# CAPITULO 5

# ARQUITECTURAS SIMD

## 5. – Generalidades [36]

En una arquitectura SIMD el paralelismo se logra por múltiples unidades de proceso llamadas Elementos de Procesamiento (PEs), cada una de las cuales es capaz de ejecutar una operación especializada autónomamente.

La arquitectura de los Array Processor y de los Procesadores Vectoriales SIMD se caracteriza por el hecho de que la misma operación es realizada en un momento dado sobre un gran conjunto de datos en todos los PEs.

Las computadoras SIMD están especialmente diseñadas para realizar cómputos vectoriales sobre matrices o arrays de datos.

Pueden utilizarse indistintamente los términos Procesadores Array, Procesadores Paralelos, y computadoras SIMD.

## 5.1 - Array Processor. [S.O.]

Un array sincrónico de procesadores paralelos se denomina un Procesador Array, el cual consiste de múltiples Elementos de Procesamiento (PEs) bajo la supervisión de una unidad de control (CU).

Un procesador array puede manejar un único flujo de instrucción y múltiples flujos de datos. En tal sentido los procesadores array son también conocidos como máquinas SIMD (Fig. 5.1).

Las procesadores Array existen en dos organizaciones arquitecturales básicas :

- Los procesadores array que utilizan memoria de acceso aleatorio
- Los procesadores asociativos que usan memorias direccionables por contenido (o memorias asociativas).

Trataremos los procesadores array y brevemente los de memoria asociativa.

Dado que un PE no constituye una unidad de proceso central completa (CPU) y que no es capaz de funcionar independientemente, el sistema es en este caso de procesamiento paralelo y no un sistema multiprocesador.

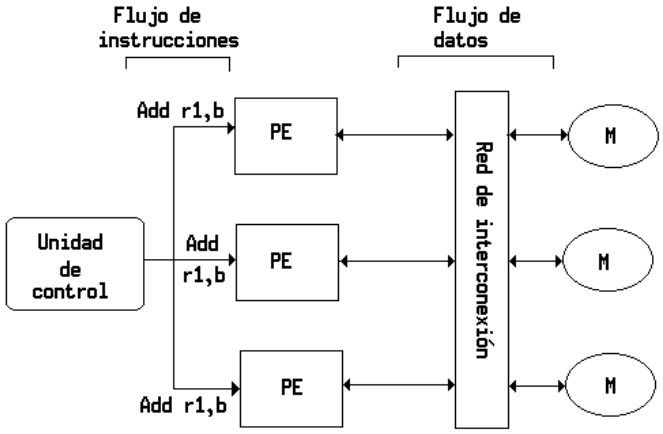



Fig. 5.1. - Ejecución SIMD.

## 5.2 - Organización de las computadoras SIMD [2]

En general, un procesador array puede tener una de dos configuraciones ligeramente diferentes.

Una ha sido implementada en la muy conocida computadora ILLIAC IV cuya esquematización puede verse en la Fig. 5.2. Esta computadora posee 256 PEs distribuidos en cuatro cuadrantes de 8 x 8 PEs (Fig. 5.3).

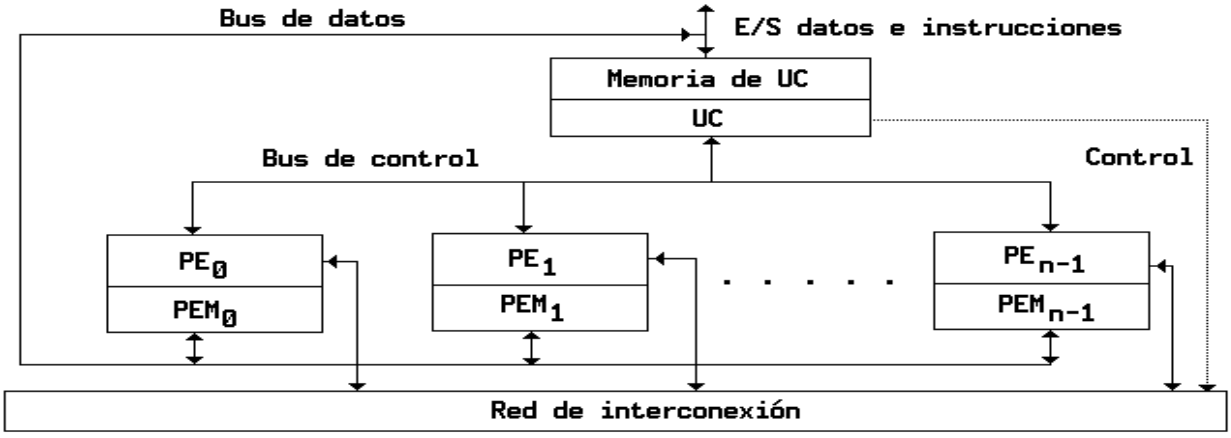



Fig. 5.2. - Un tipo de configuración de un procesador Array (ILLIAC IV).

Los PEs de un cuadrante son controlados por una unidad de control común, como resultado de esto los PEs de dicho cuadrante pueden ejecutar la misma operación simultáneamente.

La Fig. 5.3 muestra la comunicación entre los PEs dentro del cuadrante. Cada PE se comunica con su vecino, la mayor distancia entre dos PEs no vecinos es 8.

La esquematización de la ILLIAC IV de la Fig. 5.2 está estructurada con N PEs sincronizados, los que se encuentran bajo el control de una CU.

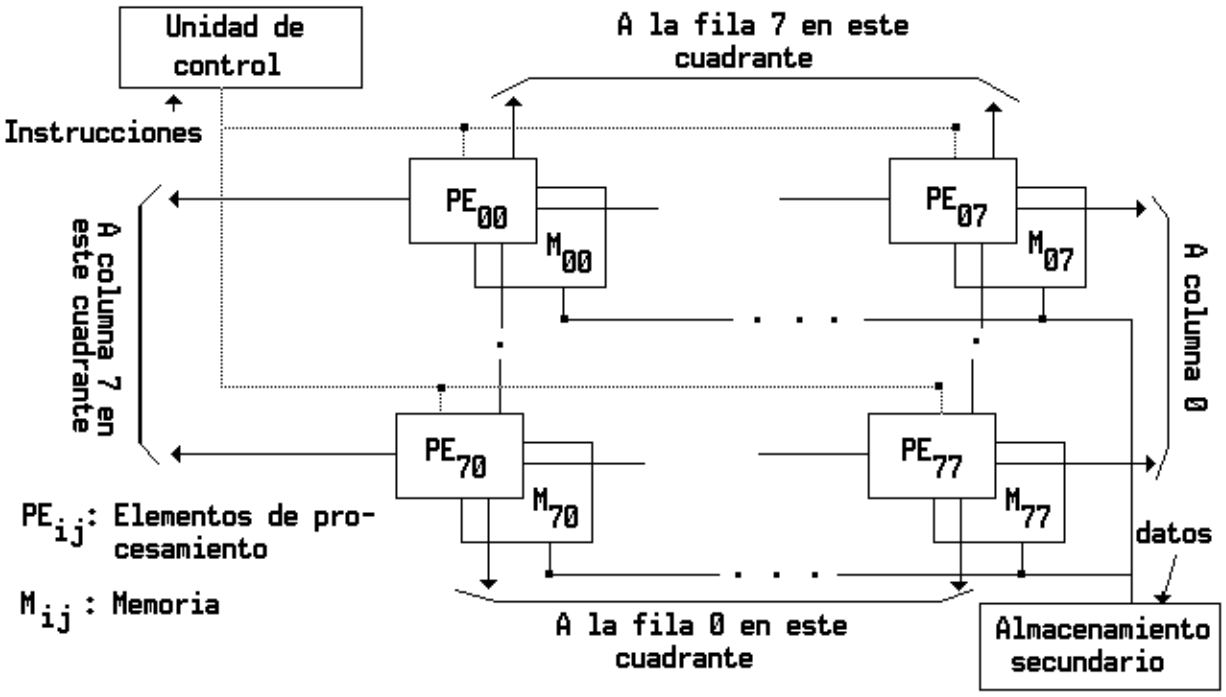



Fig. 5.3. - Cuadrante de 8 * 8 PEs de la ILLIAC IV.

Cada PE es esencialmente una unidad aritmético y lógica (ALU) con registros de trabajo que le pertenecen y una memoria local (PEM) para el almacenamiento de los datos distribuidos.

La CU cuenta también con una memoria propia para el almacenamiento de programas. En ella se cargan los programas del usuario desde el almacenamiento principal.

