# SPMD / SIMD on GPUs

Patrick Steinbrecher



# Terminology

CPU	GPU
Core	<b>EU</b> Execution Unit
Hyper-Thread	Hardware-Thread
Vector Registers	<b>GRF</b> General Register File

- Hardware-thread executes SIMD
- EU has 7 hardware-threads
  - 4 kB of GRF per thread
    - e.g. 128 SIMD8 registers for 4 byte data type
- 8 EUs == 1 Sub-Slice (SS)
  - shared L1 cache per SS



2

# **Execution Unit**

- HW-thread executes SIMD
  - Native SIMD length = 256 bit
- Flexible SIMD width
  - SIMD1, SIMD2, ..., SIMD16, SIMD32
    - might be masked or broken down in hardware for native width
  - SIMD8 enough to reach SP-peak
  - SIMD4 = ½ SP-peak
- Code branches
  - HW-thread will execute both branches in serial and mask off SIMD lanes for each branch correspondingly



#### Load/Store

- Referred to as 'send' instruction
- Scatter/Gather supported
  - Same performance as aligned block load if gather has addresses in same cache line
- 'send' can permute data in load path

# **Programming Models**

- How can parallelism be expressed?
  - SIMD: Single Instruction Multiple Data
  - SPMD: Single Program Multiple Data
- SIMD and SPMD have strong memory layout requirements
  - See next slide
- Support for SIMD and SPMD is not limited by hardware in general. It is dependent on availability of modern compilers.
- Intel GPUs and CPUs have compiler support for <u>SPMD and SIMD</u>!

# Memory Layouts

```
struct point_aos {
    int x, y, z;
};
point_aos *points = (point_aos*)malloc( N*sizeof(point_aos) );
point_aos *points_a = ...;
point_aos *points_b = ...;
for ( int i = 0; i < N; ++i ) {
    points_a[i].x += points_b[i].x;
    points_a[i].y += points_b[i].y;
    points_a[i].z += points_b[i].z;
}</pre>
```

- Array of Structs (AoS)
  - Very poor cache line utilization
- Struct of Arrays (SoA)
  - Close to 100% cache line utilization
  - In simple cases can have 100% cache line utilization
- Array of Structs of Arrays (AoSoA)
  - 100% cache line utilization
- AoSoA and SoA are required memory layouts for SPMD and SIMD programming model

```
struct point_aosoa {
    int x[8], y[8], z[8];
};
point_aosoa *points = (point_aosoa*)malloc( (N/8)*sizeof(point_aosoa) );
point_aosoa *points_a = ...;
point_aosoa *points_b = ...;
for ( int i = 0; i < N/8; ++i ) {
    for ( int j = 0; j < 8; ++j ) {
        points_a[i].x[j] += points_b[i].x[j];
        points_a[i].z[j] += points_b[i].y[j];
        points_a[i].z[j] += points_b[i].z[j];
    }
}</pre>
```

```
struct point_soa {
    int *x, *y, *z;
};
point_soa points;
points.x = (int*)malloc( N*sizeof(int) );
points.y = (int*)malloc( N*sizeof(int) );
points.z = (int*)malloc( N*sizeof(int) );
point_soa *points_a = ...;
point_soa *points_b = ...;
for ( int i = 0; i < N; ++i ) {
    points_a.x[i] += points_b.x[i];
    points_a.y[i] += points_b.y[i];
    points_a.z[i] += points_b.z[i];
}</pre>
```

# **OpenMP compile-time options**

### SPMD-mode

- OpenMP thread = SIMD lane
  - Vectorization over multiple threads
- Use GPU like a traditional GPU
  - Ideal for applications with prior OpenCL-like background

### SIMD-mode

- OpenMP thread = EU thread
  - Vectorization within a thread
- <u>Use GPU like a CPU</u>
  - Ideal for applications with prior CPU background
- Has 1 additional level of hierarchical parallelism
  - #pragma omp simd

## **OpenMP compile-time options**

#### SPMD-mode

```
#pragma omp target teams distribute parallel for
for ( int i = 0; i < TOTAL_SIZE; ++i ) {</pre>
```

```
c[i] = a[i] + b[i];
}
```

#### SIMD-mode

}

}

```
constexpr int64_t SIZE = 8192;
constexpr int64_t SIMD_SIZE = 16;
constexpr int64_t TOTAL_SIZE = SIZE*SIMD_SIZE;
```

```
#pragma omp target teams distribute parallel for
for ( int isimd = 0; isimd < SIZE; ++isimd ) {</pre>
```

```
#pragma omp simd simdlen(SIMD_SIZE)
for ( int ilane = 0; ilane < SIMD_SIZE; ++ilane ) {</pre>
```

const int index = isimd\*SIMD\_SIZE + ilane;

```
c[index] = a[index] + b[index];
```

### Kokkos with SIMD-mode on Intel GPUs

int N, M, J;

	<pre>using policy_t = Kokkos::TeamPolicy<execspace>; using team_t = typename Kokkos::TeamPolicy<execspace>::member_type; using scratch_t = Kokkos::View<int**, execspace,="" kokkos::memorytraits<kokkos::unmanaged=""> &gt;;</int**,></execspace></execspace></pre>				
therease a own target	<pre>const int ssize = scratch_t::shmem_size(M, K); const int slevel = 1;</pre>				
#pragma omp parallel	Kokkos::parallel_for( team_policy(N,Kokkos::AUTO).set_scratch_size(slevel, Kokkos::PerTeam(ssize), KOKKOS_LAMBDA(const team_t &team) {				
in braging only balance	<pre>const int iteam = team.league_rank();</pre>				
	<pre>scratch_t scratch_matrix(team.team_scratch(scratch_level), sX, sY);</pre>				
	Kokkos::parallel_for( Kokkos::TeamThreadRange(team, 0, M), [&](const int i) { #pragma omp for nowait				
	// code				
#pragma omp simd	<pre>Kokkos::parallel_for( Kokkos::ThreadVectorRange(team, 0, J), [&amp;](const int j) {</pre>				
	// code } ); } );				
#pragma omp barrier	<pre>team.team_barrier();</pre>				
	Kokkos::parallel_for( Kokkos::TeamVectorRange(team, 0, M), [&](const int i) { #pragma omp for simd nowait				
	// code } );				
if ( omp_get_thread_num() == 0 )   ———	★ Kokkos::single(Kokkos::PerTeam(team), [&]() {				
	<pre>}); }); </pre>				
	Kokkos::fence();				

