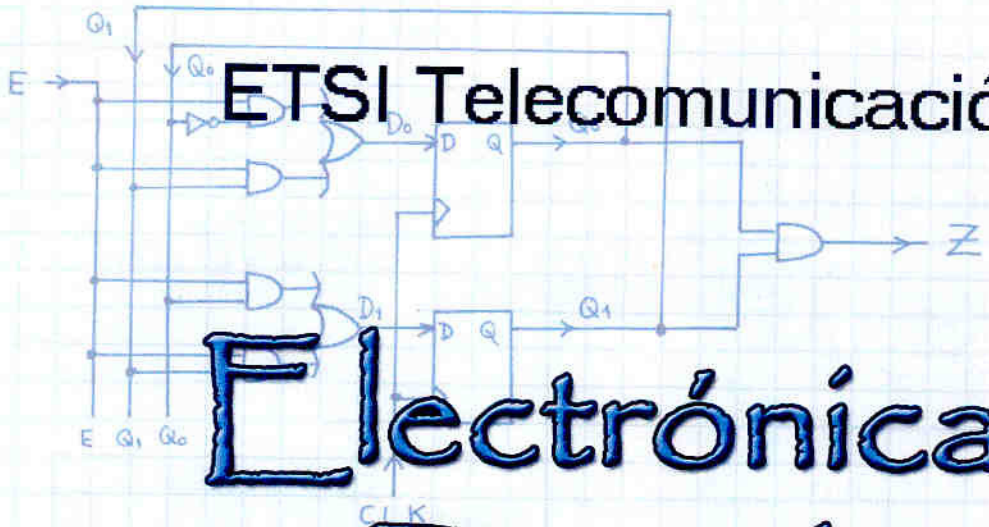
5. Implementarlo (solución Moore)



• por MEALY

1. Diagrama de estados

Asignación de estados
Espe = 00
1 uno = 01
2 unos = 11

3 estados → 2 FF

fijate que con Mealy necesitamos un estado menos que con Moore

2. Tabla de estados

la asignación de estados es a gusto de cada cual, aunque a veces elegir cierta asignación facilita el problema.

E.actual	E=0	E=1
00	00/0	01/0
01	00/0	11/0
11	00/0	11/1

3. Tabla de funcionamiento global

Entradas			E.sig ≡ Excitaciones		Salida
E	E.actual Q_1	E.actual Q_0	$Q_1 \equiv D_1$	$Q_0 \equiv D_0$	Z
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	X	X	X
0	1	1	0	0	0
1	0	0	0	1	0
1	0	1	1	1	0
1	1	0	X	X	X
1	1	1	1	1	1

nota; si el FF no fuera tipo D, el estado siguiente no coincidiría con las excitaciones. Habría q obtenerlas con tabla de excit.

← estados no esperados, al poner X estamos haciendo implementación mínimo coste

4. Karnaugh

	D_1		D_0		Z	
Q_1Q_0 \ E	0	1	0	1	0	1
0 0	0	0	0	1	0	0
0 1	0	1	0	1	0	0
1 1	0	1	0	1	0	X

$D_1 = E \cdot Q_0$
$D_0 = E$
$Z = E \cdot Q_1$
↳ es Mealy

Electrónica Digital

Apuntes de Pak (Francisco José Rodríguez Fortuño)
ETSI Telecomunicación. Universidad Politécnica de Valencia.
Primer cuatrimestre de 2º curso
Curso 2004/2005

Contenido

- Referencia rápida
- Apuntes de la asignatura.

Fecha de última actualización: 27 Agosto 2007

TEMA 2: Circuitos Lógicos

$F(A,B,C)$

$F = \sum(1,2,3,7)$

$F = \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC + ABC$ } minitérminos canónicos

$\bar{F} = \sum(4,5,6,0)$

$F = \Pi(0,4,5,6)$

$= (A+B+C)\cdot(\bar{A}+B+C)\cdot(\bar{A}+B+\bar{C})\cdot(\bar{A}+\bar{B}+C)$ } maxiterminos canónicos

$0 = 000 \quad 4 = 100 \quad 5 = 101 \quad 6 = 110$

XOR

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0

coincide con la suma binaria (sin contar con acarreo)

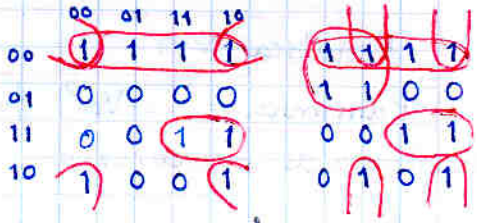
de Morgan

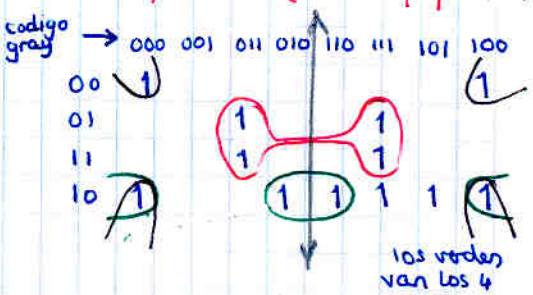
$\overline{A\cdot B\cdot C\cdots} = \bar{A} + \bar{B} + \bar{C} + \cdots$

$\overline{A+B+C+\cdots} = \bar{A}\cdot\bar{B}\cdot\bar{C}\cdot\bar{D}$

permiten cualquier función con NAND o NOR

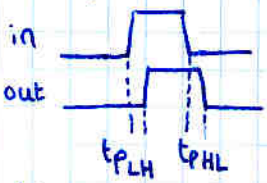
Karnaugh sólo agrupar potencias de 2





TEMA 3- FAMILIAS LÓGICAS

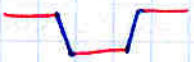
Fan-out: puertas a las que puede atacar



tiempos propagación

$\text{slew rate} = \dfrac{(V_{OH} - V_{OL})\cdot 80\%}{t_f}$ (es la pendiente)

$P = P_{dinamic} + P_{static}$



$P_{tconsumida} = V_{cc}\cdot(I_{R_1} + I_{R_2} + \ldots)$ lo que sacamos de la batería

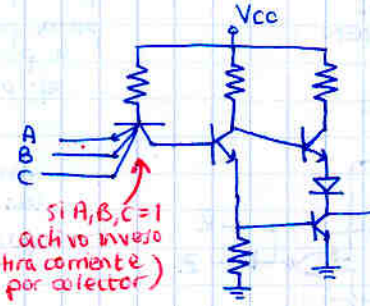
$P_{tdisipada} = I_{R_1}R_1^2 + I_{R_2}R_2^2 + \ldots$

Open collector ◇

Triestado: ▽ una entrada más I
- I=1 → puerta normal
- I=0 → salida ALTA IMPEDANCIA (los dos T del totem pole en corte)

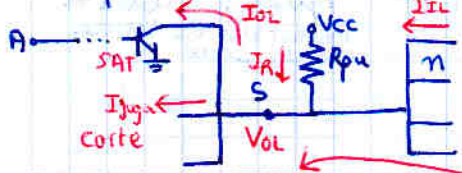
varias open collector conectadas con Rpu

TTL



S = 0: Rmin para que no se quemen las ◇

Caso peor: sólo una puerta en 0 (sat)



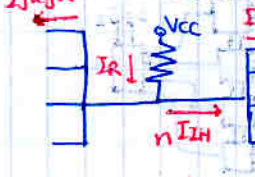
$R_{min} < R_{pu} < R_{max}$

más rápido ↔ menos consuma

$I_R + nI_{IL} < I_{OLmax} \quad (+(p-1)I_{juga})$

siendo $I_R = \dfrac{V_{cc} - V_{OL}}{R}$

S = 1: Rmax proporcionas suficiente corriente y tensión



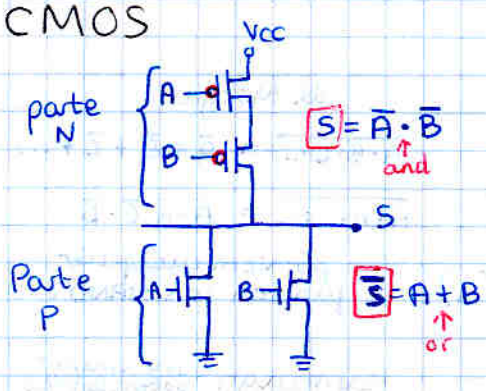
$V_S > V_{IHmin}$

$I_R = nI_{IH} \quad (+pI_{juga})$

siendo

$V_S = V_{cc} - nI_{IH}R_{pu}$

CMOS



parte P y parte N siempre son complementarias

normalmente hay 4 posibilidades

parte N → F por unos
→ F por ceros

parte P → F̄ por unos negando
→ F̄ por ceros negando

AB \ CD	00	01	11	10
00	1	0	1	0
01	0	0	0	0
11	0	0	0	0
10	0	0	0	0

$F = \bar{A}\bar{B}(\bar{C}\bar{D} + CD)$

$F = \bar{A}\bar{B}(C+\bar{D})(\bar{C}+D)$

$\bar{F} = A + B + (C+D)(\bar{C}+\bar{D})$

$\bar{F} = A + B + \bar{C}D + C\bar{D}$

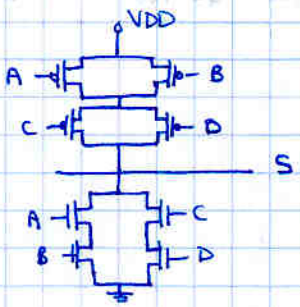
$P_{estatica} \simeq 0$

$P_{dinamica} = C_L \cdot V_{DD}^2 \cdot f_0$

ejemplo:

$Z = \bar{A}\bar{B} \cdot \bar{C}\bar{D} = (\bar{A} + \bar{B}) \cdot (\bar{C} + \bar{D})$ (OR, AND, OR)

$\bar{Z} = AB + CD$



Problemas de interconexión

Problema:
Queremos conectar un TTL estandar salida totem-pole a 5 inversores CMOS. Queremos que $\Delta 1 = 1$ V

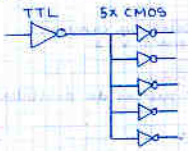
Datos:

TTL (Vcc = 5V)	CMOS (V_{DD} = 5V)
$I_{OH\,max} = -0'4$ mA	$I_{IH\,max} = 5$ nA
$I_{OL\,max} = 16$ mA	$I_{IL\,max} = -5$ nA
$V_{OH\,min} = 2'4$ V	$V_{IL\,max} = 1'5$ V
$V_{OL\,max} = 0'5$ V	$V_{IH\,min} = 3'5$ V

negativo = saliente

Vamos a probar de tres formas

a) Directo

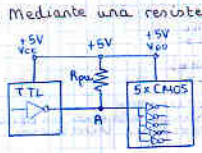


"0": $V_{OL\,max} < V_{IL\,max}$ SI; $I_{OL\,max} > 5 \cdot I_{IL\,max}$ SI

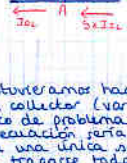
"1": $V_{OH\,min} > V_{IH\,min}$ NO!; $I_{OH\,max} > 5 \cdot I_{IH\,max}$ SI

No es posible la interconexión directa. Si lo fuera, deberíamos comprobar los márgenes de ruido.

b) Mediante una resistencia pull-up



"0":



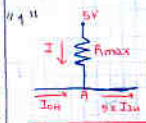
$I + nI_{IL} \leq I_{OL\,max}$

$\frac{5 - V_A}{R_{min}} + 5I_{IL} \leq I_{OL\,max}$

$R_{min} \geq \frac{5 - V_A}{I_{OL\,max} - 5I_{IL}}$

$R_{min} = 0'28$ kΩ

"1":



$V_A \geq V_{IH\,min} + \Delta 1$

$5 - IR_{max} \geq V_{IH\,min} + \Delta 1$

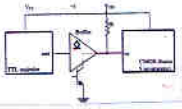
$R_{max} \leq \frac{5 - (V_{IH\,min} + \Delta 1)}{n \cdot I_{IH} - I_{OH}}$

$R_{max} = 20000$ kΩ

c) con un buffer TTL open-collector

$I_{OH\,max} = 0'25$ mA (fugas)

$I_{OL\,max} = 16$ mA



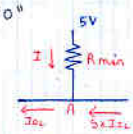
Conexión TTL → TTL-OC

- los niveles de tensión se cumplen
- los niveles de corriente se cumplen (con fanout 1 no hay problema)

Conexión Buffer TTL (open-collector) → CMOS

En este caso R_{pu} es obligatorio (como en toda puerta open-collector)

"0"



para el caso "0", es exactamente igual que en el apartado anterior, pues en el caso "0" una TTL-OC es exactamente igual que una TTL-totem pole.

$I + nI_{IL} \leq I_{OL\,max}$

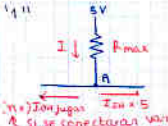
$\frac{5 - V_A}{R_{min}} + nI_{IL} \leq I_{OL\,max}$

$R_{min} = 0'28$ kΩ

Si estuviéramos haciendo uso del cableado lógico de open collector (varias salidas OC) lo cual es un caso típico de problema de examen, la ecuación sería la misma, pues el peor caso es que una única salida open collector esté en "0" y tenga que tragarse toda la corriente (sólo cambiaría por las fugas de las demás salidas)

NOTA: la corriente de fugas $I_{OH\,max}$ es: entrante: salidas open-collector; saliente: salidas totem-pole. En lógico ni recuerdan el funcionamiento

"1"



$V_A \geq V_{IH\,min} + \Delta 1$

$5 - R_{max} I \geq V_{IH\,min} + \Delta 1$ (I = (n·)$I_{OH\,fugas} + 5I_{IH}$)

$R_{max} \leq \frac{5 - (V_{IH\,min} + \Delta 1)}{nI_{IH} + I_{OH}}$

$R_{max} = 2$ kΩ

Si se conectaran varias salidas OC (cableado lógico)

TEMA 4. Circuitos combinacionales

Paridad:

Comparadores: (Less, equal, greater)

para palabras 1 bit: hacer tabla de verdad

A_0 B_0	L E G
0 0	0 1 0
0 1	1 0 0
1 0	0 0 1
1 1	0 1 0

$L = \bar{A}_0 B_0$

$E = A_0 B_0 + \bar{A}_0 \bar{B}_0$

$G = A_0 \bar{B}_0$

para palabras de n bits, hacerlo lógicamente a partir de 1 bit:

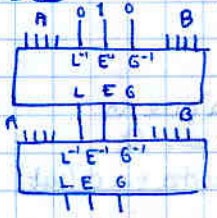
ej: palabras 4 bit:

$G(A>B) = G_3 + E_3 G_2 + E_3 E_2 G_1 + E_3 E_2 E_1 G_0$

$E(A=B) = E_3 E_2 E_1 E_0$

$L(A<B) = L_3 + E_3 L_2 + E_3 E_2 L_1 + E_3 E_2 E_1 L_0$

normalmente en comparador de 4 bits cuando E(A=B) las salidas L, E, G serán la entrada L^{-1}, E^{-1}, G^{-1} para que sea cascadable

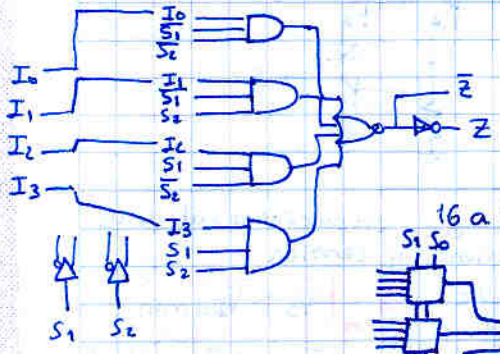


} bits de menor peso

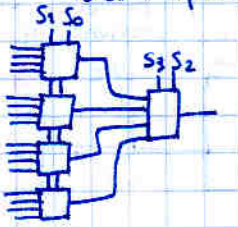
} bits de mayor peso

así cuando los de mayor peso sean iguales, saldrá el resultado de los de menor peso