La función de la CU es decodificar todas las instrucciones y determinar dónde deben ejecutarse las instrucciones decodificadas.

Las instrucciones de control de tipos o las escalares se ejecutan directamente dentro de la CU. Las instrucciones vectoriales se envían a los PEs para una distribución de la ejecución a fin de alcanzar un paralelismo espacial gracias a la multiplicación de unidades aritméticas (los PEs).

Todos los PEs realizan la misma función sincrónicamente en forma esclava bajo el control de la CU.

Los operandos vectoriales se distribuyen en las PEM antes de lanzar la ejecución paralela de los PEs.

Estos datos distribuidos pueden cargarse en las PEM desde una fuente externa a través del bus de datos del sistema, o desde la CU en una modalidad que utiliza el bus de control.

La unidad de control emite y distribuye máscaras sobre los PEs para permitir la realización o no de ciertas operaciones, para ello lleva cuenta del estado de los PEs utilizando bits de estado asociados a los mismos.

Un array processor facilita la programación brindando operaciones sobre vectores y escalares; es posible ejecutar una operación de instrucción sobre un vector completo, con las únicas restricciones del tamaño de memoria y del número de PEs disponibles.

Se utilizan esquemas de enmascaramiento para controlar el estado de cada PE durante la ejecución de la instrucción vectorial.

Cada PE puede estar en estado activo o en estado inhibido durante el ciclo de la instrucción indicado por un vector de máscaras.

En otras palabras, no es necesario que todos los PEs participen en la ejecución de una instrucción vectorial, sólo los PEs habilitados realizan el cómputo.

Los intercambios de datos entre los PEs se realizan vía una red de comunicación inter-PE. Esta red de comunicación está bajo el control de la CU.

Un array processor tiene normalmente una interfase a un computador host desde su unidad de control. El computador host es una máquina de propósito general que sirve como el "administrador operativo" del sistema en su totalidad, formado por el host y el procesador array.

Las funciones del host consisten en la administración de los recursos y la supervisión de los periféricos y las entradas/salidas.

La unidad de control del procesador array supervisa directamente la ejecución de los programas, en tanto que el host realiza las funciones ejecutivas y de E/S con el exterior. En tal sentido un procesador array puede ser considerado como un computador conectado back-end (attached), similar en cuanto a función a los procesadores pipeline attachados.

Otra configuración posible para un procesador array se ilustra en la figura 5.4. Esta difiere de la de la Fig. 5.2 en dos aspectos.

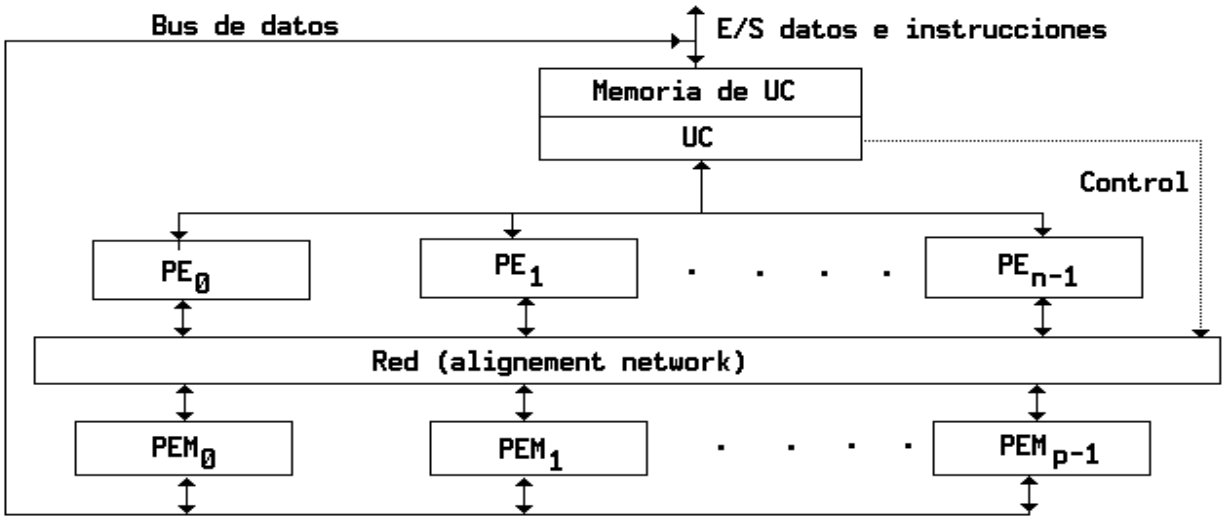



**Fig. 5.4. - Otro tipo de configuración de un procesador Array (computador BSP de Burroughs).**

Primero, las memorias locales pertenecientes a cada PE se reemplazan aquí por módulos paralelos de memoria compartidos por todos los PEs mediante una red (alignement network).

Segundo, la red de permutación de PEs se reemplaza por una red inter-PE de memoria, que se encuentra nuevamente bajo control de la CU.

Esta red (alignement) es una red de caminos conmutables entre los PEs y las memorias paralelas. Se espera que tal red sirva para liberar de los conflictos de acceso a las memorias compartidas por tantos PEs como sea posible. Un buen ejemplo de esta configuración es el Procesador Científico Burroughs (BSP).

En este tipo de configuración existen n PEs y p módulos de memoria. Ambos números no son necesariamente iguales.

## 5.3 - Estructura interna de un PE [S.O.]

En la Fig. 5.5 graficamos la estructura interna de un elemento de procesamiento :

$D_i$ indica la dirección asignada a este elemento de procesamiento.

$S_i$ indica si este procesador está o no activo para realizar un cierto cálculo.

$I_i$ es un registro índice.

$R_i$ es el registro que apunta a otros procesadores

Formalmente un procesador SIMD se caracteriza por el siguiente conjunto de parámetros :

C = { N, F, I, M }

siendo

N = el número de procesadores.

F = el conjunto de funciones de ruteo de datos previstos por la red de interconexión.

I = el conjunto de instrucciones para las operaciones escalares vectoriales, de ruteo de datos y manejo de la red.

y M = conjunto de máscaras que habilitan o deshabilitan a los procesadores.

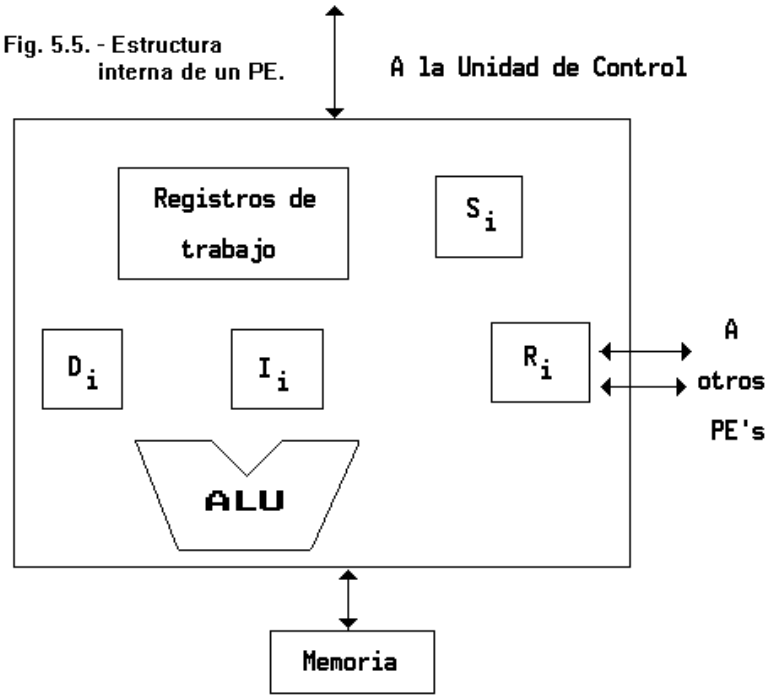



**Fig. 5.5. - Estructura interna de un PE.**

## 5.4 - Ejemplo de funcionamiento de un procesador array [2]

Para mostrar cómo es el ruteo de datos en un procesador array veremos en detalle una instrucción vectorial dentro de los N PEs.

Se desea la suma de S(k) de las primeras k componentes de un vector A para k = 0,1,....,n-1.

Sea A = (A0, A1, ...., An-1)

Necesitamos calcular las siguientes n sumas:

Este vector de n sumas puede calcularse recursivamente realizando las n-1 iteraciones definidas como :

S(0) = A(0)

S(k) = S(k-1) + A(k)  para k = 1,2,.....,n-1

Estas sumas recursivas para n = 8 se implementaron en un procesador array N = 8 PEs en una cantidad de pasos de $[\log_2 n]=3$.

