**UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO**

FACULTAD DE INGENIERIA

# DESARROLLO DE UN INSTRUMENTO VIRTUAL PARA ESTIMACION ESPECTRAL MEDIANTE TECNICAS DE PROCESAMIENTO PARALELO

# T E S I S

QUE PARA OBTENER EL GRADO DE:

**INGENIERO EN COMPUTACION**

**P R E S E N T A :**

**LUIS ALBERTO AGUILAR BELTRAN**

DIRECTOR DE TESIS: DR. FABIAN GARCIA NOCETTI

**MEXICO, D. F.** **FEBRERO DE 1997**

## Agradecimientos

**A mi Padre:**

Por su ejemplo de rectitud, sus consejos y su gran córazon

**A mi Madre:**

Por su carácter, su amor y dedicación para conmigo

**A mis Hermanos:**

Justamente por ser mis hermanos

**A mi querida universidad:**

Por darme la oportunidad de acogerme en su seno y seguir aprendiendo

**A Aida:**

Por su amor, confianza y paciencia

**A Celic y Citlali:**

Por mostrarme el rumbo que debo seguir

**A Arturo:**

Por que su sacrificio no sea en vano

**A Fabian:**

Por la oportunidad que me ha dado y por todos sus consejos y ayuda

# *TEMARIO*

# CAPÍTULO 1

## *INTRODUCCIÓN*

### 1.1 Generalidades

El Laboratorio de Procesamiento Paralelo del Departamento de Electrónica y Automatización del Instituto de Investigaciones en Matemáticas Aplicadas y Sistemas de la UNAM, cuenta con un grupo de Investigación en la línea de procesamiento paralelo de señales e imágenes. Uno de los proyectos en los que actualmente se trabaja y en los que se ha centrado especial atención, está relacionado con la utilización de técnicas de procesamiento paralelo en la estimación espectral de señales Doppler de ultrasonido, con aplicación en la detección de flujo sanguíneo [15][16][18].

Esta aplicación corresponde a un problema de estimación espectral en tiempo real, en donde se requiere descomponer segmentos consecutivos de una señal en sus distintas frecuencias o componentes espectrales. Como dichos segmentos de la señal son del orden de los mili segundos, se requiere de un sistema de procesamiento de alto desempeño que permita adquirir y procesar la señales, así como desplegar el espectro correspondiente, en un intervalo del orden de los mili segundos [17][18][19][20].

Hasta el momento, en el Laboratorio de Procesamiento Paralelo, se han realizado trabajos previos [29][30], sobre cuestiones que implican la creación de nuevos mecanismos para acelerar la obtención de muestras y mejorar la precisión de las estimaciones hechas de las señales Doppler de ultrasonido. El presente trabajo viene a completar una parte importante de la línea de investigación seguida por los trabajos anteriores: *La creación de una interface gráfica de usuario que interactue con los elementos de procesamiento y sus correspondientes algoritmos utilizados en el análisis de las señales Doppler de ultrasonido.*

Para llevar acabo dicha tarea, se ha decidido implementar un Instrumento Virtual (IV). Un IV generalmente integra elementos de hardware y software que interactuan entre sí, los cuales nos ayudan a crear la "ilusión" de manipular un instrumento real, tal como un osciloscopio, un generador de señales, un tablero de control, etc., mediante una interface gráfica (comúnmente) presentada a través de la pantalla de una computadora (despliegue). Los usuarios que utilizan el IV son capases de observar e interactuar con dicha interface diseñada para el estudio y el análisis del fenómeno particular para el cual fue implementado y diseñado el IV.

En el presente proyecto, se han integrado dos elementos fundamentales: Primeramente la estimación espectral en tiempo real de señales Doppler de ultrasonido, realizada a través del procesamiento paralelo. En segundo lugar, la implementación de un IV para la visualización y configuración del tipo de señales Doppler de ultrasonido, atendiendo a los nuevos requerimientos y necesidades de los proyectos actuales que maneja el Laboratorio de Procesamiento Paralelo.

## 1.2 Objetivos

El Objetivo general de este trabajo es el diseño, desarrollo e implementación de un Instrumento Virtual para realizar Estimación Espectral [1][3][4] de señales, utilizando una plataforma de Procesamiento Paralelo. De esto se desprenden los siguientes objetivos particulares:

1. Implementar un Algoritmo de Estimación Espectral basado en la FFT, utilizando una arquitectura de procesamiento paralelo integrada por Transputer's.

2. Desarrollo de una GUI (Graphic User Interface) que permita configurar y controlar los diversos procesos que integran el IV.

3. Aplicar el Instrumento Virtual al caso de Estimación Espectral de señales Doppler de ultrasonido

## 1.3 Contenido

En el capítulo 1 se describen las generalidades del trabajo realizado, así como sus objetivos y el contenido de los capítulos.

En el capítulo 2 del presente trabajo se presenta los fundamentos del procesamiento paralelo así como sus diversas arquitecturas y se mencionan algunas de las métricas más utilizadas para medir el desempeño de una arquitectura de procesamiento paralelo, que ejecuta un algoritmo determinado para realizar una tarea dada, así como los factores que influyen en la adecuada utilización de los recursos. En este capítulo también se presentan los principales conceptos relativos a la Estimación Espectral.

En el capítulo 3 se presenta la arquitectura general del *Transputer* y se presenta el desarrollo de un algoritmo de estimación espectral implementado en está arquitectura de procesamiento paralelo.

En el capítulo 4, se dan a conocer las herramientas que se han utilizado en el presente trabajo para el diseño e implementación, de un IV basado en un sistema de Transputer's.

En el capítulo 5 se desarrolla un caso de estudio asociado a la estimación espectral de señales Doppler de ultrasonido, aplicando los conceptos de procesamiento paralelo, estimación espectral y la creación de un instrumento virtual que controla el comportamiento de la aplicación y también se presentan los resultados de la implementación del algoritmo de estimación espectral en un arreglo de Transputer's que va de uno a cuatro procesadores. Finalmente se presentan los resultados correspondientes a la interface gráfica de usuario que permite controlar y configurar el Instrumento Virtual.

Para finalizar, en el capítulo 6, se hace mención a las conclusiones y a los resultados obtenidos, ademas de analizar las expectativas de los trabajos futuros de esta línea de investigación.

# CAPÍTULO 2

## *GENERALIDADES*

### 2.1 Procesamiento Paralelo

Desde que las computadoras seriales fueron inventadas, su rapidez ha ido en incremento para solventar las aplicaciones que así lo requieran. No obstante, la limitación física fundamental impuesta por la rapidez de la luz hace imposible que este avance sea indefinido. Recientes tendencias nos indican que el rendimiento de las computadoras actuales comienza a saturarse.

Otra problema de las computadoras seriales es el llamado "cuello de botella" de Von Neumann que consiste en como la unidad central de proceso accesa la memoria del sistema. Mientras los datos son recuperados de la memoria principal, estos son transmitidos hacia los registros de la unidad central de proceso (UCP), y después de que estos registros son incrementados, son devueltos ha la memoria. Durante este periodo la unidad central de proceso permanece en un estado inactivo.

Una comparación costo/rendimiento de computadoras seriales sobre las ultimas décadas presenta una tendencia interesante como lo muestra la figura (2.1)

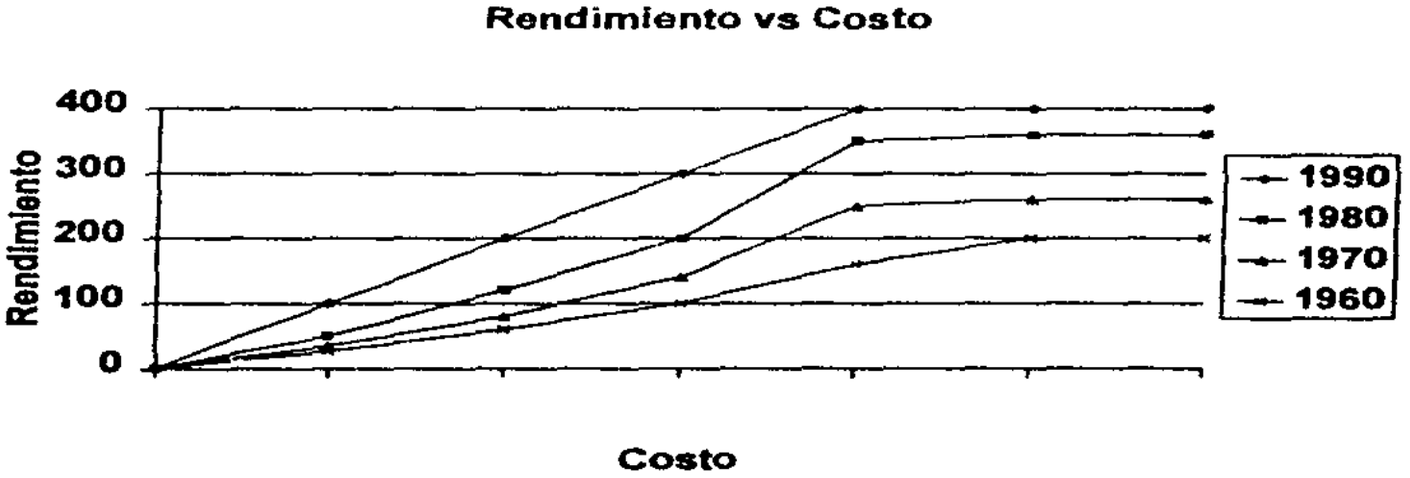



*Figura 2.1 Rendimiento de Computadores*

En las curvas más bajas, el rendimiento se incrementa más o menos linealmente con el costo. No obstante, más halla de un cierto punto en la curva, esta

comienza a saturarse y existe menor ganancia en el rendimiento con un exorbitante incremento en el costo. Además este punto de transición se vuelve más evidente conforme pasa el tiempo, primero debido a los avances en la tecnología VLSI (Very Large Scale Integration). Ahora es posible construir procesadores muy rápidos a un costo bajo. Esto ha hecho que se incremente la demanda y, por lo tanto, bajen los precios.

Esto nos lleva ha hacernos dos preguntas fundamentales:

1. ¿ Como podemos traspasar esta barrera?
2. ¿ De que manera podemos utilizar dicha solución ?

Una manera natural de solucionar la primera pregunta es utilizando un conjunto de procesadores para solucionar esta limitación, esto es, utilizar sistemas con capacidades de procesar varias tareas de manera paralela.

### 2.1.1 Un Poco de Historia

A finales de 1840, Charles Babbage concibe la manera de como realizar multiplicaciones e indexamientos aritméticos simultáneamente. El primer procesador paralelo en operación fue la **ILLIAC IV.** Esta maquina fue diseñada por Dan Slotnick en 1966, en la universidad de Illinois, y contenía 64 procesadores.

Aunque el primer sistema de procesamiento paralelo comercial fue desarrollado en 1986 por la compañía *Denelcor*, más de una docena de compañías ofrecían maquinas de procesamiento paralelo o estaban en proceso de construcción (DEC, IBM, Cray Research, Encore, Thinking Machines, etc.).

### 2.1.2 ¿ Que es el Procesamiento Paralelo ?

Si conectamos un gran número de procesadores en una computadora en paralelo, sobrepasara el punto de saturación de las velocidades logradas por las computadoras seriales y serán aún más baratas que estas ultimas cuando se equiparen en rendimiento.

La esencia de la computación paralela radica en el echo de dividir una cierta tarea entre varios procesadores, y que cada uno de ellos contribuya a realizar una parte de dicha tarea. De esta manera, la tarea será completada en menos tiempo que si se ejecutase en una computadora serial. Cabe hacer mención que al dividir un cierto

proceso "P", que tarda en ejecutarse un tiempo aproximado "T" en una computadora serial, el mismo proceso dividido entre un número de procesadores "Pn" no tardara un tiempo "Pn / T" en completarse ya que al distribuir un proceso en varios procesadores no se obtiene una relación lineal en cuanto al tiempo de proceso del algoritmo. Este tiempo dependerá de la carga asignada a cada procesador y de el **intercambio de información** que se realiza entre cada uno para compartir datos.

Con respecto a la segunda pregunta planteada unos párrafos arriba, presentaremos un ejemplo de la manera como podemos utilizar el procesamiento paralelo.

Ejemplo: Modelo de Análisis del Clima y Predicción

Considere el modelado del clima sobre un área de 5000× 5000 Km. Los parámetros deberán ser modelados a lo largo del plano vertical. Se asume que el área ha ser modelada tiene una altura de 15 Km. También se supone que el área comprendida entre estas longitudes 5000× 5000× 15 Km cúbicos es particionada en segmentos del tamaño de 0.1× 0.1× 0.1 Km cúbicos. Existen aproximadamente 2× 10 $e^{11}$ segmentos diferentes. El modelo del clima involucra al tiempo como otra dimensión. El tiempo es medido y los parámetros son calculados para cada segmento en intervalos regulares de tiempo.

Se asume que el modelo del clima se aplica durante un periodo de dos días y que los parámetros necesarios serán calculados una vez cada media hora. (Nótese que estas consideraciones son conservadoras y un modelado mas preciso requiere de más cálculos.) El calculo de los parámetros dentro de los segmentos usa ciertos valores iniciales y los valores de sus segmentos vecinos más cercanos.

Se asume que cada computación toma 100 instrucciones. Por lo tanto, una sola actualización de los parámetros en el dominio completo requiere 100× 2× 10 $e^{11}$ o 2× $e^{13}$ instrucciones. Ya que esto tiene que ser realizado aproximadamente 100 veces (2 días), El número total de operaciones es 2× 10 $e^{15}$. En una supercomputadora serial capaz de ejecutar un mil millones de instrucciones por segundo, este modelo tomaría aproximadamente 560 horas. Tomar 560 horas (o 23.3 días) para producir el pronostico del clima de los próximos dos días es irrazonable.

El procesamiento en paralelo hace posible predecir el clima no solo más rápido si no también más exactamente. Si se tiene una computadora en paralelo con miles de procesadores tipo estación de trabajo, podemos partir los 2× 10 $e^{11}$ segmentos del dominio entre estos procesadores. Cada procesador calculara los parámetros para cada

$10^8$ segmentos. Los procesadores comunican el valor de sus parámetros en sus segmentos a otros procesadores que lo requieran. Asumiendo que el poder computacional de esta computadora es de 100 millones de instrucciones por segundo y esta potencia es utilizada eficientemente, el problema puede ser solucionado en menos de 3 horas. El impacto de esta reducción en el tiempo de proceso tiene dos consecuencias. Primero, la computación en paralelo hace posible solucionar problemas que antes no era posible. Segundo, con la disponibilidad de computadoras más potentes, es posible modelar el clima más exactamente. Esto proporciona mayor precisión.

La adquisición y proceso de grandes cantidades de datos, desde fuentes como satélites, petróleo o el *análisis de señales en tiempo real* forman otra clase de problemas que requieren grandes cantidades de computo intensivo. Problemas de optimización discreta requieren dicho poder de computo para problemas como planeación, diseño de circuitos VLSI, logística y control. Los problemas de optimización discreta pueden ser solucionados utilizando técnicas de búsqueda estado-espaciales. Para muchos de estos problemas el tamaño del estado-espacio se incrementa de manera exponencial con el número de variables. Problemas que evalúan miles de millones de estados pertenecen a este dominio. Ya que procesar cada estado requiere de cantidades de cálculos no triviales, encontrar soluciones para una gran cantidad de estos problemas va más allá del alcance de la computación secuencial tradicional. Efectivamente, muchos problemas prácticos son resueltos utilizando algoritmos de aproximación que proveen una solución no muy precisa.

Otras aplicaciones que se pueden beneficiar significativamente de la computación paralela son el modelado de semiconductores, modelos oceánicos, tomografía computarizada, dinámica y diseño de vehículos, análisis de estructuras proteicas, estudio de los fenómenos químicos, procesamiento de imágenes y señales, monitores de la capa de ozono, explotación del petróleo, procesamiento del lenguaje natural, reconocimiento de voz, redes neuronales, visión, procesamiento de consultas a bases de datos y descubrimiento automático de conceptos y patrones de grandes bases de datos.

Muchas de las aplicaciones mencionadas en los párrafos anteriores, se conocen como problemas de *grandes desafíos*. Esta clase de problemas son fundamentales en la ciencia o la ingeniería ya que tienen un gran impacto económico y científico y cuyas soluciones pueden ser encontradas aplicando técnicas de computación de alto rendimiento como lo es el procesamiento paralelo.

### 2.1.3 Factores de Eficiencia

Para utilizar el computo paralelo eficientemente, se necesita tener en cuenta los siguientes puntos:

**• Diseño de la plataforma de computación paralela**

Es importante el diseño de la plataforma de computación paralela para escalar el número de procesadores y la capacidad de soporte de comunicaciones de datos a velocidades muy altas entre procesadores

**• Diseño de algoritmos eficientes**

Una plataforma paralela es de poca ayuda a menos que eficientes algoritmos paralelos estén disponibles. La cuestión de diseñar algoritmos paralelos es muy diferente al diseño de algoritmos secuenciales.

**• Métodos para evaluación de algoritmos paralelos**

Dada una plataforma de procesamiento paralelo y un algoritmo ejecutándose en ésta, se necesita evaluar el desempeño del sistema resultante. Dicho análisis nos permite respondes a preguntas tales como ¿Que tan rápido puede un problema ser solucionado utilizando el procesamiento paralelo? y ¿Que tan eficiente son los procesadores utilizados?.

**• Lenguajes para computo paralelo**

Algoritmos paralelos son implementados en computadoras paralelas utilizando lenguajes de programación. Estos lenguajes deben ser lo suficientemente flexibles para permitir una implementación eficiente y una programación relativamente fácil.

**• Herramientas de programación paralela**

Para facilitar la programación de computadoras paralelas, es importante desarrollar ambientes de programación y herramientas comprensivas. Esto con el fin de ocultar al usuario de los niveles bajos de las características de la máquina y proveerlos con herramientas de diseño y desarrollo tales como depuradores y simuladores.

**• Programas paralelos portables**

La portabilidad es uno de los problemas principales con las actuales computadoras paralelas. Típicamente un programa escrito en una computadora paralela requiere modificaciones extensivas para que logre ejecutarse en otra computadora paralela. Esta es una cuestión importante que debe recibir una considerable atención.

**• Programación automática de computadoras paralelas**

Mucho del trabajo es realizado en el diseño de paralelización de compiladores, los cuales extraen el paralelismo implícito de los programas que no han sido paralelizados explícitamente. Dichos compiladores se espera que nos permitan programar computadoras paralelas tal como si se programara una computadora secuencial. Se especula que este enfoque tenga potencial mete limitaciones para explotar el poder de computadoras paralelas de gran escala.

### 2.1.4 Métricas de Desempeño

Un algoritmo secuencial es usualmente evaluado en términos de su tiempo de ejecución, expresado como una función del tamaño de su entrada. El tiempo de ejecución de un algoritmo paralelo depende no solo del tamaño de su entrada si no también de la arquitectura paralela y del número de procesadores. Por lo tanto, un algoritmo paralelo no puede ser evaluado aisladamente de la arquitectura paralela.

Antes de mencionar las métricas más conocidas en arquitecturas paralelas, definiremos el concepto de *granularidad* [9] ya que su definición incide directamente en las métricas utilizadas para medir el desempeño de un algoritmo en una arquitectura paralela.

**A) Granularidad**

La granularidad de una arquitectura paralela puede definirse como la relación del tiempo requerido para una operación de comunicación básica al tiempo requerido para una computación básica. Las arquitecturas paralelas para las cuales esta relación es pequeña son aptas para algoritmos que requieren una comunicación frecuente; esto es, algoritmos en los cuales la granularidad es pequeña. En contraste, las arquitecturas paralelas para las cuales esta relación es grande, son aptas para algoritmos que no requieren comunicación frecuente entre sus procesadores.

**B) Tiempo de Ejecución**

El tiempo de ejecución serial de un programa es el tiempo que transcurre entre el comienzo y el final de su ejecución en una computadora serial. El tiempo de ejecución paralelo es el tiempo que transcurre desde el momento en que una computación paralela comienza al momento que el último proceso termina su ejecución. Denotaremos el tiempo de ejecución serial con $T_s$ y el tiempo de ejecución paralelo con $T_p$.

**C) Speedup**

Cuando se evalúa un sistema paralelo, nos interesamos en conocer cuanto desempeño se a ganado paralelizando una aplicación dada sobre su secuencial implementación. El *speedup* es una medida que captura los beneficios relativos de implementar un algoritmo en una plataforma paralela. El speedup esta definido como la relación del tiempo tomado para solucionar un problema en un solo procesador al tiempo requerido para solucionar el mismo problema en una computadora paralela con $p$ procesadores idénticos. Denotaremos el speedup con el símbolo $S$.

Para un problema dado, más de un algoritmo secuencial puede existir, pero no todos serán aptos de ser paralelizados de igual manera. Cuando una computadora serial es utilizada, es natural utilizar un algoritmo secuencial para solucionar el problema o implementar la aplicación requerida en el menor tiempo posible. Dado un algoritmo paralelo, es difícil juzgar su desempeño con respecto al algoritmo secuencial más rápido disponible para resolver el problema en un solo procesador, por lo que se compara el desempeño del algoritmo paralelo con el desempeño del algoritmo secuencial para el mismo problema. Se define formalmente el speedup $S$ como la relación del tiempo de ejecución del mejor algoritmo serial para un problema dado con el tiempo de ejecución para el mejor algoritmo paralelo para el problema en cuestión resuelto en $p$ procesadores. Los $p$ procesadores utilizados por el algoritmo paralelo, deberán ser idénticos al utilizado en la implantación del algoritmo secuencial.

Teóricamente el speedup no puede exceder el número de procesadores, $p$. Si el mejor algoritmo secuencial toma $T_s$ unidades de tiempo para solucionar un problema dado en un procesador, entonces el speedup de $p$ puede ser obtenido en $p$ procesadores si ninguno de estos gasta menos tiempo que $T_s / p$ . Un speedup mayor que $p$ es posible si cada procesador gasta más tiempo que $T_s / p$ para solucionar el problema. En este caso, un solo procesador puede emular $p$ procesadores y solucionar el problema en menos de $T_s$ unidades de tiempo. Esto es una contradicción porque el speedup , por definición, es calculado con respecto al mejor algoritmo secuencial. Si $T_s$ es el tiempo serial que consume un algoritmo, el problema no puede ser solucionado en un tiempo menor a $T_s$ en un solo procesador.

**D) Eficiencia**

Solo en un sistema de procesamiento paralelo ideal que contiene $p$ procesadores, puede alcanzar un speedup de igual a $p$ . En la practica, un ambiente ideal no se logra ya que mientras se ejecuta un algoritmo paralelo, el procesador puede no dedicar el 100% de su tiempo para cálculos del algoritmo. El tiempo restante esta invertido, principalmente, en las comunicaciones para intercambio de datos entre procesadores. La eficiencia es la medida de la fracción de tiempo para la cual un procesador es completamente utilizado, esto es, el porcentaje de tiempo para el cual el procesador realiza un número dado de computaciones. La eficiencia esta definida como la relación del speedup al número de procesadores. En un sistema de procesamiento paralelo ideal, el speedup es igual a $p$ y la eficiencia es igual a uno. En la práctica, el speedup es menor que $p$ y la eficiencia oscila entre cero y uno, dependiendo del grado de efectividad con la cual los procesadores son utilizados. La eficiencia se denota con el símbolo $E$ . Matemáticamente esta dada por:

$$E = \frac{S}{p} \qquad (2.1)$$

**E) Costo**

Definimos el costo de solucionar un problema en una arquitectura paralela como el producto de el tiempo de ejecución paralelo y el número de procesadores utilizado. El costo refleja la suma del tiempo que cada procesador gasta en solucionar un problema. La eficiencia puede solo ser expresada como la relación del tiempo de ejecución de el algoritmo secuencial más rápido para solucionar un problema a el costo de solucionar el mismo problema en $p$ procesadores.