Multiplexores



16 a 1 a partir de 4 a 1

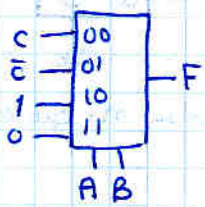


Implementación de funciones con MUX

con multiplexor de n variables control, función de n+1 variables

	A	B	C	F	
00	0	0	0	0	F = C
	0	0	1	1	
01	0	1	0	1	F = $\bar{C}$
	0	1	1	0	
10	1	0	0	1	F = 1
	1	0	1	1	
11	1	1	0	0	F = 0
	1	1	1	0	

se eligen 2 que sean las de control
ej A y B



algunos trucos:

- se puede obtener la salida negada sin más que negar cada entrada
  ⇒ si piden $F = \sum m(0,3,4,5,6,7,9,11,12,13,14,15)$
  mejor hacer $\bar{F} = \sum m(1,2,8,10)$
  y luego negar cada entrada

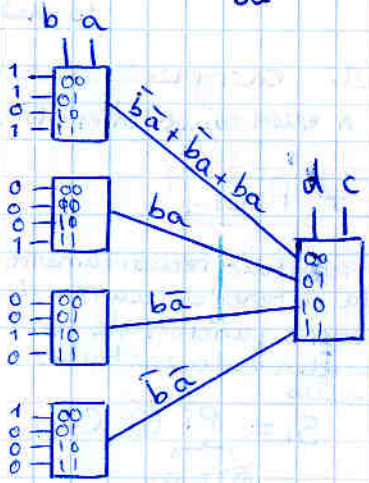
con varios MUX
truco: empezar por el final

$F = \bar{d}\bar{c}\bar{b}\bar{a} + \bar{d}\bar{c}ba + \bar{d}c\bar{b}a + \bar{d}cb\bar{a} + d\bar{c}\bar{b}\bar{a} + dcb\bar{a}$

elejimos dc como variables de control

$F = \bar{d}\bar{c}(\bar{b}\bar{a} + \bar{b}a + ba) + \bar{d}c(ba) + d\bar{c}(b\bar{a}) + dc(\bar{b}\bar{a})$

dc: 00 → $\bar{b}\bar{a} + \bar{b}a + ba$; 01 → ba; 10 → $b\bar{a}$; 11 → $\bar{b}\bar{a}$



Codificadores

ej decimal a BCD

9 entradas: la salida será la codificación BCD de la entrada activa

- con prioridad: si hay varias entradas activas hará caso a cierta prioridad (normalmente I_9 más prioritario, luego I_8, luego I_7, ...)
- sin prioridad: no se contemplan casos de pulsar varias entradas → salen cosas absurdas.

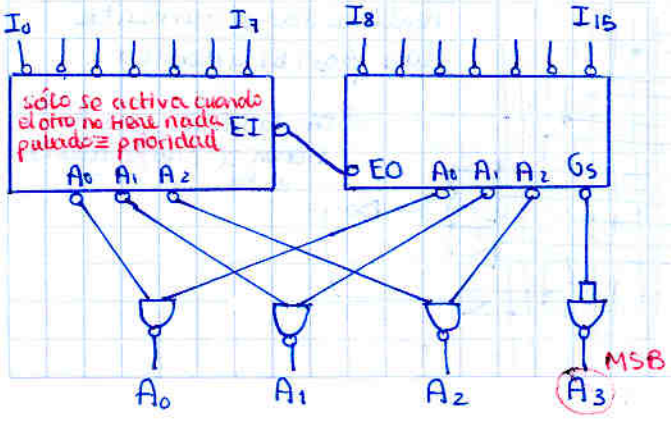
ej comercial

Decimal 7 entradas → Binario 3 salidas

además EO → activo cuando no se pulsa nada
GS → activo cuando se pulsa algo

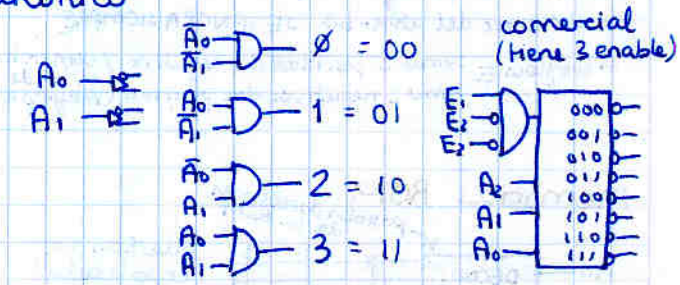
EI → enable input

permite construir codificador 16 a 4



Decodificadores

muy sencillos. cada salida es un miniterino canónico



- Aplicaciones:
  demultiplexor: → A_2, A_1, A_0 líneas de control
  ↳ enable negado es el dato

funciones lógicas:
- cada salida es un miniterino canónico negado de las variables a la entrada

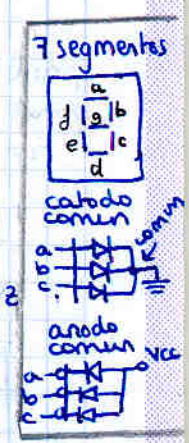
puede venir muy bien.

ej. un MUX necesita para implementar una función

wy: 00 → 0, 01 → 0, 10 → 0, 11 → $\bar{x}\bar{z} + x\bar{z} + xz$

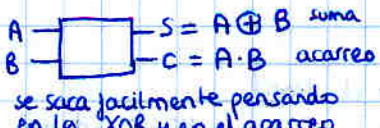
TRUCO $\bar{x}\bar{z} + x\bar{z} + xz = \overline{\bar{x}\bar{z}}$

esto se saca muy fácilmente de un decodificador



Decodificador BCD → 7 segmentos
RBI: si está activo y entra 0000 → apaga LEDs (ceros a la izq)
RBO: si está activo RBI y entra 0000 → RBO se activa (permite apagar ceros a la izquierda)

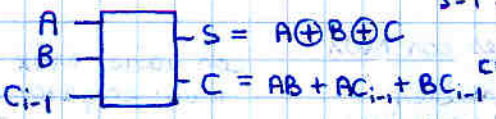
Half adder



se saca fácilmente pensando en la XOR y en el acarreo

0+0 = 0 acarreo 0
0+1 = 1 acarreo 0
1+0 = 1 acarreo 0
1+1 = 0 acarreo 1

Full adder

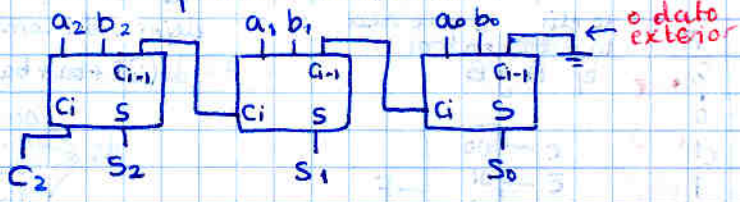


S=1 si hay uno de los tres ó los tres a la vez en 1

C=1 cuando hay al menos dos entradas a uno.

es todo lógico

permite cascadar



lo malo es que tarda n · tpd en dar la suma

Acarreo anticipado. Carry look ahead

el acarreo del bit n existe cuando en este bit se ha sumado 1+1 o cuando se ha sumado 1 + el acarreo n-1

$$C_i = G_i + P_i C_{i-1}$$

generación: $G_i = A_i \cdot B_i$

propagación: $P_i C_{i-1} = (A \oplus B) C_{i-1}$

no hay mas que sustituir recursivamente un C_i en otro para obtener el acarreo de cada bit sólo como función de las entradas → se pueden calcular todos a la vez en paralelo y luego la suma

$$C_1 = G_1 + P_1 C_0$$
$$C_2 = G_2 + P_2 C_1 + P_2 P_1 C_0$$
$$C_3 = \dots$$

$$S_i = \underbrace{P_i}_{A_i \oplus B_i} \oplus C_{i-1}$$

Desventajas: mas caro, mas grande, consume mas, necesita puertas de n entradas

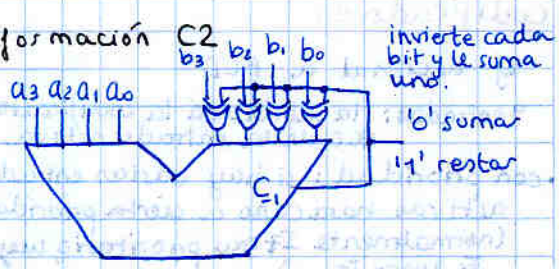
Restador:

Es un sumador pero con complemento a 2.

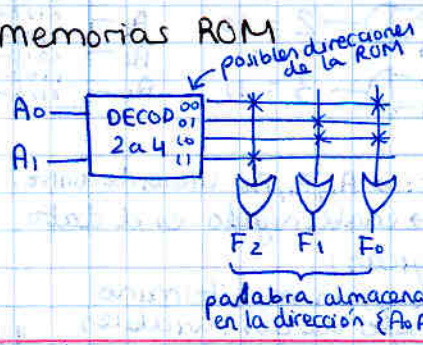
↳ el acarreo del último bit SE IGNORA SIEMPRE

↳ overflow = sumar 2 positivos da negativo / sumar 2 negativos da positivo (comprobar el bit de mayor peso)

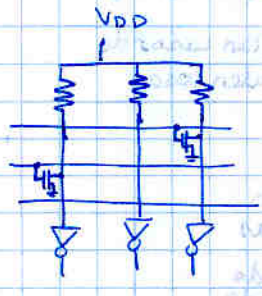
transformación C2



Memorias ROM



puertas OR cableadas para no tener puertas de n entradas



PROM: se programa quemando fusibles

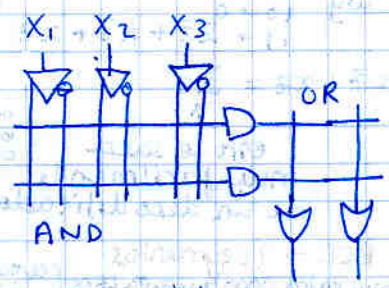
EPROM: se programa por tensión, se desprograma por luz UV

EEPROM: se programa y desprograma por tensión

para hacer funciones lógicas es muy sencillo. Poner X en el minitérmino correspondiente

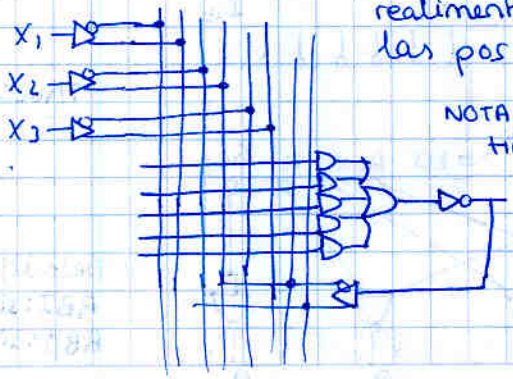
$F_2 = \Sigma m(0, 3)$ sería el ej. de arriba

PLA



son sumas de minitérminos

PAL



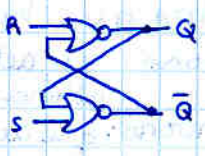
ventaja: la entrada se realimenta → aumenta las posibilidades

NOTA: las AND's que no tienen X (no se usan) están a 0

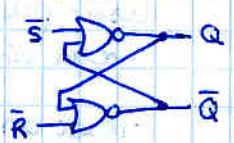
TEMA 5. Biestables y Temporización

- **RS**
  reset → Q=0
  set → Q=1
  R=0 S=0 → Q = Qt-1

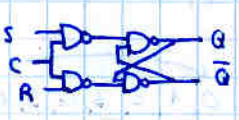
RS NOR



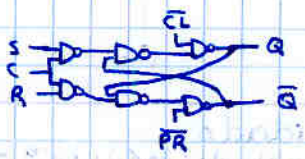
RS NAND



CRS
C=0 → memoria
C=1 → como el RS
es como un cerrojo



con entradas asíncronas
$\overline{PR}$ → preset → Q=1
$\overline{CL}$ → clear → Q=0



Tipos:

RS (set-reset)
reset → Q=0
set → Q=1
R=S=0 → memoria

nota: en RS NAND RS activas a nivel bajo

D
Q = D cuando CLK ↓

JK

J	K	Q_{t+1}
0	0	Q^t memoria
0	1	0
1	0	1
1	1	$\bar{Q}_t$ inversión

T (toggle)
T=0 → $Q_t = Q_t$ memoria
T=1 → $Q_t = \bar{Q}_t$ toggle

tsu: la señal no debe variar durante un tsetup antes de capturarla
thold: la señal no debe variar durante un thold después de haberla capturado

Diseño de un biestable 'Objetivo' a partir de un biestable 'Dato'

ej. SR a JK

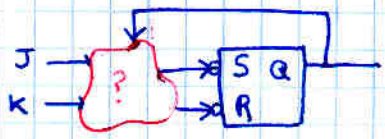
① Tabla de verdad del 'objetivo'

J	K	Q^t
0	0	Q_t
0	1	0
1	0	1
1	1	$\bar{Q}_t$

② Tabla excitación del 'Dato'

Q_t	Q_{t+1}	S	R
0	0	1	X
0	1	0	1
1	0	1	0
1	1	X	1

③ Tabla de funcionamiento global



Entradas			Q_{t+1}	Salidas	
J	K	Q_t		S	R

Tabla funcionam. del JK

tabla de excitación del RS

④ Karnaugh de las salidas

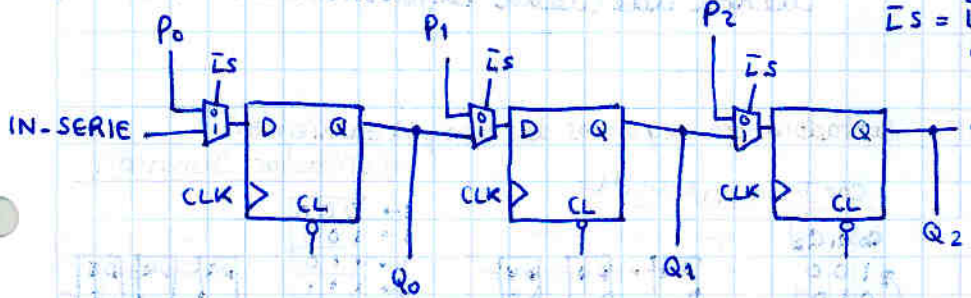
⑤ Implementarlo

PAL secuenciales → con biestables

TEMA 6. Subsistemas secuenciales

Registro de desplazamiento [FF-D]
todas las configuraciones en mismo circuito

Idea básica



$\bar{L}s$ = $\overline{load}$ shift
alterna carga paralelo y desplazamiento

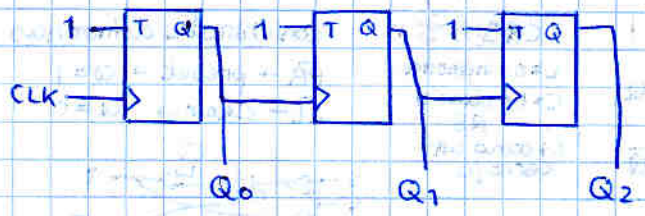
el MUX puede ser de más bits para incluir desplazamiento a izquierda y hold

utilidad: conversión serie-paralelo

$$f_{max} = \frac{1}{T_{min}} = \frac{1}{t_p + t_{su}}$$

Contadores asíncronos (divisores de frecuencia)

FF-T



- tienen estados espúreos
- para hacer cuentas de módulo $< 2^n$ usar puertas que hagan reset
  - reset síncrono: que active al llegar a última cuenta deseada
  - reset asíncrono: que se active al llegar a la primera cuenta no deseada

Cuidado contador módulo 8 = 0, 1, 2, 3, 4, 5, 6, 7

$$f_{max} = \frac{1}{T_{min}} = \frac{1}{n \cdot t_p}$$

a frecuencias tan altas se confunden las cuentas espúreas.