# HotQCD implementation using OpenMP

- One code base that runs on both CPU and GPU
  - Without any changes in kernels between both architectures
  - No defines to change code path
  - No intrinsics
- Relies on complex\_simd<float\_type,n> class which:
  - Uses internal float\_type array
  - Uses #pragma omp simd for SIMD-mode
  - Can load from AoSoA with scalar types for SPMD-mode

### **Stencil Operator**



### **Stencil Operator**



### **Multiple Right-hand Sides**



### HotQCD approach to SPMD and SIMD

Code is based on complex\_simd<float\_type,n> class

QCD<no\_simd\_memory\_layout> stencil; stencil.run<no\_simd\_kernel>();

- no\_simd\_memory\_layout: defines AoSoA memory layout
- **no\_simd\_kernel:** defines if kernel uses scalar or SIMD operators
- SIMD-mode: no\_simd\_memory\_layout == no\_simd\_kernel
- SPMD-mode: no\_simd\_kernel=scalar < no\_simd\_memory\_layout</p>
  - introduces outer loop over SIMD lanes
  - scalar types can read from AoSoA memory layout

### Vectorization using AoSoA struct matrix3x3\_aosoa { std::complex<float> e00[16], e01[16], e02[16]; std::complex<float> e10[16], e11[16], e12[16]; std::complex<float> e20[16], e21[16], e22[16]; }; Store separated sites continuously in memory

- Single Instruction = Stencil
- Multiple Data = Sites
- Use vector registers like scalars to perform matrix times vector operations





# **Performance on Integrated Graphics**

- OpenMP compiler generates ideal code vectorization
  - See next slides

- Lattice QCD kernel runs at 90% caching efficiency
  - Scales with more right-hand sides
  - Matches theoretical expectation



### **Complex Multiplication**

```
std::complex<float> *a = ...;
                                                    std::complex<float> *b = ...;
                                                    std::complex<float> *c = ...;
                                                    #pragma omp target teams distribute parallel for simd
                                                    for ( int i = 0; i < N; ++i ) {
                                                        c[i] = a[i] + b[i];
                                                    }
                                                          (81M0)
                                                                  r3.0<1>:d
                                                     shl
                                                                               r124.0<8;8,1>:d
                                                                                                3:w
                                                     add
                                                          (81M0)
                                                                  r123.0<1>:d
                                                                               r3.0<8;8,1>:d
                                                                                                r7.1<0;1,0>:d
                                                          (81M0)
                                                                  r9.0<1>:d
                                                                               r3.0<8;8,1>:d
                                                                                                r7.2<0;1,0>:d
                                                     add
                                                     add
                                                          (81M0)
                                                                  r119.0<1>:d
                                                                               r3.0<8;8,1>:d
                                                                                                r7.0<0;1,0>:d
                                                                  r121:f r123
                                                                                             0x2206C01
                                                          (81M0)
                                                                                  0xC
                                                     send
                                                     send
                                                          (81M0)
                                                                  r14:f
                                                                          r9
                                                                                  0xC
                                                                                             0x2206C02
Each send loads 16 floats
                                                          (81M0)
                                                                  r120.0<1>:f
                                                                               r14.0<8;8,1>:f
                                                                                                r122.0<8;8,1>:f
                                                     mul
                                                          (81M0)
                                                                  r13.0<1>:f
                                                                               r15.0<8;8,1>:f
                                                                                                r122.0<8;8,1>:f
                                                     mul
                                                                  r17.0<1>:f
                                                                               r120.0<2;1>:f
                                                                                                r15.0<2;1>:f
                                                                                                                r121.0<1>:f
                                                     mad
                                                          (81M0)
              SIMD8 complex mul
                                                                                                r14.0<2;1>:f
                                                     mad
                                                          (81M0)
                                                                  r16.0<1>:f
                                                                               -r13.0<2;1>:f
                                                                                                                r121.0<1>:f
                                                     sends (81M0)
                                                                  null:ud r119
                                                                                  r16
                                                                                                    0x2026C00
                                                                                         0x8C
```

