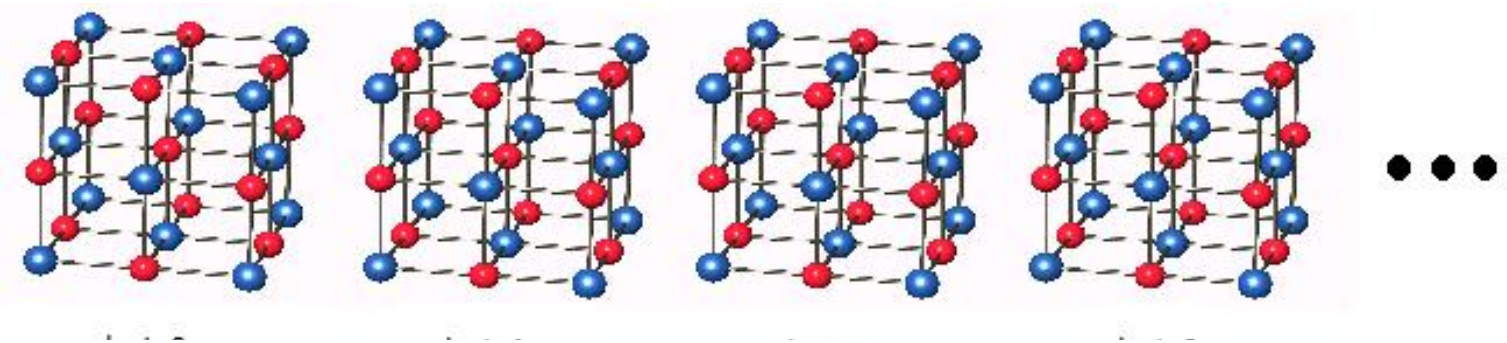
Compute body

```
for (int b = 0; b < NBETA_PER_WORD; b++) { // 91%
        1. Compute spin energy; // 48%
        2. Compute probability; // 25 %
        3. Generate random number; // 10 %
        4. Compare and accumulate to flip mask;
}
word_new = word ^ flip;
```

To reduce spin energy compute overhead



J5 J4 J3 J2 J1 J0	beta 25	• • •	beta 3	beta 2	beta 1	beta 0

J5 J4 J3 J2 J1 J0	• • •	beta 3	0	0	beta 1	0	0	beta 0

MSCT

- In previous MSCT implementation, performance drops 2x
- Global-shared memory traffic increase dramatically
- Optimize global memory access pattern

Compute body

```
for (int b = 0; b < NBETA_PER_WORD; b++) { // 91%
        1. Compute spin energy; // 48%
        2. Compute probability; // 25 %
        3. Generate random number; // 10 %
        4. Compare and accumulate to flip mask;
}
word_new = word ^ flip;
```

Compute probabilities

- exp (2 * (energy - H * spin) * beta[b])
- __exp (2 * (energy - H * spin) * beta[b])
- prob[(b << 4) + (energy << 1) + spin]

Compute body

```
for (int b = 0; b < NBETA_PER_WORD; b++) { // 91%
        1. Compute spin energy; // 48%
        2. Compute probability; // 25 %
        3. Generate random number; // 10 %
        4. Compare and accumulate to flip mask;
}
word_new = word ^ flip;
```

Handle cycle/round boundary cond

conditional generating index

```
xa = x - 1;
xb = x + 1;
if (xa == -1)
  xa = SZ - 1;
if (xb == SZ)
  xb = 0;
```

cyclic expression

```
xa = (x + SZ - 1) % SZ;
xb = (x + 1) % SZ;
```

or

```
(x + 15) ^ 15;
(x + 1) ^ 15;
```