Using the zc706_mem Platform

The zc706_mem platform adds a PL-based memory controller to the basic zc706 platform. This allows hardware functions to access an additional 1GB of DRAM that is available on the zc706 board.

- Additional Memory details: 1GB DDR3 with a 400 MHz 64b Phy interface
- Additional AXI slave interface on the platform (SP0) to access this memory: 64b @100 MHz

This platform is designed to illustrate how a second memory interface can be provided as part of a platform. It is not designed to exercise the full bandwidth of the memory. Advanced users who are familiar with Vivado, can modify this platform to increase the speed and data-width of the AXI interface.

The GP1 interface of the PS is connected to the additional memory to allow software access for testing. As a result, the GP1 interface is not available to the platform user, and the additional memory is mapped into the processor address space at 0x80000000:0xBFFFFFFF.

Structure of an Application Using the zc706_mem Platform

Several sample applications using the zc706_mem platform are provided in the `samples/zc706_mem_apps` directory. All the samples are variants of the basic matrix multiplication design where the hardware function accesses the SP0 ("MIG") memory interface for one or more of its input/output parameters. This is done using the sys_port pragma, as shown in this example:

```
#pragma SDS data sys_port(in_A:ACP, in_B:ACP, out_C:MIG)
int mmult_accel (float *in_A, float *in_B, float *out_C);
```

The additional memory is not directly visible to the linux kernel, so it is not used by the standard memory allocators such as malloc() or the special contiguous memory allocators such as sds_alloc(). It is possible to create special allocators that manage this additional 1GB memory space but for these examples, we simply use specific physical addresses in the correct range for this memory, and use sds_mmap() to produce a virtual address that the processor can use to access this memory. The hardware function uses the physical address directly. Sample code to do this is shown below:

```
float *toutBufHw=NULL;
toutBufHw = (float *)sds_mmap((void *)(0x80000000),
                              A_NROWS * B_NCOLS * sizeof(float),
                              toutBufHw);
                              …
mmult_accel(tin1Buf, tin2Buf, toutBufHw);
```

The SDSoC datamover libraries pass the correct physical address to the hardware, given the mmap-ed virtual address. The hardware function uses Vivado HLS pragmas to directly access the memory using a physical address.

```
int mmult_accel (float *in_A, float *in_B, float *out_C){
#pragma HLS INTERFACE m_axi port=out_C depth=1024 offset=direct bundle=mig
```