CSE565M: Acceleration of Algorithms in Reconfigurable Logic

Learn by Doing: DFT (Pt. 1)

Anthony Cabrera

FL24::L07

Washington University in St. Louis

Table of contents

Overview

Core Idea

An algorithm that changes a discrete signal in the time domain to the same signal in the frequency domain.

- By describing the signal as the sum of sinusoids, we can more easily compute some functions on the signal. As such, it plays an important role in many domains
  - image processing
  - wireless communications

This lecture will contain mostly some of the mathematical foundations for DFTs with some thoughts about efficient HW implementations.

Fourier Series

Mathematical Formulation

The Fourier series proves an alternative way to look at a real valued, continuous, periodic signal where the signal runs from $[-\pi, \pi]$.

The Fourier series represents any continuous, periodic signal over a period of 2π can be represented by a sum of cosines and sines with a period of 2π.

$$\begin{aligned} f(t) &\sim \tfrac{a_0}{2} + a_1\cos(t) + a_2\cos(2t) + a_3\cos(3t) + \dots \\ &\quad + b_1\sin(t) + b_2\sin(2t) + b_3\sin(3t) + \dots \\ &\sim \tfrac{a_0}{2} + \sum_{n=1}^{\infty}(a_n\cos(nt) + b_n\sin(nt)) \end{aligned} \tag{1}$$

where the coefficients $a_0, a_1, \dots$ and $b_1, b_2, \dots$ are computed as

$$\begin{aligned} a_0 &= \tfrac{1}{\pi}\int_{-\pi}^{\pi} f(t)\,\mathrm{d}t \\ a_n &= \tfrac{1}{\pi}\int_{-\pi}^{\pi} f(t)\cos(nt)\,\mathrm{d}t \\ b_n &= \tfrac{1}{\pi}\int_{-\pi}^{\pi} f(t)\sin(nt)\,\mathrm{d}t \end{aligned} \tag{2}$$

Mathematical Formulation (cont.)

- The coefficients $a_0, a_1, a_2, \ldots, b_1, b_2, \ldots$ in Equation 2 are called the Fourier coefficients.
- The coefficient a_0 is often called the direct current (DC) term, the $n = 1$ frequency is called the fundamental, while the other frequencies ($n \geq 2$) are called higher harmonics.
- The function f, and the cos() and sin() functions all have a period of 2π

Mathematical Formulation (cont.)

- The coefficients $a_0, a_1, a_2, \ldots, b_1, b_2, \ldots$ in Equation 2 are called the Fourier coefficients.
- The coefficient a_0 is often called the direct current (DC) term, the $n = 1$ frequency is called the fundamental, while the other frequencies ($n \geq 2$) are called higher harmonics.
- The function f, and the cos() and sin() functions all have a period of 2π

Mathematical Formulation (cont.)

Representing other functions not periodic on π requires a change in variables. Assume a function is periodic on $[-L, L]$ rather than $[-\pi, \pi]$. Let

$$t \equiv \frac{\pi t'}{L} \tag{3}$$

and

$$\mathrm{d}t = \frac{\pi \mathrm{d}t'}{L} \tag{4}$$

which is a simple linear translation. Solving for t' and substituting $t' = \frac{Lt}{\pi}$ into Equation 1 gives

$$f(t') = \frac{a_0}{2} + \sum_{n=1}^{\infty} (a_n \cos(\frac{n\pi t'}{L}) + b_n \sin(\frac{n\pi t'}{L})) \tag{5}$$

Solving for the a and b coefficients is similar:

$$\begin{aligned} a_0 &= \frac{1}{L} \int_{-L}^{L} f(t') \, \mathrm{d}t' \\ a_n &= \frac{1}{L} \int_{-L}^{L} f(t') \cos(\frac{n\pi t'}{L}) \, \mathrm{d}t' \\ b_n &= \frac{1}{L} \int_{-L}^{L} f(t') \sin(\frac{n\pi t'}{L}) \, \mathrm{d}t' \end{aligned} \tag{6}$$

Mathematical Formulation (cont.)

We can use Euler's formula $e^{jnt} = \cos(nt) + j\sin(nt)$ to give a more concise formulation

$$f(t) = \sum_{n=-\infty}^{\infty} c_n e^{jnt}. \tag{7}$$

In this case, the Fourier coefficients c_n are a complex exponential given by

$$c_n = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(t) e^{-jnt} \mathrm{d}t \tag{8}$$

which assumes that $f(t)$ is a periodic function with a period of 2π, i.e., this equation is equivalent to Equation 1.

DFT Background

Discrete Fourier Transform (DFT)

- The Discrete Fourier Transform requires *discrete* periodic signals.
- The DFT converts a finite number of equally spaced samples into a finite number of complex sinusoids. In other words, it converts a sampled function from one domain (most often the time domain) to the frequency domain.
- The frequencies of the complex sinusoids are integer multiples of the fundamental frequency which is defined as the frequency related to the sampling period of the input function.
- Perhaps the most important consequence of the discrete and periodic signal is that **it can be represented by a finite set of numbers**. Thus, a digital system can be used to implement the dft.

