	Result	Size	Time	Cycles	GPU	SM Frequency	Process	Attributes
Current	570 - init_cos_sin_table_kernel	(1, 180, 1)x(672, 1, 1)	3.33 us	4,765	0 - NVIDIA GeForce GTX 1660 Ti	1.43 Ghz	[58012] 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 [us] (2261.22 us)	Runtime Improvement [us] (1481.95 us)	Compute Throughput [%]	Memory Throughput [%]	# Reg
0	37.29	init_cos_sin_table_...	init_cos_sin_table_...	3.33	1.24	13.29	13.29	
1	84.74	image_RGB2BW_ke...	image_RGB2BW_ke...	238.98	202.50	87.94	12.52	
2	62.43	noiseReduction_ker...	noiseReduction_ker...	227.33	141.92	46.59	48.61	

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.*

Imc Miss Stalls
Est. Speedup: 37.29%

On average, each warp of this workload spends 6.5 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 37.3% of the total average of 17.4 cycles between issuing two instructions.

Theoretical Occupancy
Est. Speedup: 34.38%

The 5.25 theoretical warps per scheduler this kernel can issue according to its occupancy are below the hardware maximum of 8. This kernel's theoretical occupancy (65.6%) is limited by the number of warps within each block.

Achieved Occupancy
Est. Speedup: 19.46%

The difference between calculated theoretical (65.6%) and measured achieved occupancy (52.9%) 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.