OpenCL Fast Fourier Transform

Ruobing Li

Discrete Fourier Transform

$$F(x) = \sum_{n=0}^{N-1} f(n) e^{-j2\pi(x\frac{n}{N})}$$

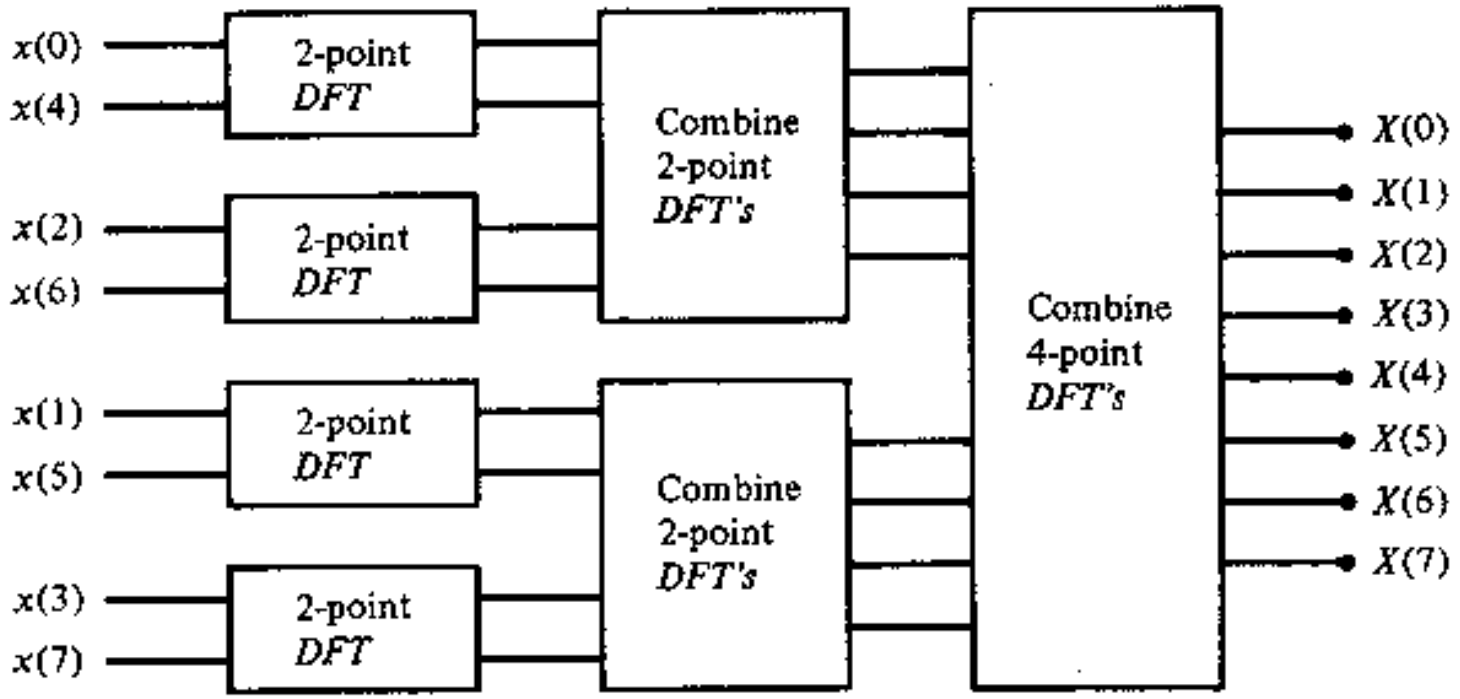
$$f(n) = \frac{1}{N} \sum_{n=0}^{N-1} F(x) e^{j2\pi(x\frac{n}{N})}$$

- Takes $O(n^2)$ with this naïve implementation.

Fast Fourier Transform

- The Cooley–Tukey algorithm is by far the most commonly used FFT algorithm.
- The idea is to build a DFT out of smaller and smaller DFTs by decomposing the input into smaller and smaller subsequences.
- Requires only $O(nlogn)$ computations to compute the N-point DFT.

Fast Fourier Transform



Stockham's FFT Algorithm

Data: input array in, output array out, size n
for $t = 1;\ t \leq logn;\ t = t + 1$ **do**
  for $j = 0;\ j \leq \frac{n}{2^t};\ j = j + 1$ **do**
    for $k = 0;\ k \leq 2^{t-1};\ k = k + 1$ **do**
      $theta = -\frac{2\pi k}{2^t}$
      $out[j2^t + k] = in[j2^{t-1} + k] + e^{i\ theta} in[j2^{t-1} + k + \frac{n}{2}]$
      $out[j2^t + k + 2^{t-1}] = in[j2^{t-1} + k] - e^{i\ theta} in[j2^{t-1} + k + \frac{n}{2}]$
    end
  end
  swap(in, out);
end

Algorithm 1: Stockham's radix-2 FFT algorithm

Implementation

- We have implemented three different kinds of kernels: radix-2, 4 and 8.
- Higher radix makes better use of private memory to processes several iterations per kernel
- Reduce the need of global communication.

Implementation

```
#define TWOPI 6.28318530718

__kernel void fft_radix2(__global float2* src, /*input array*/
                         __global float2* dst, /*output array*/
                         const int p,          /*block size*/
                         const int t) {        /*number of threads*/

    const int gid = get_global_id(0);
    const int k = gid & (p - 1);
    src += gid;
    dst += (gid << 1) - k;

    const float2 in1 = src[0];
    const float2 in2 = src[t];

    const float theta = -TWOPI * k / p;
    float cs;
    float sn = sincos(theta, &cs);
    const float2 temp = (float2) (in2.x * cs - in2.y * sn, in2.y * cs + in2.x * sn);

    dst[0] = in1 + temp;
    dst[p] = in1 - temp;
}
```

Results

	Xeon X5650	Tesla M2070
radix-2	3.38 Gflop/s	25.66 Gflop/s
radix-4	9.42 Gflop/s	39.15 Gflop/s
radix-8	11.12 Gflop/s	47.28 Gflop/s

- Performance (in Gflop/s) of different kernels on size N = 2^24

- Total number of real add+mul operations ius 5Nlog(N).

Results

	Xeon X5650	Tesla M2070
logN = 15	1.22 Gflop/s	18.76 Gflop/s
18	5.21 Gflop/s	39.19 Gflop/s
21	7.75 Gflop/s	56.53 Gflop/s
24	11.12 Gflop/s	47.28 Gflop/s
27	10.86 Gflop/s	50.05 Gflop/s

- Performance (in Gflop/s) of radix-8 kernels on different sizes (logN)