UN MODELO COMPUTACIONAL PARA EL PROCESAMIENTO DE LISTAS DE FUNCIONES RECURSIVAS LINEALES*

(A Computational Model for List Processing of Linear Recursive Functions)

Carlos Alfonso Acosta León
Eurípides Montagne!
Rina Surós

Centro de Computación Paralela y Distribuída – CCPD.
Escuela de Computación. Facultad de Ciencias. Universidad Central de Venezuela.
Apartado 47002, Caracas 1041-A. Venezuela
Teléfonos: (58-02) 6051134 Fax: 6051131
cacosta@strix.ciens.ucv.ve

Resumen

Se propone un modelo computacional orientado al procesamiento de listas de funciones recursivas lineales. En este modelo conceptual, denominado Máquina Paralelo-Recursiva o MPR, son aprovechados dos importantes esquemas de cómputo que rigen su organización y funcionamiento: *el paralelismo y la recursión*. En este enfoque es explotado el *paralelismo* en grano fino mediante el pipeline o encauzamiento, la concurrencia y el control por flujo de datos o dataflow. La *recursión* es usada para organizar la estructura de la MPR y controlar el cómputo masivo de grandes cadenas de funciones recursivas lineales. Los componentes esenciales que conforman este modelo son: una cola de funciones recursivas lineales a ser evaluadas, dos pipeline o cauces de procesadores, el intercambio de mensajes como mecanismo de comunicación entre los procesadores, y un conjunto de operadores elementales que conforman su repertorio básico de instrucciones. El presente trabajo muestra el modelo de la MPR y un estudio de su rendimiento.

Palabras Claves: Modelo de Computación, Funciones Recursivas Lineales, Recursión o Recursividad, Paralelismo, Pipeline o encauzamiento, flujo de Datos o Dataflow, Evaluación del Rendimiento.

Abstract.

We propose a computational model oriented to list processing of linear recursive functions. In this conceptual model, denominated Parallel-Recursiva Machine or PRM, two important computation concepts that govern their organization and operation are taken advantage of: the parallelism and the recursivity. In this focus the parallelism is exploited in fine grain by means of the pipeline, the concurrency and the control by dataflow. The recursivity is used to organize the structure of the PRM and to control the compute massive of large chains of linear recursive functions. The essential components that conform this model are: a list of linear recursive functions to be evaluated, two pipeline of processors, the exchange of messages like communication mechanism among the processors, and a group of elementary operators that conform the basic instructions set. The present work shows the PRM model and a study of its performance.

Keywords: Computation Model, Linear Recursive Functions, Recursivity, Parallelism, Pipeline, Dataflow, Performance Evaluation.

* Agradecimiento a la cooperación prestada por las siguientes instituciones que apoyan la investigación en Venezuela: el CDCH de la UCV, proyecto 03-02-4257-98 y el CONICIT, proyecto S1-9500706.

! Disfruta su año Sabático en: School of EECS, University of Central Florida, Orlando, FL 32816.

1. Introducción.

El avance en diferentes áreas científicas y tecnológicas ha sido impulsado por la creciente necesidad de resolver problemas complejos en cuotas razonables de tiempo. Muchos de estos problemas, en las matemáticas y en las ciencias de la computación, son inherentemente recursivos. Ejemplos de estos son: las funciones factoriales, las relaciones de recurrencia, el procesamiento de listas, el análisis sintáctico en el diseño de compiladores, los algoritmos de ordenamiento, etc., entre otros. El cálculo que involucran estos problemas, no puede llevarse a cabo eficientemente usando el esquema tradicional de cómputo secuencial de Von Neumann. Lo más natural es que sea utilizado el enfoque recursivo para construir procedimientos y algoritmos para su resolución. A raíz de esto, se han planteado arquitecturas de computación concurrentes y paralelismo para un procesamiento mas eficiente. También, en la Teoría de la Computación la recursión ha sido el método usado para describir Funciones Computables, ver los trabajos de Church y Turing [4,8]. Pero, la recursión se ha usado también como un modelo para definir la organización y funcionamiento de computadores [11]. Los computadores que siguen este enfoque son conocidos como Máquinas Recursivas, y están basados en la utilización de varios elementos de procesamiento idénticos y lógicamente recursivos, cada uno con suficiente autonomía para operar independiente y concurrentemente, interconectados mediante una estructura regular como un vector o un árbol, formando un computador descentralizado donde estos elementos de procesamiento cooperan en la ejecución de una tarea recursiva. Esta, y otras organizaciones paralelas que utilizan la concurrencia y la recursión para la resolución de problemas, están motivadas por la necesidad de incrementar el rendimiento de los computadores, lo cual, en muchos casos, no puede lograrse utilizando solamente la tecnología [7,13].

Sin embargo, hasta ahora los intentos por implantar eficientemente la recursión han sido infructuosos. Por ejemplo, en los lenguajes de programación imperativos, la recursividad se ha implantado mediante un mecanismo de software para el manejo y administración de pilas en memoria, que causan un "overhead" en tiempo de ejecución. Por otro lado, en los lenguajes funcionales se utiliza la recursión para la implantación de procesos iterativos y para definir las funciones recursivamente. Pero, en otros trabajos como en [2], se indica cómo el paralelismo está ligado a las estructuras de computación recursivas utilizadas por los lenguajes funcionales, y se propone que la forma eficiente de explotar la recursión lineal es a través de sistemas de procesadores encauzados que interactúen mediante la comunicación de paquetes y, la recursión no lineal o en árbol por medio de sistemas jerárquicos de procesadores. Por tal motivo se han propuesto algoritmos paralelos para la resolución eficiente de problemas recursivos en máquinas paralelas, como por ejemplo la resolución de recurrencias lineales sobre máquinas de procesadores encauzados [15] y la implantación de funciones recursivas lineales en arreglos de Transputers [5].

En el presente trabajo se muestra un modelo computacional orientado al procesamiento de grandes listas de funciones recursivas lineales o de problemas que puedan plantearse usando la recursión. Se plantea la estructura, funcionamiento y aplicaciones computacionales de este modelo conceptual denominado Máquina Paralelo-Recursiva o MPR, que explota el paralelismo y la recursión implícita en ciertos problemas. También, se hace un análisis de la MPR mediante un modelo determinístico para calcular los tiempos de ejecución con diferentes cargas de trabajo y determinar su eficiencia.

