

# Diseño de Sistemas Digitales

FCHE 2011-2

Dispositivos Lógicos Programables

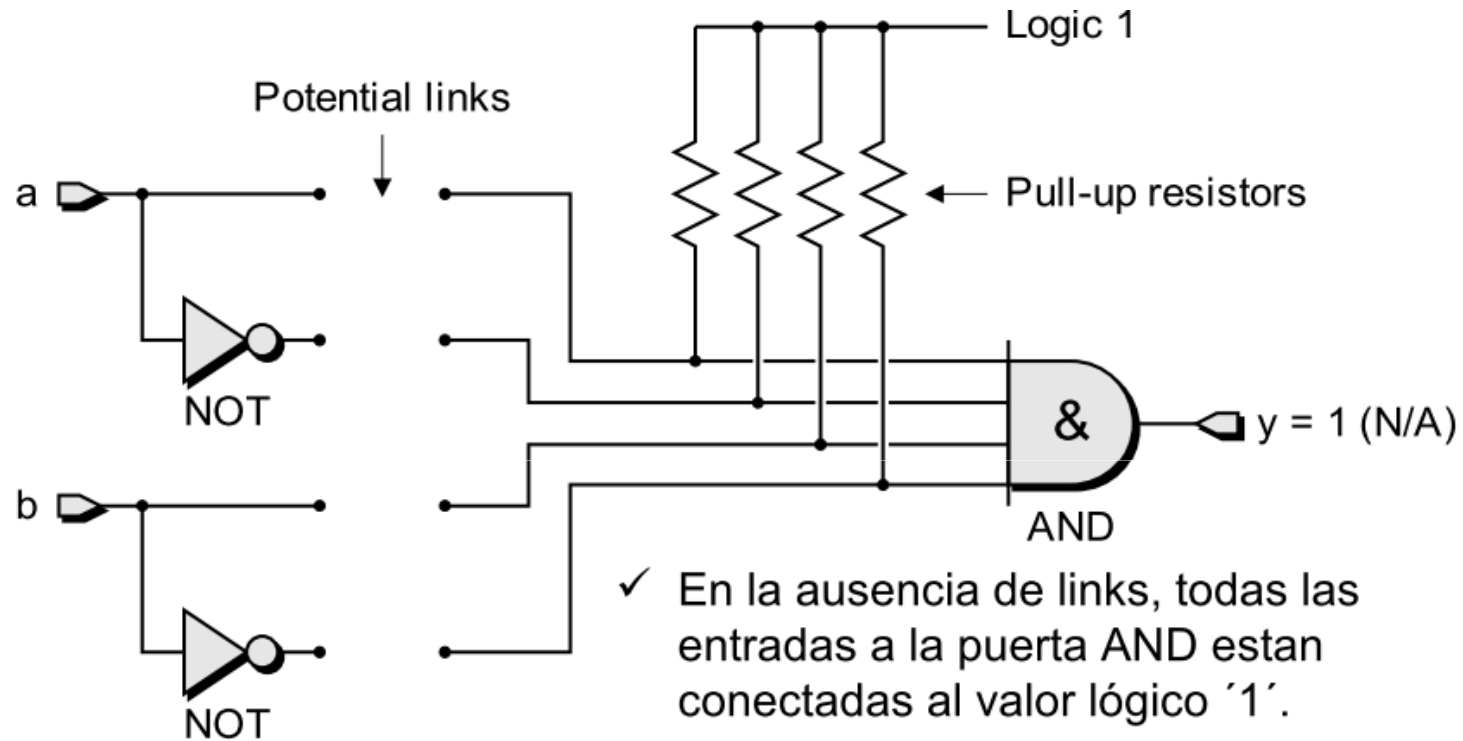
## 2 Circuitos combinacionales

**Objetivo:** El alumno conocerá los componentes electrónicos básicos involucrados en los circuitos combinacionales, así como los bloques funcionales combinacionales más utilizados en el diseño de sistemas digitales tanto en descripción estructural como por comportamiento usando HDL.

### Contenido:

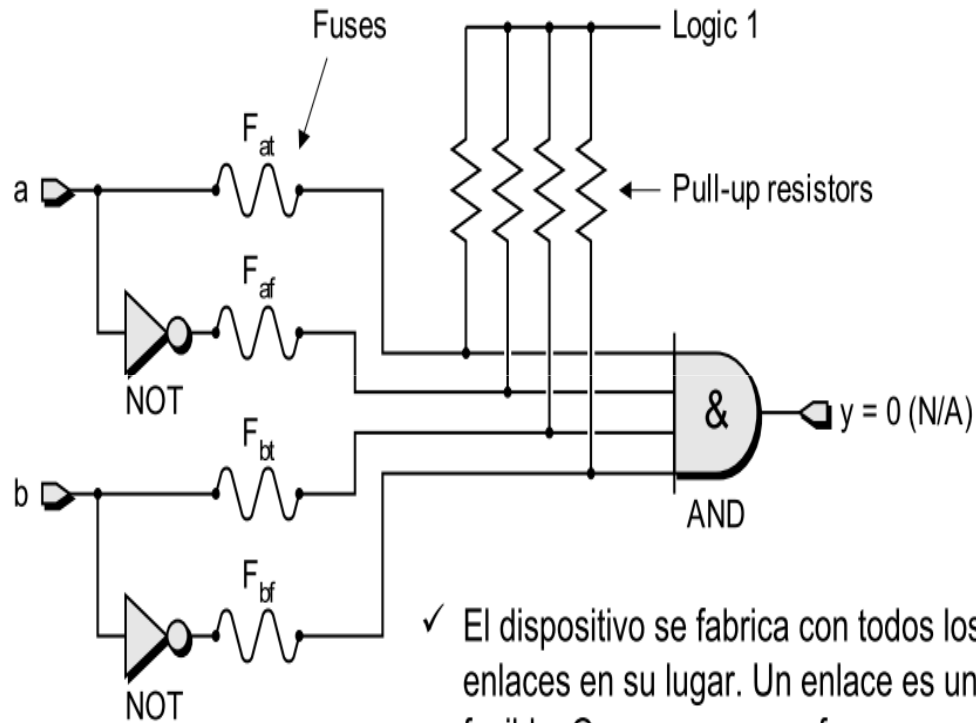
- 2.1 Compuertas lógicas AND, OR, NOT, NAND, NOR, XOR, XNOR
- 2.2 Formas canónicas, estándar, mintérminos y maxtérminos
- 2.3 Minimización de funciones booleanas con mapas de Karnaugh y Quine-McCluskey
- 2.4 Circuitos integrados, familias lógicas
- 2.5 Interpretación de parámetros en las hojas de datos de las compuertas lógicas
- 2.6 Convenciones de lógica positiva y lógica negativa
- 2.7 Representación estructural y por comportamiento de las compuertas lógicas en algún lenguaje de descripción de hardware (HDL)
- 2.8 Implementación estructural y por comportamiento de Funciones Booleanas usando algún HDL
- 2.9 Universalidad de las compuertas NAND y NOR
- 2.10 Propiedades de la XOR y NXOR para la implementación de generadores y detectores de paridad
- 2.11 Análisis de tiempo en la implementación de funciones booleanas
- 2.12 Descripción estructural y por comportamiento, usando HDL, de los bloques combinacionales fundamentales: Medio sumador, sumador completo, sumador/restador de "n" bits, comparadores de "n" bits, sumadores BCD, multiplicadores de "NxM" bits, decodificadores, codificadores, decodificadores BCD – 7 SEG, multiplexores, demultiplexores
- 2.13 Dispositivos lógicos programables elementales: ROM, PLA, PAL
- 2.14 Implementación de funciones booleanas con decodificadores, multiplexores, ROM, PLA y PAL

# Tecnologías de Programación

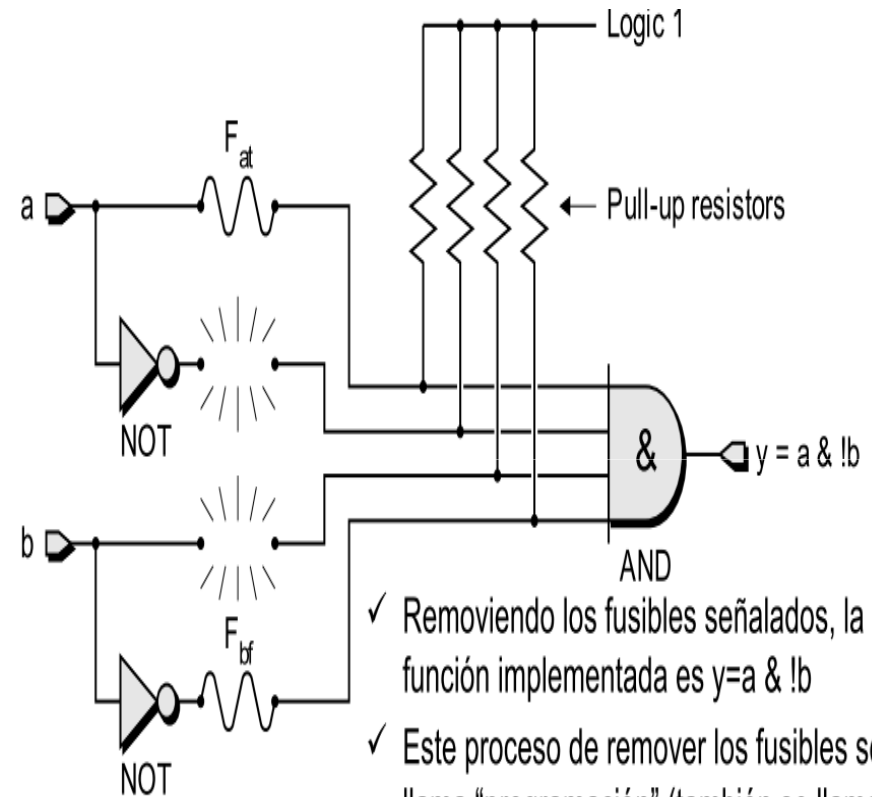


- ✓ En la ausencia de links, todas las entradas a la puerta AND están conectadas al valor lógico '1'.
- ✓ Los pull-up resistors mantienen débilmente el valor lógico '1'.
- ✓ Para realizar una función hay que buscar un mecanismo que permita establecer uno o más links.