<pre>template <class float_type,="" int="" no_simd_memory=""></class></pre>					
<pre>template <parity_t int="" no_rhs,="" no_simd="" parity,=""></parity_t></pre>					
<pre>void action<float_type,no_simd_memory>::stencil( packed_lattice_vectors</float_type,no_simd_memory></pre>	<pre><float_type,no_simd_memory> const &amp;</float_type,no_simd_memory></pre>	&vec_in	,		
packed_lattice_vectors<	<pre><float_type,no_simd_memory></float_type,no_simd_memory></pre>	&vec_out	) {		
constavant int no isf $-$ no implicit sites no simd no simd memory $O$ :		12526			
constexpr int no_tsr = no_tmpricit_sites <no_sima,no_sima_memory>(),</no_sima,no_sima_memory>		L2536:			-2 0 1 1
		CWD	Shi (81M0)		r2.0<1>:0
<pre>#pragma omp target teams distribute parallel for simd collapse(2) si</pre>	imdlen(no_isf)	(W)	add (SIMO)		r9.1<1>.q
<pre>for ( int siteh_sf = 0; siteh_sf &lt; _lattice.half_volume_sf(); ++site</pre>	eh_sf ) {		add (81M0)		r2 0-1-1d
for ( int isf = $0$ ; isf < no_isf; ++isf ) {		(W)	cmp (81M0)	(ea)f0.1	null<1>:a
		()	mul (81M0)		r3.0<1>:d
colon vector sinds float type no sinds wine mbs];			mul (81M0)		r2.0<1>:d
cotor_vector_stilla< rtoat_type, ho_stilla > v[ho_rhs],			add (81M0)		r3.0<1>:d
			add (81M0)		r2.0<1>:d
<pre>const site_shift<no_simd> site( siteh_sf, parity, isf );</no_simd></pre>			add (81M0)		r11.0<1>:d
			add (81M0)		r4.0<1>:d
for ( int imu = 0; imu < 4; ++imu ) {			add (81M0)		r5.0<1>:d
			add (81M0)		r10.0<1>:d
const auto site up - site shift $accurs(imu ) attice ):$			add (81M0)		r6.0<1>:d
const auto site du site shift coudry ( inu, lattice ),			ada (81M0)		r7.0<1>:0
const auto site_an = site.snitt_eo <an>( imu, iattice );</an>			send (81M0)		rou.1 rz
			send (81M0)		r74.f r5
<pre>su3_simd&lt; float_type, no_simd &gt; link = field.get_eo( sit</pre>	te, imu );	$\rightarrow$	send (81M0)		r66:f r1
			send (81M0)		r4:f r1
for ( int irhs = 0; irhs < no_rhs; ++irhs ) {		$\rightarrow$	send (81M0)		r72:f r6
			add (81M0)		r10.0<1>:d
$v[irbs] \rightarrow link * vec in cet(site up irbs)$			send (81M0)		r68:f r7
			add (81M0)		r11.0<1>:d
3			send (81M0)		r6:f r3
			add (81M0)		r2.0<1>:d
link = field.get_eo( site_dn, imu );			ada (81M0)		r5.0<1>:0
			send (81M0)		n70.1 11
for ( int irhs = 0; irhs < no_rhs; ++irhs ) {	no_ist 3x3 complex matrices		send (81M0)		r64:f r2
			send (81M0)		r2:f r3
v[inhs] = link * vac in act (site dn inhs)	no isf == 8		mul (81M0)		r10.0<1>:f
			mul (81M0)		r12.0<1>:f
3			mul (81M0)		r15.0<1>:f
// end imu	<ul> <li>Requires 9 SIMD16 loads</li> </ul>		mul (81M0)		r11.0<1>:f
			mul (81M0)		r14.0<1>:f
<pre>for ( int irhs = 0; irhs &lt; no_rhs; ++irhs ) {</pre>			mul (81M0)		r29.0<1>:f
			mul (81M0)		r19.0<1>:f
vec out stream( $0.5*v$ [irbs] site irbs):			MUL (81M0)		r17.0<1>:f
1			mul (81M0)		r20.0<1>:T
s 1// and icf			mul (81M0)		r16.0<1>.f
3// enu LST			mad (81M0)		r10.0<1>:f
}// ena siten_st			mad (81M0)		r29.0<1>:f
}			mad (81M0)		r12.0<1>:f

18

r4.0<1>:f

r6.0<1>:f

r4.0<1>:f

r8.0<8;8,1>:d

r9.1<0;1,0>:q

r2.0<8;8,1>:d

r9.1<0;1,0>:q

r8.0<8;8,1>:d

r2.0<8;8,1>:d

r102.0<8;8,1>:d

r58.0<8;8,1>:d

r3.0<8;8,1>:d

r2.0<8;8,1>:d

r2.0<8;8,1>:d

r2.0<8;8,1>:d

r2.0<8;8,1>:d

r2.0<8;8,1>:d

0xC

0xC

0xC

0xC

0xC

0xC

0xC

0xC r2.0<8;8,1>:d

0xC

0xC

0xC

0xC

r2.0<8;8,1>:d

r2.0<8;8,1>:d

r3.0<8;8,1>:d

r79.0<8;8,1>:f

r73.0<8;8,1>:f

r67.0<8;8,1>:f

r79.0<8;8,1>:f

r67.0<8;8,1>:f

r81.0<8;8,1>:f

r75.0<8;8,1>:f

r69.0<8;8,1>:f

r81.0<8;8,1>:f

r75.0<8;8,1>:f

r69.0<8;8,1>:f

-r10.0<2;1>:f

-r29.0<2;1>:f

-r12.0<2;1>:f

r2

r4

r5

r10

r11

r6

r7

r3

r10

r11

r2

r3

r61.0<8;8,1>:d

5:w

1:w

4:w 192:w

576:w

64:w

64:w 192:w

448:w

256:w

384:w

0x02206C01

0x02206C01

0x02206C01

0x02206C01

0x02206C02

0x02206C01

0x02206C01

0x02206C01

0x02206C01

0x02206C01

0x02206C02

r5.0<8;8,1>:f

r5.0<8;8,1>:f

r5.0<8;8,1>:f

r4.0<8;8,1>:f

r4.0<8;8,1>:f

r7.0<8;8,1>:f

r7.0<8;8,1>:f

r7.0<8;8,1>:f

r6.0<8;8,1>:f

r6.0<8;8,1>:f

r6.0<8;8,1>:f

r78.0<2;1>:f

r80.0<2;1>:f

r72.0<2;1>:f

128:w

320:w 0x02206C02

512:w

128:w

r83.0<8;8,1>:d

r9.4<0;1,0>:d

r3.0<8;8,1>:d

r2.0<8;8,1>:d

temp	late	<class< th=""><th><pre>float_type,</pre></th><th>int</th><th>no_simd_memory&gt;</th><th></th></class<>	<pre>float_type,</pre>	int	no_simd_memory>	

