DSP INTEGRATED CIRCUITS
STUDY NOTES
UNIT I
DIGITAL SIGNAL PROCESSING

The sampling theorem:

Due to the increased use of computers in all engineering applications, including signal processing, it is important to spend some more time examining issues of sampling. In this chapter we will look at sampling both in the time domain and the frequency domain.

We have already encountered the sampling theorem and, arguing purely from a trigonometric-identity point of view have established the Nyquist sampling criterion for sinusoidal signals. However, we have not fully addressed the sampling of more general signals, nor provided a general proof. Nor have we indicated how to reconstruct a signal from its samples. With the tools of Fourier transforms and Fourier series available to us we are now ready to finish the job that was started months ago.

To begin with, suppose we have a signal $x(t)$ which we wish to sample. Let us suppose further that the signal is band limited to $B$ Hz. This means that its Fourier transform is nonzero for $-2\pi B < \omega < 2\pi B$. Plot spectrum.

We will model the sampling process as multiplication of $x(t)$ by the “picket fence” function.

We encountered this periodic function when we studied Fourier series. Recall that by its Fourier series representation we can write...
\[ \delta_T(t) = \frac{1}{T} \sum_n e^{in\omega_s t} \]

where \( \omega_s = \frac{2\pi}{T} \). The frequency \( f_s = \omega_s/(2\pi) = 1/T \) is the sampling frequency in samples/sec. Suppose that the sampling frequency is chosen so that \( f_s > 2B \), or equivalently, \( \omega_s > 4\pi B \).

The sampled output is denoted as \( x(t) \), where

\[ \widehat{x}(t) = x(t)\delta_T(t) \]

Using the F.S. representation we get

\[ \widehat{x}(t) = x(t) \frac{1}{T} \sum_n e^{in\omega_s t} \]

Now let's look at the spectrum of the transformed signal. Using the convolution property,

\[ \overline{X}(\omega) = \frac{1}{2\pi} \frac{1}{T} X(\omega) * \sum_n 2\pi \delta(\omega - n\omega_s) = \frac{1}{T} \sum_n X(\omega - n\omega_s). \]

Plot the spectrum of the sampled signal with both \( \omega \) frequency and \( f \) frequency. Observe the following

The spectrum is periodic, with period \( 2 \), because of the multiple copies of the spectrum.

The spectrum is scaled down by a factor of \( 1/T \).

Note that in this case there is no overlap between the images of the spectrum.

Now consider the effect of reducing the sampling rate to \( f_s < 2B \). In this case, the duplicates of the spectrum overlap each other. The overlap of the spectrum is aliasing.

This demonstration more-or-less proves the sampling theorem for general signals. Provided that we sample fast enough, the signal spectrum is not distorted by the sampling process. If we don’t sample fast enough, there will be distortion. The next question is: given a set of samples, how do we get the signal back? From the spectrum, the answer is to filter the signal with a low-pass filter with cutoff \( \omega_c \geq 2\pi B \). This cuts out the images and leaves us with the original spectrum. This is a sort of idealized point of view, because it assumes that we are filtering a continuous-time function \( x(t) \), which is a sequence of weighted delta functions. In practice, we have numbers \( x[n] \) representing the value of the function \( x[n] = x(nT) = x(n/f_s) \). How can we recover the time function from this?
Theorem 1 (The sampling theorem)

If \( x(t) \) is band limited to \( B \) Hz then it can be recovered from signals taken at a sampling rate \( f_s > 2B \). The recovery formula is

\[
x(t) = \sum_n x(nT)g(t - nT)
\]

where

\[
g(t) = \frac{\sin(\pi f_s t)}{\pi f_s t} = \text{sinc}(\pi f_s t).
\]

Show what the formula means: we are interpolating in time between samples using the sinc function.

We will prove this theorem. Because we are actually lacking a few theoretical tools, it will take a bit of work. What makes this interesting is we will end up using in a very essential way most of the transform ideas we have talked about.