# Tecnologías de Programación 1. fusible

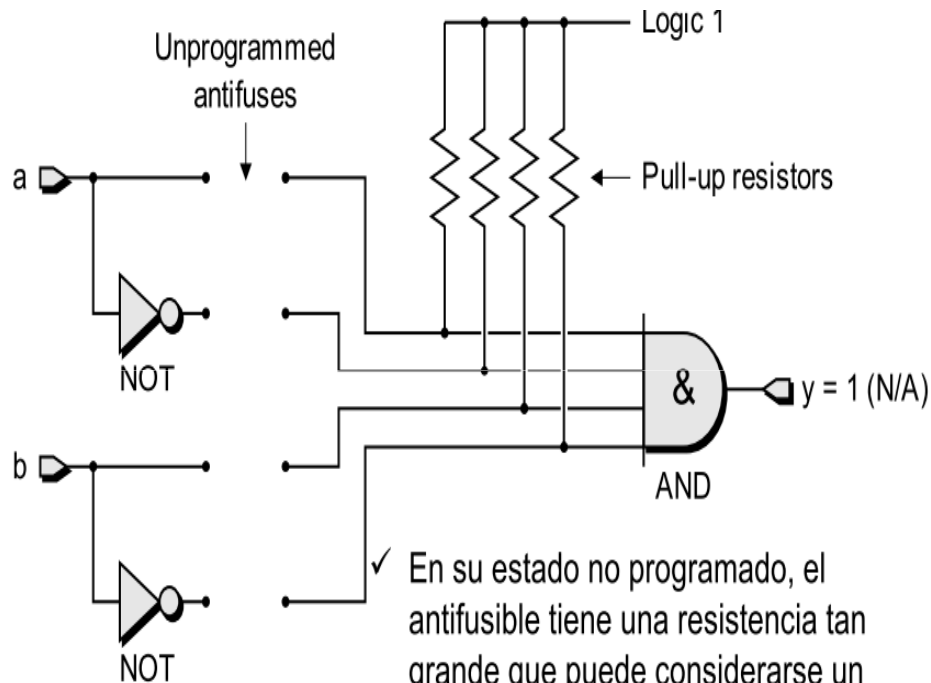


- ✓ El dispositivo se fabrica con todos los enlaces en su lugar. Un enlace es un fusible. O sea que, en su forma no programada, la función valdrá siempre '0'.
- ✓ Para remover los fusibles se aplican pulsos de un voltaje alto a las entradas

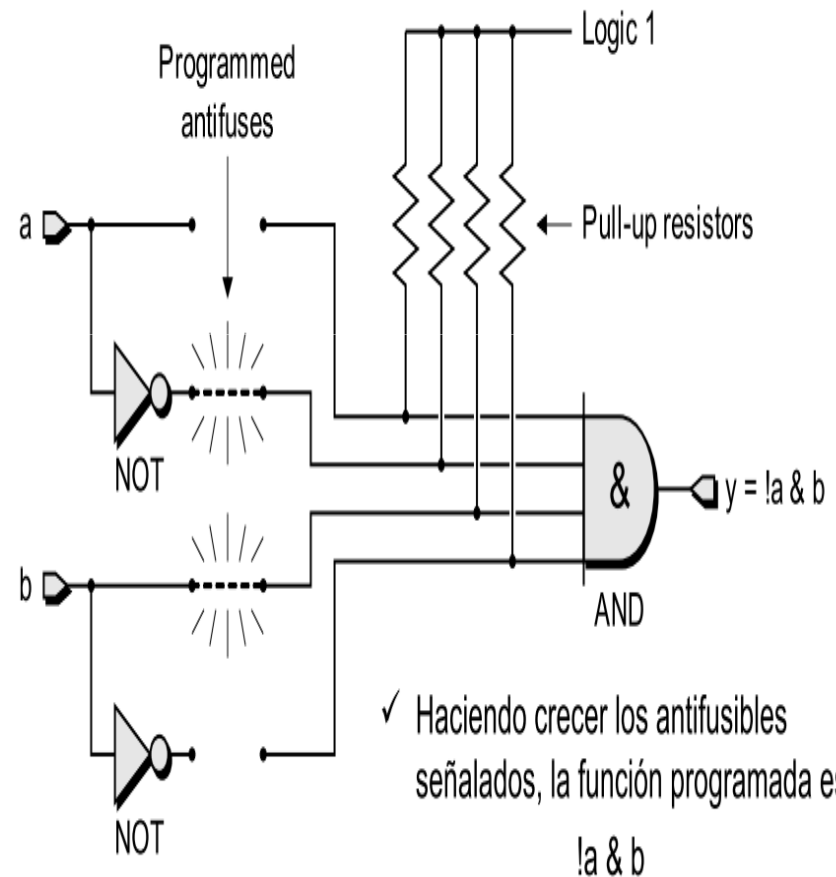


- ✓ Removiendo los fusibles señalados, la función implementada es  $y = a \& !b$
- ✓ Este proceso de remover los fusibles se llama "programación" (también se llama "blowing" o "burning")
- ✓ Los dispositivos son OTP, porque el fusible no puede recuperarse después de haberse quemado.

# Tecn. de Prog. 2. antifusible

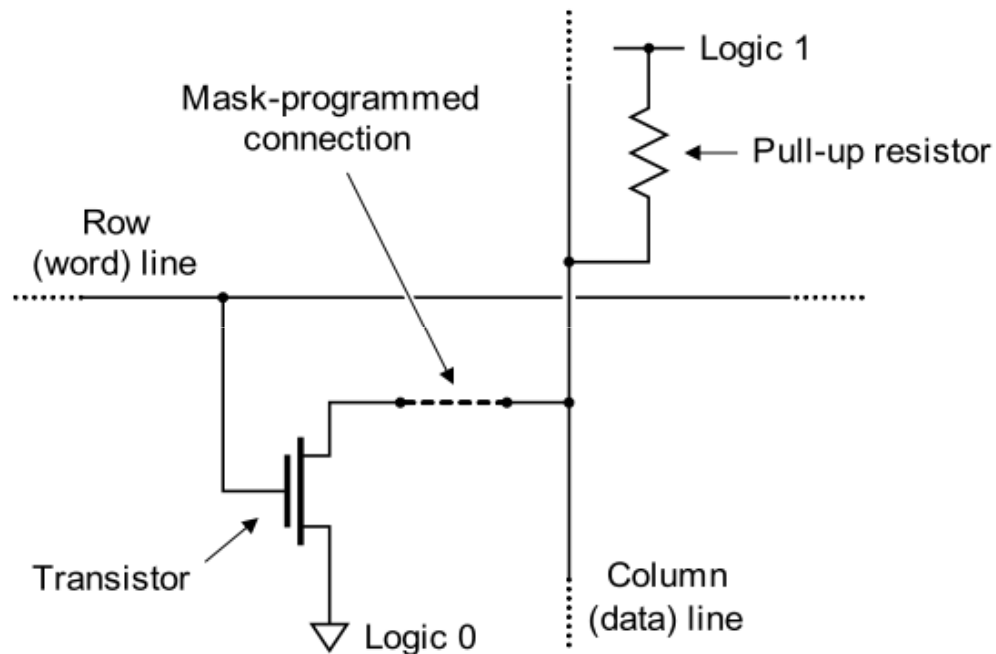


- ✓ En su estado no programado, el antifusible tiene una resistencia tan grande que puede considerarse un circuito abierto.
- ✓ Cuando se programa, (se dice que ha sido crecido (grown)), aplicando pulsos de alto voltaje y corriente a las entradas del dispositivo.



