WHITE PAPER

Uday Gurnani,
Applications Engineer,
Embedded Processors
Texas Instruments

Texas Instruments

Medical Software Development on KeyStone™ Processors

*The role of embedded software developers has become increasingly important in the innovation of medical technology. Embedded processors used in many medical systems these days need to handle high levels of computational complexity in real-time and at low power. For example, innovative digital signal processors (DSPs) from Texas Instruments (TI) have enabled diagnostic systems that achieve higher resolution image scans and provide caregivers access to key clinical data in real time. This ability to meet the processing constraints of medical systems is not only a function of the inherent parallelism in TI's KeyStone architecture but also a function of how well software developers are able to leverage its advanced architecture: multiple cores, rapid data paths, high-speed peripherals and other features. To that end, TI has created a solid software development ecosystem to enable the quick and easy development of optimized medical applications that fully utilize TI's high-performance **KeyStone-based system-on-chip (SoC)**.*

Getting started

A typical development environment consists of a host computer with TI tools installed on it and an Eclipse-based **Code Composer Studio™ (CCS) IDE**. CCS IDE has been designed as a one-stop shop that allows code editing, simulation and running/debugging on **evaluation modules (EVMs)**, all within a single environment. The host computer and EVM are connected via an emulator that can provide advanced trace and debug capabilities. Instrumentation, profiling and visualization tools are useful features within CCS IDE that can be leveraged by developers to fine-tune their applications. TI's free **Multicore Software Development Kit (MCSDK)** incorporates the core software building blocks, including platform software, a real-time operating system **SYS/BIOS™**, the Linux operating system for ARM®, processors low-level drivers, high-level APIs and algorithm libraries, all in one package. Out-of-box demonstration applications and examples are part of the MCSDK and are a great starting point for developers. These enable developers to learn how to leverage APIs within the MCSDK to create an application that runs on multiple cores and takes advantage of the various hardware features within KeyStone-based processors.

Development tools

With the foundational software building blocks available with the MCSDK, developers can focus their attention on algorithm innovation. Often addressing new clinical needs translates into the research and implementation of novel algorithms that differentiate a medical product and create new intellectual property. A programmable processor, like a DSP, provides innovators the flexibility of trying various approaches in software before deciding on what works best. To give developers a kick start, TI has released multiple algorithm libraries like DSPLIB, MATHLIB and IMGLIB that include commonly-used signal-processing, math and image-processing functions optimized for TI's DSPs. Codecs for many video, audio and speech applications that have been optimized for various TI platforms can be accessed at the **TI codec page**. For medical imaging developers, there is the **Software Toolkit for Medical Imaging (STK-MED)**, which includes optimized building blocks for ultrasound and optical coherence tomography applications. These include functions for B-Mode processing, Doppler signal processing, scan conversion, cubic spline interpolation, FFT and IFFT, and magnitude and log computation. Since the algorithm source code is available as part of STK-MED, developers are able to easily extend these functions to create customized modules to differentiate their medical systems.

