

Cooperative Matrix in NVK


Karol Herbst, Red Hat

Sep 30, 2025

NVK support



- Int8, fp16, fp32
- All hw supported sizes (e.g. 16x8x32, 16x8x16)
- CUDA documentation is pretty helpful

Layout: int8

Row \ Col	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T0:{a0, a1, a2, a3}				T1:{a0, a1, a2, a3}				T2:{a0, a1, a2, a3}				T3:{a0, a1, a2, a3}			
1	T4:{a0, a1, a2, a3}				T5:{a0, a1, a2, a3}				T6:{a0, a1, a2, a3}				T7:{a0, a1, a2, a3}			
2																
..																
7	T28:{a0, a1, a2, a3}				T29:{a0, a1, a2, a3}				T30:{a0, a1, a2, a3}				T31:{a0, a1, a2, a3}			

%laneid:{fragments}

Layout: fp16, fp32, int32

Row\Col	0	1	2	3	4	5	6	7
0	T0: {c0, c1}		T1: {c0, c1}		T2: {c0, c1}		T3: {c0, c1}	
1	T4: {c0, c1}		T5: {c0, c1}		T6: {c0, c1}		T7: {c0, c1}	
2								
..								
7	T28: {c0, c1}		T29: {c0, c1}		T30: {c0, c1}		T31: {c0, c1}	

%laneid:{fragments}

Performance

- Micro benchmark: `vk_cooperative_matrix_perf`
- Initially 20% perf compared to Nvidia
- Up to 70% on main
- Up to 92% on devel branch

LDSTM

- Quads load consecutive 128 bits into GPR, 32 per invocation
- Can load 1, 2 or 4 groups of 128 bits
- Address taken from invocation $quad_id + 8 * group$
- *LDSTM.X2 R4 [R1 + 0x1200]*
 - Invocation 0 (quad 0) loads Invocation 0's $R1 + 0x1200 + 0x0$ into R4
 - Invocation 0 (quad 0) loads Invocation 8's $R1 + 0x1200 + 0x0$ into R5
 - ...
 - Invocation 9 (quad 2) loads Invocation 2's $R1 + 0x1200 + 0x4$ into R4
 - ...
- No idea if useful for anything else

Occupancy

- Registers and shared memory shared between workgroups
- Shared memory config limits concurrency of workgroups
- E.g. ran 1 of 2 possible workgroups in benchmarks
- !37135
 - Phomes benchmarked this MR:
 - Lego Builders Journey 22 → 28 fps
 - Atomic heart 40 → 48 fps
 - Hitman 3 benchmark 83 → 91 fps
 - Smaller gains in various other games
- +100% perf in vk_cooperative_matrix_perf

Memory barriers

- MEMBAR instruction
- Used .GPU scope → slow
- Use .CTA instead → fast
- +50% perf in vk_cooperative_matrix_perf

Address calculation

- Load/Stores use GPR form only
- hardware can do: $[GPR.stride + UGPR + offset]$
 - stride: x1, x4, x8, x16 (shared memory only)
 - offset: 24 bit signed, unsigned if *RZ* + *URZ*
 - GPR can be 32 bits if UGPR is 64 bits
- Range analysis + *nir_opt_offset*
- +5% perf in vk_cooperative_matrix_perf
- !36113