Capítulo 5

Adecuación e Implementación de los Algoritmos de la Transformada de Fourier Rápida.

La esencia de la transformada de Fourier de una señal con forma de onda es descomponerla o separarla en una suma de sinusoidales de diferentes frecuencias. Si la suma de estas sinusoidales da como resultado la forma de onda original, se ha determinado la Transformada de Fourier de la onda. La representación gráfica de la Transformada de Fourier es un diagrama en el que se muestra la amplitud y la frecuencia de cada una de las sinusoidales.[13]

La Transformada de Fourier Discreta (*Discrete Fourier Transform, DFT*) puede ser empleada para obtener resultados esencialmente equivalentes a los de la transformada de Fourier continua. La DFT es una de las operaciones más importantes en el procesamiento digital de señales. Su implementación usando el algoritmo de la Transformada de Fourier Rápida (*Fast Fourier Transform, FFT*) ha hecho posible su uso en aplicaciones para procesamiento de señales en tiempo real.

La DFT de una señal $x(k)$ en tiempo discreto se define como

$$X(n) = \sum_{k=0}^{N-1} x(k) W_N^{nk} \qquad n = 0, 1, ..., N-1 \tag{5.1}$$

donde $W_N^{nk} = e^{-j2\pi nk/N}$ constituye las funciones complejas base o factores de fase de la DFT.

Para definir el factor de fase se puede tomar una secuencia de $N = 8$ muestras y así representarlo como un vector en el círculo unitario.

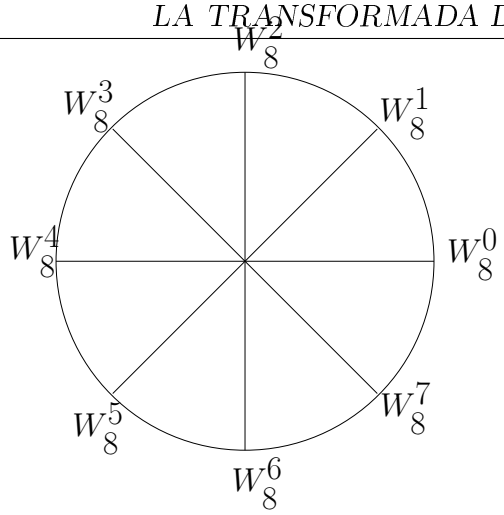

Figura 5.1: Representación en círculo unitario del factor de fase.

Se puede apreciar en la figura 5.1 que los vectores son:

1. Periódicos
2. Simétricos
3. Igualmente espaciados al rededor del círculo con espacios iguales a ΔW

Donde

$$\Delta W = \frac{2\pi}{N} = \frac{\Omega}{N}$$

Ω Es la frecuencia de muestreo, N es el número de muestras y ΔW es llamado frecuencia de resolución o espacios de salida de la DFT.

El mapeo del factor de fase en el círculo unitario es periódico. La frecuencia normalizada de muestreo es 2π, entonces la DFT resulta periódica con respecto a la frecuencia de muestreo.

El factor de fase es inversamente simétrico con respecto al origen.

$$W_8^1 = -W_8^5 \quad ; \quad W_8^2 = -W_8^6$$

Esto quiere decir que solamente la primera mitad del factor de fase contiene toda la información necesaria.[14]

Dado $X(n)$ que es una secuencia de valores frecuenciales, la Transformada de Fourier Discreta Inversa (*Inverse Discrete Fourier Transform, IDFT*) da la secuencia temporal

$$x(k) = \frac{1}{N} \sum_{n=0}^{N-1} X(n) W_N^{-nk} \tag{5.2}$$

La forma de la ecuación de la IDFT es idéntica a la forma de la DFT salvo el factor de normalización $1/N$ y el signo del exponente del factor de fase.

5.1. La Transformada de Fourier Rápida.

Casi siempre aplicaciones de análisis espectral requieren de DFTs en tiempo real para muestras de entrada continuas. Al ver la importancia de la DFT en varias aplicaciones del procesamiento digital de señales, como filtrado lineal, análisis de correlación y análisis espectral, su calculo eficiente es un tema que ha recibido considerable atención por matemáticos, ingenieros y científicos especializados.

La FFT es un algoritmo rápido para la implementación eficiente de la DFT, en donde un número N de muestras temporales se transforman en N muestras frecuenciales.

El calculo de la DFT para N muestras de entrada requiere N^2 multiplicaciones complejas y $N^2 - N$ sumas complejas. Lo cual significa un número elevado de operaciones, así como de datos almacenados. Los algoritmos de la FFT factoriza una DFT con gran cantidad de puntos en varias DFTs con pocos puntos para si poder reducir el número de operaciones.

Existen dos familias principales:

- Radix
- Algoritmos de Factor Primo (*Prime Factor Algorithm, PFA*)

Por ser los más utilizados en los sistemas de modulación OFDM para redes de comunicación inalámbrica WiMAX se revisarán los algoritmos de la familia Radix.

5.2. Radix-2.

5.2.1. Decimación en el tiempo (DIT).

El algoritmo Radix-2 con decimación en el tiempo (*Decimation In Time, DIT*) es el más simple y la forma más común del algoritmo, también es llamado algoritmo Cooley-Tukey. El algoritmo Radix-2 DIT divide la DFT de longitud N en dos DFTs de longitud $N/2$.

El algoritmo Radix-2 DIT primero calcula la transformada de Fourier de las componentes pares $x(2k)$ y de las componentes impares $x(2k+1)$, después combina los dos resultados para producir la transformada de Fourier de la secuencia completa. Al dividir

la ecuación (5.1) en componentes pares e impares se obtiene:

$$X(n) = \sum_{k=0}^{(N/2)-1} x(2k)W_N^{2kn} + \sum_{k=0}^{(N/2)-1} x(2k+1)W_N^{(2k+1)n}$$

Si $x_1(k) = x(2k)$ y $x_2(k) = x(2k+1)$ donde $k = 0, 1, ..., (N/2)-1$ se puede reescribir la ecuación anterior como:

$$X(n) = \sum_{k=0}^{(N/2)-1} x_1(k)W_{N/2}^{kn} + W_N^n \sum_{k=0}^{(N/2)-1} x_2(k)W_{N/2}^{kn}$$

donde $W_N^{2k} = W_{N/2}^k$. En su forma general quedaría de la siguiente manera:

$$\begin{aligned} X(n) &= X_1(n) + W_N^n X_2(n) \\ X(n+N/2) &= X_1(n) - W_N^n X_2(n) \end{aligned} \tag{5.3}$$

La ecuación (5.3) se conoce como la mariposa de la FFT Radix-2 DIT[15]. En la figura 5.2 se muestra gráficamente la mariposa.

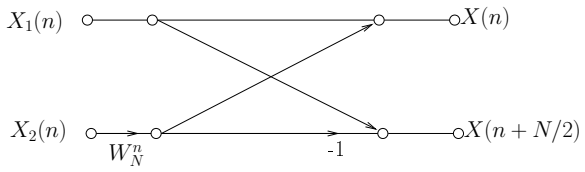

Figura 5.2: Representación de la mariposa del algoritmo de la FFT Radix-2 DIT.

De esta primera descomposición se obtienen 2 DFTs de longitud $N/2$. Los datos de entrada a la mariposa se encuentran separados por $N/2$ muestras y los exponentes de los factores de fase son consecutivos. Si a cada una de las DFTs de longitud $N/2$ se le aplica la misma descomposición obtenemos de cada una de ellas 2 DFTs de longitud $N/4$. Los datos de entrada ahora se encuentran separados por $N/4$ muestras y los exponentes de los factores de fase se separan por un factor de 2.

El proceso de descomposición se repite hasta generar DFTs de 2 puntos. Cada descomposición es conocida como etapa y el número total de etapas está dado por

$$M = log_2 N$$

De esta manera, una DFT de 16 muestras requiere de cuatro etapas. En la figura 5.3 se muestra la gráfica de flujo del algoritmo FFT Radix-2 para una señal de 16 muestras.