En la implementación se utilizan tanto el ruteo de datos como el enmascaramiento de PEs.

Inicialmente cada A(i) que reside en cada PEM(i) es movido al registro D(i) en cada PE(i) para i = 0,1,....,n-1 (asumimos aquí n=N=8).

En el primer paso A(i) es ruteado desde R(i) a R(i+1) y sumado al A(i+1), obteniéndose la suma A(i) + A(i+1) en R(i+1) para i=0,1,....,6.

Las flechas en la Fig. 5.6 muestran las operaciones de ruteo y la notación abreviada $\Sigma A_i a A_j$ se utiliza indicando la suma intermedia de A(i) + A(i+1)+ ... +A(j).

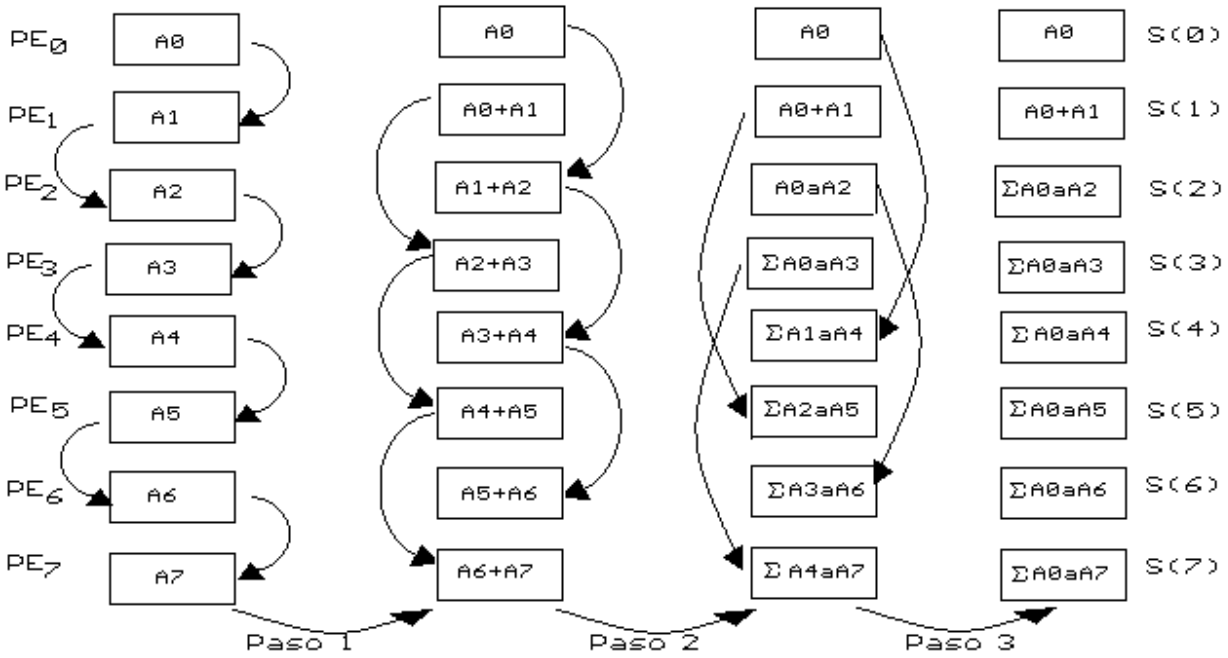



**Fig. 5.6. - Cálculo de la suma S[k] = $\Sigma$ A[k], para k = 0,...,7 en una máquina SIMD.**

En el paso 2, las sumas intermedias en R(i) son ruteadas a R(i+2) para i = 0,1,....,5.

En el paso final las sumas intermedias en R(i) son ruteadas a R(i+4) para i = 0,..,3

Consiguientemente, el PE(k) tiene el valor final de S(k) para k = 0,1,...,7.

Cuanto más lejos se extienden las operaciones de ruteo cada vez menos PEs son los intervinientes. Nótese que en el paso 1 PE(7) recibe pero no transmite; en el paso 2 son los PE(6) y PE(7); en tanto que en el paso 3 son los PE(4), PE(5), PE(6) y PE(7) los que reciben solamente.

Estos PEs que no se desea que transmitan son enmascarados durante el paso correspondiente.

Durante las operaciones de suma PE(0) permanece inhibido en el paso 1; PE(0) y PE(1) se vuelven inactivos en el paso 2; en tanto que en el paso 3 los PE(0), PE(1), PE(2) y PE(3) son inactivos.

Los PEs que son enmascarados en cada paso dependen de la operación (ruteo de datos o suma aritmética) que se realice.

Aún así los patrones de enmascaramiento varían en los diferentes ciclos de la operación.

Nótese que las operaciones de ruteo y enmascaramiento pueden ser mucho más complejas cuando la longitud del vector es n $f$ N.

Un array de PEs es una unidad aritmética pasiva esperando ser invocada para las tareas de cómputo paralelo.

## 5.5. - Procesadores Array de estructura Bit-Plane [32]

Una variante de las arquitecturas de procesadores array utiliza una gran cantidad de procesadores de un bit. En las arquitecturas **bit-plane** los procesadores array se colocan en una grilla (mesh) simétrica (por ej. de 64 * 64) y se asocian con múltiples planos de bits en memoria que se corresponden a las dimensiones de la grilla de procesadores (ver Fig. 5.7).

El procesador n (Pn) situado en la grilla en la posición (x,y) opera sobre los bits de memoria ubicados en la posición (x,y) en todos los planos de memoria asociados.

Usualmente se proveen además operaciones del tipo de copia, enmascarado y operaciones aritméticas que operan sobre todos los planos de memoria tanto sobre columnas como sobre filas.

El Procesador Masivo Paralelo de Loral y el Procesador Array Distribuido de ICL ejemplifican esta clase de arquitecturas utilizadas muy frecuentemente para aplicaciones de procesamiento de imágenes mapeando pixels desde la estructura planar de la memoria.

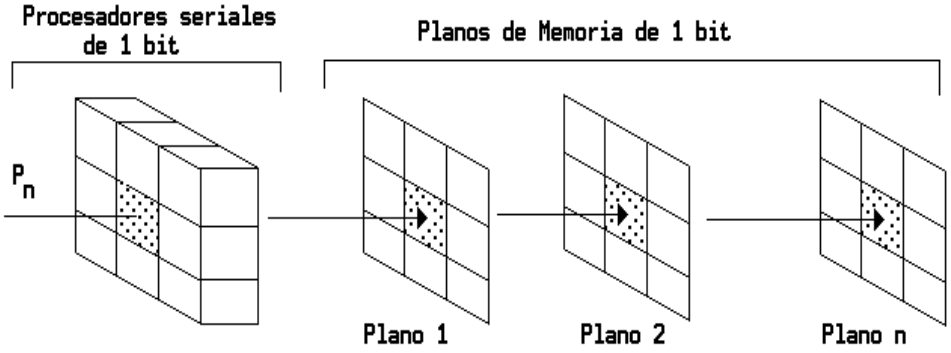



Fig. 5.7. - Procesador array de estructura bit-plane.

La Máquina de Conexión de Thinking Machine tiene organizada la memoria como 65.536 procesadores de un bit en conjuntos de a cuatro procesadores unidos en una topología de hipercubo.

## 5.6. - ARQUITECTURA DE PROCESADORES ARRAY DE MEMORIA ASOCIATIVA [32]

Las computadoras construidas sobre memorias asociativas constituyen un tipo muy distintivo de máquinas SIMD que utilizan una lógica de comparación para acceder en paralelo, de acuerdo al contenido, a los datos almacenados.

Las investigaciones sobre la construcción de memorias asociativas comenzaron en los últimos años de la década de los 50 con el objetivo obvio de lograr recorrer la memoria en paralelo buscando datos que coincidieran con una clave específica.

Los procesadores de memorias asociativas "modernos" desarrollados en los 70 (por ej. el Conjunto de Elementos de Procesamiento en Paralelo -PEPE- de los laboratorios Bell) y las arquitecturas más recientes (el procesador Asociativo de Loral, ASPRO) han sido construidos lógicamente orientados a aplicaciones de bases de datos tales como búsquedas y navegaciones.

