WHITE PAPER

Vivek Chengalvala,
Technical Lead,
Video and High-Performance Computing

Sam Nelson Siluvaimani,
Technical Lead,
Multicore Software Development Kit

Texas Instruments

Texas Instruments

Accelerating high-performance computing development with Desktop Linux™ Software Development Kit

Over the years, applications have become more complex and demanding. Many applications that were once restricted to research are being widely adopted across industries as the processing capabilities of processors rise and the solutions become practical. Financial analytics, medical imaging, gaming, cloud computing, multimedia and security are examples of such fields where applications typically require a large number of mathematical operations to be performed quickly. Many of these applications also have constraints on latency; for the system to work, the operation must be completed within some fixed time. Digital Signal Processors (DSPs) are well suited for such applications providing a lower-cost solution, with better performance, lower latency, flexibility, programmability and lower power requirements compared with other general-purpose processors or hardware-based accelerators. There are physical restrictions to the processing power of a single-core processor or the number of cores that can be put into a single device, thus paving a way for the applications to embrace multi-processor environments.

Texas Instruments offers PCIe-based acceleration cards that can host several TMS320C66x DSP processors. Based on the application requirements, PCIe cards with one, four or eight DSPs can be used. For more demanding applications, several of these cards can be plugged into standard PCs with free PCIe slots.

For faster time to market, Texas Instruments offers a Desktop Linux™ Software Development Kit (SDK) that enables the offload of computation-intensive processing to the DSPs, thus accelerating the application development. Desktop Linux SDK is a user-mode software package developed to work with Ubuntu Linux. The out-of-box demo application demonstrates how to map host memory to DSPs, transfer data between DSP and host memory and dispatch tasks to all the cores/processors on the acceleration card.

Multiple software components that comprise the Desktop Linux SDK are shown in Figure 1. These software components manage the DSP memory resources, download code to the DSPs and enable control path communication between the DSPs and host processor. The Desktop Linux SDK also offers memory mapping between the processors for high-speed data exchange. All combinations of memory mapping are feasible and system architects can decide the best way that suits the application. Examples of memory mapping include:

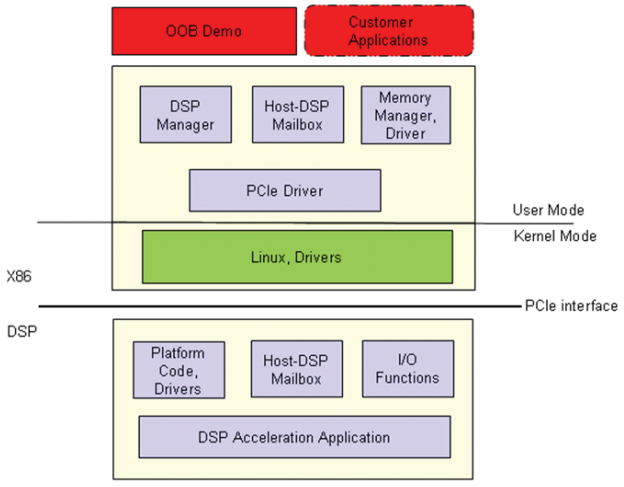

Figure 1: Desktop Linux SDK Architecture

- Mapping a section of host memory to be visible concurrently to all of the DSPs
- Mapping so that each DSP can map a separate section of host memory
- Mapping so that memory of one DSP is visible directly to another DSP without host memory involvement

Modules included in the Host Processor Package are buffer manager, mailboxes, download manager, contiguous memory driver, PCIe user space driver and demos. The DSP package includes mailboxes, platform code/drivers, Code Composer Studio™ IDE Project to compile the DSP image and demo code that runs on the DSP.

Data I/O between host and DSP

As functionality is offloaded to the DSP, the host processor has to send and receive data to the DSPs for processing. This data exchange between host processor and DSP can be achieved in two ways.

1. Data is copied by the host processor into the DSP's DDR before the DSP starts processing
2. DSP picks up the data from the host memory buffer directly as the DSP consumes the data (Host memory is made visible to the DSP for the life time of processing of that data)

PCIe driver provided in the SDK allows both these mechanisms to do data exchange. In addition, a certain memory region on the host memory can be reserved to be mapped permanently to be visible in all the DSP chips/cores. Such mapped memory could be used as "global shared" memory. It is implied that any accesses to the shared memory have to be done by properly acquiring the lock prior to access and release the lock after access.

Using the DSP's EDMA to do the data copy over the PCIe interface gives much higher throughput compared to using CPU read/writes. When the host processor wants to send/receive data buffer to/from the DSP, the X86 programs the DSP's EDMA engine to initiate the data I/O. In order for the EDMA to work, the source and destination buffers have to be contiguous in physical memory. The DSP's memory (DDR) is anyways contiguous. On the host, it is recommended to allocate physically contiguous buffers (using the CMEM module supplied with the SDK) instead of using malloc.