- ✓ Haciendo crecer los antifusibles señalados, la función programada es:  
 $!a \& b$

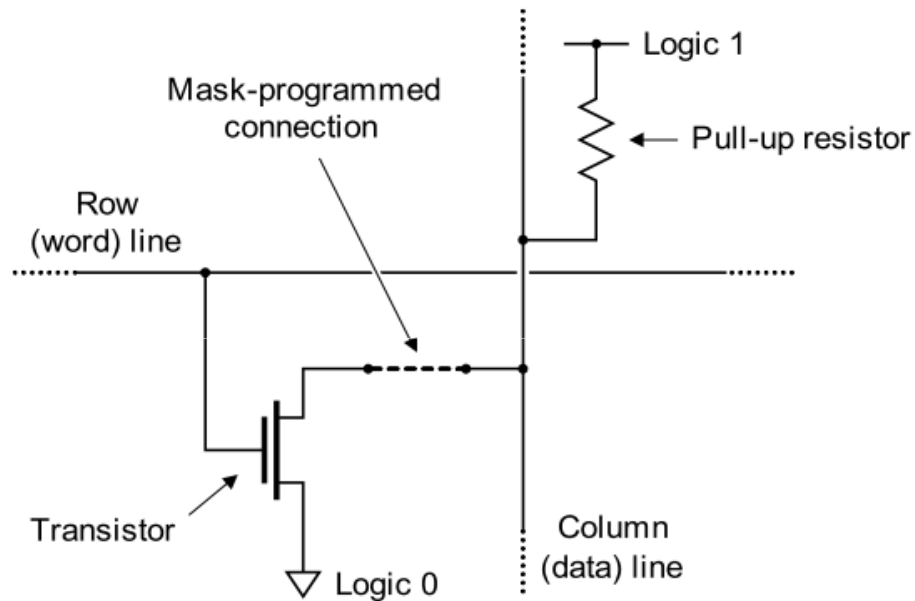
# Tecn. de Prog. ROM



Celda de una memoria ROM

- ✓ Consiste de array de filas (row) y columnas
- ✓ Cada columna tiene un único pull-up que intenta mantener a "1" esa columna
- ✓ Cada intersección fila/columna tiene un transistor y una "conexión" potencial
- ✓ La ROM se preconstruye y la misma arquitectura puede usarse para múltiples clientes.

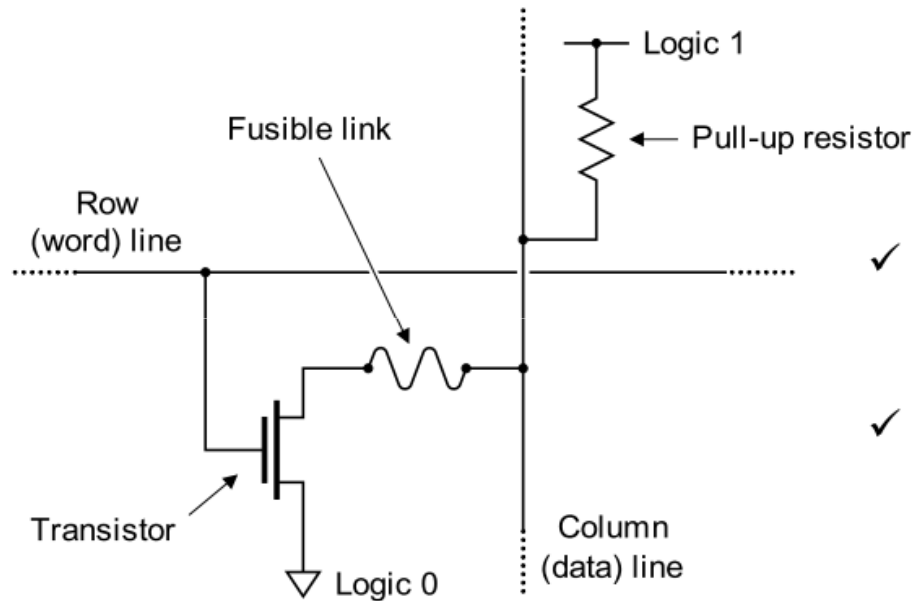
# Tecn. de Prog. ROM prog. con mascara



Celda de una memoria ROM

- ✓ Se pre-construyen y, para adaptarlas a los requerimientos del cliente se utiliza una máscara fotográfica para definir cuales celdas tendrán o no una conexión programada.
- ✓ Si la línea de fila se activa, el transistor se activa y :
  - ✓ Si hay conexión, en la columna aparece el valor lógico 0
  - ✓ Si no hay conexión, en la columna sigue el valor '1' del pullup.

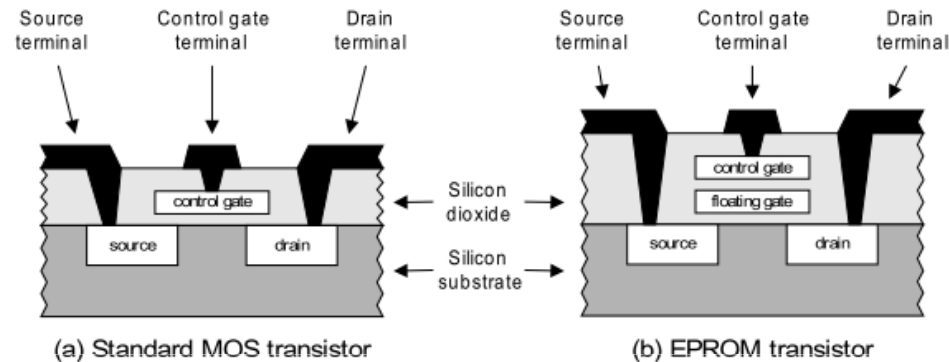
# Tecn. de Prog. PROM



Celda de una memoria PROM

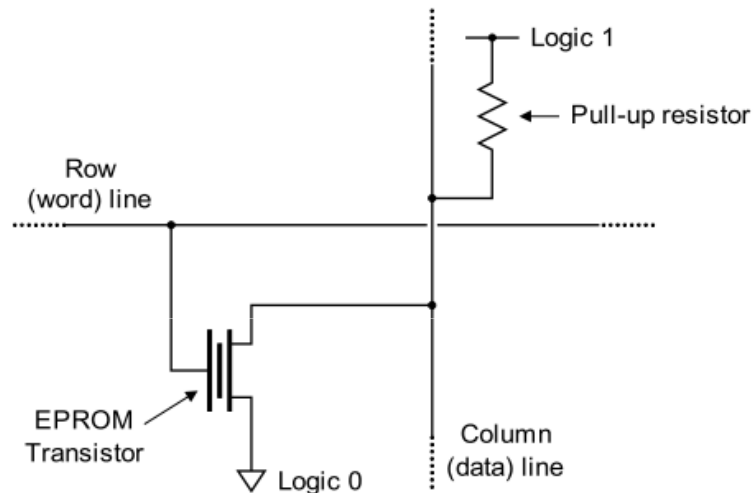
- ✓ Problema de los dispositivos programados con máscara: son caros! Se hacen en la fábrica y solo salen a cuenta si son muchísimos
- ✓ Programmable ROM (1970) están basados en la tecnología de fusible link.
- ✓ En su estado no programado, tal como se compra, todos los enlaces están presentes. O sea, si la línea se activa, la columna conduce '0'.
- ✓ La programación al remover los enlaces, hace que la celda almacene un '1'.

# Tecn. de Prog. EPROM



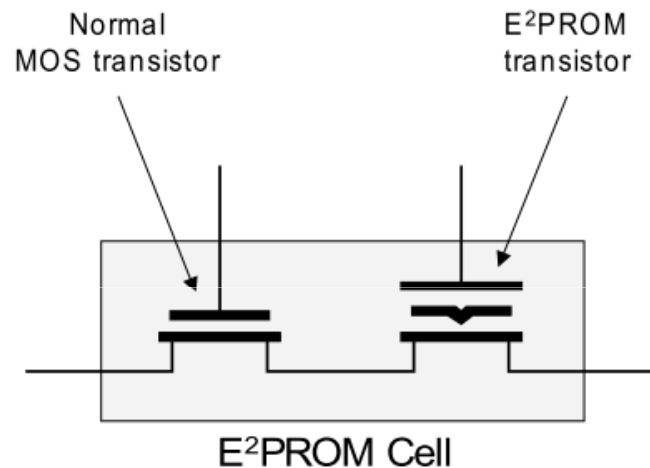
- ✓ Problema con las tecnologías basadas en fusible links y máscaras → son OTP.
- ✓ Erasable Programmable EPROM (1971) : los transistores tienen una puerta adicional de polisilicio : puerta flotante
- ✓ En su estado no programado, la puerta flotante no está cargada y no afecta el normal funcionamiento del transistor.
- ✓ Al programar el transistor, se carga la puerta flotante, inhibiendo la normal operación del transistor, y distinguiendo aquellas celdas que han sido programadas, de las que no lo han sido.

# Tecn. de Prog. EPROM



- ✓ En este caso, no es necesario el fusible.
  - ✓ En su estado no programado, tal como se compra, todas las puertas flotantes están descargadas. O sea, si la línea se activa, se activa el transistor y la columna conduce '0'.
  - ✓ La programación, al cargar la puerta flotante, inhibe la operación del transistor, por lo tanto la columna conduce '1'.
  - ✓ Para descargar esa puerta, se utiliza radiación ultravioleta.
- ✓ Para borrar la EPROM hay que quitarla del circuito.
  - ✓ Problemas: mucho tiempo para ser borradas (20'). Cuanta mas integración, se necesita mas radiación → mas tiempo de exposición.

# Tecn. de Prog. EEPROM y flash



- ✓ Electrically Erasable Programmable ROM
- ✓ Necesita dos transistores, el normal se utiliza para el borrado.
- ✓ Son 2,5 veces mas grandes que los EPROM.

- ✓ FLASH: borran mas rápido que EPROM.
- ✓ Usan diversas arquitecturas, pero todas permiten ser borradas eléctricamente. Estas arquitecturas con similares de las EEPROM.