1. The first step is to notice that the spectrum of the sampled signal,

\[
\overline{X}(\omega) = \frac{1}{T} \sum_n X(\omega - n\omega_s)
\]

is periodic and hence has a Fourier series. The period of the function in frequency is \( \omega_s \), and the fundamental frequency is

\[
p_0 = \frac{2\pi}{\omega_s} = \frac{1}{f_s} = T.
\]

By the F.S. we can write

\[
\overline{X}(\omega) = \sum c_ne^{jn\omega T}
\]

where the \( c_n \) are the F.S. coefficients

\[
c_n = \frac{1}{\omega_s} \int_{\omega_s} \overline{X}(\omega)e^{-jn\omega T}d\omega = \frac{2\pi}{\omega_s} \frac{1}{2\pi} \int_{-\infty}^{\infty} \frac{1}{T} X(\omega)e^{-jn\omega T}d\omega.
\]
But the integral is just the inverse F.T., evaluated at \( t = -nT \):

\[
c_n = \frac{1}{T} \frac{2\pi}{\omega_s} x(t) \bigg|_{t=-nT} = x(-nT),
\]

so

\[
\mathcal{X}(\omega) = \sum_n x(-nt)e^{j\omega T} = \sum_n x(nt)e^{-j\omega T}.
\]

2. Let \( g(t) = \text{sinc}(\pi f_s t) \). Then

\[
g(t) \leftrightarrow T \text{rect} \left( \frac{\omega}{2\pi f_s} \right).
\]

3. Let

\[
y(t) = \sum_n x(nT)g(t-nT).
\]

We will show that \( y(t) = x(t) \) by showing that \( Y(\omega) = X(\omega) \). We can compute the F.T. of \( y(t) \) using linearity and the shifting property:

\[
Y(\omega) = \sum_n x(nT)T \text{rect} \left( \frac{\omega}{2\pi f_s} \right) e^{-j\omega nT} = T \text{rect} \left( \frac{\omega}{2\pi f_s} \right) \sum_n x(nT)e^{-j\omega nT}
\]

Observe that the summation on the right is the same as the F.S. we derived in step 1:

\[
Y(\omega) = T \text{rect} \left( \frac{\omega}{2\pi f_s} \right) \mathcal{X}(\omega).
\]

Now substituting in the spectrum of the sampled signal (derived above)

\[
Y(\omega) = T \text{rect} \left( \frac{\omega}{2\pi f_s} \right) \left( \frac{1}{T} \sum_n X(\omega - n\omega_s) \right) = X(\omega)
\]

since \( x(t) \) is bandlimited to \(-\pi f_s < \omega < \pi f_s \) or \(-f_s/2 < f < f_s/2\).

Notice that the reconstruction filter is based upon a sinc function, whose trans-form is a rect function: we are really just doing the filtering implied by our initial intuition.

In practice, of course, we want to sample at a frequency higher than just twice the bandwidth to allow room for filter roll off.

**Adaptive DSP algorithms:**

**DFT-The Discrete Fourier Transform:**

**Introduction**

The sampled discrete-time fourier transform (DTFT) of a finite length, discrete-time signal is known as the discrete Fourier transform (DFT). The DFT contains a finite number of samples equal to the number of samples \( N \) in the given signal. Computationally efficient algorithms for implementing the DFT go by the generic name of fast Fourier transforms (FFTs). This chapter describes the DFT and its properties, and its relationship to DTFT.
Definition of DFT and its Inverse

Lest us consider a discrete time signal $x(n)$ having a finite duration, say in the range $0 \leq n \leq N-1$. The DTFT of the signal is

$$X(\omega) = \sum_{n=0}^{N-1} x(n)e^{-j\omega n}$$

Equation (0.2) is known as $N$-point DFT analysis equation. Fig 0.1 shows the Fourier transform of a discrete – time signal and its DFT samples.
While working with DFT, it is customary to introduce a complex quantity
\[ w_N = e^{j2\pi/N} \]
Also, it is very common to represent the DFT operation
\[ X(k) = \text{DFT}(x(n)) = \sum_{n=0}^{N-1} x(n) w_N^{kn}, \quad 0 \leq n \leq N-1 \]