La Fig. 5.8 muestra las características de las unidades funcionales de un procesador de memoria asociativa. Un controlador de programa (un computador serial) lee y ejecuta las instrucciones invocando a un controlador array especializado cuando se detectan instruccio-

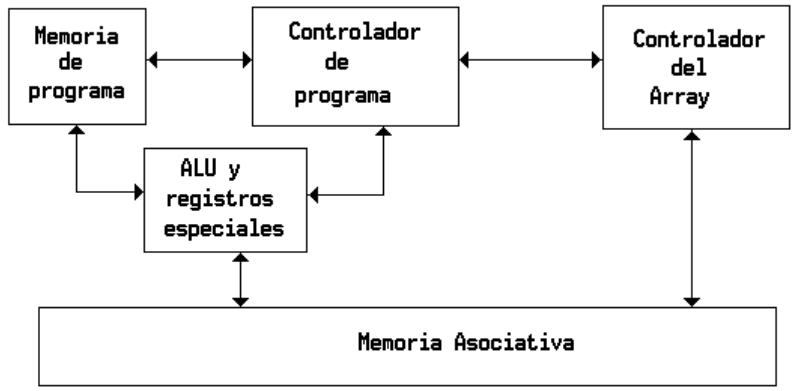
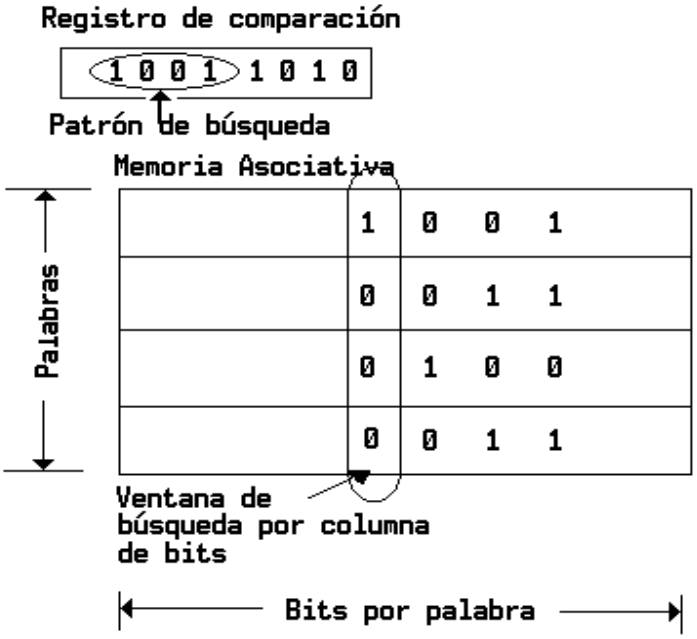



Fig. 5.8. - Organización de un procesador de memoria asociativa.



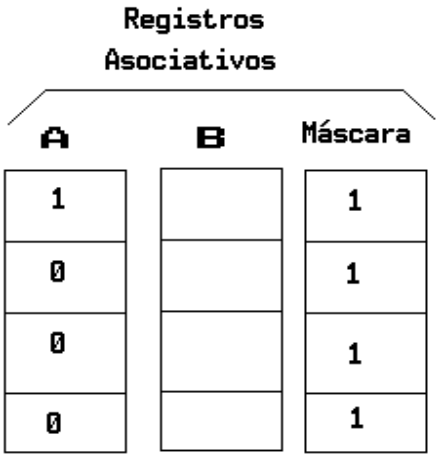




Fig. 5.9. - Comparación orientada a fila.

nes de memoria asociativa.

Registros especiales habilitan al controlador de programa y a la memoria asociativa para compartir datos.

La mayoría de los procesadores de memoria asociativa utilizan una organización serial de bits, la cual implica operaciones concurrentes sobre un solo bit-slice para todas las palabras en la memoria asociativa.

Cada palabra de memoria asociativa, que generalmente tiene una gran cantidad de bits (por ej. 32768) se asocia con registros especiales y una lógica de comparación que funcionalmente constituye un procesador. Por lo tanto, un procesador asociativo con 4096 palabras tiene efectivamente 4096 elementos de procesamiento.

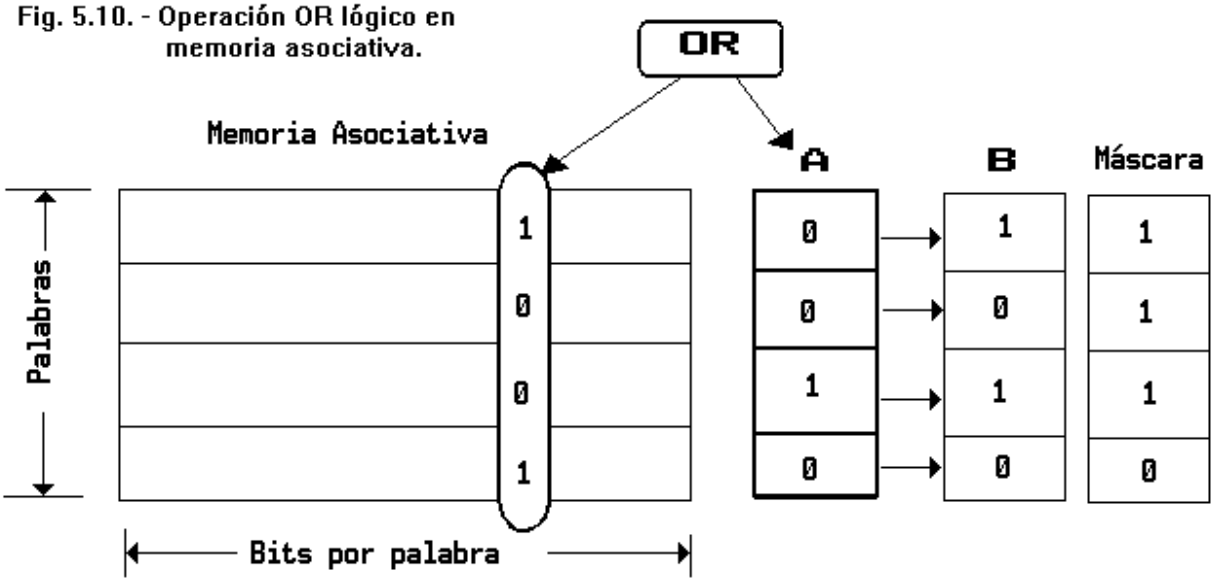
La Fig. 5.9 muestra una comparación orientada a fila para una arquitectura genérica de bit-serial. Una parte del registro de comparación contiene el valor que debe coincidir.

Todos los elementos de procesamiento (PE) comienzan en una columna específica de la memoria y comparan el contenido de cuatro bits consecutivos en la fila con el contenido del registro, encendiendo un bit en el registro A para indicar si la fila equipara o no.

En la Fig. 5.10 podemos ver una operación de un OR lógico realizada entre una columna de bits y el vector de bits en el registro A, siendo el registro B el que recibe el resultado.

Un cero en el registro de máscara indica que la palabra asociada no debe incluirse en esta operación.

Fig. 5.10. - Operación OR lógico en memoria asociativa.




## 5.7.- SYSTOLIC ARRAYS ( SISTÓLICOS) [36]

Los procesadores sistólicos son el resultado de los avances en tecnología de semiconductores y en las aplicaciones que requieren un amplio rendimiento.

### 5.7.1 - Introducción a los procesadores Sistólicos [36]

Fue en 1978 cuando H. T. Kung y C. E. Leiserson introdujeron el término "sistólico" y el concepto subyacente, para resolver problemas de sistemas de propósito específico que deben balancear el bandwidth entre una intensiva cantidad de cálculos y gran cantidad de requerimientos de E/S.

Los procesadores convencionales están muy a menudo limitados por la disparidad del bandwidth de Entrada y el bandwidth de Salida, el cual se produce debido a que los ítems de datos deben ser leídos y grabados cada vez que se los referencia.

Una razón para elegir el término "systolic" como parte de Systolic Array puede verse en la analogía con el sistema de circulación humano, en el cual el corazón entrega y recibe una gran cantidad de sangre como resultado del bombeo rítmico e ininterrumpido de pequeñas cantidades de ese fluido a través de venas y arterias. En esta analogía el corazón corresponde a la fuente y destino de los datos, como si fuera una memoria global; y la red de venas es equivalente al array de procesadores y sus conexiones.

Las arquitecturas Sistólicas (array sistólicos) son *multiprocesadores pipelinizados* en los cuales los datos se bombean en forma rítmica desde la memoria y a través de la red de procesadores antes de ser devueltos a la memoria (ver Fig. 5.11).

La información circula entre los procesadores como en un pipeline, pero sólo los procesadores frontera mantienen comunicación con el exterior.

