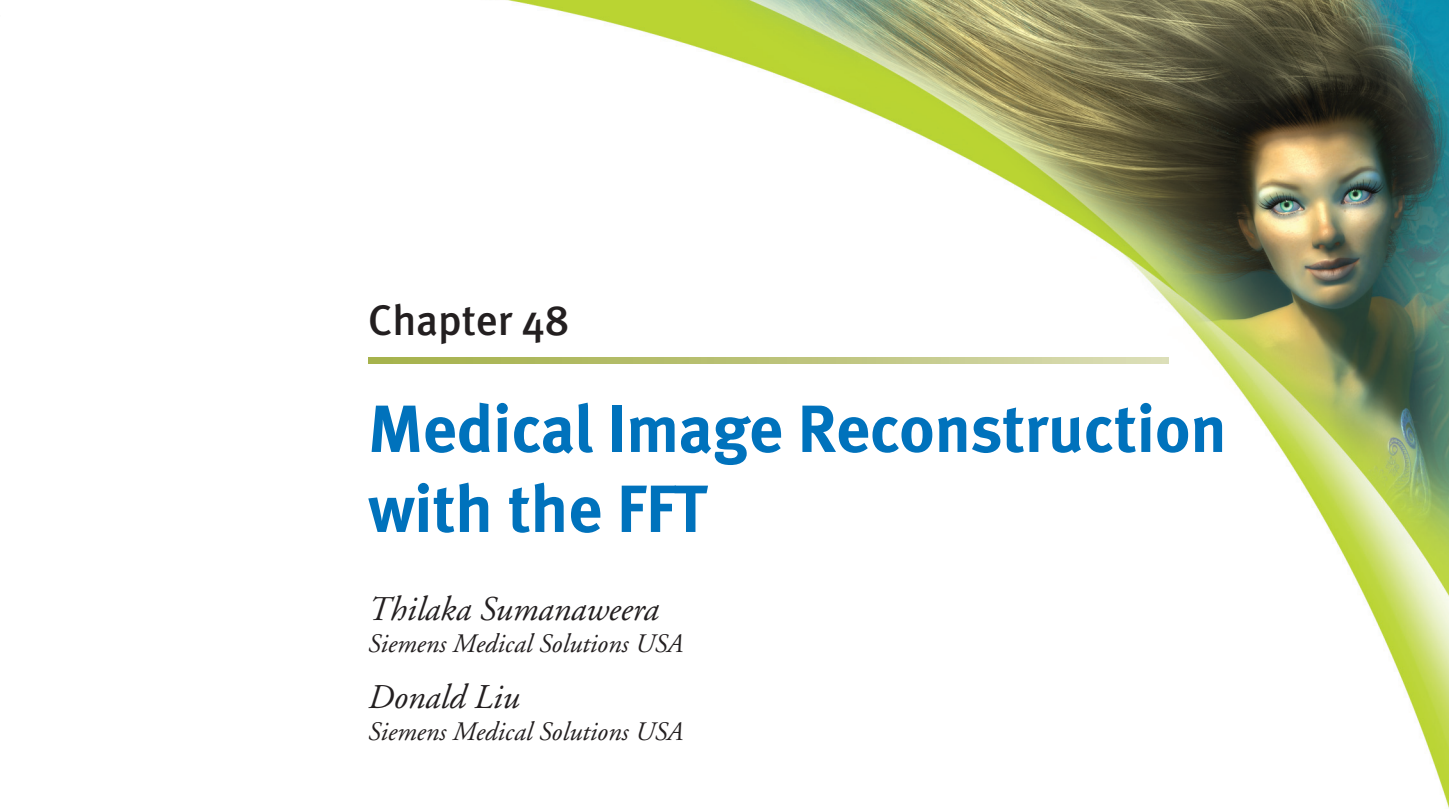


Chapter 48

Medical Image Reconstruction with the FFT

Thilaka Sumanaweera
Siemens Medical Solutions USA

Donald Liu
Siemens Medical Solutions USA

In a number of medical imaging modalities, the Fast Fourier Transform (FFT) is being used for the reconstruction of images from acquired raw data. In this chapter, we present an implementation of the FFT in a GPU performing image reconstruction in magnetic resonance imaging (MRI) and ultrasonic imaging. Our implementation automatically balances the load between the vertex processor, the rasterizer, and the fragment processor; it also uses several other novel techniques to obtain high performance on the NVIDIA Quadro NV4x family of GPUs.

