Tema 6: Procesadores Vectoriales

- **Objetivos**
- **Referencias**
- **Arquitectura básica**
  - **Diseño de un procesador vectorial**
  - **Sistema de memoria**
  - **Problemas**
  - **Mejoras a la arquitectura**
- **Vectorización del código**
- **Máquinas SIMD**
- **Redes de interconexión para máquinas SIMD**

1

Objetivos:

- **Comprender la existencia de paralelismo de datos en los programas.**
- **Conocer las distintas formas de explotar este paralelismo.**
- **Comprender las limitaciones que se presentan a la hora de intentar explotarlo.**

2

Referencias:

- **Para máquinas vectoriales, el Hennessy-Patterson, 2ª edición (muchas figuras y ejemplos están tomadas de él)**
- **Para máquinas SIMD, el Kai Hwang**

Utilidad de un procesador vectorial

Ej.: calcular **Y** = a**X** + **Y**

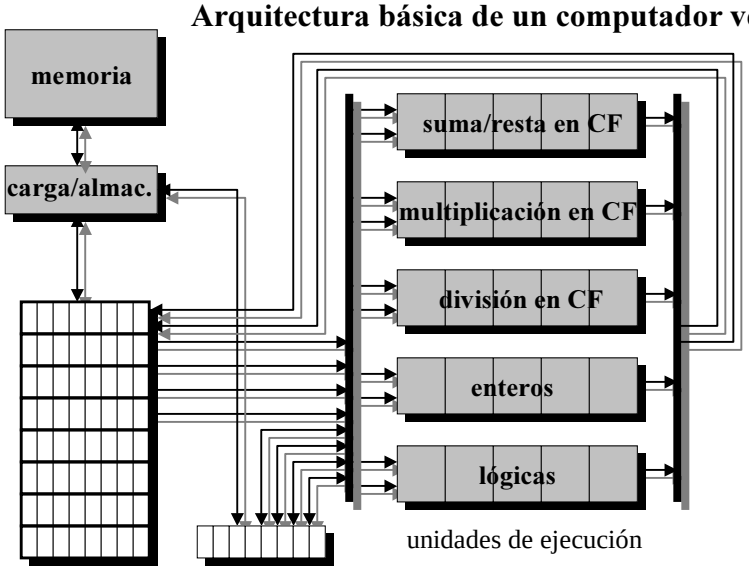
escalar

```
      ld    f0,a
      addi  r4,rx,#512
loop: ld    f2,0(rx)
      multd f2,f0,f2
      ld    f4,0(ry)
      add   f4,f2,f4
      sd    f4,0(ry)
      addi  rx,rx,#8
      addi  ry,ry,#8
      sub   r20,r4,rx
      bnz   r20,loop
```

vectorial

```
ld    f0,a
lv    v1,rx
multsv v2,f0,v1
lv    v3,ry
addv  v4,v2,v3
sv    ry,v4
```

Arquitectura básica de un computador vectorial



Elementos de la arquitectura

Registros vectoriales:
- contienen los operandos vectoriales en máquinas de registros
- no existen si la máquina es memoria-memoria
- valores típicos de componentes son 64 o 128
- deben tener al menos 2 puertos de lectura y uno de escritura

Unidades funcionales vectoriales:
- ejecutan las operaciones vectoriales
- están segmentadas, y suelen tener latencia 1
- una unidad de control vigila las dependencias

Unidad de carga y almacenamiento:
- gestiona transferencias de vectores desde/a memoria
- puede estar segmentada
- también puede ocuparse de los datos escalares

Elementos de la arquitectura (II)

Registros escalares:
- contienen los operandos escalares
- se usan en operaciones vectoriales y para calcular direcciones
- se necesitan varios puertos de lectura y escritura

Unidades funcionales escalares:
- pueden existir para operaciones específicamente escalares
- pueden no existir si para operaciones escalares se usan las unidades vectoriales

Ventajas de las máquinas vectoriales

- **Proporcionan gran aprovechamiento del paralelismo (de datos) con un control relativamente sencillo.**

- **Una única instrucción especifica una gran cantidad de trabajo, reduciendo la necesidad de ancho de banda de instrucciones.**

- **Optimizan el uso de la memoria con accesos predecibles que se pueden solapar.**

- **Eliminan dependencias de control e instrucciones de comprobación y contabilidad.**

Sistema de memoria

En una máquina vectorial, los accesos no son a datos individuales, sino a colecciones de ellos (vectores).