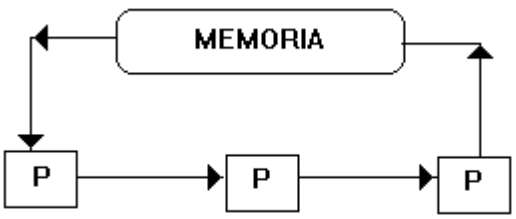



Fig. 5.11. - Flujo de datos desde y hacia la memoria.

Un reloj global conjuntamente con mecanismos explícitos de retardo *sincronizan* el flujo de datos a través del pipe que se conforma con los datos obtenidos de la memoria y los resultados parciales que usa cada procesador.

Los procesadores modulares unidos mediante una red local y regular proveen los ladrillos básicos para construir una buena variedad de sistemas de propósito específico.

Durante cada intervalo de tiempo estos procesadores ejecutan una secuencia corta e invariante de instrucciones.

El término "array" se origina en la similitud de los systolic array con una grilla o red en la cual cada punto corresponde a un procesador y cada hilo a una conexión entre los procesadores. Visto como esta estructura los sistólicos son descendientes de las arquitecturas del tipo array, tales como los arrays interactivos, los autómatas celulares y los procesadores array.

Si bien la estructura array caracteriza las interconexiones en los sistólicos, es el término "systolic" el que capta el comportamiento innovador y distintivo de estos sistemas.

"Systolic" en este contexto significa que los cálculos pipeline se realizan en todas las dimensiones del array y brindan como resultado un muy alto rendimiento computacional. Son sistemas de cómputos altamente especializados en los que se debe realizar operaciones concurrentes y que se caracterizan por la gran cantidad y diversidad de procesadores.

La tecnología apta para esto es la VLSI (Integración en muy alta escala). De todas maneras existe un límite tecnológico, inclusive el de la velocidad de la luz. Luego para conseguir altas velocidades de procesamiento se recurre al uso simultáneo de procesadores. O sea que los algoritmos a utilizar deben permitir un alto grado de pipelining y multiprocesamiento.

En otras palabras, los algoritmos sistólicos administran los cálculos de manera tal que un ítem de dato no se usa solamente cuando es input sino también es *reutilizado moviéndose a través del pipeline en el array.*

En estos casos se hacen críticas las redes de interconexión. La misma consiste en un conjunto de procesadores interconectados que realizan operaciones simples. La interconexión puede generar vectores, matrices, árboles, etc.

Esto resulta en un balance del ancho de banda (bandwidth) entre el procesamiento y la entrada/salida, especialmente en problemas de compute-bound que tienen más cálculos a realizar que entradas y salidas.

La idea (como se ve en la Fig. 5.11) es que desde la memoria se "bombea" la información como desde un corazón, y esta fluye de un Procesador al siguiente.

**Fig. 5.12. - Aplicaciones sobre procesadores Sistólicos.**

*** Procesamiento de señales/imágenes y reconocimiento de patrones**
**Filtros digitales**
**Figuras en dos dimensiones**
**Transformadas discretas de Fourier**
**Interpolación**
**Alabeo geométrico**
**Lineamientos de extracción**
**Estadísticas de orden**
**Clasificación de distancia-mínima**
**Cálculo de covarianza de matrices**
**Coincidencia de patrones (pattern matching)**
**Reconocimiento de patrones sintácticos**
**Procesamiento de señales de radar**
**Detección de curvas**
**Animación de figuras**
**Comparación de imágenes**
*** Aritmética de matrices**
**Multiplicación y triangulación de matrices**
**Descomposición QR**
**Operaciones sobre matrices dispersas**
**Solución de sistemas lineares triangulares**
*** Aplicaciones No-numéricas**
**Estructuras de datos - stacks y colas, ordenamiento**
**Algoritmos gráficos - clausura transitiva, minimización de árboles de máxima dimensión**
**Conexión de componentes**
**Reconocimiento del lenguaje**
**Programación dinámica**
**Operaciones sobre bases de datos relacionales**
**Aritmética sobre arrays**

Esto es adecuado para procesos con mucho tiempo de cálculo y donde varias operaciones se realizan en forma repetida sobre un mismo dato.

Este tipo de arquitecturas es la apropiada para resolver problemas de alta repetición y muy específicos, o sea se desarrollan para casos especiales y no se busca la resolución de problemas en general.

### 5.7.2. - Características de los Procesadores Sistólicos. [36]

La Fig. 5.12 es una lista de las aplicaciones disponibles sobre diseños sistólicos.

Es apropiado hablar brevemente sobre los tres factores que caracterizan a los sistólicos así como fueron originalmente propuestos, a saber: tecnología, procesamiento pipeline/paralelo, y aplicaciones.

Estos factores identifican también las razones para el éxito del concepto, especialmente eficacia de costos, alta performance, y la abundancia de aplicaciones para las cuales son utilizados.

#### 5.7.2.1 - Tecnología y eficacia de costos [36]

Hoy en día la madurez de la tecnología VLSI/WSI (Wafer Scale Integration) permite fabricar circuitos cuyas dimensiones mínimas rondan los 1 a 3 micrones. La exactitud en el campo de los procesos de fabricación VLSI/WSI hace posible la implementación de circuitos de hasta 1/2 millón de transistores a un costo razonable (incluso para pequeñas cantidades de producción).

Sin embargo las ventajas de esta tecnología no pueden utilizarse cabalmente a menos que se utilicen diseños simples, regulares y modulares. Los sistólicos tratan de cumplir estas restricciones topológicas mediante elementos de procesamiento sencillos, que, conjuntamente con un patrón de interconexión, también sencillo, se distribuyen en una o más dimensiones.

El costo, la regularidad, y la modularidad son factores que influyen en el diseño y la optimización de cada elemento de procesamiento individualmente y en el de sus respectivas interconexiones.

La consideración de estos tres factores indican que los procesadores arrays son una solución de un costo efectivo al problema de la construcción de sistemas con muchos elementos de procesamiento.

Debido a que los sistólicos tienen una gran cantidad de módulos iguales, salvo algunas pocas excepciones, el proceso de refinamiento sobre un gran sistema y el posterior diseño de cada subcomponente es más veloz y sencillo de lo que resulta en sistemas con una gran variedad de tipos de módulos.

#### 5.7.2.2 - Procesamiento pipeline/paralelo [36]

La eficacia en los cálculos de los procesadores sistólicos deriva del *multiprocesamiento y la técnica pipeline.* El multiprocesamiento es una consecuencia natural de las actividades que están viajando simultáneamente en distintos elementos de procesamiento de la red. El pipeline puede pensarse como una forma de multiprocesamiento que optimiza la utilización de recursos y saca ventaja de las dependencias entre los cálculos.

En los procesadores sistólicos, el pipeline de datos reduce el bandwidth de E/S permitiendo que un ítem de dato sea reutilizado una vez que ingresó a la red. Típicamente, los inputs entran en la red a través de elementos de procesamiento periféricos y se propagan a los elementos de procesamiento vecinos para ulteriores procesos.

Estos movimientos de datos a través de la red se producen en una *dirección fija* en la cual existe una conexión entre elementos de procesamientos vecinos y además se producen en forma periódica.

Además del pipeline de datos, los procesadores sistólicos se caracterizan también por un *pipeline de cómputo, en el cual la información fluye de un elemento de procesamiento hacia otro según un orden preestablecido.* Esta información puede ser interpretada por el receptor tanto como un dato, o información de control, o una combinación de ambos.

La ejecución procede de manera tal que el output generado por un elemento de procesamiento es utilizado como input de un elemento de procesamiento vecino.

Mientras que las operaciones se suceden como un flujo de datos a través de cada procesador, el cálculo completo no es una operatoria de flujo de datos (Dataflow), ya que las operaciones se ejecutan de acuerdo a un itinerario determinado por el diseño del array sistólico.

Luego de que un elemento de procesamiento genera un output intermedio y lo entrega al elemento de procesamiento vecino, continúa calculando otro output intermedio. Como resultado de esto los recursos de procesamiento se utilizan eficientemente.

Generalmente cada elemento de procesamiento puede estar construido como un procesador pipeline. Tal construcción recibe el nombre de Array Sistólico Pipelinizado de Dos Niveles y brinda un mayor troughput.

#### 5.7.2.3 - Aplicaciones y algoritmos [36]

Los algoritmos adecuados para las implementaciones de arrays sistólicos pueden hallarse en gran variedad de aplicaciones, tales como el procesamiento digital de señales e imágenes, el álgebra lineal, el reconoci-

miento de patrones, programación lineal y dinámica y problemas de grafos. De hecho, muchos de los algoritmos en la lista de aplicaciones contienen gran cantidad de cálculos y necesitan de las arquitecturas sistólicas para poder ser implementados cuando se utilizan en entornos de tiempo real.