48.1 Background

The FFT has been implemented in GPUs before (Ansari 2003, Spitzer 2003, Moreland and Angel 2003). The improvements presented in this chapter, however, obtain higher performance from GPUs through numerous optimizations. In medical imaging devices used in computed tomography (CT), MRI, and ultrasonic scanning, among others, specialized hardware or CPUs are used to reconstruct images from acquired raw data. GPUs have the potential for providing better performance compared to the CPUs in medical image reconstruction at a cheaper cost, thanks to economic forces resulting from the large consumer PC market.

We review the Fourier transform and the classic Cooley-Tukey algorithm for the FFT and present an implementation of the algorithm for the GPU. We then present several specific approaches that we have employed for obtaining higher performance. We also briefly review the image-formation physics of MRI and ultrasonic imaging and present results obtained by reconstructing sample cine MRI and ultrasonic images on an NVIDIA Quadro FX NV40-based GPU using the FFT.

48.2 The Fourier Transform

The Fourier transform is an important mathematical transformation that is used in many areas of science and engineering, such as telecommunications, signal processing, and digital imaging. We briefly review the Fourier transform in this section. See Bracewell 1999 for a thorough introduction to Fourier transforms.

The Fourier transform of the 1D function $f(x)$ is given by:

$$F(s) = \int_{-\infty}^{+\infty} f(x) e^{-j2\pi sx}\, dx, \tag{1}$$

and the inverse Fourier transform is given by:

$$f(x) = \int_{-\infty}^{+\infty} F(s) e^{j2\pi sx}\, ds. \tag{2}$$

The Fourier transform can also be extended to 2, 3, . . . , N dimensions. For example, the 2D Fourier transform of the function $f(x, y)$ is given by:

$$F(p,q) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x,y) e^{-j2\pi px} e^{-j2\pi qy}\, dx\, dy. \tag{3}$$

Note that the 2D Fourier transform can be carried out as two 1D Fourier transforms in sequence by first performing a 1D Fourier transform in x and then doing another 1D Fourier transform in y:

$$F(p,q) = \int_{-\infty}^{+\infty}\left[\int_{-\infty}^{+\infty} f(x,y) e^{-j2\pi px}\, dx\right] e^{-j2\pi qy}\, dy. \tag{4}$$

When processing digital signals, it is necessary to convert Equation 1 into a discrete form, turning the Fourier transform into a Discrete Fourier Transform (DFT):

$$F(k) = \sum_{n=0}^{N-1} f(n) W_N^{kn}, \tag{5}$$

where

$$W_N^{kn} = e^{-j\frac{2\pi kn}{N}}; \; f(n), \; n = 0, \ldots, N-1,$$

is a set of discrete samples of a continuous signal and $F(k)$, $k = 0, \ldots, N-1$, are the coefficients of its DFT. The inverse DFT is given by:

$$f(n) = \frac{1}{N} \sum_{k=0}^{N-1} F(k) W_N^{-kn}. \tag{6}$$

Note that both $f(n)$ and W_N^{kn} are complex numbers. When performing the DFT, $F(k)$ is to be evaluated as shown in Equation 5 for all k, requiring N^2 complex multiplications and additions. However, the most widely used algorithm for computing the DFT is the FFT algorithm, which can perform the DFT using only $N \log_2 N$ complex multiplications and additions, making it substantially faster, although it requires that N be a power of two. This algorithm is known as the Cooley-Tukey algorithm (Cooley and Tukey 1965), although it was also known to Carl Friedrich Gauss in 1805 (Gauss 1805, Heideman et al. 1984). We review this FFT algorithm in the next section.

48.3 The FFT Algorithm

For simplicity, let us consider a discrete digital signal of eight samples: $x(0), \ldots, x(7)$. Figure 48-1 shows a flow chart of the FFT algorithm (the so-called *decimation-in-time butterfly algorithm*) for computing the DFT.

