**Introduction**

Multicore software development represents a significant investment that can make a big difference in a product’s ultimate success. Multicore System-on-Chip (SoC) complexity directly translates to software complexity and while multicore software architecture partitioning, task scheduling, dispatching and coordination among cores add another layer of programming challenges though they do not have to be overwhelming. How to accelerate multicore software development and still deliver a robust and top-quality system solution are the utmost concerns for multicore developers. This paper explores how Texas Instruments KeyStone multicore SoCs offload many software functions into hardware AccelerationPacs or other architectural elements to reduce the amount of software needed and to automate many of the more complex multicore management tasks. It also describes aspects of TI’s Multicore Software Development Kit (MCSDK), a free foundational software package used to jump start multicore application development on KeyStone devices.

**KeyStone multicore family**

Texas Instruments’ KeyStone SoCs are a family of multicore DSPs, multicore ARM®s and multicore DSPs plus ARMs together with application-specific hardware accelerators (AccelerationPacs). Some of the AccelerationPacs include radio AccelerationPac for 3GPP Layer 1 processing, packet and security AccelerationPacs as well as an on-chip Ethernet switch enabling exceptional power and area efficiency while offloading many computation-intensive tasks from the cores. KeyStone multicore SoCs have a unique multicore infrastructure that facilitates high-performance applications. Large on-chip memory combines with the KeyStone Multicore Shared Memory Controller (MSMC) to provide low-latency external memory access. The high-throughput TeraNet supports non-blocking data transfer between system end points, making KeyStone SoCs excellent for high-performance applications. Multicore Navigator with packet-aware Direct Memory Access (DMA) transfer engine and queue manager provides an excellent platform for high-level multicore programming with very efficient inter-core communication, multicore job scheduling and dispatching. Figure 1 offers a high-level view of the KeyStone multicore architecture.

*Figure 1: KeyStone multicore architecture*
The KeyStone Multicore Software Development Kit (MCSDK) is a package of key platform software that enables SoC features and demonstrates key SoC capabilities on a reference evaluation hardware platform. The MCSDK is designed to enable faster time to market to end-user products by providing a stable foundation of software components, easy-to-use APIs, built-in multicore programming methodologies and application demonstration software to showcase SoC key features and multicore software framework with full source code.

The MCSDK supports SYS/BIOS™, TI’s RTOS for C66x DSP cores and Linux™ SMP for ARM® cores. It also provides foundational software for low-level drivers, optimized DSP libraries for various applications and the Multicore System Analyzer for debug instrumentation enabling real-time event monitoring, real-time profiling, benchmarking and data visualization. The MCSDK also provides an industry-standard multicore programming model to ease the programming effort and to facilitate achieving multicore entitlement. Figure 2 shows the major components of the MCSDK.

Inter-core communication (IPC) is at heart of multicore programming. The KeyStone architecture enables both shared-memory IPC and Navigator-based IPC. Open Even Machine (OpenEM) is a runtime software component inside the MCSDK that leverages Navigator hardware for efficient IPC, job scheduling and rendering. Share memory IPC and OpenEM enable ease of implantation for industry-standard multicore programming models such as Open Multiprocessing (OpenMP) and Open Computing Language (OpenCL). Integrating OpenMP and OpenCL into the MCSDK help overcome the main multicore programming challenges to achieve the maximum performance with minimum effort in order to balance the task partitioning, program expression and enable scalability. Properly utilized, these tools can result in increased productivity and improved software and system performance.
OpenMP is an API that explicitly directs multi-threaded, shared memory parallelism in C, C++ and Fortran. OpenMP uses directives to specify sections of code to be executed in parallel, distributing work into multiple threads and coordinating the thread access to shared data. OpenMP can be applied to existing code to express parallelism and it is portable across many different platforms. The MCSDK includes an OpenMP runtime library for both DSP and ARM. Figure 3 shows the OpenMP implementation inside the MCSDK.

OpenMP can be used for many multicore applications to benefit big data parallel processing. Running OpenMP on the MCSDK from multicore DSPs demonstrates nearly linear speedup in performance scaling relative to the number of cores. Figure 4 shows multicore speedup on large FFT and image processing running on one, two, four or eight DSP cores.

![Figure 3: OpenMP implementation in TI’s MCSDK](image)

![Figure 4: OpenMP multicore speedup with MCSDK demo](image)
OpenCL facilitates open, industry-standard parallel programming across heterogeneous processors including CPU, DSP, GPU and accelerators. OpenCL is designed to be portable using low-level abstractions for highly parallel task execution, which is a perfect fit for High Performance Computing (HPC), image, video and graphics processing, as well as many other embedded applications.

OpenCL is a parallel programming framework that consists of four different models: platform, memory, execution and programming. The platform model includes one host connecting to one or more OpenCL devices; each device can have one or more compute units which includes processing elements for parallel processing. The OpenCL execution model includes a host program running on a host CPU and kernels which are application algorithms running on one or more devices. OpenCL host and device memory models are independent from each other. Kernel work items can have access to four distinct memory regions: global memory, constant memory, local memory and private memory. And, OpenCL can support a data parallel-programming model or task parallel-programming model.

OpenCL uses C99 C language with modifications to enable program parallelism and vector processing to express kernels running on the OpenCL devices. The host creates a data structure called a command-queue which contains kernels to be dispatched and executed on the OpenCL devices. The command-queue can be scheduled either in-order or out-of-order through synchronization commands.

