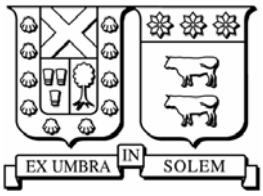

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA

LECTURA 5
TRANSFORMADA RÁPIDA DE FOURIER FFT

CURSO	LABORATORIO DE PROCESAMIENTO DIGITAL DE SEÑALES
SIGLA	ELO–385
PROFESOR	PABLO LEZANA ILLESCA
AYUDANTE	JORGE MURATT RODRÍGUEZ

Valparaíso, de 20 de Mayo de 2005

Introducción

El análisis frecuencial de señales en tiempo discreto se realiza, normalmente, de forma más conveniente en procesadores de señales digitales que pueden ser computadores de propósito general (PC's) o hardware especialmente diseñado. Para realizar el análisis frecuencial de una señal discreta es necesario convertir la secuencia $\{x(n)\}$ en el dominio del tiempo en una secuencia equivalente en el dominio de la frecuencia.

La *Transformada Discreta de Fourier* (DFT) juega un papel importante en numerosas aplicaciones de procesamiento de señales digitales. Entre otras destacan: filtrado lineal, análisis de correlación y análisis espectral. Una de las razones de la importancia de la DFT es la existencia de algoritmos sumamente eficientes para su cálculo. Uno de los más utilizados es el denominado Radix 2, el cual se basa en el hecho de analizar una secuencia que posee un número tamaño N que es una potencia de 2. Existe un algoritmo denominado Radix 4, cuyo nombre deriva de la misma razón del Radix 2. Estos tipos de algoritmos permiten calcular en forma rápida y sumamente eficiente la DFT. Se denominan FFT por la sigla en inglés de *Fast Fourier Transform*.

Como veremos más adelante, la DFT requiere de grandes cantidades de operaciones matemáticas para lograr el mismo resultado que los algoritmos FFT. En particular, el cálculo de la FFT (con el algoritmo Radix 2) usando 1024 puntos puede mejorar la rapidez hasta en alrededor de 200 veces la velocidad de calculo directo la DFT.

En las siguientes secciones se muestra el algoritmo FFT radix 2 que utiliza el criterio de Decimación en Frecuencia, el cual se basa en la descomposición sucesiva de la entrada en subsecuencias cada vez más pequeñas.

1 TRANSFORMADA DISCRETA DE FOURIER – DFT

Básicamente, el problema de cálculo de la DFT es obtener la secuencia $\{X(n)\}$ de longitud N según la fórmula dada en (1):

$$X(k) = \sum_{n=0}^{N-1} x(n) W^{nk} \tag{1}$$

donde $N = 2^a$, a=1,2,3,…

El factor $W = e^{\frac{-j2\pi}{N}} = \cos\left(\frac{2\pi}{N}\right) - j\sin\left(\frac{2\pi}{N}\right)$, denominado Twiddle Constant.

Note que:

$$W^n = \cos\left(\frac{2\pi}{N}n\right) - j\sin\left(\frac{2\pi}{N}n\right) \tag{2}$$

Con lo anterior, el cálculo de un punto de la transformada discreta de fourier está dada por:

$$X(k) = x(0)W^0 + x(1)W^k + x(2)W^{2k} \ldots + x(N-1)W^{k(N-1)}, \text{ k=0,1,…,N-1} \tag{3}$$

Desarrollando (3) para los N valores posibles de k se obtiene una matriz de tamaño NxN. De (3) se puede calcular el número de operaciones necesarias para realizar la transformación de los datos mediante este algoritmo. El número de sumas complejas que se deben realizar es de (N-1)N y la cantidad de multiplicaciones complejas asciende a N^2. Es claro que ésta cantidad de operaciones es alta y requiere de un enorme poder de cálculo.

El cálculo directo de la DFT no es eficiente debido, fundamentalmente, a que no explota las propiedades de simetría y periodicidad del factor de fase W_N.