El costo de solucionar un problema en un solo procesador es el tiempo de ejecución del algoritmo secuencial más rápido conocido. Un sistema de procesamiento paralelo se dice que es óptimo en costo si el costo de solucionar el problema en una arquitectura paralela es proporcional al tiempo de ejecución de el algoritmo secuencial más rápido conocido ejecutándose en un solo procesador. Ya que la eficiencia es la relación del costo secuencial a el costo paralelo, un sistema de procesamiento paralelo tendrá una eficiencia de uno.

### 2.1.5 Fuentes de Subutilización (Overhead) en Arquitecturas Paralelas

Los sistemas de procesamiento paralelo reales no llegan a alcanzar una eficiencia de 1 o un speedup de $p$ aunque se tengan muchos procesadores. La razón de esto es que parte de el tiempo es gastado en al comunicación entre procesadores. En general, detrás de la comunicación entre procesos, pueden existir otras causas para una perdida de eficiencia en un sistema de procesamiento paralelo. Todas las causas que contribuyen para no lograr un óptimo desempeño de un sistema de procesamiento paralelo se conocen colectivamente como *subutilización* (overhead) de un sistema de procesamiento paralelo.

Se define el overhead total o la función de overhead de un sistema de procesamiento paralelo, como parte de su costo (producto de el tiempo de proceso) que no es afectado para el más rápido algoritmo secuencial conocido ejecutándose en una computadora secuencial. Éste es el tiempo total gastado por todos los procesadores que es adicional al requerido para solucionar el mismo problema en una computadora secuencial con el algoritmo más rápidamente conocido. La función de overhead se denota con el símbolo $T_o$ . $T_o$ es función de $W$ y $p$ y se suele escribir como $T_o$ ($W$ , $p$) . (Un algoritmo que requiere $W$ básicos pasos computacionales, puede ser mapeado en un máximo de $p$ procesadores. Con este mapeo, cada procesador ejecuta un solo paso del algoritmo secuencial).

El costo de resolver un problema de tamaño $W$ en $p$ procesadores, o el tiempo total empleado en solucionar un problema sobre todos los procesadores es $p\,T_p$ .. $W$ unidades. Éste tiempo es consumido realizando un trabajo útil, y el resto es overhead. Además, la relación entre el costo ($p\,T_p$), tamaño del problema ($W$), y la función de overhead ($T_o$) esta dado por:

$$T_o = p\,T_p - W \qquad (2.2)$$

La discusión anterior muestra que la función de overhead caracteriza a un sistema de procesamiento paralelo. Dada una función de overhead, se puede expresar el tiempo de ejecución paralelo, speedup, eficiencia y costo de un sistema de procesamiento paralelo en términos de dos parámetros básicos: el tamaño del problema $W$ y el número de procesadores $p$ . A continuación se presentan los principales factores que pueden contribuir al overhead total del sistema de procesamiento paralelo.

**A) Comunicación Entre Procesos**

Cualquier sistema paralelo no trivial requiere la comunicación entre procesadores. El tiempo para transferir datos entre procesadores es usualmente la fuente más significativa de overhead en un sistema de procesamiento paralelo. Si cada uno de los procesadores $p$ consume $t_{comm}$ tiempo de comunicación, entonces la comunicación entre procesadores contribuye $t_{comm} \times p$ a la función de overhead.

**B) Carga Desbalanceada**

En muchas aplicaciones (por ejemplo búsquedas y optimizaciones) es imposible (o al menos muy difícil) predecir el tamaño de las sub tareas asignadas a varios procesadores. Por lo tanto, el problema no puede ser subdividido estáticamente entre los procesadores mientras se mantiene uniforme la carga de proceso. Si distintos procesadores tienen diferente carga de trabajo, algunos procesadores pueden estar desocupados durante parte del tiempo mientras los otros están procesando.

Frecuentemente algunos u otros procesadores deben sincronizarse en ciertos puntos durante la ejecución del programa. Si todos los procesadores no están listos para la sincronización al mismo tiempo, entonces los que están listos primero estarán ociosos hasta que los demás estén listos. Sin importar las causas de esta espera, el tiempo total de ocio de todos los procesadores contribuye a la función de overhead. Un caso especial de overhead que hace que el procesador se encuentre ocioso, es la presencia de un componente secuencial en el algoritmo paralelo. Parte del algoritmo puede ser no paralelizable, permitiendo que solo un procesador trabaje en ésta parte. Expresaremos el tamaño del problema para un algoritmo parecido como la suma de dos componentes. $W_S$ , el trabajo realizado por el componente secuencial, y $W_P$ , el trabajo realizado por el componente paralelo. Mientras un procesador está trabajando en $W_S$ , el resto, $p$ - 1, están ociosos. Como resultado, un componente serial $W_S$ contribuye $(p - 1)\ W_S$ a la función de overhead en un sistema paralelo de $p$ procesadores.

**C) Computación Extra**

El algoritmo secuencial más rápido conocido para un problema puede ser difícil o imposible de paralelizar, forzándonos a utilizar un algoritmo paralelo basado en una pobre pero fácil implementación de un algoritmo secuencial (esto es, uno con un alto grado de concurrencia). Hagamos que $W$ sea el tiempo de ejecución del programa secuencial más rápido conocido para un problema y $W'$ sea el tiempo de ejecución de el algoritmo más pobre conocido para el mismo problema. Entonces la diferencia $W' - W$ debe ser la parte correspondiente de la función de overhead ya que expresa la cantidad de trabajo extra realizado para solucionar el problema en paralelo.

Un algoritmo paralelo basado en el mejor algoritmo secuencial puede realizar un mayor número de operaciones que su contra parte secuencial. Un caso particular es el de la Transformada Rápida de Fourier. En su versión serial, el resultado de ciertos cálculos puede ser re utilizado. Sin embargo, en su versión paralela, estos resultados no pueden ser re utilizados ya que son generados en distintos procesadores. Además algunos cálculos son ejecutados varias veces en distintos procesadores. Estos cálculos extra contribuyen a la función de overhead.

## 2.2 Estimación Espectral

### 2.2.1 ¿ Que es la Estimación Espectral ?

Dada una señal culquiera, estamos interesados en conocer sus principales características (amplitud, forma de onda, período, etc). Si analizamos una señal en el dominio del tiempo nos resulta más difícil obtener dichas características, por lo que un análisis en el dominio de la frecuencia resulta en una mejor calidad de la información buscada. El análisis de una señal involucra la descomposición de está en sus distintas frecuencias que la componen. Ha está descomposición se le denomina *espectro de la señal*. El espectro de una señal provee una identificación única ya que cualquier otra señal con el mismo espectro, será un sinónimo de está.

Una característica de las señales de la mayoría de los fenómenos físicos que existen, es que su comportamiento es aleatorio (random). Muchos de los fenómenos que ocurren en la naturaleza son mejor caracterizados de una manera estadística en términos de medias estadísticas (promedios). Por ejemplo, los fenómenos meteorológicos tales como las fluctuaciones en la temperatura del aire y la presión son mejor caracterizados estadísticamente como procesos aleatorios. El ruido en el voltaje causado por el calor a través de una resistencia y en los dispositivos electrónicos son ejemplos adicionales de señales físicas que son bien modeladas como procesos aleatorios. Por lo que el proceso de obtener el espectro de una señal dada, a partir de recopilar información de manera directa o indirecta de dicha señal, se le denomina *estimación espectral* [1][3]. Por el contrario, si el espectro de una señal se puede obtener por medios puramente matemáticos, decimos que hemos realizado su *análisis espectral*.

Debido a las fluctuaciones aleatorias que existen en las señales del mundo real, debemos adoptar un punto de vista estadístico, el cual trata con las características de la media de las señales aleatorias; pues es imposible modelar matemáticamente el comportamiento de dichas señales (al menos por el momento). Debido a que se utilizan métodos estadísticos para modelar dichos fenómenos, decimos que estamos realizando una estimación (aproximación) del espectro de la señal original.

Una vez introducida la idea básica del concepto de estimación espectral, revisaremos algunos otros tópicos relacionados a éste tema.

### 2.2.2 Elementos Básicos para Estimación Espectral

A continuación, presentaremos brevemente los conceptos básicos del análisis de señales y de estimación espectral, ya que estos son piezas fundamentales para el desarrollo del presente trabajo.

**A) Señales**

Una señal esta definida como cualquier cantidad física que varia con el tiempo, espacio o cualquier otra variable o variables independientes. Las señales pueden ser, en algunos casos, absolutamente determinadas por medios matemáticos o no. Si una señal puede ser modelada completamente por una ecuación matemática, se dice que es una señal *determinlstica*. Si por el contrario, la señal no puede ser modelada por una ecuación matemática se le denomina *aleatoria*. En el mundo real, la mayoría de las señales son del tipo aleatorio. Por ejemplo, una señal de voz no puede ser descrita funcionalmente por una expresión matemática. En general , un segmento de voz puede ser representado con gran precisión como una suma de varias senoides de diferentes amplitudes y frecuencias, esto es:

$$\sum_{i=1}^{N} A_i(t)\,sen(2\pi F_i(t)t + \theta_i(t)) \qquad (2.3)$$

donde $A_i(t)$, $F_i(t)$ y $\theta_i(t)$ son un conjunto de amplitudes, frecuencias y fases, respectivamente, de la senoide. En efecto, el camino para interpretar el contenido de la información del mensaje de voz es medir la amplitud, frecuencia y fase contenida en el lapso de tiempo de donde se obtuvo el segmento de la señal.

Otros ejemplos de señales son los electrocardiogramas, electroencefalogramas, señales Doppler, señales sísmicas, imágenes, etc.

**B) Sistemas**

Asociado con la naturaleza de las señales es el medio por el cual estas son generadas. Por ejemplo, las señales de voz son generadas por la fuerza del aire que atraviesa las cuerdas vocales. Las imágenes son obtenidas por la exposición de una película fotográfica a la escena de un objeto. Una señal Doppler es generada o puede ser generada mediante un transductor ultrasónico. La generación de estas señales es usualmente asociada con un sistema que responde a estímulos o fuerzas. Los estímulos en combinación con el sistema es denominado la *fuente de la señal*

Un sistema puede ser definido como un dispositivo físico que ejecuta una operación en una señal. por ejemplo un filtro que es usado para reducir el ruido y la interferencia en la transmisión de información que conlleva una señal es llamado un sistema. En este caso el filtro ejecuta alguna operación en la señal, la cual tiene el efecto de reducir el ruido y la interferencia de la información deseada que conlleva la señal.

**C) Procesamiento de Señales**

Cuando una señal a pasado a través de un sistema, se dice que se ha procesado la señal. En general, el sistema es caracterizado por el tipo de operación que es realizada en la señal. Esta operación es denominada el *procesamiento de la señal.*

Actualmente se considera que un sistema puede ser realizado como un dispositivo físico o como una pieza de software que será ejecutada en una computadora. En el procesamiento digital de señales, las operaciones ejecutadas en una señal consisten en un número de operaciones matemáticas especificadas por un programa de software. En este caso, el programa representa una implementación de un sistema en software. Por ejemplo, una computadora digital puede ser programada para realizar un filtrado digital. Alternativamente, el procesamiento digital de la señal puede ejecutarse por medio de hardware digital, que es configurado para realizar la operaciones especificadas. En dicha realización, se tiene un dispositivo físico que ejecuta las operaciones especificas. Más ampliamente, un sistema digital puede ser realizado por la combinación de hardware digital y software.

### 2.2.3 Sistemas de Procesamiento Digital de Señales

El procesamiento digital de señales es un área de las ciencias y la ingeniería que se ha desarrollado rápidamente en los pasados 20 años. Este rápido desarrollo es el resultado del significativo avance en la tecnología de las computadoras digitales y en la fabricación de circuitos integrados. Las computadoras digitales y su hardware digital asociado de dos décadas atrás eran relativamente grandes y caros y, como consecuencia, su uso fue limitado a cálculos de propósito general en el área científica y empresarial. El rápido desarrollo en tecnología de circuitos integrados a propiciado el desarrollo de poderosos, pequeños, rápidos y flexibles microprocesadores digitales y

hardware de propósito especial. Estos circuitos baratos y relativamente rápidos han hecho posible construir sistemas digitales altamente sofisticados que son capases de ejecutar funciones de procesamiento digital de señales en tiempo real muy complejas y tareas que son usualmente muy difíciles o caras de realizar en circuitos analógicos. Por lo que, muchas de las tareas de procesamiento digital de señales que son convencionalmente ejecutados por medios analógicos, son realizados actualmente por menos dinero y de manera más confiable en sistemas digitales.

Con esto no se afirma que el procesamiento digital de señales es la mejor solución para todos los problemas de procesamiento de señales. Efectivamente, para muchas señales con un ancho de banda muy grande el procesamiento en tiempo real es un requisito. Para este tipo de señales, el procesamiento analógico o también, el procesamiento óptico de señales son las únicas soluciones. No obstante, donde el procesamiento digital se puede aplicar es usualmente preferido.

Los circuitos digitales no solo son utilizados por ser pequeños y confiables para procesamiento de señales, si no que estos tienen otras ventajas. En particular, el hardware de procesamiento digital nos permite tener operaciones programables. A través de la implementación en software de un cierto algoritmo, nosotros podemos modificar las funciones que serán ejecutadas por el hardware. Este hardware digital y su software proveen un mayor grado de flexibilidad en el diseño del sistema. Además, se puede alcanzar un alto grado de precisión con el hardware y el software digital comparado con el procesamiento de señales con circuitos analógicos. Por todas estas razones, ha existido un explosivo crecimiento en la teoría de procesamiento digital de señales.

La mayoría de las señales encontradas en ciencias e ingeniería son del tipo analógico en su naturaleza. Esto es, las señales son función de una variable continua, tal como el tiempo o el espacio y usualmente toman valores continuos en un rango dado. Estas señales pueden ser procesadas directamente por un apropiado sistema analógico tal como un filtro o u analizador de frecuencias o un multiplicador de frecuencias con el propósito de cambiar sus características o extraer cierta información deseada. En este caso se dice que la señal ha sido procesada directamente en su forma analógica. La señal de entrada y la de salida están en forma analógica, tal como lo muestra la siguiente figura (2.2).


***Figura 2.2 Procesamiento Analógico de Señales***

Un procesador digital de señales puede ser una computadora programable muy grande o un pequeño microprocesador que es programado para ejecutar las acciones deseadas en la señal de entrada. Este también puede ser un procesador digital creado a partir de dispositivos discretos (lógicos) que es programado para ejecutar un conjunto especifico de operaciones en la entrada de la señal. Las maquinas programables otorgan una gran flexibilidad para cambiar la operación de procesamiento de la señal mediante un cambio en el software, independientemente del hardware. Por consecuencia, los procesadores de señales programables son muy comunes en la practica. El siguiente diagrama (2.3) presenta un sistema común de un sistema de procesamiento digital.


***Figura 2.3 Sistema de Procesamiento Digital de Señales***

Como se pude apreciar en el diagrama anterior, el sistema de procesamiento digital[3] tiene un bloque a la entrada de la señal analógica, el cual es denominado *convertidor analógico a digital* y a su salida tiene un bloque denominado *convertidor digital a analógico*.

Existen muchas razones por las cuales el procesamiento digital de una señal es preferido al procesamiento analógico directo de la misma, algunas de las cuales ya han sido mencionadas y a continuación ampliaremos. Primeramente, un sistema digital programable permite flexibilidad en la reconfiguración de la operación del sistema simplemente cambiando el programa. La reconfiguración de un sistema analógico usualmente implica un re diseño del hardware, pruebas y verificación de su correcto funcionamiento.

Consideraciones de precisión también juegan un papel importante en la definición de la forma del procesador de señales. El procesamiento digital de señales provee un mejor control de los requerimientos de precisión. La tolerancia en los componentes de los circuitos analógicos hace extremadamente difícil para el diseñador de sistemas, controlar la precisión de un sistema de procesamiento analógico. Estos requerimientos, consisten en especificar la precisión en el convertidor A/D y en el procesador de señales, en términos de longitud de palabra, aritmética de punto flotante contra punto fijo y otros factores similares.

Las señales digitales son fáciles de almacenar en un medio magnético (disco o cinta). Como consecuencia la señal será transportable y podrá ser procesada fuera de línea en algún laboratorio remoto. El método de procesamiento digital de señales permite la implementación de algoritmos de procesamiento de señales más sofisticados.

En algunos casos la implementación de un sistema de procesamiento digital es más barata que su contra parte analógica. El bajos costo puede hacer que efectivamente el hardware digital sea más barato, o que tal vez este resulte en un mayor grado de flexibilidad para las modificaciones provistas por la implementación digital.

# CAPÍTULO 3

## *ESTIMACIÓN ESPECTRAL CON TRANSPUTER'S*

### 3.1 Transputer

El Transputer es una arquitectura VLSI que soporta explícitamente concurrencia y sincronización. La diferencia entre un Transputer y una microcomputadora ordinaria, es que éste contiene un planificador de actividades (scheduler) integrado, capaz de distribuir su tiempo entre un número de procesos concurrentes. De esta forma el Transputer, además de ejecutar procesos en modo secuencial, puede ejecutar procesos concurrentemente. Varios Transputer's se pueden agrupar de una mane rápida y directa para formar redes y arreglos de procesadores. Cada Trasputer trabaja en su propia actividad y usa su memoria local. Para que los Transputer's puedan cooperar en un sistema necesitan comunicarse entre si, esto es realizado a través de interfaces seriales denominadas "link's", cada uno con un canal de entrada y otro de salida.

#### 3.1.1 La Arquitectura del Transputer

El diseño interno del Transputer[7] diverge con respecto al de cualquier otro microprocesador. Un concepto central en el Transputer son los procesos y como se ven estos a través del conjunto de instrucciones. Un proceso representa la un hilo de control individual y el Transputer cambia de un proceso a otro para proveer la ilusión de que estos se ejecutan de manera simultánea. Este cambio de un proceso a otro comúnmente se denomina multitarea y es normalmente manejado por los sistemas operativos, pero en el Transputer ésta es interconstruida totalmente por el hardware y por micro código en el procesador. La figura (3.1) presenta la arquitectura del Transputer.

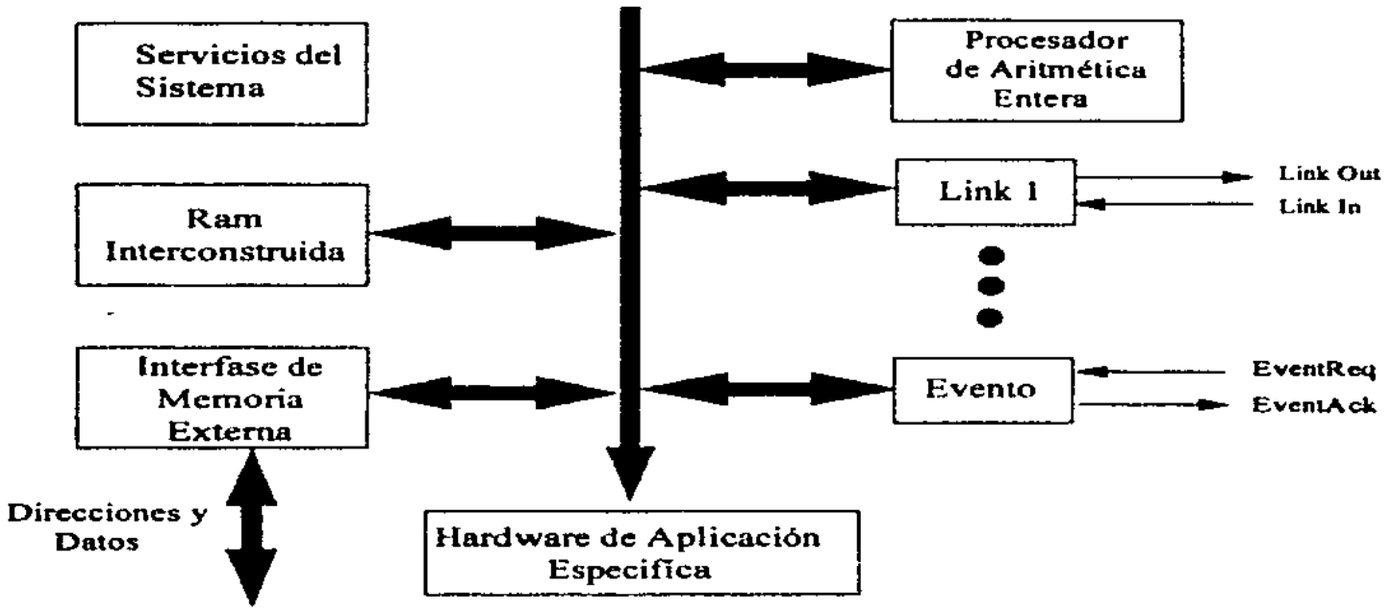



***Figura 3.1 Arquitectura Genérica de un Transputer***

Una de las principales características que hacen del Transputer un diseño muy especial, es la parte de hardware de comunicaciones que le permite interconectar otros Transputers para formar cualquier topología requerida por una aplicación especifica. La manera en como se comunican y como se conectan 2 o más Transputer's es muy fácil y flexible, ya que no se tiene que agregar soporte de hardware adicional, todo esta interconstruido en el mismo microprocesador.

