Rodger Lee

204914719

CS33

5.19 ◆◆◆

In Problem 5.12, we were able to reduce the CPE for the prefix-sum computation to 3.00, limited by the latency of floating-point addition on this machine. Simple loop unrolling does not improve things.

Using a combination of loop unrolling and reassociation, write code for a prefix sum that achieves a CPE less than the latency of floating-point addition on your machine. Doing this requires actually increasing the number of additions performed. For example, our version with two-way unrolling requires three ad- ditions per iteration, while our version with four-way unrolling requires five. Our best implementation achieves a CPE of 1.67 on our reference machine.

Determine how the throughput and latency limits of your machine limit the minimum CPE you can achieve for the prefix-sum operation.

void psum(float a[], float p[], long n)

{

long i;

float val1, val2;

val1 = val2 = p[0] = a[0];

for(i=1; i < n - 1; i += 2) {

val1 = val1 + a[i];

val2 = val2 + (a[i] + a[i+1]);

p[i] = val1;

p[i + 1] = val2;

}

if(i < n)

{

p[i] = val2 + a[i];

}

else

{

p[i-1] = val2;

}

}

Minimum CPE is capped or bounded by the machine’s latency and throughput bounds. Unrolling and reassociation techniques can only reduce the CPE by a certain amount.