# Tecn. de Prog. SRAM

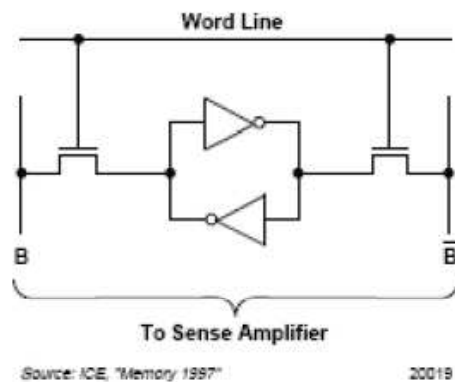
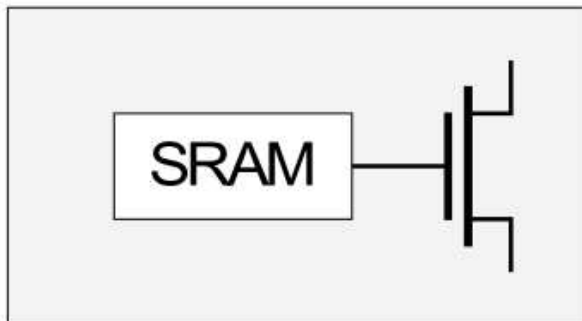




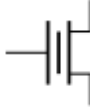
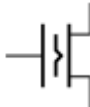

Figure 8-3. SRAM Cell



- ✓ Es un multitransistor formado por 4 a 6 transistores configurados como un latch. Dos de los seis transistores controlan el acceso al latch.
- ✓ Cuando la celda no se direcciona, los dos transistores de control están cerrados y los datos se mantienen dentro del latch.
- ✓ Consumen mucha área
- ✓ Pierden la información cuando dejan de ser alimentados.
- ✓ Pueden ser reprogramados rápidamente y repetidamente.

La tecnología avanza...  
MRAM.... (magnetic RAM) ??

# Resumen de Tecnologías

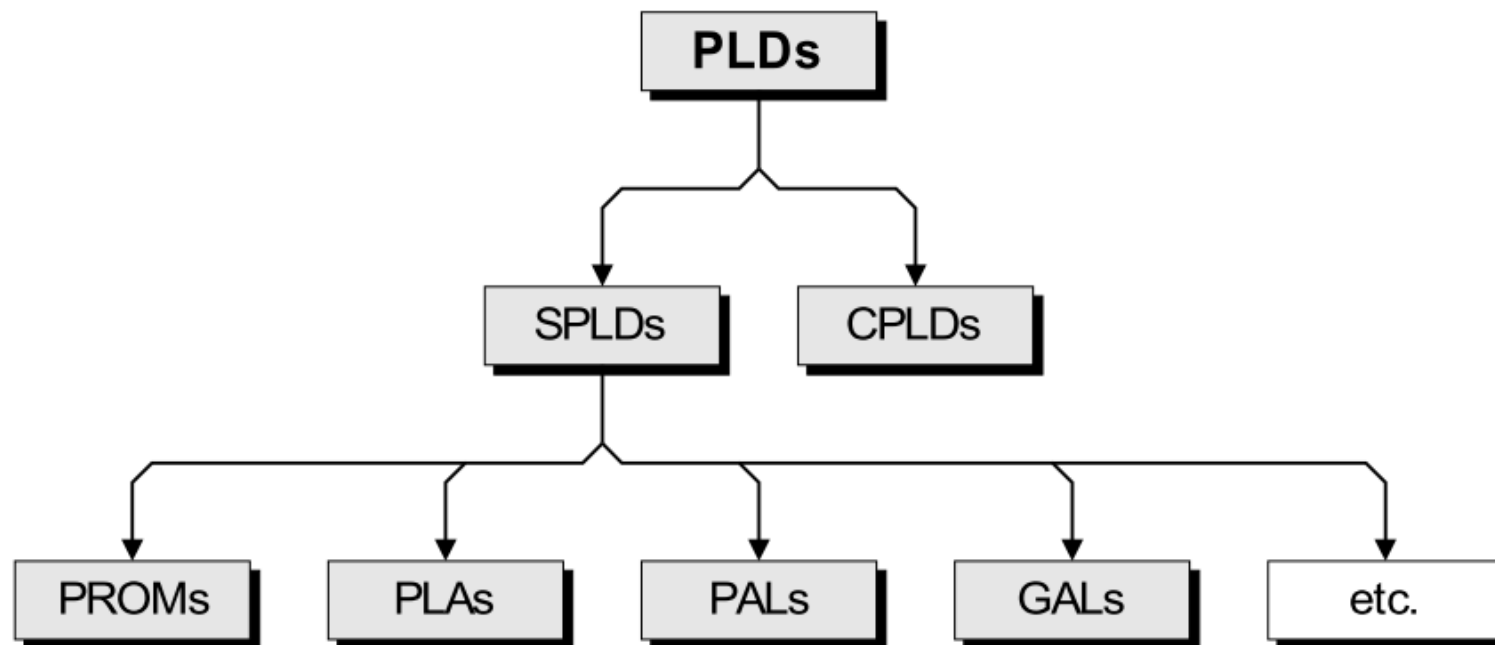
Technology	Symbol	Predominantly associated with ...
Fusible-link		SPLDs
Antifuse		FPGAs
EPROM		SPLDs and CPLDs
E <sup>2</sup> PROM/ FLASH		SPLDs and CPLDs (some FPGAs)
SRAM		FPGAs (some CPLDs)

Feature	SRAM	Antifuse	E2PROM / FLASH
Technology node	State-of-the-art	One or more generations behind	One or more generations behind
Reprogrammable	Yes (in system)	No	Yes (in-system or offline)
Reprogramming speed (inc. erasing)	Fast	----	3x slower than SRAM
Volatile (must be programmed on power-up)	Yes	No	No (but can be if required)
Requires external configuration file	Yes	No	No
Good for prototyping	Yes (very good)	No	Yes (reasonable)
Instant-on	No	Yes	Yes
IP Security	Acceptable (especially when using bitstream encryption)	Very Good	Very Good
Size of configuration cell	Large (six transistors)	Very small	Medium-small (two transistors)
Power consumption	Medium	Low	Medium
Rad Hard	No	Yes	Not really

# Dispositivos Lógicos Programables

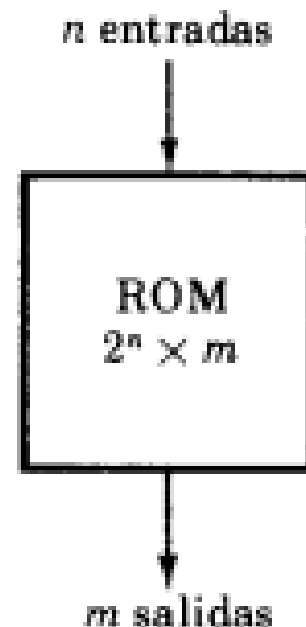
## Programmable Logic Device

- Dispositivos cuya arquitectura interna está predeterminada por el fabricante, pero pueden ser configurados por los ingenieros “en el campo” para realizar una variedad de funciones.



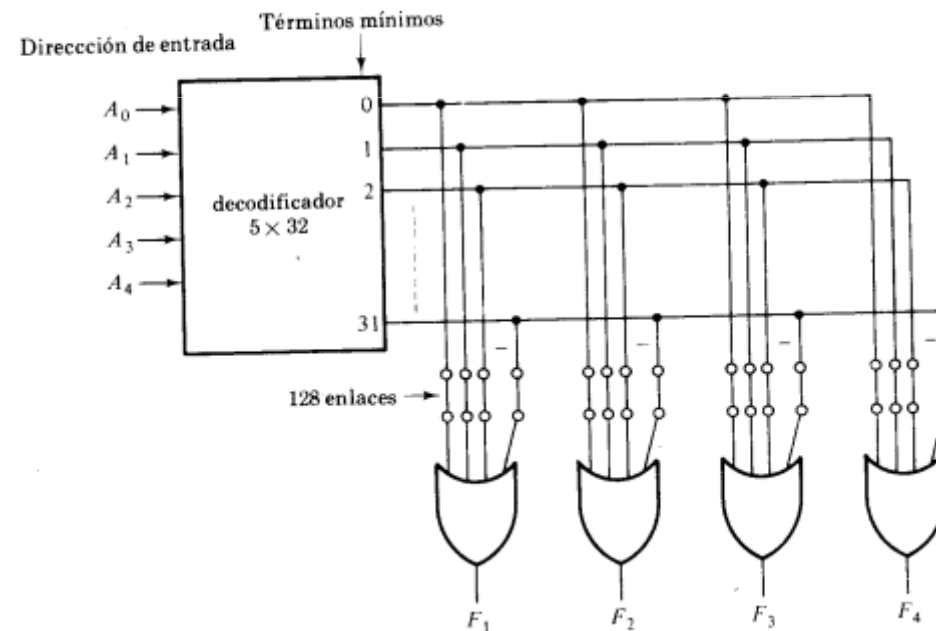
# Disp. Log. Programables elementales

Una memoria de solo lectura (ROM) que viene de Read Only Memory) es un elemento que incluye el decodificador y las compuertas OR dentro de una sola cápsula de CI. Las conexiones entre las salidas del decodificador y las entradas de las compuertas OR pueden especificarse para cada configuración particular “programando” la ROM. La ROM se usa a menudo para configurar un circuito combinacional complejo en una cápsula de CI y así eliminar los cables de conexión.



# ROM.

Una ROM es esencialmente un dispositivo (o acumulador) de memoria en el cual se almacena un conjunto fijo de información binaria. La información binaria debe especificarse por el usuario y luego enclavarse en la unidad para formar el patrón de interconexión requerida. Las ROM vienen con enlaces internos especiales que pueden estar fusionados o abiertos.