1.1 Organización del documento.

El trabajo comienza con una breve revisión de conceptos asociados, como funciones recursivas y computabilidad, diferentes modelos de computación paralela - encauzamiento y flujo de datos, en la sección 2. En la sección 3 se explican los fundamentos, estructura y funcionamiento del modelo de cómputo propuesto en la MPR. Se explica el caso de solución a un problema recursivo lineal, el factorial, en la sección 4. En la sección 5 se presenta el análisis de rendimiento del modelo. Finalmente, en la sección 6 se presentan algunas conclusiones relativas al modelo de la MPR en el cual se determinan aquellos aspectos que permiten optimizar su rendimiento.

2. Recursividad, computabilidad y modelos de computación.

2.1 Funciones Recursivas y Computabilidad

Una función recursiva es una función que está definida en términos de sí misma. La definición recursiva está determinada de forma tal, que especifica un procedimiento efectivo para evaluar la función, y envuelve: Una base o condición límite de parada, que establece explícitamente ciertos valores que son, por definición, parte de la función. Una regla de construcción recursiva, la cual indica como determinar otros valores de la función a partir de valores ya conocidos.

Así como la recursión es una manera natural y compacta para definir muchas funciones, al mismo tiempo es suficientemente poderosa para describir funciones computables. Las funciones computables son todas aquellas funciones que pueden calcularse por medios algorítmicos, sin importar como pueda expresarse o implantarse ese algoritmo [4]. Estas funciones pueden calcularse por medio de Máquinas de Turing (MT). La tesis de Turing representa de forma explícita los límites de los procesos computacionales. Otra forma de estudiar la computabilidad y sus capacidades, es utilizando un enfoque funcional [4,8]. Se comienza con una colección de funciones, llamadas Funciones Base, cuya sencillez es tal, que no hay duda acerca de su computabilidad. Así, si un sistema computacional abarca todas las Funciones Base y a las estrategias de construcción de funciones, se concluye que el sistema tiene todo el poder computacional posible. Pero, la extensa naturaleza de las Funciones Recursivas Primitivas no abarca a todas las funciones computables. Para ampliar el conjunto de funciones computables más allá de las funciones recursivas primitivas, se aplica la técnica minimalización, junto con la composición y la recursividad primitiva, lo que da lugar a las Funciones Recursivas Generales. Así como la tesis de Turing propone que la clase de Máquinas de Turing posee el poder computacional de cualquier sistema de computación, la clase de Funciones Recursivas Generales contiene a todas las funciones computables. Esta tesis se conoce como tesis de Church. Por ello se afirma, que todo proceso computacional realizado por un MT es, en realidad, el cálculo de una Función Recursiva General.

Este concepto, además de los métodos de computación paralela que a continuación se mencionarán, son el centro de interés en el modelo propuesto.

2.2 Modelos de cómputo paralelo.

Existen varios modelos o esquemas de computación paralela. Por ejemplo se pueden mencionar a propósito del presente trabajo: el ***Cómputo Encauzado o en Pipeline,*** que explota el tratamiento de varias instrucciones o datos, aprovechando que su ejecución se puede dividir en varias etapas seriales e independientes. Esto introduce un paralelismo a nivel instrucciones que permite tomar ventaja del procesamiento de problemas de grano fino. Otro modelo de gran importancia para el presente trabajo es

el ***Cómputo por Flujo de Datos.*** Aquí, un sistema de Flujo de Datos se construye a partir de módulos de procesamiento, interconectados a través de enlaces usados para el intercambio de mensajes. Cada módulo ejecuta una parte del programa y comunica a otro su resultado. La estructura de un sistema de Flujo de Datos puede describirse mediante un grafo dirigido donde cada nodo es un operador o función primitiva y los arcos representan las dependencias entre instrucciones. Los valores son transmitidos como 'tokens" de datos, a través de estos arcos. El nivel de concurrencia, se puede incrementar enormemente conectando muchos elementos de procesamiento a través de un sistema de comunicación.

2.3 Rendimiento de los Modelos de cómputo paralelo.

La velocidad de operación de un computador tipo Von Neumann convencional está limitada principalmente por el ancho de banda de las conexiones entre procesador y memoria y las limitaciones físicas impuestas por las tecnologías de construcción de circuitos [7]. Una forma de obtener las prestaciones necesarias es utilizando procesamiento paralelo, donde varios procesadores trabajan sobre el mismo problema. La regla es mejorar el rendimiento de los sistemas de computación. El rendimiento de estos sistemas, puede medirse a través de varios parámetros [7,10,13] de los cuales, el más frecuentemente utilizado es la Aceleración (Ver Ley de Amdahl). Veamos,

Ley de Amdahl. Aceleración (Sp: Speed up): Sea **p** un número de procesadores, la aceleración se define como la razón del tiempo estimado cuando se ejecuta un programa en un solo procesador (T_{seq}) entre el tiempo estimado cuando se ejecuta el mismo programa en un sistema con **p** procesadores (T_{par}); es decir: $Sp = \frac{T_{seq}}{T_{par}}$

Teóricamente, la aceleración debería ser una relación lineal; es decir, si se tienen p procesadores se podría ejecutar el programa p veces más rápido. En la práctica, esto no se cumple debido a varios factores como son el "overhead" de comunicación, las relaciones de dependencias entre instrucciones y el desbalance de la carga de trabajo entre los procesadores. De hecho, la Conjetura de Minsky [7,13] dicta una Aceleración máxima de Log2p. Por otra parte, la Ley de Amdahl [7,10] establece una cota superior determinada por la fracción del algoritmo que no puede ser paralelizada de la siguiente forma:

Sea **a** la porción del programa que no puede ser paralelizada y **p** el número de procesadores, entonces:

$$T_{par} = \alpha T_{seq} + (1-\alpha) T_{seq}/p \text{ , por lo tanto: } Sp = p/(1+(p-1)\alpha)$$

Esto significa, que aunque el número de procesadores aumente, la ganancia obtenida estará limitada siempre por la fracción secuencial, esto se puede ver claramente si:

$$\lim_{p \to \infty} Sp = \lim_{p \to \infty} p/(1+(p-1)\alpha) = \lim_{p \to \infty} 1/[1/p + (1-1/p)*\alpha] = 1/\alpha$$

Por otro lado, para algunos problemas la fracción secuencial disminuye a medida que aumenta el tamaño de los mismos, es decir, **a** se expresa como función de **n**, **a(n).** Sea **n** el tamaño del problema, entonces se tiene que: $\lim_{n \to \infty} Sp = p/(1+(p-1)\alpha(n)) = p$

Estas ecuaciones serán usadas mas adelante como una herramienta para determinar el rendimiento determinístico del modelo computacional propuesto en este trabajo.