### 5.7.3 - CUESTIONES DE IMPLEMENTACIÓN [36]

#### 5.7.3.1 - Sistemas sistólicos de propósito general y específico [36]

En forma típica, un array sistólico puede pensarse como un sistema algorítmicamente especializado en el sentido en que su diseño refleja necesidades de un algoritmo específico.

Sin embargo, sería deseable diseñar arrays sistólicos capaces de ejecutar eficientemente más de un algoritmo para una o más aplicaciones.

Existen dos propuestas posibles para el diseño de estos sistemas "multipropósito", y existe un compromiso entre las dos que se encuentra generalmente en muchas de las implementaciones actuales.

Una propuesta consiste en agregar mecanismos de hardware para poder reconfigurar la topología y el patrón de interconexiones del sistólico y poder así, emular los requerimientos de un diseño especializado.

Un ejemplo concreto de esto lo constituye la computadora Configurable de Alto Paralelismo (Configurable Highly Parallel, CHiP), que cuenta con una matriz de switches programable con el propósito de reconfiguraciones.

La otra propuesta utiliza software para mapear diferentes algoritmos dentro de una arquitectura fija de array. Esta propuesta necesita de la existencia de lenguajes de programación capaces de expresar cálculos en paralelo, así como del desarrollo de traductores, sistemas operativos y ayudas de programación.

Estos requerimientos son de aplicación, por ejemplo, en el sistema sistólico WARP desarrollado en la Universidad de Carnegie Mellon.

Para cada algoritmo, el diseñador necesita identificar el diseño sistólico eficiente y los mapas y las técnicas apropiadas para utilizar. La cuestión de las técnicas apropiadas es de gran importancia, ya que la performance final, el costo y la exactitud del diseño dependen de las mismas.

#### 5.7.3.2 - Técnicas de diseño y mapeo [36]

Para producir un array sistólico a partir de la descripción de un algoritmo, el diseñador necesita una comprensión total del mismo, y estar familiarizado con los principios subyacentes a cuatro cosas: el cálculo de tipo sistólico, la aplicación, el algoritmo y la tecnología.

Se han hecho progresos en el desarrollo de técnicas de diseño sistemáticas para automatizar este proceso.

Típicamente, las especificaciones incluyen el tamaño y la topología del array, las operaciones realizadas por cada elemento de procesamiento, el orden y la temporalidad de la comunicación de los datos, y los inputs y outputs.

De forma limitada, estas técnicas pueden tener en cuenta factores tecnológicos y la relación del array sistólico en sí mismo con el resto del sistema. Sin embargo, no son completas ya que solo pueden utilizarse en el nivel de especificaciones y sólo de una manera indirecta.

#### 5.7.3.3 - Granularidad [36]

La operación básica realizada en cada ciclo por cada elemento de procesamiento en los diferentes arrays sistólicos varía desde una simple operación de bit, hasta una multiplicación y suma a nivel de palabra, e incluso hasta la completa ejecución de un programa.

La elección del grado de refinamiento está determinado por la aplicación o por la tecnología, o por ambos.

Cuando se programan arrays sistólicos esta granularidad o grado de refinamiento puede también determinarse mediante compromisos entre el grado deseado y el nivel de programabilidad.

#### 5.7.3.4 - Extensibilidad [36]

Muchos arrays sistólicos especializados pueden verse como implementaciones hardware de un dado algoritmo.

En tal caso, el procesador sistólico puede ejecutar solamente un determinado algoritmo que está diseñado para un problema de un tamaño específico.

Si uno desea ejecutar el mismo algoritmo para un problema de gran tamaño, o bien debe construir un array mucho mayor o debe particionar el problema.

El primer caso es simple de conceptualizar y solo requiere que más elementos de procesamiento se utilicen para construir una versión extendida del array original.

Sin embargo, en cuanto a implementación se refiere, debemos recordar que pueden existir factores que no afecten la performance en arrays pequeños pero que sí pueden afectarla en sistemas de gran tamaño. Estos factores incluyen la sincronización del reloj, la confiabilidad, las necesidades de alimentación eléctrica, las limitaciones en el tamaño del chip, y las restricciones de los pin de E/S.

### 5.7.3.5 - Confiabilidad [36]

Se pueden utilizar sencillas leyes de la probabilidad para demostrar porqué el incrementar el tamaño del array resulta en la disminución de la confiabilidad a menos que se incorpore redundancia o se disponga de mecanismos tolerantes a fallas.

De hecho, la confiabilidad de un array de procesadores es igual a la confiabilidad de uno de los procesadores elevada a la cantidad de procesadores que contiene el array:

C (conjunto de m procesadores.) = $C^m$ (de 1 procesador)

Debido a que la confiabilidad de un procesador es un número menor a 1, la confiabilidad de todo el array se aproxima rápidamente a cero a medida que el número de procesadores aumenta.

Un buen diseño de estos sistemas tolerantes a fallas tiene como meta la maximización de la confiabilidad mientras que se minimiza el correspondiente overhead.

En los arrays sistólicos las soluciones posibles planteadas incluyen extensiones sencillas de las técnicas bien conocidas utilizadas en las computadoras digitales convencionales. Sin embargo estas técnicas no aprovechan las características propias de los arrays sistólicos ni de los algoritmos que ellos ejecutan.

### 5.7.3.6 - Particionamiento de grandes problemas [36]

Cuando es necesario ejecutar problemas de gran tamaños sin construir arrays sistólicos de gran tamaño, el problema, entonces, debe ser particionado de manera tal que el mismo algoritmo pueda utilizarse para resolver el problema más pequeño, así un array chico y de tamaño fijo puede resolverlo.

Una técnica consiste en identificar particiones del algoritmo y un orden de ejecución de esas particiones de manera tal que la correctitud se mantenga y el array original puede ejecutar cada partición.

El resultado que se observa de esta técnica es que el array "navega" a través del conjunto de los cálculos del algoritmo en un orden correcto hasta "cubrir" todos los cálculos.

Otra técnica trata de replantear el problema a resolver de manera tal que el problema se convierta en un conjunto de problemas más pequeños similares al problema original y que puedan resolverse por el array sistólico.

Mientras que la segunda técnica tiene menos generalidad y es más difícil de automatizar que la primer técnica, puede tener mejor performance cuando se la utiliza.

## 5.7.4. – EJEMPLIFICACIÓN [S.O.]

A continuación veremos algunos ejemplos con algunas soluciones posibles.

### 5.7.4.1. - Ejemplo 1) [S.O.]

Se desea evaluar los polinomios :

$Y_j = \Sigma_i \; W_i * X_{i+j-1}$ ; para j de 1 a ≡

o sea :

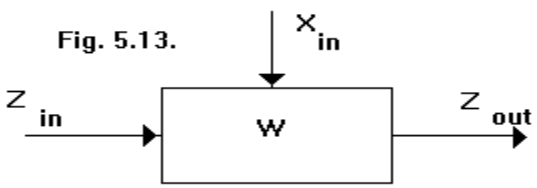
$Y_1 = W_1 * X_1 + W_2 * X_2 + W_3 * X_3$

$Y_2 = W_1 * X_2 + W_2 * X_3 + W_3 * X_4$

$Y_3 = W_1 * X_3 + W_2 * X_4 + W_3 * X_5$

..... etc .....

Se propone que cada procesador haga la siguiente función :




siendo $Z_{out} = Z_{in} + W * X_{in}$

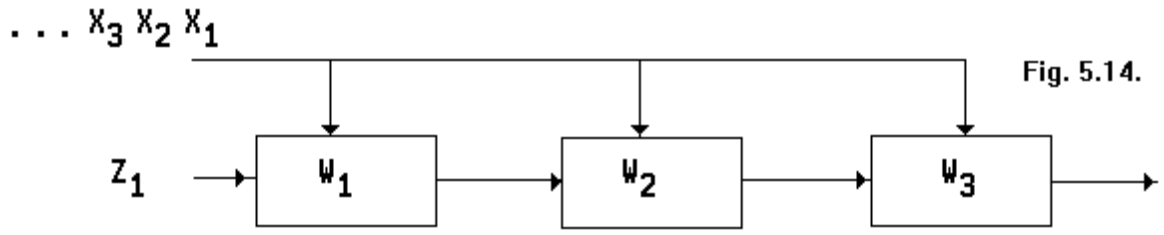


Fig. 5.14.