Contadores síncronos:

$$f_{max} = \frac{1}{T_{min}} = \frac{1}{t_p + t_{comb} + t_{su}}$$

ejemplo:
Diseñar contador síncrono módulo 5 con FF JK

1- ¿Cuantos JK necesitamos? $2^2 = 4$, $2^3 = 8$ → 3 FF

2- Recordemos la tabla de verdad del JK y Hagamos su tabla de excitaciones

J	K	Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	$\overline{Q_t}$

⇒

Q_t	Q_{t+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

3- Hagamos la tabla de verdad del circuito

	Estado Actual Q_C	Q_B	Q_A	Estado Siguiente Q_C	Q_B	Q_A	Excitaciones J_C	K_C	J_B	K_B	J_A	K_A	
0	0	0	0	0	0	1	0	X	0	X	1	X	
1	0	0	1	0	1	0	0	X	1	X	X	1	
2	0	1	0	0	1	1	0	X	X	0	1	X	
3	0	1	1	1	0	0	1	X	X	1	X	1	
4	1	0	0	0	0	0	X	1	0	X	0	X	
5	1	0	1	X	X	X	X	X	X	X	X	X	estados que no pertenecen a la cuenta — mínimo coste
6	1	1	0	X	X	X	X	X	X	X	X	X	
7	1	1	1	X	X	X	X	X	X	X	X	X	
5	1	0	1	0	0	0	X	1	0	X	X	1	mínimo riesgo
6	1	1	0	0	0	0	X	1	X	1	0	X	
7	1	1	1	0	0	0	X	1	X	1	X	1	

Podemos poner cualquier estado siguiente para cualquier estado actual, de modo que la cuenta puede ser como nos salga de las narices, incluso mas entrada y segun la entrada cuente de una u otra forma.

Implementación mínimo coste: Suponemos que nunca se entrará en los estados 5, 6 y 7 (teoricamente no se entrará). Permitirá un circuito combinacional más simple

Implementación mínimo riesgo: Si por ruido se entrara en 5, 6, 7 se volvería al cero.

4. Elegimos mínimo coste. Hay que hacer 6 Karnaughs

(dos de ellos son obvios: $K_C = 1$, $K_A = 1$)

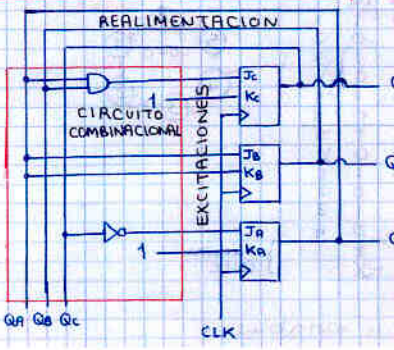
$J_C = Q_A Q_B$ $J_B = Q_A$ $K_B = Q_A$ $J_A = \overline{Q_C}$

hacemos 4 karnaughs (aprovechando la misma tabla para ahorrar espacio)

5. Lo implementamos

$J_C = Q_A Q_B$ $J_B = Q_A$ $J_A = \overline{Q_C}$
$K_C = 1$ $K_B = Q_A$ $K_A = 1$

esto puede ser util para las siguientes preguntas

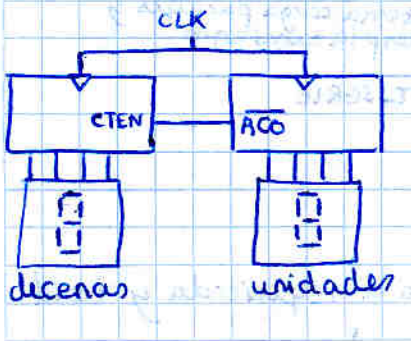


y': entrada up/down

La estructura de los contadores síncronos se puede implementar casi siempre en una PAL registrada

C.I. comerciales

74190 → contador módulo 10
- up/down
- parallel load
- cascadable
  - RCO: se activa subiendo en 9 bajando en 0
  - CTEN: count enable



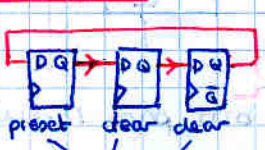
contadores LS 160 - muy típicos
4 modelos: modulo 10, 16 con reset síncrono o asíncrono
TC - terminal carrut para conectar a CEP (count enable) y cascadar

Si hay que hacer contador de módulo menor al dado; puertas lógicas que hagan reset en la cuenta adecuada (según síncrono o asíncrono)

Contadores con registros de desplazamiento:

Contador en anillo

Q_0	Q_1	Q_2
1	0	0
0	1	0
0	0	1



n flipflops
n cuentas
inicia la cuenta

Contador Johnson

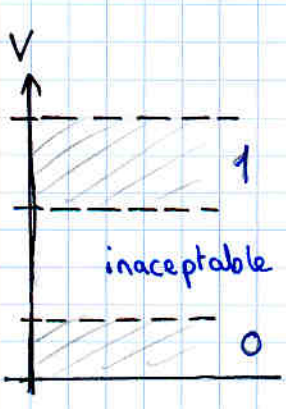
1→	0	0	0
1→	1	0	0
1→	1	1	0
0→	1	1	1
0→	0	1	1
0→	0	0	1

2n cuentas

son muy rápidos

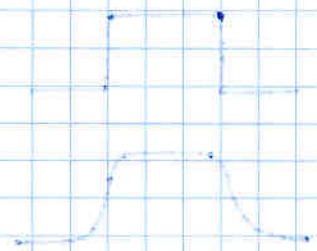
TEMA 1. Introducción a la Electrónica Digital.

1. Introducción
2. Circuitos Lógicos
3. Familias Lógicas
4. Circuitos y Subsistemas Combinacionales
5. Biestables y Temporización
6. Circuitos



Onda Ideal

Onda Real



TEMA 2. CIRCUITOS LÓGICOS

- Variable lógica:
  Puede tomar dos valores ($\{0, 1\}$)
- Función lógica:
  Su resultado es una variable lógica
- Puerta lógica:
  Implementación en circuito electrónico de una función lógica.

ejemplo:

$$F(a,b,c,d) = a \cdot b + c \cdot d + b \cdot c$$

(↑ variables lógicas)

operaciones básicas ·, +

- Funciones de una única variable

$F(a) = a$

A	Z
0	0
1	1

IEEE

$F(a) = \bar{a}$

A	Z
0	1
1	0

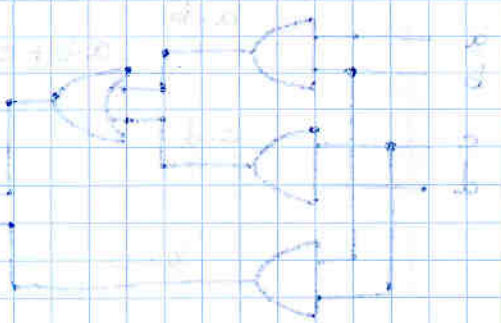
- Función de varias variables

$F(a,b,c,d)$ 2^4 posibles

$$F(a,b,c,d) = a \cdot b + c \cdot d + b \cdot c$$

a	b	c	d	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

↑ ↑ ↑ ↑
1/8 1/4 1/2 mucha frec.



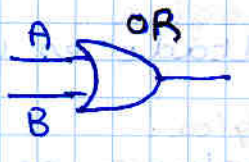
- Funciones de dos variables

$Z = A \cdot B$

A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1

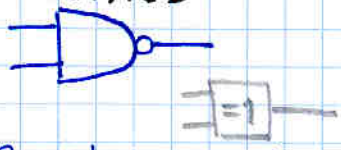
$Z = A + B$

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1



$Z = \overline{A \cdot B}$

NAND



$Z = \overline{A + B}$

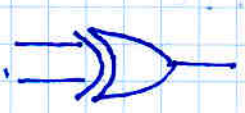
NOR

Puertas no tan básicas:

$Z = A \oplus B$

XOR (ORE)

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0



suma binaria (excepto el que se lleva 1)

$Z = \overline{A \oplus B}$

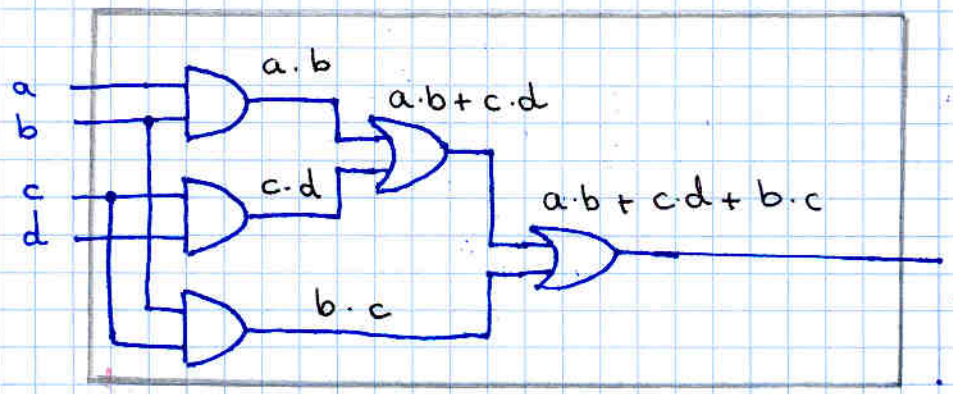
XNOR (NORE)

A	B	Z
0	0	1
0	1	0
1	0	0
1	1	1

detector de iguales

ejemplo:

$F(a, b, c, d) = a \cdot b + c \cdot d + b \cdot c$



Álgebra de Boole

$a+(b\cdot c)=(a+b)\cdot(a+c)$

- Conjunto: Variables lógicas
- Operaciones: $+$, $\cdot$
- Propiedades axiomáticas:
  - Conmutativa: $a+b=b+a$; $a\cdot b=b\cdot a$
  - Elem. neutros: $a+0=a$; $a\cdot 1=a$
  - Asociativa: $a+(b+c)=(a+b)+c$; $a\cdot(b\cdot c)=(a\cdot b)\cdot c$
  - Distributiva: $a\cdot(b+c)=a\cdot b+a\cdot c$; $a+(b\cdot c)=(a+b)\cdot(a+c)$
  - Complementariedad: $a+\bar{a}=1$; $a\cdot\bar{a}=0$

$(a+b)\cdot(a+c)$
$=a\cdot a+a\cdot c+b\cdot a+bc$
$=a(1+c+b)+bc$
$=a+bc$

- Teoremas

  De 1 variable: $a+1=1$ $\quad a+a=a$
  $a\cdot 0=0$ $\quad a\cdot a=a$

  De varias variables: ...

Leyes de De Morgan

1) $\overline{A\cdot B\cdot C\cdot\ldots}=\bar{A}+\bar{B}+\bar{C}+\ldots$

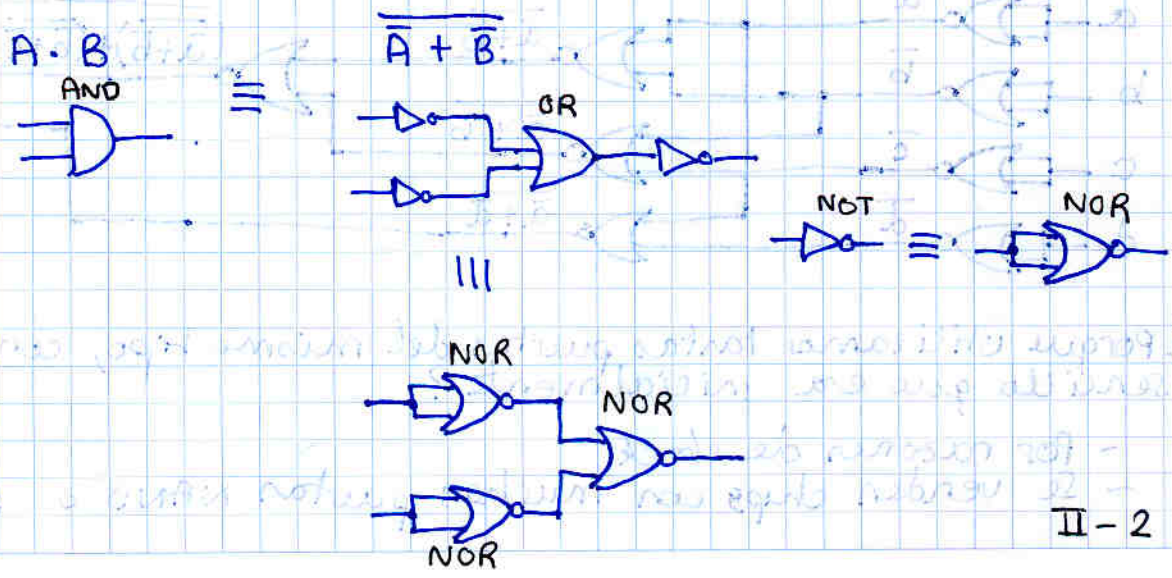
2) $\overline{A+B+C+\ldots}=\bar{A}\cdot\bar{B}\cdot\bar{C}\cdot\ldots$

permite convertir productos en sumas y sumas en productos

ej $A+B=\overline{\overline{A+B}}=\overline{\bar{A}\cdot\bar{C}}$

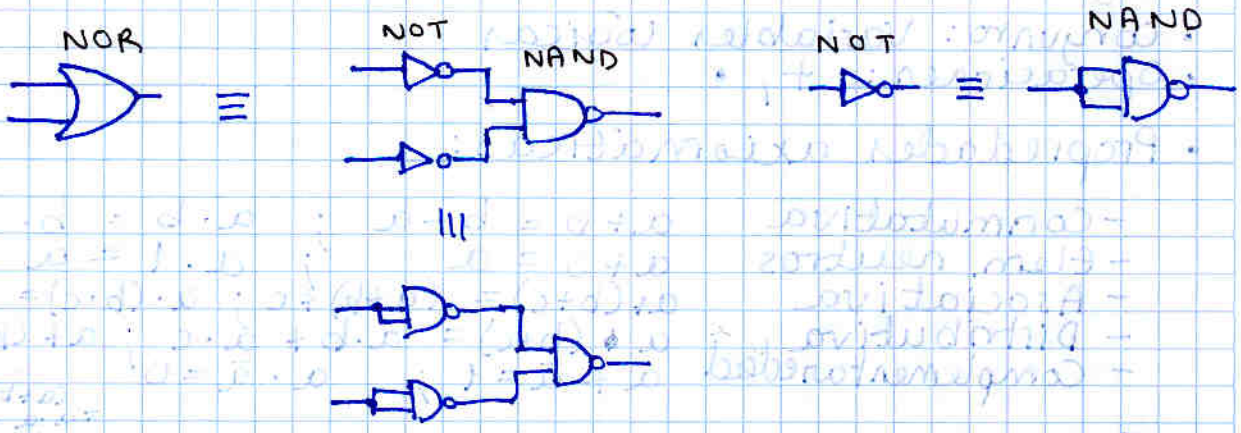
Podemos representar cualquier función lógica con puertas NAND o puertas NOR

ejemplo:



ejemplo

$A + B = \overline{\overline{A+B}} = \overline{\overline{A} \cdot \overline{B}}$



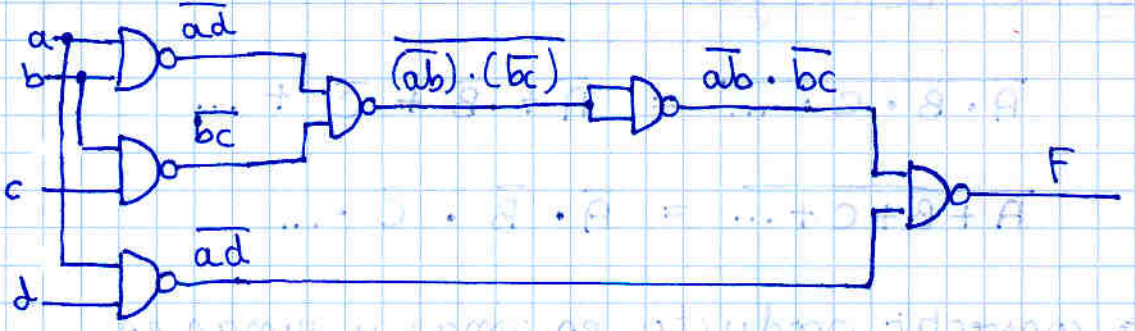
ejercicio:

Sea $F = ab + bc + ad$ sólo como sumas y sólo como productos.