De la observación de (3) es claro que no es necesario realizar las N_2 multiplicaciones ya que los valores de los factores $W_0 = 1$ no son necesarios de multiplicar. Además existen propiedades de periodicidad y simetría en estos factores de forma tal que:

$$W^{k+N} = W^k \tag{4}$$

$$W^{k+\frac{N}{2}} = -W^k \tag{5}$$

La simetría y periodicidad de los factores W quedan de manifiesto en la figura siguiente. El ejemplo es para $N = 8$.
Note además que la DFT es obtenida al multiplicar los datos de entrada por una cantidad finita de senoidales de frecuencia $2\pi n/N$, por lo que DFT's de N puntos entregan N componentes espectrales equiespaciadas cada $2\pi/N$, donde 2π representa la frecuencia de muestreo f_s.

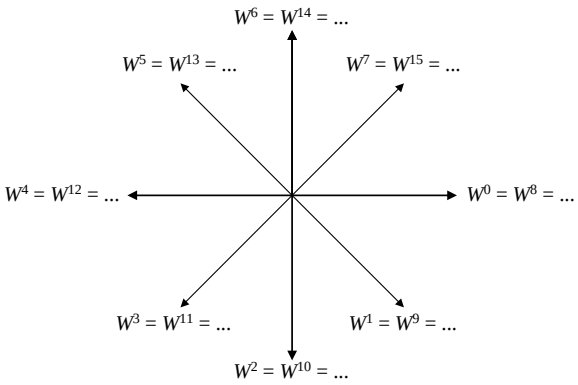

Figura 1: Periodicidad y simetría de los factores *W*.

A partir del algoritmo DFT y las consideraciones anteriores es posible llegar a un método de cálculo mucho más eficiente, que entrega los mismos resultados y con un número menor de operaciones. Es el llamado algoritmo de Transformada Rápida de Fourier (FFT, Fast Fourier Transform).

1.1 DESARROLLO DEL ALGORITMO FFT RADIX – 2. DECIMACIÓN EN FRECUENCIA.

Sea la secuencia de datos $\{x(n)\}$, con n: [0 : N-1]. Separando $\{x(n)\}$ en dos secuencias para n's pares e impares es posible aplicar la DFT a la secuencia de datos $\{x(n)\}$, quedando de la siguiente forma:

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} x(n)W^{nk} + \sum_{n=\frac{N}{2}}^{N-1} x(n)W^{nk} \tag{6}$$

Si se reemplaza $n=u+N/2$ en la segunda sumatoria de (6) queda:

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} x(n)W^{nk} + \sum_{u=0}^{\frac{N}{2}-1} x\left(u+\frac{N}{2}\right)W^{\left(u+\frac{N}{2}\right)k} \tag{7}$$

donde $W^{\frac{N}{2}k}$ sale de la sumatoria por no estar en función de x. Además reemplazado en (7) $u=n$:

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} x(n)W^{nk} + W^{k\frac{N}{2}} \sum_{n=0}^{\frac{N}{2}-1} x\left(n+\frac{N}{2}\right)W^{nk} \tag{8}$$

Usando:

$$\begin{aligned} W^{k\frac{N}{2}} &= \cos\left(\frac{2\pi}{N}\frac{N}{2}k\right) - j\sin\left(\frac{2\pi}{N}\frac{N}{2}k\right) \\ &= \cos(\pi k) = -1^k \end{aligned} \tag{9}$$

Por lo que (8) queda de la forma:

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} \left[x(n) + (-1)^k x\left(n + \frac{n}{2} \right) \right] W^{nk} \tag{10}$$

La ecuación (10) puede ser separada en sumatorias para valores de *k* pares e impares:

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} \left[x(n) + x\left(n + \frac{n}{2} \right) \right] W^{nk} \text{ , para } k \text{ par} \tag{11}$$

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} \left[x(n) - x\left(n + \frac{n}{2} \right) \right] W^{nk} \text{ , para } k \text{ impar} \tag{12}$$

Si se reemplaza $k=2k'$ en (11) y $k=2k'+1$ en (12) pueden ser reescritas como:

$$X(2k') = \sum_{n=0}^{\frac{N}{2}-1} \left[x(n) + x\left(n + \frac{n}{2} \right) \right] W^{n2k'} \text{ , para } k \text{ par} \tag{13}$$

$$X(2k'+1) = \sum_{n=0}^{\frac{N}{2}-1} \left[x(n) - x\left(n + \frac{n}{2} \right) \right] W^{n2k'} W^{n} \text{ , para } k \text{ impar} \tag{14}$$

Para facilitar la escritura y aprovechando que los factores *W* son funciones de período *N*, los factores pueden ser reescritos de la forma siguiente: W_N. Luego W_N^2 puede ser representado como $W_{N/2}$.