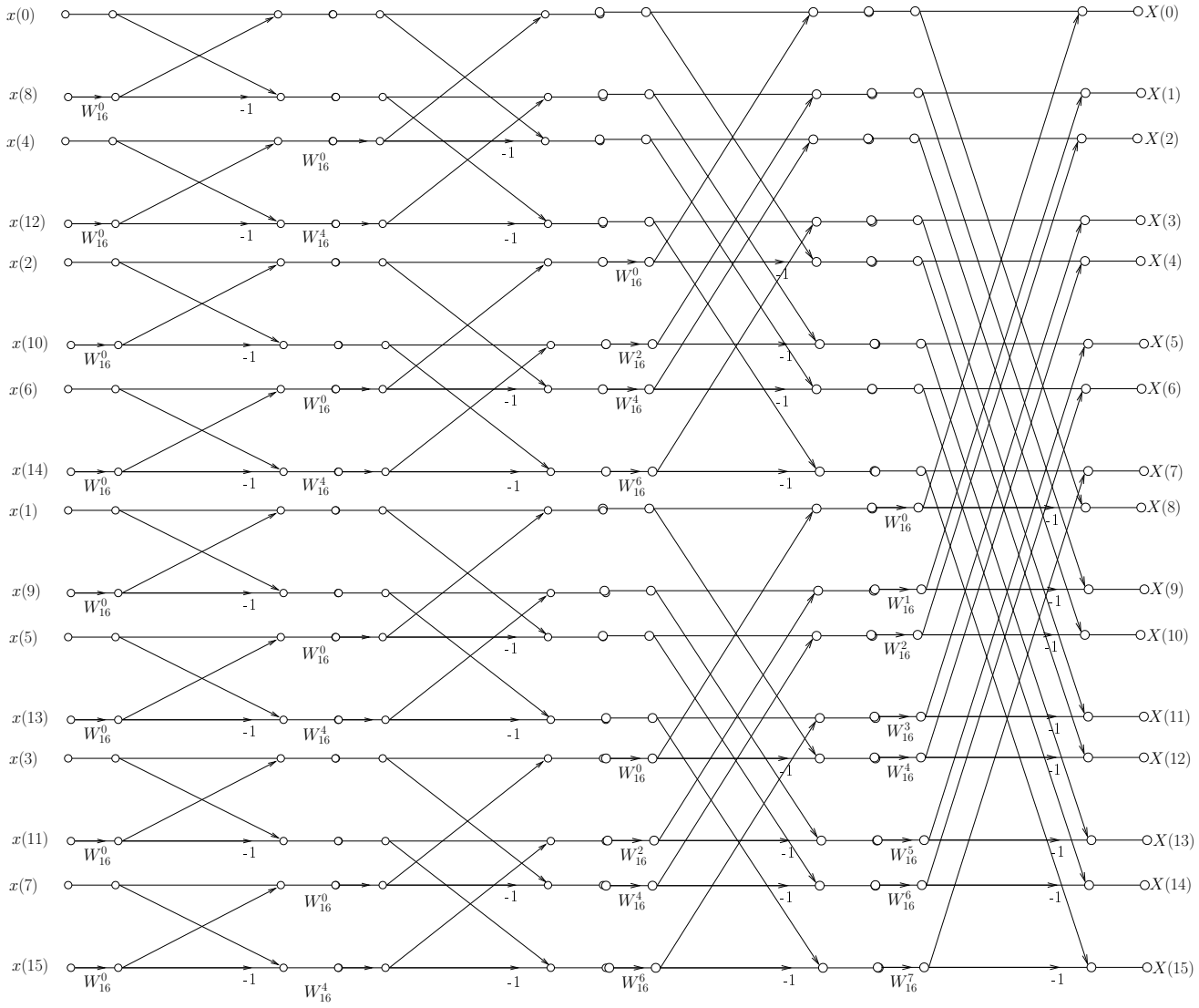

Figura 5.3: Gráfica de flujo del algoritmo FFT Radix-2 DIT para una señal con 16 muestras.

Se puede observar en la figura 5.3 que el algoritmo requiere que la señal de entrada este guardada en orden de bit inverso (*bit reversal order*) en localidades de memoria continuos. Para lograr éste orden en la señal de entrada cada uno de los indices decimales de las muestras temporales son convertidos a su representación binaria. La cadena binaria se invierte y convirtiendo la nueva cadena binaria a decimal da el orden de bit inverso en

los indices de las muestras temporales. En la tabla 5.1 se muestra este proceso para una señal temporal con 16 muestras.

Orden original		Orden de bit inverso	
Decimal	Binario	Binario	Decimal
0	0000	0000	0
1	0001	1000	8
2	0010	0100	4
3	0011	1100	12
4	0100	0010	2
5	0101	1010	10
6	0110	0110	6
7	0111	1110	14
8	1000	0001	1
9	1001	1001	9
10	1010	0101	5
11	1011	1101	13
12	1100	0011	3
13	1101	0111	11
14	1110	0111	7
15	1111	1111	15

Tabla 5.1: Proceso del ordenamiento de una señal con 16 muestras por bit inverso.

Para el cálculo del algoritmo FFT Radix-2 DIT se requieren $\frac{N}{2}log_2N$ multiplicaciones complejas y $Nlog_2N$ sumas complejas, lo cual se puede observar que es mucho menor que los cálculos requeridos para la DFT.

5.2.2. Decimación en Frecuencia (DIF).

El desarrollo del algoritmo de la FFT Radix-2 DIF es muy similar al desarrollo del algoritmo FFT Radix-2 DIT. De la misma manera, los datos se separan en grupos de $N/2$ muestras de datos continuos. $X(n)$ se puede expresar como:

$$X(n) = \sum_{k=0}^{(N/2)-1} x(k)W_N^{nk} + \sum_{k=0}^{(N/2)-1} x(k+(N/2))W_N^{n(k+N/2)}$$

A continuación se factoriza el término W_N^{nk} y se separan $X(n)$ en muestras pares e impares quedando de la siguiente manera:

$$
\begin{aligned}
X(2n) &= \sum_{k=0}^{(N/2)-1} [x(k) + x(k + (N/2))] W_{N/2}^{nk} \\
X(2n+1) &= \sum_{k=0}^{(N/2)-1} [x(k) - x(k + (N/2))] W_N^k W_{N/2}^{nk}
\end{aligned} \tag{5.4}
$$

La ecuación 5.4 es la mariposa Radix-2 DIF y en la figura 5.4 se muestra gráficamente.

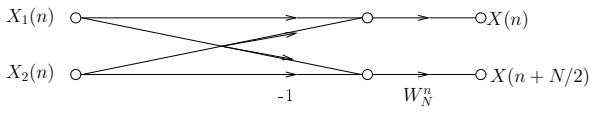

Figura 5.4: Representación de la mariposa del algoritmo FFT Radix-2 DIF.

De nuevo, la descomposición se realiza hasta que en la última etapa tenemos DFTs de dos puntos. En la figura 5.5 se tiene una gráfica de flujo del algoritmo FFT Radix-2 DIF para una señal con 16 muestras.

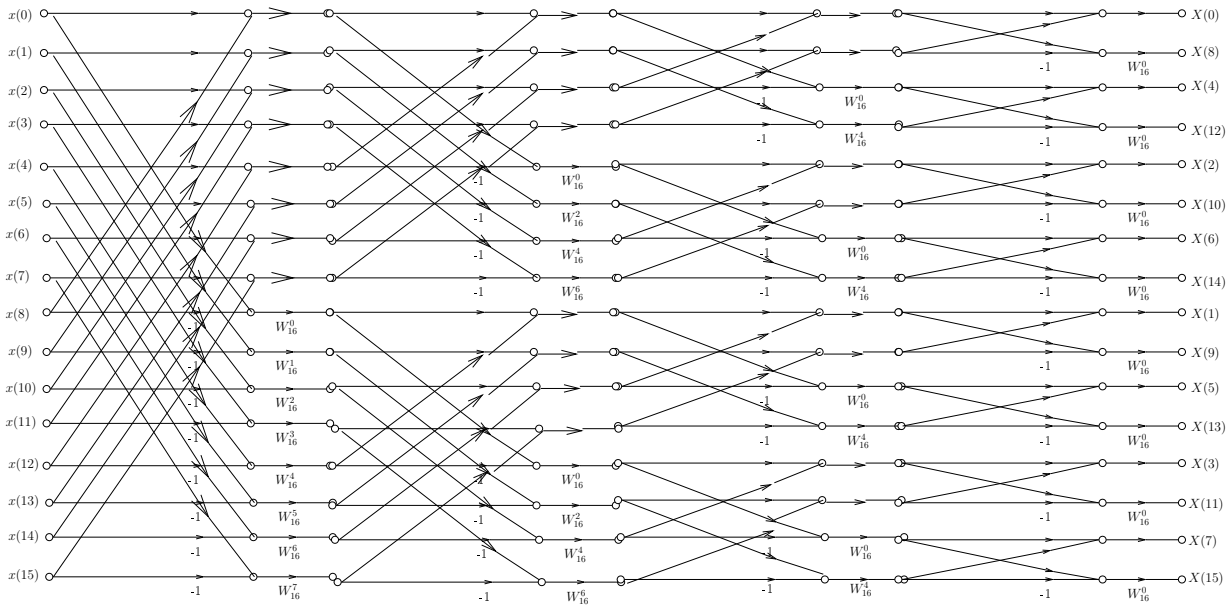

Figura 5.5: Gráfica de flujo del algoritmo FFT Radix-2 DIF para una señal con 16 muestras.