SOLO • (and)

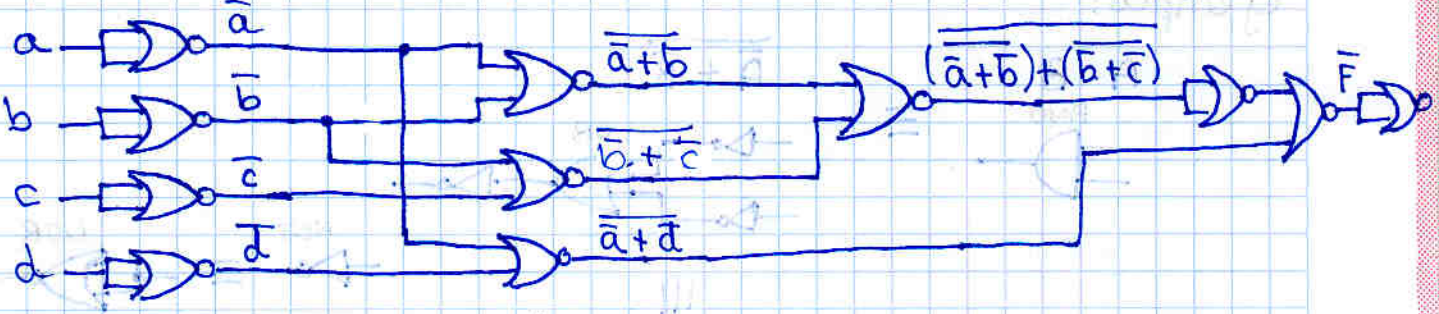
$F = ab + bc + ad = \overline{ab + bc + ad} = \overline{\overline{ab} \cdot \overline{bc} \cdot \overline{ad}}$

Sólo con puertas NAND de 2 entradas



SOLO + (NOR de 2 entradas)

$F = ab + bc + ad = \overline{\overline{ab} + \overline{bc} + \overline{ad}} = \overline{(\overline{a} + \overline{b})} + \overline{(\overline{b} + \overline{c})} + \overline{(\overline{a} + \overline{d})}$



¿Porque utilizamos tantas puertas del mismo tipo, con lo sencillo que era inicialmente?

- Por razones de stock
- Se venden chips con muchas puertas NAND o NOR

Tablas de verdad

Se da la salida para cada combinación de las entradas

ej:

	A	B	C	F
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

F vale 1 cuando
- A y B son 0 y C es 1
- ó A y C son 0 y B es 1
- ó A es '0' y B y C son 1
- ó A y B y C son 1

$$F = \underbrace{\bar{A}\cdot\bar{B}\cdot C}_{\text{miniterminos}} + \underbrace{\bar{A}\cdot B\cdot\bar{C}}_{\text{(productos de variables)}} + \bar{A}BC + ABC$$

Minitérmino canónico: contiene todas las variables

Función canónica: son todo minitérminos canónicos

Al estar ABC juntos se puede ~~pasar~~ utilizar su representación decimal

$$F(A,B,C) = \sum(1,2,3,7)$$

$$\bar{F}(A,B,C) = \sum(0,4,5,6)$$

$$\bar{F} = \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + A\bar{B}C + AB\bar{C}$$

$$F = \overline{\bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + A\bar{B}C + AB\bar{C}}$$

$$= \overline{\bar{A}\bar{B}\bar{C}}\cdot\overline{A\bar{B}\bar{C}}\cdot\overline{A\bar{B}C}\cdot\overline{AB\bar{C}}$$

$$F = \underbrace{(A+B+C)}\cdot\underbrace{(\bar{A}+\bar{B}+C)}\cdot\underbrace{(\bar{A}+B+\bar{C})}\cdot\underbrace{(\bar{A}+\bar{B}+C)}$$

Maxiterminos

Maxitérminos canónicos si tienen todas las variables

Los decimales que representan a un maxitermino es lo contrario (negados) (es decir $\bar{A}+\bar{B}+C$ es el $110 = 6$)

$$F(A,B,C) = \Pi(0,4,5,6)$$

Tabla de verdad

donde estan los unos

↳ Suma de productos (miniterminos) $\sum(1,2,3,7)$

↳ Producto de sumas (maxiterminos) $\Pi(0,4,5,6)$

donde estan los ceros.

En productorio cambian los sentidos lógicos de las variables

Al pasar lo al producto de maxiterminos era al reves

$0 \rightarrow (A+B+C)$

$4 \rightarrow 100 \rightarrow (\bar{A}+B+C)$

ej:

OR: $Z = A+B$

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

$$Z(A,B) = \sum_2 m(1,2,3) = \bar{A}B + A\bar{B} + AB$$

$\sum_2$: indica que hay 2 variables

m: indica miniterm.

$(1,2,3)$: valores decimales donde vale 1

$$Z(A,B) = \prod_2 M(0) = \overline{(A+B)}$$

las dos a 1 porque en maxiterminos es al reves

$$\bar{Z}(A,B) = \sum_2 m(0) = \bar{A}\bar{B}$$

ej: NAND $\overline{AB}$

A	B	Z
0	0	1
0	1	1
1	0	1
1	1	0

$Z(A,B) = \sum_2 m(0,1,2) = \bar{A}\bar{B} + \bar{A}B + A\bar{B}$

$z(A,B) = \prod_2 M(3) = \bar{A} + \bar{B}$

$\bar{Z}(A,B) = \sum_2 m(3) = A \cdot B$

ej: XOR

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0

$Z(A,B) = \sum_2 m(1,2) = \bar{A}B + A\bar{B}$

$Z(A,B) = \prod_2 M(0,3) = (A+B)\cdot(\bar{A}+\bar{B})$

$\bar{Z}(A,B) = \sum_2 m(0,3) = \bar{A}\bar{B} + AB$

expresión formal del XOR

$A \oplus B := \bar{A}B + A\bar{B}$

Minimización de funciones

ej:

$F(a,b) = \overline{ab + \bar{a}b + \bar{b}a} = \overline{ab} \cdot \overline{\bar{a}b} \cdot \overline{\bar{b}a} = (\bar{a}+\bar{b})\cdot(a+\bar{b})\cdot(b+\bar{a})$

$= (\bar{a}a + \bar{a}\bar{b} + \bar{b}a + \bar{b}\bar{b})\cdot(b+\bar{a})$

$= \bar{b}\cdot(1 + a + \bar{a})\cdot(b+\bar{a})$

$= \bar{b}\cdot(b+\bar{a})$

$= \bar{a}\bar{b}$

Sacar factor comun es muy util $\quad ab + a\bar{b} = a$

Habia una forma mas facil

$F(a,b) = \overline{ab + \bar{a}b + \bar{b}a}$

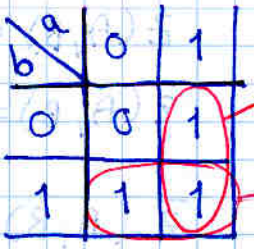
11 01 10

F vale 1 cuando no 11, 01, 10

$\Rightarrow$ F vale 1 cuando 00

Mapas de Karnaugh

a	b	F
0	0	0
0	1	1
1	0	1
1	1	1



$a\bar{b} + ab = a$

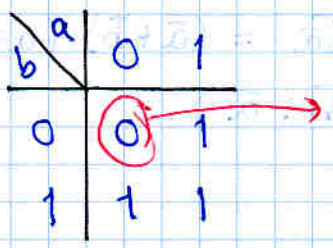
$b\bar{a} + ba = b$

$F = a + b$

$$F = \bar{a}b + a\bar{b} + ab = b(\bar{a} + a) + a\bar{b}$$
$$= b + a\bar{b} \quad \text{DISTRIBUTIVA RARA}$$
$$= (b + a)(b + \bar{b})$$
$$= b + a$$

$$F(a, b) = \sum_2 m(1, 2, 3) = \prod_2 M(\emptyset)$$

se puede también agrupar por ceros (con maxiterminos)

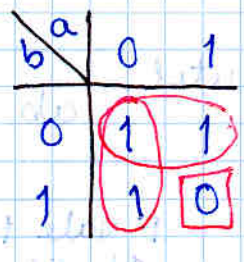


el Maxitermino correspondiente a 00

$F = (a + b)$

Ejemplo 2.

a	b	F
0	0	1
0	1	1
1	0	1
1	1	0



$F = \bar{a} + \bar{b}$

ejemplo 3

a	b	F_3
0	0	0
0	1	1
1	0	1
1	1	0

es una XOR

por unos

b \ a	1	0
1	0	1 → $a\bar{b}$
0	1 → $\bar{a}b$	0

$F_3 = a\bar{b} + \bar{a}b$

por ceros

b \ a	1	0
1	0 → $\bar{a}+\bar{b}$	1
0	1	0 → $a+b$

$F_3 = (a+b)\cdot(\bar{a}+\bar{b})$

ejemplo 4

b \ a	0	1
0	1	1 → F = 1
1	1	1

Únicamente se pueden agrupar potencias de 2.
(Si agrupas solo 3, 5, ... no podrías simplificar)

ejemplo 5 Karnaugh de 3 variables

a	b	c	F_5
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

$F(a,b,c) = \sum_3 m(1,2,4,5,6)$

$= \prod_3 M(0,3,7)$

c \ ab	00	01	10	11
0			→ NO SON VECINAS!	
1				

Para que dos celdas sean vecinas, tiene que mantenerse al menos 1 variable constante de las de arriba

reordenar

c \ ab	00	01	11	10
0	0 (a+b+c)	1	1	1
1	1	0	0	1

es un cilindro

$F = a\bar{b} + b\bar{c} + \bar{b}c$

$F = (\bar{b}+\bar{c})\cdot(a+b+c)$

$a\bar{b}$, $b\bar{c}$, $\bar{b}c$, $\bar{b}+\bar{c}$

ejemplo 6

c \ ab	00	01	11	10
0	1	0	1	1
1	1	0	1	1

$a + \bar{b}$ $\quad a$ $\quad \bar{b}$

$$F = a + \bar{b}$$

ejemplo 7

c \ ab	00	01	11	10
0	1	1	1	0
1	0	1	1	0

$$F = b + \bar{a}\bar{c}$$

$$F = (\bar{a} + b) \cdot (b + \bar{c})$$

ejemplo 8

bc \ a	0	1
00	0	0
01	1	0
11	1	1
10	0	0

$$F = bc + \bar{a}c$$

$$F = c \cdot (\bar{a} + b)$$

ejemplo 9

bc \ a	0	1
00	1	0
01	0	1
11	1	0
10	0	1

(0,0,0) (1 0 1) (0 1 1) (0 0 1)

$$F = \bar{a}\bar{b}\bar{c} + a\bar{b}c + \bar{a}bc + ab\bar{c}$$

$$F = (\bar{a}+b+c) \cdot (\bar{a}+\bar{b}+c) \cdot (a+b+c) \cdot (\bar{a}+\bar{b}+c)$$

(1 0 0) (0 0 1) (1 1 1) (0 1 0)

ejemplo 10

bc \ a	0	1
00	1	1
01	1	0
11	0	1
10	1	1

$$F = \bar{c} + \bar{a}\bar{b} + ab$$

$$F = (\bar{a}+b+\bar{c}) \cdot (a + \bar{b} + \bar{c})$$

bc \ ab	00	01	11	10
0	1	1	0	1
1	0	1	1	1

bien

1	1	0	1	
0	1	1	1	

bien

1	1	0	1	
0	1	1	1	

mal, nos hemos parado.
F sería correcta pero no estaría minimizada

Karnaugh de 4 variables

ejemplo

a	b	c	d	F

se une 4b por arriba

cd \ ab	00	01	11	10	
00	1	1	1	1	$\bar{c}\bar{d}$
01	1	1	0	0	$\bar{a}\bar{c}$
11	0	0	1	1	acd
10	0	1	0	1	$a\bar{b}\bar{d}$

$\bar{a}b\bar{d}$

Cuanto más agrupamos menor es el miniterminio

1	1	1	1
1	1	0	0
0	0	1	1
0	1	0	1

$F = (\bar{a} + c + \bar{d}) \cdot (a + \bar{c} + \bar{d}) \cdot (a + b + \bar{c}) \cdot (\bar{a} + \bar{b} + \bar{c} + d)$

ejemplo

cd \ ab	00	01	11	10
00	1	0	1	1
01	1	1	1	0
11	0	0	0	0
10	1	1	1	1

ejemplo

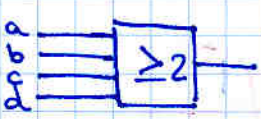
cd \ ab	00	01	11	10
00	1	1	1	1
01	0	0	0	0
11	0	0	1	1
10	1	0	0	1

MAL!

cd \ ab	00	01	11	10	
00	1	1	1	1	
01	0	0	0	0	$\bar{c}\bar{d}$
11	0	0	1	1	acd
10	1	0	0	1	$\bar{b}\bar{d}$

¡los 4 extremos son vecinos!

Ejercicio: Circuito NAND de 2 entradas q detecte cuando 2 de 4 pulsadores estan activos



cd \ ab	00	01	11	10
00	0	0	1	0
01	0	1	1	1
11	1	1	1	1
10	0	1	1	1

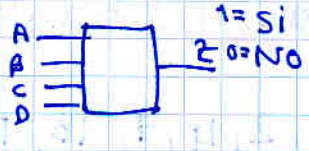
Todas las agrupaciones tienen al menos un termino propio → está bien

$$F = cd + ab + bd + ad + bc + ac$$

era lógico: son todas las parejas

Problema A-3

Socio A	45%
Socio B	30%
Socio C	15%
Socio D	10%



se acepta si se consigue 50%.

A	B	C	D	Z
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

CD \ AB	00	01	11	10
00	0	0	1	0
01	0	0	1	1
11	0	1	1	1
10	0	0	1	1

AD, AC, CDB, AB

$$Z = AB + AC + AD + CDB$$

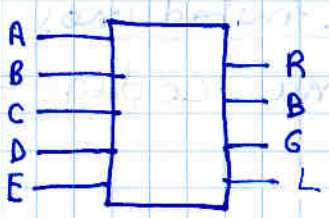
$$Z = (A+B)\cdot(A+C)\cdot(A+D)\cdot(B+C+D)$$

Problema A-5

Iluminación discoteca ← roja, azul, verde, laser

E — apaga todo el equipo cuando está a cero

Roja — A ó B y no C
Azul — no B ó D y no A
Verde — no C ó D
Rayo laser — un nº par de luces encendidas



$$R = E(A + B\bar{C})$$

$$B = E(\bar{B} + D\bar{A})$$

$$G = E(\bar{C} + D)$$

$$L = E\cdot(RB + RG + BG)\cdot(\overline{RBG})$$

¿L está minimizado?

G \ RB	00	01	11	10
0	0	0	1	0
1	0	1	0	1

$$L = \bar{R}BG + RB\bar{G} + R\bar{B}G$$

Haciendo Karnaugh de 5 variables para 4 funciones habría sido bárbaro.