}

template <parity\_t parity, int no\_rhs, int no\_simd>

&vec\_out ) {

<pre>constexpr int no_isf = no_implicit_sites<no_simd,no_simd_memory>();</no_simd,no_simd_memory></pre>		L2536	:					
			shl (81M0)		r2.0<1>:d	r8.0<8;8,1>:d	5:w	
#pragma omp target teams distribute parallel for simd collapse(2) s	simdlen(no isf)	(W)	add (11M0)		r9.1<1>:q	r9.1<0;1,0>:q	1:w	
for ( int either of a sitch of a lattice hold value of () and			add (81M0)		r8.0<1>:d	r2.0<8;8,1>:d	r83.0<8;8,1>:d	
for ( int siten_sf = 0; siten_sf < _lattice.half_volume_sf(); ++sit	en_st ) {		add (81M0)		r2.0<1>:d	r61.0<8;8,1>:d	r9.4<0;1,0>:d	
for ( int isf = 0; isf < no_isf; ++isf ) {		(W)	cmp (81M0)	(eq)f0.1	null<1>:q	r9.1<0;1,0>:q	4:w	
			mul (81M0)		r3.0<1>:d	r8.0<8:8.1>:d	192:w	
and an experiment of the transformer and the second states of the second states and the second states and the second states are second states and the second states are second			mul (81M0)		r2.0<1>:d	r2.0<8:8.1>:d	576:w	
color_vector_sima< float_type, no_sima > v[no_rns];			add (81M0)		r3 0<1>.d	r102 0-8.8 1>.d	r3 0-8.8 12.d	
			add (81M0)		r2 0-1>:d	r58 0-8.8 1.d	r7 0~8.8 1~.d	
const site shift <no simd=""> site( siteh sf parity isf ):</no>			add (81M0)		n11 0-1.	n3 0-8-8 1d	64:w	
					m1 Q 1	n 2 0 . 0 . 0 . 1 d	04.W	
					r4.0<1>.0	r2.0<8;8,1>:0	04.W	
for ( int imu = 0; imu < 4; ++imu ) {			ada (81M0)		r5.0<1>:0	r2.0<8;8,1>:d	192:W	
			add (81M0)		r10.0<1>:d	r2.0<8;8,1>:d	448:w	
and all aits an aits shift as an Cime lattice Se			add (81M0)		r6.0<1>:d	r2.0<8;8,1>:d	256:w	
const duto site_up = site.snift_eo <up>( imu, lattice );</up>			add (81M0)		r7.0<1>:d	r2.0<8;8,1>:d	384:w	
<pre>const auto site_dn = site.shift_eo<dn>( imu, lattice );</dn></pre>			send (81M0)		r80:f r2	0xC	0x02206C01	
			send (81M0)		r78:f r4	0xC	0x02206C01	
and sinds floot times up sinds link field out of si	to imu ).		send (81M0)		r74:f r5	0xC	0x02206C01	
sus_sima< float_type, no_sima > link = fleta.get_eo( si	te, Imu );		send (81M0)		r66:f r10	0xC	0x02206C01	
			send (81M0)		r4:f r11	0xC	0x02206C02	
for (int int = $0$ : int < no rest ++ints) {			send (81M0)		r72:f r6	0xC	0x02206C01	
			add (81M0)		r10.0<1>:d	r2.0<8:8.1>:d	128:w	
			send (81M0)		r68.f r7	0xC	0x02206C01	
v[irhs] += link * vec_in.get( site_up, irhs ); <					r11 0-1-1d	r7 0-8.8 12.1	320.W	
3			sond (81M0)		n6.f n3	0vC	0x07206(02	
			add (21M0)		n2 0 1 1 d	m2 0.20.0 1 d	512.m	
					r2.0<1>.0	r2.0<0,0,1>.0	129	
link = field.get_eo( site_dn, imu );					r5.0<1>:0	r5.0<8;8,1>:0	128:W	
			send (81M0)		r/6:f r10	ØXC	0X02206C01	
for (int into - 0; into a no mark winto ) (			send (81M0)		r/0:f r11	ØXC	0x02206C01	
for $($ the true $= 0$ , true $< n_0$ rule, $++$ true $)$ {			send (81M0)		r64:f r2	ØxC	0x02206C01	
			send (81M0)		r2:f r3	0xC	0x02206C02	
v[irhs] -= link * vec_in.get( site_dn, irhs ):	no_isf 3-dim complex vectors		mul (81M0)		r10.0<1>:f	r79.0<8;8,1>:f	r5.0<8;8,1>:f	
	-		mul (81M0)		r12.0<1>:f	r73.0<8;8,1>:f	r5.0<8;8,1>:f	
S S S S S S S S S S S S S S S S S S S			mul (81M0)		r15.0<1>:f	r67.0<8;8,1>:f	r5.0<8;8,1>:f	
}// end imu	no_ist == 8		mul (81M0)		r11.0<1>:f	r79.0<8;8,1>:f	r4.0<8;8,1>:f	
			mul (81M0)		r14.0<1>:f	r67.0<8;8,1>:f	r4.0<8;8,1>:f	
for (int ints - 0, ints < no the units) $\int$			mul (81M0)		r29.0<1>:f	r81.0<8;8,1>:f	r7.0<8;8,1>:f	
	Requires 3 SIMD 16 loads		mul (81M0)		r19.0<1>:f	r75.0<8:8.1>:f	r7.0<8:8.1>:f	
			mul (81M0)		r17.0<1>:f	r69.0<8:8.1>:f	r7.0<8:8.1>:f	
<pre>vec_out.stream( 0.5*v[irhs], site, irhs );</pre>			mul (81M0)		r28 0<1>.f	$r81 \ 0 < 8 \cdot 8 \ 1 > \cdot f$	r6.0 < 8.8.1 > f	
					r18 0-1-1	r75 0-8.8 1.1	$r6 0 - 8 \cdot 8 1 - 1 = 1$	
					n16 0-1-1	n60 0-8.8 1f	n6 0-8.8 1	
}// end ist			mad (2140)		10.0<1>.1	100.0<0,0,12.T	10.0<0,0,1>.T	n1 0 .1f
}// end siteh_sf			mad (SIMO)		"10.0<1>.T	-1110.0<2;1>;T	170.U<2;1>:1	1'4.U<1>:T
					129.0<1>:T	-1'29.0<2;1>:T	1'00.0<2;1>:T	ro.u<1>:f
			mad (81MØ)		r12.0<1>:f	-r12.0<2;1>:f	r/2.0<2;1>:f	r4.0<1>:f