La organización interna del Transputer juega un papel fundamental en el desempeño y en su capacidad para ejecutar más de una sección de código de manera concurrente. El modelo de registros que componen a un Transputer se muestran a continuación en la figura (3.2)

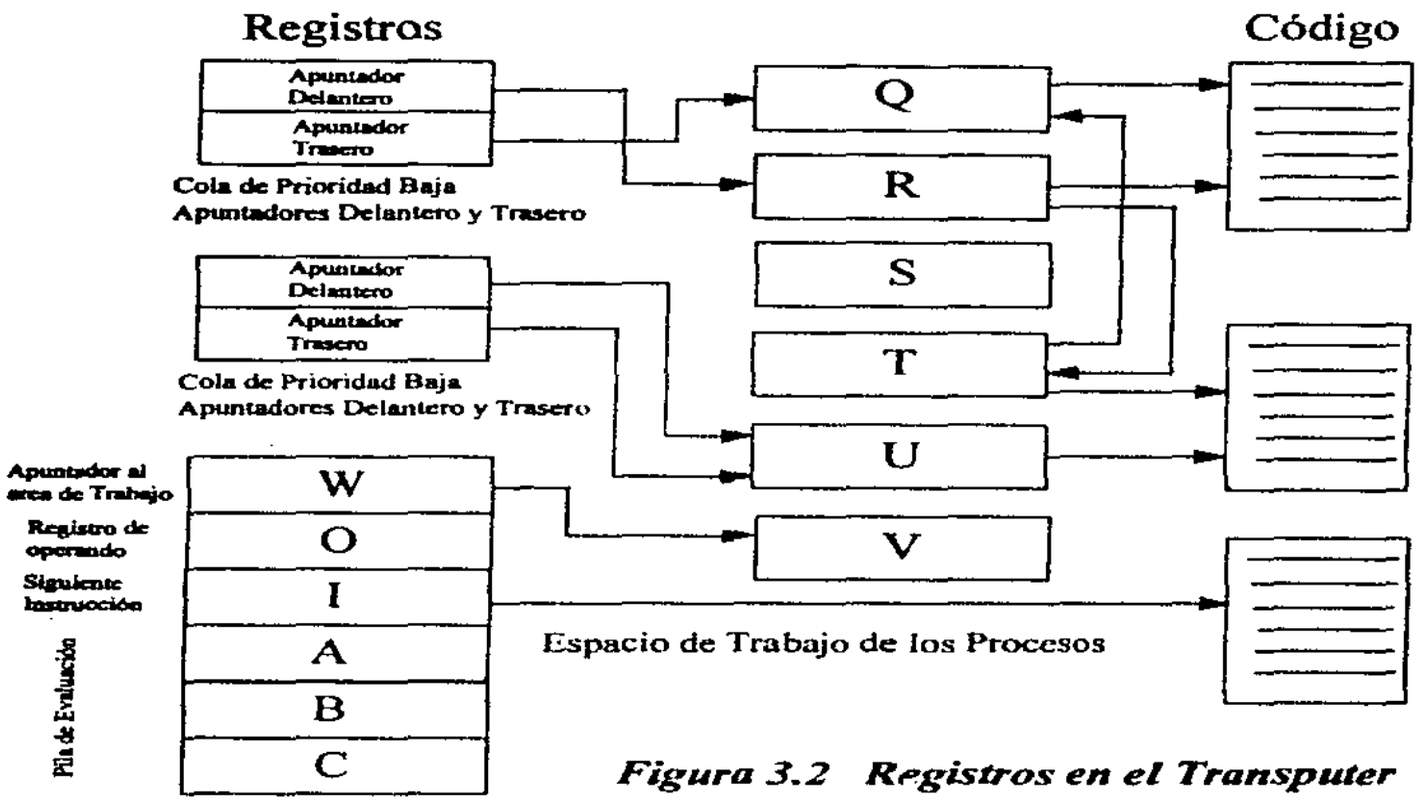



***Figura 3.2 Registros en el Transputer***

Todos los registros son de 16 o de 32 bits de largo, dependiendo de la longitud de palabra del Transputer. Los registros A, B y C de la pila de evaluación y las instrucciones del Transputer son diseñadas al rededor del uso de esta pila en vez del modelo clásico de registros de propósito general. La profundidad de la pila de tres registros es un compromiso entre la habilidad para evaluar la mayoría de las expresiones en la pila y el tener que salvar los menos registros posibles cuando ocurre un cambio de contexto a otro proceso.

El registro W es el apuntador al espacio de trabajo. Apunta a las variables locales asociadas con el proceso actualmente en ejecución. Muchas instrucciones refieren sus datos a través de un desplazamiento con respecto al registro W. El apuntador de instrucciones Y apunta a la siguiente instrucción a ser ejecutada. El registro de operando O es usado en la construcción de los operandos.

**A) Planificador de Tareas**

Una característica única del Transputer es que contiene un planificador en micro código el cual mantiene dos colas de procesos, una para baja prioridad y otra para alta prioridad. Los procesos en la prioridad más alta, son ejecutados hasta su terminación hasta que se requiere una petición de entrada/salida. No obstante, los procesos que se ejecutan en la prioridad baja, se les asigna una rebanada de tiempo, que cuando se consume, deja en su lugar al siguiente proceso que compite por la (UCP).

**B) Comunicaciones**

El Transputer ha sido diseñado para arquitecturas paralelas que utilizan la técnica de paso de mensajes y que tiene un gran soporte para comunicación entre procesos. El Transputer ha sido diseñado para que la diferencia en cuanto a la programación en la parte de comunicación entre procesos sea transparente sin importar si la comunicación se da entre dos procesos que se ejecutan en el mismo procesador o si ambos procesos residen en procesadores separados. Esta transferencia esta sincronizada a través de un área en memoria denominada *palabra de control de canal.* La comunicación solo se efectúa cuando el proceso que envía los datos y el proceso que los recibe están listos. Esto es, los procesos son sincronizados a través del paso de mensajes entre ellos. La comunicación entre dos procesos que residen en Transputer's diferentes utilizan el mismo mecanismo, la sincronización por medio de palabras de control que residen en las partes bajas de la memoria. Las mismas instrucciones son utilizadas para establecer la comunicación, y el controlador de los "links" de cada Transputer toman cuidado de que las transferencias que usan el acceso directo a memoria (DMA, Direct Memory Access) entre los links y la memoria, se lleve a cabo sin la intervención del procesador.

Los Transputers actualmente disponibles, tienen cuatro links externos. Los cuales cuentan con una comunicación full duplex lo cual hace posible intercambiar datos con otros Transputer's a un velocidad de 5, 10 o 20 Mega-bits por segundo (Mbps). La velocidad es elegida por el voltaje aplicado al Transputer.

Los datos son transmitidos a través de los links como un conjunto de bytes seriales. Cada byte comienza con un acknowledged (si reconozco) por el Transputer receptor. Ningún intento de detectar errores es hecho en el link; se asume que la comunicación se ha realizado sin errores, o que otras capas más externas en el nivel de comunicación detectaran y corregirán estos.

**C) Interrupciones**

El Transputer tiene una sola fuente de interrupciones externas, la entrada *EventReq*. La programación de la interface de esta entrada es implementada tal como si esta fuese otra palabra de control localizada en memoria. Un proceso es conminado a esperar en el canal y será removido de la cola de procesos listos para ejecutarse hasta que EventReq sea habilitada y entonces el proceso será reprogramado para ejecución. Cuando esto ocurre el evento EventAck es habilitado. Para proveer un rápido tiempo de respuesta para atender una interrupción (como en cualquier procesador, el proceso queda en un estado de espera cuando ésta ocurre), el proceso debe ejecutarse en prioridad alta y debe ser el único proceso de prioridad alta que se este ejecutando. Si estas condiciones son cumplidas, en el peor de los casos, el tiempo de respuesta es de cerca de cinco ciclos de reloj del procesador. Este rápido tiempo de respuesta es obtenido haciendo que las instrucciones largas puedan ser interrumpidas.

**D) Memoria**

Los miembros actuales de la familia de Transputer's tienen 2 o 4 Kbytes de RAM estática de alta velocidad dentro del mismo procesador, la cual ocupa la parte más baja del espacio de memoria. Parte de esta memoria es reservada para funciones en microcódigo del procesador. Esta es una inadecuada cantidad de memoria para la mayoría de las aplicaciones, pero el Transputer contiene un interface de memoria externa. El acceso a la memoria interna es muy rápido y solo consume un ciclo de reloj del procesador. El acceso a la memoria externa es más lento y consume 2, 3, 4 o 5 ciclos del procesador en el caso de utilizar memorias dinámicas.

Las ocho palabras más bajas de la memoria contienen las palabras de control para cada link externo, con EventReq en la novena palabra. Las siguientes dos palabras corresponden a los apuntadores delanteros para los temporizadores de las colas de prioridad alta y baja.

### 3.1.2 Lenguajes de Programación

Cuando el Transputer fue desarrollado por INMOS en el año de 1986, existía solo un único lenguaje para programarlo, OCCAM [14]. Un lenguaje de alto nivel que fue desarrollado junto con el Transputer. Se dice que OCCAM es el mejor lenguaje para programar en el Transputer ya que la arquitectura del software de OCCAM está muy relacionada con la arquitectura del Transputer mismo. No obstante, existen otros lenguajes que contienen otras ventajas que no están incluidas en OCCAM. Por ejemplo, OCCAM no maneja memoria dinámica, recursión, estructuras de datos (a excepción de la entrada y salida), y tampoco tiene tipos definidos por el usuario. Sin

embargo existen otros lenguajes para el desarrollo de aplicaciones basadas en el Transputer. Tal es el caso de el lenguaje "C"[27][28] y de FORTRAN entre otros. Lenguajes como estos últimos son muy populares debido a que existen muchas aplicaciones que están escritas en alguno de estos lenguajes y que para el caso de OCCAM, el código puede ser inexistente. Aunque hay que resaltar que FORTRAN y C, a diferencia de OCCAM, no contienen ningún soporte para la programación en paralelo y por lo tanto esta debe ser adicionada en alguna de dos formas: por medio de extensiones al lenguaje o a través de soporte en bibliotecas. La segunda opción es la más comúnmente elegida ya que es más fácil de implantar que el agregar nuevas características a un lenguaje.

### 3.1.3 Sistema de Desarrollo Basado en Transputer's

Actualmente las tarjetas de Transputer's pueden ser insertadas directamente en sistemas computacionales ya existentes (add-on boards) tales como: estaciones de trabajo Sun, SGI, DEC o en plataformas de tipo PC, las cuales se utilizan para un rango amplio de aplicaciones y en configuraciones de uno o más procesadores. La naturaleza de su respuesta en tiempo real hace posible utilizarlo en sistemas de simulación de componentes lógicos (tales como los circuitos VLSI).

Los fabricantes de tarjetas tipo Transputer, suministran los llamados "módulos" *TRAMS*. Los *TRAMS* son módulos que contienen un Transputer y algunos otros circuitos tales como memoria RAM o algún controlador para un dispositivo determinado. El utilizar estos sistemas modulares tiene una gran ventaja: *Expandibilidad*. Los usuarios pueden expandir sus sistemas al mismo tiempo que crecen sus necesidades.

Los sistemas basados en Transputer's tipo "add-on", cuentan con los servicios necesarios para comunicarse con el computador en el que residen. La interface con los servicios del sistema incluye las señales necesarias para inicializar y configurar la red de Transputer's, determinar la velocidad del reloj del procesador y de sus links para poder captar las señales de error y para responder a eventos externos.

La filosofía que comúnmente se sigue para comunicarse con las aplicaciones que se ejecutan en un sistema de procesamiento paralelo basado en Transputer's, es la de conectar dicho sistema a un servidor que se encargara de la interacción con el mundo exterior. Esto es, el servidor se encargara de controlar el envío y recepción de los distintos periféricos que se requieran por parte de la aplicación, para poder enviar o recibir datos del mundo exterior.

En la siguiente figura se muestra como se integran diversos periféricos a un sistema basado en Transputer's.

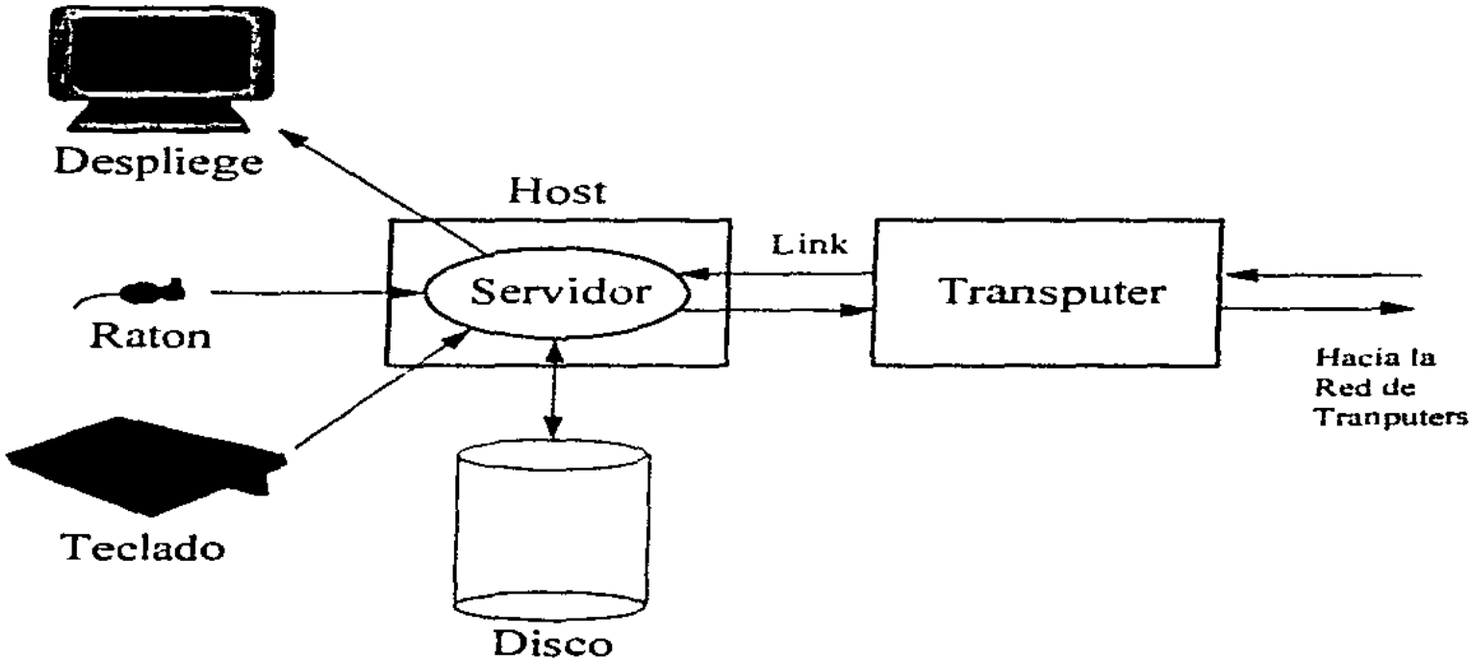



***Figura 3.3 Sistema de Desarrollo para Transputer's***

El camino más fácil para proveer la interacción de la plataforma con otros periféricos es el de utilizar la máquina en donde ha sido insertado el sistema basado en Transputer's conocido como *computador anfitrión* (Host), ejecutando un programa que sirve de enlace de comunicación con el sistema de Transputer's. El anfitrión. En el caso de una plataforma de procesamiento paralelo basada en Transputer's, el anfitrión debe esta conectada a través de un "link" de comunicación del Transputer raíz. Lo que quiere decir que si algún otro proceso que no reside en el Transputer raíz, y desea comunicarse con el computador anfitrión, deberá enviar la información al Transputer raíz para que el proceso que reside en éste, pueda a su vez transmitirla al computador anfitrión.

## 3.2 Transputer's en Estimación Espectral

En está sección se describe el uso de la arquitectura del Transputer en la implementación de un método de estimación espectral basado en el algoritmo de la DFT.

### 3.2.1 Sistema de Desarrollo Basado en Transputer's

Este método consiste en el cálculo de la PSD a partir de la implementación del algoritmo de FFT elegido, en un sistema de Transputer's. En el apéndice C describimos el algoritmo de la FFT utilizado. La figura 3.4 muestra el diagrama a bloques del algoritmo de estimación espectral.

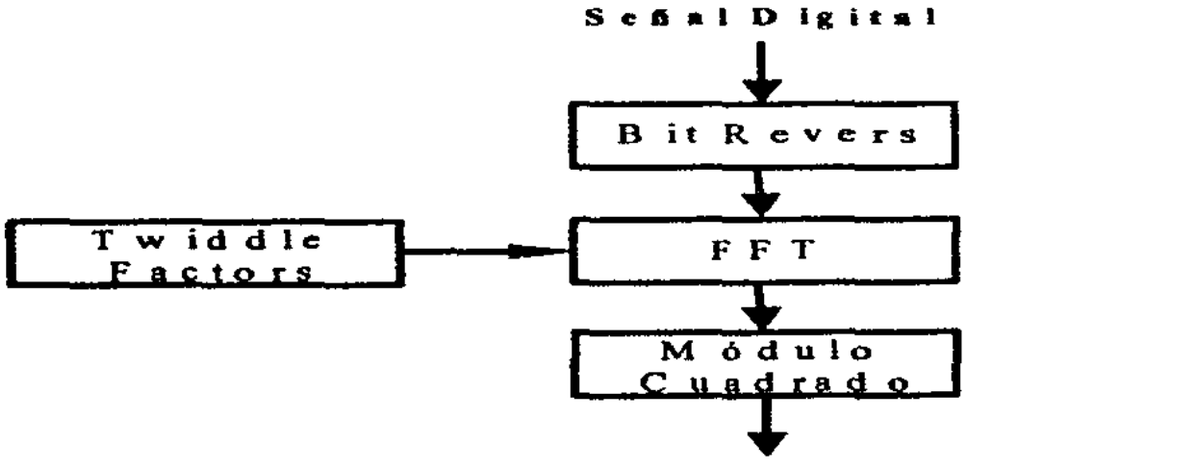



*Figura 3.4 Diagráma de Bloques del Método de Estimación Espectral*

Primeramente se realiza el cálculo de la tabla de senos y cosenos correspondiente a los factores de corrimiento o "twiddle factors", los cuales serán almacenados en una matriz para ser utilizados en el algoritmo.

Posteriormente, se asume que los datos entran de forma ordenada hacia el proceso que calculará el algoritmo de FFT. Por lo tanto, se procede a calcular los "bit-

reverse" de la secuencia de datos de entrada. Esto implica que los datos serán ordenados en modo de "bit-reverse" para que a la salida, los datos estén en sus posiciones originales de entrada. Esto es, ordenados.

A continuación, se aplica el algoritmo de FFT a la secuencia de datos ordenada en "bit-reverse", con lo cual obtenemos las componentes espectrales de los datos de la señal de entrada. Los datos son entregados en forma de una secuencia de números complejos, una parte real y una parte imaginaria por cada dato. De esta secuencia, podemos despreciar la mitad de los datos ya que estos son "espejo" de la otra mitad complemento.

El paso final consiste en calcular la Densidad Espectral de Potencia (PSD, Power Spectral Density) mediante el cálculo del modulo cuadrado de cada uno de los datos arrojados por la FFT.

### 3.2.2 Implementación con Transputer's

El modelo computacional utilizado para la implementación paralela de éste algoritmo, considera el procesamiento de la señal de un número de segmentos consecutivos de la señal, en forma simultánea mediante el uso de *n* Transputer's, uno por cada segmento. La figura 3.5 describe el modelo computacional para implementar el método de estimación espectral utilizando cuatro procesadores, en donde el denominado "maestro", recibe cuatro segmentos consecutivos de la señal a procesar, de los cuales distribuye tres a los procesadores "trabajadores". Cada procesador calcula la PSD del segmento correspondiente. Finalmente, el procesador maestro colecta y combina las componentes espectrales generadas por los trabajadores para integrar, junto con el que éste proceso calculó, el espectro final correspondiente a los segmentos de datos de entrada.

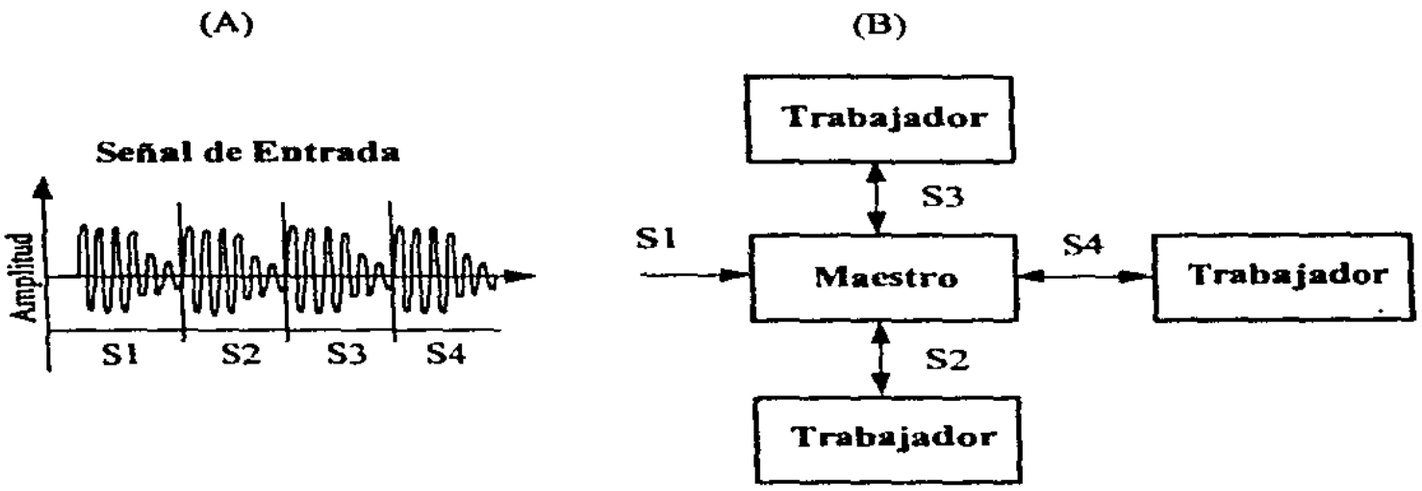



*Figura 3.5 A) Señal de Entrada*
*B) Modelo Computacional para Realizar la Estimación Espectral de Cuatro Segmentos de la Señal*

# CAPÍTULO 4

## *INSTRUMENTACIÓN VIRTUAL CON TRANSPUTER'S*

### 4.1 Instrumentación Virtual

Hasta hace poco tiempo, la instrumentación virtual era inexistente[25]. Es a raíz de que las computadoras personales y sus periféricos han bajado sus precios, haciendo que estas sean utilizadas en aplicaciones de control e instrumentación. Este cambio se debe a que en la actualidad, la mayoría de los instrumentos reales (todos aquellos instrumentos físicos que son creados con un propósito especial y solo sirven para dicho propósito) conlleva un proceso de desarrollo muy largo y que consiste de:

- **Investigación de mercado**
- **Desarrollo de un Prototipo Funcional**
- **Pruebas del Sistema y Re diseño (En la parte electrónica o el hardware)**
- **Fabricación en Gran Escala**
- **Futuros Desarrollos**

Los puntos anteriores se aplican a instrumentos comerciales y un subconjunto de éstos se aplica en el desarrollo de instrumentos utilizados en la investigación o para solucionar un problema particular. Como podemos apreciar, se requiere de una gran infraestructura tanto humana como de recursos materiales, lo cual es muy costoso, y tiempo para poder desarrollar un instrumento real que cumpla nuestras necesidades específicas.

Actualmente, la instrumentación virtual ha encontrado cavidad al resolver muchos de los problemas a los que nos enfrentaríamos si tuviéramos que crear el instrumento real.

- **Fácil y Rápida Creación de Aplicaciones**
- **Prueba de Sistema y Flexibilidad en el Re diseño (En el software)**
- **Mayor Calidad en la Visualización de la Información**
- **Costos muy Reducidos**

Por estas razones en poco tiempo es muy probable que la instrumentación virtual inunde los principales mercados de instrumentación y control haciendo que esta tarea sea más rápida y más barata que antes.

Un IV consta de dos partes fundamentalmente: La primera contiene la *Aplicación* y se encarga del proceso de la información correspondiente. La segunda se compone de una *Interface Gráfica* con la cual el usuario puede interactuar directamente.

