Performing Reductions in OpenCL

Here's What You Would Change in your Host Program

Here's the Problem We are Trying to Solve

Reduction Takes Place in a Single Work-Group

The Arguments to the Kernel
Reduction Takes Place Within a Single Work-Group
Each work-item is run by a single thread

Thread #0:
\[ prods[0] += prods[1] \]
Thread #2:
Thread #4:
Thread #6:

Reduction Takes Place in a Single Work-Group
Each work-item is run by a single thread

Thread #0:
\[ prods[0] += prods[2] \]
Thread #4:
Thread #0:
\[ prods[0] += prods[4] \]

Reduction Performance
Work-Group Size = 32

And, Finally, in your Host Program

Wait(cmdQueue);
double start = omp_get_wtime();
status = clEnqueueNDRangeKernel(cmdQueue, kernel, 1, NULL, globalWorkSize, localWorkSize, 0, NULL, NULL);
PrintCLError(status, "clEnqueueNDRangeKernel failed: ");
Wait(cmdQueue);
double time1 = omp_get_wtime();
status = clEnqueueReadBuffer(cmdQueue, dC, CL_TRUE, 0, numWorkGroups*sizeof(float), hC, 0, NULL, NULL);
PrintCLError(status, "clEnqueueReadBuffer failed: ");
Wait(cmdQueue);
float sum = 0.0f;
for(int i = 0; i < numWorkgroups; i++)
{ sum += hC[i]; }

A Review of Bitmasks

Remember Truth Tables?

\[ F \& F = F \]
\[ F \& T = F \]
\[ T \& F = F \]
\[ T \& T = T \]
\[ 0 \& 0 = 0 \]
\[ 0 \& 1 = 0 \]
\[ 1 \& 0 = 0 \]
\[ 1 \& 1 = 1 \]

Or, with Bits:

\[ \begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 \\
\end{array} \]

Or, with Multiple Bits:

\[ \begin{array}{cccc}
000 & 001 & 010 & 011 \\
001 & 011 & 100 & 101 \\
\end{array} \]