template <cl template <pa void action&lt;</pa </cl 	lass float_type, int no_simd_memory> arity_t parity, int no_rhs, int no_simd> <float_type,no_simd_memory>::stencil( packed_lattice_vectors&lt; packed_lattice_vectors&lt;</float_type,no_simd_memory>	<pre>cfloat_type,no_simd_memory&gt; const &amp;v cfloat_type,no_simd_memory&gt; &amp;v</pre>	ec_in , ec_out ) {
constexp	<pre>or int no_isf = no_implicit_sites<no_simd,no_simd_memory>();</no_simd,no_simd_memory></pre>		mul (81M0 mul (81M0 mul (81M0
#pragma for ( in for	<pre>omp target teams distribute parallel for sima collapse(2) si nt siteh_sf = 0; siteh_sf &lt; _lattice.half_volume_sf(); ++site   ( int isf = 0; isf &lt; no_isf; ++isf ) {</pre>	<pre>malen(no_lst) h_sf ) {</pre>	mul (81M0 mul (81M0 mul (81M0 mul (81M0 mul (81M0
	<pre>color_vector_simd&lt; float_type, no_simd &gt; v[no_rhs];</pre>		mul (81Me mul (81Me mul (81Me
	<pre>const site_shift<no_simd> site( siteh_sf, parity, isf );</no_simd></pre>		mad (81Mg mad (81Mg
	for ( int imu = $0$ ; imu < 4; ++imu ) {		mad (81Me mul (81Me mad (81Me
	<pre>const auto site_up = site.shift_eo<up>( imu, lattice ); const auto site_dn = site.shift_eo<dn>( imu, lattice );</dn></up></pre>		mul (81M6 mad (81M6 mad (81M6 mul (81M6
	<pre>su3_simd&lt; float_type, no_simd &gt; link = field.get_eo( sit</pre>	e, imu);	mul (81Me mul (81Me mad (81Me
	<pre>for ( int irhs = 0; irhs &lt; no_rhs; ++irhs ) {</pre>		mad (81M0 mad (81M0 mul (81M0 mul (81M0
	<pre>v[irhs] += link * vec_in.get( site_up, irhs ); }</pre>	no_isf matrix times vector	mul (81Me add (81Me mad (81Me
	<pre>link = field.get_eo( site_dn, imu );</pre>		add (81Me mad (81Me
	<pre>for ( int irhs = 0; irhs &lt; no_rhs; ++irhs ) {</pre>		add (81Me mad (81Me
	<pre>v[irhs] -= link * vec_in.get( site_dn, irhs );</pre>		mad (81Me mad (81Me mad (81Me
	}// end imu		mad (81Me mad (81Me
	for ( int irhs = $0$ ; irhs < no_rhs; ++irhs ) {		add (81M) add (81M) add (81M)
	<pre>vec_out.stream( 0.5*v[irhs], site, irhs ); }</pre>		add (81M0 add (81M0 add (81M0
}// }// end	end isf siteh_sf		add (81Me add (81Me

