6.12.8 Synchronization Functions

The OpenCL C programming language implements the following synchronization function.

Function	**Description**
void **barrier** (cl_mem_fence_flags *flags*)	All work-items in a work-group executing the kernel on a processor must execute this function before any are allowed to continue execution beyond the **barrier**. This function must be encountered by all work-items in a work-group executing the kernel. If **barrier** is inside a conditional statement, then all work-items must enter the conditional if any work-item enters the conditional statement and executes the **barrier**. If **barrier** is inside a loop, all work-items must execute the **barrier** for each iteration of the loop before any are allowed to continue execution beyond the **barrier**. The **barrier** function also queues a memory fence (reads and writes) to ensure correct ordering of memory operations to local or global memory. The *flags* argument specifies the memory address space and can be set to a combination of the following literal values. CLK_LOCAL_MEM_FENCE - The **barrier** function will either flush any variables stored in local memory or queue a memory fence to ensure correct ordering of memory operations to local memory. CLK_GLOBAL_MEM_FENCE – The **barrier** function will queue a memory fence to ensure correct ordering of memory operations to global memory. This can be useful when work-items, for example, write to buffer or image objects and then want to read the updated data.

Table 6.16 *Built-in Synchronization Functions*

6.12.9 Explicit Memory Fence Functions

The OpenCL C programming language implements the following explicit memory fence functions to provide ordering between memory operations of a work-item.

Function	Description
void **mem_fence** (cl_mem_fence_flags *flags*)	Orders loads and stores of a work-item executing a kernel. This means that loads and stores preceding the **mem_fence** will be committed to memory before any loads and stores following the **mem_fence**. The *flags* argument specifies the memory address space and can be set to a combination of the following literal values: CLK_LOCAL_MEM_FENCE CLK_GLOBAL_MEM_FENCE.
void **read_mem_fence** (cl_mem_fence_flags *flags*)	Read memory barrier that orders only loads. The *flags* argument specifies the memory address space and can be set to to a combination of the following literal values: CLK_LOCAL_MEM_FENCE CLK_GLOBAL_MEM_FENCE.
void **write_mem_fence** (cl_mem_fence_flags *flags*)	Write memory barrier that orders only stores. The *flags* argument specifies the memory address space and can be set to to a combination of the following literal values: CLK_LOCAL_MEM_FENCE CLK_GLOBAL_MEM_FENCE.

Table 6.17 *Built-in Explicit Memory Fence Functions*