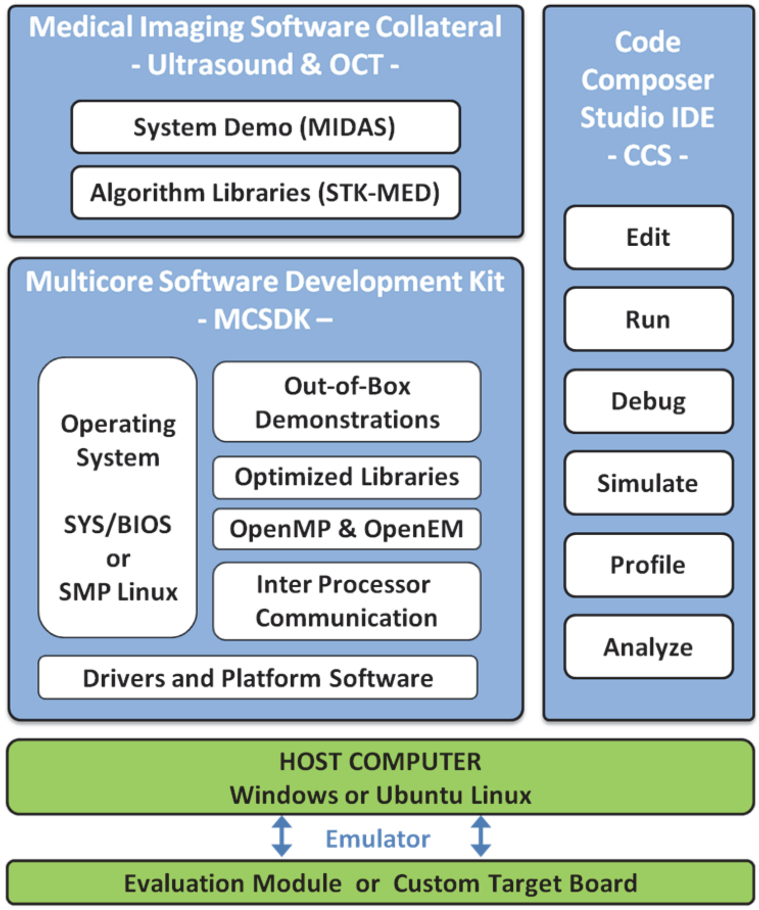

Figure 1: The block diagram summarizes the various software components that create a solid development environment for medical developers who choose TI's KeyStone-based processors for their system designs.

Testing and validating

Software developers also play a critical role in collecting validation data for the 510(k) or PMA submission to the Food and Drug Administration (FDA) for device clearance or approval. In medical-imaging systems, for example, developers may need to prove the accuracy and integrity of each signal-processing block, showing that the device's final image output is accurate and bit-exact to the expected output. TI's STK-MED incorporates test projects and reference test vectors for the included algorithms. These projects showcase an example for developers using TI's CCS environment and integrated software tools to design their own validation tests for output accuracy and performance profiling.

Debug and trace tools can also be very useful in the testing and validation phase, and throughout the software development life cycle. For TI's KeyStone-based processors, the **CToolsLib** provides the ability to do core trace and system trace. Core trace provides program counter and instruction timing and the ability to monitor core memory accesses and core events for the DSP. System trace provides multicore application instrumentation, including core visibility and processing relative to the entire system, as well as correlation between traces. The fault management library enables applications to store DSP register data upon any fault detection. The **Multicore System Analyzer**, which is integrated into CCS IDE, can also be used as a real-time tool to analyze, visualize and profile application performance and behavior.

System-level implementation

Once algorithms have been implemented, the next step is to integrate them together in a system-level implementation that maps the complete signal chain on the target embedded device. Resource management, inter-processor communication and data movement techniques are some of the important considerations during this stage. TI provides software components for its KeyStone-based SoCs that can be used to achieve these goals. Some of these include the **resource management low level drivers** (LLDs), inter-processor communication (IPC) package and enhanced direct memory access LLD, which are part of the MCSDK, as well as the framework components library with its Codec Engine interface that allows algorithms to easily plug-and-play together. Developers might also want to leverage high-speed interfaces like PCI Express, HyperLink, Gigabit Ethernet or Serial RapidIO® (SRIO) to move data across devices. They can also leverage the Multicore Navigator to transport data between dedicated memories within a device. Again the MCSDK incorporates relevant LLDs and usage examples for each of these peripherals.

To appreciate the scalability and ease-of-use of these software components, let's look at the IPC software package as an example. IPC APIs enable communication between threads that either run on the same core, on different cores or on different devices. There are various possibilities with IPC transports: for communication between cores that reside on the same device there's either shared memory transport or navigator transport. For communication between cores across two devices one might want to use the SRIO interface. What's great about IPC from the developer's perspective is the ability to use the same set of APIs irrespective of the transport underneath. In a heterogeneous system like TI's **KeyStone II family**, with ARM on one side and DSP on the other side, IPC APIs can be used on the DSP side, and corresponding APIs can be used on the ARM side.