The complex quantity \( W_n \) is periodic with a period equal to \( N \). That is,
\[ w_N^{a+N} = e^{j2\pi/N(a+N)} = e^{j2\pi/N} n = w_N^a \text{ where } a \text{ is any integer.} \]

Figs. 0.2(a) and (b) shows the sequence for \( 0 \leq n \leq N-1 \) in the \( z \)-plane for \( N \) being even and odd respectively.

\[ w_N^{kn}, \quad 0 \leq k \leq N-1 \]
Inverse DFT

The DFT values \( X(k), 0 \leq k \leq N-1 \), uniquely define the sequence \( x(n) \) through the inverse DFT formula (IDFT):

\[
x(n) = \text{IDFT}(X(k)) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) \text{ } W_N^k
\]

The above equation is known as the synthesis equation.
Proof: \[ \sum_{k=0}^{N-1} X(k) W_{N}^{-kn} = \frac{1}{N} \left[ \sum_{k=0}^{N-1} x(m) W_{N}^{km} \right] = \sum_{k=0}^{N-1} x(m) \left[ \sum_{m=0}^{N-1} W_{N}^{(n-m)k} \right] \]

It can be shown that \[ \sum_{k=0}^{N-1} W_{N}^{(n-m)k} = \begin{cases} N, & n = m \\ 0, & n \neq m \end{cases} \]

Hence,
\[ \frac{1}{N} \sum_{m=0}^{N-1} x(m) \delta(n-m) = x(n) \]

( sifting property)

**FFT-FAST FOURIER TRANSFORM:**
The FFT is a basic algorithm underlying much of signal processing, image processing, and data compression. When we all start interfacing with our computers by talking to them (not too long from now), the first phase of any speech recognition algorithm will be to digitize our speech into a vector of numbers, and then to take an FFT of the resulting vector. So one could argue that the FFT might become one of the most commonly executed algorithms on most computers.

There are a number of ways to understand what the FFT is doing, and eventually we will use all of them:

- The FFT can be described as multiplying an input vector \( x \) of \( n \) numbers by a particular \( n \)-by-\( n \) matrix \( F_n \), called the DFT matrix (Discrete Fourier Transform), to get an output vector \( y \) of \( n \) numbers: \( y = F_n \cdot x \). This is the simplest way to describe the FFT, and shows that a straightforward implementation with 2 nested loops would cost \( 2n^2 \) operations. The importance of the FFT is that it performs this matrix-vector in just \( O(n \log n) \) steps using divide-and-conquer. Furthermore, it is possible to compute \( x \) from \( y \), i.e. compute \( x = F_n^{-1} y \), using nearly the same algorithm, and just as fast. Practical uses of the FFT require both multiplying by \( F_n \) and \( F_n^{-1} \).
The FFT can also be described as evaluating a polynomial with coefficients in $x$ at a special set of $n$ points, to get $n$ polynomial values in $y$. We will use this polynomial-evaluation-interpretation to derive our $O(n \log n)$ algorithm. The inverse operation in called *interpolation*: given the values of the polynomial $y$, find its coefficients $x$.

To pursue the signal processing interpretation mentioned above, imagine sitting down at a piano and playing a chord, a set of notes. Each note has a characteristic frequency (middle A is 440 cycles per second, for example). A microphone digitizing this sound will produce a sequence of numbers that represent this set of notes, by measuring the air pressure on the microphone at evenly spaced sampling times $t_1, t_2, \ldots, t_i$, where $t_i = i \cdot \Delta t$. $\Delta t$ is the interval between consecutive samples, and $1/\Delta t$ is called the sampling frequency. If there were only the single, pure middle A frequency, then the sequence of numbers representing air pressure would form a sine curve, $x_i = d \cdot \sin(2 \cdot \pi \cdot t_i \cdot 440)$. To be concrete, suppose $1/\Delta t = 45056$ per second (or 45056 Hertz), which is a reasonable Sampling frequency for sound. The scalar $d$ is the maximum amplitude of the curve, which depends on how loud the sound is; for the pictures below we take $d = 1$. Then a plot of $x_i$ would look like the following (the horizontal axis goes from $i = 0$ ($t_0 = 0$) to $i = 511$ ($t_i \approx .011$ seconds) so that $2 \cdot \pi \cdot t_i \cdot 440$ goes from 0 to $\approx 10\pi$, and the sine curve has about 5 full periodic cycles:

![Plot of x(i) = sin(2*pi*t(i)*440), t(i) = i/45056](image)

The next plots show the absolute values of the entries of $y = F_n \cdot x$. In the plot on the left, you see that nearly all the components are about zero, except for two near $i=0$ and $i=512$. To see better, the plot on the right blows up the picture near $i=0$ (the other end of the graph is symmetric):
We will see that the fact that only one value of $y_i$ is large for $i$ between 0 and 256 means that $x$ consists of a sine wave of a single frequency, and since the large value is $y_6$ then that frequency is one that consists of $5=6-1$ full periodic cycles in $x_i$ as $i$ runs from 0 to 512.

To see how the FFT might be used in practice, suppose that the signal consists not just of the pure note A (frequency 440), but also an A one octave higher (i.e. with twice the frequency) and half as loud: $x_i = \sin(2\pi t, 440) + .5 \sin(2\pi t, 880)$. Both $x_i$ and its FFT (in the range $i = 0$ to 20) are shown below. The larger nonzero component of the FFT corresponds to our original signal $\sin(2\pi t, 440)$, and the second one corresponds to the sine curve of twice the frequency and half the amplitude.
But suppose that there is “noise” in the room where the microphone is recording the piano, which makes the signal $x$ “noisy” as shown below to the left (call it $x_\text{noisy}$). (In this example we added a random number between $-.5$ and $.5$ to each $x_i$ to get $x_\text{noisy}$.) $x$ and $x_\text{noisy}$ appear very different, so that it would seem difficult to recover the true $x$ from noisy $x_\text{noisy}$. But by examining the FFT of $x_\text{noisy}$ below, it is clear that the signal still mostly consists of two sine waves. By simply zeroing out small components of the FFT of $x_\text{noisy}$ (those below the “noise level”) we can recover the FFT of our two pure sine curves, and then get nearly the original signal $x(i)$ back. This is a very simple example of filtering, and is discussed further in other EE courses.
Definition of the DFT Matrix:

We next need to define the primitive $n$-th root of unity

$$\omega = e^{2\pi in} = \cos \frac{2\pi}{n} + i \cdot \sin \frac{2\pi}{n}$$

It is called an $n$-th root of unity because

$$\omega^n = e^{(2\pi in)n} = e^{2\pi i} = \cos 2\pi + i \cdot \sin 2\pi = 1 + i \cdot 0 = 1$$

Note that all other integer powers $\omega^j$ are also $n$-th roots of unity; since they are all powers of $\omega$, $\omega$ is called primitive.

Finally, we can define the $n$-by-$n$ DFT matrix $F_n$. We will denote its entries by $f_{jk}(n)$ or just $f_{jk}$ if $n$ is understood from context. To make notation easier later we will let the subscripts $j$ and $k$ run from 0 to $n-1$ instead of from 1 to $n$. Then

$$f(n) = \omega_{jk} = e^{\frac{2\pi ijk}{n}}$$

Here is an important formula that shows that the inverse of the DFT matrix is very simple:

**Lemma.** Let $G_n = \frac{1}{n} F_n$, that is

$$G_{jk} = \frac{1}{n} \omega_{jk}$$

Then $G_n$ is the inverse matrix of $F_n$. In other words, if $y = F_n \cdot x$, then $x = G_n \cdot y$.