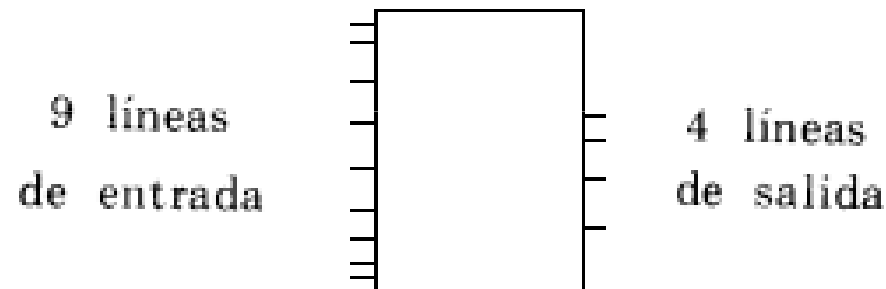
Construcción lógica de una ROM de  $32 \times 4$

Cada combinación de bits de las variables de entrada se llama una *dirección*.

Cada combinación de bits que sale por las líneas de salida se llama una *palabra*

# ROM

Una ROM se especifica algunas veces por el número total de bits que contiene, el cual será  $2^n \times m$ . Por ejemplo, una ROM de 2048 bits puede organizarse como 512 palabras de 4 bits cada una.



$2^9 = 512$  palabras.

total de bits  
 $512 \times 4 = 2.048.$

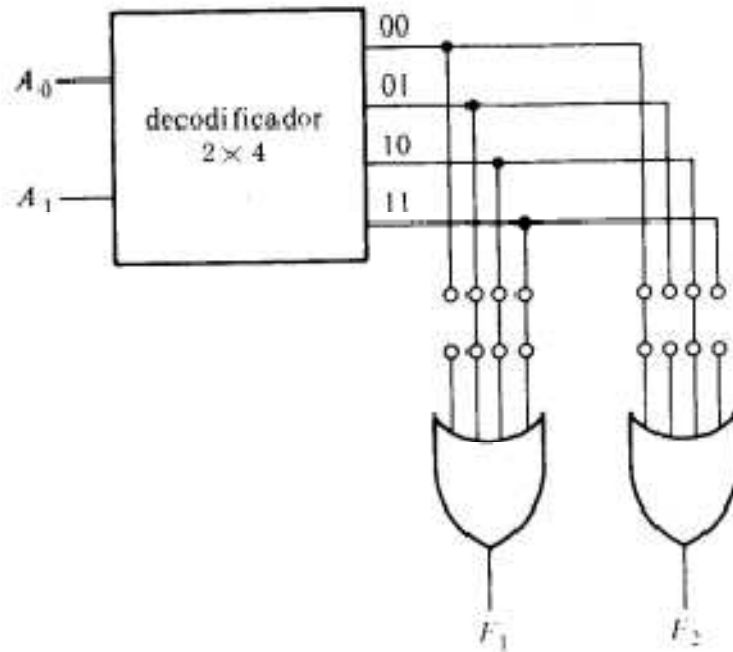
# Ejemplo

$$F_1(A_1, A_0) = \Sigma(1, 2, 3)$$

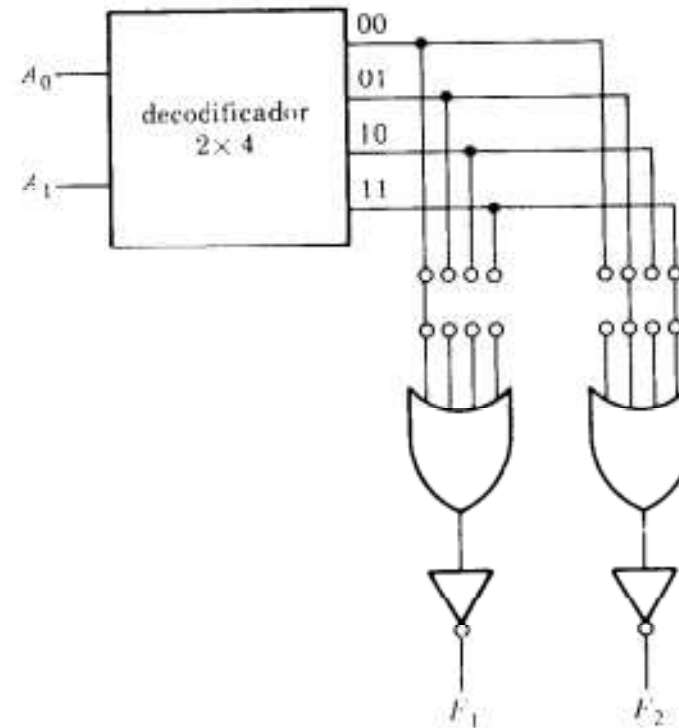
$$F_2(A_1, A_0) = \Sigma(0, 2)$$

$A_1$	$A_0$	$F_1$	$F_2$
0	0	0	1
0	1	1	0
1	0	1	1
1	1	1	0

(a) Tabla de verdad



(b) ROM con compuertas AND-OR



(c) ROM con compuertas AND-OR-invertido

**EJEMPLO** Diseñar un circuito combinacional usando una ROM. El circuito acepta un número de 3 bits y genera un número binario de salida igual al cuadrado del número de entrada.

Entradas			Salidas						Decimal	
$A_2$	$A_1$	$A_0$	$B_5$	$B_4$	$B_3$	$B_2$	$B_1$	$B_0$		
0	0	0	0	0	0	0	0	0	0	0x0
0	0	1	0	0	0	0	0	1	1	1x1
0	1	0	0	0	0	1	0	0	4	2x2
0	1	1	0	0	1	0	0	1	9	3x3
1	0	0	0	1	0	0	0	0	16	4x4
1	0	1	0	1	1	0	0	1	25	5x5
1	1	0	1	0	0	1	0	0	36	6x6
1	1	1	1	1	0	0	0	1	49	7x7

$$1 \times 2^5 + 1 \times 2^4 + 1 \times 2^0 = 49$$

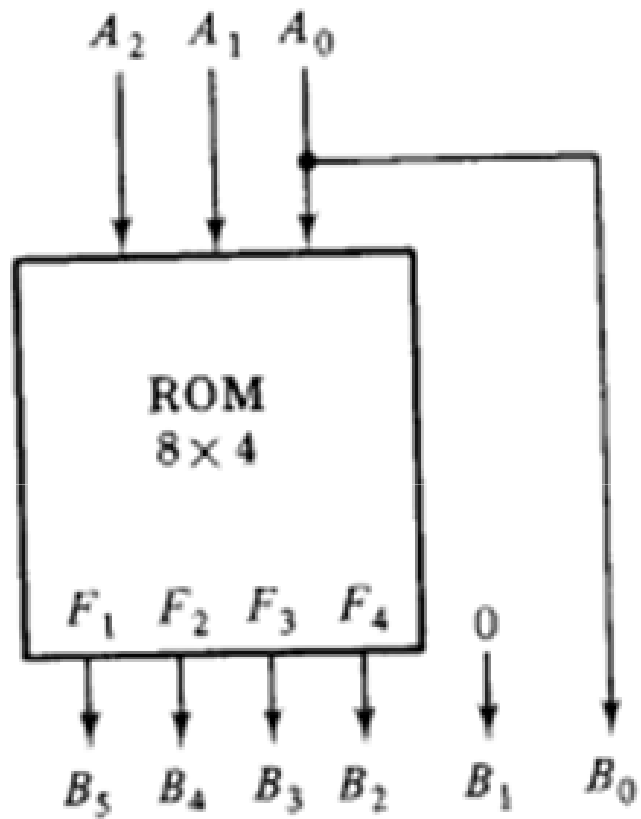
$$32 + 16 + 1$$

Entradas			Salidas					Decimal	
$A_2$	$A_1$	$A_0$	$B_5$	$B_4$	$B_3$	$B_2$	$B_1$	$B_0$	
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1	1
0	1	0	0	0	0	1	0	0	4
0	1	1	0	0	0	0	0	1	9
1	0	0	0	1	0	0	0	0	16
1	0	1	0	1	1	0	0	1	25
1	1	0	1	1	0	1	0	0	36
1	1	1	1	1	0	0	0	1	49

*$B_5$  es siempre igual a la entrada  $A_0$*

Entradas			Salidas						Decimal
$A_2$	$A_1$	$A_0$	$B_5$	$B_4$	$B_3$	$B_2$	$B_1$	$B_0$	
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1	1
0	1	0	0	0	0	0	0	0	4
0	1	1	0	0	0	0	0	1	9
1	0	0	0	1	0	0	0	0	16
1	0	1	0	1	1	0	0	1	25
1	1	0	1	1	0	1	0	0	36
1	1	1	1	1	0	0	0	1	49

*$B_1$  es siempre 0*



$A_2$	$A_1$	$A_0$	$F_1$	$F_2$	$F_3$	$F_4$
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	1
0	1	1	0	0	1	0
1	0	0	0	1	0	0
1	0	1	0	1	1	0
1	1	0	1	0	0	1
1	1	1	1	1	0	0

## Tipos de ROM

<i>programación por máscara</i>	<i>memoria programable de solo lectura o PROM</i>
la hace el fabricante durante el último proceso de fabricación	(de <i>programable read-only memory</i> ).
	usuario programar en su propio laboratorio
	<i>programadores de PROM</i>