Sean:

$$a(n) = x(n) + x(n + N/2) \tag{15}$$

$$b(n) = x(n) - x(n + N/2) \tag{16}$$

Las ecuaciones (15) y (16) pueden ser escritas más claramente como 2 DFT de *N*/2 puntos,

$$X(2k') = \sum_{n=0}^{\frac{N}{2}-1} a(n) W^{n2k'} \tag{17}$$

$$X(2k'+1) = \sum_{n=0}^{\frac{N}{2}-1} b(n) W^{n} W^{n2k'} \tag{18}$$

Note que tanto (17) como (18) poseen la misma forma que (1), pero con sumatorias de N/2 puntos cada una.

La figura 2 muestra la descomposición de una DFT de *N* puntos en dos DFT de *N*/2 puntos para el caso de $N = 8$. Aplicando (17) y (18) es posible llegar a obtener los *X*'s de la primera etapa de la FFT.

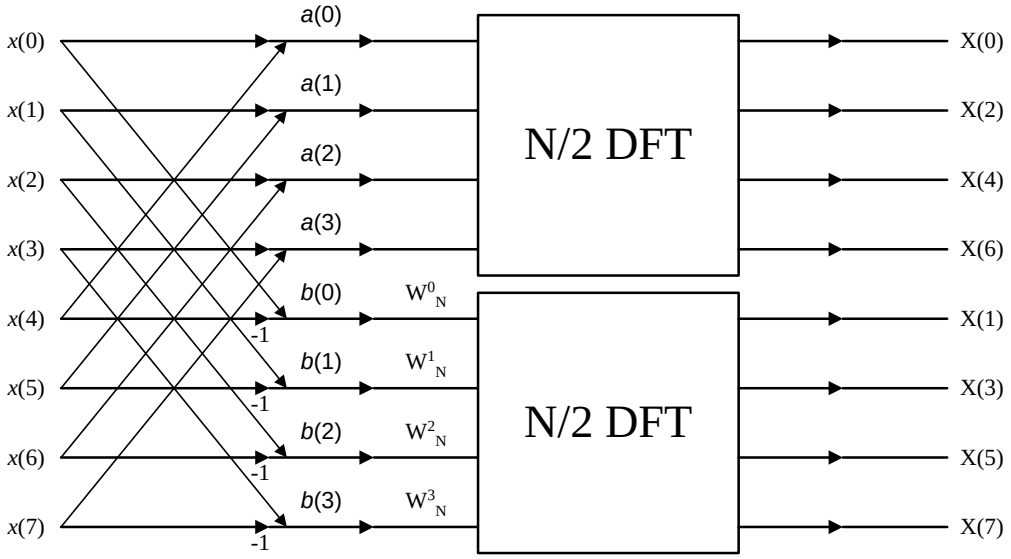

Figura 2: Descomposición de una DFT de *N* puntos en 2 DFTs de *N*/2 puntos, para *N* = 8.

Como resultado del proceso de descomposición, los *X*s quedan ordenados en grupos de pares e impares. El proceso de descomposición puede ser repetido nuevamente pero esta vez para *N*/4, que para el caso de 8 puntos, es la etapa final. El número de etapas, o DFTs, se deberá repetir hasta llegar a la DFT de 2 puntos. En general una FFT de N puntos tendrá *a* etapas, con $N=2^a$