Funciones Incompletas. No totalmente especificados

$$F(a\,b\,c\,d) = \sum_4 m(1,2,4,7,13,14) + O(\emptyset, 6, 8, 9)$$

no especificadas

(F)

cd \ ab	00	01	11	10
00	0 X	4 1	12 0	8 X
01	1 1	5 0	13 1	9 X
11	3 0	7 1	15 0	11 0
10	2 1	6 X	14 1	10 0

¿Qué hago con X?
serán ceros cuando quiera
serán unos cuando quiera
según lo que más convenga

$F = \bar{b}\bar{c} + \bar{a}\bar{d} + \bar{a}bc + a\bar{c}d + bc\bar{d}$

Karnaugh de 5 variables

de \ abc	000	001	011	010	110	111	101	100
00	1							1
01			1			1		
11			1			1		
10	1			1	1	1	1	1

$\bar{C}D\bar{E}$ — se pliega — $A\bar{D}\bar{E}$

este no podría coger una fila de 4 porque amigos de distintas tablas tienen que tener simetría

← sólo cambia un bit

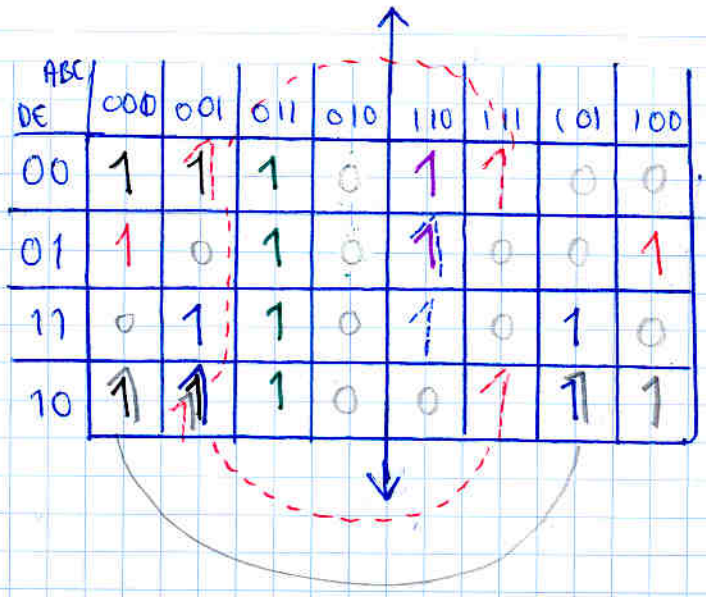
sabiendo 00 01 11 10 se refleja − 10 11 01 00

y se añade 0 a los 4 primeros y 1 a los 4 sig.

Con 4 variables se haría también un reflejo de todo y la mitad ceros y la mitad unos

Se llama código GRAY

Existen algoritmos matemáticos que simplifican las funciones (lo que hacen los ordenadores)



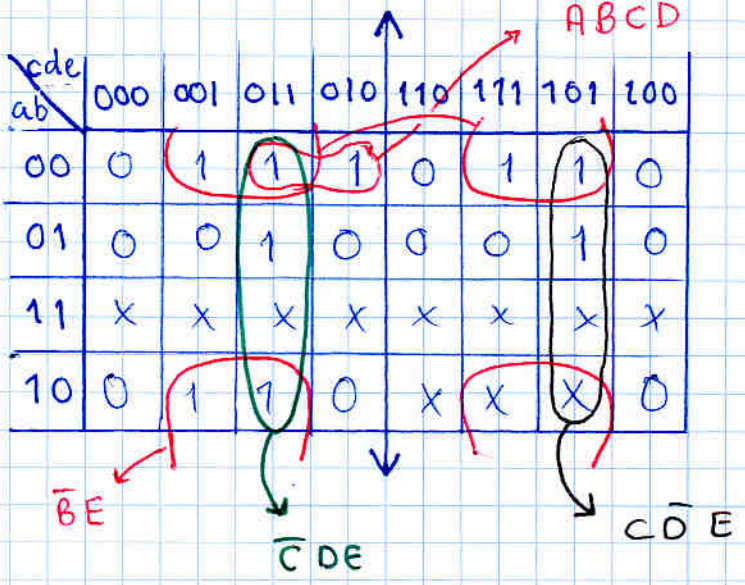
$\bar{a}\,\bar{b}\,\bar{e}$
$\bar{a}\,b\,c$
$\bar{b}\,\bar{c}\,\bar{d}\,e$
$\bar{b}\,c\,d$
$\bar{b}\,d\,\bar{e}$
$b\,c\,\bar{e}$
$a\,b\,\bar{c}\,\bar{d}$
$a\,b\,\bar{c}\,e$

Problema

Diseñe circuito NOR 2 entradas que detecte numeros primos entre el cero y el veinte

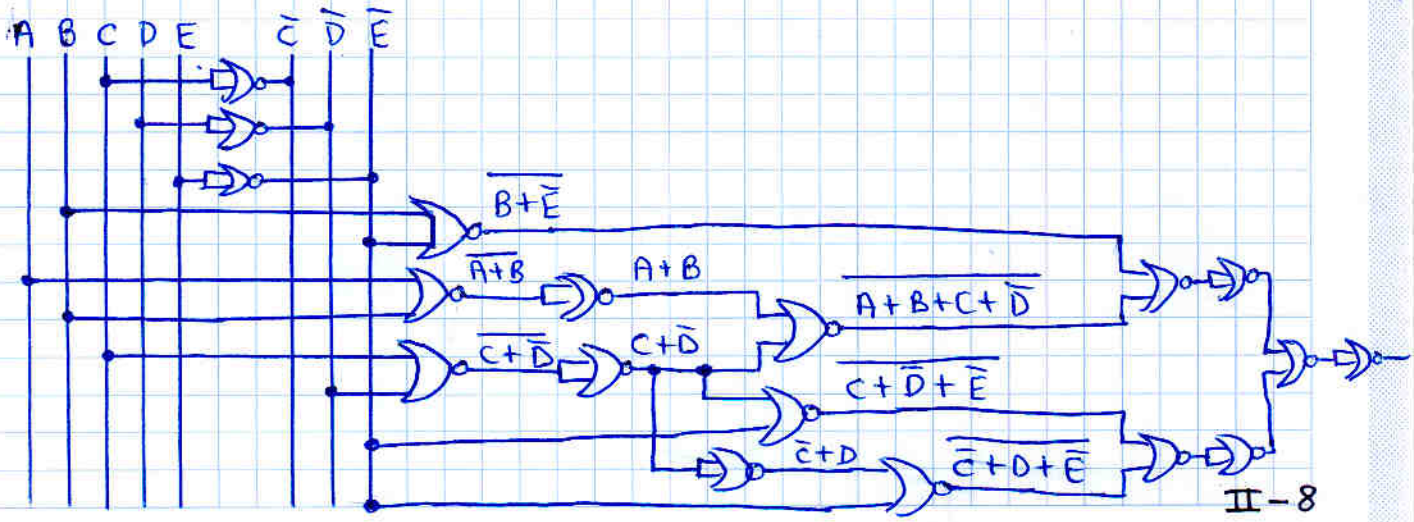
a	b	c	d	e	F
0	0	0	0	0	
0	0	0	0	1	
0	0	0	1	0	
0	0	0	1	1	
0	0	1	0	0	
0	0	1	0	1	
0	0	1	1	0	
0	0	1	1	1	
0	1	0	0	0	
0	1	0	0	1	
0	1	0	1	0	
0	1	0	1	1	
0	1	1	0	0	
0	1	1	0	1	
0	1	1	1	1	
1	0	0	0	0	
1	0	0	0	1	
		⋮			

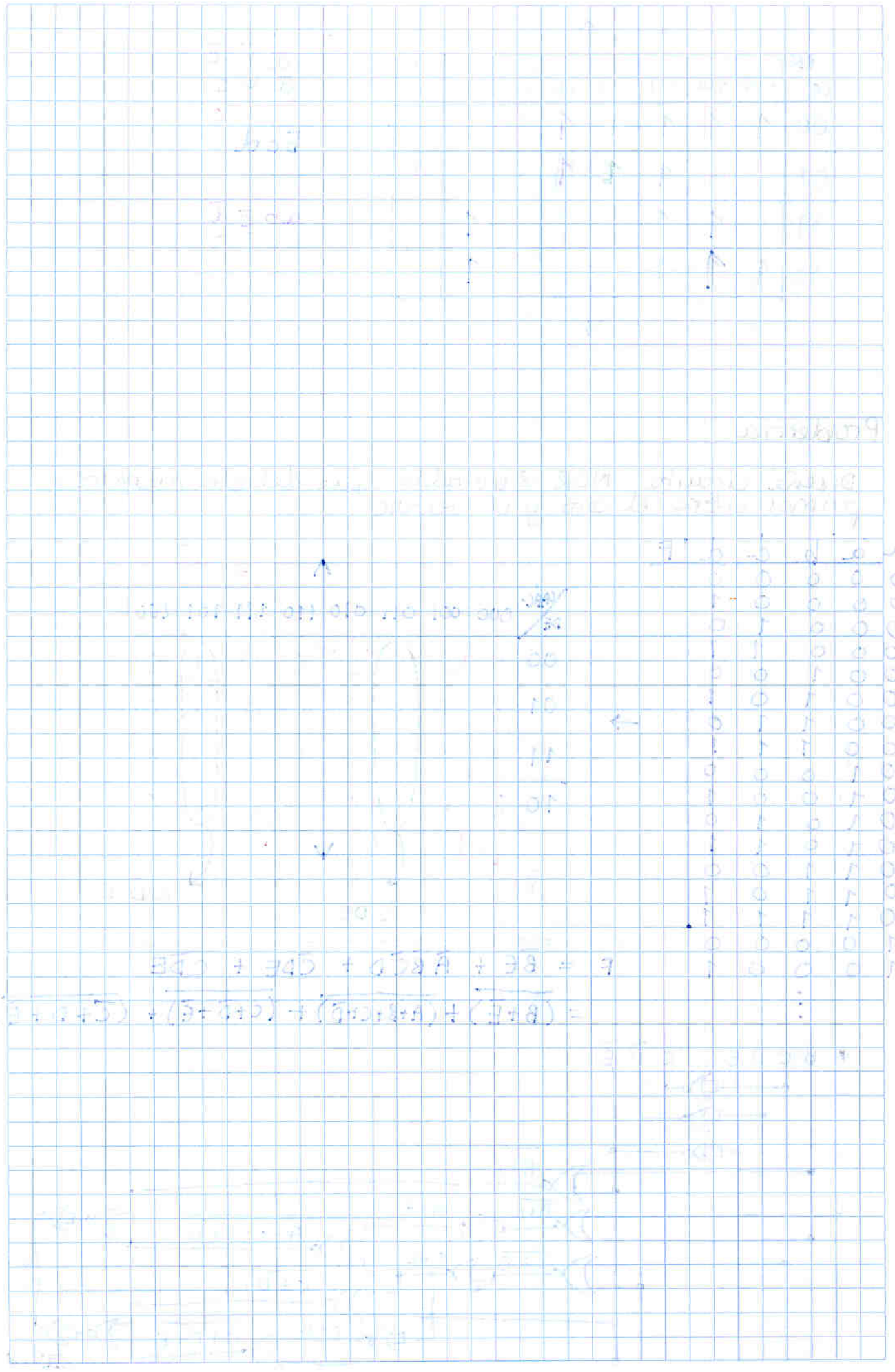
→



$$F = \overline{\bar{B}E + \bar{A}B\bar{C}D + \bar{C}DE + C\bar{D}E}$$

$$= \overline{(B+\bar{E})} + \overline{(A+B+C+\bar{D})} + \overline{(C+\bar{D}+\bar{E})} + \overline{(\bar{C}+D+\bar{E})}$$





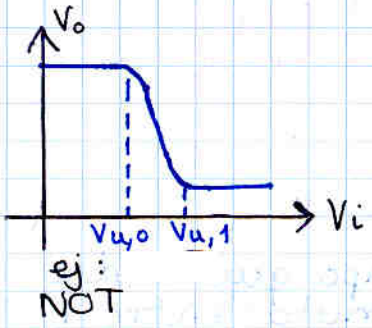
TEMA 3. FAMILIAS LÓGICAS

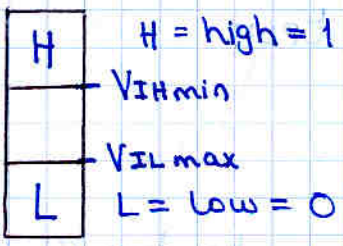
→ Fan-out: el 1 que da una puerta lógica no puede utilizarse para infinitas puertas; "todas tienen que comer". Fan-out es el máximo de puertas

→ Unidad de carga: U.L. (Unit Load).
la unidad con la cual se mide la corriente
ej U.L. = 1'5 mA de una familia de puertas

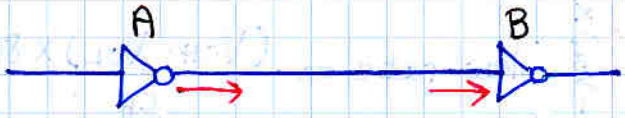
→ Fan-In: el número de unidades de carga que pide una cierta puerta; ej Fan-In de 7 unidades de carga

→ Tensiones Umbrales

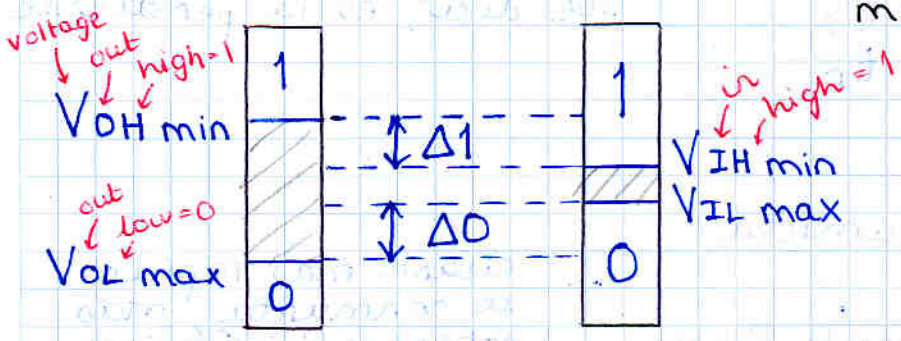




→ **Márgenes de ruido**



Salida A — Entrada B



Δ1: la tensión que se puede perder en el cable margen de ruido para el 1

Δ0: la tensión que se puede absorber en el cable margen de ruido para el 0

Datos del fabricante
- $V_{OH\,min}$
- $V_{OL\,max}$
- $V_{IH\,min}$
- $V_{IL\,max}$
- I_{OH}
- I_{OL}
- I_{IH}
- I_{IL}
- Δ1
- Δ0
- Fan-out

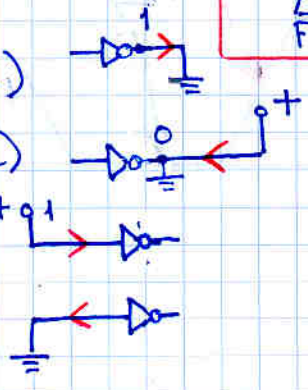
Exactamente lo mismo ocurre para corrientes

I_{OH} (out, high) < 0 (corriente sale)

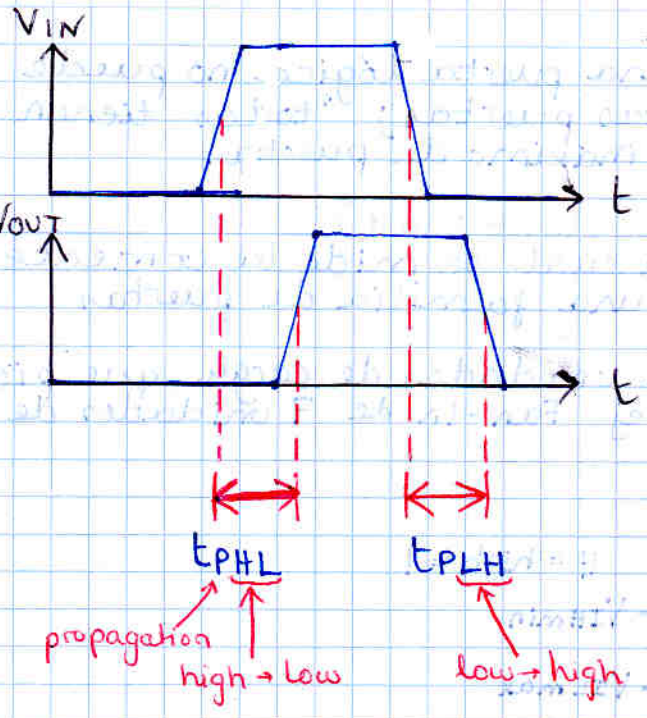
$I_{OL} > 0$ (corriente entra)

$I_{IH} > 0$ (entra)

$I_{IL} < 0$ (sale)



→ Tiempo de propagación



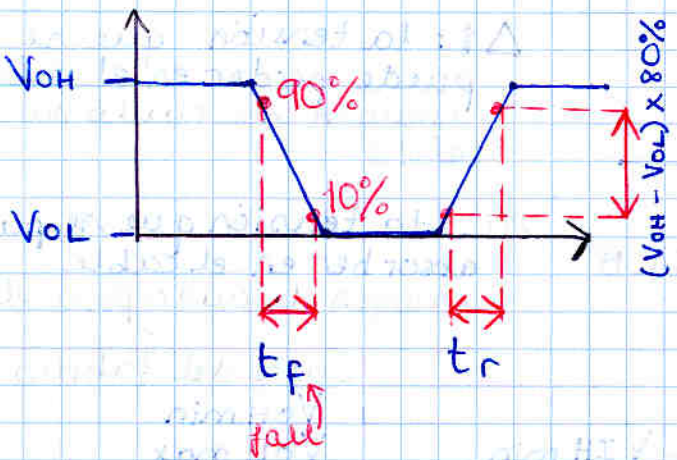
El tiempo que tarda en reaccionar a un cambio en la entrada.
Es decir, lo que tarda en "calcular" la salida

$$t_{PD} = \frac{t_{PLH} + t_{PHL}}{2}$$

↑ tiempo de propagación

→ Slew-rate

tf, tr: El tiempo que tarda en conmutar entre 1 ó 0. Cuanto más corto, más caro.