La distribución de estos datos en memoria sigue una ecuación generalmente sencilla. Por ejemplo, en una matriz almacenada por filas:

- leer un vector-fila es leer posiciones de memoria consecutivas
- leer un vector-columna es leer posiciones distanciadas en *n*, donde *n* es el número de elementos de una fila

El sistema de memoria se diseña de forma que:

- se puedan realizar accesos a varios elementos a la vez
- el tiempo de inicio del acceso sea sólo para el primer elemento

9

Sistema de memoria (II)

Se utilizan principalmente dos opciones:

- memoria entrelazada (generalmente de orden inferior, con factor de entrelazado palabra)
- bancos de memoria independientes

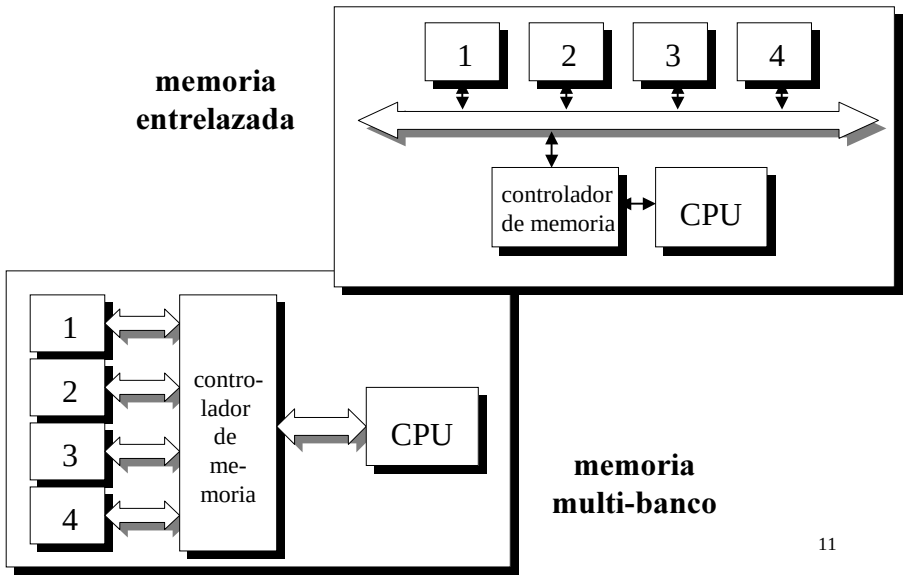
La memoria entrelazada tiene un diseño más sencillo y menos costoso, pero menos flexible

La memoria en bancos independientes es más costosa (cada banco necesita su bus), pero es más flexible

Se puede pensar en soluciones intermedias: memoria en bancos independientes con un bus compartido por todos los bancos

10

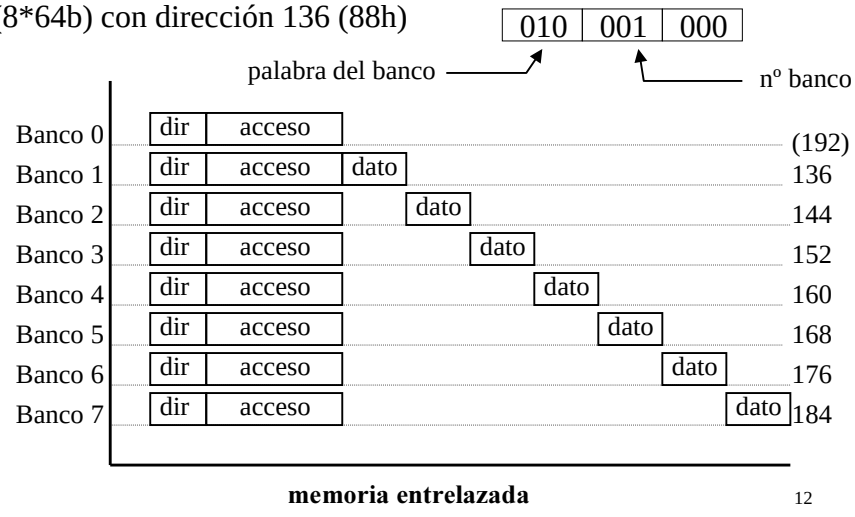
Sistema de memoria (III)