3. Descripción del Modelo Computacional de la MPR.

El modelo conceptual de computación propuesto para el procesamiento de funciones recursivas lineales en paralelo se basa en los estudios realizados en [3,4,5], donde se plantea un modelo de máquina orientada al procesamiento de grandes listas de funciones recursivas lineales o de problemas cuya solución pueda expresarse de forma recursiva. Esta carga de trabajo representa un procesamiento de grano fino. A este modelo se le ha denominado Máquina Paralelo Recursiva –MPR– e involucra conceptos computacionales importantes, tales como: *Paralelismo* (explotado mediante *el Pipeline, la Concurrencia, el control por el Flujo de Datos) y Recursión* (usada en la organización de su estructura y el control del cómputo).

En el modelo se identifican los siguientes elementos esenciales: la cola de funciones recursivas lineales a ser evaluadas, dos cauces o pipeline de procesadores, el intercambio de mensajes como mecanismo de comunicación entre los procesadores, y el repertorio de operadores elementales de los procesadores.

Esta máquina esta concebida para funcionar como un procesador numérico especializado conectado a un computador Front-End o Host del cual se obtiene la carga inicial de trabajo "batch" a procesar en forma de una lista o cola ordenada de funciones recursivas lineales y a quien devuelve los resultados al finalizar el procesamiento. A continuación se describe la estructura y funcionamiento de la MPR.

3.1 Organización y Estructura de la MPR. El modelo de la MPR es concebida con una topología estática donde se integran cuatro componentes básicos. Ver Figura 1a. Esto es,

- $\mathbf{L_{qent}}$: es una gran lista de entrada ordenada con las funciones recursivas lineales a procesar,
- **UCM:** es la una unidad controladora de la asignación y liberación de direcciones de memoria,
- $\mathbf{MPR_n}$: es una pila formada por ***n*** niveles Paralelos-Recursivos, o sea, la profundidad de la MPR, y
- $\mathbf{L_{qsal}}$: es una gran lista de salida con los resultados de las funciones evaluadas.

La MPR_n posee una organización con una jerarquía de múltiples niveles idénticos, y donde cada nivel es una n-tupla de objetos de la forma $\mathbf{N_i = (PB_i, ML_i, PS_i)}$. Ver Figura 1a**.** En esta estructura, el procesador $\mathbf{PB_i}$ es denominado procesador de bajada del i–esimo nivel de la MPR, y es especificado como $\mathbf{PB_i = (CC_i, Rp)}$, donde:

- $\mathbf{CC_i = (C_{ent}, C_{sal}, C_{int})}$ es el conjunto de canales de comunicación unidireccionales, confiables y punto a punto que lo interconecta con los demás elementos de la MPR, y
- $\mathbf{Rp = (I_1, I_2, ..., I_n)}$**.** es el repertorio de operaciones elementales común a todos los procesadores.

El conjunto de canales $\mathbf{CC_i}$ están definidos como:

- $\mathbf{C_{ent}}$: es el canal de entrada del $\mathbf{PB_i}$ que lo interconecta a $\mathbf{PB_{i-1}}$ en el nivel predecesor,
- $\mathbf{C_{sal}}$: es el canal de salida del $\mathbf{PB_i}$ que lo interconecta físicamente a $\mathbf{PB_{i+1}}$ en el nivel sucesor, y
- $\mathbf{C_{int}}$: es el canal de salida que interconecta a $\mathbf{PB_i}$ con $\mathbf{PS_i}$, ambos procesadores del mismo nivel.

El procesador $\mathbf{PS_i}$ es denominado procesador de subida del i–esimo nivel, y se especifica de forma similar a $\mathbf{PB_i}$, con la excepción de que solo posee dos canales de comunicación $\mathbf{C_{ent}}$ **y** $\mathbf{C_{sal}}$. El canal de

entrada $\mathbf{C_{ent}}$ interconecta a $\mathbf{PS_i}$ con el $\mathbf{PS_{i+1}}$ en el nivel predecesor, y $\mathbf{C_{sal}}$ es el canal de salida que interconecta a $\mathbf{PS_i}$ con el $\mathbf{PS_{i-1}}$ en el nivel sucesor. Ver Figura 1b.

También, los procesadores de un nivel i comparten una memoria local de dos puertos $\mathbf{ML_i}$, donde cada procesador del nivel esta conectado, cada uno, a un puerto exclusivo de esta memoria.

Por último, los procesadores $\mathbf{PB_0}$ y $\mathbf{PS_0}$ del primer nivel de la MPR, hacen de interfaz de entrada/salida respectivamente con el computador Host o Front-End que servirá de fuente y sumidero de las listas de funciones recursivas.

El modelo teórico de la MPR puede adaptarse al tamaño de los problemas recursivos, expandiendo recursivamente su estructura de forma estática en base a la profundidad de la función recursiva lineal de mayor orden o argumento en la lista $\mathbf{L_{qent}}$ a procesarse. Al expandirse la MPR se forma una pila de $\boldsymbol{n}$ niveles, creándose a su vez dos cauces o pipeline de procesadores: un cauce conformado por los procesadores de bajada ($\boldsymbol{PB_0, PB_1, \ldots, PB_{i-1}, PB_i, PB_{i+1}, \ldots, PB_{n-1}}$) y el otro con los procesadores de subida ($\boldsymbol{PS_{n-1}, \ldots, PS_{i+1}, PS_i, PS_{i-1}, \ldots, PS_1, PS_0}$). De esta forma surge una nueva perspectiva en la organización de la máquina al considerarse el encauzamiento de la lista de funciones recursivas para su procesamiento. Ver Figura 1c.