}

1	mul (81M0)	r12.0<1>:f	r73.0<8;8,1>:f	r5.0<8;8,1>:f	
	mul (81M0)	r15.0<1>:f	r67.0<8;8,1>:f	r5.0<8;8,1>:f	
	mul (81M0)	r11.0<1>:f	r79.0<8;8,1>:f	r4.0<8;8,1>:f	
	mul (81M0)	r14.0<1>:f	r67.0<8;8,1>:f	r4.0<8;8,1>:f	
	mul (81M0)	r29.0<1>:f	r81.0<8;8,1>:f	r7.0<8;8,1>:f	
	mul (81M0)	r19.0<1>:f	r75.0<8;8,1>:f	r7.0<8;8,1>:f	
	mul (81M0)	r17.0<1>:f	r69.0<8;8,1>:f	r7.0<8;8,1>:f	
	mul (81M0)	r28.0<1>:f	r81.0<8;8,1>:f	r6.0<8;8,1>:f	
	mul (81M0)	r18.0<1>:f	r75.0<8;8,1>:f	r6.0<8;8,1>:f	
	mul (81M0)	r16.0<1>:f	r69.0<8;8,1>:f	r6.0<8;8,1>:f	
	mad (81M0)	r10.0<1>:f	-r10.0<2;1>:f	r78.0<2;1>:f	r4.0<1>:f
	mad (81M0)	r29.0<1>:f	-r29.0<2;1>:f	r80.0<2;1>:f	r6.0<1>:f
	mad (81M0)	r12.0<1>:f	-r12.0<2;1>:f	r72.0<2;1>:f	r4.0<1>:f
	mul (81M0)	r27.0<1>:f	r77.0<8;8,1>:f	r3.0<8;8,1>:f	
	mad (81M0)	r19.0<1>:f	-r19.0<2;1>:f	r74.0<2;1>:f	r6.0<1>:f
	mul (81M0)	r31.0<1>:f	r71.0<8;8,1>:f	r3.0<8;8,1>:f	
	mad (81M0)	r17.0<1>:f	-r17.0<2;1>:f	r68.0<2;1>:f	r6.0<1>:f
	mad (81M0)	r15.0<1>:f	-r15.0<2;1>:f	r66.0<2;1>:f	r4.0<1>:f
	mul (81M0)	r6.0<1>:f	r65.0<8;8,1>:f	r3.0<8;8,1>:f	
	mul (81M0)	r13.0<1>:f	r73.0<8;8,1>:f	r4.0<8;8,1>:f	
	mad (81M0)	r28.0<1>:f	r28.0<2;1>:f	r80.0<2;1>:f	r7.0<1>:f
	mad (81M0)	r18.0<1>:f	r18.0<2;1>:f	r74.0<2;1>:f	r7.0<1>:f
	mad (81M0)	r16.0<1>:f	r16.0<2;1>:f	r68.0<2;1>:f	r7.0<1>:f
	mul (81M0)	r30.0<1>:f	r77.0<8;8,1>:f	r2.0<8;8,1>:f	
	mul (81M0)	r32.0<1>:f	r71.0<8;8,1>:f	r2.0<8;8,1>:f	
	add (81M0)	r10.0<1>:f	r10.0<8;8,1>:f	r29.0<8;8,1>:f	
	mad (81M0)	r27.0<1>:f	-r27.0<2;1>:f	r76.0<2;1>:f	r2.0<1>:f
	mul (81M0)	r7.0<1>:f	r65.0<8;8,1>:f	r2.0<8;8,1>:f	
	add (81M0)	r12.0<1>:f	r12.0<8;8,1>:f	r19.0<8;8,1>:f	
	mad (81M0)	r31.0<1>:f	-r31.0<2;1>:f	r70.0<2;1>:f	r2.0<1>:f
	add (81M0)	r15.0<1>:f	r15.0<8;8,1>:f	r17.0<8;8,1>:f	
	mad (81M0)	r6.0<1>:f	-r6.0<2;1>:f	r64.0<2;1>:f	r2.0<1>:f
	mad (81M0)	r11.0<1>:f	r11.0<2;1>:f	r78.0<2;1>:f	r5.0<1>:f
	mad (81M0)	r14.0<1>:f	r14.0<2;1>:f	r66.0<2;1>:f	r5.0<1>:f
	mad (81M0)	r13.0<1>:f	r13.0<2;1>:f	r72.0<2;1>:f	r5.0<1>:f
	mad (81M0)	r30.0<1>:f	r30.0<2;1>:f	r76.0<2;1>:f	r3.0<1>:f
	mad (81M0)	r32.0<1>:f	r32.0<2;1>:f	r70.0<2;1>:f	r3.0<1>:f
	mad (81M0)	r7.0<1>:f	r7.0<2;1>:f	r64.0<2;1>:†	r3.0<1>:f
	add (81M0)	r5.0<1>:f	r27.0<8;8,1>:f	r10.0<8;8,1>:f	
	ada (81M0)	r3.0<1>:f	r31.0<8;8,1>:f	r12.0<8;8,1>:f	
		r6.0<1>:t	r6.0<8;8,1>:†	r15.0<8;8,1>:f	
		r11.0<1>:f	r11.0<8;8,1>:f	r28.0<8;8,1>:f	
		r13.0<1>:f	r13.0<8;8,1>:f	r18.0<8;8,1>:f	
		r14.0<1>:t	r14.0<8;8,1>:f	r10.0<8;8,1>:f	
		r5/.0<1>:f	r5.0<8;8,1>:f	r103.0<8;8,1>:f	
		r5.0<1>:f	r5.0<8;8,1>:f	riii.0<8;8,1>:f	
	aaa (81M0)	r3.0<1>:f	r113.0<8;8,1>:f	ro.0<8;8,1>:t	