11

Sistema de memoria (IV)

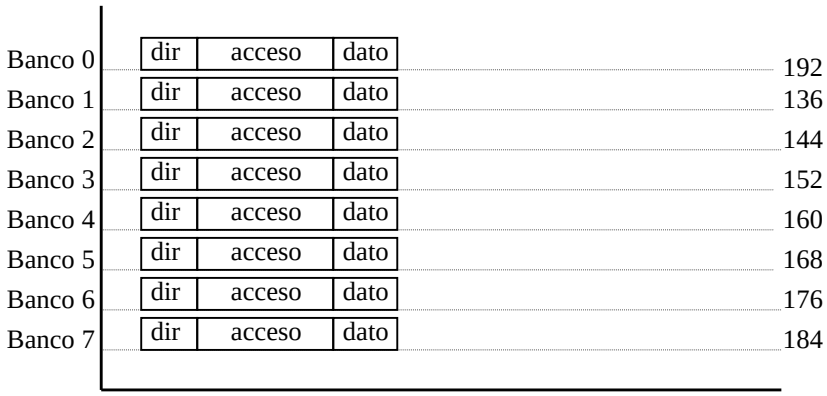
Ejemplo de funcionamiento: acceso al vector de 8 dobles palabras (8*64b) con dirección 136 (88h)



12

Sistema de memoria (V)

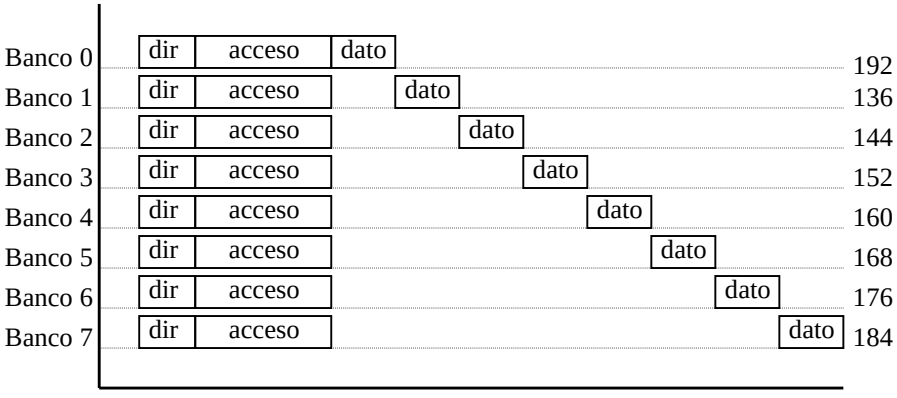
Ej.: el mismo acceso



memoria multibanco (bus dedicado)

Sistema de memoria (VI)

Ej.: el mismo acceso



memoria multibanco (bus de datos compartido)

Problemas

Problema 1: la longitud del vector en la aplicación no tiene por qué coincidir con el tamaño de los registros (generalmente no lo hace)

Solución: se añade un registro escalar llamado VLR (*vector length register*, registro de longitud).

- Este registro contiene el número de elementos de los registros que se utilizan en cada operación
- Si un cálculo necesita vectores mayores que lo que cabe en un registro, se divide en bloques y se hace un bucle (*strip-mining*)

Problemas (II)

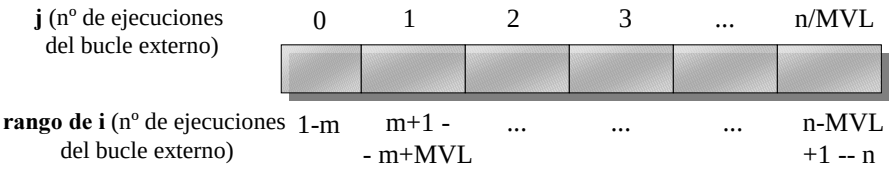
Ejemplo de *strip-mining*:

```
      do   10 i=1,n
10    Y(i) = a*X(i) + Y(i)
```

```
inicio=1
VL=(n mod MVL)
do 1 j=0,(n/MVL)
      do 10 i=inicio,inicio+VL-1
            Y(i)=a*X(i)+Y(i)
      10 continue
      incio=inicio+VL
      VL=MVL
1 continue
```

Problemas (III)