Uno de los requerimientos actuales cuando se construye un IV, es el de utilizar una interface con el usuario que sea lo más amigable posible, más fácil de manejar que un instrumento real, y que sea altamente configurable por el usuario. En la sección 4.2 se dan a conocer las herramientas que se han utilizado en el presente trabajo para el diseño e implmentación de un IV basado en un sistema de Transputer's.

## 4.2 Transputer's e Instrumentación Virtual

Como se mencionó anteriormente, en el lado de la plataforma de procesamiento paralelo, se ejecuta la *aplicación*, en donde se calculan los distintos segmentos de datos proporcionados a está. En el caso del presente trabajo, se calcula la PSD. Dichos datos serán entregados al proceso de despliegue, que a su vez, después de ciertos cálculos, los retransmitiría hacia la *interface gráfica* de usuario del computador anfitrión. Para poder establecer el vínculo entre el proceso de despliegue y la interface gráfica, se deben tomar como base un cierto "protocolo" de programación, previamente establecido, conocido como API (Aplication Program Interface) y está interconstruido dentro del *WSP* (*Windows Server Package*).

Cuando utilizamos una plataforma de procesamiento paralelo basada en Transputer's, teniendo una computadora personal como anfitrión para interaccionar con el mundo exterior, necesitamos un programa que sirva como interface entre el computador anfitrión y la plataforma de Transputer's. Dicho programa, el *ISERVER* (que es distribuido junto con la plataforma de Transputer's), es el que se encarga de interpretar los mensajes u ordenes enviadas desde la plataforma hacia el computador anfitrión. Así, para poder enviar un carácter a la pantalla del anfitrión, la aplicación que existe en el Transputer envía la orden especifica para llevar a cabo esta tarea y el ISERVER se encarga de procesar la petición en el computador anfitrión. Este programa se ejecuta directamente en el computador anfitrión y por lo tanto es capaz de manejar todas las facilidades del medio ambiente en donde se ejecuta (todas las que los programadores de esta aplicación, el ISERVER, le hubiesen interconstruido). Para poder llevar a cabo la mediación entre el Transputer y el anfitrión se deben cumplir la siguiente condición: ***La única Aplicación que se esta ejecutando en el Transputer, que***

*puede comunicarse con el computador anfitrión, es aquella que se esta ejecutando en el Transputer raíz.*

Con el ISERVER, tenemos acceso a la mayoría de las funciones que se pueden realizar en el computador anfitrión, pero para poder crear un instrumento virtual, este conjunto de funciones esta limitado pues el despliegue en pantalla solo soporta un ambiente en modo texto. Para cumplir con uno de los objetivos del presente trabajo, es necesario que el despliegue de nuestra aplicación se realice en un ambiente en modo gráfico. El software para servidor windows, descrito en la sección (4.3), ha sido utilizado para este propósito.

## 4.3 Software para Servidor Windows

Existe actualmente una versión del programa ISERVER para Windows, el *WSP* fabricado por Elcom Ltd. en Moscow, Rusia[22][23][24]. Esta versión consiste de una aplicación de las capacidades del ISERVER en cuanto a lo que despliegue se refiere, pues tiene la capacidad de interactuar con el ambiente *Windows* de Microsoft, el cual es un ambiente gráfico y por cierto el más comercial que existe para plataformas de computación personal. Esta es la herramienta de desarrollo que ha sido propuesta en el presente trabajo y en el apéndice "A" se dan detalles de sus características y requerimientos.

### 4.3.1 Revisión del Ambiente Windows

El medio ambiente Windows[21] tiene características que no contiene el clásico ambiente DOS, en el que se ejecuta el ISERVER. Por tal motivo, la programación para Windows contiene un grado de complejidad mayor. Las principales características se listan a continuación:

- **Interface de usuario**
- **Manejo de una cola de entrada**
- **Gráficas independientes del dispositivo**
- **Multitarea**
- **Intercambio de datos entre aplicaciones**

Cuando se escribe una aplicación para el ISERVER, la mayoría de los programas usan la biblioteca estándar de C para enviar su salida, entrada, manejo de memoria, acceso a archivos y otras actividades. La biblioteca estándar de C asume un ambiente operativo estándar que consiste de una terminal basada en modo carácter para la entrada y salida, y acceso exclusivo a la memoria del computador anfitrión, como también el control propietario sobre los dispositivos de entrada y salida del anfitrión.

Bajo el ambiente Windows, las afirmaciones anteriores ya no son validas. Las aplicaciones, ya sea las creadas para el Transputer con el WSP o cualesquiera otras creadas exclusivamente para el ambiente Windows, comparten los recursos del computador anfitrión, incluyendo la UCP. Las aplicaciones Windows interactuan con el usuario a través del despliegue gráfico, el teclado y el ratón del computador anfitrión. A continuación profundizaremos en las principales características de este medio ambiente

### 4.3.2 Interface de Usuario

Uno de los principales puntos del diseño de aplicaciones Windows, es proveer un acceso visual a todas, las aplicaciones que estén ejecutandose en un momento dado. En un ambiente multitarea es importante dar a las aplicaciones una parte de la pantalla; esto asegura que el usuario puede interactuar con todas las aplicaciones. Una aplicación comparte la pantalla con otras aplicaciones al utilizar una *ventana* para interaccionar con el usuario. Técnicamente hablando, una ventana es más que una porción rectangular de la pantalla del computador anfitrión en donde la aplicación se ejecuta. En realidad, una ventana es una combinación de dispositivos visuales completos, tales como menús, controles, barras de desplazamiento, etc. que el usuario utiliza para dirigir las acciones sobre la aplicación. En el ambiente estándar en el que se desenvuelve el ISERVER, el sistema automáticamente prepara la pantalla del computador anfitrión para la aplicación en curso.

Otra ventaja de desarrollar aplicaciones en el ambiente Windows es que, en contraste con un programa escrito en C estándar, el cual tiene acceso solamente a una pantalla, las aplicaciones Windows pueden crear y usar cualquier número de ventanas sobrepuestas para desplegar información en muy distintos formatos (texto, gráficas, imagenes).

Como se mencionó antes, la programación para un ambiente gráfico como Windows, es más compleja, pero a cambio se obtienen más facilidades parael usuario. Ademas, muchas de las características del ambiente gráfico de Windows se manejan de forma automática, tal como impedir que dos aplicaciones accesen la misma parte de la pantalla al mismo tiempo.

### 4.3.3 Cola de Entrada

Una de las más grandes diferencias entre las aplicaciones desarrolladas para el ISERVER y las aplicaciones WSP es el la manera en que un programa recibe la entrada del usuario. En el ambiente del ISERVER, una aplicación que lee de el teclado hace una referencia explícita a una función de la biblioteca estándar de C tal como *getchar*. La función típicamente espera hasta que el usuario presione una tecla para retornar el código del carácter a la aplicación. En contraste, en el ambiente Windows, y por lo tanto en el ambiente WSP, se reciben todas las entradas del teclado, el ratón y el temporizador del computador anfitrión y coloca las entradas en la *cola de mensajes* de la aplicación apropiada. Cuando la aplicación esta lista para recuperar su entrada, simplemente lee el siguiente mensaje de su cola de mensajes.

En el ambiente estándar del ISERVER, la entrada es típicamente en el formato de caracteres de 8-bit de largo desde el teclado. Las funciones de la biblioteca de C estándar getchar y fscanf, leen caracteres desde el teclado y retornan su código ASCII correspondiente a la tecla presionada. Un programa también puede interceptar interrupciones desde dispositivos de entrada tales como el ratón o el temporizador para utilizar la información provista por estos dispositivos.

En Windows, una aplicación recibe la entrada en la forma de un *mensaje de entrada* que windows envia. Un mensaje de entrada contiene información que excede en mucho el tipo de información disponible en el ambinete estandar del ISERVER en combinación con DOS. Este mensaje especifica el tiempo del sistema, la posición del ratón, el estado del teclado, el código de busqueda de la tecla presionada (si ha sido presionada), el botón del ratón que fue presionado y el dispositivo que genero el mensaje. Por ejemplo, hay dos mensajes del teclado, WM_KEYDOWN y WM_KEYUP, que corresponden al momento en que una tecla es presionada y liberada, respectivamente. Con cada mensaje del teclado, es provisto un código virtual independiente del dispositivo, que identifica la tecla, el código de busqueda dependiente del dispositivo generado por el teclado, como tambien el estatus de otras teclas como SHIFT, CONTROL, y NUMLOCK. Los mensajes del mouse y del temporizador tienen el mismo formato y se manipulan de la misma manera.

### 4.3.4 Gráficas Independientes del Dispositivo

Dentro del ambiente de Windows, tenemos acceso a un gran conjunto de operaciones gráficas independientes del dispositivo. Esto significa que la aplicación puede dibujar fácilmente líneas, rectángulos, círculos y regiones complejas. Gracias a la independencia del dispositivo que Windows provee, se pueden utilizar las mismas funciones para pintar un circulo en una impresora de matriz de puntos o en una pantalla de alta resolución.

Windows, al igual que la mayoria de los sistemas operativos, requiere de un *manejador de dispositivos* para convertir la salidad gráfica a la salida hacia la impresora, la pantalla o un plotter. Un manejador de dispositivo es una biblioteca ejecutable especial que la aplicación puede cargar y conectar a un dipositivo específico o a un puerto. Un *dispositivo de contexto* representa el manejador del dispositivo, el dispositivo de salida y a veces, el puerto de comunicación.

### 4.3.5 Multitarea

Windows es un sistema multitarea en donde un número de aplicaciones pueden estar corriendo al mismo tiempo. En el ambinete DOS, no existe este concepto a menos que sea implementado por el usuario mismo, lo cual es otro proyecto a parte de la aplicación. Los programas escritos para DOS asumen que tienen el control absoluto de todos los recursos del computador anfitrión, incluyendo los dispositivos de entrada y salida, memoria, pantalla del sistema y también la UCP. En Windows no obstante, las aplicaciones deben compartir los recursos con las otras aplicaciones que se estan ejecutando. Debido a esto, Windows debe controlar el acceso a cada recurso muy cuidadosamente, y requiere a las aplicaciones utilizar cierta interfase de programación que garantice el control de dichos recursos.

En el ambiente estandar de DOS, un programa tiene acceso a toda la memoria no utilizda por el sistema, por la aplicación o por los programas TSR (terminate-but-stay-resident). Esto significa que la aplicación puede utilizar el resto de la memoria en al manera que más le convenga. En las aplicaciones Windows, la memoria es un recuso compartido. Ya que más de una aplicación puede estar ejecutandose al mismo tiempo, cada aplicación debe cooperativamente liberar memoria para no agotar los recursos existentes.

Las aplicaciones pueden reservar memoria al subsistema de manejo de memoria, de dos fuentes distintas: memoria global, para grandes aplicaciones, y memoria local, para pequeñas aplicaciones. Para hacer más eficiente el uso de la memoria, Windows a menudo mueve o también descarta bloques de memoria. Esto significa que no podemos saber con precision si un bloque de memoria que fue reservado, permanecerá en el mismo lugar o en cualquier otro. Entre más aplicaciones existan ejecutándose al mismo tiempo, Windows permanecerá moviendo o descartando bloques de manera constante.

Otro ejemplo de compartición de recursos es la pantalla del sistema, ya que ésta debe ser compartida por diversas aplicaciónes, no podemos desde una aplicación dada, cambiar el modo de video ni apropiarnos del mismo, tampoco podemos accesar directamante a la memoria de video si no que dejamos ese control a Windows.

## 4.4 Funciones del Servidor Windows (WSP)

El WServer trabaja bajo el ambiente gráfico de Windows 3.1/Windows para trabajo en grupo. Este se encarga de realizar las siguientes funciones básicas:

- Cargar los programas en el hardware del Transputer
- Proveer el ambiente en tiempo de ejecución mediante el cual los programas puedan comunicarse con el computador host.
- Servir a los programas que utilizan la biblioteca TWL (Transputer Windows Library) (una interface API completa para windows, llamadas DPMI (Dos Protected Mode Interfase), funciones de manipulación de recursos, acceso a memoria en el host (virtual), operaciones de entrada/salida en archivos, despacho de mensajes, etc)
- Establecer la comunicación entre los programas de la red de transputer's y las aplicaciones windows a través del portapapeles o mediante DDE (Dynamic, Data Exchange, Intercambio Dinámico de Datos).

En el nivel de aplicación, todas las comunicaciones con el host son a través de la biblioteca estándar de entrada salida. El servidor host provee una interface intermedia mediante la cual las funciones de entrada y salida se comunican con el WServer. Dicha interface esta basada en un protocolo fijo y es implementada por un conjunto de funciones escritas en lenguaje "C". Actualmente existen cerca de 170 funciones distintas en las TWL. Para un listado y una explicación detallada de dichas funciones, refiérase a [23].

Antes que un programa sea cargado en la red de transputers, este debe ser compilado y enlazado con las bibliotecas necesarias. Además, se debe configurar el WServer para aceptar el código de la tarjeta de transputers en la cual se desea ejecutar la aplicación (para más detalles acerca de las tarjetas soportadas, ver el apéndice A).

## 4.5 El Modelo de Programación de Windows

La mayoría de las aplicaciones windows utilizan los siguientes elementos para interactuar con el usuario:

• Ventanas: Una ventana es el dispositivo de entrada/salida primario de cualquier aplicación Windows. Esto quiere decir que cualquier aplicación puede accesa al sistema de despliegue. Una ventana es una combinación de un titulo para ésta, un menú, barras de desplazamiento, bordes y demas características que ocupan el "rectángulo" de la aplicación. El programador especifica dichas características cuando la ventana es creada.

• Menús: Un menú el el principal medio de entrada/salida de una aplicación Windows. Un menú es una lista de comandos que el usuario puede ver y seleccionar. Cuando se crea un menú, el programador elige los nombres y los comandos del mismo.

• Cajas de Diálogo: Una caja de diálogo es una ventana temporal que puede ser desplegada para que el usuario pueda suministrar mayores datos a la aplicación dada. Una caja de diálogo contiene uno o más controles. Un control es una pequeña ventana que realiza una entrada/salida de datos simple tal como capturar un nombre

• El ciclo de Mensajes (message loop): Una vez que la aplicación ha recivido una entrada de información, esta es alimentada a una lista de mensajes. Dicha lista es vaciada y cada comando es procesado dentro de un ciclo denominado ciclo de Mensajes. Cada mensaje es debidamente enviado/recibido de la ventana en donde se origino.

Las referencias [21][22][23][24] dan una buena explicación de los tópicos anteriores.

El capítulo 5 describe el diseño y desarrollo de un IV basado en una plataforma de procesamiento paralelo en donde se utiliza el software WSP para crear la interface gráfica de usuario de dicho instrumento.

# CAPÍTULO 5

# *CASO DE ESTUDIO: DETECTOR DE FLUJO SANGUÍNEO*

## 5.1 Introducción

En estudios médicos de ultrasonido, las señales Doppler de ultrasonido son consecuencia del propio movimiento de la sangre, debido a que un cambio en la frecuencia de la señal Doppler de ultrasonido depende de la velocidad de este movimiento. Por tanto la información que nos presentan las variaciones en frecuencia de la señal ultrasónica corresponden a la velocidad de movimiento de cierta clase de partículas que conlleva el flujo sanguíneo. Es natural preguntar como la velocidad puede estar relacionada a otras variables psicológicas , la presión, razón de flujo y resistencia por ejemplo. ¿Que patrones de velocidad deben ser considerados normales para una vena dada y como deben de cambiar en presencia de una lesión que invade la cavidad de la vena?. A continuación describiremos algunos de los conceptos involucrados en la descripción del flujo sanguíneo y en donde entra en juego la interpretación clínica del análisis de señales de flujo sanguíneo obtenidas. Así como también una de las partes fundamentales de este trabajo: *El detector de flujo Sanguíneo.*

## 5.2 Lesiones y Problemas en las Arterias

### 5.2.1 Dinámica de una Obstrucción

La propiedad de un fluido que determina la dirección y la velocidad del flujo es su energía total [19][20]. La energía de un fluido puede tomar diversas formas, las cuales son generalmente combinadas en situaciones reales:

1. *Energía de Presión*: La presión en un fluido puede pensarse como la habilidad para realizar un trabajo(que sobrepasa la resistencia debido a la viscosidad) por lo que forma una energía potencial.

2. *Energía Cinética*: Cada volumen de fluido en movimiento tiene una energía en virtud de su masa y de su movimiento, determinado por $\frac{1}{2}\rho v^2$, donde $\rho$ es la densidad y $v$ es la velocidad del flujo.

3. *Energía Potencial Gravitacional*: Un fluido en un nivel más alto en un punto arbitrario es capaz de realizar un trabajo; su peso imparte una energía potencial igual a $\rho gh$, donde $h$ es la altura, $\rho$ la densidad y $g$ la aceleración de la gravedad.

4. *Energía Viscosa*: El flujo a través de una resistencia debida a su viscosidad puede producir una perdida de engría, en la cual al energía cinética o potencial del fluido es transformada en energía cinética.

Si se considera una sola línea por donde corre un flujo, según el principio de la conservación de la energía, se tiene que la suma de las distintas energías individuales, la energía total $E$, permanecerá constante en cada punto a lo largo de la trayectoria del flujo. Esto podemos representarlo como:

$$E = P + \rho gh + \frac{1}{2}\rho v^2 + R_L = Constante \qquad (5.4)$$

Donde $R_L$ es la energía perdida debido a la viscosidad del fluido. Esta es la forma de la ecuación de Bernoulli, una representación descendiente de las leyes de Newton clásicas de mecánica aplicada a fluidos. La aplicación de estos principios fundamentales de mecánica de fluidos es de gran ayuda en el entendimiento de situaciones clínicas, como el flujo de la sangre a través de una lesión [16][17].

Al analizar la energía total del flujo sanguíneo pasando a través de una lesión y aplicando el principio de Bernoulli, es posible predecir los patrones de cambio de velocidad y por lo tanto las características del fluido en su viaje a través de venas tan importantes como la carótida o la orta. Esto también explica por que basados en un criterio acerca de la velocidad resulta más sensible que el medir, por ejemplo, la presión de la sangre. Además proporciona una indicación de como la medición de la velocidad por métodos no invasivos puede ser utilizada para determinar el punto en el cual la presión y el flujo se reducen indicando esto una posible lesión.

### 5.2.2 Características para el Análisis del Flujo Sanguíneo

Las mediciones de las señales de ultrasonido se utilizan de primera instancia para detectar y medir los efectos de una estenosis (la contracción del diámetro de una vena o arteria) y oclusiones.

Una estenosis es presentada en forma de diagrama en la figura (5.1). La estenosis afecta la velocidad, el flujo sanguíneo, la presión, crea turbulencias en el flujo y puede

generar una embolia. Una estenosis se forma preferentemente, pero no exclusivamente, en las ramificaciones de las venas (por ejemplo, la ramificación de la carótida a las arterias carótidas interna y externa, y a la unión de la orta - iliac).

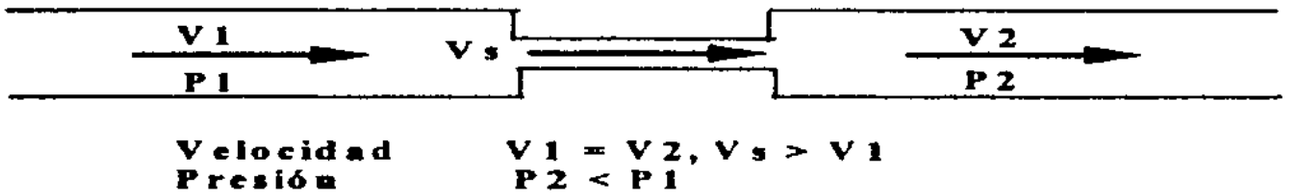



*Figura 5.1 Comportamiento de la Presión y Velocidad en las Vecindades de una Lesión*

Cuando una estenosis incrementa su severidad (el diámetro de la vena decrece) existe un progresivo incremento de presión en la periferia de la estenosis, un decremento en el flujo de la sangre dentro de la vena y un incremento en la velocidad del flujo sanguíneo en las vecindades de la estenosis.

Nótese que se requiere un grado avanzado de la estenosis antes que la cantidad de flujo sanguíneo descienda considerablemente (el incremento de presión rompe la tendencia para mantener la velocidad de flujo). Existe un incremento en la velocidad dentro de la vena o arteria alrededor de la estenosis debido a que el mismo volumen de sangre pasa por el área de una sección transversal más pequeña. Conforme el diámetro de la vena disminuye, la velocidad se incrementa hasta un punto y entonces cae rápidamente, alcanzando un alto total, lo mismo que el flujo, y por lo tanto ocurre una oclusión completa. Se debe tener cuidado de la velocidad en los limites de la estenosis ya que parece normal hasta que se alcanza un alto grado de severidad. La turbulencia provocada por la combinación del flujo de alta velocidad que sale de la estenosis con el flujo de más baja velocidad después de esta (en dirección del flujo), y la alta velocidad provocada por la estenosis pueden ser detectadas no solo dentro de la misma, si no también a una distancia muy cercana (en la dirección del flujo) hasta que la turbulencia se disipe conforme el flujo avanza. La velocidad de la sangre dentro de las arterias tiene una naturaleza pulsátil como resultado de la acción de los latidos del corazón, y una estenosis afecta la forma de onda de la velocidad. La combinación de una resistencia que se incrementa al paso del flujo más la naturaleza elástica de la pared de una vena ahoga la oscilación de la velocidad pulsátil de la sangre.

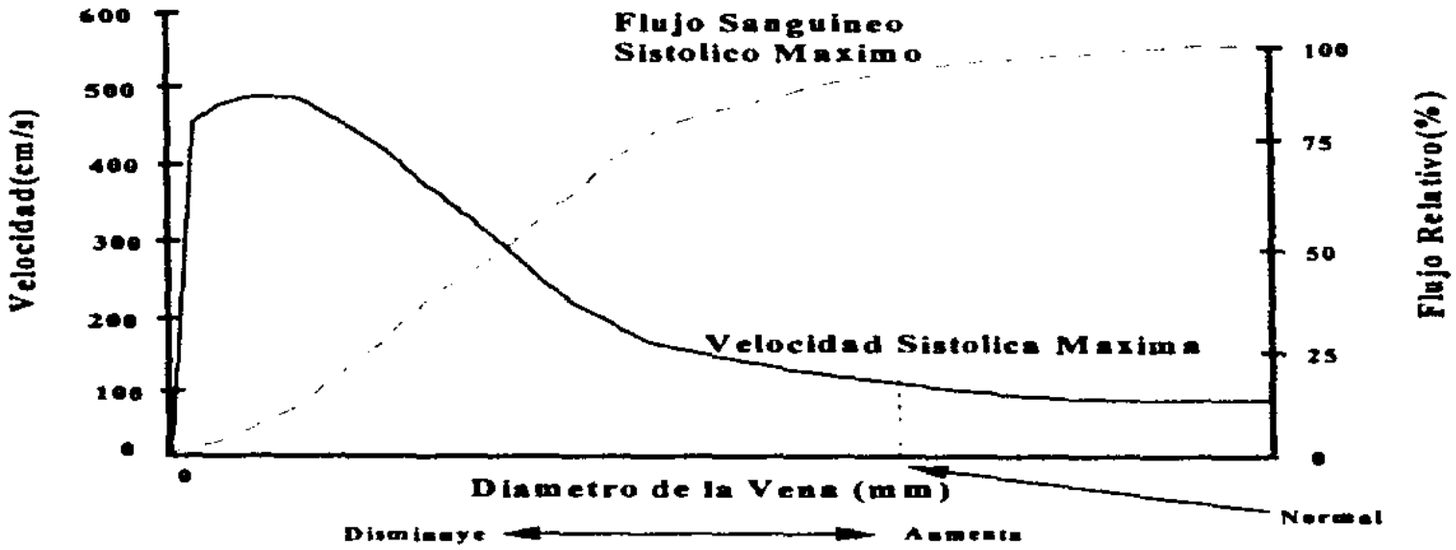



*Figura 5.2 Presión, Velocidad y Relación de Flujo en las Vecindades de una Lesión*

Aunque la estenosis no fuera suficientemente severa para causar una reducción significativa en el flujo sanguíneo, además de reducir el transporte de oxigeno y nutrientes, la lesión puede tener efectos severos al desintegrarse por si misma bajo la acción del flujo pulsátil dentro de la vena y por la flexión periódica de sus paredes. Éstas pequeñas partículas puede ser lo suficientemente grandes para bloquear o tapar pequeñas venas en las ramificaciones posteriores. Dichos bloqueos pueden no causar severos problemas en el flujo sanguíneo, pero pueden causar un infarto o una embolia cuando se presentan en venas que alimentan el cerebro.