<pre>template <class float_type,="" int="" no_simd_memory=""></class></pre>						
template <pre>sparity t parity, int no rhs, int no simd&gt;</pre>						
void action float type no simd memory :: stencil( packed lattice vectors float	t type no simd memory const &ve	c in				
vota account tout_type, no_stind_memory 2.1. sceneric packed_lattice_vectors at to	te type, no_stind_memory const aver	,				
packea_lattlce_vectors <tloo< td=""><td>It_type,no_sima_memory&gt; &amp;veo</td><td>c_out ) {</td><td></td><td></td><td></td><td></td></tloo<>	It_type,no_sima_memory> &veo	c_out ) {				
constexpr int no isf = no implicit sites <no memory="" no="" simd="">():</no>		add (81M0)	r5.0<1>:f	r3.0<8:8.1>:f	r111.0<8:8.1>:f	
		add (81M0)	r3.0<1>:f	r113.0<8:8.1>:f	r6.0<8:8.1>:f	
		or $(8 M0)$	r6.0<1>:d	r8.0<8:8.1>:d	1:w	
<pre>#pragma omp target teams distribute parallel for simd collapse(2) simdle</pre>	en(no_ist)	add (81M0)	r4.0<1>:f	r30.0<8:8.1>:f	r11.0<8:8.1>:f	
<pre>for ( int siteh_sf = 0; siteh_sf &lt; _lattice.half_volume_sf(); ++siteh_sf</pre>	5) {	add (81M0)	r2.0<1>:f	r32.0<8:8.1>:f	r13.0<8:8.1>:f	
for (int isf = $0$ ; isf < no isf; ++isf) {		add (81M0)	r7.0<1>:f	r7.0<8:8.1>:f	r14.0<8:8.1>:f	
		mul (81M0)	r6.0<1>:d	r6.0<8;8,1>:d	192:w	
		add (81M0)	r50.0<1>:f	r4.0<8;8,1>:f	r104.0<8;8,1>:f	
color_vector_simd< float_type, no_simd > v[no_rhs];		add (81M0)	r4.0<1>:f	r2.0<8;8,1>:f	r112.0<8;8,1>:f	
		add (81M0)	r2.0<1>:f	r114.0<8;8,1>:f	r7.0<8;8,1>:f	
const site shift <no simd=""> site( siteh sf parity isf ).</no>		add (81M0)	r6.0<1>:d	r102.0<8;8,1>:d	r6.0<8;8,1>:d	
		add (81M0)	r7.0<1>:d	r6.0<8;8,1>:d	64:w	
		🚽 send (81M0)	r12:f r6	0xC	0x02206C02	
for ( int imu = 0; imu < 4; ++imu ) {		add (81M0)	r6.0<1>:d	r6.0<8;8,1>:d	128:w	
		send (81M0)	r10:f r7	0xC	0x02206C02	
const auto site up = site shift $e_0 < up > (imu, lattice)$ :		send (81M0)	r6:f r6	0xC	0x02206C02	
const auto site da - site shift coada (ima, lattice);		mul (81M0)	r30.0<1>:f	r75.0<8;8,1>:f	r13.0<8;8,1>:f	
const auto site_an = site.sniit_eovan>( ind, iattice ),		mul (81M0)	r28.0<1>:f	r69.0<8;8,1>:f	r13.0<8;8,1>:f	
		mul (81M0)	r29.0<1>:f	r75.0<8;8,1>:f	r12.0<8;8,1>:f	
<pre>su3_simd&lt; float_type, no_simd &gt; link = field.get_eo( site, i</pre>	.mu );	mul (81M0)	r27.0<1>:f	r69.0<8;8,1>:f	r12.0<8;8,1>:f	
		mul (81M0)	r33.0<1>:f	r81.0<8;8,1>:f	r13.0<8;8,1>:f	
for (int into $-0$ ; into $-0$ ; it is $-0$		mul (81M0)	r32.0<1>:f	r81.0<8;8,1>:f	r12.0<8;8,1>:f	
		mul (81M0)	r14.0<1>:f	r79.0<8;8,1>:f	r11.0<8;8,1>:f	
		mul (81MØ)	r16.0<1>:f	r73.0<8;8,1>:f	r11.0<8;8,1>:f	
v[irhs] += link * vec_in.get( site_up, irhs );	Load next vector iteration	mul (81MØ)	r19.0<1>:f	r67.0<8;8,1>:f	r11.0<8;8,1>:f	
}		mul (81M0)	r15.0<1>:f	r/9.0<8;8,1>:f	r10.0<8;8,1>:f	
5	<ul> <li>irbalaan is uprollad</li> </ul>	mul (81MØ)	r17.0<1>:f	r/3.0<8;8,1>:f	r10.0<8;8,1>:f	
	<ul> <li>Instoop is unrotted</li> </ul>	mul (81MØ)	r18.0<1>:f	r67.0<8;8,1>:f	r10.0<8;8,1>:f	
link = field.get_eo( site_an, imu );		maa (81M0)	r30.0<1>:f	-r30.0<2;1>:t	r74.0<2;1>:f	r12.0<1>:f
		mad (81M0)	r55.0<1>.f	11.0<0,0,1>.T	17.0<0,0,1>1	n10 0 1
for ( int irhs = 0; irhs < no_rhs; ++irhs ) {		mad (81M0)	n16.0<1>.1	-114.0<2,1>.1	n72 0-2:1>.1	n10.0<1>.1
		mad (81M0)	r10.0<1>.1	-110.0<2,1>.1	$r_{66} 0 < 2.1 < .1$	r10.0<1>.1
ulight light * yes in set site on inter ly		mad (81M0)	r28 0-1-1	$-r28 \ 0 < 2 \cdot 1 > \cdot f$	$r68 \ 0 < 2 \cdot 1 > \cdot f$	r12 0~1>.f
v[trns] -= tink * vec_th.get( site_dh, trns );			r10 0<1>.1	$r65 \ 0 < 8 \cdot 8 \ 1 > \cdot f$	$r7 0 - 8 \cdot 8 1 > \cdot f$	112.0(12.1
}		mad (81M0)	r15.0<1>17	r15 0 < 2:1 > 15	r78 0 < 2:1 > 1	r11 0<1>.f
}// end imu		mad (81M0)	r17.0<1>:f	r17.0<2:1>:f	r72.0<2:1>:f	r11.0<1>:f
		mad (81M0)	r18.0<1>:f	r18.0<2:1>:f	r66.0<2:1>:f	r11.0<1>:f
for (intinhe at inhe and the winher) [		mul (81M0)	r34.0<1>:f	r77.0<8:8.1>:f	r6.0<8:8.1>:f	
for $($ the trus = $0$ , trus < no_rus, ++trus ) {		mul (81M0)	r36.0<1>:f	r71.0<8:8.1>:f	r6.0<8:8.1>:f	
		mul (81M0)	r11.0<1>:f	r65.0<8;8,1>:f	r6.0<8;8,1>:f	
<pre>vec_out.stream( 0.5*v[irhs], site, irhs );</pre>		mad (81M0)	r35.0<1>:f	-r35.0<2;1>:f	r70.0<2;1>:f	r6.0<1>:f
3		add (81M0)	r16.0<1>:f	r16.0<8;8,1>:f	r30.0<8;8,1>:f	
1/1 and isf		add (81M0)	r19.0<1>:f	r19.0<8;8,1>:f	r28.0<8;8,1>:f	
		mad (81M0)	r10.0<1>:f	-r10.0<2;1>:f	r64.0<2;1>:f	r6.0<1>:f
}// ena siten_st		mul (81M0)	r31.0<1>:f	r77.0<8;8,1>:f	r7.0<8;8,1>:f	
}		mad (81M0)	r34.0<1>:f	r34.0<2;1>:f	r76.0<2;1>:f	r7.0<1>:f