Buffer manager

A generic malloc or free causes memory fragmentation. In order to circumvent this problem, buffer manager (bufMgr) is created. bufMgr divides memory into equal-sized chunks – so memory can be allocated and freed without any fragmentation. Multiple pools (all chunks in a pool are of the same size) are needed to emulate malloc/free. Each pool is tracked with a pool handle created by the create API and needs to be supplied to all bufMgr APIs (delete, alloc, free) to do the operation on the appropriate buffer pool. Number

of pools, size of a chunk in each pool, and number of chunks in the pool are configurable by the application. bufMgr can manage any type of memory (X86, DSP, PCIe, etc.). A typical application has multiple pools. Another feature of bufMgr worth noting is that Single alloc, Multiple free supported (when there are multiple consumers). A buffer is really freed when all the consumers call free(). The number of consumers is supplied during allocation of a buffer from the bufMgr. There are two ways to create a buffer pool.

1. Create a buffer pool from a discrete array of buffers
2. Create a buffer pool from contiguous memory (the buffer manager internally chunks up that large memory based on supplied chunk size)

Once a pool is created, there is no distinction with respect to how the pool is created. Memory allocation for the chunks is done outside of bufMgr. Typical use case is the bufMgr pool is created on DSP DDR memory or PCIe memory. The host can allocate a buffer from this pool, fill up the buffer with the data for DSP processing and queue it into the DSP for processing. Once the DSP finishes processing, the buffer can be recycled.

Mailbox

Mailbox is used for exchanging control messages between the host and individual DSP cores. A mailbox is uni-directional, either host —> DSP or DSP —> host. Mailboxes operate using memory areas in DSP. The memory areas used by the Mailbox and the max payload size are configured during the creation of the mailbox from both host and DSP. The mailboxes are tracked using local handle returned by the create API. An empty Mailbox slot must be allocated prior to sending a message. Receiving a message frees a slot and marks it as being empty. Mailboxes can be queried to obtain the number of unread messages within the mailbox.

Download manager

Download Manager provides APIs to download and reset DSPs. On boot up, the DSP is out of reset and in Idle loop, waiting for the entry point to be set. Download Manager API downloads the DSP code and writes the DSP entry point. In general, prior to downloading of application code, the DSP DDR memory needs to be initialized. This is achieved by downloading a small DSP image (init.hex). Once the init code finishes the DDR, the DSP waits for the next entry point. At this time, the application program must be loaded through the Download Manager API. The boot process also supports boot configuration data to be written at a specified location along with the download.

Contiguous memory driver

Memcpy-based PCIe transfers are very inefficient for throughput. Hence, it is advisable to use DMA to do data transfer between the DSP and host processor. A prerequisite for the DMA to function is that the memory needs to be contiguous in physical memory. Memory allocated using malloc is not contiguous. Ubuntu Linux 12.04 doesn't have any user-mode APIs to allocate contiguous physical memory, and hence, the need for a kernel mode driver to allocate contiguous physical memory is necessary. This CMEM module basically accomplishes that. There is a kernel module inserted to allocate/free physically contiguous memory. The CMEM driver is a user-mode module with the APIs that the application can invoke.

In the kernel space, dma_coherent_alloc is invoked to allocate contiguous memory. Alloc uses mmap to map the physical memory to user space address. The application can fill data into the buffers (network or disk read) using user-mode address. The DMA can be initiated from the same buffer using the corresponding physical address. As there is no garbage collection in the driver, calling alloc/free with different sizes will eventually result in memory fragmentation. It is hence recommended to allocate all contiguous buffers up-front at the start of the process and use the buffer Manager to manage the buffers (bufMgr is a fragmentation-free memory management system. Refer to the bufMgr chapter for more details). At the end of the process, it frees buffers to the Kernel module and unloads the module.

There is a restriction from Linux that we cannot allocate more than 4 MB of physical contiguous memory using dma_coherent_alloc. If the application needs a buffer > 4MB, multiple CMEM buffers need to be allocated and grouped together in a descriptor. These discontinuous 4MB buffers can be mapped to be visible as contiguous buffers in the DSP address space.

As an alternative, the CMEM module can be configured to use reserved memory at boot time in Linux. Scripts are provided to configure the amount of memory to be reserved at boot time and the driver uses the reserved memory to allocate the memory requested by the application instead of dma_coherent_alloc. This method only works if the amount of physical memory present in the desktop is more than 4 GB plus the required reserved memory needed. With this scheme, the driver allows allocation of buffers bigger than 4 MB.

User-space PCIe driver

The user space PCIe driver uses the pciaccess library to gain access to the TI PCIE devices. On driver open, the driver searches and finds the TI 667x PCI devices in the system, probes the devices to activate and enable communication with DSP using the PCIe BAR regions by mapping them to user space. Once open, the following functions are available through the PCIE user space driver.

- Support read and write to DSP memory
- Support transfer of data to DSP memory through DSP DMA
- Support mapping of Host contiguous memory to DSP memory
- Access to DSP boot configuration register to configure entry point

There are several examples and demos that Texas Instruments offers to demonstrate the ease of programming and versatility of Desktop Linux SDK foundation software. A separate SDK (called Multicore Software Development Kit for Video, in short – MCSDK-Video) is available from Texas Instruments as well. MCSDK-Video is built on top of the Desktop Linux SDK demonstrating video processing offload.

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.

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

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