Se puede observar que la descomposición es de izquierda a derecha y las relaciones de simetría se invierten con respecto al algoritmo DIT, por lo que el reordenamiento de la

señal se realiza a la salida, en la frecuencia en lugar de en el tiempo.

Los cálculos requeridos en este algoritmo son los mismos que en el algoritmo DIT.

5.3. Radix-4.

5.3.1. DIT y DIF.

Los algoritmos FFT Radix-4 con decimación en el tiempo y decimación en frecuencia son más rápidos que los radix-2 puesto que reutilizan los resultados intermedios para el cálculo de la DFT. Esto se logra al descomponer la DFT en cuatro partes de longitud $N/4$. La ecuación (5.1) queda expresada de la siguiente manera:

$$X(n) = \sum_{k=0}^{(N/4)-1} x(k)W_N^{kn} + \sum_{k=N/4}^{(N/2)-1} x(k)W_N^{kn} + \sum_{k=N/2}^{(3N/4)-1} x(k)W_N^{kn} + \sum_{k=3N/4}^{N-1} x(k)W_N^{kn}$$

Se puede reescribir de la siguiente manera:

$$\begin{aligned} X(n) = & \sum_{k=0}^{(N/4)-1} x(k)W_N^{kn} + W_N^{Nn/4} \sum_{k=0}^{(N/4)-1} x\left(k+\frac{N}{4}\right) W_N^{kn} \\ & + W_N^{Nn/2} \sum_{k=0}^{(N/4)-1} x\left(k+\frac{N}{2}\right) W_N^{kn} + W_N^{3Nn/4} \sum_{k=0}^{(N/4)-1} x\left(k+\frac{3N}{4}\right) W_N^{kn} \end{aligned}$$

De la definición del factor de fase tenemos

$$W_N^{Nn/4} = (-j)^n \qquad W_N^{Nn/2} = (-1)^n \qquad W_N^{3Nn/4} = (j)^n$$

Por lo que

$$X(n) = \sum_{k=0}^{(N/4)-1} \left[x(k) + (-j)^n x\left(k+\frac{N}{4}\right) + (-1)^n x\left(k+\frac{N}{2}\right) + (j)^n x\left(k+\frac{3N}{4}\right) \right] W_N^{nk}$$

Si esta expresión se divide en 4 secuencias de $N/4$ puntos se obtiene la ecuación de la mariposa del algoritmo FFT Radix-4.

$$
\begin{aligned}
X(4h) &= \sum_{k=0}^{(N/4)-1} \left[x(k) + nx\left(k+\frac{N}{4}\right) + x\left(k+\frac{N}{2}\right) + x\left(k+\frac{3N}{4}\right) \right] W_{N/4}^{nk} \quad (5.5) \\
X(4n+1) &= \sum_{k=0}^{(N/4)-1} \left[x(k) - jx\left(k+\frac{N}{4}\right) - x\left(k+\frac{N}{2}\right) + jx\left(k+\frac{3N}{4}\right) \right] W_N^k W_{N/4}^{nk} \\
X(4n+2) &= \sum_{k=0}^{(N/4)-1} \left[x(k) - x\left(k+\frac{N}{4}\right) + x\left(k+\frac{N}{2}\right) - x\left(k+\frac{3N}{4}\right) \right] W_N^{2k} W_{N/4}^{nk} \\
X(4n+3) &= \sum_{k=0}^{(N/4)-1} \left[x(k) + jx\left(k+\frac{N}{4}\right) - x\left(k+\frac{N}{2}\right) - jx\left(k+\frac{3N}{4}\right) \right] W_N^{3k} W_{N/4}^{nk}
\end{aligned}
$$

En la figura 5.6 se muestra la estructura de mariposa del algoritmo FFT Radix-4 DIT [16].

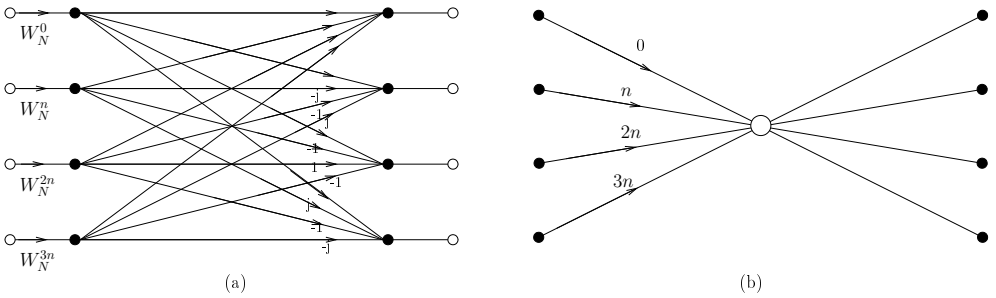

Figura 5.6: Representación de la mariposa del algoritmo FFT Radix-4. (a) Forma completa (b) Forma simplificada

Esta descomposición se realiza hasta tener DFTs de 4 puntos. El número de etapas para este algoritmo esta dado por:

$$M = \frac{log(N)}{log(4)}$$

donde N es la longitud de la señal de entrada. Para poder aplicar el algoritmo Radix-4 N debe ser múltiplo de 4.

En la figura 5.7 se muestra la gráfica de flujo del algoritmo FFT Radix-4 DIT para una señal con $N = 16$ muestras.