template <class float_type,="" int="" no_simd_memory=""></class>				
template <parity int="" no_rhs,="" no_simd="" parity,="" t=""></parity>				
void action afford type no simd memory cost encil ( nacked lattice vectors afford type no simd memory cost &vec	in			
vota activitie toat_type, no_stind_memory >sterictic packed_tactive_vectorset toat_type, no_stind_memory > const avec	,			
packed_lattice_vectors <float_type,no_simd_memory> &amp;vec</float_type,no_simd_memory>	c_out ) {			
	1 1 2 2 5 6			
<pre>constexpr int no_isf = no_implicit_sites<no_simd,no_simd_memory>();</no_simd,no_simd_memory></pre>	L12056:			40
	MUL (81M0)	r4.0<1>:d	r59.0<8;8,1>:d	48:W
the same tenant terms distribute neurollal for sind callenge(2) sindler(se is()		r2.0<1>:0	r62.0<8;8,1>:0	F9.0<0;1,0>:d
#pragma omp target teams aistribute parallel for sima collapse(2) simalen(no_ist)		r10.0<1>:f	$r_{104.0<8;8,1>:t}$	0.5:T
for ( int siteh_sf = 0; siteh_sf < _lattice.half_volume_sf(); ++siteh_sf ) {		r9.0<1>:f	r103.0<8;8,1>:T	0.5:1
<pre>for ( int isf = 0; isf &lt; no_isf; ++isf ) {</pre>		r7.0<1>;f	r111.0<0;0,1>:T	0.5:1
		ro.U<1>.1	r112.0<0,0,1>.1	0.5.1
color uncher sind. Cleat time, as sind, using whele		r5.0<1>.t	r115.0<0;0,1>:T	0.5.T 2.w
color_vector_sima< float_type, ho_sima > v[ho_rns];		r4.0<1>:0	r4.0<8;8,1>:0	2:W 0 5.£
		11 0 1 .F	r114.0<0,0,1>.1	0.5.1
const site shift <no simd=""> site( siteh sf. parity, isf ):</no>		r11.0<1>.f	104.0<0;0,1>:1	0.5:1
		r12.0<1>.1	noo.0<0,0,1>.1	0.5.1 m1 0.2.2 1
		r4.0<1>.0	r2.0<0,0,1>.u	F4.0<0;0,1>:u
for ( int imu = $0$ ; imu < 4; ++imu ) {	ada (SIMO)	r5.0<1>:0	r4.0<0;0,1>:0	04.W
			1.9 UXOL	0, 5, f
const auto site up = site shift $eo < up > (imu, lattice)$ :		n10 0 1 f	100.0<0,0,1>.1	0.5.1
const quite site dn - site shift eadh (imu, lattice);	sonds (81MQ)	nullud n3	n7 0v90	0.3.1
const duto site_an = site.shirt_eokan>( tind, tattice ),	add (81M0)	n3 0-12.1	n1 0-8.8 1d	178·w
		r7 0-1-1	r88 0-8.8 1f	120.W 0 5.f
<pre>su3_simd&lt; float_type, no_simd &gt; link = field.get_eo( site, imu );</pre>		n8 0-1-1	n80 0-8.8 1	0.5.f
	sends (81MQ)	null ud r3	r5 0v80	0.J.T 0x02026C03
for (intrinte Oright on the with ) [	or (81MQ)	null.uu 15	n59 0-8.8 12.1	1.w
For $($ the trns = $\forall$ ; trns < no_rns; ++trns ) {		n3 0-12.4	n5 0-2.1 0-1.1	1.W
		r5 0-2-:ud	r3 0-8.8 12.11d	0~30.114
v[irhs] += link * vec_in.get( site_up, irhs );		r3 0~1>.a	r3 0~8.8 1~.ud	0x30.ud
3	mov (81M0)	r4 0<1>.4	r3 1~2.1 0>.d	0,50.00
		r3 0/1/1	r5.1 < 2.1	0x30.11W
	add (81M0)	r5 1-2-1d	$r4.0 - 8 \cdot 8.1 - 1 - 1$	r3 0-8.8 1.d
link = field.get_eo( site_dn, imu );	sh1 (81M0)	r4 0<1>.d	r5.0 < 2.1.0 > .d	2·w
	or $(8 M0)$	r5 0<1>.a	r59 0~8.8 1>.d	2:w
for (int interaction of the second rest thinks) $\{$		r4 0<1>.d	$r^{2} 0 < 8 \cdot 8 \rightarrow 1 > 0$	r4 0~8.8 1>.d
no_isf 3-dim complex vec	ctors add (81M0)	r3 0<1>.d	r4.0<8:8.1>:d	64:w
	sends (81M0)	null:ud r4	r11 0x8C	0x02026C03
v[irhs] -= link * vec_in.get( site_dn, irhs );	mul (8 M0)	r11.0<1>:f	r90.0<8:8.1>:f	0.5:f
}	mul (81M0)	r12.0<1>:f	r91.0<8:8.1>:f	0.5:f
1/ and imu	sends (81M0)	null:ud r3	r9 0x80	0x02026C03
Requires 3 SIMD16 st	ores add (81M0)	r3.0<1>:d	r4.0<8:8.1>:d	128:w
Requires 5 on 10 to set	mul (8 M0)	r9.0<1>:f	r92.0<8:8.1>:f	0.5:f
for ( int irhs = 0; irhs < no_rhs; ++irhs ) {	mul (81M0)	r10.0<1>:f	r93.0<8:8.1>:f	0.5:f
	sends (81M0)	null:ud r3	r7 Øx8C	0x02026C03
vec out stream( 0 5*v[irbs] site irbs ):	mov (81MØ)	r3.0<1>:ud	r5.0<2;1.0>:ud	
	mul (81M0)	r7.0<1>:f	r94.0<8:8.1>:f	0.5:f
3	mul (81MØ)	r8.0<1>:f	r95.0<8;8,1>:f	0.5:f
}// end isf	mul (81M0)	r5.0<2>:ud	r3.0<8;8,1>:ud	0x30:uw
}// end siteh sf	mul (81M0)	r3.0<1>:q	r3.0<8;8,1>:ud	0x30:ud
	mov (81M0)	r4.0<1>:d	r3.1<2;1,0>:d	
1				

# Summary

 The GPU programming style is not determined by hardware

- Intel GPU allows to program in the style that fits your application background. Both OpenMP and DPC++ support:
  - SPMD and SIMD programming model

#pragma omp simd allows to use Intel GPU like a CPU

### Notices & Disclaimers

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors.

Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit <u>www.intel.com/benchmarks</u>.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details. No product or component can be absolutely secure.

Refer to http://software.intel.com/en-us/articles/optimization-notice for more information regarding performance and optimization choices in Intel software products

Configurations details: Performance shown for HotQCD stencil kernel is using 32^3x128 lattice volume on Intel<sup>®</sup> Xeon<sup>®</sup> Processor E3-1585 v5 with Intel<sup>®</sup> Iris<sup>®</sup> Pro Graphics P580 using Intel<sup>®</sup> oneAPI Compiler beta09.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.