El ultrasonido es utilizado para detectar la presencia de una estenosis y medir sus efectos[15][16] al detectar los síntomas presentados anteriormente. Existen diversas características que podemos obtener a través del análisis del flujo sanguíneo, tales como la presión sistólica (presión sistólica: la máxima presión de la sangre que ocurre durante la contracción del corazón), y la velocidad. El enfoque del presente trabajo esta orientado a obtener información de la velocidad del flujo sanguíneo a partir de sus componentes espectrales [18][20] y las distintas formas de onda de una vena o arteria sana comparadas con venas o arterias que presentan un cierto grado de estenosis.

Llegado a este punto podemos preguntarnos ¿de que manera podemos obtener la información necesaria acerca del flujo sanguíneo para realizar un análisis y detectar una posible lesión?.

Inicialmente, los datos son adquiridos a través de una sonda ultrasónica[15] que contiene un transmisor y un receptor integrados. Tanto la fuente de radiación de ondas (el transmisor), como el observador (el receptor) están estacionarios con respecto del medio. Cualquier *reflector* o *dispersor* de ultrasonido puede ser considerado tanto como el observador de la señal de ultrasonido transmitida, así como la fuente de ultrasonido para el receptor. Si el dispersor o reflector es estacionario, el receptor percibirá la misma frecuencia con la que la señal fue transmitida. No obstante, si el dispersor esta en movimiento, alejándose de la sonda, el observador de la señal trasmitida vera más bajas frecuencias que las que originalmente fueron transmitidas, y la fuente de emisión de señales de ultrasonido las vera con una longitud de onda más grande. En el caso del dispersor en movimiento, hay dos cambios en la señal ultrasónica en la misma dirección y el observador experimentara más bajas frecuencias que el transmisor. A la inversa, si el dispersor se mueve hacia la sonda, el observador experimentara más altas frecuencias que el transmisor. El cambio en la señal ultrasónica (la diferencia entre el observador y el transmisor) es proporcional a la velocidad del dispersor. En el ultrasonido médico se considera a los *glóbulos rojos* como el dispersor a ser tomado en cuenta. El por que de tomar en cuenta a los glóbulos rojos para obtener la información acerca de la velocidad del flujo sanguíneo radica en la composición de la sangre:

La sangre consiste de un plasma líquido con una densidad aproximada a la del agua ($1.02 g/cm^3$) donde las partículas celulares están suspendidas:

- Plaquetas: en grandes cantidades ($3.5 \times 10^5 / mm^3$) pero muy pequeños

- Leucocitos: grandes pero en poca proporción ($7.5 \times 10^3 / mm^3$)

- Eritrocitos: en grandes cantidades ($5.1 \times 10^6 / mm^3$) y medianas ($7 \times 2 \mu m$)

Los *glóbulos rojos* (o *Eritrocitos)* contabilizan cerca del 45% de el volumen de la sangre. A esta fracción se le conoce con el nombre de *hematrocitos*. Estos son los responsables de la *dispersión de las señales de ultrasonido por parte de la sangre* (dispersores).

### 5.2.3 Análisis Espectral de las Señales Doppler de Ultrasonido

El análisis espectral de las señales Doppler de ultrasonido, es comúnmente utilizado para detectar la presencia de lesiones estenóticas al indicar un incremento en la velocidad de flujo a través de la estenosis, junto con el incremento correspondiente de frecuencias (y con esto un cambio en el espectro de la señal así como un cambio en la forma de onda) [15][18][19] como resultado de los cambios en los rangos de velocidad de la sangre y los distintos ángulos entre el haz y la dirección del flujo en la región vecina a dicha lesión donde se presentan disturbios en el flujo sanguíneo (figura 5.3 y 5.4).

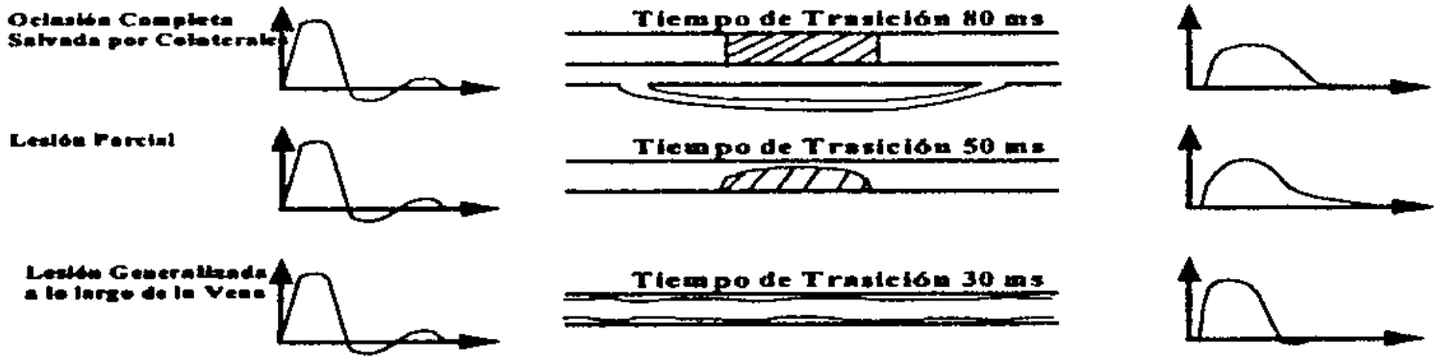



*Figura 5.3 Cambios en la Forma de Onda y en los Tiempos de Transición para distintos Grados de Estenosis*

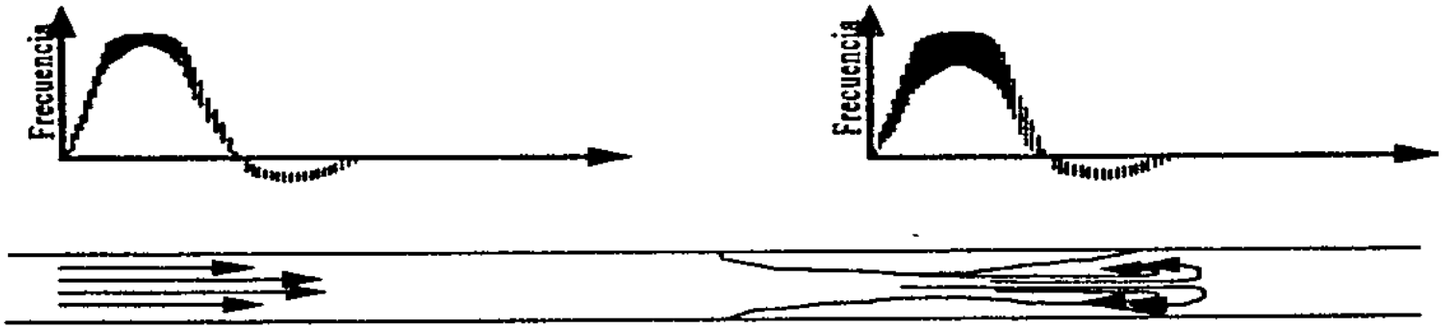



*Figura 5.4 Espectro en Tiempo Real. El Espectro Varia entre una Vena Sana y una Vena que es Afectada debido a una Estenosis*

Para nuestro trabajo se ha elegido el análisis espectral de señales *Doppler de ultrasonido*, como un medio de obtener las características de velocidad del flujo sanguíneo y la *densidad de potencia espectral* (*PSD*) correspondiente para cualquier vena, y de esta manera extraer poder detectar la presencia de una lesión ó la corroboración de que se trata de una vena sana.

Debido a que la señal Doppler de ultrasonido se genera a partir de la detección de un conjunto aleatorio de glóbulos rojos que pasan a través de un haz ultrasónico generado por una sonda, la señal Doppler de ultrasonido por si misma es de naturaleza aleatoria. Ya que el espectro en distintos momentos del ciclo cardiaco es estimado en segmentos de tiempo muy cortos (2 - 20 ms) de la señal Doppler de ultrasonido, las mediciones del espectro para un ciclo cardiaco son muy irregulares. Para poder hacer una medición del ancho de banda del espectro y así auxiliar en la discriminación entre diferentes grados de una estenosis, la medición del espectro en puntos similares dentro de un ciclo cardiaco puede ser promediada para lograr una mayor exactitud en la medición del ancho de banda del espectro. Para nuestro caso de estudio, definimos diversos colores para identificar el rango de frecuencia en el que cae cada segmento de datos extraído de la señal ultrasónica.

La frecuencia de transmisión de la señal ultrasónica oscila entre el rango de 2 - 10 Mhz dependiendo de la penetración requerida. En los instrumentos que emplean señales Doppler de ultrasonido para el estudio del flujo sanguíneo, la determinación de esta frecuencia se somete ha consideración del compromiso entre un incremento en la atenuación y una mayor dispersión de los glóbulos rojos.

El espectro de la velocidad de interés cae en el rango audible del oído humano (100 Hz a 15 kHz) y es lo más común en instrumentos que emplean señales Doppler de ultrasonido.

Gracias a que existe un rango de velocidades diversas en una vena, la señal Doppler de ultrasonido contiene un rango de frecuencias correspondiente al rango de velocidades que existen en la sangre. La velocidad de la sangre dentro de las venas varia con el tiempo y con la respiración, y en las arterias tiene una pulsación regular correspondiente a los latidos del corazón.

## 5.3 Implementación de un Instrumento Virtual para Detección de Flujo Sanguíneo

En los párrafos anteriores mencionamos en que consiste el análisis espectral de señales Doppler de ultrasonido. Ahora revisaremos su aplicación directa para posteriormente materializar un instrumento que nos ayude en la interpretación de dichas señales para su estudio y ser capaces de discriminar clínicamente si la vena en cuestión es sospechosa de un posible padecimiento fisiológico.

### 5.3.1 Generador Doppler de Ultrasonido

En el presente trabajo se ha utilizado el llamado Instrumento Doppler de Onda Continua [15]. En la figura (5.9) se presenta un diagrama general del mismo.

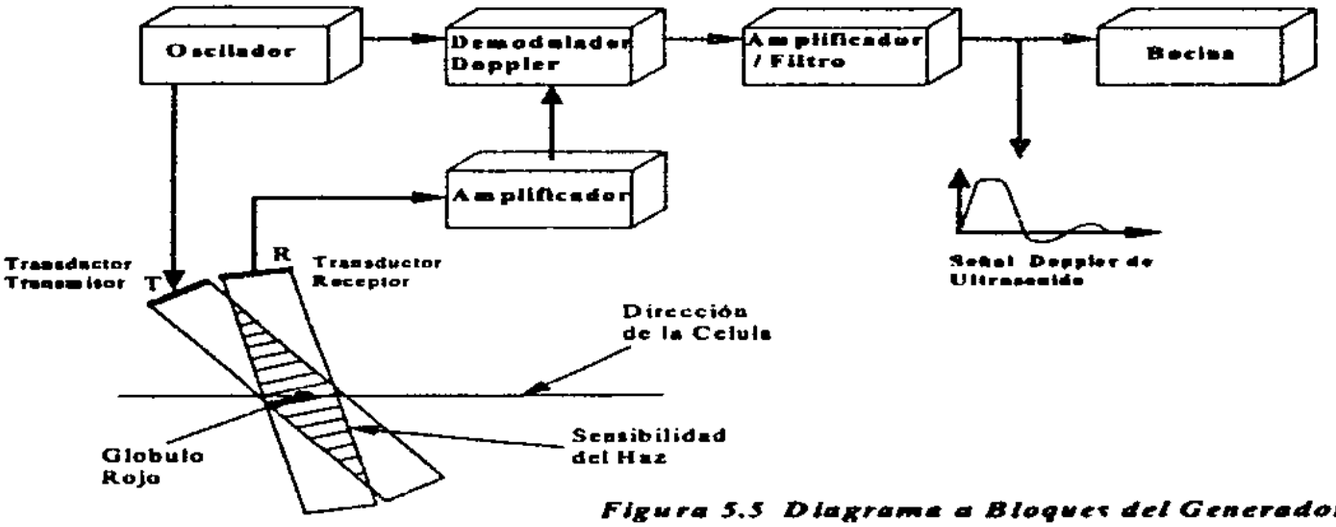



*Figura 5.5 Diagrama a Bloques del Generador Doppler Ultrasónico*

El transductor de transmisión (T) esta conectado a un oscilador para transmitir la onda ultrasónica de manera continua. Las ondas ultrasónicas reflejadas por los elementos de la sangre son captadas por el transductor receptor (R) montado en la misma sonda. La salida eléctrica del transductor receptor es amplificada y alimentada al demodulador, donde es mezclada con la señal del transmisor oscilador (la señal de referencia) con lo que obtenemos la señal Doppler de ultrasonido resultante que puede ser amplificada y alimentada a una

bocina y al *Sistema de Adquisición de Datos del IV*. Este ultimo constituye la parte central de lo que es el *instrumento virtual* (*IV*) pues engloba los componentes necesarios para crear una interface directa con el usuario de tal manera que en dicho instrumento se puedan configurar sus múltiples parámetros, los cuales afectaran directamente el comportamiento del mismo así como las características principales del despliegue gráfico.

### 5.3.2 Arquitectura del Instrumento Virtual

Uno de nuestros objetivos primarios consiste en desarrollar un IV que nos muestre el análisis espectral de las señales generadas por un flujo sanguíneo. En la figura (5.6) se presenta un diagrama simplificado del IV. El contenido de frecuencia de una señal Doppler de ultrasonido puede presentarse en una gráfica donde se muestre la *PSD* de la señal a una frecuencia determinada. El espectro para condiciones de flujo constantes, con un perfil de

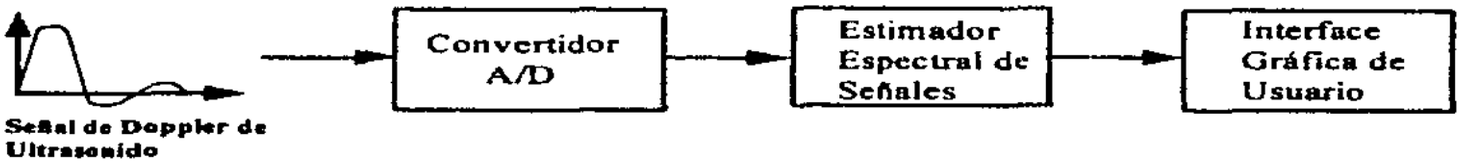



*Figura 5.6 Diagrama a Bloques del Instrumento Virtual*

velocidad parabólica y un rastreo uniforme de la vena entera, tiene una *PSD* constante al presentar su gráfica contra la frecuencia. Si el perfil de velocidad es mermado, existe una concentración de la intensidad de la señal en las frecuencias altas y menos en las frecuencias bajas, significando que existe una posible obstrucción en la vena o arteria. Aunado a esto, lógicamente tenemos que capturar la señal para alimentar los datos hacia el IV. Un diagrama de como y cuales procesos son necesarios para crear el IV, se presenta en la figura (5.7).

### 5.3.3 Procesos del Instrumento Virtual

En la figura (5.7) podemos observar los componentes principales que integran el IV. En donde podemos identificar cuatro procesos que lo constituyen :

- *Proceso de Conversión Analógico/Digital*
- *Proceso de Análisis de la Señal y Obtención de la PSD*
- *Proceso de Despliegue y Configuración*
- *Computador Anfitrión*

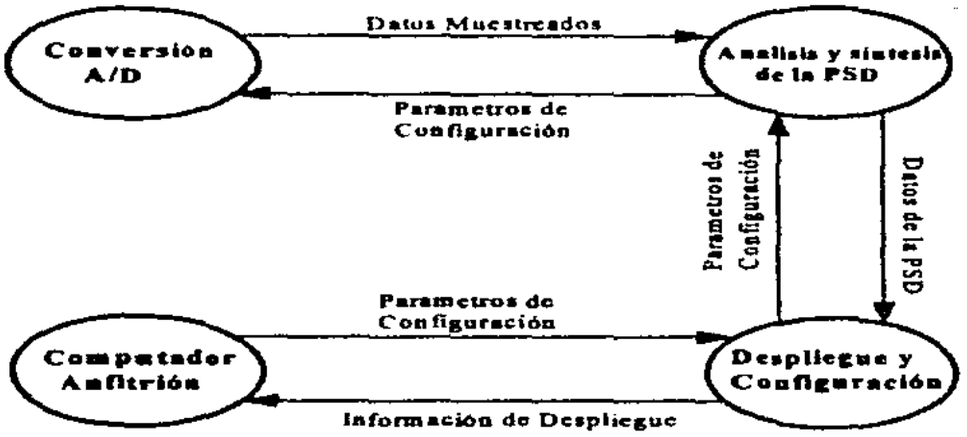



**Figura 5.7 Interacción Lógica de los Procesos en el Instrumento Virtual**

Cada uno de estos procesos tiene una estrecha inter-relación con alguno de los otros procesos y además intercambian información entre si. Esta es una vista lógica por función de cada proceso y se muestra el flujo de información que viaja entre cada uno de ellos que esta conectado de alguna manera con otro. Consideramos al computador anfitrión también como un proceso debido a su importancia dentro del contexto del IV. Cabe recordar que el IV esta construido en base a elementos de proceso discretos, los *Transputers*, que interactuan en conjunto para construir el IV. Por su especial construcción, los Transputers son ideales para utilizarlos en procesos que requieren una respuesta en tiempo real, ya que explotan las características del *procesamiento paralelo* y la *comunicación entre procesos*. Gracias ha éstas dos características, hemos construido un IV que no solo responde a la necesidad de procesar la información en tiempo real, si no que también contamos son un IV altamente *escalable* y altamente *configurable*. Escalable por que si es necesario contar con un mayor volumen de proceso de datos o incrementar la velocidad de respuesta de algún proceso,

bastara con agregar más elementos de proceso al sistema y re distribuir los datos entre dichos elementos. Configurable por que las herramientas de desarrollo ya existentes permiten de una manera relativamente fácil, el agregar un mayor número de elementos de proceso sin invertir un gran esfuerzo en realizar esta adaptación. A demás, gracias a una interface de usuario gráfica (GUI)[21], el usuario puede configurar los parámetros del IV de una manera fácil e intuitiva.

A continuación detallaremos el papel y otros aspectos de cada uno de los procesos que forman el IV.

A) Proceso de Conversión Analógico/Digital

Este proceso se encarga de obtener las muestras de las señales tomadas del mundo exterior, que se encuentran en formato analógico, y las transforma a muestras en un formato digital. Gracias a este proceso de conversión podemos tomar un conjunto de muestras de la señal, llámense *ventana de datos*, y obtener las características necesarias de dicha ventana para poder detectar la existencia de un posible problema en alguna vena o arteria que sea el sujeto de estudio. El proceso que controla al convertidor analógico/digital esta configurado con una velocidad de muestreo de información de *10 kHz* . Esta escala es variable ya que puede ser configurada a un rango de valores que van desde 1 Hz a 100 KHz. Esto nos permite tener un muestreo de la señal en tiempo real sin ambigüedades en la misma (Aliasing).

Adicionalmente, en esta fase se aplica una *ventana* ("window") a la señal de entrada. Esto no es más que aplicar (multiplicar en el dominio de el tiempo y una convolución en el dominio de la frecuencia) la parte correspondiente de una función bien conocida a los datos adquiridos por el convertidor A/D. El motivo de realizar dicha tarea con las muestras de la señal adquirida es el de reducir o suavizar la oscilación que se introduce en la señal (esto es, ruido) cuando es adquirida a través de la toma de muestras de la señal (conocido como fenómeno de *Gibbs*). El IV esta habilitado para soportar cuatro distintos tipos de ventanas: Hanning, Hamming, Bartlett y rectangular[4].

El convertidor A/D esta montado sobre una tarjeta especial que sirve para albergar a los Transputers y ocupa un TRAM [30] de tamaño dos en la misma. Al igual que un Transputer, el dispositivo convertidor A/D tiene cuatro link's, dos de entrada y dos de salida de datos con velocidades configurables.

La información que viaja hacia/desde este proceso por los link's de comunicaciones, se presenta a continuación. Parte de esta información sirve para configurar los parámetros del IV y algunas otras sirven como áreas de datos o variables de control.

**Parámetros enviados desde el proceso de Análisis hacia el proceso de adquisición A/D:**

- *Bandera de estado de Continuación del Proceso*
- *Máximo Valor en la escala del Eje del Tiempo (Eje X)*
- *Frecuencia de Muestreo. Configuración de los parámetros del IV*
- *Selección de la Ventana ha ser Aplicada a los Datos*
- *Bandera de Estado para indicar si todos los Procesos Operan Correctamente*

**Parámetros Datos enviados desde el proceso de adquisición A/D hacia el proceso de Análisis:**

- *Buffer de Almacenamiento de los Datos*
- *Número de Muestras Utilizadas en el Buffer*

**B) Proceso de Análisis de la Señal Ultrasónica y Obtención de la *PSD***

Una vez habiendo obtenido los datos del mundo real, el siguiente paso es realizar un análisis acerca de cuales son los componentes espectrales de la señal para de esta manera poder detectar si es que existe algún indicio de una posible lesión. Dicho análisis se realiza a través de un estimador espectral, en nuestro caso hemos implementado un algoritmo, la *FFT*, para obtener las componentes espectrales de nuestra señal adquirida mediante el proceso de conversión A/D.

Adicionalmente a sus funciones fundamentales, el proceso de análisis sirve como un puente de comunicación entre los procesos de adquisición y el proceso de despliegue y configuración.