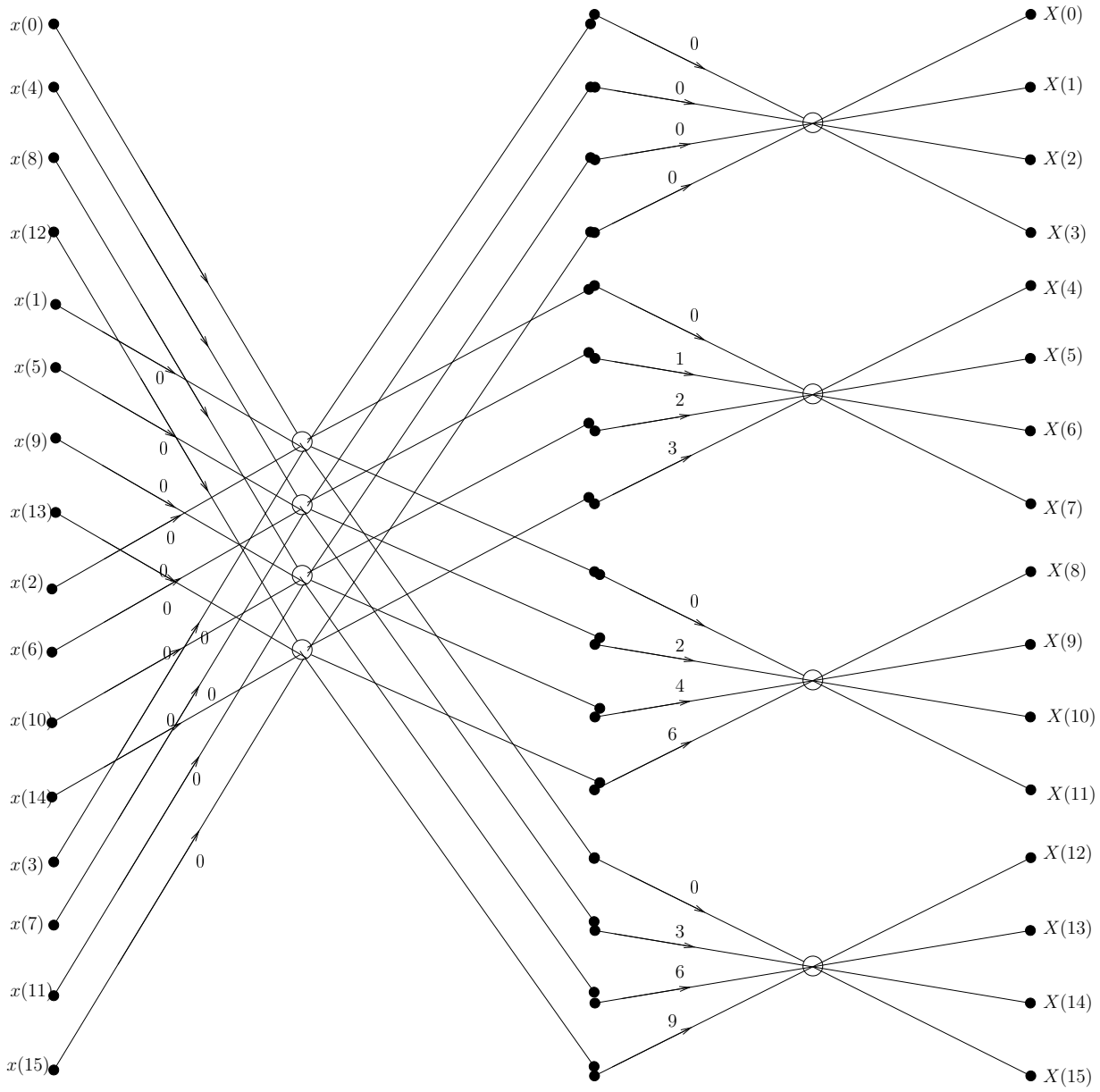

Figura 5.7: Gráfica de flujo del algoritmo FFT Radix-4 DIT.

El orden de los datos de entrada están reordenados por medio del orden de dígito inverso (*reverse digit order*), el cual consiste en convertir el índice temporal de forma decimal a su forma en base 4 y, al igual que en el orden de bit inverso, invertir la cadena y así obtener el nuevo orden de la señal. En la tabla 5.2 se muestra este proceso para una señal temporal con 16 muestras.

Orden original		Orden de dígito inverso	
Decimal	Base 4	Base 4	Decimal
0	00	00	0
1	01	10	4
2	02	20	8
3	03	30	12
4	10	01	1
5	11	11	5
6	12	21	9
7	13	31	13
8	20	02	2
9	21	12	6
10	22	22	10
11	23	32	14
12	30	03	3
13	31	13	7
14	32	23	11
15	33	33	15

Tabla 5.2: Proceso del ordenamiento de una señal con 16 muestras por dígito inverso base 4.

El número de operaciones para estos algoritmos son:
$\frac{3}{8}Nlog_2N$ multiplicaciones complejas (% 25 menos que en la FFT Radix-2)
$Nlog_2N$ sumas complejas (igual que en la FFT Radix-2)

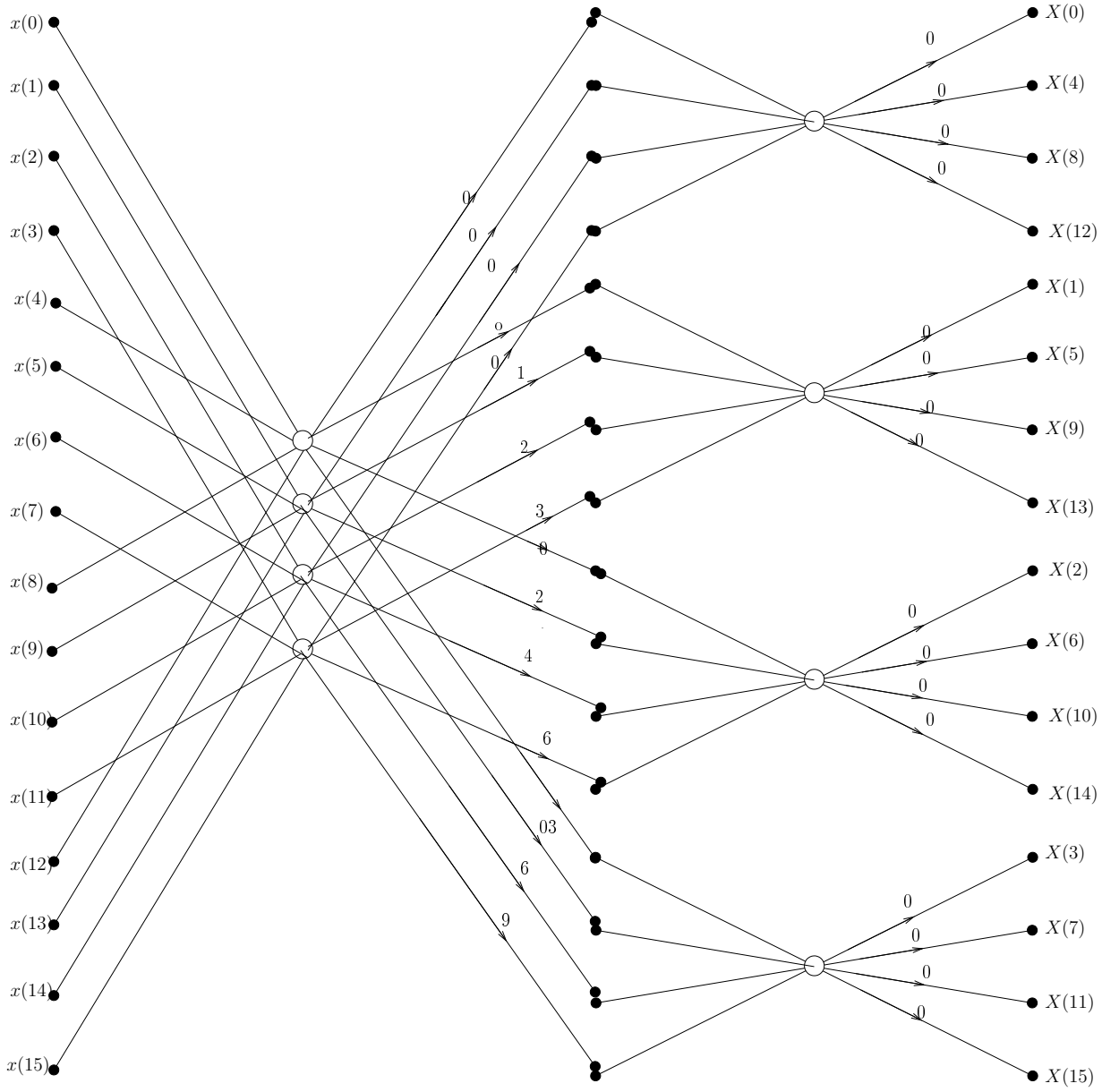

Figura 5.8: Gráfica de flujo del algoritmo FFT Radix-4 DIF.

En la figura 5.8 se muestra la gráfica de flujo del algoritmo FFT Radix-4 DIF para una señal con 16 muestras. Al igual que en el algoritmo FFT Radix-2 DIF, la diferencia con el DIT es que la entrada se realiza de manera ordenada y la salida se tiene que reordenar por dígito inverso base 4.

5.4. Complejidad Aritmética de los Algoritmos FFT en Tiempo Real.

Las aplicaciones de procesamiento digital de señales requieren que se realice la FFT en tiempo real para una cadena de datos N con una tasa de entrada de datos F [15]. En la tabla 5.3 se muestra un comparativo de los requerimientos en cuanto a multiplicaciones complejas por segundo (*complex multiplies per second, CMPS*) y sumas complejas por segundo (*complex additions per second, CAPS*) requieren tanto la DFT así como los algoritmos FFT Radix-2 y FFT Radix-4.

	DFT	FFT Radix-2	FFT Radix-4
CMPS	NF	$\frac{F}{2}log_2N$	$\frac{3F}{8}log_2N$
CAPS	$(N-1)F$	$Flog_2N$	$Flog_2N$

Tabla 5.3: Número de operaciones necesarias para el cálculo de la DFT y los diferentes algoritmos para ser implementadas en tiempo real.

Para WiMAX $F = floor(n \cdot BW/8000) * 8000$
Donde $n = 8/7$;
$BW = 10\ MHZ$
$\therefore F = 11'424,000[s/s]$

En la tabla 5.4 se muestran el número de cálculos aproximados necesarios para los requerimientos de WiMAX en tiempo real y en la tabla 5.5 se muestran el porcentaje de ahorro en el número de cálculos dentro de las FFTs.