Otra consideración importante es que los procesadores son elementales, o sea, de grano fino y ejecutan operaciones **Rp** primitivas como: restar, sumar, decrementar, multiplicar, enviar y recibir.

3.2 Funcionamiento de la MPR. La lista de funciones linealmente recursivas, $\mathbf{L_{qent}} = (R_0, R_1, \ldots, R_i, \ldots R_{n-1})$ son requerimientos que provienen del **Host**. La Unidad Controladora de Memoria de la MPR **(UCM)** administra la asignación y liberación de las direcciones de memoria para cada función lineal recursiva que se evalúa. Estas direcciones son únicas y exclusivas para cada una de estas, evitándose los conflictos de acceso a memoria, esto es, $\{\forall R_i \in \mathbf{L_{qent}} \wedge R_j \in \mathbf{L_{qent}} \exists \text{Asig}(R_i) = D_i \wedge \text{Asig}(R_i) = D_j / D_i \neq D_j\}$. Cuando un requerimiento entra a la **MPR**, la UCM le asigna una dirección exclusiva que usará en cada nivel hasta culminar su procesamiento, momento en el cual esa dirección es liberada por la **UCM**. Ver Figura 1c.

Un mensaje $\mathbf{m_i}$ es un paquete de comunicación y se representa mediante una tupla de la forma: **(f, a, d),** donde **f** representa el operador o función, **a** el argumento de la función y, **d** la dirección de memoria local asignada donde la función f almacena sus resultados parciales. La comunicación entre los procesadores en cada cauce se establece mediante el intercambio de mensajes a través de sus canales de comunicación, esto es, durante la secuencia $\mathbf{t_1, t_2, \ldots, t_i}$ de tiempos se transmiten los mensajes $\mathbf{m_1, m_2, \ldots, m_i}$ en el sistema, donde los tiempos $\mathbf{t_1 £ t_2 \ldots £ t_i £ \ldots}$ están ordenados cronológicamente**.** De igual forma los procesadores de un mismo nivel, interactúan comunicándose mediante el intercambio de mensajes por el canal de comunicación punto a punto, o mediante la memoria multipuerto que comparten. Ver Figura 1b.

La evaluación de un requerimiento de la lista de entrada $\mathbf{L_{qent}}$ se inicia en el procesador $\mathbf{PB_0}$, quien toma y procesa el argumento, almacena el resultado parcial en su memoria local, y luego prepara un paquete de comunicación que transmite a su sucesor en el pipeline de bajada. Este proceso continúa secuencialmente en cada $\mathbf{PB_i}$ del pipeline de procesadores de bajada hasta que el requerimiento alcanza

la condición de parada de la recursión. En este momento, el requerimiento se transmite al procesador del pipeline de subida. Este recibe en un mensaje la información necesaria para continuar la evaluación. Para ello, cada $\mathbf{PS_i}$ recupera oportunamente el resultado intermedio almacenado por cada procesador de bajada en su memoria local, formando cada uno un paquete de comunicación que transmite secuencialmente a su sucesor en el pipeline de subida. Este proceso continúa paso a paso hasta la evaluación completa del requerimiento, que ocurre en $\mathbf{PS_0}$.

Este procesamiento en Pipeline permite que los procesadores de la MPR queden libres al evaluar un paso recursivo del requerimiento y transmitir el paquete resultante a otros procesadores vecinos. En consecuencia, se puede explotar este comportamiento de la máquina para evaluar muchos requerimientos concurrente y solapadamente. Esto conlleva a un paralelismo en pipeline al nivel de requerimientos. También, se aprovecha la presencia o llegada de un requerimiento para activar las operaciones en los procesadores (paralelismo por flujo de datos).