Las principales tareas ha realizar por este proceso son:

*I* *Comunicar los Datos/Parámetros desde el proceso de despliegue hacia el proceso de análisis y hacia el proceso de adquisición*

Espera los siguientes parámetros a través del canal del monitor:

- Bandera de estado de continuación
- Valor de período cardiaco
- Máximo valor en la escala en el eje del Tiempo (Eje X)
- Selección del Estimador a utilizar (por ahora solo FFT)
- Selección de la ventana ha ser aplicada
- Frecuencia de Muestreo
- Tamaño de la Muestra
- Número de colores a utilizar (máximo 15)

*II* *Calcula la cantidad de segmentos tanto en el eje del tiempo como en el eje de la frecuencia ha ser desplegados, de acuerdo al número de datos adquiridos*

El IV esta construido de tal manera que es posible configurar la escala de valores que existe tanto en el eje del tiempo (eje X) como en el eje de la frecuencia (eje Y). Otros factores que afectan la escala son la selección del período cardiaco, la frecuencia de muestreo y el tamaño de la muestra. En base a estos parámetros, se calcula el tamaño y el número de segmentos a desplegar

*III* *Inicializa los parámetros y calcula los valores de la PSD*

Para iniciar el proceso de análisis, se deben calcular los parámetros iniciales del algoritmo de estimación espectral, los cuales incluyen el calculo de los factores de peso (twildde factor) y la obtención de los índices para ordenar los valores finales (bit-revers). Se calcula la Transformada Rápida de Fourier a partir de los datos muestreados y de esta manera se obtienen las componentes espectrales de frecuencia de dichos datos.

Una vez calculada la FFT, hemos obtenido la descomposición en frecuencias de la señal de entrada. Para gráfica los datos obtenidos por la FFT, calculamos la PSD (Densidad Espectral de Potencia) correspondiente: Se calcula el módulo cuadrado de cada punto obtenido por la FFT y se divide por el número de muestras determinado.

*IV* *Se busca el valor máximo de la PSD y con ellos se realiza la asignación de colores a ser utilizados por el proceso de despliegue y configuración*

Divide las muestras en un número de grupos igual o menor, dependiendo del valor de la muestra, al del número de colores configurados tomando como base el máximo valor obtenido para esa muestra

*V* *Revisa los canales de entrada para detectar si existen posibles cambios en la configuración del IV ó si ocurre algún otro evento*

Si el usuario decide cambiar los parámetros del IV, dichos parámetros deberán ser transmitidos a todos los procesos para que se re configuren y se sincronicen nuevamente para procesar los nuevos datos que lleguen al sistema

Cabe hacer mención que todos los procesos deben permanecer en un estado de sincronía para que cualquier cambio se vea reflejado en la salida gráfica y el usuario obtenga la información deseada del IV.

La flexibilidad del sistema que realiza la parte del proceso, le puedan ser añadidos cualquier número de procesadores (Transputers) que ayuden a incrementar el número de las muestras a ser procesadas por unidad de tiempo. En la figura (5.8) tenemos un diagrama a bloques en donde se procesan cuatro segmentos de la señal mediante el uso de cuatro procesadores paralelamente.

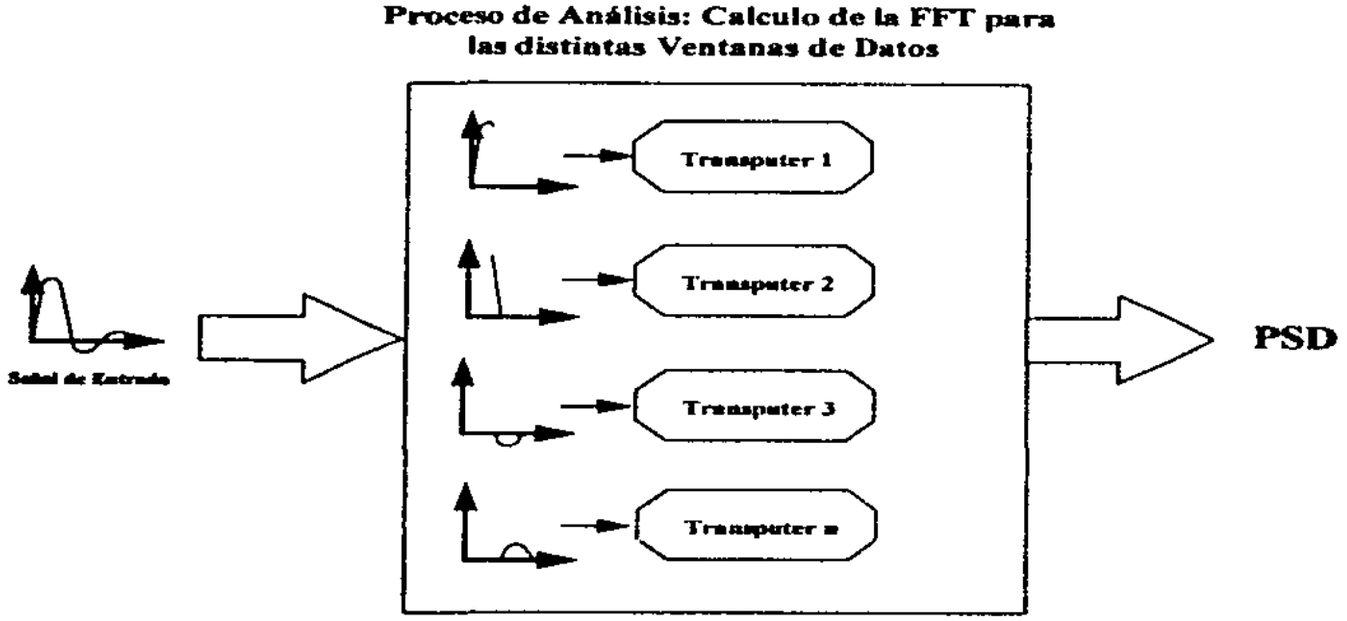



**Figura 5.8 División de la señal de Entrada en distintos Segmentos para Aprovechar el Procesamiento Paralelo**

Como podemos ver en la figura (5.8), pueden tomarse distintas ventanas de datos y procesarlas en forma paralela cada una en un procesador distinto. Posteriormente el proceso de análisis se encargara de conjuntar cada una de dichas ventanas y enviarlas al proceso de despliegue para su graficación. El caso presentado en la figura muestra el ejemplo para cuatro procesadores pero este número se puede extender aun más dependiendo de los requerimientos de la aplicación. Es recomendable tomar siempre un número par de procesadores debido a el funcionamiento interno del algoritmo de la FFT en éste caso.

Actualmente el proceso de análisis esta construido para que utilice esta arquitectura de distribución de datos pues el estimador ha sido diseñado para este fin.

C) Proceso de Despliegue y Configuración

Este proceso es una de las partes vitales del IV. Aquí se encuentra la construcción de la interface hacia el usuario tanto en la parte del despliegue como en la parte de la configuración y éstas funcionan como el "pegamento" para los otros procesos que componen el IV. A continuación revisaremos con mayor detalle los componentes principales de este proceso.

## *I Configuración del Instrumento Virtual*

En esta primera versión del IV, existen siete distintos parámetros que pueden ser modificados y que inciden directamente tanto en el funcionamiento como en el desempeño del sistema. Dichos parámetros son:

- Tipo de Estimador (En esta versión solo FFT)
- Ventana a Aplicar (Hamming, Hanning, Bartlett, Rectangular)
- Período Cardiaco
- Frecuencia de Muestreo
- Tamaño de la Muestra
- Número de Colores
- Escala de Tiempo y Frecuencia

Cuando el IV se ejecuta, los distintos parámetros de configuración son determinados a sus valores por defecto. Dichos parámetros toman los siguientes valores:

| | | |
|---|---|---|
| • Tipo de Estimador | = | FFT |
| • Ventana a Aplicar | = | Rectangular,Hamming,Hanning,Bartlett |
| • Período Cardiaco | = | 600 - 1000 ms |
| • Frecuencia de Muestreo | = | 5.0 - 50.0 KHz |
| • Tamaño de la Muestra | = | 16 - 256 Muestras |
| • Número de Colores | = | 5 - 15 Colores |
| • Escala de Tiempo y Frecuencia | = | 0.1 seg/div y 0.64 KHz/div |

Para realizar cualquier operación que se desee en el IV, debemos elegir alguna de las opciones presentadas en el menú inicial como se muestra en la figura (5.9)

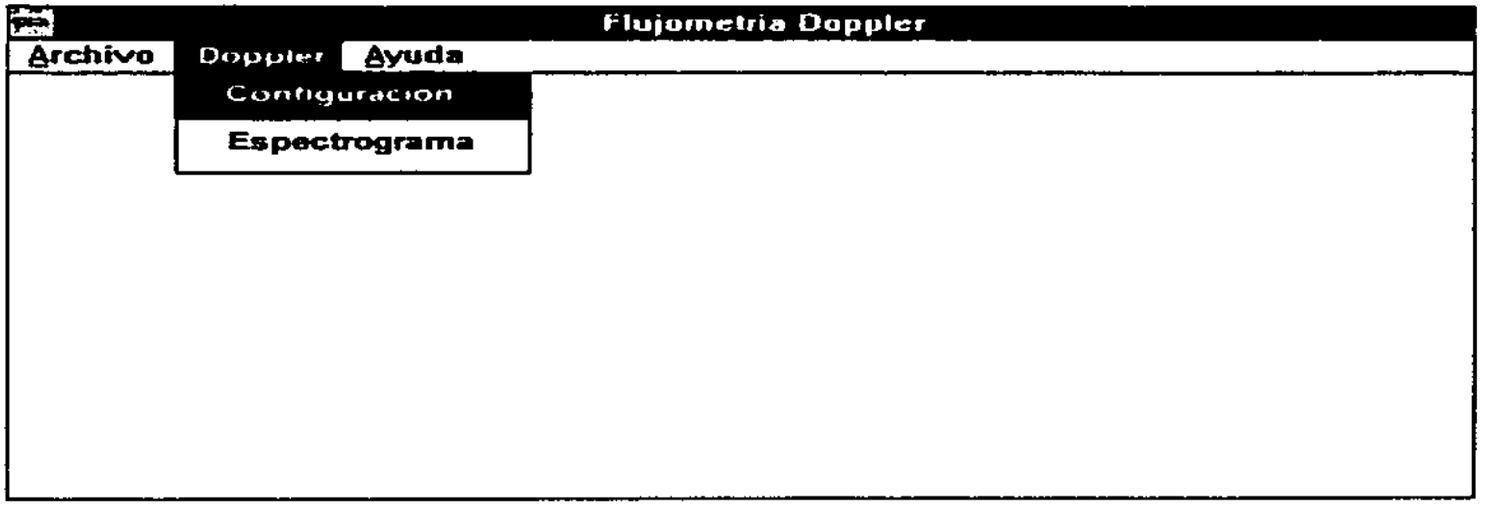



**Figura 5.9 Pantalla Inicial del Instrumento Virtual**

Si el usuario del IV desea cambiar los valores por defecto, puede realizar esta operación accesando la opción de configuración del menú principal. Cualquier cambio a la configuración será inmediatamente reflejada en la sesión actual y los cambios permanecerán hasta que la sesión actual del IV no sea cerrada. Esto implica que los cambios no se hacen permanentes y serán restaurados en cuanto se vuelva ha ejecutar el IV.

La pantalla de configuración tiene la interface típica de cualquier aplicación Windows[21][22]. En esta pantalla podemos visualizar los valores de los diversos parámetros así como la unidades en los que están dados tales parámetros.

Una vez finalizado el proceso de configuración de parámetros del IV, en el proceso de despliegue y configuración, estos deberán hacerse del conocimiento de los demás procesos para que trabajen todos conjuntamente sobre una misma base. Esto es necesario ya que todos los procesos deben de intercambiar información entre si, y dicha información debe contener las mismas características.

*II Despliegue del Espectro grama*

Todo los procesos de adquisición, análisis y configuración, son necesarios para obtener un despliegue gráfico en donde se muestran las diversas componentes espectrales de la señal ha ser estudiada, en nuestro caso las señales Doppler de ultrasonido, extraídas del movimiento del flujo sanguíneo a través de las venas o arterias de una persona.

Cada color en la gráfica muestra un rango de componentes espectrales extraídas a partir de la velocidad del flujo sanguíneo. La forma de onda representa la variación de la velocidad con respecto del tiempo que experimenta dicho flujo sanguíneo. Si se realiza cualquier afectación a la configuración del IV, los cambios se verán reflejados directamente en el despliegue gráfico y podrán ser fácilmente visualizados por el usuario.

El proceso de despliegue debe realizar el cálculo de los ejes así como el tamaño de los segmentos y su color correspondiente dependiendo de los parámetros configurados.

D) Computador Anfitrión

El computador anfitrión tiene una importancia fundamental en el IV ya que gracias a el se refleja toda la interface gráfica que observara el usuario y a través de ella el usuario interaccionara con el mismo. A continuación se presenta un diagrama (5.10) que engloba los procesos principales del IV y en donde se puede observar el papel desempeñado por el computador anfitrión.

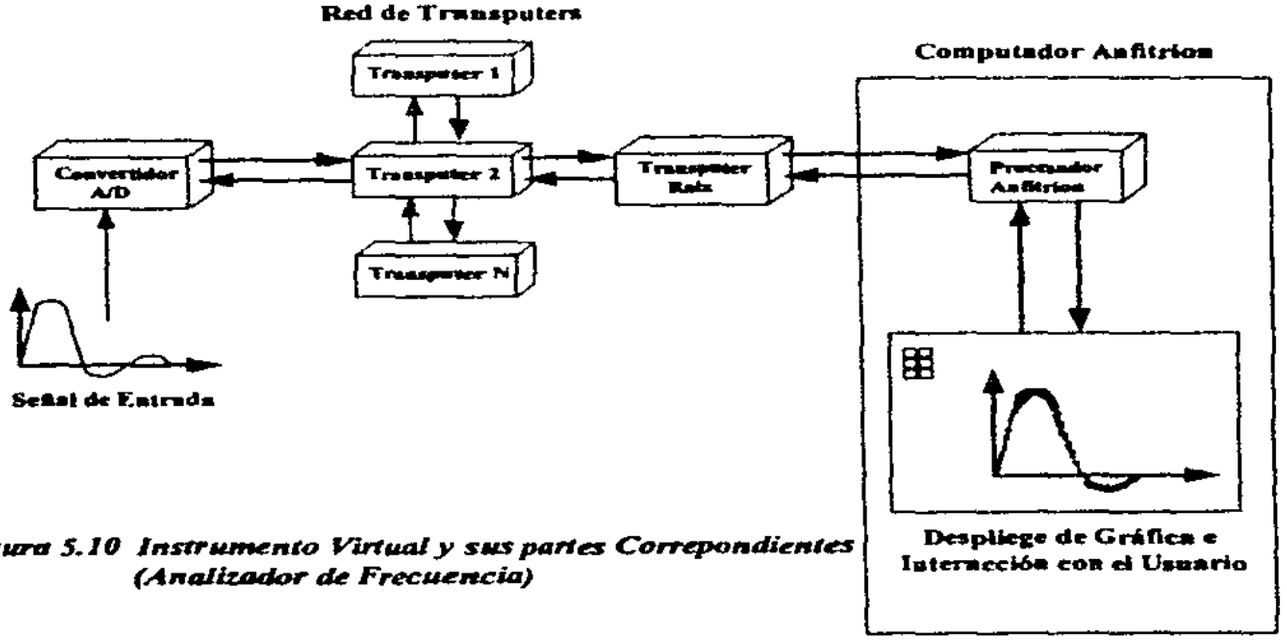



***Figura 5.10 Instrumento Virtual y sus partes Correpondientes (Analizador de Frecuencia)***

El computador anfitrión se encarga de recibir todos los requerimientos originados por el usuario y los transmite hacia los procesos almacenados en los Transputer's. En este punto debemos recordar que el IV esta creado bajo la interface gráfica de Windows y por lo tanto debe cumplir con los estándares y con el manejo de una aplicación Windows. Una de las

partes más importantes con respecto ha esta cuestión es el manejo de los mensajes dentro del ambiente Windows. Cada que el usuario accesa a un menú, mueve el ratón, oprime una tecla, selecciona un botón, la aplicación despliega algún objeto, etc., se genera un mensaje[21][24]. Este mensaje debe ser manejado por el computador anfitrión para transmitirlo a cualquiera de las aplicaciones actualmente activas dentro del mismo. En el caso del IV, los mensajes son canalizados a través del *Windows Server*[22] hacia el proceso de despliegue y configuración que reside en el Transputer raíz (aquel que esta conectado directamente con el computador anfitrión). Este proceso se encargara de definir que acción se deberá tomar dependiendo del tipo de mensaje que haya arribado y tomara las acciones necesarias para responder a el siempre tomando en cuenta la interacción entre el computador anfitrión y el usuario del IV.

## 5.4 Resultados

En esta sección se presentan los resultados obtenidos en el presente trabajo. Estos resultados han sido útiles para la implementación de un analizador de espectro en tiempo real para detección de padecimientos cardiovasculares. Primeramente presentamos los resultados correspondientes al desempeño del modelo computacional utilizado en la implementación del estimador espectral. Posteriormente presentamos los resultados correspondientes a la Interface Gráfica de Usuario para configurar y controlar el IV desarrollado en el presente trabajo.

### 5.4.1 Desempeño del Modelo Computacional del Estimador Espectral

El desempeño del modelo computacional utilizado en la realización del estimador espectral, se presenta en la figura (5.11). Dicho modelo fue implementado en una plataforma de Transputer's del tipo T805 - 30. Las mediciones correspondientes a los tiempos de ejecución han sido llevadas a cabo mediante el uso del "timer" interno del Transputer, considerando segmentos de 10 ms con un número variable de muestras (tamaño de la ventana). Además se asumen diferentes tamaños de la ventana y un número variable de procesadores. Cabe señalar que los tiempos de ejecución, para el caso de un solo procesador, se obtienen al medir el tiempo de ejecución de una ventana en un solo procesador. Para el caso de dos procesadores, se debe entender que el tiempo de ejecución corresponde a procesar simultáneamente dos ventanas en dos procesadores. Y así sucesivamente.

| # Datos de la Ventana | #Procesadores 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 32 | 0.767 | 0.945 | 0.974 | 1.003 |
| 64 | 1.886 | 2.229 | 2.267 | 2.309 |
| 128 | 4.452 | 5.112 | 5.170 | 5.237 |
| 256 | 10.659 | 11.687 | 11.793 | 11.906 |
| 512 | 23.659 | 26.236 | 26.430 | 26.646 |
| 1024 | 53.342 | 58.470 | 58.857 | 59.273 |

**Tabla 5.11 Desempeño del Modelo Computacional utilizando en la implementación del Estimador Espectral**

Al realizar un análisis de los tiempos de ejecución mostrados en la figura (5.11), se observa que para el caso de un procesador ejecutando una sola ventana de datos, se obtiene ejecución en tiempo real solamente para los casos de *32, 64* y *128* datos. Utilizando dos procesadores ejecutando dos ventanas consecutivas, podemos obtener ejecución en tiempo real hasta de 256 datos. Utilizando tres procesadores podemos obtener una ejecución en

tiempo real en ventanas de hasta 512 datos. Por último, podemos observar que si utilizamos cuatro procesadores podremos tener una ejecución en tiempo real de ventanas de hasta 512 datos. Si se requiriera de procesar una ventana de información de más de 512 datos, necesitaríamos más que cuatro procesadores para obtener ejecución en tiempo real.

### 5.4.2 Interface Gráfica de Usuario y Control del Instrumento Virtual

En éste punto se presentan los resultados correspondientes a la Interface Gráfica de Usuario y a los componentes involucrados en la visualización, configuración y control del IV desarrollado.

A continuación se muestra en la figura (5.12), el diálogo de configuración del IV. En dicho diálogo podemos observar todos los parámetros que son factibles de ser modificados en el IV y algunos parámetros que en esta versión no han sido habilitados pues corresponden a la posterior linea de investigación a seguir en al utilización de otros estimadores espectrales.

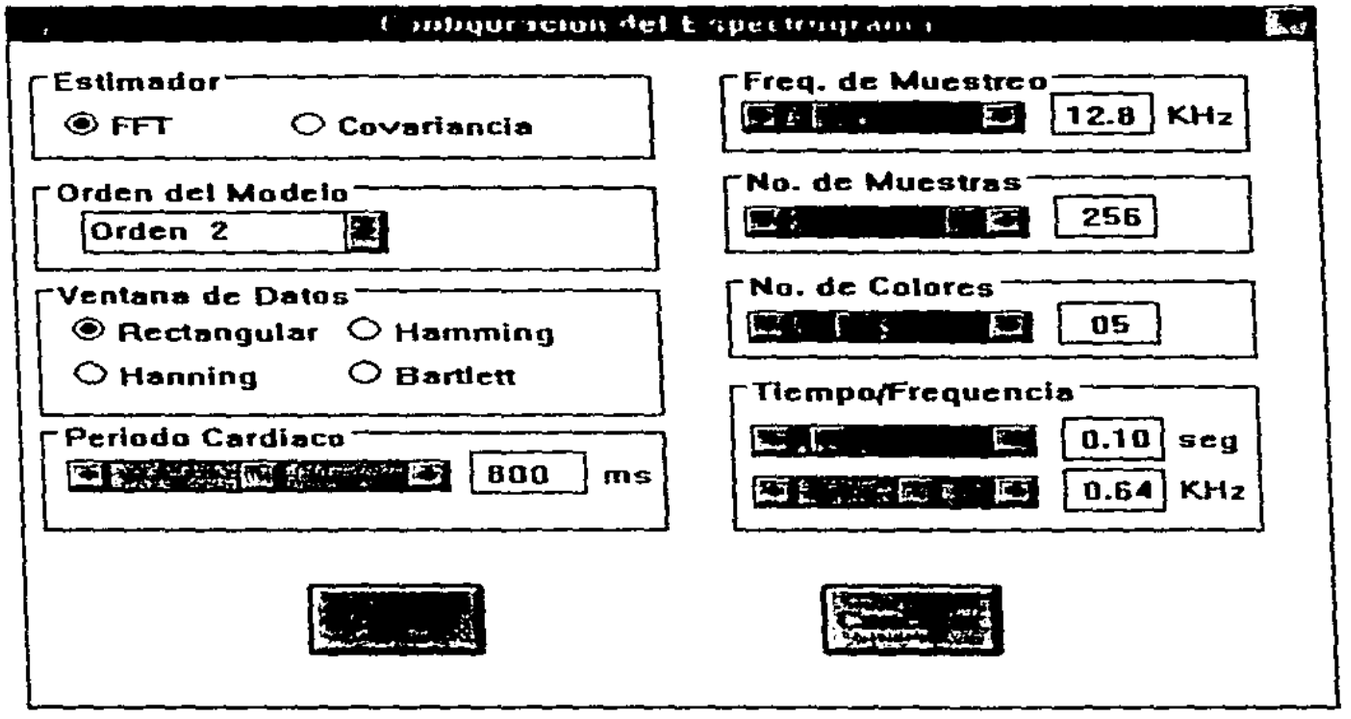



**Figura 5.12 Pantalla de Configuración del Instrumento Virtual**

Como ya se menciono, en ésta figura (5.12) podemos observar los diversos parámetros que pueden ser configurados en el IV. En el caso del presente trabajo, el IV está orientado a la detección de flujo sanguíneo y por lo tanto los parámetros también. Esta flexibilidad en cuanto a la configuración es una de las características más importantes del IV, ya que son configuradas vía software y no hardware. Por lo tanto, la extensibilidad del IV está limitada solo por las necesidades y los alcances de su diseño. A continuación describimos dichos parámetros:

- Estimador: Es el algoritmo seleccionado por el usuario para extraer las componentes espectrales para el cálculo de la PSD de la señal Doppler de ultrasonido. Para los fines del presente trabajo, solo se ha implementado dicho algoritmo en base a la FFT. En los próximos desarrollos de ésta línea de investigación, se pretende adicionar otros algoritmos para realizar el proceso de estimación.