Ejecución del bucle con *strip-mining:*



donde: m=n mod MVL

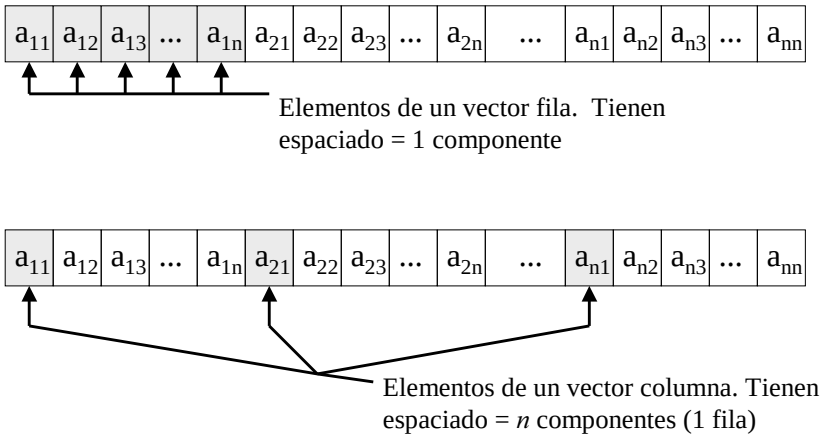
Problemas (IV)

Problema 2: el espaciado (‘*stride’*) de un vector en memoria no tiene por qué ser 1 (es decir, los elementos de un vector pueden no estar contiguos en memoria).

Ejemplo: almacenamiento de una matriz en memoria (por filas)

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}$$

Problemas (V)



Problemas (VI)

Solución: se indica el espaciado con un registro específico o con un registro de datos. Se pueden incluir instrucciones específicas para acceso a vectores espaciados, o hacer que las instrucciones de carga y almacenamiento de vectores siempre trabajen con espaciado.

Ejemplo: en DLX, versión vectorial, existen las instrucciones:

- –LVWS V1,(R1,R2): carga desde la dirección en R1 con el espaciado que indica R2 en V1
- –SVWS (R1,R2),V1: almacena V1 a partir de la dirección en R1 con el espaciado que indica R2

Problemas (VII)

Efecto sobre el sistema de memoria:

- Si el espaciado es 1, una memoria entrelazada, o un diseño por bancos independientes, proporcionan el mejor rendimiento.
- Sin embargo, con espaciado no unidad, puede no alcanzarse el rendimiento óptimo. Ejemplo: si el espaciado es 8, y hay 8 bancos, todos los accesos van al mismo banco (=>secuencialización).
- Por esto, es interesante que el espaciado y el número de bancos sean primos entre sí.
- Como el espaciado es una característica del problema, el compilador puede intervenir si ocurre una coincidencia como la del ejemplo (por ejemplo, añadiendo una columna "hueca" en la matriz).

Problemas (VIII)

Problema 3: el bucle que se quiere vectorizar lleva dentro sentencias condicionales.

Ejemplo: **C** = **A** / **B**

Queremos que se calcule la división sólo si B(i) no es 0 (para que no se genere una excepción).

```
      do 100 i = 1,64
           if (B(i).ne. 0) then
                C(i)=A(i)/B(i)
           endif
100   continue
```

Problemas (IX)

Solución: utilizar vectores de máscara. Un vector de máscara es un vector de valores booleanos (tantos como componentes tenga el vector). Si para una componente la máscara es 0, la operación con esa componente no se ejecuta.

```
lv   v1,ra
lv   v2,rb
ld   f0,#0
snesvf0,v1
divv v1,v1,v2
cvm
sv   ra,v1
```

Activa o no él bit de máscara correspondiente a cada componente

Divide utilizando el vector de máscara

Borra (pone a 1) los bits del vector de máscara

Mejoras a la arquitectura

Encadenamiento ('chaining')

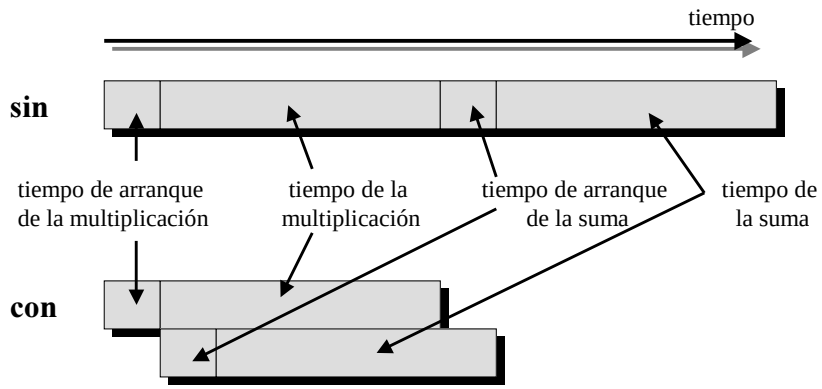
Consiste en solapar la ejecución de distintas instrucciones vectoriales (aunque haya dependencias entre ellas).
Ejemplo: el par de instrucciones:

```
multv v1,v2,v3
addv  v4,v1,v5
```

