	Result	Size	Time	Cycles	GPU	SM Frequency	Process	Attributes
Current	570 - init_cos_sin_table_kernel	(12, 1, 1)x(16, 1, 1)	1.86 us	2,632	0 - NVIDIA GeForce GTX 1660 Ti	1.41 Ghz	[11724] image	

Summary | Details | Source | Context | Comments | Raw | Session

Compare | Tools | View | Export

This table shows all results in the report. Use the column headers to sort the results in this report. Double-click a result to see detailed metrics. Double-click on demangled names to rename it.

ID	Estimated Speedup [%]	Function Name	Demangled Name	Duration [ms] (6.45 ms)	Runtime Improvement [ms] (4.27 ms)	Compute Throughput [%]	Memory Throughput [%]	# Registers [register/thread
0	93.70	init_cos_sin_table_...	init_cos_sin_table_...	0.00	0.00	0.32	2.36	18
1	85.54	image_RGB2BW_ke...	image_RGB2BW_ke...	0.23	0.20	87.88	12.78	16
2	81.61	noiseReduction_ker...	noiseReduction_ker...	2.03	1.66	82.03	5.03	40

The following performance optimization opportunities were discovered for this result. Follow the rule links to see more context on the Details page.
Note: *Speedup estimates provide upper bounds for the optimization potential of a kernel assuming its overall algorithmic structure is kept unchanged.*

Achieved Occupancy
Est. Speedup: 93.70%

The difference between calculated theoretical (50.0%) and measured achieved occupancy (3.1%) can be the result of warp scheduling overheads or workload imbalances during the kernel execution. Load imbalances can occur between warps within a block as well as across blocks of the same kernel. See the CUDA Best Practices Guide for more details on optimizing occupancy.

Imc Miss Stalls
Est. Speedup: 53.08%

On average, each warp of this workload spends 8.9 cycles being stalled waiting for an immediate constant cache (IMC) miss. A read from constant memory costs one memory read from device memory only on a cache miss; otherwise, it just costs one read from the constant cache. Immediate constants are encoded into the SASS instruction as 'c[bank][offset]'. Accesses to different addresses by threads within a warp are serialized, thus the cost scales linearly with the number of unique addresses read by all threads within a warp. As such, the constant cache is best when threads in the same warp access only a few distinct locations. If all threads of a warp access the same location, then constant memory can be as fast as a register access. This stall type represents about 53.1% of the total average of 16.7 cycles between issuing two instructions.

Block Size
Est. Speedup: 50.00%

Threads are executed in groups of 32 threads called warps. This kernel launch is configured to execute 16 threads per block. Consequently, some threads in a warp are masked off and those hardware resources are unused. Try changing the number of threads per block to be a multiple of 32 threads. Between 128 and 256 threads per block is a good initial range for experimentation. Use smaller thread blocks rather than one large thread block per multiprocessor if latency affects performance. This is particularly beneficial to kernels that frequently call __syncthreads(). See the Hardware Model description for more details on launch configurations.