6.13.8 Synchronization Functions

The OpenCL C programming language implements the following synchronization functions.

Function	Description
void **work_group_barrier**[39] (cl_mem_fence_flags *flags*) void **work_group_barrier** (cl_mem_fence_flags *flags*, memory_scope *scope*[40])	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 **work_group_barrier**. This function must be encountered by all work-items in a work-group executing the kernel. These rules apply to ND-ranges implemented with uniform and non-uniform work-groups. If **work_group_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 **work_group_barrier**. If **work_group_barrier** is inside a loop, all work-items must execute the **work_group_barrier** for each iteration of the loop before any are allowed to continue execution beyond the **work_group_barrier**. The **work_group_barrier** function also supports a variant that specifies the memory scope. For the **work_group_barrier** variant that does not take a memory scope, the *scope* is `memory_scope_work_group`. The *scope* argument specifies whether the memory accesses of work-items in the work-group to memory address space(s) identified by *flags* become visible to all work-items in the work-group, the device or all SVM devices. The **work_group_barrier** function can also be used to specify which memory operations i.e. to global memory, local memory or images become visible to

[39] The built-in function **barrier** has been renamed **work_group_barrier**. For backward compatibility, **barrier** is also supported.

[40] Refer to *section 6.13.11* for description of memory_scope.

	the appropriate memory scope identified by *scope*. The *flags* argument specifies the memory address spaces. This is a bitfield and can be set to 0 or a combination of the following values ORed together. When these flags are OR'ed together the **work_group_barrier** acts as a combined barrier for all address spaces specified by the flags ordering memory accesses both within and across the specified address spaces. CLK_LOCAL_MEM_FENCE - The **work_group_barrier** function will ensure that all local memory accesses become visible to all work-items in the work-group. Note that the value of *scope* is ignored as the memory scope is always `memory_scope_work_group`. CLK_GLOBAL_MEM_FENCE – The **work_group_barrier** function ensure that all global memory accesses become visible to the appropriate scope as given by *scope*. CLK_IMAGE_MEM_FENCE – The **work_group_barrier** function will ensure that all image memory accesses become visible to the appropriate scope as given by *scope*. The value of *scope* must be `memory_scope_work_group`. The values of *flags* and *scope* must be the same for all work-items in the work-group.

Table 6.16 *Built-in Synchronization Functions*