Como hay una dependencia (v1), no podrían ejecutarse en paralelo.
Sin embargo, no es necesario que la suma espere a que termine *toda* la multiplicación: a medida que se van generando componentes de v1, la suma puede irlas procesando

Mejoras a la arquitectura (II)

Ejecución del código anterior sin y con encadenamiento



25

Vectorización del código

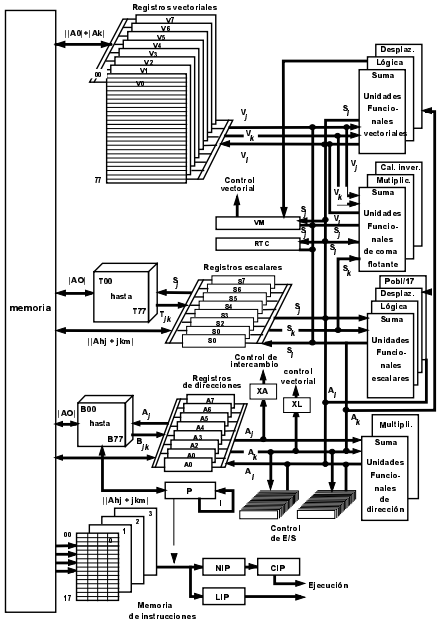
Hay varios requisitos para poder vectorizar el código:

1º: que no haya dependencias de datos
(por ejemplo, las hay en el cálculo de la suma de las componentes de un vector)
2º: que el programa esté expresado de una forma que permita su vectorización (al compilador)

La calidad del compilador también es importante

Hay compiladores que son capaces de detectar más fácilmente código vectorizable que otros.

26



Arquitectura del CRAY-1

Arquitectura de una máquina SIMD (memoria distribuida)

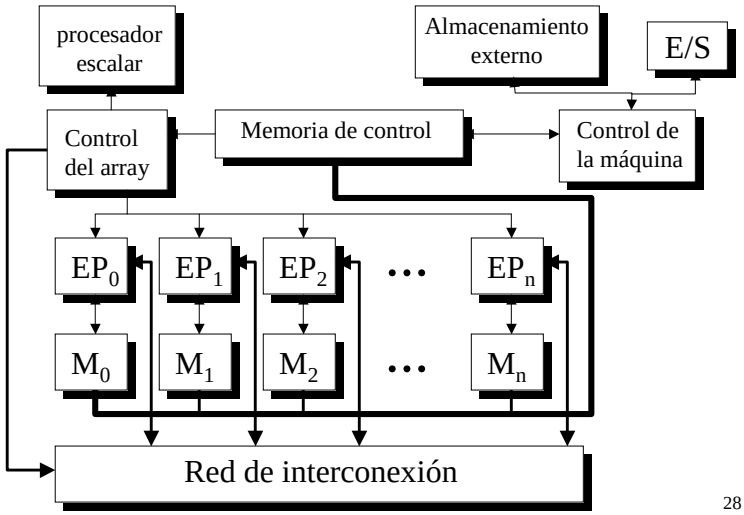


Figura tomada de Kai Hwang, 'Advanced Computer Architecture'

Arquitectura de una máquina SIMD (memoria compartida)