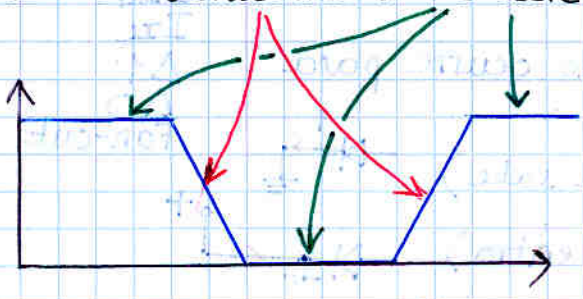


$$\text{slew-rate} = \frac{(V_{OH} - V_{OL}) \times 80\%}{t_f \text{ (ó } t_r)}$$

slew-rate es la pendiente

→ Potencia consumida

P = Pdinamic + Pstatic



Cuanto más rápido se conmuta, más potencia dinámica se gasta.
ej: microprocesador

Clasificación familias lógicas

Transistores Bipolares

- **TTL**
  - la 1ª q triunfó
  - estandarizó todo (nombres, parámetros)
  - 74 ≡ TTL estándar
- **ECL**
  - muy caro
  - muy rápido
  - fallo: poco margen de ruido

Transistores MOS

- puerta de metal
- puerta de silicio
  - 74C

Vamos a utilizar para entender una familia histórica que no se utiliza.

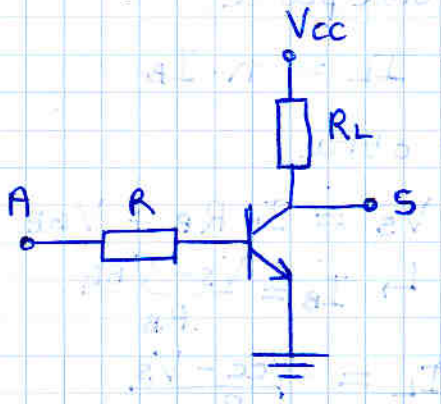
Familias Históricas

· Familia RTL

- no llegó a ser comercial
- poco margen de ruido

Valores típicos
$R = 450\ \Omega$
$R_L = 640\ \Omega$
$V_{cc} = 5V$
$\beta = 100$

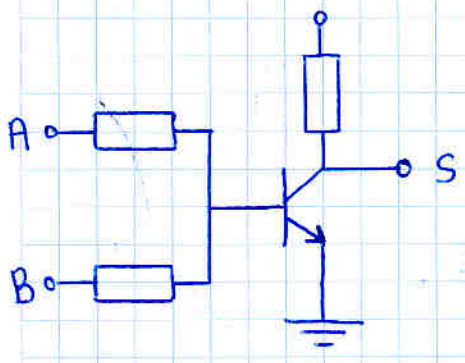
ej: puerta NOT

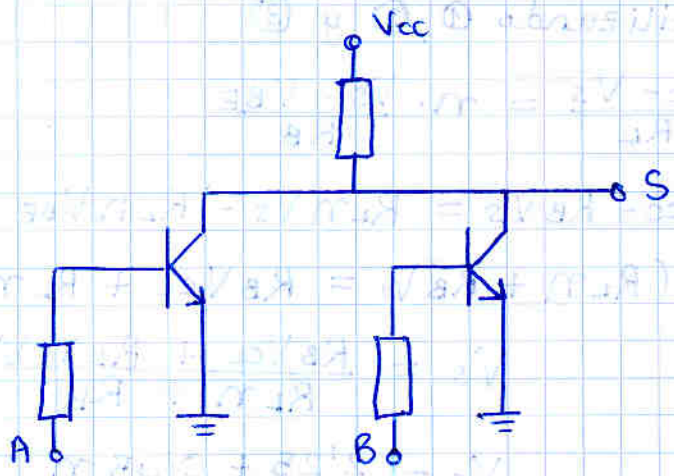


si A='0', A conectado a tierra
↳ transistor en corte
↳ $S = V_{cc} - R_L I_C$ = '1' lógico

si A='1', transistor en saturación (está diseñado así)
$S = 0'2\ V \simeq 0$ = '0' lógico

ej: puerta NOR

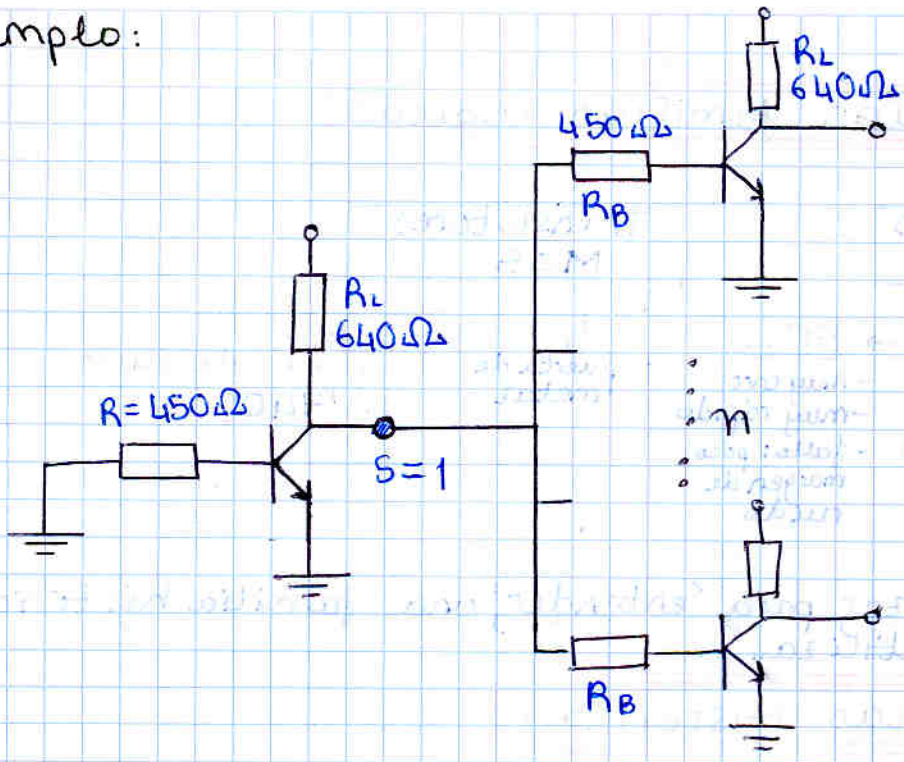




A	B	TRT	S
0	0	corte	0
0	1	sat	1
1	0	sat	1
1	1	corte	0

} está en sat porque el '1' se reparte entre el '0' = tierra que llega, y la tierra del transistor

ejemplo:

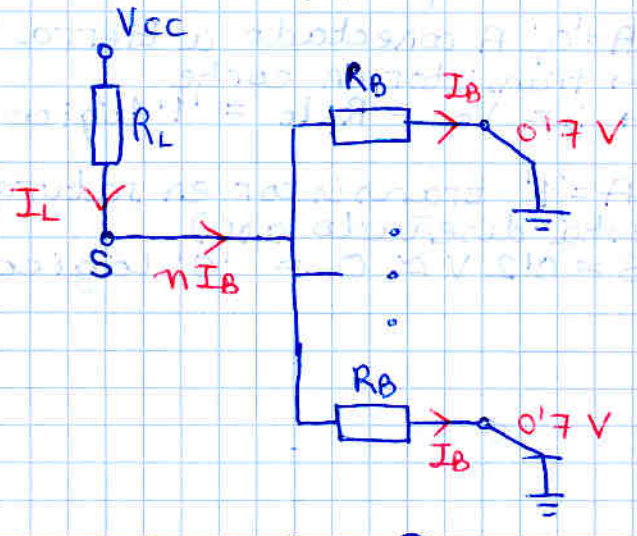


normalmente
$V_{cc} = 5V$

Datos
$V_{BE(sat)} = 0'7\,V$
$V_{CE(sat)} = 0'3\,V$

Calcular:
a) la tensión de salida V_s si se le conectan n puertas

El transistor está en corte, y los demas están en saturación, por tanto



Por una parte

① $I_L = n \cdot I_B$

por otra

②

$$I_B = \frac{V_s - V_{BE}}{R_B}$$

③ $$I_L = \frac{V_{cc} - V_s}{R_L}$$

utilizando ①, ② y ③

$$\frac{V_{cc} - V_s}{R_L} = n \cdot \frac{V_s - V_{BE}}{R_B}$$

$$R_B V_{cc} - R_B V_s = R_L n V_s - R_L n V_{BE}$$

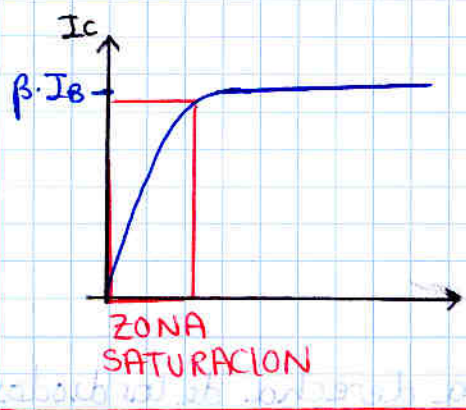
$$V_s (R_L n + R_B) = R_B V_{cc} + R_L n V_{BE}$$

$$V_s = \frac{R_B V_{cc} + R_L n V_{BE}}{R_L n + R_B}$$

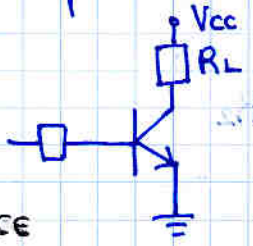
$$V_s = \frac{2'25 + 0'45\,n}{0'45 + 0'64\,n}$$

b) El número de puertas que podemos conectar (fan-out) que garantice la saturación de los transistores ($\beta = 100$)

IMPORTANTE



para saturación: $I_C < \beta \cdot I_B$



$$I_C = \frac{V_{CC} - V_{CE\,SAT}}{R_L}$$

apartado a) $V_S = I_B R_B + V_{BE}$

$$I_B > \frac{I_C}{\beta}$$

$$V_S > \frac{I_C}{\beta} R_B + V_{BE}$$

$$I_C = \frac{V_{CC} - V_{CE\,SAT}}{R_L}$$

$$V_S = \frac{2'25 + 0'45\,n}{0'45 + 0'64\,n}$$ apartado a)

$$\frac{2'25 + 0'45\,n}{0'45 + 0'64\,n} > \frac{V_{CC} - V_{CE\,SAT}}{R_L \beta} R_B + V_{BE}$$

$$> \frac{2115}{64000} + 0'7 = 0'7330$$

$$2'25 + 0'45\,n > 0'3299 + 0'46915\,n$$

$$0'01915\,n < 1'9201$$

$$n < 100'3$$

fan-out = 100

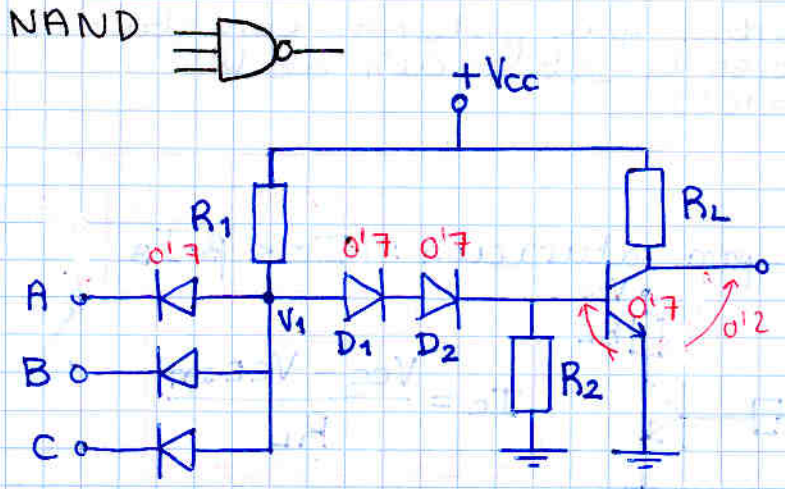
nota: con el máximo fan-out; Vs da justo lo necesario para alimentar a 100 puertas. No puede caer nada en el cable.
Margen de ruido = 0V

Si n es menor, menos corriente por R_L de 1ª puerta, y mayor Vs. Esta Vs podrá caer hasta el Vs dado por el fanout.

c) Pot. en la puerta excitadora $P = V_{CC} \cdot I_C = V_{CC} \cdot \frac{V_{CC} - V_S(n)}{R_C}$

Familia DTL

NAND



- Caso A = B = C = '1'
  Por diseño será mayor V_A, V_B, V_C que a la derecha de los diodos V_1
  ↳ Diodos A, B, C → OFF
  ↳ Diodos D_1, D_2 → ON ⟶ TRT = saturado → S = 0'2 = '0'
  (gracias a la polarización de diseño (p.ej R_2))

- Caso cualquier entrada = '0' → Diodo = ON
  la corriente que da Vcc la chupa tierra
  D_1 y D_2 necesitan 2 x 0'7 para conducir; gana el diodo de la entrada
  → TRT = corte → S = '1'

ejercicio

Cálculo $V_{IH\,min}$: en HIGH uno de los 3 diodos de la entrada debe conducir (p.ej el D_A)

$V_{DA} = V_1 - V_A$ } $V_A = V_{IH}$
$V_{DA} > 0'7$ } $V_{IH\,min} = \underbrace{V_1}_{V_1 = 0'7+0'7+0'7 = 2'1} - 0'7 = 1'4\,V$

Potencia en HIGH:

$$i_{R_1} = \frac{V_{cc} - V_1}{R_1} = \frac{V_{cc} - 2'1}{R_1}$$

$$i_{R_2} = \frac{V_{BE}}{R_2} = \frac{0'7}{R_2}$$

$$i_{R_L} = \frac{V_{cc} - V_{CE}}{R_L} = \frac{V_{cc} - 0'2}{R_L}$$

Pot consumida $= V_{cc} \cdot (I_{R_1} + I_{R_L})$ la que sacamos de la batería

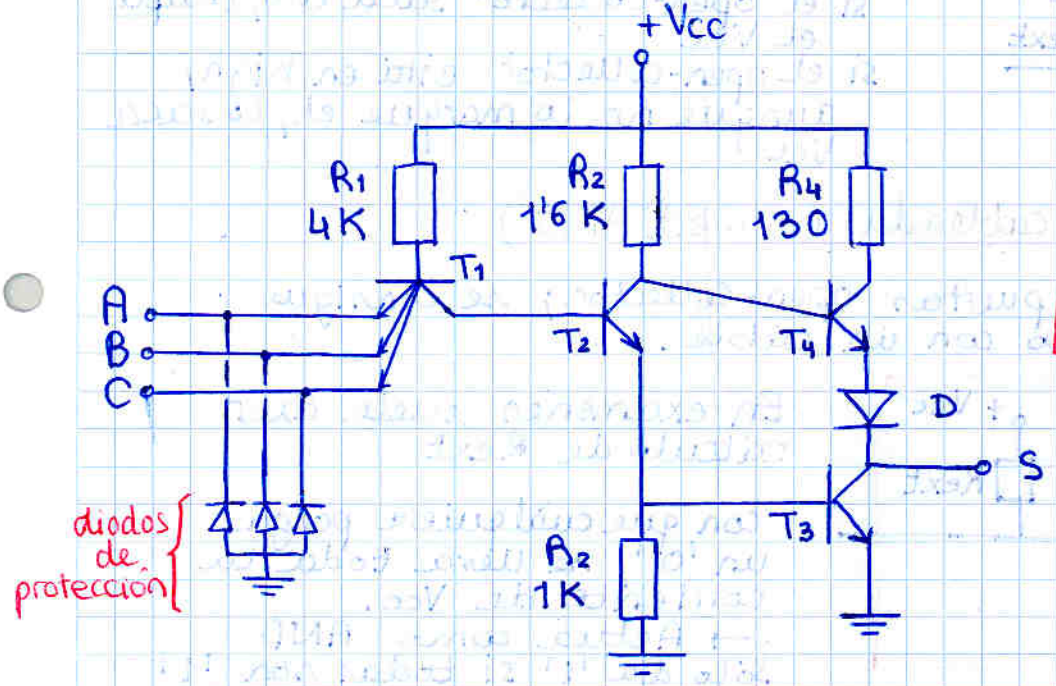
Pot disipada $= I_{R_1}^2 R_1 + I_{R_2}^2 R_2 + I_{R_L}^2 R_L$

Familias Comerciales

- Familia TTL

TTL ≡ transistor transistor logic
Fue la 1ª que funcionó muy bien
Estandarizó todo → V_{IH}, V_{IL}, ...