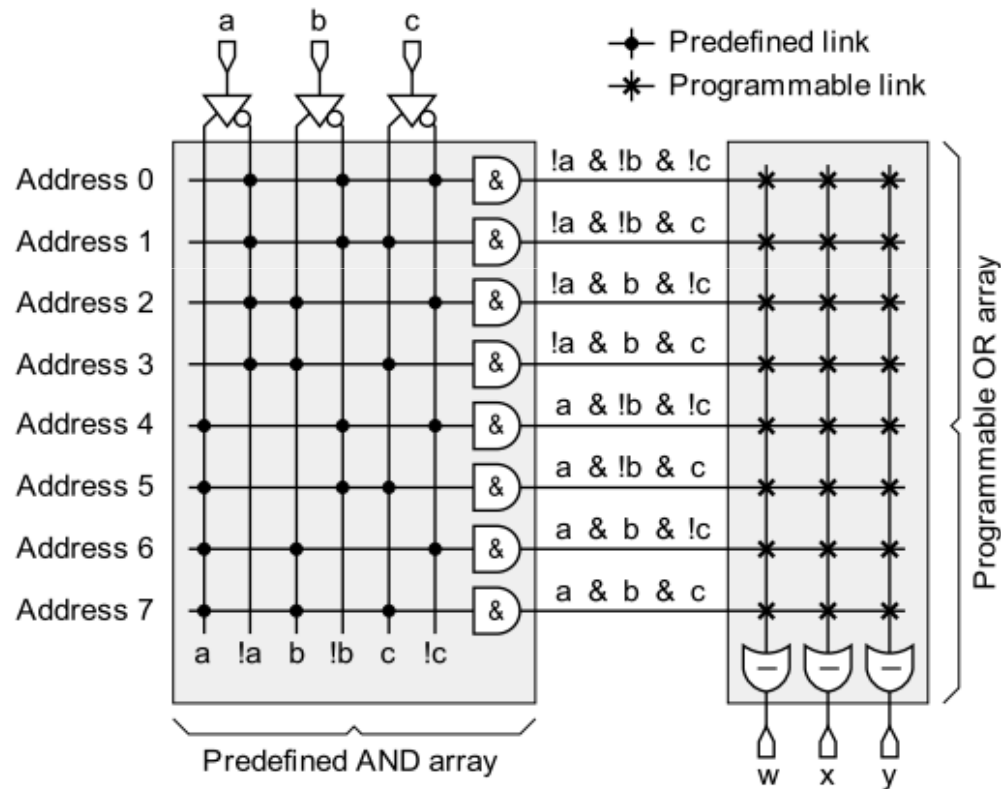
*PROM borrable o EPROM (de erasable PROM).*

recuperadas a su valor inicial  
aunque se hayan cambiado previamente.

## ARREGLO LOGICO PROGRAMABLE (PLA)

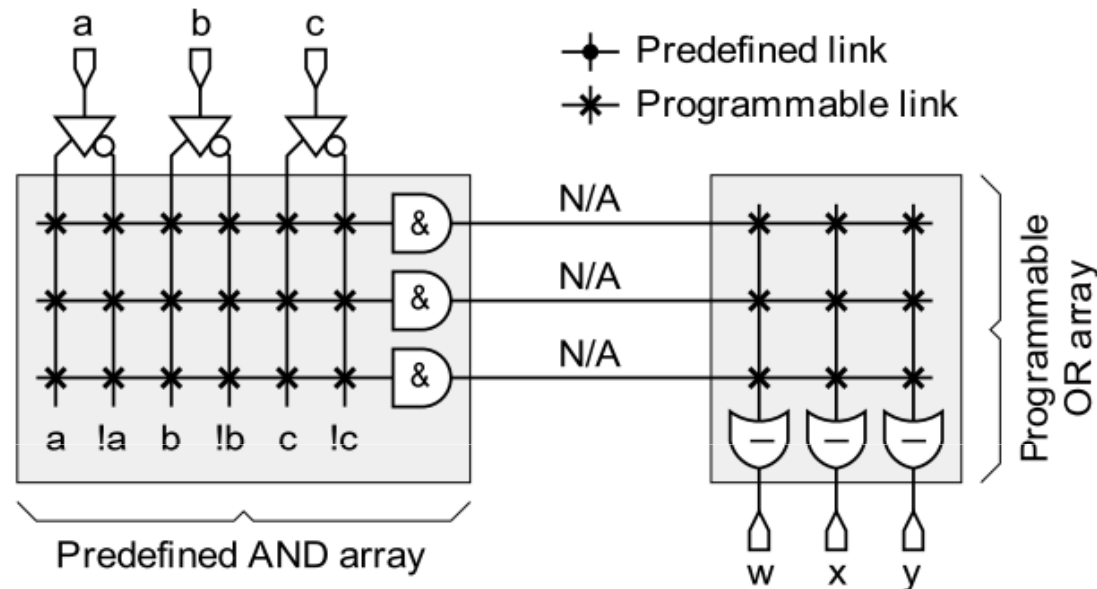
Un PLA es similar a una ROM en concepto; sin embargo el PLA no produce la decodificación completa de las variables y no genera todos los términos mínimos como en una ROM. En un PLA, el decodificador se reemplaza mediante un grupo de compuertas AND, cada una de las cuales pueden ser programadas para generar un término producto de las variables de entrada.

# PROMs (1970)



- ✓ La programación se puede realizar con cualquiera de las tecnologías vistas (fusibles, EPROM, EEPROM).
- ✓ Solo es configurable la matriz OR.
- ✓ Útiles para ecuaciones con pocas entradas y muchos términos producto.

# PLA (Programmable Logic Array)



- ✓ Disponible a partir de 1975, se pueden programar los dos arrays.
- ✓ Se hicieron algunas variantes: arrays AND con arrays NOR. No mucho éxito en el mercado
- ✓ Son útiles cuando diversas funciones usan o comparten términos producto.
- ✓ Son mas lentas que las PROMS

# PLA (...morris)

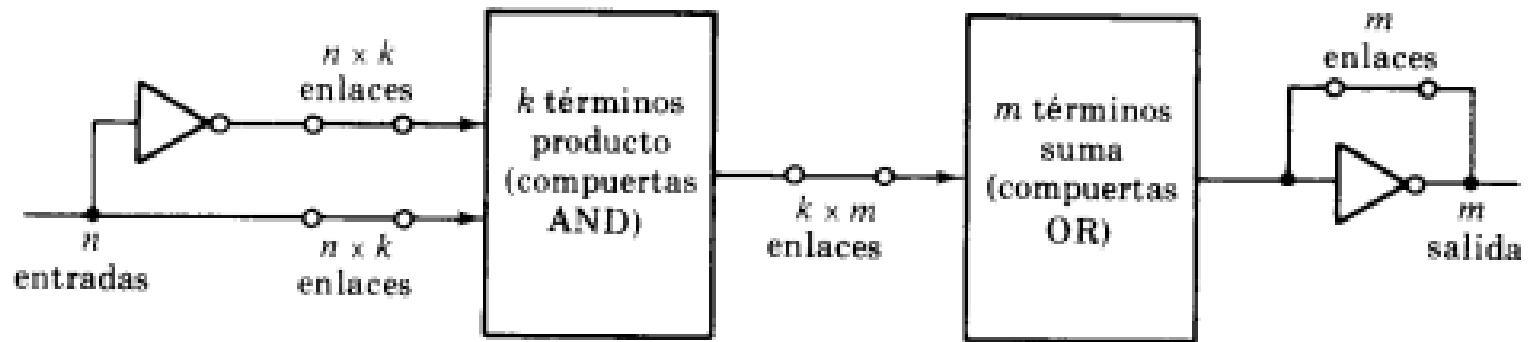
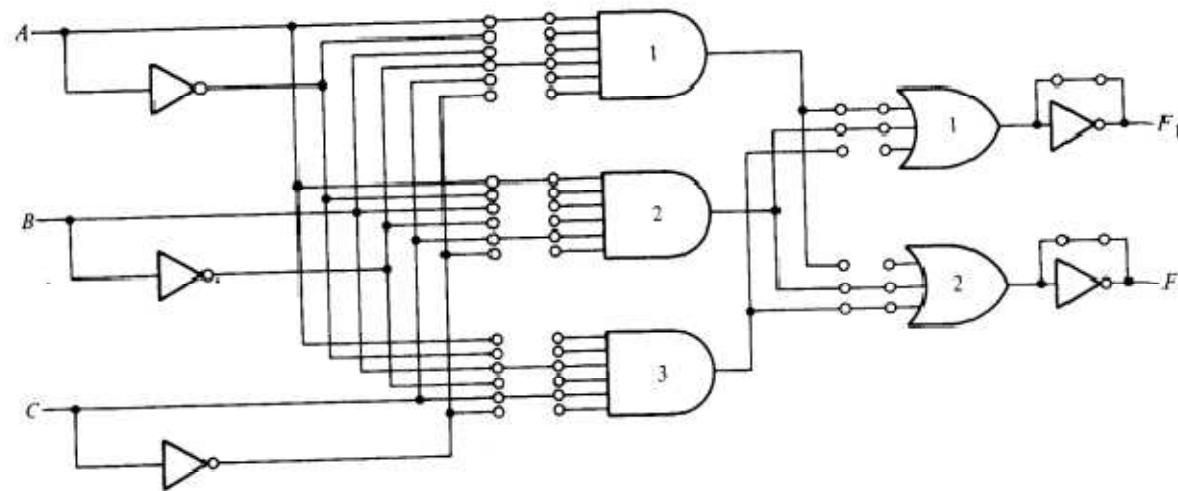