DFT (cont)

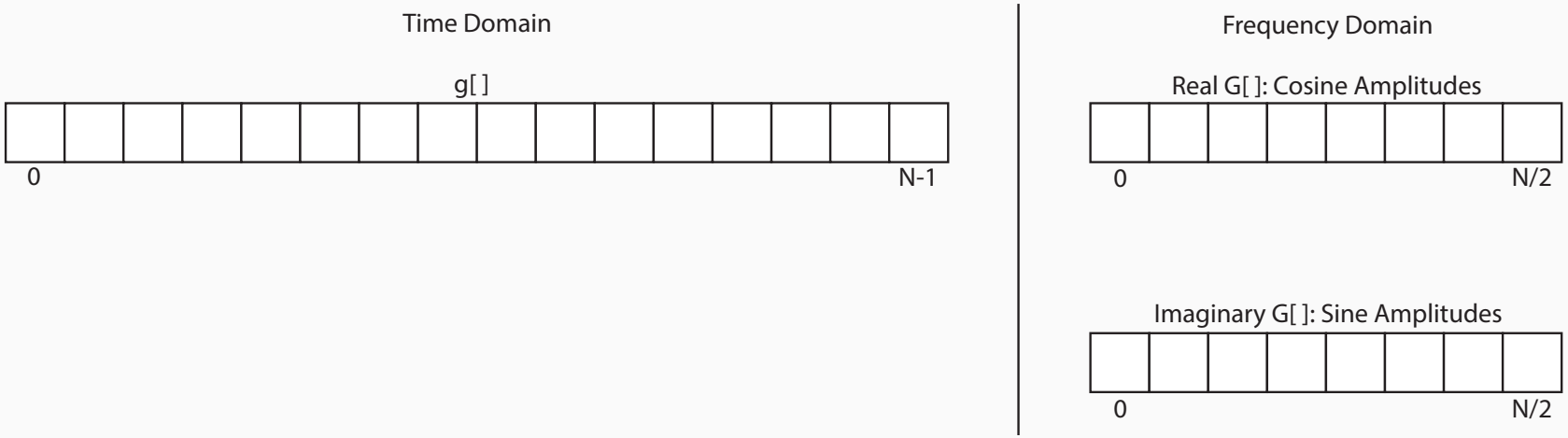

Figure 1: A real valued discrete function $g[]$ in the time domain with N points has a frequency domain representation with $N/2 + 1$ samples. Each of these frequency domain samples has one cosine and one sine amplitude value. Collectively these two amplitude values can be represented by a complex number with the cosine amplitude representing the real part and the sine amplitude the imaginary part.

DFT (cont)

An N point dft can be determined through a $N \times N$ matrix multiplied by a vector of size N, $G = S \cdot g$ where

$$S = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & s & s^2 & \cdots & s^{N-1} \\ 1 & s^2 & s^4 & \cdots & s^{2(N-1)} \\ 1 & s^3 & s^6 & \cdots & s^{3(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & s^{N-1} & s^{2(N-1)} & \cdots & s^{(N-1)(N-1)} \end{bmatrix} \tag{9}$$

and $s = e^{\frac{-j2\pi}{N}}$. Thus the samples in frequency domain are derived as