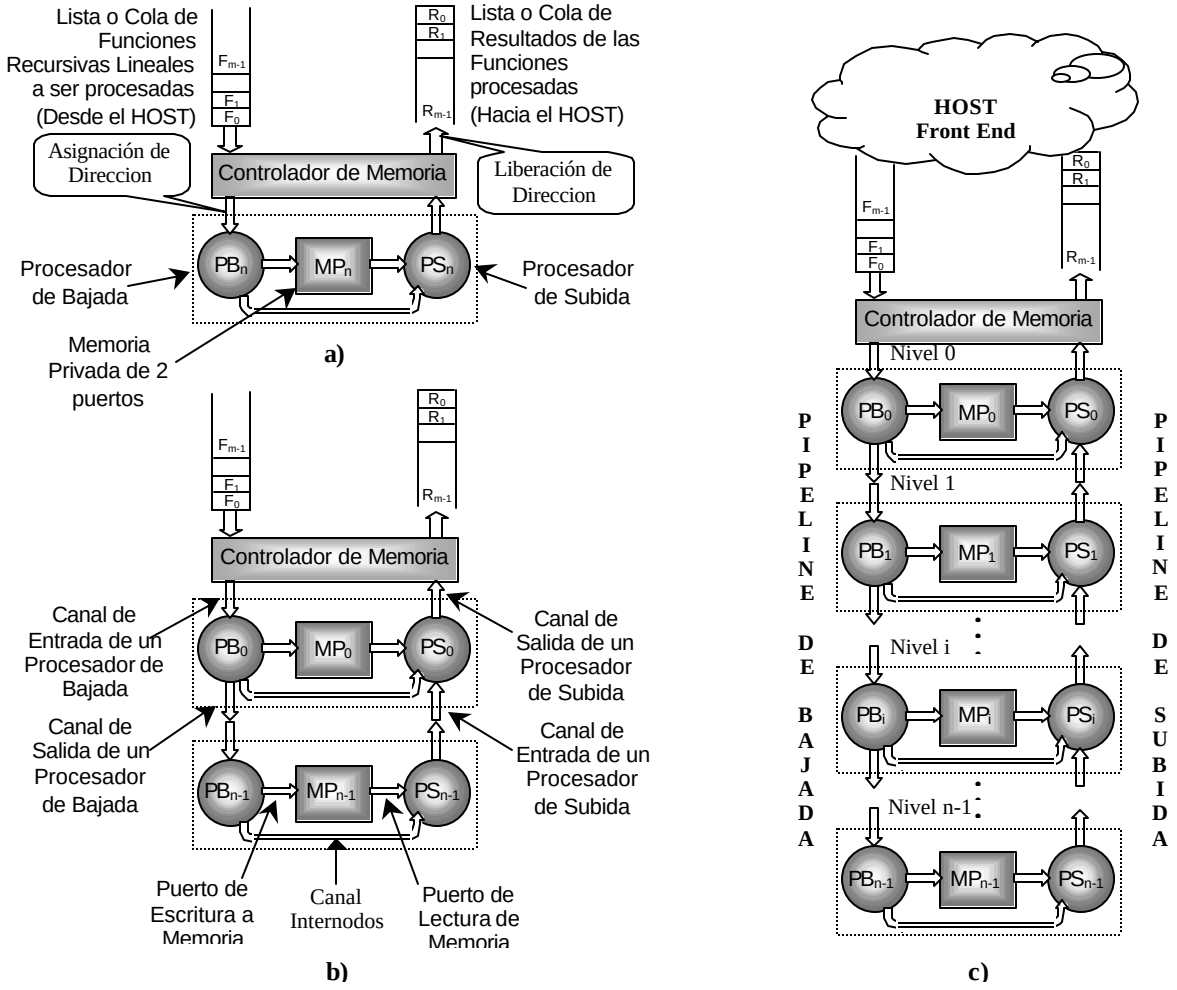

Figura 1. a) Organización basica de un nivel del Modelo Conceptual de la MPR con 2 Procesadores. **b)** MPR de 2 niveles. **c)** Vision multinivel con N procesadores donde se observa la organizacion recursiva de la MPR..

4. Procesamiento de Funciones Recursivas Lineales usando la MPR.

Caso 1: Evaluación de una Función Factorial (ver Algoritmo I, Figura 3 y Tabla A). A continuación se describe la evaluación de una función Factorial, adecuándola al modelo de la **MPR** para 4 niveles. Se utiliza un paquete de comunicación que contiene **<función, argumento, dirección>**. Los procesadores de bajada utilizan como operación aritmética la resta y los procesadores de subida la multiplicación. Así, el proceso de evaluar varias funciones factoriales con distintos argumentos y que no superen el tamaño de la **MPR** se pueden realizar de forma solapada en pipeline, aprovechando el paralelismo y la recursividad que provee la máquina.

```
Función Factorial(X : entero): entero
  comienzo
   si X < 2 entonces
       Factorial(X) : = 1
    sino
      Factorial(X) : = X * Factorial(X-1)
    fsi
  fin
Fin Función Factorial
```

Algoritmo I. Algoritmo pseudoformal de la Funcion Factorial.

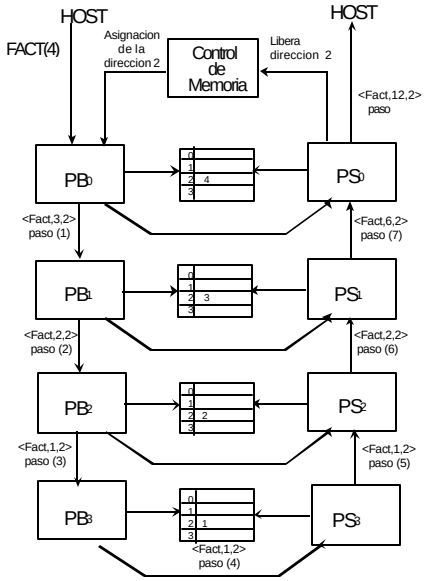

Figura 3. MPR de 4 niveles evaluando una Función Factorial.

Paso 0: El procesador **PB0** recibe el argumento X (en este caso con valor de 4, y lo almacena en la dirección asignada (dirección 2). Decrementa el argumento X y lo pasa a **PB1** e n un paquete de comunicación, junto con el valor de la dirección asignada.

Pasos del 1 y 2: Los procesadores **PB1 y PB2** reciben secuencialmente en pipeline el paquete de comunicación, y almacenan el argumento en la dirección de memoria indicada. Decrementan el argumento y lo transmiten al procesador siguiente, junto con la dirección asignada.

Paso 3: El procesador **PB3**, decrementa el argumento y se detecta la condición de parada para la función (en este caso X=1).

Paso 4: PB3 construye un paquete con este valor y lo pasa a **PS3**.

Pasos 5 al 7: Los procesadores **PS3, PS2 y PS1,** secuencialmente en pipeline toman los datos almacenados en la memoria por lo procesadores de bajada, en la dirección transmitida en el paquete. Realizan la multiplicación de este dato con el argumento recibido y transmiten el resultado al procesador siguiente.

Paso 8: PS0 toma el dato almacenado en la dirección indicada por el paquete, lo procesa y culmina la evaluación de la función factorial. A continuación despacha al Host la función evaluada y libera la dirección asignada.

Tabla A. Descripción de la evlaución de una Función Factorial.

5. Análisis del modelo de la MPR.

A continuación se procede a hacer un análisis de los componentes del modelo y evaluar su desempeño.