Diagrama de bloque del PLA

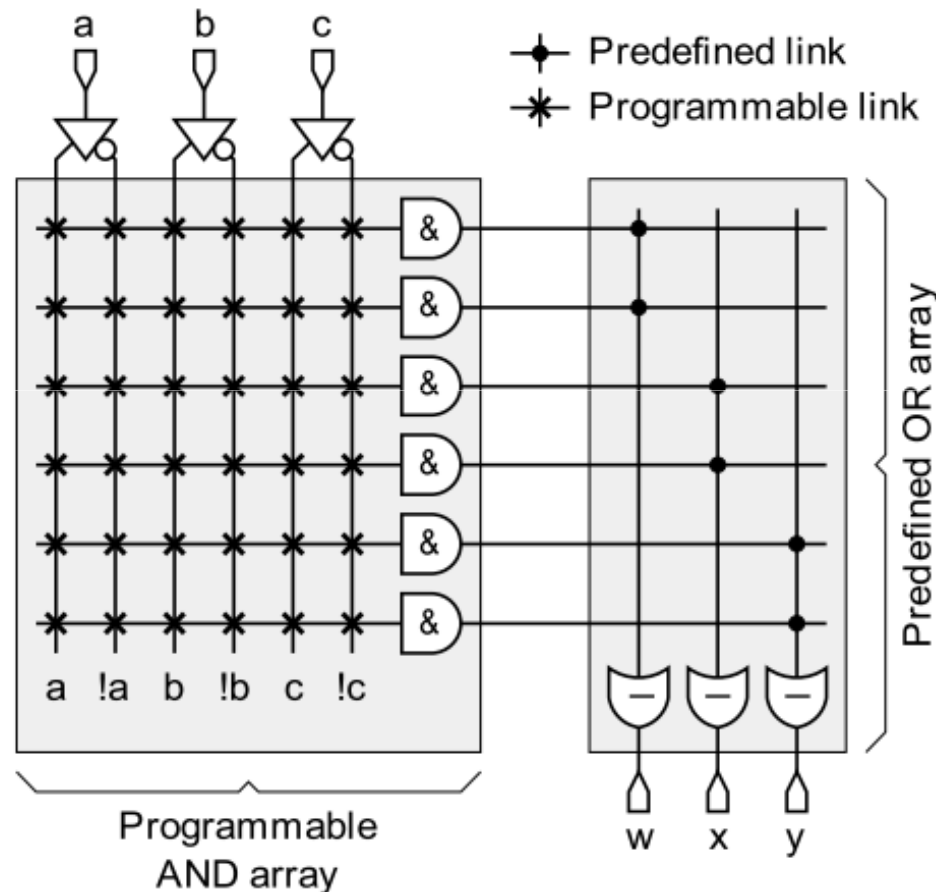


PLA con 3 entradas, 3 términos producto y 2 salidas;

El uso de un PLA debe ser considerado para los circuitos combinacionales que tienen un gran número de entradas y salidas. Es superior a una ROM para circuitos que tienen un gran número de condiciones de no importa.

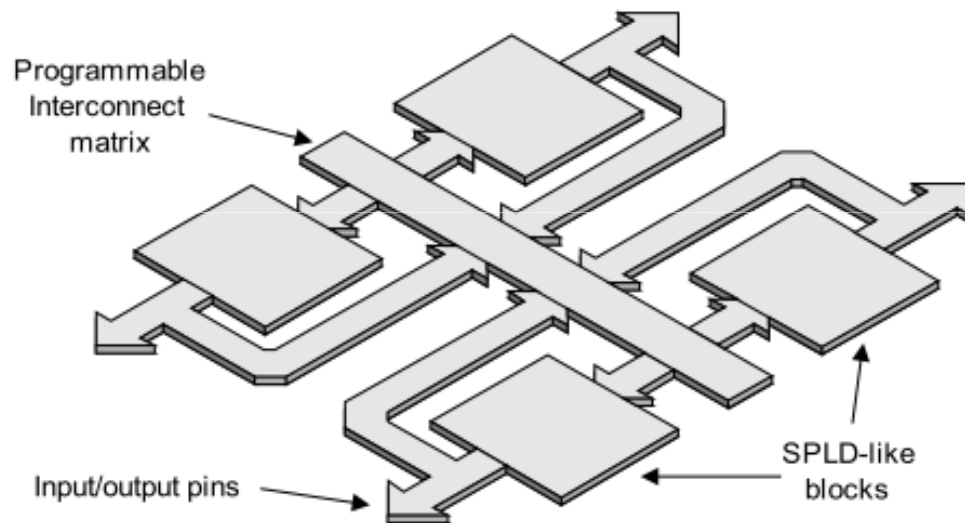
El tamaño del PLA se especifica por el número de entradas, el número de términos de producto y el número de salidas (el número de términos de suma es igual al número de salidas). Un típico PLA tiene 16 entradas, 48 términos producto y 8 salidas. El número de enlaces programados es  $2n \times k + k \times m + m$  mientras que los de la ROM son  $2^n \times m$ .

# PAL (Programmable Array Logic)



- ✓ Al revés de las PROM, la parte programable es la matriz AND
- ✓ Las GAL (Generic Array Logic) son variaciones de las PAL, mas sofisticados (EE)
- ✓ Todos estos dispositivos, aparecen en el mercado con una variedad de opciones: inversión de las salidas, salidas triestado, salidas registradas, etc. Además de tener un número mas grande de entradas y salidas.

# CPLD's

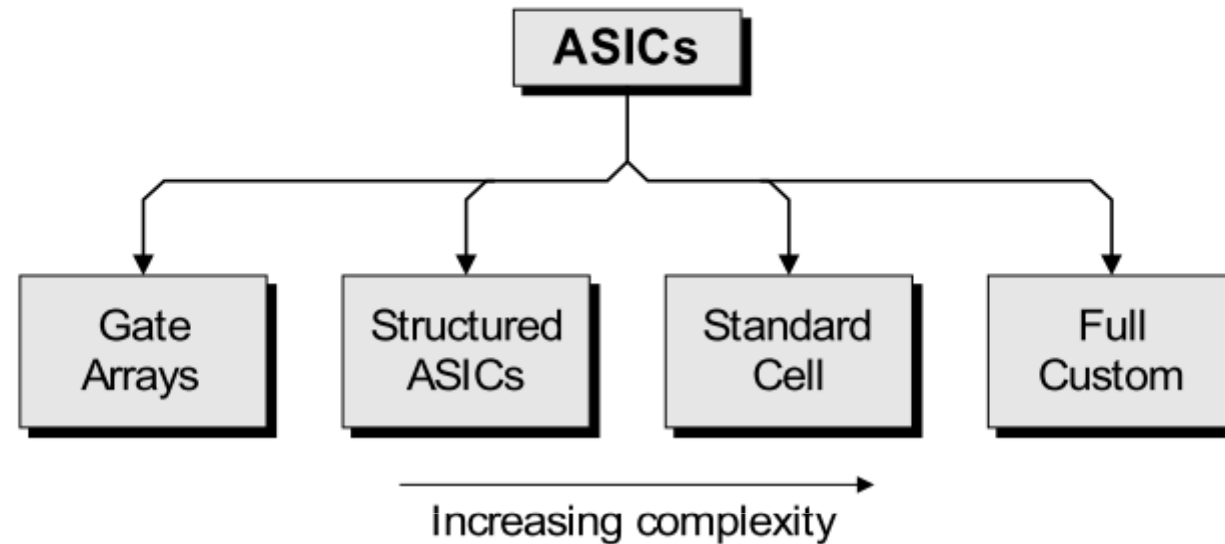


- A finales de los 70, los inventores de la PAL, introducen el Mega-Pal, dispositivo con 4 Standard Pals interconectadas de alguna manera. No funcionó. Consumía mucho.
- 1984: Altera (nueva empresa) introduce el CPLD basado en tecnología CMOS y EPROM.
- Las conexiones entre los bloques se programan mediante la matriz de interconexión.

# ASIC

- Application Specific IC
  - Diseñado para una función específica.
  - Contienen cientos de millones de puertas lógicas y pueden ser usados para crear funciones complejas.
  - El proceso de diseño y construcción de un ASIC es largo y caro, y finaliza en su realización en silicio.
  - No puede ser usado ni testeado antes de su fabricación.

# ASICs

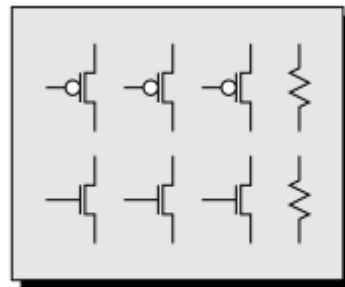


- ASIC: Es un chip (circuito integrado, IC) diseñado para una determinada aplicación y para una determinada compañía.
- Full custom: hecho enteramente por encargo (a medida): desde componentes pequeños, a microprocesadores diseñados y fabricados para una compañía específica.