	DFT		Radix-2		Radix-4	
N	CMPS	CAPS	CMPS	CAPS	CMPS	CAPS
128	1462272000	1450848000	39984000	79968000	-	-
256	2924544000	2913120000	45696000	91392000	34272000	91392000
512	5849088000	5837664000	51408000	102816000	-	-
1024	11698176000	11686752000	57120000	114240000	42840000	114240000
2048	23396352000	23384928000	62832000	125664000	-	-

Tabla 5.4: Número de cálculos necesarios para la FFT en OFDM WiMAX .

	Radix-2		Radix-4	
N	% CMPS	% CAPS	% CMPS	% CAPS
128	97.27	94.49	-	-
256	98.44	96.86	98.83	96.86
512	99.12	98.24	-	-
1024	99.51	99.02	99.63	99.02
2048	99.73	99.46	-	-

Tabla 5.5: Porcentaje de ahorro en el número de cálculos necesarios para la FFT en OFDM WiMAX .

Se puede observar que el número de cálculos requerido por los algoritmos de la FFT son mucho menores que los requeridos por la DFT. También se puede observar que el algoritmo FFT Radix-4 requiere menor número de multiplicaciones complejas para el cálculo de la transformada.

Una vez introducidos los cuatro algoritmos de la FFT de nuestro interés, serán implementados en el programa Matlab para así poder compararlos y posteriormente concluir cuál de ellos es el más apto para su implementación en un coprocesador en la cadena de la modulación OFDM de la capa física de WiMAX.

5.5. Programación de los algoritmos FFT.

5.5.1. FFT Radix-2 DIT.

En la siguiente sección se describe la programación del algoritmo Radix-2 DIT para el cálculo de la transformada de Fourier rápida.

5.5.1.1. Programación.

Para la programación del algoritmo Radix-2 fue necesario programar la estructura de la mariposa. Se debe tomar en cuenta cuales son los datos que van a entrar a la mariposa en cada etapa. El número de etapas se relaciona directamente con el número de datos que componen la señal de entrada, representados por N. La mariposa se realizara $N/2$ veces en cada una de las v etapas, en donde $v = log_2(N)$. N debe ser potencia de 2.

Los datos que se utilizan en la primera etapa deben pasar por el proceso de decimación del algoritmo, el cual consiste en separar la señal en 2 conjuntos de datos, los pares y los impares de acuerdo con su posición en el vector de datos. Este proceso se repite con cada conjunto hasta que se tengan N vectores con un dato cada uno.

En cada una de las etapas se utilizan diferentes factores de fase, los cuales se pueden volver a utilizar, por lo que se calculan todos los necesarios antes de entrar a la mariposa y al ser utilizados solamente serán llamados.

Los datos a la salida de la última etapa están en orden, por lo que no se debe realizar ningún acomodo. Estos datos representan la transformada de Fourier rápida de la señal de entrada.

En la figura 5.9 se presenta el diagrama de flujo ocupado para la programación del algoritmo Radix-2 DIT.

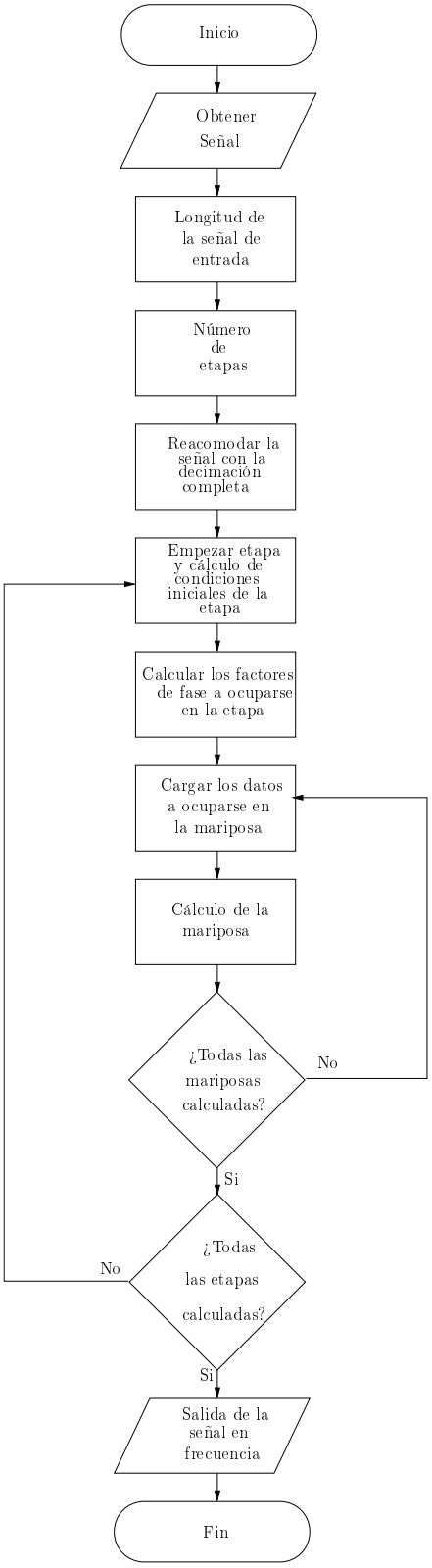

Figura 5.9: Diagrama de Flujo del Radix-2 DIT FFT.

Con base en este diagrama de flujo del algoritmo se programará la FFT Radix-2 DIT.

5.5.1.2. Programa.

El siguiente es el programa en Matlab de la FFT Radix-2 DIT.

```
    % Algoritmo Radix-2 DIT
function [X] = fftradix2dit(x)
N=length(x); % Long. de la señal de entrada.
v=log2(N); % Núm. de etapas.
N2=N/2;
X=bitrevorder(x); % Decimación de la señal.
for e=1:v, % Inicio de etapa.
  L=2^e; % Condiciones iniciales.
  n1=(2^(e-1))-1;
    r=2^(v-e);
    W=exp((-2*pi*i)*[0:n1]*r/N); % Cálculo de factores de fase.
    n2=1;
    for k=0:L:(N-L), % Inicio del cálculo de las mariposas.
        for n=0:(L/2)-1,
            a1=n+k+1;
            b1=(L/2)+a1;
            a=X(a1); % Carga valor 1.
            b=X(b1)*W(n2); % Carga valor 2 * factor de fase.
            X(a1) = a + b; % Salida 1 de la mariposa.
            X(b1) = a - b; % Salida 2 de la mariposa.
            n2=n2+1; % Proceso para el siguiente factor de fase.
            if n2>n1+1,
                n2=1;
            end
        end
    end % Fin de mariposas.
end % Fin de etapa.
end % Fin de función.
```

Lo primero que se hace es obtener la longitud de la señal, recordando que la única restricción para este tipo de algoritmos es que sea potencia de 2. Con la longitud se obtiene el número de etapas necesarias para la realización del algoritmo.

La función *bitrevorder(x)* de Matlab realiza la decimación de la señal. Ésta consiste en reacomodar la señal original de acuerdo con su posición en el vector de la señal original, se realiza al leer el número de la posición en binario de derecha a izquierda, por ejemplo $001 = 1$ se leerá $100 = 4$.

Las estructuras de mariposa se repiten $\frac{N}{2}$ veces cada una de las v etapas, por lo que se inicia el ciclo de cada una de las etapas. Dentro del ciclo se deben calcular las condiciones iniciales de cada etapa así como los factores de fase a utilizarse. También incluye otros

dos ciclos con los cuales se llama a los valores a utilizar en cada mariposa. Así como la variable que llama al factor de fase a utilizarse en cada mariposa.

Puesto que la salida de la etapa es la entrada de la siguiente, el resultado se guarda en una sola variable reemplazando los valores pasados por lo que se deben cargar los valores a ocuparse antes de entrar a la mariposa, asegurando así que utilizamos los valores correctos.

5.5.2. FFT Radix-2 DIF.

A continuación se describe la programación para el algoritmo Radix-2 DIF para el cálculo de la FFT.

5.5.2.1. Programación.

Una vez programado el algoritmo Radix-2 DIT es mucho mas sencillo programar el algoritmo Radix-2 DIF. La primera diferencia es el reacomodo de la señal de entrada, en la Decimación en Frecuencia el reacomodo se realiza al final del algoritmo. Como a la entrada del algoritmo se tiene la señal completa, los datos que entran a las mariposas son n y $n + N/2^e$ donde e es la etapa del algoritmo. La estructura de la mariposa es la misma que el Radix-2 DIT, la diferencia radica en los datos que entran a las mariposas en cada etapa y en donde se ocupa el factor de fase.