5.1. Modelo Determinístico de la MPR.

Estimación de la Capacidad del Sistema de Memoria. Teóricamente, el número de palabras de memoria en cada etapa de la máquina depende de la profundidad de las funciones recursivas a evaluar. El modelo de memoria de la MPR se basa en un control centralizado que simplifica su implantación, aunque requiere un espacio de direcciones más grande para cada $nivel_i$ (i = 2, ..., n), el cual será igual al

utilizado por el $nivel_1$ (2*n). Si **P** es el número de niveles o profundidad de la MPR y **m** el número de palabras de memoria por nivel, se tiene: $\mathbf{M = 2*n^2}$

Estimación del Tiempo de Procesamiento de una Lista o Carga de Trabajo*.** La MPR aprovecha el paralelismo de grano fino mediante el procesamiento de cargas de trabajo cuya simplicidad estriba en evaluar requerimientos que implican cadenas o listas de funciones recursivas lineales. Así, el procesamiento de una Lista de funciones recursivas de tamaño ***m genera una cadena de ***m*** requerimientos. Entonces, una aplicación que utilice la MPR, debe descomponerse previamente en un conjunto de requerimientos independientes, donde cada uno de dichos requerimientos debe expresarse como una función linealmente recursiva. La solución completa al problema, estará conformada por una lista con cada uno de los resultado de estas funciones. Este conjunto de requerimientos formará la carga de trabajo de la MPR. Por lo tanto, para determinar el tiempo de ejecución T_{MPR}, se asumen las siguientes proposiciones: **a.** Se tiene un conjunto de **k** funciones linealmente recursivas o requerimientos $\mathbf{R_i}$ (i = 1, ..., k) que esperan por entrar a la máquina y que forman una lista o cola de trabajo. **b.** Cada requerimiento R_i se caracteriza por el parámetro $\mathbf{a_i}$ el cual representa el argumento de la función. Este argumento siempre es mayor que cero. **c.** Las funciones a procesar en la lista o cola de trabajo son iguales u homogéneas, de tal forma que el tiempo de procesamiento en cada procesador $\mathbf{PB_i}$ y $\mathbf{PS_i}$ (i = 1, ..., n) es $\mathbf{t_b}$ y $\mathbf{t_s}$ respectivamente, para todas las funciones.

Durante el procesamiento paralelo de la lista de requerimientos, se pueden solapar los tiempos de ejecución de los diferentes requerimientos. Se asume $t_b = t_s =$ **t**, lo cual fija el intervalo de entrada de cada R_i a la MPR. El procesamiento es tal, que puede entrar un requerimiento a la máquina en cada ciclo de tiempo **t**. La carga de trabajo puede estar constituida por una lista con una gran cantidad de argumentos asociados cada uno con la evaluación de una función Factorial. En cada $\mathbf{PB_i}$ (los valores locales asociados y acumulados para cada R_i en cada nivel) se utiliza como operador elemental de "bajada" la resta y para los $\mathbf{PS_i}$ (los valores locales y los resultados parciales) se utiliza como operador elemental de "subida" la multiplicación.

Como se puede observar en la Figura 3, existe un requerimiento R_j, que "divide" la carga de trabajo de tal forma que el tiempo antes de entrar R_j es $(j-1)t_b$ y el tiempo después de entrar R_j es $a_j(t_b+t_s)$. El tiempo de ejecución en la MPR, T_{MPR}, para una carga de trabajo es entonces: $T_{MPR} = (j-1)t_b + a_j(t_b+t_s)$.

Es importante notar que es conveniente que tb sea aproximadamente igual a ts, para que se mantenga un equilibrio en el sistema. Si tb > ts, los procesadores **PSi** estarán mucho tiempo ociosos, en espera de un requerimiento. Si tb < ts, se pueden formar colas a la entrada de los **PSi.**

Una vez obtenida la expresión que determina el tiempo de ejecución paralelo en la MPR (T_{MPR}), se realizó un análisis de este valor dependiendo de las características de las cargas de trabajo.

Cargas de Trabajo Experimentales usadas para la Evaluación del Desempeño.

Caso Regular: Todos los a_i (con i = 1, ..., k) son iguales a **n**, donde **n** representa el número de niveles de la máquina (niveles recursivos), por lo cual: $\mathbf{T_{MPR} = (k-1)t_b + n(t_b+t_s)}$

Caso no Regular: Los a_i (con i = 1, ..., k) son menores o iguales a **n**, donde **n** es el número de niveles recursivos de la máquina, por lo tanto: $\mathbf{T_{MPR} = (j-1)t_b + a_j(t_b+t_s)}$

Sin embargo, es importante notar que en este caso, ya que las funciones podrían no alcanzar la última etapa, se pueden presentar colisiones en los **PSi** debido a que ellos pueden recibir datos por dos canales de comunicación diferentes. Las colisiones que pueden presentarse en estos procesadores, podría determinarse examinando la carga de trabajo. Por ejemplo, para n = 4, se tiene:

Ubicación de la Colisión	Condición de Colisión
En PS_4 no hay colisiones.	--------
En PS_3 pueden ocurrir colisiones si:	$(a_i = 4)$ y $(a_{i+2} = 3)$, i = 1 ... (k-2)
En PS_2 pueden ocurrir si:	$(a_i = 3)$ y $(a_{i+2} = 2)$, i = 1. .. (k-2) ó $(a_i = 4)$ y $(a_{i+4} = 2)$, i = 1 ... (k-4)
En PS_1 pueden ocurrir si:	$(a_i = 2)$ y $(a_{i+2} = 1)$, i = 1 ... (k-2) ó $(a_i = 3)$ y $(a_{i+4} = 1)$, i = 1 ... (k-4) ó $(a_i = 4)$ y $(a_{i+6} = 1)$, i = 1 ... (k-6)

Se pueden generalizar las condiciones bajo las cuales pueden presentarse colisiones en un PS_m, m = 1..(n-1), de la siguiente forma:

Ubicación de la Colisión	Condición de Colisión
En un PS_m, m = 1..(n-1) pueden ocurrir colisiones si:	$\bigvee_{j=(n-1)}^{m} \left[\bigvee_{I=1}^{k-2(n-j)} (a_I = n-j+m) \text{ y } a_{I+2(n-j)} = m) \right]$, donde $\bigvee$ = Suma lógica

El efecto de estas colisiones en T_{MPR} no se puede establecer de forma determinística, ya que es muy dependiente de los valores de los argumentos de las funciones y de como se encuentren éstas ordenadas en la cola que conforma la carga de trabajo. La colisión puede presentarse y sin embargo, no afectar a R_j, También puede ocurrir que la ocurrencia de una colisión en algún **PS_i,** pueda generar otra colisión en otro procesador. En todo caso, una vez determinada R_j, el efecto de las colisiones sobre este requerimiento es retrasar su salida de la MPR, lo cual en cualquier caso no puede exceder de $(k-j)t_b$.