Luego el arreglo sistólico sería (para nuestro ejemplo de polinomios de tres componentes) :
Comenzamos con $Y_0 = 0$ resulta el siguiente proceso :

| | | | |
|---|---|---|---|
| 0 + W1*X1 | 0 + W2*X1 | 0 + W3*X1 | (1) |
| 0 + W1*X2 | W1*X1 + W2*X2 | W2*X1 + W3*X2 | (2) |
| 0 + W1*X3 | W1*X2 + W2*X3 | W1*X1 + W2*X2 + W3*X3 | (3) |
| 0 + W1*X4 | W1*X3 + W2*X4 | W1*X2 + W2*X3 + W3*X4 | (4) |
| 0 + W1*X5 | W1*X4 + W2*X5 | W1*X3 + W2*X4 + W3*X5 | (5) |
| .... | ..... | ..... | |

-(1) Se obtiene W3*X1, se descarta.
-(2) Se obtiene W2*X1 + W3*X2, se descarta.
-(3) Se obtiene W1*X1 + W2*X2 + W3*X3, primer resultado válido.
-(4) Se obtiene W1*X2 + W2*X3 + W3*X4, segundo resultado válido
y así sucesivamente....

**5.7.4.2. - Ejemplo 2) Multiplicación de matrices** [S.O.]

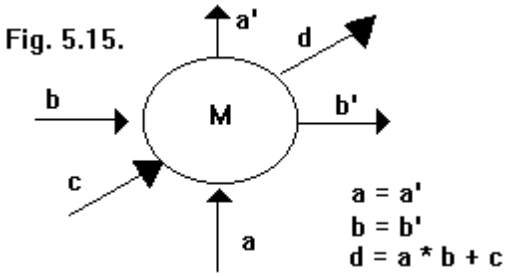


Fig. 5.15.

En la Fig. 5.15 se puede ver la estructura de una celda del sistólico.

Aquí las M son celdas (procesadores) multiplicativas-aditivas. En general se requerirán $3(n^2 - n) + 1$ celdas procesadoras, donde n es la dimensión de la matriz, y el cálculo se resolverá en $3( n - 1 )$ períodos (ciclos de reloj).

Veamos paso a paso el caso de la multiplicación de dos matrices de 2 * 2, donde se requieren cinco intervalos para obtener la matriz resultado.

$$A = \begin{matrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{matrix} \qquad B = \begin{matrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{matrix} \qquad A * B = C = \begin{matrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{matrix}$$

Intentaremos ver los que sucede en el sistólico por cada instante de tiempo, es decir, mostraremos por cada ciclo como fluye la información dentro del arreglo de celdas procesadoras.

Los PEs en el extremo superior izquierdo y en el inferior derecho no se utilizan en este cálculo, por lo tanto se los inactiva y su función es solamente la de rutear los datos que recibe.

Cantidad de celdas procesadoras : $3(n^2 - n) + 1$

Para n = 2 resulta : 7 celdas procesadoras (son efectivamente 7 las celdas de las cuales se obtienen resultados válidos).

La cantidad de períodos o ciclos en los que se obtiene el resultado final responden a la fórmula :
(3n - 1) que resulta en 5 intervalos t.
i) Instante $t_1$

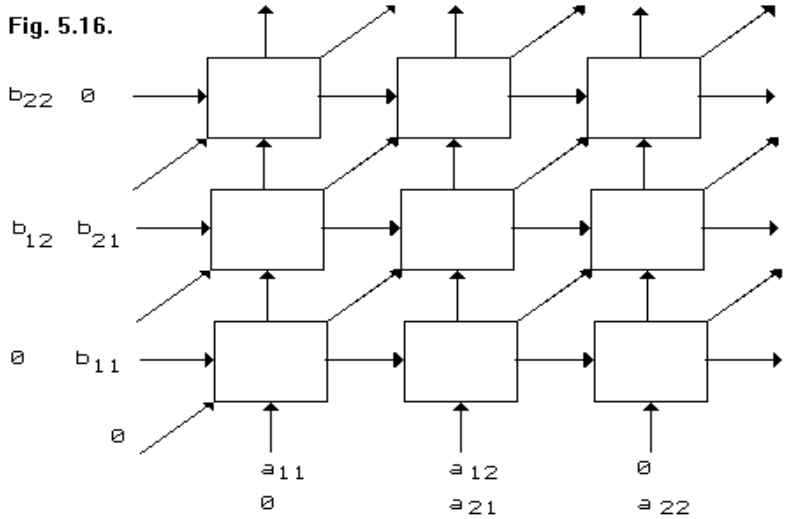
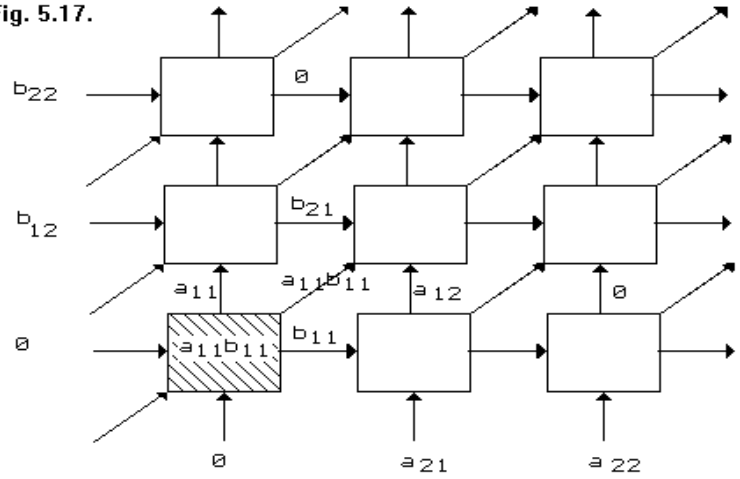


Fig. 5.16.

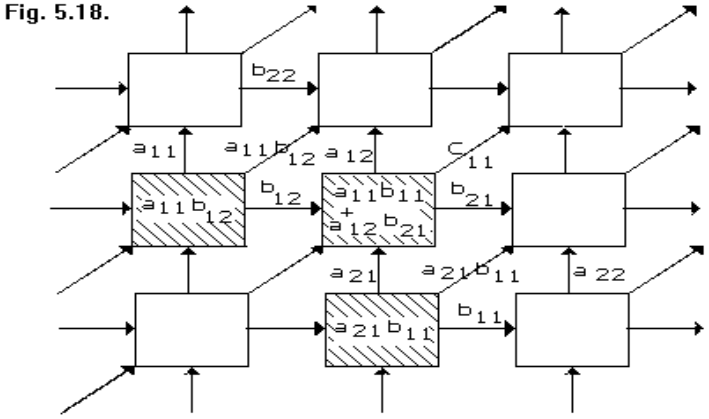
ii) Instante $t_2$

Fig. 5.17.



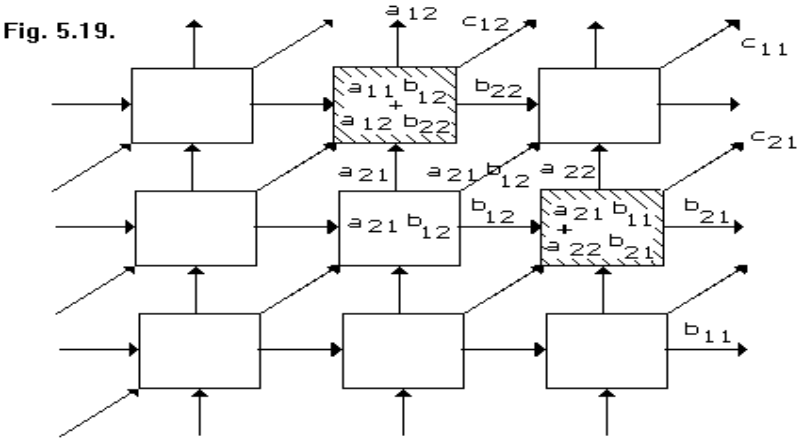

iii) Instante $t_3$

Fig. 5.18.