En la figura 5.10 se presenta el diagrama de flujo ocupado para la programación del algoritmo Radix-2 DIF.

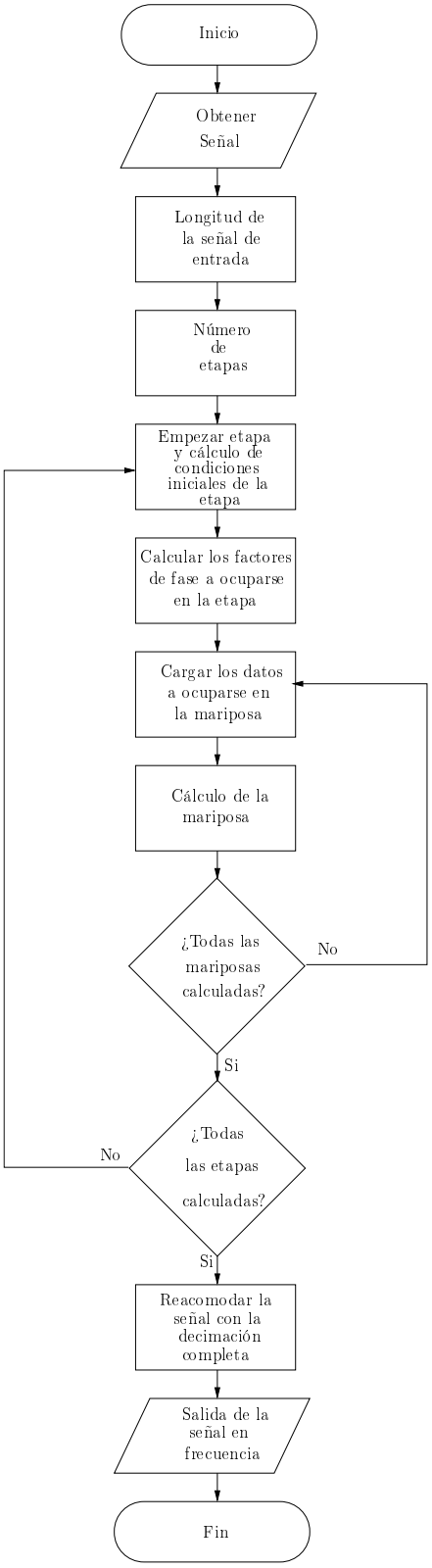

Figura 5.10: Diagrama de Flujo del Radix-2 DIF FFT.

5.5.2.2. Programa.

El siguiente es el programa en Matlab de la FFT Radix-2 DIF.

```
    % Algoritmo Radix-2 DIF
function [X] = fftradix2dif(x)
N=length(x); % Long. de la señal de entrada.
v=log2(N); % Núm. de etapas.
N2=N/2;
X=x; % Reasignación de la señal original.
for e=v:-1:1, % Inicio de etapa.
  L=2^e; % Condiciones iniciales.
  n1=(2^(e-1))-1;
  r=2^(v-e);
  W=exp((-2*pi*i)*[0:n1]*r/N); % Cálculo de factores de fase.
  n2=1;
  for k=0:L:(N-L), % Inicio del cálculo de las mariposas.
      for n=0:(L/2)-1,
          a1=n+k+1;
          b1=(L/2)+a1;
          a=X(a1); % Carga valor 1.
          b=X(b1); % Carga valor 2.
          X(a1) = a + b; % Salida 1 de la mariposa.
          X(b1) = (a - b) * W(n2); % Salida 2 de la mariposa.
          n2=n2+1; % Proceso para el sig. factor de fase.
          if n2>n1+1,
              n2=1;
          end
      end
  end % Fin de mariposas.
end % Fin de etapa.
X=bitrevorder(x); % Reacomodo de la señal de salida.
end % Fin de función.
```

En general, el programa es muy parecido al del Radix-2 cambiando en detalles mínimos. El primer cambio es el reacomodo de la señal de entrada, en la decimación en frecuencia se realiza al final de las etapas.

Una de las diferencias significativas de este algoritmo con el del Radix-2 DIT radica en cuales son los datos que entran a cada mariposa en las diferentes etapas. como ya se había mencionado, los datos que entran a cada mariposa son n y $n + N/2^e$ donde e es la etapa en la que se encuentra el algoritmo. Como tenemos la señal completa al principio del algoritmo, el contador e de las etapas irá desde v decrementándose hasta 1 en donde $v = log_2(N)$.

Otra diferencia esta en la multiplicación por el factor de fase, en este algoritmo se multiplica la segunda salida de la mariposa por el factor de fase, $X(b1) = (a-b)*W(n2)$, a diferencia del algoritmo Radix-2 DIT que la multiplicación se realiza por el segundo dato que entra a la mariposa antes de la suma o la resta respectiva de la mariposa.

5.5.3. FFT Radix-4 DIT.

A continuación se describirá la programación del algoritmo Radix-4 DIT para el cálculo de la transformada de Fourier rápida.

5.5.3.1. Programación.

Con la experiencia obtenida al programar el algoritmo Radix-2, la programación del Radix-4 fue un poco mas sencilla. La estructura de mariposa de este algoritmo es un poco diferente a la mariposa del Radix-2 puesto que entran y salen 4 datos de ella. Al igual que en el Radix-2, se deben realizar $N/4$ veces la mariposa en v etapas en donde, en este caso, $v = log_4(N)$. Debido a esto, la longitud N del vector de entrada debe ser potencia de 4.

La decimación en el tiempo del Radix-4 es diferente a la realizada en el Radix-2. Ésta divide en 4 conjuntos de datos dependiendo de su posición en el vector de datos. Al igual que en la decimación en el Radix-2 esta división se realiza repetidamente hasta tener N vectores con un dato cada uno.

En el Radix-4 se calculan todos los factores de fase antes de entrar a las etapas del algoritmo y son llamados de memoria cuando son utilizados.

Por ser caso de decimación en el tiempo, la salida se encuentra en orden y representa la transformada de Fourier rápida de la señal de entrada.

En la figura 5.11 se presenta el diagrama de flujo ocupado para la programación del algoritmo Radix-4 DIT.

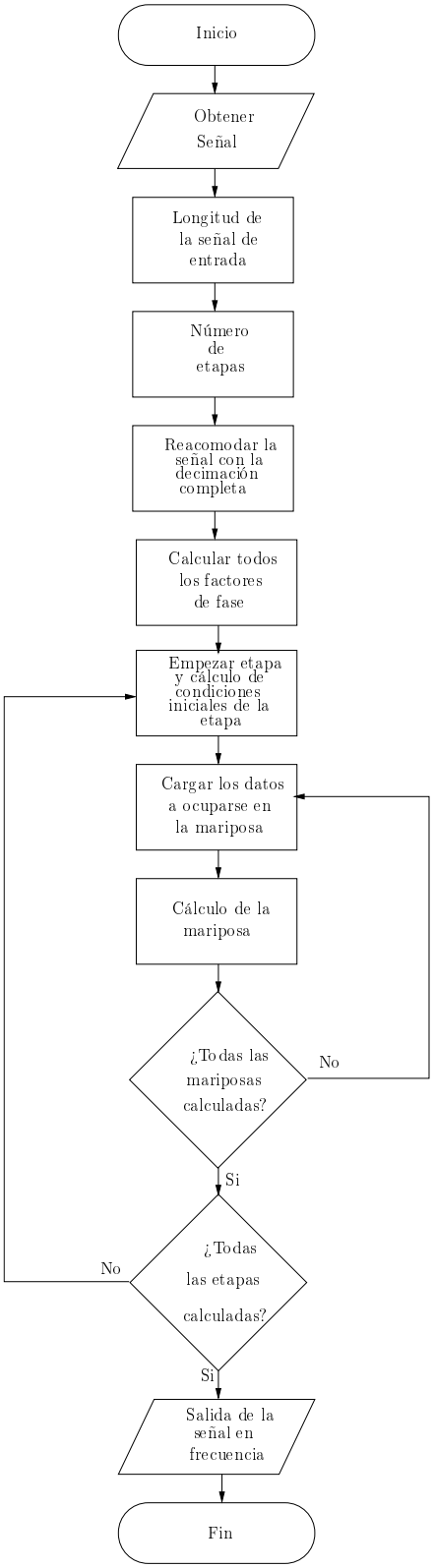

Figura 5.11: Diagrama de Flujo del Radix-4 DIT FFT.

