phyCORE-AM57x C66x DSP / phyCORE-AM57x using C66x DSP to accelerate computation

https://develop.phytec.com/phycore-am57x/latest/software-development/application-development/offloading-computation-to-dsps

AM57x CPUC66x DSP

DSPDSPDSP

BSPOpenCLAPIBSP

http://downloads.ti.com/mctools/esd/docs/opencl/offload.html

CPU100x100

matmul_arm.cpp

```
#include <cassert>
#include <cstdlib>
#include <iostream>

using namespace std;

const int DIM       = 100;
const int mat_N     = DIM;
const int mat_K     = DIM;
const int mat_M     = DIM;

void mat_mpy(const float *A, const float *B, float *C, int mat_N, int mat_K, int mat_M)
{
    for (int col = 0; col < mat_M; ++col)
        for (int row = 0; row < mat_N; ++row)
        {
            C[row*mat_M+col] = 0;
            for (int i = 0; i < mat_K; ++i)
                C[row*mat_M+col] += A[row*mat_K+i] * B[i*mat_M+col];
        }
}

int main(int argc, char *argv[])
{
    size_t mat_size = DIM * DIM * sizeof(float);

    // Allocate matrices
    float *A      = (float *) malloc(mat_size);
    float *B      = (float *) malloc(mat_size);
    float *C      = (float *) malloc(mat_size);
    // Ensure memory was successfully allocated
    assert(A != nullptr && B != nullptr && C != nullptr && C != nullptr);

    // Initialize matrices
    srand(time(0));
    for (int i=0; i < mat_N * mat_K; ++i) A[i] = rand() % 5 + 1;
    for (int i=0; i < mat_K * mat_M; ++i) B[i] = rand() % 5 + 1;
    for (int i=0; i < mat_N * mat_M; ++i) C[i] = 0.0;

    // Multiply matrices C = A x B
    mat_mpy(A, B, C, mat_N, mat_K, mat_M);

    free(A);
    free(B);
    free(C);

    return 0;
}
```

phyCORE AM57x SDK / phyCORE AM57x how to install SDK and use

```
$CXX -std=c++11 matmul_arm.cpp -o matmul_arm
```

```
[linux-devkit]:~> $CPP -O matmul_arm.cpp -o matmul_arm
[linux-devkit]:~> ls
matmul_arm  matmul_arm.cpp
```

OpenCLDSP

```
const std::string kernelSrc = R"(
    kernel void ocl_matmpy(const global float *a, const global float *b, global float *c, int mat_K, int mat_N)
    {
        int col    = get_global_id(0);
        int mat_M  = get_global_size(0);

        for (int row = 0; row < mat_N; ++row)
        {
            c[row * mat_M + col] = 0;
            for (int i = 0; i < mat_K; ++i)
                c[row * mat_M + col] += a[row*mat_K+i] * b[i*mat_M+col];
        }
    }
)";
```

OpenCL KernelOpenCL-CCOpenCL-C

kernel

1.
2. Queue
3. OpenCL-Ckernel
4. QueueKernel
5.

