MCSDK HPC 3.x Examples

HPC (High Performance Computing) Development Tools for MCSDK

Version 3.0

Examples Guide

Last updated: 05/22/2015

Contents

Introduction

This page describes all of the examples provided with the MCSDK HPC release. Once installed, all examples will be located inside the */usr/share/ti/examples* directory.

OpenCL Examples

The OpenCL examples will be installed in the directory */usr/share/ti/examples/opencl*.

Sample Application	**Details**
platforms	This example uses the OpenCL C++ bindings to discover key platform and device information from the OpenCL implementation and print it to the screen.
simple	This is a 'hello world' type of example that illustrates the minimum steps needed to dispatch a kernel to a DSP device and read a buffer of data back.
mandelbrot/mandelbrot_native	The 'mandelbrot' example is an OpenCL demo that uses OpenCL to generate the pixels of a Mandelbrot set image. This example also uses the C++ OpenCL binding. The OpenCL kernels are repeatedly called generating images that are zoomed in from the previous image. This repeats until the zoom factor reaches 1E15. This example illustrates several key OpenCL features: ▪ OpenCL queues tied to potentially multiple DSP devices and a dispatch structure that allows the DSPs to cooperatively generate pixel data, ▪ The event wait feature of OpenCL, ▪ The division of one time setup of OpenCL to the repetitive en-queuing of kernels, and ▪ The ease in which kernels can be shifted from one device type to another. The 'mandelbrot_native' example is non-OpenCL native implementation (no dispatch to the DSPs) that can be used for comparison purposes. It uses OpenMP for dispatch to each ARM core. Note: The display of the resulting Mandelbrot images is currently disabled when run on the default EVM Linux file system included in the MCSDK. Instead it will output frame information.
ccode	This example illustrates the TI extension to OpenCL that allows OpenCL C code to call standard C code that has been compiled off-line into an object file or static library. This mechanism can be used to allow optimized C or C callable assembly routines to be called from OpenCL C code. It can also be used to essentially dispatch a standard C function, by wrapping it with an OpenCL C wrapper. Calling C++ routines from OpenCL C is not yet supported. You should also ensure that the standard C function and the call tree resulting from the standard C function do not allocate device memory, change the cache structure, or use any resources already being used by the OpenCL runtime.
matmpy	This example performs a 1K x 1K matrix multiply using both OpenCL and a native ARM OpenMP implementation (GCC libgomp). The output is the execution time for each approach (

	OpenCL dispatch to the DSP vs. OpenMP dispatching to the 4 ARM A15s).
offline	This example performs a vector addition by pre-compiling an OpenCL kernel into a device executable file. The OpenCL program reads the file containing the pre-compiled kernel in and uses it directly. If you use offline compilation to generate a .out file containing the OpenCL C program and you subsequently move the executable, you will either need to move he .out as well or the OpenCL application will need to specificy a non relative path to the .out file.
vecadd	The same functionality as the vecadd_openmp example, but expressed fully as an OpenCL application without OpenMP. Included for comparison purposes.
vecadd_mpax	The same functionality as the vecadd example, but with extended buffers. The example iteratively traverses smaller chunks (sub-buffers) of large buffers. During each iteration, the smaller chunks are mapped/unmapped for read/write. The sub-buffers are then passed to the kernels for processing. This example could also be converted to use a pipelined scheme where different iterations of CPU computation and device computation are overlapped. NOTE: The size of the buffers in the example (determined by the variable 'NumElements') is dependent on the available CMEM block size. Currently this example is configured to use buffers sizes for memory configurations that can support 1.5 GB total buffer size. The example can be modified to use more (or less) based on the platform memory configuration.
dsplib_fft	An example to compute FFT's using a routine from the dsplib library. This illustrates Calling a standard C library function from an OpenCL kernel.
ooo, ooo_map	This Application illustrates several features of OpenCL. ▪ Using a combination of In-Order and Out-Of-Order queues ▪ Using native kernels on the CPU ▪ Using events to manage dependencies among the tasks to be executed. A JPEG in this directory illustrates the dependence graph being enforced in the application using OpenCL events. The ooo_map version additionally illustrates the use of OpenCL map and unmap operations for accessing shared memory between a host and a device. The Map/Unmap protocol can be used instead of read/write protocol on shared memory platforms.
null	This application is intended to report the time overhead that OpenCL requires to submit and dispatch a kernel. A null(empty) kernel is created and dispatched so that the OpenCL profiling times queried from the OpenCL events reflects only the OpenCL overhead necessary to submit and execute the kernel on the device. This overhead is for the roundtrip for a single kernel dispath. In practice, when multiple tasks are being enqueued, this overhead is pipelined with execution and can approach zero.
edmamgr	This application illustrates how to use the edmamgr api to asynchronously move data around the DSP memory hierarchy from OpenCL C kernels. The edmamgr.h header file in this directory enumerates the APIs available from the edmamgr package
sgemm	This example illustrates how to efficiently offload the CBLAS SGEMM routine (single precision matrix multiply) to the DSPs using OpenCL. The results obtained on the DSP are compared against a cbas_sgemm call on the ARM. The example reports performance in GFlops for both DSP and ARM variants.

