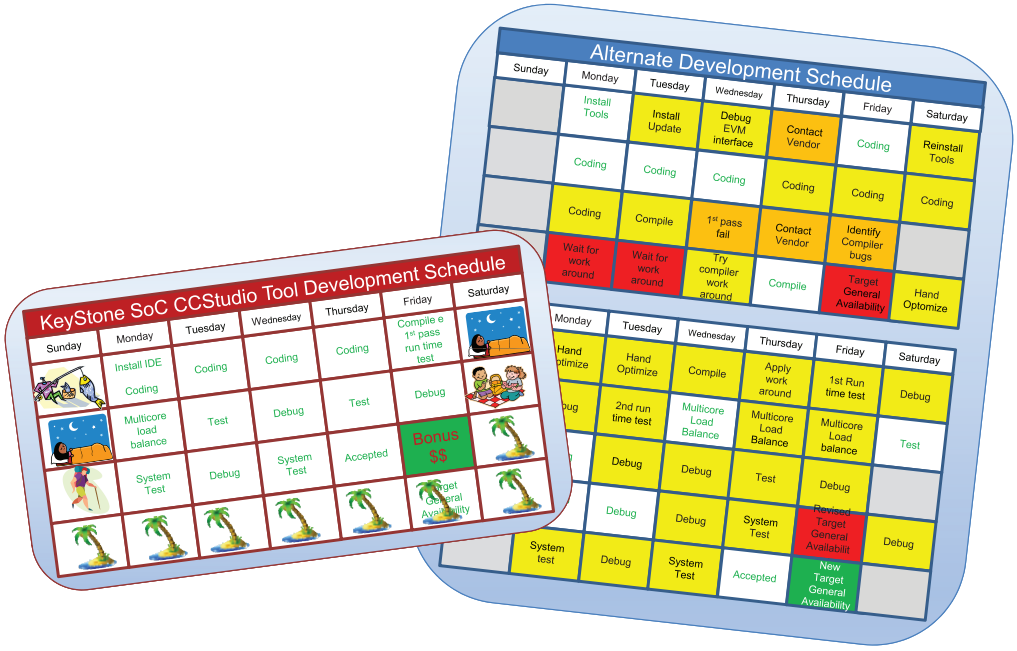

Figure 4. The CCStudio tool suite with KeyStone extensions provides the most efficient path to product delivery.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Audio	www.ti.com/audio	Communications and Telecom	www.ti.com/communications
Amplifiers	amplifier.ti.com	Computers and Peripherals	www.ti.com/computers
Data Converters	dataconverter.ti.com	Consumer Electronics	www.ti.com/consumer-apps
DLP® Products	www.dlp.com	Energy and Lighting	www.ti.com/energy
DSP	dsp.ti.com	Industrial	www.ti.com/industrial
Clocks and Timers	www.ti.com/clocks	Medical	www.ti.com/medical
Interface	interface.ti.com	Security	www.ti.com/security
Logic	logic.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Power Mgmt	power.ti.com	Transportation and Automotive	www.ti.com/automotive
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com	Wireless	www.ti.com/wireless-apps
RF/IF and ZigBee® Solutions	www.ti.com/lprf		
	TI E2E Community Home Page		e2e.ti.com

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265