5.2. Estudio del Desempeño de la MPR.

Para determinar la eficiencia de la MPR, se utilizó como medida la aceleración. Para encontrar la expresión de este parámetro, es necesario calcular el tiempo secuencial para una carga de trabajo dada. En este caso, el tiempo de una tarea está determinado por el tiempo que demora en ejecutarse más el tiempo que debe esperar para su ejecución, es decir: **Tiempo del requerimiento 1 = $S_1 = a_1(t_b + t_s)$, Tiempo del requerimiento 2 = $S_2 = S_1 + a_2(t_b + t_s)$, ... , Tiempo del requerimiento k = $S_k = S_{k-1} + a_k(t_b + t_s)$**

donde k indica el número total de requerimientos. Como el tiempo total se mide desde el momento en que se empieza a procesar el primer requerimiento hasta que termina el procesamiento del último requerimiento, se tiene que:

$$T_{SEQ} = S_k = a_k(t_b+t_s) + a_{k-1}(t_b+ts) + ...+ a_1(t_b+t_s) = T_{SEQ} = (t_b+t_s) \sum_{I=1}^{k} a_I$$

Para el análisis de la Aceleración se plantearon dos casos de carga de trabajo: regular y no regular.

Caso Regular. No se presentan colisiones en los PS_i , y

$$Ac = \frac{(t_b + t_s)\sum_{I=1}^{k} a_I}{(k-1)t_b + a_k(t_b + t_s)} = \frac{(t_b + t_s)kn}{(k-1)t_b + n(t_b + t_s)}$$

Se puede demostrar que esta expresión es mayor o igual a 1 de la siguiente forma:

$$\frac{kn(t_b + t_s)}{(k-1)t_b + n(t_b + t_s)} = \frac{knt_b + knt_s}{(k-1)t_b + n(t_b + t_s)} = \frac{knt_b + t_b - t_b + knt_s}{(k-1)t_b + n(t_b + t_s)} = \frac{(\overbrace{kt_b + kt_b + \ldots + kt_b}^{n \text{ veces}}) + knt_s + t_b - t_b}{(k-1)t_b + n(t_b + t_s)}$$

$$= \frac{(k-1)t_b + k(n-1)t_b + knt_s + t_b}{(k-1)t_b + n(t_b + t_s)} = \frac{(k-1)t_b + knt_b - kt_b + knt_s + t_b}{(k-1)t_b + n(t_b + t_s)} = \frac{(k-1)t_b + kn(t_b + t_s) - kt_b + t_b}{(k-1)t_b + n(t_b + t_s)}$$

$$= \frac{(k-1)t_b + kn(t_b + t_s) - kt_b + t_b}{(k-1)t_b + n(t_b + t_s)} = \frac{(k-1)t_b + (\overbrace{n(t_b + t_s) + n(t_b + t_s) + \ldots + n(t_b + t_s)}^{k \text{ veces}}) - kt_b + t_b}{(k-1)t_b + n(t_b + t_s)}$$

$$= \frac{(k-1)t_b + n(t_b + t_s)}{(k-1)t_b + n(t_b + t_s)} + \frac{n(k-1)(t_b + t_s) - kt_b + t_b}{(k-1)t_b + n(t_b + t_s)} = 1 + \frac{n(k-1)(t_b + t_s) - kt_b + t_b}{(k-1)t_b + n(t_b + t_s)}$$

En este caso, la Aceleración es igual a 1 cuando se tiene un solo requerimiento como carga de trabajo. Por otra parte, es posible establecer una cota superior para la Aceleración, cuando la lista crece.

Si tb = ts: $\lim_{k\to\infty} Ac = \lim_{k\to\infty} \frac{2kn}{(k-1) + 2n} = \frac{2n}{1} = 2n$

Si $t_b > t_s$: $\lim_{k\to\infty} Ac = \lim_{k\to\infty} \frac{nk(t_b + t_s)}{(k-1)t_b + n(t_b + t_s)} = \frac{n(t_b + t_s)}{t_b} = n + \frac{nt_s}{t_b}$

Si tb < ts: $\lim_{k\to\infty} Ac = \lim_{k\to\infty} \frac{nk(t_b + t_s)}{(k-1)t_s + n(t_b + t_s)} = \frac{n(t_b + t_s)}{t_s} = n + \frac{nt_b}{t_s}$

Se puede observar, que la Aceleración está acotada por dos veces el nivel recursivo, que es el número total de procesadores de la MPR.

Caso No Regular. La Aceleración en este caso, puede estudiarse tomando en cuenta el momento en que entra T_j a la MPR. Se tienen entonces dos políticas de despacho para la carga de trabajo:

- *Carga ordenada en forma ascendente*, donde $a_i \leq a_{i+1}$, i=1..(k-1). T_j es la última en entrar, no hay colisiones en los PS_i, y:

$$T_{EPR} = (k-1)t_b + a_k(t_b+t_s), \quad \text{por lo tanto} \quad Ac = \frac{a_1(t_b+t_s) + a_2(t_b+t_s) + \ldots + a_k(t_b+t_s)}{(k-1)t_b + a_k(t_b+t_s)} =$$

$$Ac = \frac{(k-1)t_b + a_k(t_b + t_s) + a_1(t_b + t_s) + \ldots + a_{k-1}(t_b + t_s) - (k-1)t_b}{(k-1)t_b + a_k(t_b + t_s)} =$$

$$Ac = \frac{(k-1)t_b + a_k(t_b + t_s)}{(k-1)t_b + a_k(t_b + t_s)} + \frac{a_1(t_b + t_s) + \ldots + a_{k-1}(t_b + t_s) - (k-1)t_b}{(k-1)t_b + a_k(t_b + t_s)} = 1 + \frac{a_1(t_b + t_s) + \ldots + a_{k-1}(t_b + t_s) - (k-1)t_b}{(k-1)t_b + a_k(t_b + t_s)}$$

Se puede observar que solo cuando la carga de trabajo está compuesta por un único requerimiento (k=1), la Aceleración es igual a 1.