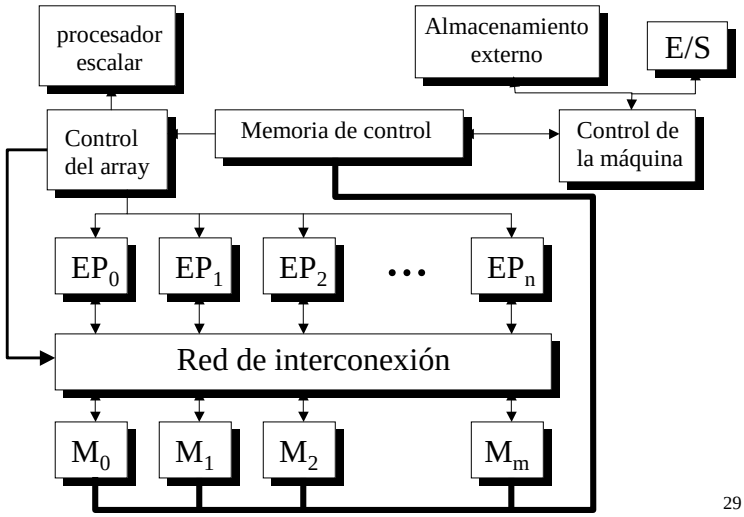


Figura tomada de Kai Hwang, 'Advanced Computer Architecture'

Máquinas SIMD

•Las máquinas SIMD plantean otra forma de aprovechar el paralelismo en los datos.

•Ejecutan la misma instrucción sobre datos distintos de forma simultánea.

•Constan de un array de elementos de proceso (ALUs) que son controlados (en cuanto a la operación que realizan y los datos que usan) por un módulo de control del array.

•Existe también un procesador escalar, para las operaciones SISD.

•Los elementos de proceso están unidos por una red de interconexión.

Máquinas SIMD (II)

•**El módulo de control recibe información (presente en el programa) sobre**:
la instrucción que hay que ejecutar
el estado de la red de interconexión (por si los datos de los EP no se encuentran en su memoria local)
•**Transmite la operación que hay que realizar y la localización de los operandos a los elementos de proceso.**

•**Configura la red de interconexión según convenga.**

•**En ese momento, los EPs ejecutan la instrucción**

Máquinas SIMD (III)

El programa debe:
- **especificar las instrucciones que son para ejecutar en modo SIMD y las que son escalares**
- **especificar la distribución de los datos en las memorias**
- **especificar la configuración de la red en cada instante**

Si la memoria es compartida:
- **los datos no son locales a cada elemento de proceso**
- **el número de módulos de memoria no tiene por qué coincidir con el de elementos de proceso**
- **el resto de las características son parecidas**

Máquinas SIMD (IV)

•Es posible combinar las características de una máquina vectorial y de una SIMD.

•El diseño sería básicamente el de una máquina vectorial que tiene unidades funcionales repetidas.

•Las unidades funcionales estarían conectadas como los elementos de proceso de una máquina SIMD, para poder trabajar de forma simultánea.

•Ejemplo: una máquina vectorial con cuatro unidades de suma. La primera trabajaría con las componentes 0, 4, 8, etc de cada vector, la segunda con la 1, 5, 9, etc, y así sucesivamente.

33

Redes de interconexión

•Las distintas redes de conexión se pueden clasificar de varias maneras:

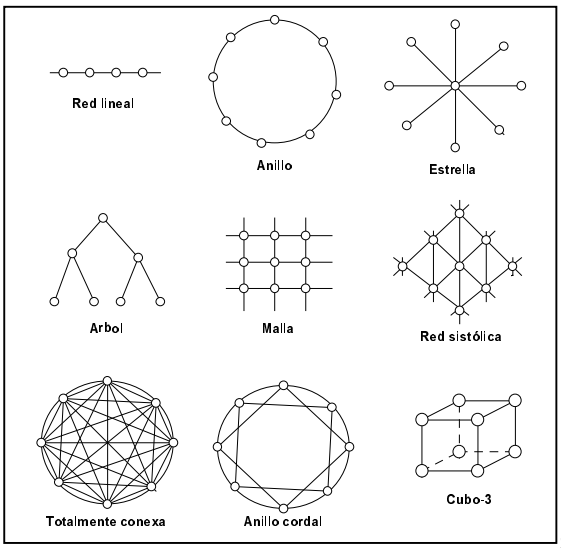
–por el modo de operación: síncronas o asíncronas
–por la estrategia de control: centralizado o distribuido
–por la metodología de conmutación: de circuitos o de paquetes (también existe la integrada, que puede funcionar de ambas formas)
–por la topología: estáticas o dinámicas.

•Las máquinas SIMD utilizan redes síncronas, con control centralizado y con conmutación de circuitos. Pueden ser estáticas o dinámicas.