OpenCL + OpenMP Examples

The OpenCL + OpenMP examples will be installed in the directory */usr/share/ti/examples/opencl+openmp*.

Sample Application	Details
vecadd_openmp	This is an OpenCL + OpenMP example. OpenCL program is running on the host, managing data transfers, and dispatching an OpenCL wrapper kernel to the device. The OpenCL wrapper kernel will use the ccode mode (see ccode example) to call the C function that has been compiled with OpenMP options (omp). To facilitate OpenMP mode, the OpenCL wrapper kernel needs to be dispatched as an OpenCL Task to an In-Order OpenCL Queue.
vecadd_openmp_t	This is another OpenCL + OpenMP example, similar to vecadd_openmp. The main difference w.r.t vecadd_openmp is that this example uses OpenMP tasks within the OpenMP parallel region to distribute computation across the DSP cores.
vecadd_mpax_openmp	Similar to vecadd_mpax example, but used OpenMP to perform the parallelization and the computation. This example also illustrates that printf() could be used in OpenMP C code for debugging purpose.
dgemm	This example illustrates how to efficiently offload the CBLAS DGEMM routine (double precision matrix multiply) to the DSPs using OpenCL. The results obtained on the DSP are compared against a cbas_dgemm call on the ARM. The example reports performance in GFlops for both DSP and ARM variants.

OpenMP Accelerator Model Examples

The OpenMPAcc examples are installed in the directory */usr/share/ti/examples/openmpacc*.

Example	Description

printf_debug	Demonstrates the use of printf() within an OpenMP target region for debugging. This example also demonstrates the use of OpenMP runtime library functions.
dotprod	Performs vector dot product using C66x intrinsic functions within target region. This example is written entirely in C. It also depends on the c_intrinsics_host_port package. Instructions are included in the Makefile.
dsplib_fft	Demonstrates the use of C66x libraries with the OpenMP Accelerator Model. It has a target region which calls a DSPLIB library function.
edmamgr	Demonstrates the use of EDMA Manager library functions to do accelerated memory transfers within an OpenMP target region. EDMA Manager is used to perform 1D and 2D memory transfers using EDMA3 co-processors.
vecadd	Small vector addition using a single OpenMP parallel region offloaded to the DSP using #pragma omp target.
vecadd_t	Small vector addition using OpenMP tasks within an OpenMP parallel region offloaded to the DSP using #pragma omp target.
vecadd_big	Large Vector addition using *__malloc_ddr* extension to pre-allocate memory in shared ARM-DSP CMEM address space in order to eliminate data copy overheads during OpenMP target region offload.
vecadd_complex	Similar to vecadd, but elements of the vectors are of complex type.
local	Demonstrates the use of the *local* map type. This example is similar to vecadd, but uses an intermediate local buffer (in L2 SRAM) to accumulate the sum of the input vectors.
sub_section	Demonstrates the use of array sections. This example is similar to vecadd, but the vector addition is achieved in two steps with each step accumulating the sum of subsections of the vectors.
null	Measures time overhead associated with offloading a single target region from ARM to DSP.
target_implicit_map	Demonstrates implicit mapping of array sections when offloading target regions within a '#pragma omp target data' region.
target_orphan_call	Demonstrates placing a target region within a function and calling this function from within a '#pragma omp target data' region.
target_update	Demonstrates how to use the '#pragma omp target update' clause to synchronize host and device array sections within a '#pragma omp target data' region.
dotprod_fileread	Demonstrates how target regions may be used within a parallel region. Four OpenMP threads on the ARM concurrently read the input arrays. Then each thread offloads the computation of the dot product of its input to the DSPs, one at a time, within a critical region.
vecadd_lib	Demonstrates how to build a CLACC library. In the Makefile, the CLACC option --make_lib is used to generate CLACC libraries libvecadd.a and libvecadd_t.a.
vecadd_mpi	Demonstrates the use of offloaded target regions along with MPI calls using the TI OpenMPI library to perform vector addition across multiple nodes.
edmabw	Measures the average access times between different memory regions (DDR, MSMC, L2 SRAM) for a DSP core using EDMA manager.
openmpbench_C_v3	EPCC v3 microbenchmarks to evaluate performance of various OpenMP 3.0 constructs within an offloaded target region.