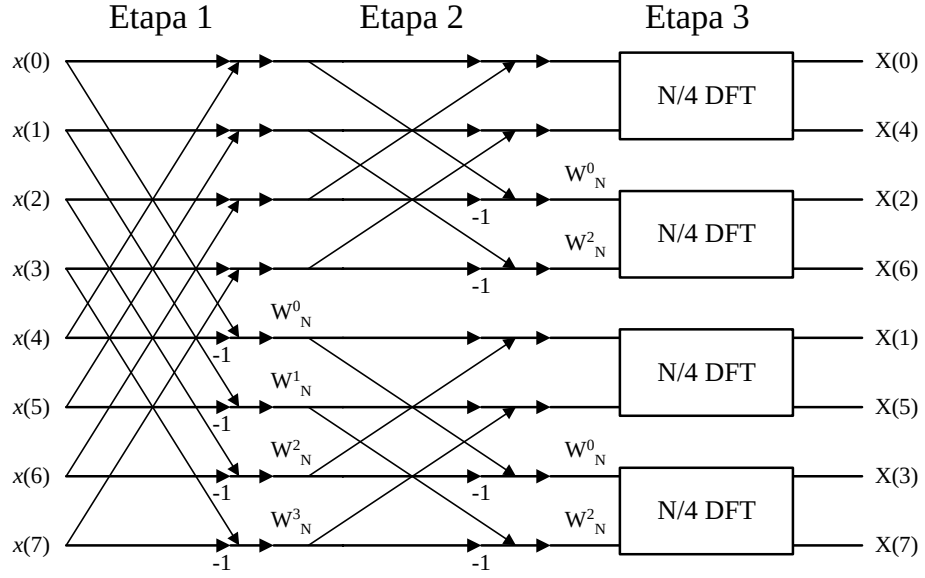

Figura 3: Descomposición de una DFT de *N*/2 puntos en 4 DFTs de *N*/4 puntos, para *N* = 8.

En la figura 3 es posible ver que la salida del algoritmo está desordenada. Esto se debe a que en las etapas que componen el algoritmo cada vez se realiza un agrupamiento de los datos almacenados en posiciones pares de memoria y los almacenados en posiciones impares. Por esta razón la salida necesita ser reordenada. El proceso que deja correctamente ordenados los datos de salida se denomina *bit-reverse* y será explicado más adelante.

La última descomposición, ya que se ha llegado a aplicar la DFT de 2 puntos, es la más baja descomposición del algoritmo Radix 2. Una DFT de 2 puntos las salidas *X*(n) pueden ser escritas como se observa en (19) y (20)

$$X(0) = x(0)W^0 + x(1)W^0 = x(0) + x(1) \tag{19}$$

$$X(1) = x(0)W^0 + x(1)W^1 = x(0) - x(1) \tag{20}$$

Las ecuaciones (16) y (17) pueden ser representadas en un diagrama de flujo conocido como "*Mariposa*". Este es el diagrama de flujo de la etapa final de todo algoritmo FFT que utiliza la decimación en frecuencia.

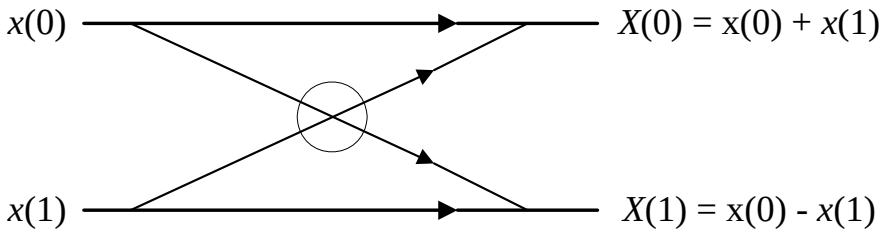

Figura 4: Diagrama de flujo de una FFT de 2 puntos - *Mariposa*

La decimación en Frecuencia adquiere su nombre del hecho de que la secuencia de salida *X(k)* es descompuesta (decimada) en subsecuencias más pequeñas, continuando por *a* etapas o iteraciones. La salida *X(k)* posee componentes tanto reales como imaginarios, y el algoritmo FFT se puede acomodar a entradas tanto reales como complejas. La FFT no es una aproximación de la DFT, sino que es un algoritmo más eficiente que se vuelve más y más importante a medida que *N* aumenta. El diagrama final del algoritmo Radix 2, con decimación en frecuencia es mostrado en la figura 5.

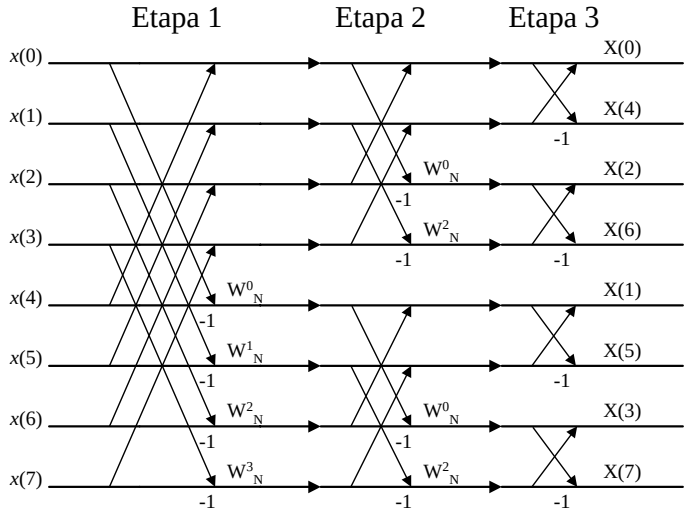

Figura 5: Diagrama de flujo de cálculo de una FFT de 8 puntos usando decimación en frecuencia