NAND - Salida Totem Pole



[Salida Totem Pole]

T_4 sat → T_3 corte
T_4 corte → T_3 sat

- Supongamos A = B = C = '1'
  T_1: BE = OFF
  BC = ON !! (se llama activo inverso)
  ↳ T_2: saturado → cortocircuito V_{CE}
  ↳ T_3: saturado → T_4 corte ¿Porque? (V_{BE} = 0'9 - 0'9 Por el Diodo = 0 < 0'7)
  ↳ S = '0'

- Supongamos cualquier entrada = '0'
  ↳ sumidero: absorbe corriente → Tanto por R_1 como de T_2
  ↳ a T_2 le absorben corriente → Corte muy rápido (conmutación veloz)
  ↳ T_3: corte → T_4: saturado
  ↳ $S = V_{CC} - I_{R_4} R_4 - V_{CE_{T4}} - V_D$ = '1'

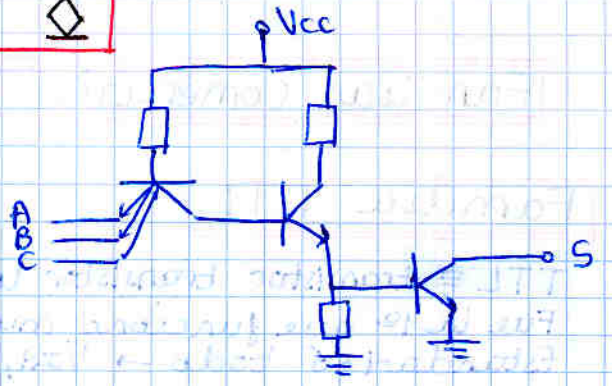
Diodos de protección contra sobretensiones negativas que podrían quemar T_1 (absorbería mucha corriente)
se absorben si $V_{A,B,C} < -0'7$

Salida Open-Collector ◇

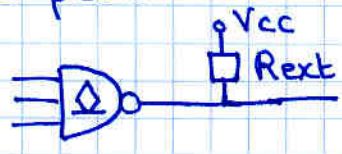
quitando R_4, T_4 y D

Funciona igual en LOW
En HIGH no saca 1's

◇ sólo saca ceros!



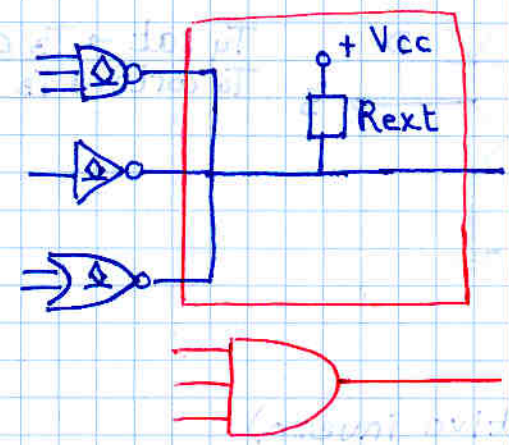
Se puede solucionar conectando un Vcc fuera



si el open-collector saca cero, chupa el Vcc
si el open-collector está en high, aunque no lo marque el, lo saca Vcc

Puerta AND cableada (wired AND)

Uniendo varias puertas open-collector; se consigue 'sumarlas' sólo con un cable.

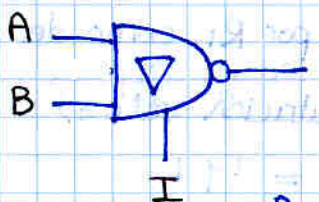


En exámenes suele caer cálculo de Rext

Con que cualquiera ponga un '0', se lleva toda la corriente de Vcc.
→ Actúa como AND
Sólo da '1' si todas son '1'

Se utiliza en la conexión de periféricos a un microprocesador

Salida Triestado ▽



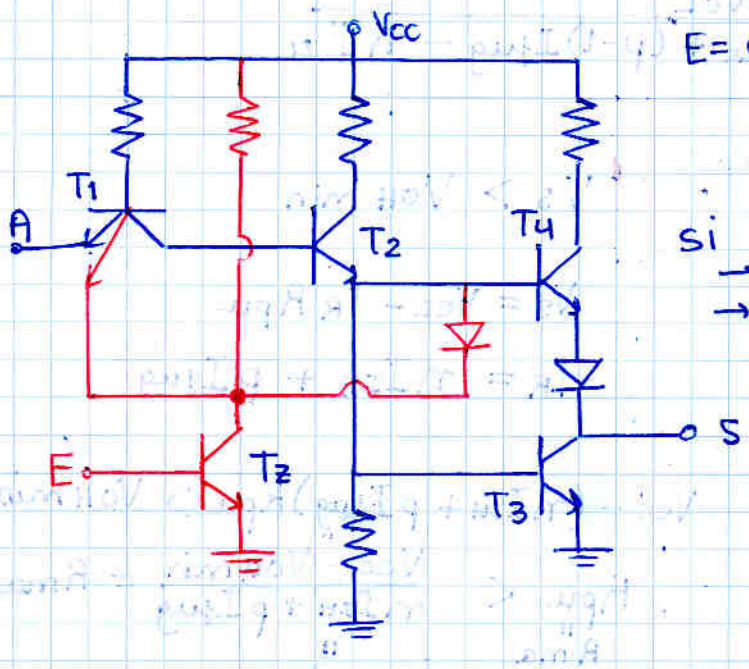
I = 1 → actúa como puerta normal

I = 0 → la puerta saca ni '1' ni '0' saca 'ALTA IMPEDANCIA'
(como si fuera un cable al aire)
(en realidad son GΩs)
se consigue cuando los 2 transistores del Totem-Pole están en corte gracias a la entrada I

	A	B	I	S
	X	X	0	Z
puerta AND	0	0	1	0
	0	1	1	0
	1	0	1	0
	1	1	1	1

Utilidad: varios chips conectados a un mismo bus de datos. Cuando uno de ellos da datos, los demás deben estar desconectados (alta impedancia) lo cual lo puede controlar el µP con la I.

estructura triestado



E = 0 → Tz corte
Todo sigue como si nada
S = $\overline{A}$

si E = 1 → Tz saturación
→ conexión a tierra
→ chupa corriente
↳ T4 corte (diodo)
↳ T2 corte → T3 corte

T3 & T4 corte ⇒ S = Z
alta impedancia

ejercicio: Open Collector → calcular Rext ≡ Rpu

pull up

p open collectors atacan a n TTL's:

Calcular Rpu: $R_{min} < R_{pu} < R_{max}$

→ Rpu debe ser > Rmin para que cuando la salida S sea un '0', no absorban tanta corriente que se quemen, o no tengan suficiente Vce como para salir de saturación

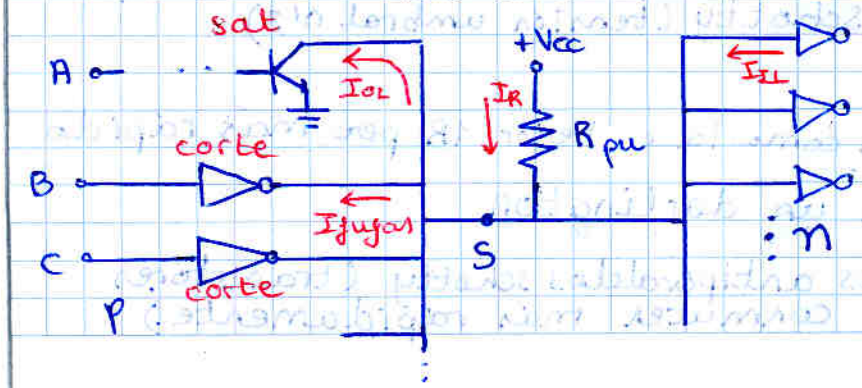
→ Rpu debe ser < Rmax para que permita un '1' en S cuando los open collectors estén todos en corte.

Datos:

$V_{CC} = 5V$	$V_{OH\,min} = 2'4V$
$V_{OL} = 0'2V$	$I_{IH} = 40\mu A$
$I_{IL} = -1'6\,mA$	
$I_{OL} = 16\,mA$	

$I_{fugas\,CE\,corte} = 250\mu A$

• Caso S = '0': Rmin
Peor caso: un sólo transistor en saturación (él se lleva toda la tensión) corriente)



$$I_R + n I_{IL} < I_{OL\,max} + (p-1) I_{fug}$$

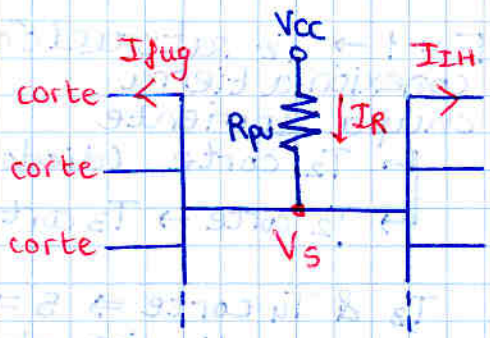
$$I_R = \frac{V_{CC} - V_{OL}}{R_{pu}}$$

$V_{OL} = V_S$

$$\frac{V_{CC} - V_{OL}}{R_{pu}} + n I_{IL} < I_{OL\,max} + (p-1) I_{fug}$$

$$R_{min} = \frac{V_{CC} - V_{OL\,min}}{I_{OL\,max} + (p-1) I_{fug} - n I_{IL}}$$

- caso $S = 1\,1$; R_{max}



$$V_S > V_{OH\,min}$$

$$V_S = V_{CC} - I_R R_{pu}$$

$$I_R = n I_{IH} + p I_{fug}$$

$$V_{CC} - (n I_{IH} + p I_{fug}) R_{pu} > V_{OH\,min}$$

$$R_{pu} < \frac{V_{CC} - V_{OH\,min}}{n I_{IH} + p I_{fug}} = R_{max}$$

$$R_{min} < R_{pu} < R_{max}$$

R_{pu} ↔ R_{min}: más consumo, más rápido

R_{pu} ↔ R_{max}: menos consumo

Siempre se calcula igual:

R_{min}: $I_R + n I_{IL} < I_{OL\,max}$ $(+(p-1) I_{fug})$ $\left(\frac{V_{CC} - V_{OL}}{R}\right)$

R_{max}: $V_S > V_{OH\,min}$ $\left((V_{CC} - (n I_{IH}) R_{pu}) (+ p I_{fug})\right)$

normalmente $I_{fug} = 0$

Familia LSTTL

low-schotty TTL

- bajo consumo
- rápida

En lugar de multiemisor, se vuelve a diodos, pero diodos schotty (tensión umbral 0'3)

Algunas mejoras:

→ R_4, R_7, Q_0 actúan como la anterior R pero más rápido

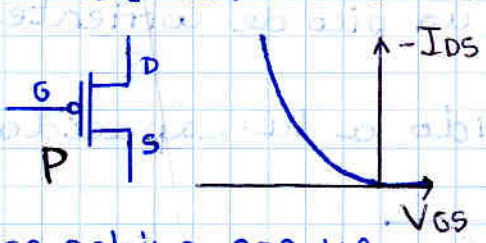
→ Q_3 y Q_4 crean un darlington

→ D_3 y D_4 diodos antiparalelos schotty (trasistores de arriba conmuten más rápidamente)

TRANSISTORES CMOS

- no funciona por corriente sino por tensión
  - ↳ consume menos
  - ↳ grandísimo fan-out
  - ↳ más pequeño (no hay Resistencias)
  - ↳ pega: no puede alimentar cosas que necesiten corriente
- CMOS = complementary MOS
  - ↳ cada N tiene su P
  - ↳ si N en paralelo → P en serie
    si N en serie → P en paralelo

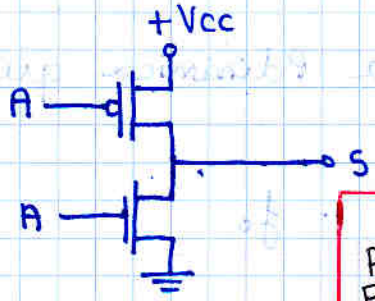
↳ es muy facil implementar funciones lógicas:
↳ es muy facil deducir la función lógica mirando la circuiteria.



se activa con un cero en puerta

se activa con un uno en puerta

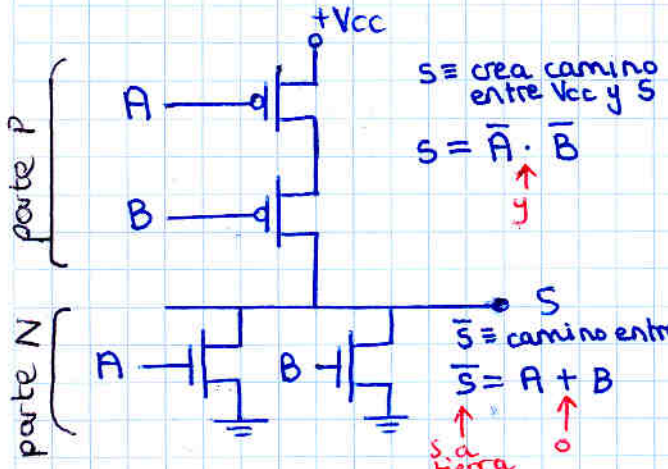
NOT



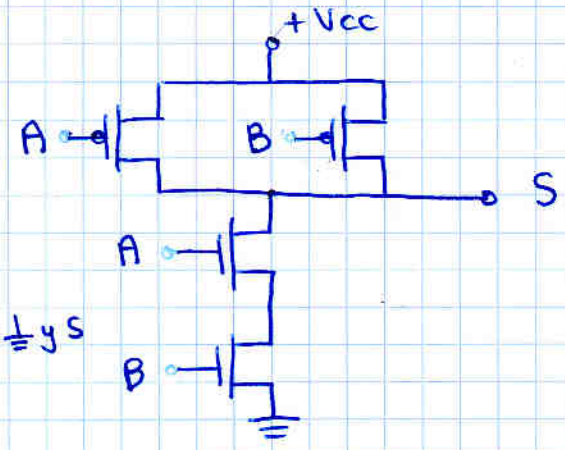
si A=0 → se conectan Vcc y S ⇒ '1'
si A=1 → se conectan ⏚ y S ⇒ '0'