OpenCL is an excellent programming model for cloud computing, HPC and many other applications on KeyStone multicore devices to accelerate computation-intensive tasks. By leveraging the MCSDK, OpenCL device drivers and platform software, an OpenCL device library can be built for the host to enqueue and dispatch parallel kernel functions into DSP (and ARM) pools to increase the computation performance. An example of a Mandelbrot set zooming imaging application can be demonstrated using a host PC connected through PCIe to an Advantech DSPC 8681 card populated with four TMS320C6678 multicore SoCs each equipped with eight C66x DSP cores. The results show significantly faster processing compared to running the kernel on X86 based multicore CPUs.

Figure 5 shows the demo test setup.

---

**Figure 5: OpenCL HPC Mandelbrot demo with Advantech DSPC 8681**
OpenCL host can also run on a device with ARM® cores, such as TI’s TCI6614 SoC which has one ARM Cortex™-A8 core and four C66x DSP cores or TI's TCI6636 SoC which has four ARM Cortex-A15 cores and eight C66x DSP cores. The OpenCL host on an ARM core queues the command in the host, schedules and dispatches the kernel task through OpenEM using multicore Navigator into the DSP cores. Figure 6 shows an OpenCL programming model on KeyStone SoCs. Running OpenCL multicore applications on KeyStone multicore DSPs delivers much higher energy efficiency than alternative solutions.

**Figure 6: OpenCL example on KeyStone SoC**

**LTE small-cell transport solution based on the MCSDK**

TI’s KeyStone architecture enables carrier-grade scalable multi-standard wireless base station solutions. It features a Network coprocessor and Ethernet switch that offloads the majority of Ethernet packet and security processing from the software to enable low-latency network solutions. The MCSDK includes low-overhead Linux™ and a transport library enabled for the Network Coprocessor to deliver best-performance fast-path transport for wireless base stations.

The transport library component of the MCSDK provides a set of Service Access Point Libraries designed to meet various application requirements such as enhanced fast path for small-cell base stations as well as general data path and flow management. It also enables embedded server and router applications. Low-latency transport solutions are achieved by running fast path with the MCSDK taking advantage of the Network Coprocessor and Ethernet switch in the KeyStone architecture.

The example software implementation leveraging the MCSDK and transport library can be demonstrated on LTE small-cell base stations built on the TCI6614 KeyStone SoC. The TCI6614 SoC integrates four C66x...
DSP cores with an ARM® Cortex™-A8 core. The hardware infrastructure includes a Network Coprocessor for network and security acceleration, Multicore Navigator with dedicated queue manager and packet DMA subsystem. LTE’s baseband physical layer processing (PHY) is running on two DSP cores while most standard computation-intensive algorithms are offloaded into Layer 1 accelerators to improve area and power efficiency. The LTE user-plane processing for Layer 2 runs on the remaining two DSP cores. The ARM core runs the LTE control-plane processing, including Layer 3, transport, Radio Resource Management and Operations and Maintenance as well as other LTE-supporting applications. All of the LTE application software runs on top of the MCSDK which provides low-level foundational and runtime software to simplify application layer development. Maximum LTE packet throughput can be achieved in KeyStone-based solutions by leveraging the enhanced fast-path transport library in the MCSDK. Figure 7 below shows an LTE small-cell implementation on top of the MCSDK on a TCI6614 SoC.

![Figure 7: MCSDK Transport lib provides fast path for LTE small-cell implementation](image)
**Conclusion**

TI's MCSDK enables rapid deployment of many real-life multicore applications such as communications infrastructure, cloud computing, high-performance computing, oil and gas exploration, video infrastructure, embedded analytics, image, video and graphics processing and many more. The MCSDK is a free software package for KeyStone multicore SoCs providing a solid foundation for multicore software development. It not only provides a user-friendly API for streamlined application development, it also has built-in runtime libraries to enable industry-standard OpenMP and OpenCL multicore programming. The MCSDK can significantly reduce multicore development efforts and enable developers to focus on features that offer differentiation to their products. In addition, the transport library of the MCSDK is optimized for the KeyStone architecture to enable low-latency carrier-grade transport solutions.

**Acknowledgments**

The authors would like to acknowledge and thank the following individuals for their contributions to the white paper: Naga Chandrashekar, Arnon Friedmann, Andy Fritsch, Frank Fruth, Debbie Greenstreet, Dave Halliday, Doug Kay, Dave Lide, Pekka Varis and Alan Ward.
### IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

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

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

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license 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 significant portions 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. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

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

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI's goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have not been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

<table>
<thead>
<tr>
<th>Products</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audio</td>
<td><a href="http://www.ti.com/audio">www.ti.com/audio</a></td>
</tr>
<tr>
<td>Amplifiers</td>
<td>amplifier.ti.com</td>
</tr>
<tr>
<td>DSP</td>
<td>dsp.ti.com</td>
</tr>
<tr>
<td>Interface</td>
<td>interface.ti.com</td>
</tr>
<tr>
<td>Logic</td>
<td>logic.ti.com</td>
</tr>
<tr>
<td>Power Mgmt</td>
<td>power.ti.com</td>
</tr>
<tr>
<td>Microcontrollers</td>
<td>microcontroller.ti.com</td>
</tr>
<tr>
<td>RFID</td>
<td><a href="http://www.ti-rfid.com">www.ti-rfid.com</a></td>
</tr>
<tr>
<td>OMAP Applications Processors</td>
<td><a href="http://www.ti.com/omap">www.ti.com/omap</a></td>
</tr>
<tr>
<td>Wireless Connectivity</td>
<td><a href="http://www.ti.com/wirelessconnectivity">www.ti.com/wirelessconnectivity</a></td>
</tr>
</tbody>
</table>

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2013, Texas Instruments Incorporated