Una alternativa al diagrama de flujo mostrado en la figura 5 puede ser obtenida con entradas desordenadas (aplicando bit reverse a la secuencia de entrada) y salidas ya ordenadas.

1.1.1 Un ejemplo completo para el caso de $N = 8$

Sea la secuencia de entrada $x(0) = x(1) = x(2) = x(3) = 1$ y $x(4) = x(5) = x(6) = x(7) = 0$. Se encontrará su secuencia $X(k)$, $k = 1, 2, ...,7$.

Para N = 8, los coeficientes W pueden ser calculados sólo una vez y luego almacenados para ocuparlos luego durante la aplicación del algoritmo. No resulta eficiente generalizar la función que los calcula e incluirla dentro de las iteraciones de cálculo. Los valores son:

$$W^0 = 1$$
$$W^1 = e^{-j2\pi/8} = \cos(\pi/4) - j\sin(\pi/4) = 0.707 - j0.707$$
$$W^2 = e^{-j2\pi/4} = \cos(\pi/2) - j\sin(\pi/2) = -j$$
$$W^3 = e^{-j2\pi 3/8} = \cos(3\pi/4) - j\sin(3\pi/4) = -0.707 - j0.707$$

Las salidas intermedias de la secuencia de salida pueden ser obtenidas para cada etapa:

Etapa 1:

$$\begin{aligned}
&x(0) + x(4) = 1 && \rightarrow x'(0)\\
&x(1) + x(5) = 1 && \rightarrow x'(1)\\
&x(2) + x(6) = 1 && \rightarrow x'(2)\\
&x(3) + x(7) = 1 && \rightarrow x'(3)\\
&[x(0) - x(4)]W^0 = 1 && \rightarrow x'(4)\\
&[x(1) - x(5)]W^1 = 0.707 - 0.707j && \rightarrow x'(5)\\
&[x(2) - x(6)]W^2 = -j && \rightarrow x'(6)\\
&[x(3) - x(7)]W^3 = -0.707 - 0.707j && \rightarrow x'(7)
\end{aligned}$$

Aquí $x'(0)$, $x'(1)$,..., $x'(7)$ representan las salidas intermedias luego de la primera iteración. Ahora éstas son utilizadas como entradas para la siguiente etapa.

Etapa 2:

$$\begin{aligned}
&x'(0) + x'(2) = 2 && \rightarrow x''(0)\\
&x'(1) + x'(3) = 2 && \rightarrow x''(1)\\
&[x'(0) - x'(2)]W^0 = 0 && \rightarrow x''(2)\\
&[x'(1) - x'(3)]W^2 = 0 && \rightarrow x''(3)\\
&x'(4) + x'(6) = 1 - j && \rightarrow x''(4)\\
&x'(5) + x'(7) = -1.414j && \rightarrow x''(5)\\
&[x'(4) - x'(6)]W^0 = 1 + j && \rightarrow x''(6)\\
&[x'(5) - x'(7)]W^2 = -1.414j && \rightarrow x''(7)
\end{aligned}$$

La secuencia *x''*(0), *x''*(1),..., *x''*(7) representan las salidas intermedias luego de la segunda iteración. Nuevamente son utilizadas como entradas para la etapa final:

$$\begin{aligned}
X(0) &= x''(0) + x''(1) && = 4 \\
X(4) &= \left[x''(0) - x''(1)\right]W^0 && = 0 \\
X(2) &= x''(2) + x''(3) && = 0 \\
X(6) &= \left[x''(2) - x''(3)\right]W^0 && = 0 \\
X(1) &= x''(4) + x''(5) && = 1 - 2.414\,j \\
X(5) &= \left[x''(4) - x''(5)\right]W^0 && = 1 + 0.414\,j \\
X(3) &= x''(6) + x''(7) && = 1 - 0.414\,j \\
X(7) &= \left[x''(6) - x''(7)\right]W^0 && = 1 + 2.41\,j
\end{aligned}$$

Ahora la notación $X(k)$ representa la salida final del algoritmo. Los valores $X(0)$, $X(1)$,..., $X(7)$ forman la secuencia de salida ordenada.
Un ejemplo completo para el caso de una FFT de 16 puntos utilizando el algoritmo Radix 2 se presenta a continuación en la figura 6. La secuencia de entrada corresponde a un escalón temporal.