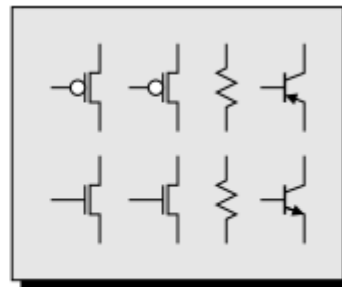
# Fabricación de un IC

- Los transistores y sus conexiones se construyen mediante muchas capas (typical 10 to 15 in CMOS) puestas unas sobre las otra
- Cada capa tiene una forma especial definida por una máscara. Algunas de las capas o niveles forman transistores, otras los planos de conexión.
- Un aspecto importante de un IC es el tamaño del mas pequeño transistor que puede ser fabricado:
  - Este es medido en micrones ( $\mu\text{m}$ ,  $10^{-6}$  meter)
  - Por ejemplo, decimos que un IC está construido con un proceso de  $0.50 \mu\text{m}$
  - Tal cual como profetizó Moore, el proceso continúa mejorando, o sea haciendose mas pequeño....
  - En este momento, el proceso de miniaturización es menor que  $0.1 \mu\text{m}$  (deep sub-micron)

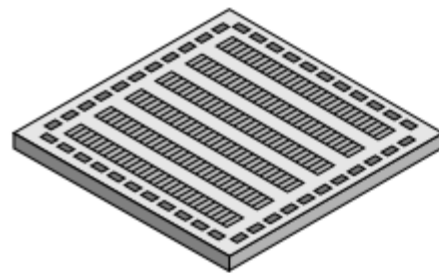
# Gate Array



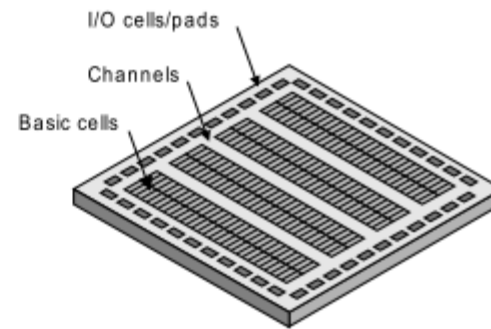
(a) Pure CMOS basic cell



(b) BiCMOS basic cell



(a) Single-column arrays



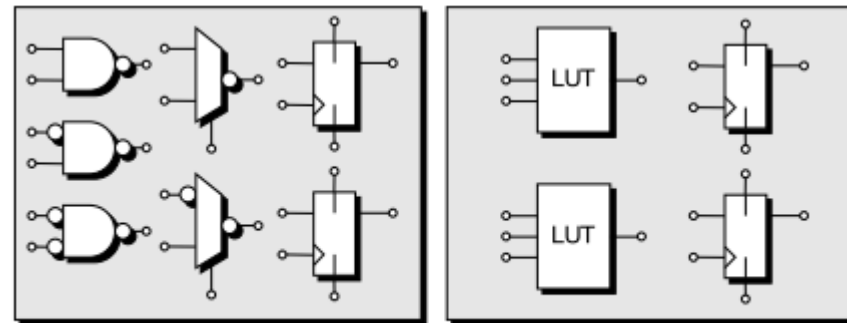
(b) Dual-column arrays

- Gate Array: (1975) basado en la idea de celdas básicas formadas por transistores y resistencias sin conexión.
- Cada fabricante de ASIC determina que incluir en una celda básica, y construye chips presiliconados formados por arrays de celdas. (sea of cells).
- Los fabricantes definen una librería funciones lógicas (puertas primitivas, multiplexores, y registros) que son los que usan los ingenieros de la aplicación.
- Los ingenieros diseñan hasta llegar a nivel de netlist. Luego se hace el mapeo, ubicación y routing con las herramientas provistas por el fabricante.
- El resultado de este proceso son las máscaras con las cuales se crean los niveles de metalización que unirán las celdas básicas entre sí, así como los componentes dentro de las celdas básicas.

# Structured ASIC

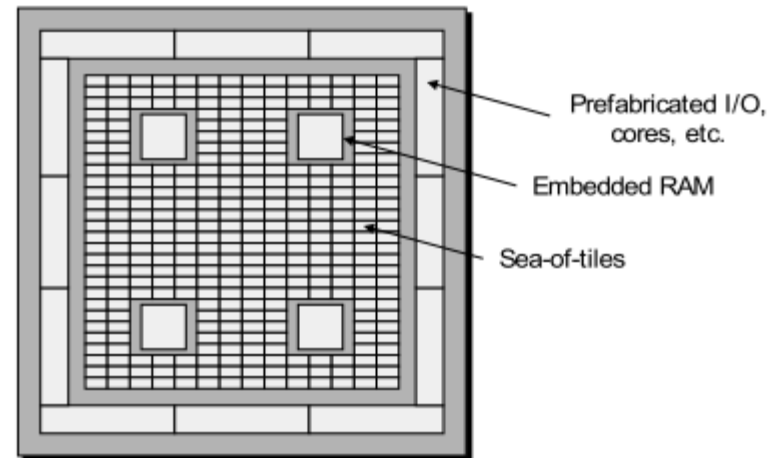
- (2002) Cada vendedor tiene su arquitectura.
- Cada dispositivo está formado por elemento básico llamado módulo (tiles) que contiene una mezcla de lógica prefabricada (multiplexores, puertas, lookup table) junto con uno o mas registros y posiblemente algo de RAM
- Un array (sea) de estos elementos se prefabrica sobre la superficie del chip. Además, en los bordes de este mar de "tiles" (tejas, baldosas, ladrillo) hay bloques de RAM, generadores de reloj, etc.
- Cada dispositivo se "particulariza" mediante niveles de metalización, aunque muchos de estos niveles ya están también predefinidos. Solo 2 o 3 niveles se deben aplicar. Reducción de costos.
- Consumen mas que un standard cell. También ocupan mas. (dos o tres veces mas).

## Structured ASIC tiles



(a) Gate, mux, and flop-based

(b) LUT and flop-based



Generic structured ASIC

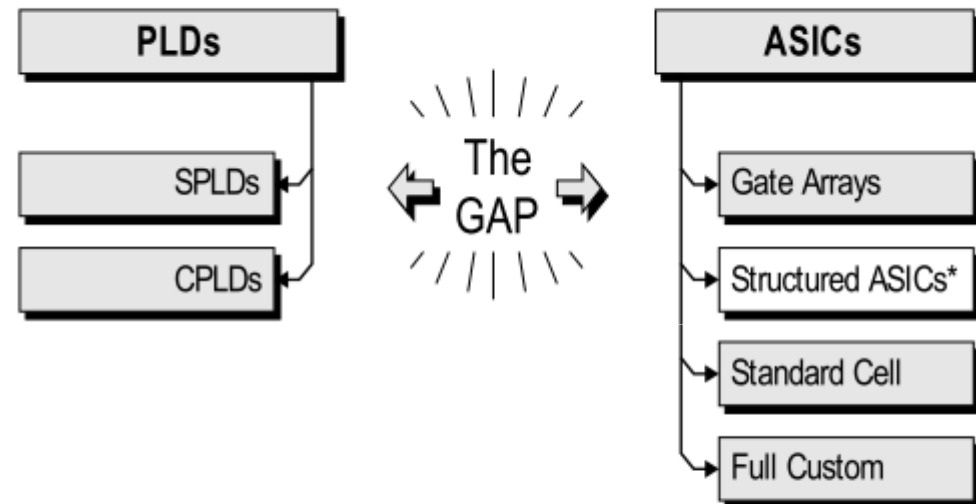
# Standard Cell

- Como en el caso de Gate Arrays, el fabricante define un conjunto de bloques básicos (multiplexores, registros, puertas, etc) que ofrece al ingeniero en forma de librerías.
- También ofrece librerías que pueden incluir microprocesadores, elementos de comunicación, funciones de ROM y RAM.
- Además hay IP que los ingenieros pueden reusar.
- IP: Intellectual Property: bloques funcionales creados por algun otro. Se compran.
- Los ingenieros, con todos esos elementos hacen el diseño hasta llegar a nivel de netlist, que describe las puertas lógicas que usarán y sus conexiones. Las herramientas de diseño son, incluso hoy, muy sofisticadas)
- La diferencia con las Gate Arrays es que no hay nada prefabricado. Cada función se crea con el mínimo número de transistores necesarios, sin componentes redundantes.
- Mas eficiente uso del silicio que Gate Arrays.

# Circuitos Dedicados (full custom)

- Los ingenieros tienen el control completo sobre cada una de las máscaras usadas para fabricar el chip.
- El vendedor del ASIC no prefabrica ningún componente en el silicio y no provee ninguna librería ni puertas predefinidas.
- Por medio de las herramientas apropiadas, los ingenieros pueden modelar “a mano” las dimensiones de los transistores y pueden crear sus propias funciones basados en estos transistores. Incluso, las propias herramientas con las que ellos hacen estas cosas son diseñadas por ellos.
- El proceso es altamente complejo, y consume mucho tiempo, pero el chip resultante contiene la máxima cantidad de lógica con el mínimo desperdicio de silicio.

- Hacia 1980, es evidente que hay un GAP entre el mundo de los IC.
- Por un lado, los dispositivos programables, muy sencillos pero muy configurables. Por el otro, los ASIC's, soportando funciones complejas, pero muy caros, y muy costosos en tiempo de diseñar. Además, una vez el diseño estaba hecho, quedaba congelado en el silicio.
- Para salvar ese GAP, Xilinx lanza al mercado en 1984, una nueva clase de IC: FPGA.



\*Not available circa early 1980s