How developers leverage each core and map functions within a multicore embedded device is another aspect of system-level implementation. Two typical models are functional parallelization and data parallelization. To explain the difference, let's use ultrasound and optical coherence tomography as examples. For an ultrasound system's mid-end processing, one might choose to implement B-mode processing on one DSP core, Doppler processing on another DSP core, scan conversion on a third DSP core and provide display and user interface functionality via an ARM core. This corresponds to a functional parallelization strategy. In the case of optical coherence tomography, each frame of input data could be sliced and divided across multiple DSP cores and processed through the same set of algorithms: background subtraction, resampling, FFT, magnitude computation and log compression running on each core. This is an example of data parallelization.

TI's **Medical Imaging Demo Application Starter (MIDAS)** software package implements the ultrasound and optical coherence tomography signal chain described above on TI's KeyStone-based C6678 processor. MIDAS leverages the algorithm modules from STK-MED and uses the MCSDK and other software components to realize a system-level implementation that meets real-time constraints. This serves as a great reference for developers to see an example of how medical algorithms can be integrated together to realize a complete signal processing flow on TI's embedded processors. (See Figure 2.)

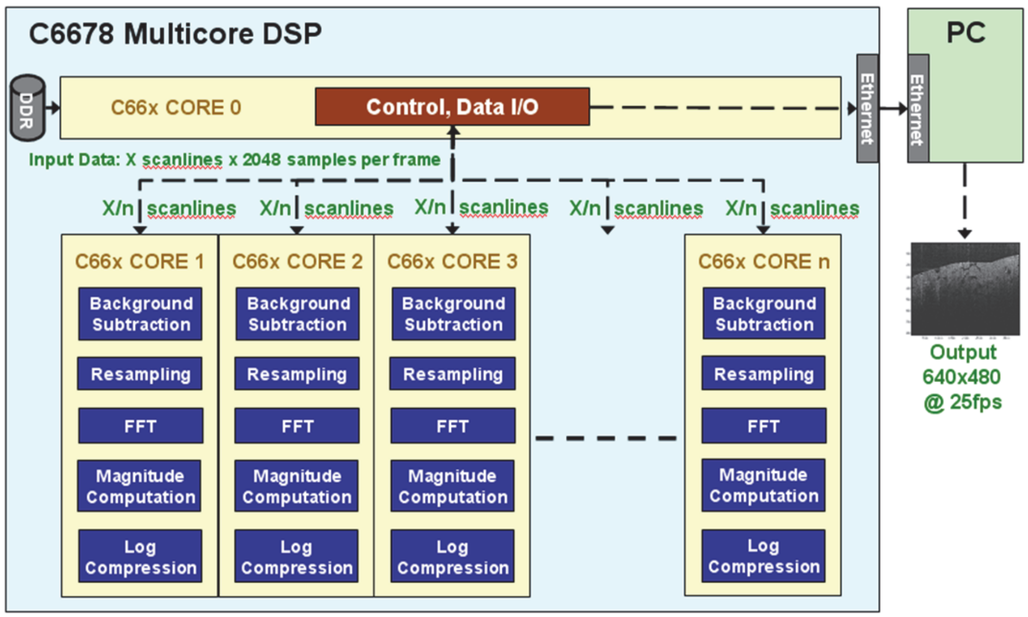

Figure 2: TI's MIDAS OCT v1.0 showcases a system-wide demo implementation of the mid-end and back-end signal chain for optical coherence tomography, and leverages TI's KeyStone-based C6678 multicore DSP. This implementation is an example of data-based partitioning across cores.

Open-source tools

Software developers also have the option of leveraging some open-source parallel programming APIs on TI's KeyStone-based processors, with **OpenMP** and OpenEM software packages included in TI's MCSDK as well as OpenCL support in the works. OpenMP includes compiler directives and library routines that make it possible to quite easily and incrementally parallelize an application across multiple cores. OpenMP implementations are based on a fork-join model as shown in Figure 3. An OpenMP program begins with an initial thread (known as a master thread) in a sequential region. When a parallel region is encountered—indicated by the compiler directive "`#pragma omp parallel`"—extra threads called worker threads are automatically created by the scheduler. This team of threads executes simultaneously to work on the block of parallel code. When the parallel region ends, the program waits for all threads to terminate, then resumes its single-threaded execution for the next sequential region. OpenEM is a centralized runtime system that dynamically schedules events across all resources of the system in an optimized way. OpenEM is event driven and is also easily scalable from one to many cores.