34

Redes de interconexión (II)

Ejemplos de redes estáticas



35

Redes de interconexión (III)

Redes dinámicas: son redes cuya configuración puede modificarse. Hay dos tipos:

- monoetapa
- multietapa

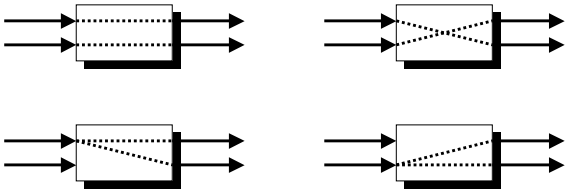
Las redes monoetapa realizan conexiones entre elementos de proceso en 1 paso.
Puede no ser posible llegar desde cualquier elemento a cualquiera, por lo que puede ser necesario recircular la información (=>redes recirculantes)

Las redes multietapa realizan conexiones entre los elementos de proceso en más de 1 paso.

36

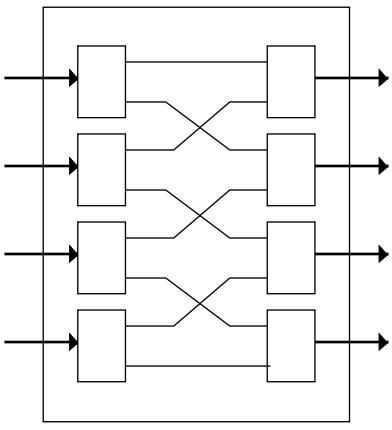
Redes de interconexión (V)

Las redes multietapa se forman a partir de cajas de conmutación



Las cuatro configuraciones posibles de una caja de conmutación de 2 entradas

Redes de interconexión (IV)

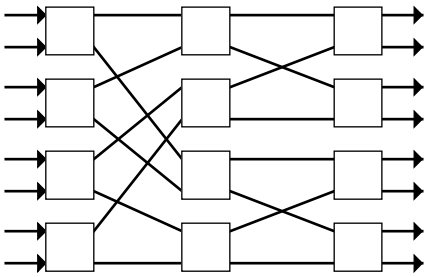


Ejemplo de red monoetapa

- Las redes monoetapa tienen un Selector de Entrada y un Selector de Salida.
- El SE es básicamente un demultiplexor, y el SS un multiplexor.
- Un caso particular de red monoetapa es la red de barras cruzadas.

Redes de interconexión (VI)

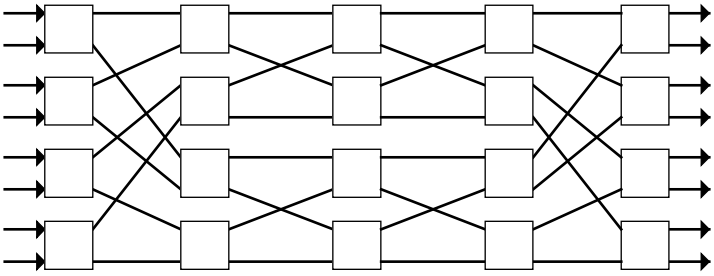
Redes multietapa. Grupo 1: con bloqueo: la conexión simultánea de varias parejas de terminales puede producir conflictos.



Red de línea base 8 x 8

Redes de interconexión (VII)

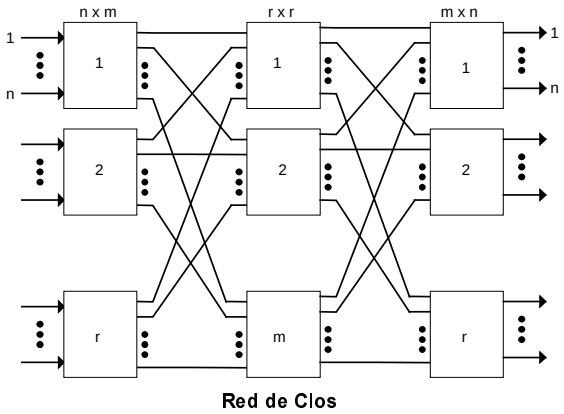
Redes multietapa. Grupo 2: reordenable: reordenando la configuración es capaz de establecer una línea de comunicación para una nueva pareja.



Red de Benes 8 x 8

Redes de interconexión (VIII)

Redes multietapa. Grupo 3: sin bloqueo: puede manejar todas las conexiones posibles sin que se produzcan bloqueos.



Red de Clos

41