5.5.3.2. Programa.

A continuación se muestra el código del programa del algoritmo Radix-4 DIT.

```
    %Algoritmo Radix-4 DIT
function [X] = fftradix4dit(x)
N=length(x); % Long. de la señal de entrada.
v=log(N)/log(4); % Núm. de etapas.
N4=N/4;
X=x; % Reasignación de la entrada.
W=exp((-2*pi*i)*[0:N-1]/N); % Cálculo de los factores de fase.
X=digitrevorder(X,4); % Reacomodo de la señal de salida.
for e=1:v, % Inicio de etapa.
    L=4^e; % Condiciones iniciales.
    n2=4^(v-e);
    if e==1, n2=0; end
    for k=0:L:(N-L), % Inicio del cálculo de las mariposas.
        n1=0;
        for n=0:(L/4)-1,
            a=n+k+1;
            b=(L/4)+a;
            c=(L/4)+b;
            d=(L/4)+c;
            A=X(a); % Carga valor 1.
            B=X(b)*W((1*n1*n2)+1); % Carga valor 2 * Factor de fase.
            C=X(c)*W((2*n1*n2)+1); % Carga valor 3* Factor de fase.
            D=X(d)*W((3*n1*n2)+1); % Carga valor 4* Factor de fase.
            X(a)=A+B+C+D; % Salida 1 de la mariposa.
            X(b)=(A-(j*B)-C+(j*D)); % Salida 2 de la mariposa.
            X(c)=(A-B+C-D); % Salida 3 de la mariposa.
            X(d)=(A+(j*B)-C-(j*D)); % Salida 4 de la mariposa.
            n1=n1+1; % Sig. factor de fase
        end
    end % Fin de mariposa.
end % Fin de etapa.
end % Fin de función.
```

La estructura el programa del algoritmo Radix-4 DIT es muy parecida a la del Radix-2 DIT. Lo primero que se realiza es obtener la longitud del vector de datos de entrada, recordando que debe ser potencia de 4. Una vez obtenida, se calcula el número de etapas que deberá realizar el programa. Al no poderse calcular directamente el $log_4(N)$ en Matlab se calcula de la siguiente manera: $v = \frac{log(N)}{log(4)}$.

Se calculan todos los factores de fase para luego ser llamados de memoria al utilizarlos. También se realiza la decimación con ayuda del comando $digitrevorder(X, 4)$ al cual se le debe indicar en vector a ser decimado, en este caso X, y el número de dígitos sobre el cual se basará para realizarla, en este caso es 4.

Se realizarán $N/4$ mariposas por cada etapa. En cada mariposa entran 4 datos, los cuales se combinan para obtener 4 datos de salida. Los datos 2,3 y 4 se multiplican por

un correspondiente factor de fase. La salida se guarda en la misma variable puesto que será la entrada en la siguiente etapa del programa.

Al terminar las v etapas obtenemos la transformada de Fourier rápida de los datos que entraron en el programa.

5.5.4. FFT Radix-4 DIF.

En la siguiente sección se describe la programación del algoritmo Radix-4 DIF para el cálculo de la transformada de Fourier rápida.

5.5.4.1. Programación.

Al igual que en el caso del algoritmo Radix-2, la programación de la variante con decimación en frecuencia después de haber programado la decimación en el tiempo es muy sencilla. Los cambios son en la decimación o reacomodo de la señal de entrada, en éste caso se realiza a la salida del algoritmo al terminar todas las etapas. Otra diferencia se encuentra el la multiplicación por los factores de fase, se realizan a la salida de la mariposa y no a la entrada como en la decimación en el tiempo.

En la figura 5.12 se presenta el diagrama de flujo ocupado para la programación del algoritmo Radix-4 DIF.

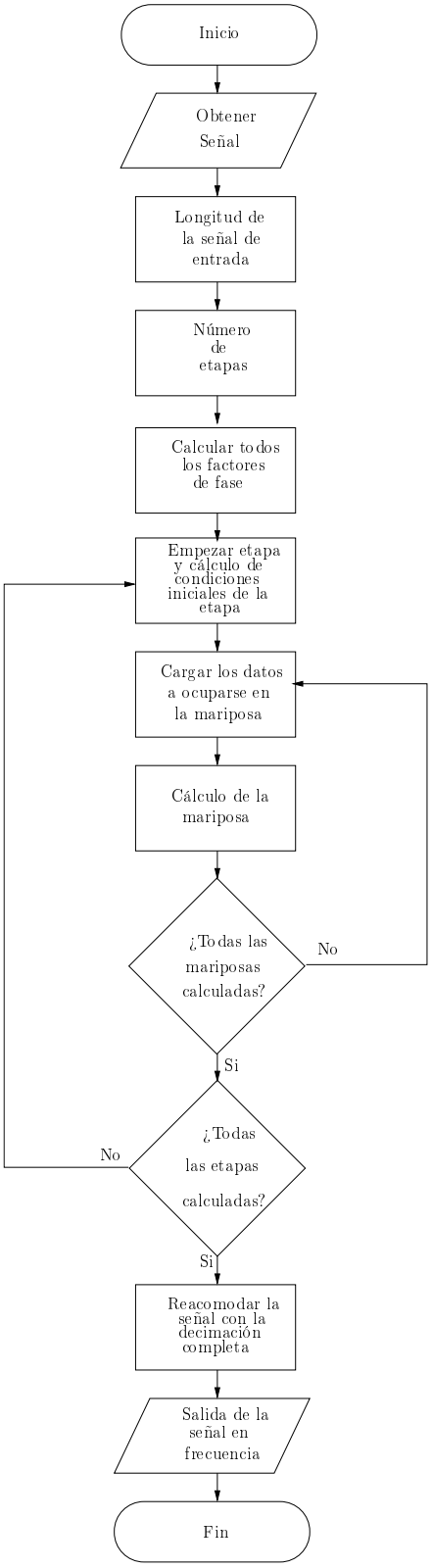

Figura 5.12: Diagrama de Flujo del Radix-4 DIF FFT.

5.5.4.2. Programa.

El siguiente es el programa en Matlab de la FFT Radix-4 DIF.

```
 %Algoritmo Radix-4 DIF
function [X] = fftradix4dif(x)
N=length(x); % Long. de la señal de entrada.
v=log(N)/log(4); % Núm. de etapas.
N4=N/4;
X=x;
W=exp((-2*pi*i)*[0:N-1]/N); % Cálculo de los factores de fase.
for e=v:-1:1, % Inicio de etapa.
    L=4^e; % Condiciones iniciales.
    n2=4^(v-e);
    if e==1, n2=0; end
    for k=0:L:(N-L), % Inicio del cálculo de las mariposas.
        n1=0;
        for n=0:(L/4)-1,
            a=n+k+1;
            b=(L/4)+a;
            c=(L/4)+b;
            d=(L/4)+c;
            A=X(a); % Carga valor 1.
            B=X(b); % Carga valor 2.
            C=X(c); % Carga valor 3.
            D=X(d); % Carga valor 4.
            X(a)=A+B+C+D; % Salida 1 de la mariposa.
            X(b)=(A-(j*B)-C+(j*D))*W((1*n1*n2)+1); % Salida 2 de la mariposa.
            X(c)=(A-B+C-D)*W((2*n1*n2)+1); % Salida 3 de la mariposa.
            X(d)=(A+(j*B)-C-(j*D))*W((3*n1*n2)+1); % Salida 4 de la mariposa.
            n1=n1+1; % Sig. factor de fase
        end
    end % Fin de mariposa.
end % Fin de etapa.
X=digitrevorder(X,4); % Reacomodo de la señal de salida.
end % Fin de función.
```

La estructura del programa es muy similar a la estructura del programa del Radix-4 DIT. El primer cambio se encuentra el las condiciones iniciales de los contadores que nos permiten apuntar a los datos que entraran en cada una de las etapas a las diferentes mariposas.

Los factores de fase son llamados de memoria a la salida de las mariposas, multiplicando las combinaciones de los datos de entrada en las salidas 2, 3 y 4.

Como ya se había mencionado, la decimación se realiza al final de las etapas antes de que salga la señal y así obtenemos la transformada de Fourier rápida de los datos de entrada.

Una vez programados los algoritmos de nuestro interés se realizó la validación de estos.

5.6. Prueba y Validación de los Algoritmos FFT.

La validación de los algoritmos programados se realizó con la función de Matlab, se probó con diferentes tipos de señales, la respuesta al impulso de un sistema con un polo y la respuesta al impulso de tres sistemas que representan tres canales de comunicación de fase mínima. Las funciones de transferencia de estos sistemas se muestran en la tabla 5.6.