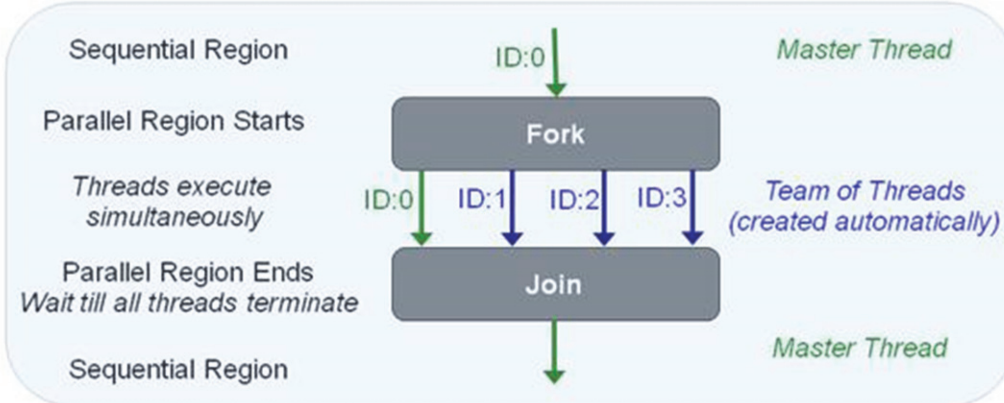

Figure 3: OpenMP's fork-join model makes it possible for developers to incrementally parallelize their applications with ease

Additional support

In addition to the host of software packages available to medical software developers, the TI **E2E™ Community** and third parties form an important part of the TI embedded ecosystem. The E2E Community allows system developers to directly interact with TI engineers on their technical questions. TI also works closely with third parties in **TI's Design Network** to bring their software tools into the development ecosystem that provide additional multicore programming, profiling and analysis capabilities. To name a few, Poly-Platform from PolyCore consists of tools and runtime software that provide a programming model for applications to scale from one to many cores. Prism from CriticalBlue provides multicore analysis and exploration, and allows developers to evaluate various parallelization strategies.

Learning collateral

Extensive learning collateral in the form of user guides, white papers, application notes, online training, as well as **reference diagrams for specific medical applications** can also serve as in-depth resources for developers. The **TI multicore web page** provides links to much of this collateral. The **Embedded Processors (EP) Wiki** is another great resource with technical articles, guides, tips and tricks, written by TI employees and community members, with KeyStone SoC-specific articles summarized at the **Multicore Wiki page**. TI conducts **training seminars** around the world on KeyStone SoCs, programming and other relevant topics, with many **training videos** available to watch online anytime. TI also offers many analog components, microcontrollers and other processors for medical technology, which can be found in TI's **HealthTech product guide**.

Summary

TI's KeyStone-based processors represent the cutting edge in hardware and the very best in multicore processing at low power. This hardware offering is coupled with an industry-leading, stable software development ecosystem that supports a comprehensive multicore framework, which ensures that medical system designers spend most of their time innovating and differentiating and less time in addressing system-level issues. As we continue to innovate and create new hardware, we also continue to innovate with our software, keeping in mind a modular software approach so that our customers can simply migrate from one generation of SoCs to the next. With this integrated ecosystem, our objective is to help our fellow engineers in the medical technology world create great applications that get to market fast and effectively solve the many clinical needs of today.

Disclaimer

System and equipment manufacturers and designers are responsible to ensure that their systems (and any TI devices incorporated in their systems) meet all applicable safety, regulatory and system-level performance requirements. All application-related information in this white paper (including application descriptions, suggested TI devices and other materials) is provided for reference only. This information is subject to customer confirmation, and TI disclaims all liability for system designs and for any applications assistance provided by TI. Use of TI devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless TI from any and all damages, claims, suits or expense resulting from such use.