OpenMPI Examples

The OpenMPI examples will be installed in the directory */usr/share/ti/examples/openmpi*.

Sample Application	Details
testmpi	Basic MPI loopback test from node A to node B (good to verify A->B connectivity). Reports host names of connected nodes.
nbody	Simplified 3D nbody example, up to 1000 particles
mpi_bw	It does measurement of MPI BW for each node pair.

OpenMPI + OpenCL Examples

The OpenMPI + OpenCL examples will be installed in the directory */usr/share/ti/examples/openmpi+opencl*.

Sample Application	Details
mutinode_batch_fftdemo	This example highlights using OpenMPI and OpenCL together. The example offloads the FFT calculations in batches of 64 FFTs to node over MPI. Each individual EVM them further divides this into groups of 8 FFTs to be dispatched to a single DSP core. Each DSP core uses DMA to concurrently perform data transfers while a single FFT is being processed. MPI is then used to aggregate the results to an output file while retaining the order of the FFTs.

OpenMPI + OpenCL + OpenMP Examples

The OpenMPI + OpenCL + OpenMP examples will be installed in the directory */usr/share/ti/examples/openmpi+opencl+openmp*.

Sample Application	Details
mutinode_fftdemo	This example highlights using OpenMPI, OpenCL, and OpenMP. OpenmPI delegates tasks to nodes in a cluster while OpenCL offloads these tasks to the DSP cores on each node. OpenMP is used for parallel processing over the DSP cores.

OpenMPI + OpenMPAcc Examples

The OpenMPI + OpenMPAcc examples will be installed in the directory */usr/share/ti/examples/openmpi+openmpacc*.

Sample Application	Details
vecadd_mpi	Use of offloaded target regions along with MPI calls using the TI OpenMPI library to perform vector addition across multiple nodes.

{{ 1. switchcategory:MultiCore= - For technical support on MultiCore devices, please post your questions in the C6000 MultiCore Forum - For questions related to the BIOS MultiCore SDK (MCSDK), please use the BIOS Forum Please post only comments related to the article **MCSDK HPC 3.x Examples** here.	Keystone= - For technical support on MultiCore devices, please post your questions in the C6000 MultiCore Forum - For questions related to the BIOS MultiCore SDK (MCSDK), please use the BIOS Forum Please post only comments related to the article **MCSDK HPC 3.x Examples** here.	C2000=*For technical support on the C2000 please post your questions on The C2000 Forum. Please post only comments about the article* ***MCSDK HPC 3.x Examples*** *here.*	DaVinci=*For technical support on DaVincoplease post your questions on The DaVinci Forum. Please post only comments about the article* ***MCSDK HPC 3.x Examples*** *here.*	MSP430=*For technical support on MSP430 please post your questions on The MSP430 Forum. Please post only comments about the article* ***MCSDK HPC 3.x Examples*** *here.*	OMAP35x=*For technical support on OMAP please post your questions on The OMAP Forum. Please post only comments about the article* ***MCSDK HPC 3.x Examples*** *here.*	OMAPL1=*For technical support on OMAP please post your questions on The OMAP Forum. Please post only comments about the article* ***MCSDK HPC 3.x Examples*** *here.*	MAVRK=*For technical support on MAVRK please post your questions on The MAVRK Toolbox Forum. Please post only comments about the article* ***MCSDK HPC 3.x Examples*** *here.*	*For technic please pos questions a http://e2e.t Please pos comments article* ***MC 3.x Examp*** }}

Links

Processors

- ARM Processors

Amplifiers & Linear
Audio
Broadband RF/IF & Digital Radio
Clocks & Timers
Data Converters

DLP & MEMS
High-Reliability
Interface
Logic
Power Management

- Digital Signal Processors (DSP)
- Microcontrollers (MCU)
- OMAP Applications Processors

Switches & Multiplexers
Temperature Sensors & Control ICs
Wireless Connectivity

Retrieved from "https://processors.wiki.ti.com/index.php?title=MCSDK_HPC_3.x_Examples&oldid=200594"

This page was last edited on 22 May 2015, at 13:46.