- Ventana de Datos: Actualmente se puede elegir una de cuatro ventanas a aplicar a la señal de entrada: Rectangular, Hamming, Hanning o Bartlett. El aplicar una ventana u otra implica "suavizar" ciertos segmentos de la ventana de datos que será procesada.

- Período Cardiaco: Se refiere al la duración de una pulsación realizada por el corazón y varia de una persona a otra en un rango aproximado de 600 a 1000 mili segundos.

- Frecuencia de Muestreo: Es la velocidad a la cual se extraen las muestras de la señal Doppler de ultrasonido. Este parámetro, en el caso del IV para análisis de flujo sanguíneo, es dependiente del hardware que se utiliza para capturar la señal del mundo real, ya que configura directamente al convertidor A/D.

- Número de Muestras: Se refiere al tamaño de la ventana de datos ha ser adquirida y procesada por el IV.

- Número de Colores: Número de colores configurado para desplegar la PSD de la señal.

- Escala de Tiempo y Frecuencia: Variación de la escala de los ejes x (tiempo) y eje y (frecuencia).

Como resultante del proceso y de acuerdo a los parámetros de configuración, obtenemos una salida gráfica denominada "Espectro grama" que se muestra a continuación en la figura (5.13). Dicho espectro grama corresponde a la señal Doppler de ultrasonido.

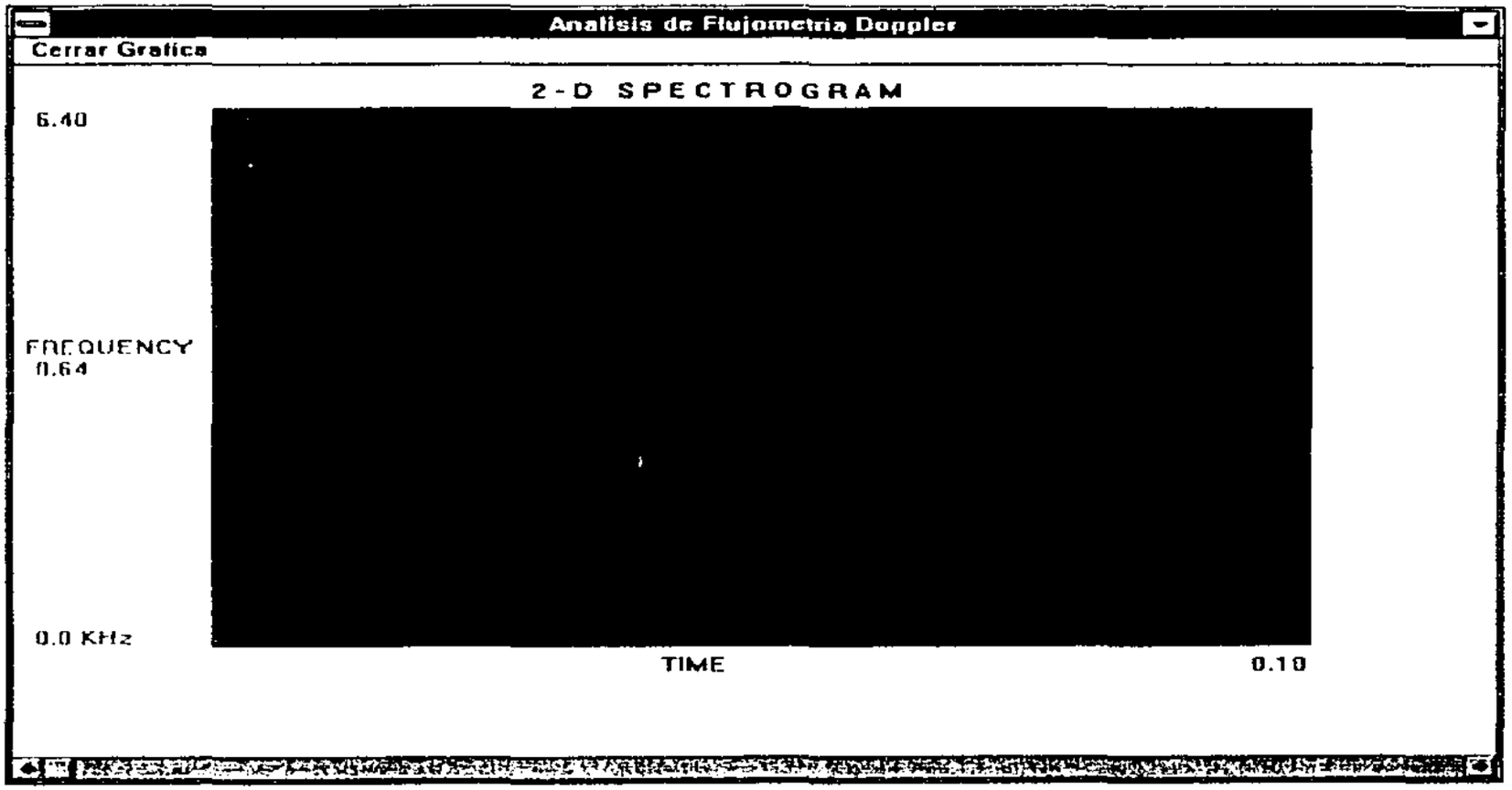



**Figura 5.13 Salida Gráfica del Instrumento Virtual**

El espectro grama muestra la descomposición en frecuencia de la señal Doppler de ultrasonido, esto es, la densidad espectral de potencia, para una vena en estudio. Aqui podemos apreciar la forma de onda de la señal correspondiente a una vena sana, captada a través del generador Doppler ultrasónico. Dicha forma de onda es de gran utilidad al ser una herramienta que nos permite identificar si estamos tratando con una vena sana, o si un cambio en ella representa la presencia de algún padecimiento (tal como una oclusión, acumulación de grasa o tejido, etc. ). Los colores indican una variación de las componentes de frecuencia de la señal y por tanto, una variación de la PSD. Ademas, podemos apreciar las escalas de tiempo y frecuencia que está utilizando el IV. Estas escalas son variables y pueden ser predeterminadas a un valor conveniente, dependiendo de las características de la señal y de los requerimientos del usuario. En el caso del eje del tiempo, podemos visualizar un número de periodos variables en el espectro grama; esto es de gran ayuda si se desean analizar varios ciclos de la señal en una misma imagen del espectro grama. En la escala del eje de la frecuencia, podemos controlar el rango de frecuencias a ser visualizado. Esto se reflejara en un escalamiento de la señal en donde podremos apreciar un espectro de frecuencias variable dependiendo del tipo de la señal y de las necesidades del usuario.

# CAPÍTULO 6

## *CONCLUSIONES*

### 6.1 Conclusiones Generales

Al finalizar el presente trabajo, podemos decir que se han cumplido los objetivos presentados:

Se Implementó un Algoritmo de Estimación Espectral basado en la FFT, utilizando una arquitectura de procesamiento paralelo.

Se implementó un Instrumento Virtual para realizar Estimación Espectral de señales Doppler de ultrasonido, utilizando una plataforma de Procesamiento Paralelo.

Se desarrolló una Interface Gráfica de Usuario que permite configurar y controlar los diversos procesos que integran el Instrumento Virtual.

Se aplicó dicho Instrumento Virtual al caso de Estimación Espectral de señales Doppler de ultrasonido.

Lo anterior nos permite observar en tiempo real, una gráfica del contenido espectral de una señal Doppler de ultrasonido, captada de una vena y de ésta manera poder verificar si el análisis de frecuencias de la señal, la PSD, coincide con el de una vena sana o existe alguna clase de padecimiento en ella. Éste instrumento es de gran valía ya que su costo es menor que el de otros equivalentes en el mercado y ademas es factible de ser mejorado en su totalidad.

### 6.2 Trabajos Futuros

De acuerdo con las líneas de investigación a seguir por el laboratorio de Procesamiento Paralelo del Departamento de Electrónica y Automatización, se puede realizar la implementación de otros algoritmos de estimación espectral, que pueden proporcionar una mejor resolución del espectro de la señal, tales como aquellos basados en modelos paramétricos o en filtros adaptativos.

Otra línea de investigación es la utilización de arquitecturas heterogéneas para la implantación del algoritmo de estimación de las señales. Esto contribuiría en un mejor desempeño del Instrumento Virtual, ya que actualmente están surgiendo nuevas arquitecturas que tienen un mejor desempeño por estar orientadas a tratamiento de señales.

Trabajo adicional en la presentación de la interface gráfica ayudaría a resaltar ciertas particularidades especificas de la señal, tal como una visión tridimensional de la misma o la posibilidad de ver otro tipo de presentación con respecto a lo que la interface gráfica se refiere.

El agregar funciones adicionales al Instrumento Virtual, tal como la posibilidad de sobreponer el espectro de dos señales distintas, el congelamiento de la imagen, la escritura/lectura de los datos hacia/desde un medio magnético, serian deseables, pues darían una mayor versatilidad y se acercaría hacia un posible desarrollo comercial de dicho instrumento.

# APÉNDICE A - Requerimientos del Sistema

En este apéndice se presentan las características del servidor de transputers WSP 4.0 para windows, así como sus requerimientos de hardware y software y las tarjetas de transputers soportadas.

**Hardware**

-Una computadora personal IBM PC AT o compatible con un procesador Intel 286/386/486/P5 o compartible; 4 Mb en RAM; una tarjeta de video VGA/SVGA.

-Un tarjeta principal de procesamiento paralelo Inmos B004 o compatible

**Software**

-Microsoft MS DOS 5.0/6.0;

-Microsoft Windows 3.0/3.1/Windows 95;

-Inmos ANSI C Toolset IMSD7214 o superior;

-Cualquier compilador que soporte la programación para Windows (tal como Microsoft C/C++, Borland C/C++ o Zortech C/C++) - para escribir las DLL de extensione; y un compilador de recursos para Windows

**Tarjetas Soportadas**

Inmos:

- B004
- B008

Transtech:

- TMB04
- TMB08

Parsytec:

- MTM-PC
- TPM-PC
- BBK-PC
- BBK-PC-2

# APÉNDICE B - Clasificación de Arquitecturas Paralelas

## B.1 Mecanismos de Control

Flynn M. J. propuso un sencillo modelo[13] para clasificar todos las computadoras[8] . Observó el paralelismo de los flujos de instrucciones y de datos exigidos por las instrucciones en los componentes más restringidos de la máquina, y coloco a todos las computadoras en una de cuatro categorías:

**Flujo único de instrucciones, flujo único de datos (SISD, uniprocesador)**
**Flujo único de instrucciones, flujos múltiples de datos (SIMD)**
**Flujos múltiples de instrucciones, flujo único de datos (MISD)**
**Flujos múltiples de instrucciones, flujos múltiples de datos (MIMD)**

Este es un modelo tosco, ya que algunas máquinas son híbridos de estas categorías.

En este punto nos preguntamos acerca de a que nos referimos con "flujo único" o "flujo múltiple". Una máquina que suma un número de 32 bits en un ciclo de reloj parecería tener múltiples flujos de datos cuando se compare con una computadora de bits en serie que emplee 32 ciclos de reloj para la misma operación. Esta es la idea aproximada de este modelo que es el más aceptado.

### B.1.1 SISD (Single Instruction Stream, Single Data Stream)

Las máquinas SISD son el modelo clásico de computo establecido por John von Neumann y es el más difundido en la actualidad. Este modelo computacional toma una sola secuencia de instrucciones y opera en una sola secuencia de datos.

La rapidez de una computadora SISD esta limitada por dos factores: la velocidad de ejecución de las instrucciones y la velocidad a la cual la información es intercambiada entre la memoria y el UCP. Esto ultimo puede incrementarse si incrementamos el número de canales de para accesar los datos simultáneamente. Esto se puede lograr si dividimos la memoria en un número determinado de bancos, cada uno de los cuales es accesado independientemente. Esto es llamado "memory interleaving".

Otro camino para incrementar la velocidad de intercambio de información entre el UCP y la memoria es utilizar una memoria relativamente pequeña pero muy

rápida para actuar como almacenamiento temporal de la memoria primaria. Esta memoria es llamada "cache memory". La memoria cache utiliza el principio de que una vez accesada una localidad de memoria, posteriores accesos a la memoria serán a localidades contiguas al ultimo acceso.

La velocidad de ejecución de instrucciones puede también ser incrementado si se solapan las operaciones de ejecución de una instrucción y el traer la siguiente instrucción a ser ejecutada a la UCP (más especifica mente, a la cola de instrucciones). Esta técnica se le conoce como "instrucción pipelining". En otra técnica relacionada, llamada "execution pipelining", múltiples instrucciones son ejecutadas en varias unidades funcionales de la UCP tales como multiplicadores o sumadores.

Memory interleaving, memoria cache y pipelining ahora son comúnmente utilizados en la computadoras SISD de alto rendimiento; no obstante todos tienen sus limitaciones. Memory interleaving y por extensión , pipelining son completamente utilizados solo si un pequeño conjunto de operaciones es ejecutado en grandes arreglos de datos. Las memorias cache incrementan el ancho de banda de procesador-memoria, pero su velocidad esta limitada por la tecnología de hardware. Un camino alternativo para elevar la velocidad en la ejecución de instrucciones es utilizar múltiples UCP y unidades de memoria interconectadas de alguna manera. La velocidad de proceso de dichos sistemas crece cuando el número de UCP y unidades de memoria se incrementa.

### B.1.2 SIMD (Single Instruction Stream, Multiple Data Streams)

Múltiples unidades de proceso en una computadora en paralelo operan en una de dos maneras, bajo el control centralizado de una sola unidad de control o trabajan independientemente. El primer caso es el referido al modelo SIMD. Una sola unidad de control despacha instrucciones a cada unidad de proceso. En una computadora paralelo SIMD, una sola instrucción es ejecutada sincronamente por todas las unidades de proceso. Las unidades de proceso pueden ser deshabilitadas durante un ciclo de instrucción. La motivación original para SIMD fue amortizar el coste de las unidades de control mediante docenas de unidades de ejecución. Una ventaja observada más recientemente es el reducido tamaño de la memoria de programa ya que SIMD necesita sólo una copia del código que se esta ejecutando simultáneamente.

### B.1.3 MISD (Multiple Instruction Stream, Single Data Stream )

Existen pocos ejemplos de computadores MISD, uno de ellos es el de los "Arrays sistólicos". Estos evolucionaron de intentos de obtener un ancho de banda de calculo mas eficiente. Este se puede considerar como un método para desarrollar

computadoras de propósito especial para equilibrar recursos, ancho de banda de E/S y cálculo. Basándose en la segmentación, los datos fluyen en etapas de desde memoria a un arreglo de unidades de cálculo y regresan a la memoria. Recientemente, la investigación sobre arrays sistólicos se ha desplazado desde muchos chips de propósito especial dedicados a menos chips, más potentes, que son programables.

### B.1.4 MIMD (Multiple Instruction Stream, Multiple Data Stream)

Son elementos de procesos capaces de ejecutar un programa independientemente de otros. Los procesadores individuales en una arquitectura MIMD son más complejos por que cada procesador tiene su propia unidad de control, lo cual los hace mas costosos que los procesadores SIMD. No obstante es posible construir una computadora paralela con microprocesadores de propósito general como unidades MIMD, los cuales son producidos en grandes cantidades y eso abarata su costo, a diferencia de los elementos de una arquitectura SIMD en donde suelen ocuparse procesadores de propósito específico.

MIMD ha conseguido una posición consolidada frente a las demás arquitecturas. Actualmente se estima que una arquitectura MIMD puede ser más efectiva para una carga de trabajo en tiempo compartido que un SISD. Un determinado programa no emplea menos tiempo de UCP, pero se puede completar un mayor número de tareas independientemente por hora - un argumento de productividad frente a latencia. Solucionado el problema de ensamble de unidades de proceso, ahora se requiere la interacción entre ellas. El paso de mensajes y la memoria compartida proveen dos maneras distintas de realizar dicha tarea.

## B.2 Organización del Espacio de Direcciones

### B.2.1 Arquitecturas de Paso de Mensajes

En esta arquitectura los procesadores están conectados a través de redes de interconexión de paso de mensajes. Cada procesador tiene su propia memoria llamada **local** o **privada**, la cual es accesible solo al procesador. A continuación se muestra la arquitectura general de una computadora de paso de mensajes en la figura (C.1).

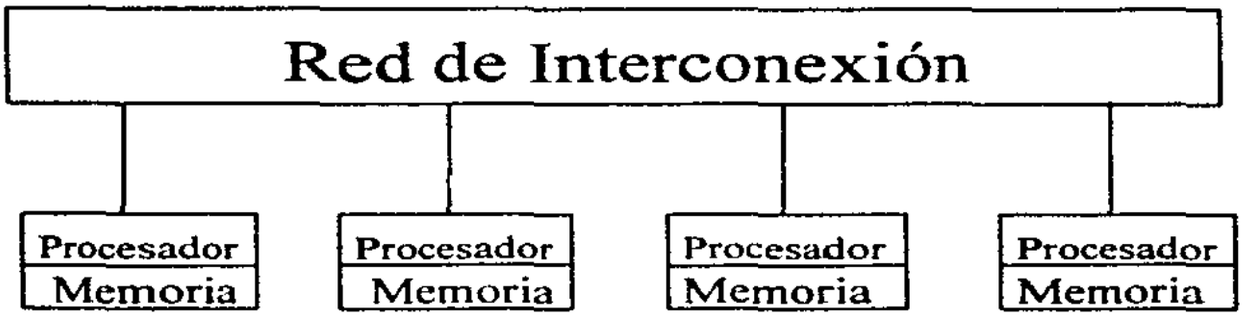
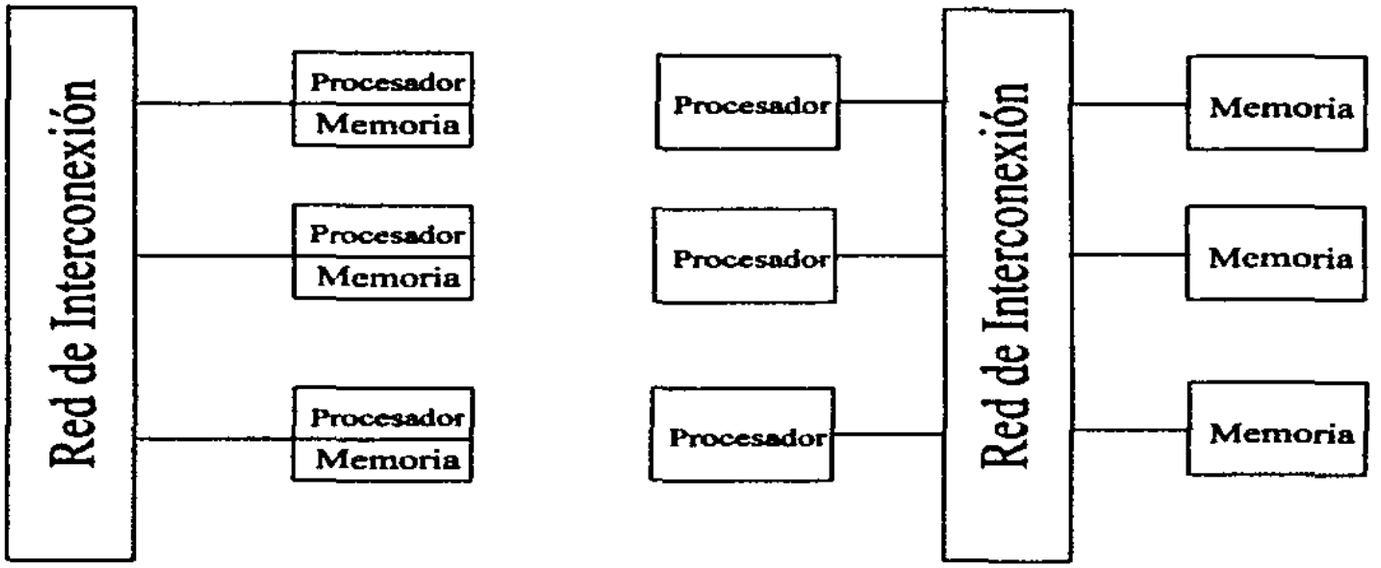



***Figura B.1 Arquitectura de Paso de Mensajes***

Para que dos (o más) procesadores puedan intercambiar información, estos deben establecer una línea de comunicación ya sea estática o dinámica, esto es, que dicha línea sea inamovible o que pueda dirigirse de un procesador a cualquier otro.

### B.2.2 Arquitecturas de Memoria Compartida

Las arquitecturas de memoria compartida proveen soporte de hardware para la lectura y escritura para compartir dicha memoria. Los procesos interactuan modificando los objetos almacenados en el espacio de direccionamiento de memoria. En una arquitectura como esta, el principal problema a resolver estriba en que dos procesos en distintos procesadores deseen accesar las mismas localidades de memoria en un mismo instante. Para solucionar dicho problema se deben establecer los mecanismos adecuados de sincronización (por ejemplo el uso de semáforos). En la figura (C.2) se presenta la arquitectura de una computadora de memoria compartida.




**Arquitectura de Memoria de Acceso No Uniforme y Memoria Local**

**Arquitectura de Memoria de Acceso Uniforme**

***Figura B.2 Arquitecturas de Memoria Compartida***

### B.2.3 Redes de Interconexión

Las arquitecturas paralelas de memoria compartida y las de paso de mensajes pueden ser construidas conectando unidades de proceso y memoria utilizando una red de interconexión. Una red de interconexión se puede clasificar en **estática o dinámica.** Las redes estáticas consisten de enlaces (links) de comunicación punto a punto entre procesadores. Las redes estáticas son comúnmente utilizadas para construir arquitecturas de paso de mensajes. Las redes dinámicas son construidas utilizando enlaces de comunicación y "switchs". Los enlaces son conectados unos con otros de manera dinámica a través de los switchs para establecer trayectorias de comunicación entre los procesadores y la memoria. Las redes dinámicas son referidas como redes **indirectas** y son usadas normalmente para construir arquitecturas de memoria compartida.

### B.2.4 Granularidad de Procesadores

Una computadora de procesamiento paralelo puede ser constituida de un número pequeño de procesadores muy potentes; o de un gran número de procesadores relativamente menos potentes[8] [12]. Las computadoras pertenecientes al primer grupo son llamados frecuentemente computadoras de grano grueso (coarse grain). Y las que se encuentran en el segundo grupo son denominadas computadoras de grano fino (fine grain).

Las computadoras de grano grueso son considerablemente más caras que aquellas de grano fino. La razón de esto es que los rápidos procesadores utilizados en las arquitecturas de grano grueso, no son producidas en gran escala. Además, estos procesadores requieren de técnicas de fabricación muy costosas. Aunque esta aseveración tiene ciertos fundamentos, en la actualidad una computadora de grano grueso puede ser construida a partir de procesadores comerciales que son fabricados en grandes volúmenes con lo cual reduciría el costo extra antes mencionado.