On the left side of the figure, in Stage 1, the input samples are first sent through an *input scrambler* stage. If n is the index of the sample, that sample is sent to the kth output of the input scrambler, where

```
k = BIT_REVERSE(n)
```

For example, sample index 3 (that is, 011 in binary) is sent to the output index (110 binary), which is 6. Every other output of the input scrambler is then multiplied by W_N^0,

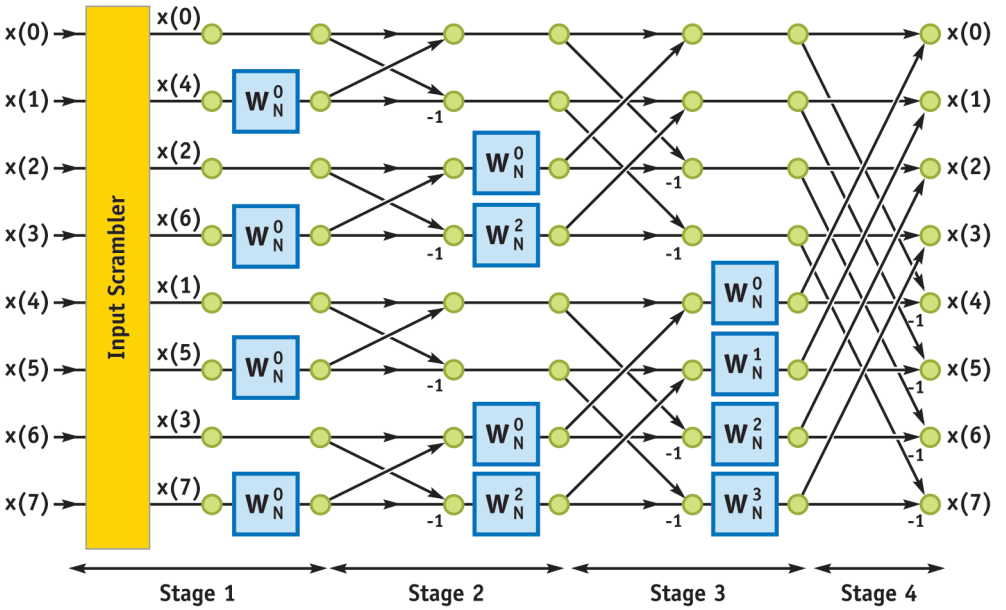


Figure 48-1. The Decimation-in-Time Butterfly Algorithm
Used for computing the DFT of a discrete digital signal of eight samples.

which has the value 1, and all outputs are sent to Stage 2. The input to Stage 2 is then scrambled again, multiplied by weights, and summed to generate the output of Stage 2. Each output of Stage 2 is generated by linearly combining two inputs. This process repeats for Stage 3 and Stage 4. The Fourier transform of the input signal is generated at the output of Stage 4.

In the more general case, if there are N input samples, there will be $\log_2 N + 1$ stages, requiring $N \log_2 N$ complex multiplications and additions. The preceding algorithm can also be applied to do 2D, 3D, and ND FFTs. For example, consider an image, a 2D array of numbers. Using Equation 4, we could do a 1D FFT across all columns first and then do another 1D FFT across all rows to generate the 2D FFT.

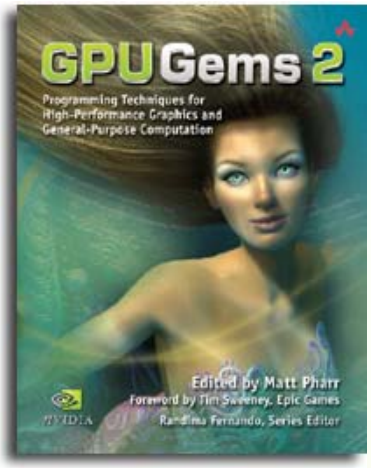
48.4 Implementation on the GPU

The FFT can be implemented as a multipass algorithm. Referring to Figure 48-1, we observe that each stage can be implemented as a single pass in a fragment program. Note also that we can combine the input scrambler in Stage 1 and the scrambler in Stage 2

The full chapter appears in *GPU Gems 2*:

GPU Gems 2

Programming Techniques for High-Performance Graphics and General-Purpose Computation



- **880 full-color pages, 330 figures**
- **Hard cover**
- **$59.99**
- **Available at GDC 2005 (March 7, 2005)**
- **Experts from universities and industry**

Graphics Programming

- **Geometric Complexity**
- **Shading, Lighting, and Shadows**
- **High-Quality Rendering**

GPGPU Programming

- **General Purpose Computation on GPUs: A Primer**
- **Image-Oriented Computing**
- **Simulation and Numerical Algorithms**

For more information, please visit:

http://developer.nvidia.com/object/gpu_gems_2_home.html