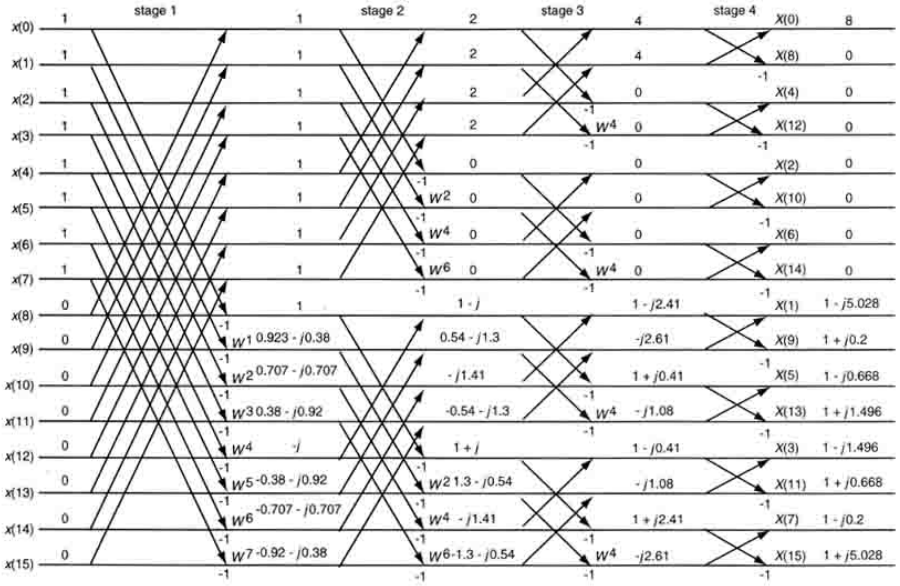

Figura 6: FFT de 16 puntos utilizando decimación en frecuencia.

3 ALGORITMO BIT REVERSAL PARA ORDENAMIENTO DE DATOS

Este algoritmo permite que una secuencia de entrada o salida del cálculo de la FFT sea reordenada para obtener un resultado deseado. Como se vio en el caso de DIF, la salida es la que se desea reordenar.

Para ilustrar este algoritmo se verá el caso de $N = 8$, representado por tres bits. El bit 3 y el 1 deben ser intercambiados de sus posiciones. Ejemplo, (100)b queda (001)b. Esto hace que el dato que estaba en la posición de memoria 4 = (100)b pase a la posición 1 = (001)b. Similarmente, (110)b es intercambiado por (011)b.

Lo anterior se resume gráficamente en la siguiente figura:

RESULTADO DEL BIT REVERSE PARA EL CASO DE 3 BIT FFT DE 8 PUNTOS

$n_2\ n_1\ n_0$	$n_0\ n_1\ n_2$
000	000
001	100
010	010
011	110
100	001
101	101
110	011
111	111

Otra forma de realizar el Bit Reverse es crear una tabla con las ubicaciones finales de los coeficientes y ordenarlos mediante el uso de punteros. El siguiente código crea dicha tabla para transformadas de *N* puntos:

```
p=1;
for (q=0;q<N;q++)
        bit_rev[q]=q;
while(p<N)
        {
        for (q=0;q<p;q++)
                {
                bit_rev[q]=bit_rev[q]*2;
                bit_rev[q+p]=bit_rev[q]+1;
                }
        p=p*2;
        }
```

Este cálculo es independiente de los valores de los *N* puntos por lo que puede realizarse fuera de línea utilizándose solamente su resultado.

Una vez obtenido el resultado de la FFT en la variable ff_temp, es necesario aplicarle el algoritmo Bit Reverse para reordenarlo, lo que se obtiene fácilmente con:

```
for (i=0;i<N;i++)
        fft_out[bit_rev[i]]=fft_temp[i]; //fft_out FFT ordenada.
```