$$G[k] = \sum_{n=0}^{N-1} g[n] s^{kn} \text{ for } k = 0, \ldots, N-1 \tag{10}$$

Visualize with an 8-point DFT

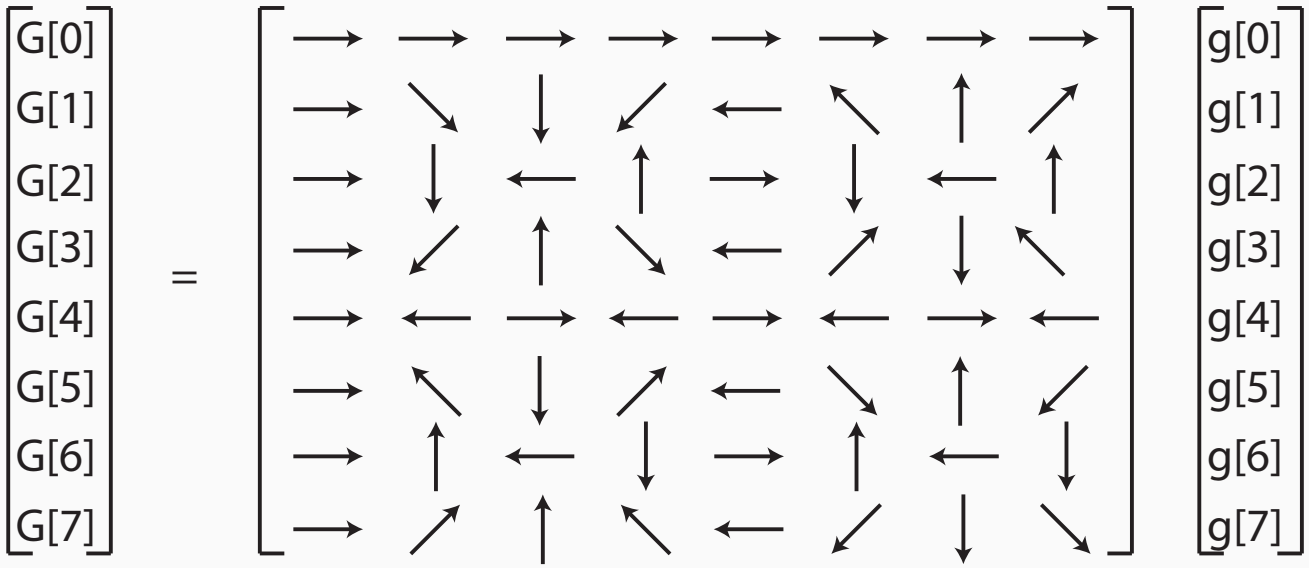

Figure 2: The elements of the S shown as a complex vectors.

By taking advantage of symmetry, we can just focus on the first $N/2$ frequency domain values.

Visualize with an 8-point DFT (cont)

Figure 2 provides a visualization of the dft coefficients for an 8 point dft operation. The eight frequency domain samples are derived by multiplying the 8 time domain samples with the corresponding rows of the S matrix. Row 0 of the S matrix corresponds to the DC component which is proportional to the average of the time domain samples. Multiplying Row 1 of the S matrix with g provides the cosine and sine amplitudes values for when there is one rotation around the unit circle. Since this is an 8 point dft, this means that each phasor is offset by 45°. Performing eight 45° rotations does one full rotation around the unit circle. Row 2 is similar except it performs two complete rotations around the unit circle, i.e., each individual rotation is 90°. This corresponds to a higher frequency. Row 3 does three complete rotations; Row 4 four rotations and so on. Each of these row times column multiplications gives the appropriate frequency domain sample.

Matrix Vector Multiplication Optimizations

Baseline Code

```
#define SIZE 8
typedef int BaseType;

void matrix_vector(BaseType M[SIZE][SIZE], BaseType V_In[SIZE],
     BaseType V_Out[SIZE]) {
  BaseType i, j;
data_loop:
  for (i = 0; i < SIZE; i++) {
    BaseType sum = 0;
  dot_product_loop:
    for (j = 0; j < SIZE; j++) {
      sum += V_In[j] * M[i][j];
    }
    V_Out[i] = sum;
  }
}
```

Figure 3: Simple code implementing a matrix-vector multiplication.

This relatively simple code has many design choices that can be performed when mapping to hardware.

Design Choices

Memory organization is one of the more important decisions. The question boils down to *where do you store the data from your code?* There are a number of options when mapping variables to hardware. The variable could simply be a set of wires (if its value never needs saved across a cycle), a register, RAM or FIFO. All of these options provide tradeoffs between performance and area.

Another major factor is the amount of parallelism that is available within the code. Purely sequential code has few options for implementation. On the other hand, code with a significant amount of parallelism has implementation options that range from purely sequentially to fully parallel.

Design Choices

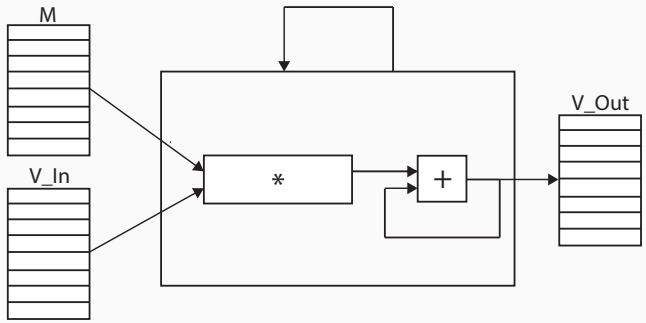

Figure 4: A possible implementation of matrix-vector multiplication from the code in Figure 3.

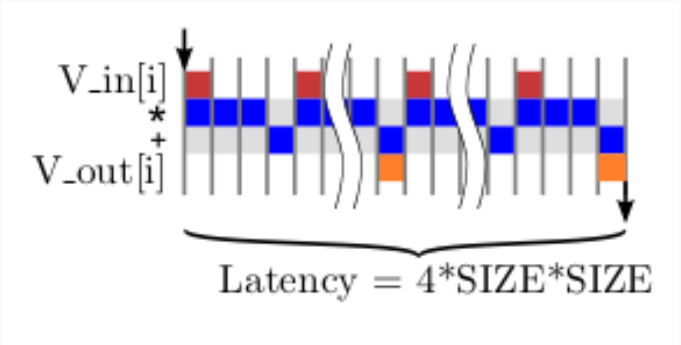

Figure 5: Resulting schedule. Does not consume a lot of area, but the task latency and task interval are relatively large.

References i