- *Carga ordenada en forma descendente*:
  a. Si descendente estricto, donde $a_i > a_{i+1}$, i=1..(k-1), no hay colisiones, entonces:

$$T_{EPR} = a_1(t_b + t_s), \quad \text{por lo cual} \quad Ac = \frac{a_1(t_b+t_s) + a_2(t_b+t_s) + \ldots + a_k(t_b+t_s)}{a_1(t_b+t_s)}$$

$$= \frac{a_1(t_b + t_s)}{a_1(t_b + t_s)} + \frac{a_2(t_b + t_s) + \ldots + a_k(t_b + t_s)}{a_1(t_b + t_s)} = 1 + \frac{a_2(t_b + t_s) + \ldots + a_k(t_b + t_s)}{a_1(t_b + t_s)}$$

  b. *Si descendente no estricto*, donde $a_i \geq a_{i+1}$, i=1..(k-1), pueden presentarse colisiones y se tiene:

  $T_{EPR} = (j-1)t_b + a_j(t_b+t_s)$, entonces,

$$Ac = \frac{a_1(t_b+t_s) + \quad + a_j(t_b + t_s) + \ldots + a_k(t_b+t_s)}{(j-1)t_b + a_j(t_b+t_s)} = \frac{(j-1)t_b + a_j(t_b + t_s) + a_1(t_b + t_s) + \ldots + a_k(t_b + t_s) - (j-1)t_b}{(j-1)t_b + a_j(t_b + t_s)}$$

$$\frac{(j-1)t_b + a_j(t_b+t_s)}{(j-1)t_b + a_j(t_b+t_s)} + \frac{(t_b+t_b)(a_1 + \ldots + a_{j-1} + a_{j+1} + \ldots + a_k) - (j-1)t_b}{(j-1)t_b + a_j(t_b + t_s)} = 1 + \frac{(t_b+t_s)(a_1+\ldots+a_{j-1} + a_{j+1}+\ldots+ a_k)-(j-1)t_b}{(j-1)t_b + a_j(t_b + t_s)}$$

Se observa que solamente cuando la carga de trabajo está compuesta por una tarea, la aceleración es igual a 1.

6. Conclusiones.

En este trabajo se desarrolló y usó una Máquina Paralela y Recursiva basada en los conceptos de Paralelismo (Pipeline, Concurrencia, Flujo de Datos) y Recursividad (en el modo de funcionamiento y organización de la arquitectura), al cual nos referimos como el Modelo Conceptual de la Máquina Paralelo-Recursiva -**MPR**. Este modelo puede utilizarse en aquellos casos donde se tiene un conjunto de procesos que pueden expresarse en forma recursiva y que cooperan para la solución de un problema dado. Es un modelo escalable y adaptable. Su conexión con el Host es sencilla. Gracias a su estructura regular, pueden utilizarse técnicas de diseño VLSI usando FPGA para su construcción. Se obtiene el mejor rendimiento cuando se procesan grandes listas ordenadas de funciones recursivas, dado que se disminuye el efecto de las colisiones en el cauce de subida de la máquina.

7. Referencias.

[1] Allen, Arnold."Probality, Statics and Queueing Theory with Computer Science Applications". Academic Press. 1978.

[2] Amimaya, Makoto; Takesue, Masaru; Hasegawa, Ryuso y Mikami, Hirohide. "Implementation and Evaluation of a List-Processing-Oriented Data Flow Machine". Proc. 13th. Annual International Symposion Computer Architecture. IEEE. 1986. pp. 10-18.

[3] Nuñez, Haydemar. " Evaluador Paralelo de Funciones Recursivas Lineales: EPR.", Trabajo de Grado. Msc. Computación U.C.V. Postgrado de Computación. 1994.

[4] Montagne, Euripides.; Rukoz M. y Surós, R. "A Recursive Approach to Parallel Computation". Proceedings of the 13th International Symposium of Computer and Information Sciences. p.p. 26-28. Belek-Antalya, Turkey, IOS Press. Octubre de 1.998,

[5] Wedler, Chistoph y Lengauer, Christian. "On Linear List Recursion in Parallel". Acta Informatica, Volume 35. Springer-Verlag. 1998. pp. 875-909.

[6] Darlington, John; Reeve, Mike y Wright, Sue. "Declarative Languages and Program Transformation for Programming Parallel Systems: a case study". Concurrency: Practice and Experience. Vol 2. #3. 1990.

[7]Decegama, Angel. "The Technology of Parallel Processing. Vol I: Parallel Processing Architectures and VLSI Hardware". Prentice Hall. 1989.

[8] Denning, Peter; Dennis Jack y Qualitz, Joseph. "Machines, Languages and Computation". Prentice Hall. 1978.

[9] Dennis, Jack. "The Varieties of Data Flow Computers". MIT Reporte Interno. Agosto 1979.

[10]Eager, Alan. "Speedup versus Efficiency in Parallel Systems" IEEE trans. on Computer. Vol 38. #3. 1989.

[11] Glushkov; V.M, Ignatyev, M.; Myasnikov, V. y Torgashev, V. "Recursives Machines and Computing Technology". Proc. of the IFIP Congress. 1974.

[12] Gurd, J.R.; Kirham, C.C. y Watson, I. "The Manchester Prototype Data Flow Computer". Comm. of the ACM. Vol 28. #1. 1985. pp 34-52.

[13] Hwang, Kai y Briggs, Fayre. "Computer Architecture and Parallel Processing". Mc Graw Hill. 1988.

[14] Hudak, Paul. "Conception, Evolution and Application of Functional Programming Languages". ACM Computing Sutveys. Vol. 21. No. 3. 1989.

[15] Kogge, Peter. "The Architecture of Pipelined Computers". Hemisphere Publishing Corporation. 1981.

[16] Kung, S.Y; Lo, S.; Jean, S. y Hwang, J. "Wavefront Array Processors - Concept to Implementation". IEEE Computer. Julio 1987. pp 18-33

[17] Luis, Jorge y Kirner, Claudio. "Development of the Basic Software of Tagged-Token Data Flow Machine". IX Conferencia Internacional de la Sociedad de la Ciencia de la Computación (Panel 89 - XV Conferencia Latinoamericana de Informática). Computación Aplicada. Vol II. Santiago de Chile. Julio 1989. pp.217-228.