Diversas aplicaciones son aptas para computadoras de grano grueso o de grano fino. Muchas aplicaciones tienen solo un número limitado de concurrencia. Dichas aplicaciones no pueden hacer un uso efectivo de un número de procesadores no muy potentes y esta es más factible de ser implementada en una arquitectura de grano grueso.

## B.3 Topologías en Arquitecturas Paralelas

Como se menciono en párrafos anteriores, existe el problema de ensamble de unidades de proceso, el cual se divide en cuatro categorías según Flynn. Dentro de esta categoría, las maquinas MIMD es la de propósito más general de las cuatro. La arquitectura más general de MIMD es frecuentemente construida de dispositivos ampliamente utilizados como el microprocesador 68XXX o los intel 80XXX. Por lo tanto, la mayor consideración de diseño en la producción de dichos sistemas multiprocesadores es la topología[11], la cual es la medida del camino en el cual las trayectorias entre los distintos procesadores es diseñada.

Existen cinco topologías clásicas para las maquinas MIMD, que son incidentalmente las mismas que existen para el diseño de una red de área local (a excepción de la topología de hipercubo). Las estructuras multiprocesadores son descritas tanto por la topología como por su nivel de interconexión. El nivel de interconexión es una medida del número de switchs a través de los cuales un mensaje debe pasar desde un procesador a otro. Las cuatro topologías básicas son la topología sin restricciones, la de bus, la de anillo y la de estrella, aunque por supuesto, existen

muchas otras variantes de cada una de estas topologías puras y muchas mezclas de ellas.

### B.3.1 Bus

La topología de bus es la más simple ya que cada procesador esta conectado a una sola línea de conexión común, denominada el bus del sistema. Es la más simple por que no solo elimina el problema del ruteo de mensajes entre procesadores, si no que además, este bus es una extensión del bus encontrado en cualquier computadora. Por lo que es posible utilizar un bus ya existente para crear un sistema multiprocesador. Todo lo que se requiere es un mecanismo que traslade el control de un microprocesador a otro.

La desventaja de la topología de bus como método para implementación de sistemas multiprocesadores radica en el problema de controlar el acceso al bus. Solamente un procesador en un instante dado puede accesar el bus. El control de que procesador debe tomar el bus en un momento dado crea un estado de ocio y un cuello de botella.

### B.3.2 Anillo

En un anillo, cada procesador se conecta solo a sus dos más cercanos vecinos. Uno de ellos es llamado el vecino de arriba y el otro es llamado el vecino de abajo. Un nodo recibe la información de el nodo vecino de abajo y puede transmitir información hacia el nodo vecino de arriba. De esta manera, la información fluye de alrededor del anillo en una sola dirección, y cada paquete de información pasa por cada nodo en el anillo. La información transmitida contiene la dirección de destino. Cuando un nodo recibe un paquete, este checa la dirección y, si el paquete corresponde a su dirección propia, el nodo lee la información. Similarmente, el nodo es capaz de adicionar paquetes propios de información al paquete y transmitirlos por el anillo.

### B.3.3 Estrella

La topología de estrella emplea un procesador central como un relevador (switch), muy parecido a un conmutador telefónico, entre los otros procesos que están conectados lógicamente (o fisicamente) alrededor del nodo central. La ventaja de esta topología es que reduce los problemas ocasionados por una conexión tipo bus, ya que no existe una conexión compartida entre distintos nodos. por otro lado, la topología tipo estrella es tan buena como su procesador central. Si este nodo falla, el sistema entero falla. Esta topología no es muy popular en las arquitecturas paralelas.

### B.3.4 Hipercubo

Un multiprocesador tipo hipercubo de n dimensiones conecta $N = 2^n$ procesadores en forma de un cubo de n dimensiones. Cada esquina (vértice o nodo) del hipercubo consiste de un elemento de proceso y su memoria asociada. Debido a la topología, cada nodo esta directamente interconectado a exactamente n nodos.

Cada procesador en el hipercubo tiene una dirección de n-bits (de 0 a $2^{n-1}$) y cada uno de los vecinos más cercanos de un nodo particular, tiene una dirección que difiere en exactamente un bit de este nodo.

El hipercubo es una de las topologías más interesantes ya que es particularmente enfocada a cierto grupo de algoritmos. En especifico a problemas que involucran la evaluación de la *FFT* (Fast Fourier Transform, Transformada Rápida de Fourier). El primer arreglo practico tipo hipercubo fue construido en Caltech en 1983. Este estaba basado en 64 procesadores 8086 más 64 co-procesadores matemáticos 8087.

### B.3.5 Topologías Híbridas

Estas son construidas en base de las distintas topologías ya vistas y producen un efecto significativo en cuanto a la versatilidad y el rendimiento de un algoritmo especifico.

# APÉNDICE C - Descripción del Algoritmo para la FFT

La técnica más antigua para realizar el análisis de una señal es la bien conocida *DFT* (Transformada Discreta de Fourier) que nos sirve para descomponer en sus partes "atómicas" una señal estacionaria dada. Este ha sido el algoritmo elegido para realizar este proyecto, pero cabe destacar que existen otros algoritmos los cuales han sido revisados brevemente en este escrito y bien podrían servir para esta tarea. A continuación describimos en que consiste el calculo de la DFT a través del bien conocido algoritmo FFT.

Básicamente el problema computacional de la DFT es el calculo de la secuencia $\{X(k)\}$ de N números complejos dada otra secuencia de datos $\{x(k)\}$ de longitud N de acuerdo a la formula (C.1)

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, \qquad 0 \le k \le N-1 \qquad \text{(C.1)}$$

donde $W_N = e^{-j2\pi/N}$

Se observa que para cada valor de k, la computación directa de la DFT de X(k) involucra N multiplicaciones complejas (4N multiplicaciones reales) y N - 1 adiciones complejas (4N - 2, adiciones reales). La computación directa de la DFT es ineficiente pues esta no explota las propiedades de *simetría* y *periodicidad* de los factores $W_N$. En particular las dos propiedades son:

$$W_N^{k+N/2} = -W_N^k \qquad \text{Simetría} \qquad \text{(C.2a)}$$

$$W_N^{k+N} = W_N^k \qquad \text{Periodicidad} \qquad \text{(C.2b)}$$

El desarrollo de algoritmos computacionalmente eficientes para el calculo de la DFT es posible si se adopta la filosofía de "divide y vencerás". Este enfoque esta basado en la descomposición de los N puntos de la DFT en segmentos sucesivos más pequeños. Para comenzar, podemos expresar la DFT en dos partes. Una parte real (R) y una parte compleja (Y), por lo que podemos expresar dichas partes como sigue:

$$X_R(k) = \sum_{n=0}^{N-1} [x_R(n)\cos(\frac{2\pi kn}{N}) + x_I(n)\sin(\frac{2\pi kn}{N})] \qquad \text{(C.3)}$$

$$X_I(k) = \sum_{n=0}^{N-1} [x_R(n) sin(\frac{2\pi kn}{N}) + x_I(n) \cos(\frac{2\pi kn}{N})] \qquad \text{(C.4)}$$

El calculo directo de (C.3) y (C.4) requiere de:

- $2N^2$ Evaluaciones de funciones trigonométricas
- $4N^2$ Multiplicaciones reales
- $4N(N-1)$ Adiciones reales
- Un cierto número de operaciones de indexacion y direccionamiento

Las operaciones calculadas en tres primeros puntos son para obtener $X_R(k)$ y $X_I(k)$. Las operaciones de indexamiento y direccionamiento se utilizan para obtener los datos de x(n), $0 \le n \le N-1$ y los factores de fase para posteriormente almacenar los resultados. La mayoría de los algoritmos para calcular la DFT optimizan alguno de estos procesos de alguna manera u otra.

En aplicaciones en donde los N elementos de la DFT de varias secuencias son computados, es ineficiente el recalcular los factores de fase cada vez. En este caso es más eficiente calcular los factores de fase una sola vez y almacenarlos en memoria. En general no es necesario almacenar todos los factores de fase de los cuatro cuadrantes. Esto reduce considerablemente el calculo de la DFT.

Otra de las técnicas empleadas es el aprovechamiento de la simetría que guardan intrínsecamente los cálculos con respecto a los datos. Para ilustrar estas nociones consideremos el calculo de una DFT de N puntos (datos) en donde N pueda ser factorizado como un producto de dos enteros de la forma:

$$N = L M$$

Ahora la secuencia $x(n), 0 \le n \le N-1$, puede ser almacenada en un arreglo de una sola dimensión indexado por n o como un arreglo de dos dimensiones indexado $l$ y $m$, donde $0 \le l \le L-1$ y $0 \le m \le M-1$. De esta maneara podemos mapear desde un índice único n hacia (l, m).

Supongamos ahora que x(n) es mapeado a un arreglo rectangular x(l, m) y X(k) es mapeado a su correspondiente arreglo rectangular X(p, q). Entonces la DFT puede

ser expresada como la doble sumatoria sobre los elementos del arreglo rectangular multiplicados por su correspondiente factor de fase. Entonces tenemos:

$$X(p,q) = \sum_{m=0}^{M-1} \sum_{l=0}^{L-1} x(l,m) W_N^{(Mp+q)(mL+l)} \qquad \text{(C.5)}$$

en donde $W_N^{(Mp+q)(mL+l)} = W_N^{MLmp} W_N^{mlq} W_N^{Mpl} W_N^{lq}$ y tomando en cuenta que $N = LM$ podemos simplificar la expresión y de esta manera obtener la ecuación (C.6):

$$X(p,q) = \sum_{l=0}^{L-1} [W_N^{lq} [\sum_{m=0}^{M-1} x(l,m) W_M^{mq}]] W_L^{pl} \qquad \text{(C.6)}$$

Esta ecuación conlleva el calculo de la DFT de longitud M y longitud L. La ecuación (C.6) permite dividir el proceso de obtención de la DFT en tres partes principales:

1. Se calculan los M puntos de la DFT:

$F(l,q) = \sum_{m=0}^{M-1} x(l,m) W_M^{mq}$ para $0 \le q \le M-1$ para cada renglón l = 0, 1, ..., L -1

2. Se calcula el nuevo arreglo rectangular G(l, q) definido como:

$G(l,q) = W_N^{lq} F(l,q)$ para $\{\frac{0 \le l \le L-1}{0 \le q \le M-1}$

3. Por ultimo, se calculan los L puntos:

$X(p,q) = \sum_{l=0}^{L-1} G(l,q) W_L^{lp}$ para cada q = 0, 1, ...,M-1 del arreglo $G(l, q)$

A primera vista podríamos decir que el procedimiento anterior requiere de un esfuerzo computacional mayor que si calculásemos directamente la DFT. Más sin embargo podemos observar como en el primer paso calculamos $L$ DFTs, cada una de $M$ datos. Por lo que este paso requiere de $LM^2$ multiplicaciones complejas y de también de $LM(M - 1)$ adiciones complejas. El segundo paso requiere de $LM$ multiplicaciones complejas. Finalmente el tercer paso requiere de $ML^2$

multiplicaciones complejas y de $ML(L - 1)$ adiciones complejas. Por lo que la complejidad computacional es de:

| | |
|---|---|
| Multiplicaciones Complejas: | $N(M + L + 1)$ |
| Adiciones Complejas: | $N(M + L - 2)$ |

donde $N = ML$. Por lo que el número de multiplicaciones ha sido reducido de $N^2$ a $N(M + L + 1)$ y el número de adiciones ha sido reducido de $N(N - 1)$ a $N(M + L - 2)$.

Cuando N es un número altamente compuesto, esto es, N puede ser factorizado en un producto de números primos de la forma:

$$N = r_1 r_2 \ldots r_\nu \qquad \text{(C.7)}$$

La descomposición anterior puede ser repetida $(\nu - 1)$ veces. Este procedimiento da como resultado DFTs más pequeñas que por lo tanto producen un algoritmo computacionalmente más eficiente. Un caso particularmente importante es cuando los factores $r$ son iguales. por lo que $N$ estará definido como $N = r^\nu$. En este caso las DFTs son de tamaño r, así que la computación de los $N$ datos de la DFT tiene un patrón regular. El número $r$ es llamado *radix* del algoritmo FFT [5][6]. Los radix más comúnmente empleados son el 2 y 4. En los párrafos siguientes consideraremos el caso particular cuando $r = 2$.

Considerando que $N = r^\nu$ datos, podemos realizar una división de N/2 datos con lo cual obtenemos dos secuencias de datos $f_1(n)$ y $f_2(n)$ que corresponden a las muestras pares y a las muestras nones de $x(n)$ respectivamente.

$$f_1(n) = x(2n)$$
$$f_2(n) = x(2n+1) \qquad n = 0, 1, \ldots, N/2$$

$f_1$ y $f_2$ son obtenidas por decimación de $x(n)$ en un factor de 2, de aquí que al resultado del algoritmo de la FFT se le conozca como: *FFT con decimación en tiempo.*

La decimación de la secuencia de datos puede ser repetida una y otra vez hasta que el resultado de las secuencias se vean reducidas a su mínima expresión. Esto es, para $N = 2^\nu$, por medio de la decimación obtenemos que $\nu = \log_2 N$ veces. Por lo que el número total de multiplicaciones complejas se vera reducido a $(N/2)\log_2 N$ y el número de sumas complejas es $N \log_2 N$[4]. En la figura siguiente (C.1) se ilustra el calculo de una transformada de ocho datos.

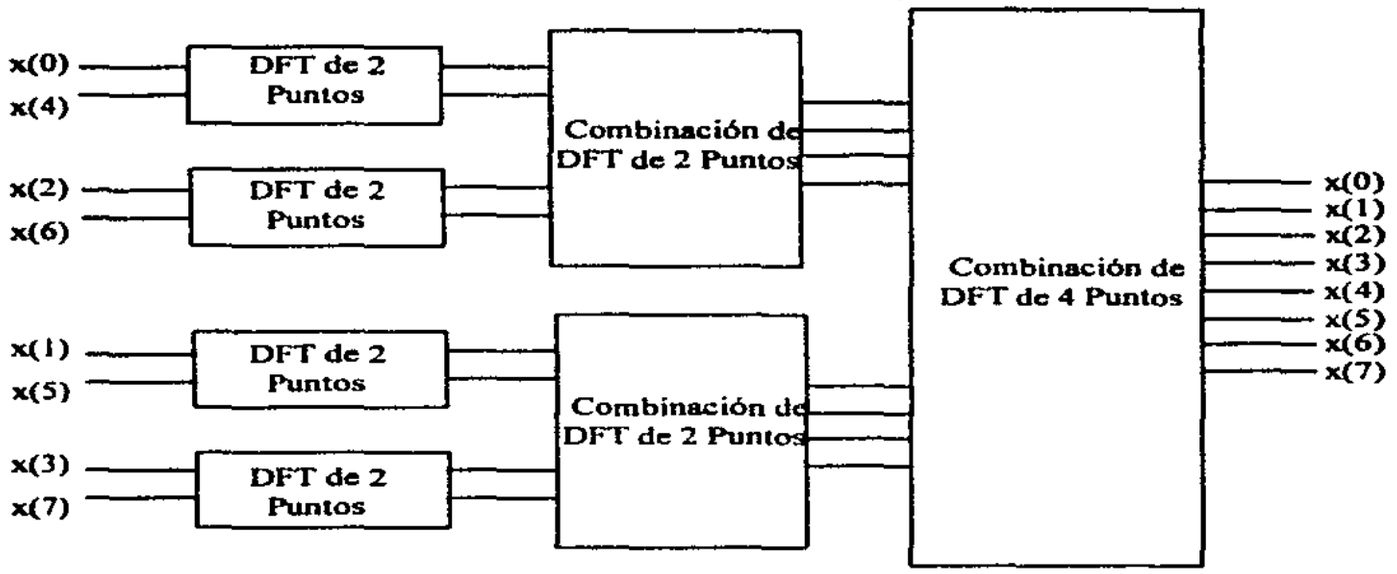



***Figura C.4 Calculo de una DFT de ocho puntos***

Como podemos observar en la figura (C.1), el orden de la secuencia de los datos de entrada es diferente al orden de la secuencia de los puntos resultado. Existe un desordenamiento en la secuencia de datos bien definida. Esto ocurre por la manera en como se realiza la decimación y los cálculos de la secuencia. La figura siguiente (C.2) muestra este desordenamiento y nos presenta de manera gráfica la solución para que la secuencia este ordenada después de ser procesada. A dicha solución se le conoce como "bit-reverse" (inversión de bits).

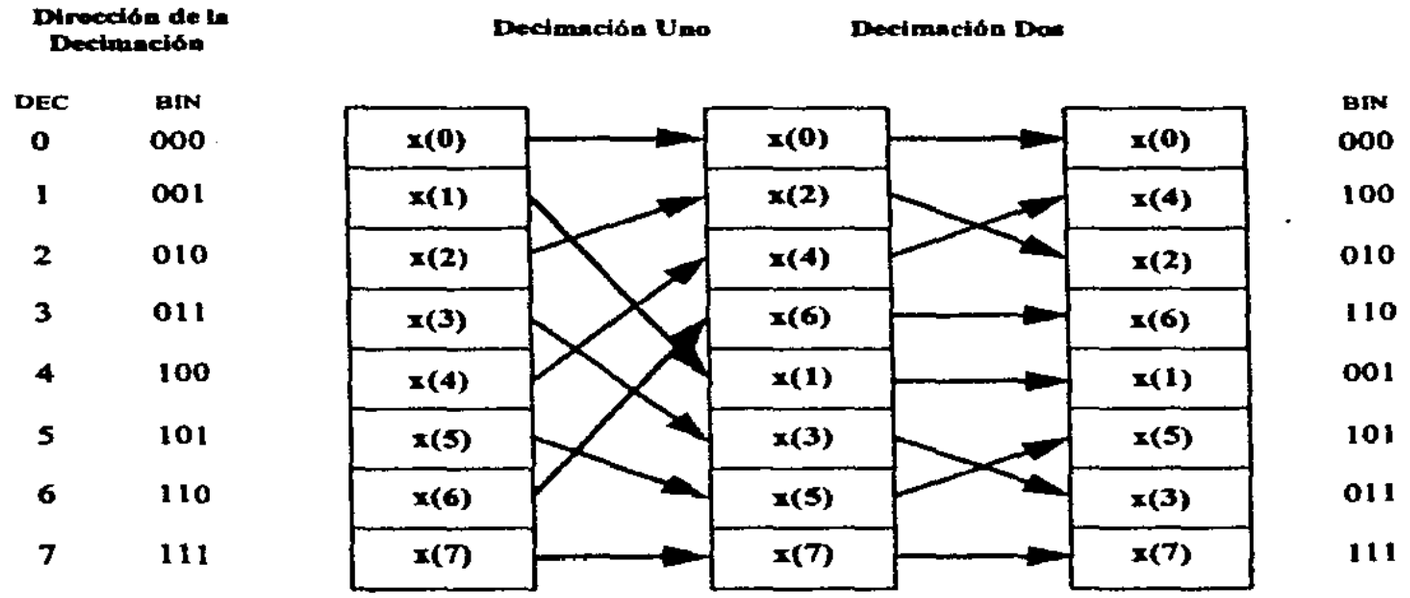



*Figura C.5 Decimación de Datos y Bit-Reverse*

# REFERENCIAS


- [1] Kay, Steve M; Modern Espectral Estimation: Theory and Application, Prentice Hall 1988

- [2] Kay, Stev M, Marple Stanley Lawrence; Spectrum Analysis - A Modern Perpective; Proceedings of IEEE 1981

- [3] Oppenheim, A. V; and R. W. Schafer; Digital Signal Processing, Prentice Hall 1975

- [4] Proakis, John G; and Manolakis, Dimitris G; Digital Signal Processing: Principles, Algorithms, and Applications, Macmillan 1992

- [5] Ramirez, Robert W; The FFT: Fundamentals and Concepts, Prentice Hall 1985

- [6] Herman Roebbers, Peter Welch, Klaas Wijbrans Van Rietschoten & Houwens; A Generalized FFT Algorithm on Transputers; University of Twente; AE Enschede; University of Kent

- [7] Graham, Ian and King, Tim; The Transputer Handbook, Prentice Hall 1991

- [8] Kumar V; Grama A; Gupta A; Karypis G; Introduction to Parallel Computing, Benjamin Cummings 1993

- [9] Howe, Carl D., Moxon, Bruce; How to Program Parallel Processors; Spectrum IEEE 1987

- [10] Kruatrachue, Boontee and Lewis, Ted; Grain Size Determination for Parallel Processing, IEEE Software, January 1988

- [11] Clements, Alan; Multiprocessor Systems, Electronics & Wirelessworld, 1987

- [12] Karp, Alan H; and Flatt, Horace P., Measuring Parallel Processor Performance, Comunications of ACM May 1990

- [13] Hennessy, Jhon L , Patterson David A.; Arquitectura de Computadores, Un Enfoque Cuantitativo; Mc Graw Hill 1993

- [14] Pountain Dick, May David; A Tutorial Introduccion to OCCAM Programing; Inmos BSP Professional Books 1988

- [15] Fish, Peter; Diagnostic Medical Ultrasound, Jhon Wiley & Sons 1990

- [16] N. Burns, Peter; Doppler Ultrasound, IEEE Short Course, Seattle, Nov 7 1995

- [17] Eugene F. Bernstein; No Invasive diagnostic techniques in Vascular Disease, The C. V. Mosby Company 1985

- [18] Graca Ruano M., Garcia Nocetti D. F., Fish P. J., Fleming P. J.; Alternativ Parallel Implementacions of an AR-Modified Covariance Spectral Estimation for Diagnostic Ultrasonic Blood Flow Studies; Parallel Computing 1993

- [19] Vaitkus P. J., Cobbold R. S. C.; A Comparativ Study and Assessment of Doppler Ultrasound Spectral Estimation Techniques Part I : Estimation Methods; Ultrasound in Medical & Biology 1988

- [20] Vaitkus P. J., Cobbold R. S. C.; A Comparativ Study and Assessment of Doppler Ultrasound Spectral Estimation Techniques Part II : Methods and Results; Ultrasound in Medical & Biology 1988

- [21] Conger, L. James; Windows API Bible; Waite Groups 1992

- [22] Transputer Windows Server Package Release 3.0, User Guide; Elcom LTD. 1995

- [23] Transputer Windows Server Package Release 3.0, Reference Guide and Functions; Elcom LTD. 1995

- [24] Transputer Windows Server Package Release 3.0, References Guide Message; Elcom LTD. 1995

- [25] Instrupedia; National Instruments 1995

- [26] Tromba Anthony J., Marsden Jerrold E.; Calculo Vectorial; Fondo de Cultura Economica 1981

- [27] Ansi C Tooset Handbook; Inmos 1992

- [28] Harbison Samuel P., Steel Guy L.; C Reference Manual; Prentice Hall 1991

- [29] Benitez Perez, Hector; Diseño de una Arquitectura para Procesamiento Paralelo de Señales Doppler de Ultrasonido; UNAM 1994

- [30] TRANSTECH Parallel Systems; Installation Maual, Transtech 1993