matmul_dsp.cpp

```
#include <iostream>
#include <cstdlib>
#include <assert.h>
#include <utility>
#include "ocl_util.h"

#define __CL_ENABLE_EXCEPTIONS
#include <CL/cl.hpp>
/*****************************************************************************
* C[N][M] = A[N][K] * B[K][M];
*****************************************************************************/
using namespace cl;

using std::cout;
using std::cerr;
using std::endl;

const int DIM       = 100;
const int mat_N     = DIM;
const int mat_K     = DIM;
const int mat_M     = DIM;

const std::string kernelSrc = R"(
        kernel void ocl_matmpy(const global float *a, const global float *b, global float *c, int mat_K,int
mat_N)
        {
            int col    = get_global_id(0);
            int mat_M  = get_global_size(0);

            for (int row = 0; row < mat_N; ++row)
            {
                c[row * mat_M + col] = 0;
                for (int i = 0; i < mat_K; ++i)
                    c[row * mat_M + col] += a[row*mat_K+i] * b[i*mat_M+col];
            }
        }
)";

void mat_mpy_ocl(float *A, float *B, float *C, int mat_N, int mat_K, int mat_M, std::size_t mat_size)
{
```

```
   try
   {
     // Initialize context and command queue
     Context context(CL_DEVICE_TYPE_ACCELERATOR);
     std::vector<Device> devices = context.getInfo<CL_CONTEXT_DEVICES>();
     CommandQueue Q (context, devices[0]);

     // Build the OpenCL program
     Program::Sources source(1, std::make_pair(kernelSrc.c_str(), kernelSrc.length()));
     Program P = Program(context, source);
     P.build(devices);

     // Create buffers from memory allocated via __malloc_ddr
     Buffer bufA(context, CL_MEM_READ_ONLY|CL_MEM_USE_HOST_PTR,   mat_size, A);
     Buffer bufB(context, CL_MEM_READ_ONLY|CL_MEM_USE_HOST_PTR,   mat_size, B);
     Buffer bufC(context, CL_MEM_WRITE_ONLY|CL_MEM_USE_HOST_PTR,  mat_size, C);

     // Create kernel and set up arguments
     Kernel K(P, "ocl_matmpy");
     K.setArg(0, bufA);
     K.setArg(1, bufB);
     K.setArg(2, bufC);
     K.setArg(3, mat_K);
     K.setArg(4, mat_N);

     // Run the kernel and wait for completion
     Event E;
     Q.enqueueNDRangeKernel(K, NullRange, NDRange(mat_M), NDRange(1), NULL, &E);
     E.wait();
   }
   catch (Error err)
   {
     cerr << "ERROR: " << err.what() << "(" << err.err() << ", " << ocl_decode_error(err.err()) << ")" << endl;
     exit(-1);
   }
}

int main(int argc, char *argv[])
{
   std::size_t mat_size = DIM * DIM * sizeof(float);

   // Allocate matrices
   float *A      = (float *) __malloc_ddr(mat_size);
   float *B      = (float *) __malloc_ddr(mat_size);
   float *C      = (float *) __malloc_ddr(mat_size);

   assert(A != nullptr && B != nullptr && C != nullptr && C != nullptr);

   // Initialize matrices
   srand(42);
   for (int i=0; i < mat_N * mat_K; ++i) A[i] = rand() % 5 + 1;
   for (int i=0; i < mat_K * mat_M; ++i) B[i] = rand() % 5 + 1;
   for (int i=0; i < mat_N * mat_M; ++i) C[i] = 0.0;

   // Multiple matrices C = A x B
   mat_mpy_ocl(A, B, C, mat_N, mat_K, mat_M, mat_size);

   // Free the matrices
   __free_ddr(A);
   __free_ddr(B);
   __free_ddr(C);

   return 0;
}
```

```
$CXX -O3 -std=c++11  matmul_dsp.cpp -lOpenCL -locl_util -o matmul_dsp
```

```
root@am57xx-phycore-kit:~# ./matmul_dsp
[ 2249.559129] omap-iommu 40d01000.mmu: 40d01000.mmu: version 3.0
[ 2249.565998] omap-iommu 40d02000.mmu: 40d02000.mmu: version 3.0
```

opencl

```
root@am57xx-phycore-kit:~# ls /usr/share/ti/examples/opencl/
Makefile                        ooo_callback
abort_exit                      persistent_clock_concurrent
buffer                          persistent_clock_spanning
ccode                           persistent_common
conv1d                          persistent_kernel_timeout
dgemm                           persistent_messageq_concurrent
dspheap                         persistent_task_concurrent
dsplib_fft                      persistent_task_spanning
edmamgr                         platforms
float_compute                   sgemm
make.inc                        simple
matmpy                          timeout
monte_carlo                     vecadd
null                            vecadd_openmp
offline                         vecadd_openmp_t
offline_embed                   vecadd_subdevice
```

makefie

```
root@am57xx-phycore-kit:/usr/share/ti/examples/opencl/simple# ls
Makefile       kernel.cl      simple.cpp
Makefile.rtos  simple         simple.o
 oot@am57xx-phycore-kit:/usr/share/ti/examples/opencl/simple# rm simple simple.o
root@am57xx-phycore-kit:/usr/share/ti/examples/opencl/simple# ls
Makefile       Makefile.rtos  kernel.cl      simple.cpp
root@am57xx-phycore-kit:/usr/share/ti/examples/opencl/simple# make
Compiling simple.cpp
g++ -c -O3 -I/usr/include -Wall simple.cpp
root@am57xx-phycore-kit:/usr/share/ti/examples/opencl/simple# ls
Makefile       kernel.cl      simple.cpp
Makefile.rtos  simple         simple.o
root@am57xx-phycore-kit:/usr/share/ti/examples/opencl/simple# ./simple
[ 2794.483006] omap-iommu 40d01000.mmu: 40d01000.mmu: version 3.0
[ 2794.488907] omap-iommu 40d02000.mmu: 40d02000.mmu: version 3.0
Done!
root@am57xx-phycore-kit:/usr/share/ti/examples/opencl/simple#
```

http://downloads.ti.com/mctools/esd/docs/opencl/examples/index.html

Martrix GUI DemoC66x DSPlinux

DSP			
opencl - Floating Point Computation	opencl	https://downloads.ti.com/mctools/esd/docs/opencl/examples/float_compute.html#	https://github.com/rcn-ee/ti-opencl/tree/master/examples/float_compute
opencl - Vector Addition	opencl	https://downloads.ti.com/mctools/esd/docs/opencl/examples/overview.html#vecadd-example	https://github.com/rcn-ee/ti-opencl/tree/master/examples/vecadd
Video Analytics Demo	opencl	http://processors.wiki.ti.com/index.php/Processor_SDK_Demos_Video_Analytics	Processor SDK Linux Installer, under "example-applications" folder
Multimedia - DSP C66 Image Processing	gstreamer	http://software-dl.ti.com/processor-sdk-linux/esd/docs/latest/linux/Foundational_Components_Multimedia.html	