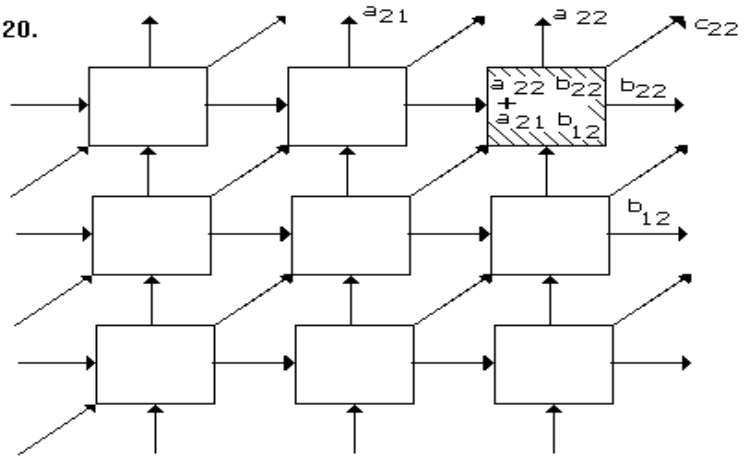

iv) Instante $t_4$

Fig. 5.19.



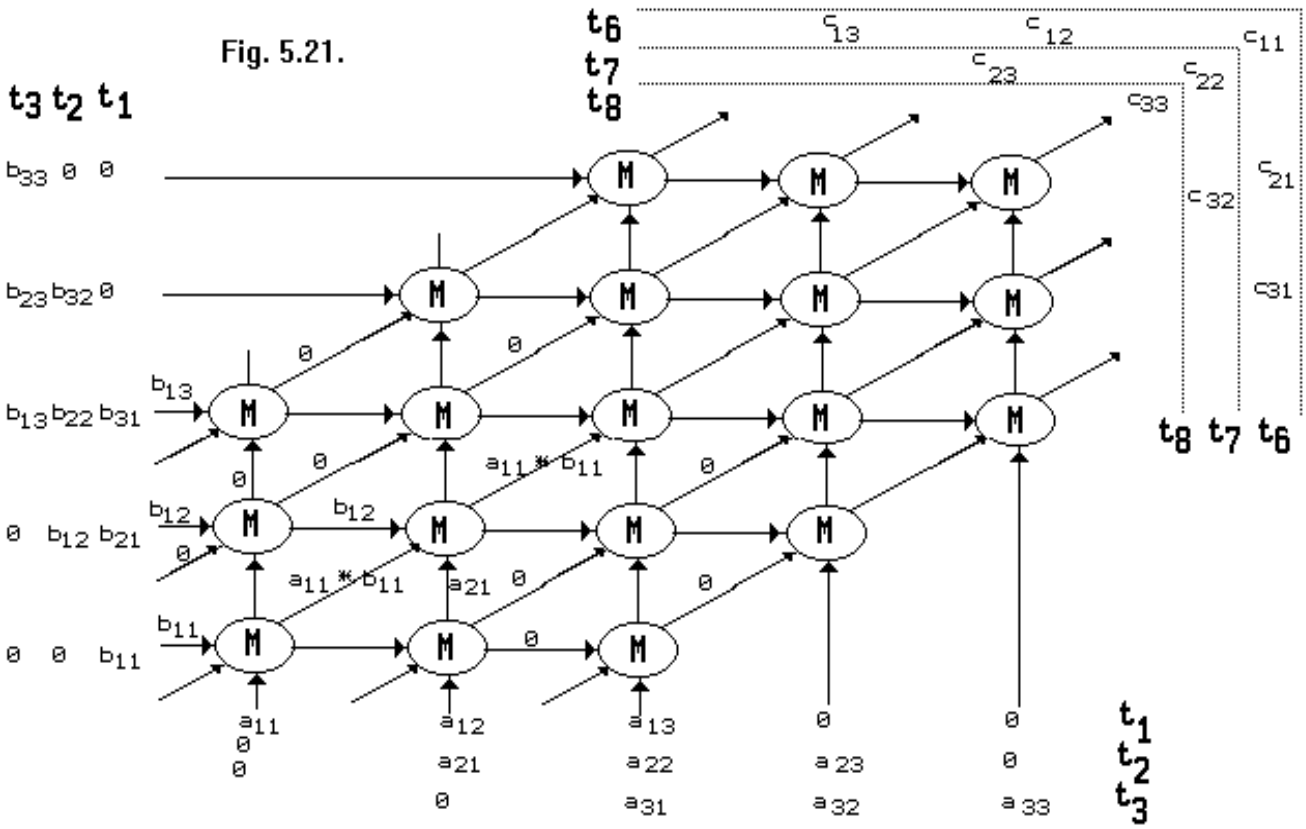
v)Instante $t_5$

Fig. 5.20.




La estructura completa para el cálculo de una multiplicación de dos matrices de 3 * 3 responde al siguiente gráfico de la Fig. 5.21.

Fig. 5.21.




**5.7.5 - Bloques de construcción universales** [36]

Los arrays sistólicos son más baratos de construir que los otros arrays debido a la profusa repetición de un pequeño número de módulos básicos y sencillos y debido a la alta densidad y eficiencia de sus layouts.

El diseño modular de los arrays sistólicos permite que aquellos diseñadores que desean armar rápidamente prototipos puedan valerse de dispositivos comunes (off-the-shelf), tales como microprocesadores, unidades de aritmética de punto flotante y chips de memoria.

Sin embargo, estas partes no fueron diseñadas para implementaciones de arrays sistólicos y pueden por consiguiente ser inadecuadas para cumplir los requerimientos del diseño.

Esto desembocó en el desarrollo de "bloques universales de construcción" que son chips que pueden utilizarse para muchos arrays sistólicos.

Algunos chips comercialmente disponibles de este tipo que merezcan esta definición incluyen al Transputer INMOS, al TI TMS32010 y TMS32020, el chip de dataflow NEC PD 7281, etc.

Los problemas que implica el uso de bloques de construcción programables incluyen el desarrollo de herramientas de programación de ayuda a los diseñadores y la provisión de un soporte para interconexiones flexibles.

### 5.7.6 - Integración en sistemas existentes [36]

A pesar de que los arrays sistólicos proveen una alta performance, su integración dentro de sistemas ya existentes es no trivial debido al amplio bandwidth de E/S que se produce, sobretodo cuando el problema debe ser particionado y los datos de entrada deben ser accedidos repetidamente.

Los problemas adicionales que deben resolverse en aquellos sistemas que cuentan con gran cantidad de arrays sistólicos incluyen la interconexión con el host, el subsistema de memoria que soporte los array sistólicos, el bufferizado y el acceso a datos para obtener la distribución especial de datos de E/S, y la multiplexación y desmultiplexación de datos cuando son insuficientes las puertas de E/S.

### EJERCICIOS

**1)** Una arquitectura de procesador array es sincrónica o asincrónica ? Justifique.
**2)** Cuál es la diferencia entre las dos implementaciones más clásicas de procesadores array ?
**3)** Un PE es una CPU ? Justifique.
**4)** Qué es y para qué se utiliza el esquema de enmascaramiento de PEs ?
**5)** El enmascaramiento y la función de ruteo en arquitecturas SIMD son la misma cosa ? Justifique.
**6)** Construya el vector de máscaras y la función de ruteo para cada uno de los pasos del ejemplo del apunte (Fig. 5.6).
**7)** Cómo trabaja un procesador array de tipo bit-plane ?
**8)** Construya una función AND para un computador de memoria asociativa que opere solamente sobre las filas pares de la memoria.
**9)** Qué es un array sistólico ? Cómo funciona ?
**8)** Puede clasificarse a los sistólicos como arquitecturas MISD ? Discuta.
**10)** Puede un array sistólico ser un procesador de propósito general ? Justifique.
**11)** Qué mecanismo se utiliza para manejar grandes problemas que se desea procesar en arrays sistólicos ?
**12)** Los procesadores sistólicos son de gran utilidad para aquellos problemas de tipo compute-bound ? Justifique.
**13)** Sea el esquema de interconexión de celdas en un array sistólico de la Figura 5.22.

Cómo deben ingresar los valores de las matrices A y B de 2 x 2 para evaluar la matriz C = A * B ? Indique expresamente paso a paso la evaluación no olvidando los valores de sincronismo.

Ayuda : los caminos horizontales y verticales son solo para datos, los caminos en diagonal se utilizan para rutear los resultados de las multiplicaciones a otras celdas que realizan las sumas.

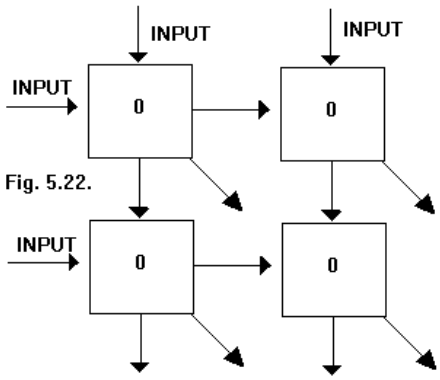


Fig. 5.22.