Señal	Función de Transferencia
yt	$H_{yt} = \frac{1}{1-1.357z^{-1}+0.9216z^{-2}}$
$yc1$	$H_{c1} = 1 + 0.2z^{-1} - 0.6z^{-2}$
$yc2$	$H_{c2} = 1 + 0.51z^{-1} + 0.1997z^{-2}$
$yc3$	$H_{c3} = 1 + 0.536z^{-1} + 0.0718z^{-2}$

Tabla 5.6: Señales empleadas en la simulación.

La longitud de estas señales fue de 256 puntos, el cual es la longitud en el estándar de IEEE para WiMAX fijo.

En las figuras 5.13, 5.14, 5.15 y 5.16 se muestran las gráficas comparativas entre la FFT de Matlab y las FFTs programadas para las señales de respuesta al impulso de los sistemas de un solo polo y los canales de comunicación.

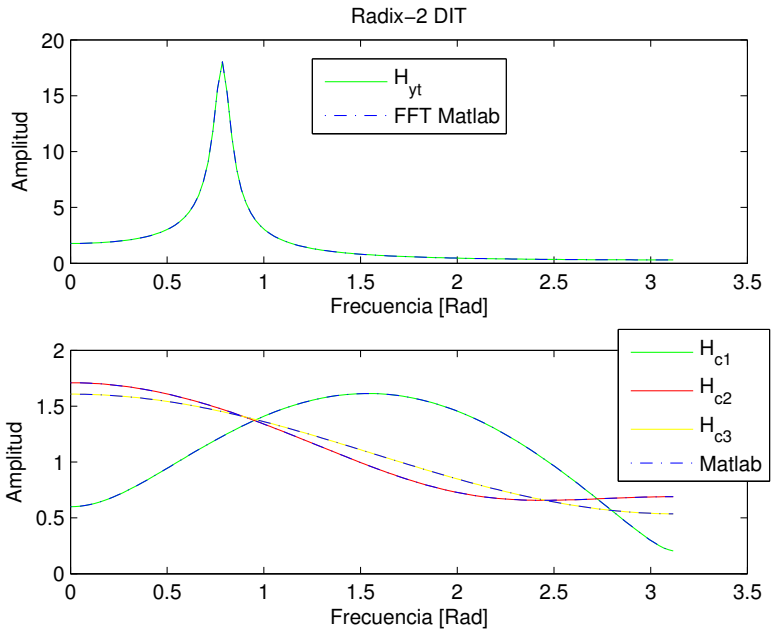

Figura 5.13: Representación en frecuencia de las señales. Comparación entre FFT de Matlab y FFT **Radix-2 DIT** programada.

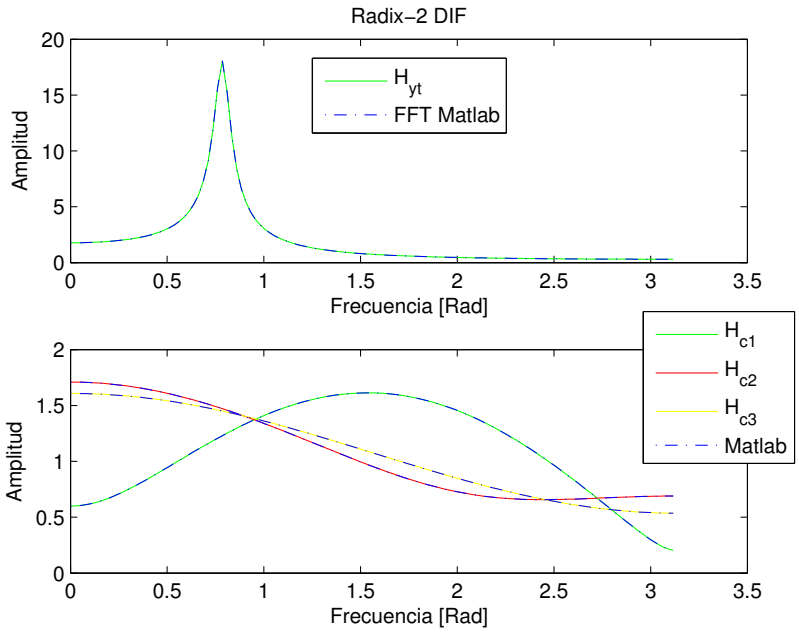

Figura 5.14: Representación en frecuencia de las señales. Comparación entre FFT de Matlab y FFT **Radix-2 DIF** programada.

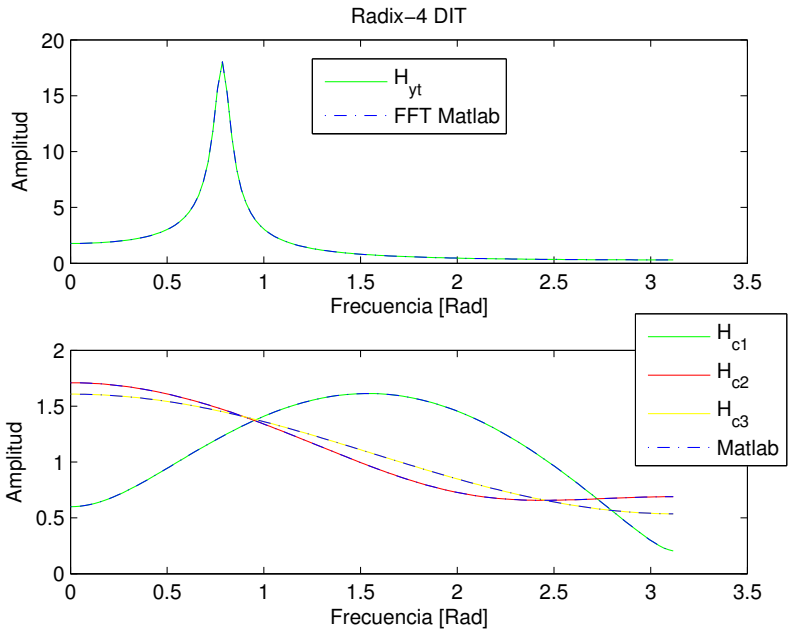

Figura 5.15: Representación en frecuencia de las señales. Comparación entre FFT de Matlab y FFT **Radix-4 DIT** programada.

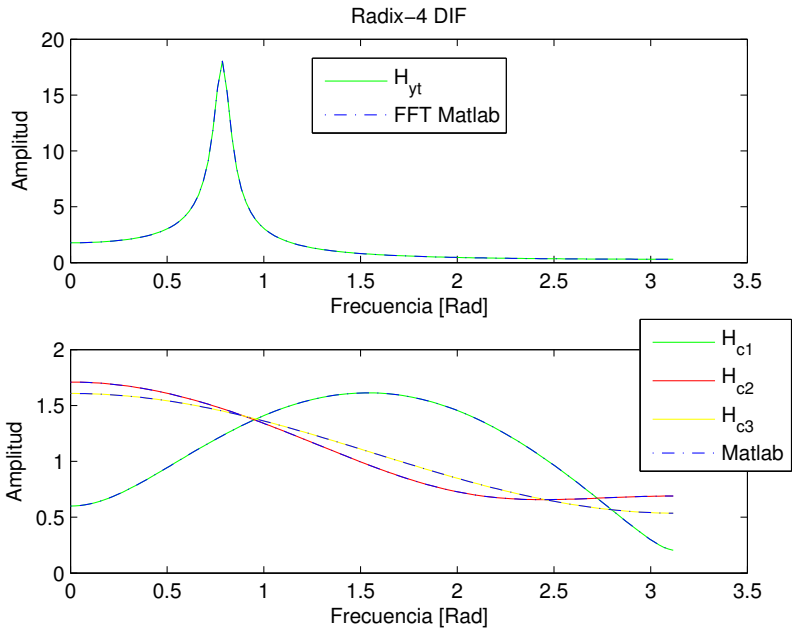

Figura 5.16: Representación en frecuencia de las señales. Comparación entre FFT de Matlab y FFT **Radix-4 DIF** programada.

Se observa en las cuatro gráficas que los espectros en frecuencia obtenidos por los algoritmos programados son idénticos a los obtenidos con la función de Matlab. Para saber que tanto se asemejan se realizó la prueba de precisión de cálculo que se muestra en el siguiente capítulo junto con la prueba de velocidad de cálculo.