**Proof:** We compute the $j, k$-th component of the matrix product $G_n \cdot F_n$, and confirm that it equals 1 if $j = k$ and 0 otherwise.

$$G_{n-1} = \sum_{r=0}^{n-1} g_{jr} f_{rk}$$

$$n-1$$

$$r=0$$

$$n-1$$
N-1 Similarly, \( x(n) \)  \( \sum_{k=0}^{N-1} X(k) W_N^{-kn} \)

\[ \Rightarrow x(n+N) = \sum_{k=0}^{N-1} X(k) W_N^{k(n+N)} \]

Since, \( W_N^{-kn} = e^{j2\pi N kn} = e^{j2\pi k} = 1 \), we get

\[ x(n+N) = 1 \sum_{k=0}^{N-1} X(k) W_N^{-kn} = x(n) \]

Since, DFT and its inverse are both periodic with period \( N \), it is sufficient to compute the results for one period (0 to \( N-1 \)). We want to emphasize that both \( x(n) \) and \( X(k) \) have a starting index of zero.

A very important implication of \( x(n) \), being periodic is, if we wish to find DFT of a periodic signal, we extract one period of the periodic signal and then compute its DFT.

Interpreting \( y \equiv F_n \cdot x \) as polynomial evaluation:

By examining the formula for \( y \) in \( y = F_n \cdot x \) we see

\[ n-1 \quad n-1 \]
\[ y_j = -f_{jk} \cdot x_k = -(\omega^j)^k \cdot x_k \quad \text{k=0} \quad \text{k=0} \]

so that \( y_j \) is just the value of the polynomial \( p(z) \) at \( z = \omega^j \), where

\[
p(z) = -z^k \cdot x_k \quad \text{k=0}^{n-1} 
\]

In other words, the \( x_k \) are just the coefficients of the polynomial \( p(z) \), from the constant term \( x_0 \) to the highest order term \( x_{n-1} \). Note that the degree of \( p(z) \), the highest power of \( z \) that appears in \( p(z) \), is at most \( n - 1 \) (if \( x_{n-1} \neq 0 \)).

By our Lemma in the last section \( x = n^{-1} F_n \cdot y \) gives a formula for finding the coefficients \( x_j \) of the polynomial \( p(z) \) given its values at the \( n \)-th roots of unity \( \omega^k \), for \( 0 \leq k \leq n - 1 \). Computing polynomial coefficients given polynomial values is called interpolation.

**Implementation of FFT algorithms:**

**Introduction:** The \( N \) point Discrete Fourier Transform (DFT) of \( x(n) \) is a discrete signal of length \( N \) is given by

\[
X(k) = \sum_{n=0}^{N-1} x(n)W_N^{kn} \quad ; \quad k = 0 \ldots N - 1 
\]

\[
W_N^{kn} = e^{-j2\pi kn/N} \quad \text{is the twiddle factor} 
\]

The Inverse DFT (IDFT) is given by eq(2)

\[
x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k)W_N^{-kn} \quad ; \quad n = 0 \ldots N - 1 
\]

By referring to above equations the difference between DFT & IDFT are seen to be the sign of the argument for the exponent and multiplication factor, \( 1/N \). The computational complexity in computing DFT / IDFT is thus same (except for the additional multiplication factor in IDFT). The computational complexity in computing each \( X(k) \) and all the \( x(k) \) is shown in table
In a typical Signal Processing System, shown in fig 6.1 signal is processed using DSP in the DFT domain. After processing, IDFT is taken to get the signal in its original domain. Though certain amount of time is required for forward and inverse transform, it is because of the advantages of transformed domain manipulation, the signal processing is carried out in DFT domain. The transformed domain manipulations are sometimes simpler. They are also more useful and powerful than time domain manipulation. For example, convolution in time domain requires one of the signals to be folded, shifted and multiplied by another signal, cumulatively. Instead, when the signals to be convolved are transformed to DFT domain, the two DFT are multiplied and inverse transform is taken. Thus, it simplifies the process of convolution.

This is a SAMPLE (Few pages have been extracted from the complete notes:-It’s meant to
show you the topics covered in the full notes and as per the course outline

Download more at our websites: www.naarocom.com

To get the complete notes either in softcopy form or in Hardcopy (printed & Binded) form, contact us on:

Call/text/WhatsApp  +254 719754141/734000520

Email:

naarocom@gmail.com
info@naarocom.com
sales@naarocom.com

Get news and updates by liking our page on facebook and follow us on Twitter

Sample/preview is NOT FOR SALE