Deducir S facilmente:
Para '1': camino en ON desde Vcc a S
Para '0': camino en ON desde ⏚ a S

NOR $S = \overline{A + B}$



NAND $S = \overline{A \cdot B}$



parte P → PMOS A y B serie
parte N → NMOS A y B paralelo

siempre complementarios

Consumo CMOS

$P = P_{estatica} + P_{dinámica}$

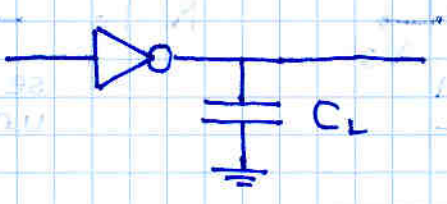
$P_{estatica}$ → no hay corriente, no hay resistores
$P_{estatica} < 1\% \simeq 0$

$P_{dinamica} = P_T + P_L$

60-80% 20-40%

P_T: durante la conmutación mientras PMOS salen de saturación, CMOS entrando en saturación
Durante un instante hay un pico de corriente

P_L: lo que se consume debido a la capacidad parásita



Utilizamos una fórmula para Pdinámica que incluye ambas.

$$P_{dinamica} = C_L \cdot V_{DD}^2 \cdot f_0$$

ejemplo: Implemente mediante función lógica CMOS la siguiente función lógica.

$$F = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}CD$$

Lo primero, comprobamos si es mínima

AB \ CD	00	01	11	10
00	1	0	1	0
01	0	0	0	0
11	0	0	0	0
10	0	0	0	0

por unos:

$$F = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}CD$$
$$= (\bar{A}\bar{B})(\bar{C}\bar{D} + CD)$$
$$\bar{F} = A + B + (C + D)(\bar{C} + \bar{D})$$

por ceros:

$$F = \bar{A}\bar{B}(C + \bar{D})(\bar{C} + D)$$
$$\bar{F} = A + B + \bar{C}D + C\bar{D}$$

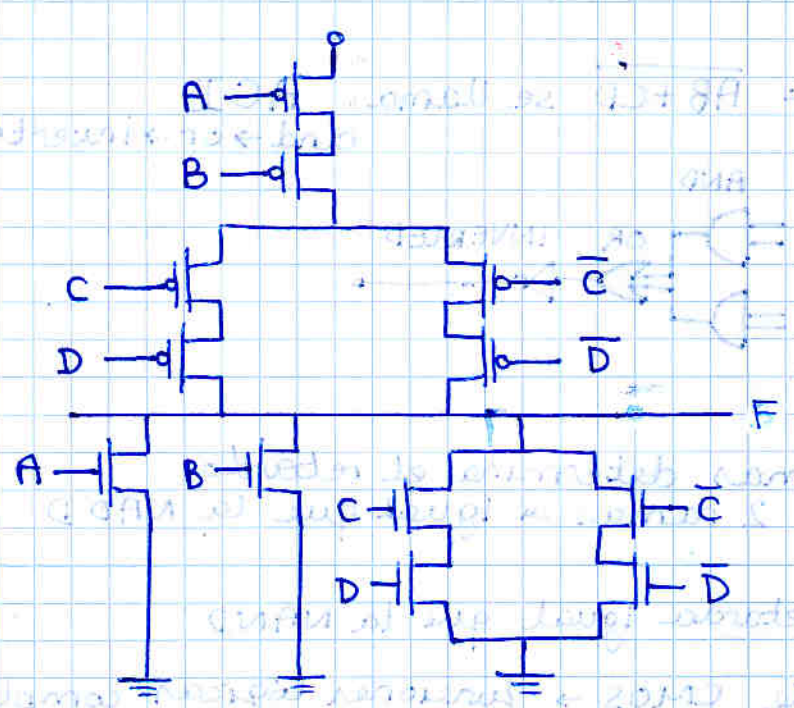
Esas son las 4 formas de expresar F de manera mínima.
Al montar el CMOS tendremos una red p y una red n

Red p → una expresión de F (de las dos posibles)

Red n → una expresión de $\bar{F}$ (de las dos posibles)

En total hay 4 posibilidades para montar la función

Una forma es:



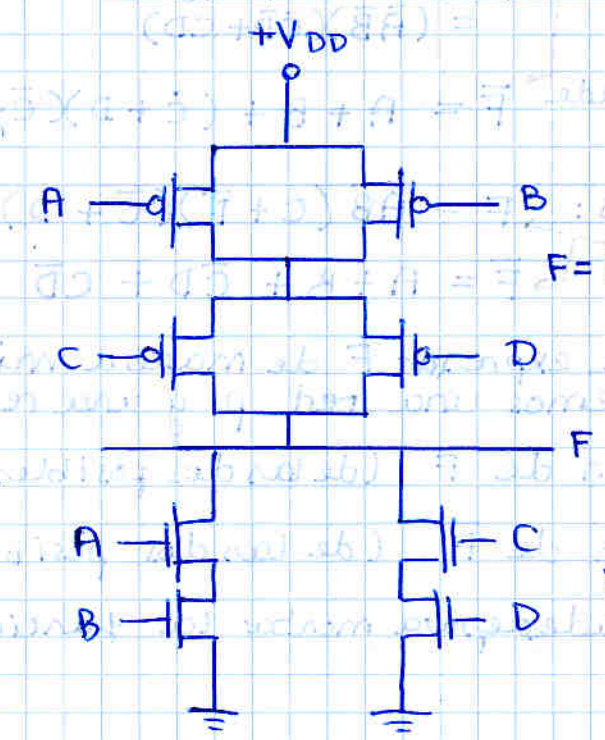
ejemplo:

Implemente con CMOS la función lógica $Z = \overline{AB+CD}$

necesitamos una expresión de Z para la red P
y una expresión de $\bar{Z}$ para la red n

$$Z = \overline{AB} \cdot \overline{CD} = (\bar{A}+\bar{B}) \cdot (\bar{C}+\bar{D})$$

$$\bar{Z} = AB + CD$$



F = (A ó B) y (C ó D)

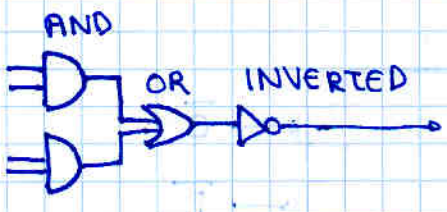
F = (A + B) · (C + D)

$\bar{F}$ = (A y B) ó (C y D)

$\bar{F} = (AB) + (CD)$

$F = \overline{AB+CD}$ se llama AOI

and → or → inverted



Retardo:

El nº de ramas determina el retardo
→ sólo hay 2 ramas → igual que la NAND

AOI retarda igual que la NAND

otra ventaja de CMOS → funciones lógicas complejas retardan poco.

ejercicio: utilizando catálogo TTL y CMOS
↳ (Phillips semiconductors)

Datos de TTL (LS)

$I_{IH\,min} = 20\,\mu A$ $\quad V_{IH\,min} = 2\,V$
$I_{IL\,max} = -0'4\,mA$ $\quad V_{IL\,max} = 0'8\,V$
$I_{OH\,min.} = -0'4\,mA$ $\quad V_{OH\,min} = 2'7\,V$
$I_{OL\,max} = 8\,mA$ $\quad V_{OL\,max} = 0'5\,V$

- Calcular fan-out

$$fan\text{-}out_H = \frac{0'4\,mA}{20\,\mu A} = 20$$

$$fan\text{-}out_L = \frac{8\,mA}{0'4\,mA} = 20$$

fan-out = 20

- Calcular $\Delta 0$, $\Delta 1$ margenes de ruido

$\Delta 1 = 2'7 - 2 = 0'7\,V$
$\Delta 0 = 0'8 - 0'5 = 0'3\,V$

Datos de CMOS (74HC)

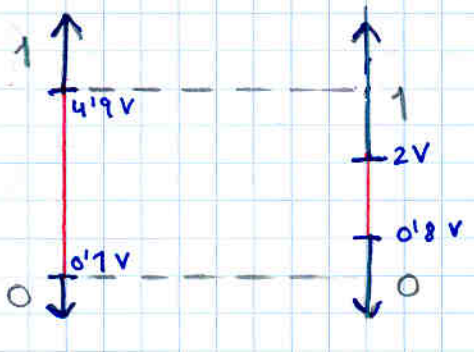
$V_{IH} = 3'1\,V$
$V_{OH} = 4'9\,V$
$V_{OL} = 0'1\,V$

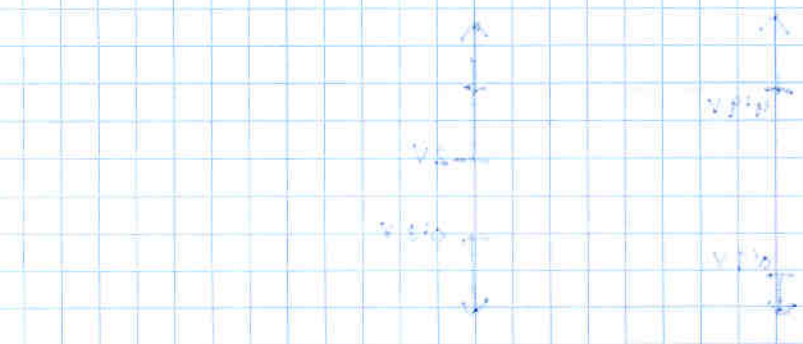
- se puede conectar TTL a CMOS ?

TTL: $V_{OH} = 2'7V$ → CMOS: $V_{IH} = 3'1\,V$ NO

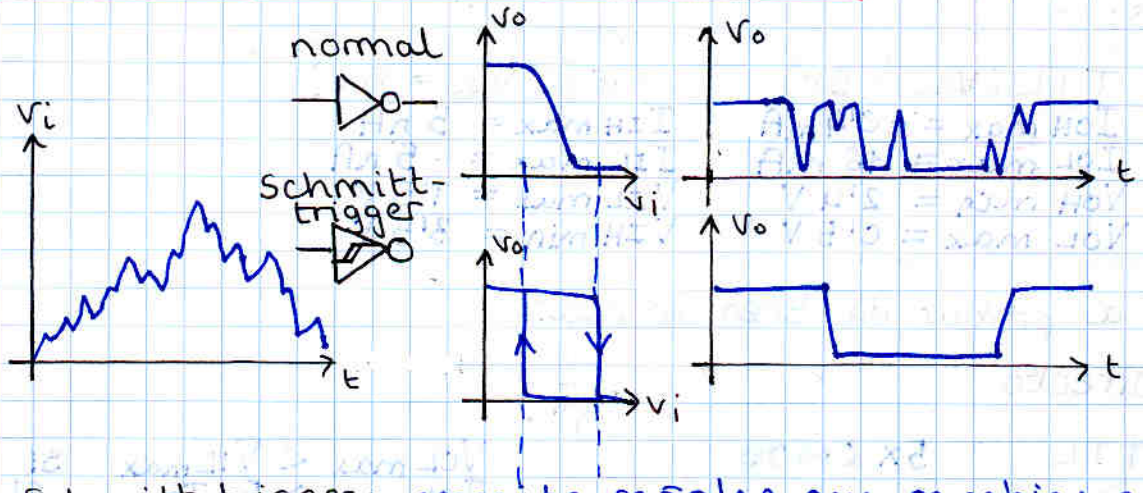
- y CMOS a TTL ?

CMOS: $V_{OH}: 4'9V$, $V_{OL}: 0'1V$ → TTL $V_{IH} = 2V$, $V_{IL\,max} = 0'8\,V$





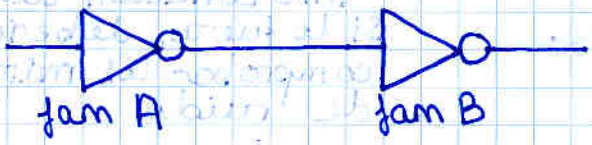
Entradas de transición lenta



Schmitt trigger: permite señales que cambian a velocidades tan lentas como 1 V/s

Problemas de interconexión

Cuando se quieren conectar puertas de distintas familias.

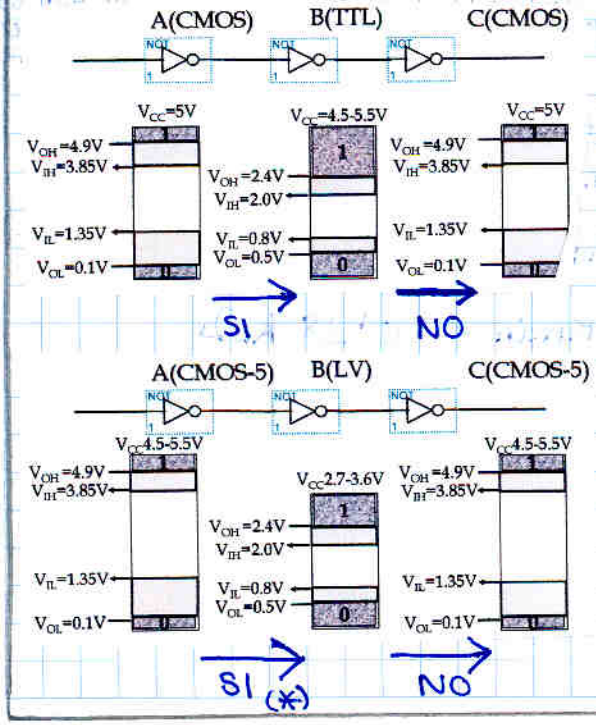


Se debe cumplir

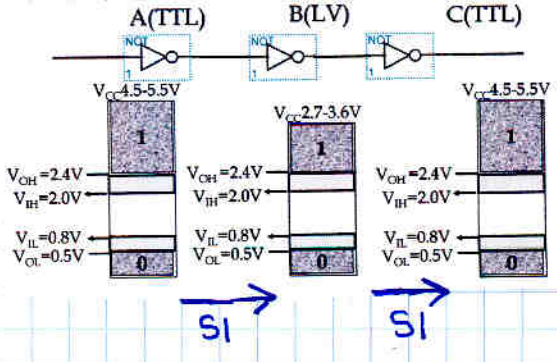
$$V_{OH_A} \geq V_{IH_B}$$
$$V_{OL_A} \leq V_{IL_B}$$

$$|I_{OH_A}| > |I_{IH_B}|$$
$$|I_{OL_A}| > |I_{IL_B}|$$

ejemplos:



LV = low voltage
familia que se alimenta con 3.3 V en lugar de 5 V (menor consumo)



(*) la familia LV lleva diodos de protección por si recibe tensiones altas

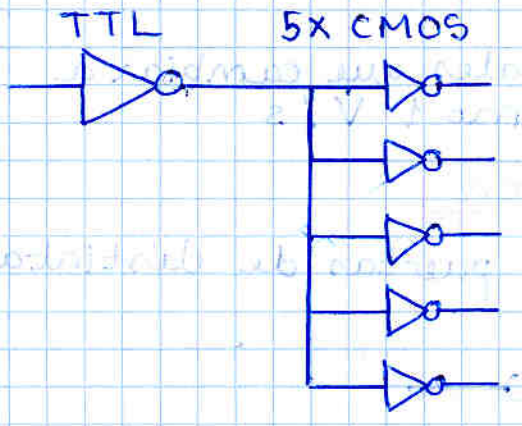
Problema.
Queremos conectar un TTL estandar salida totem-pole a 5 inversores CMOS. Queremos que $\Delta 1 = 1\,V$

Datos:

TTL ($V_{CC} = 5V$)	CMOS ($V_{DD} = 5\,V$)
$I_{OH\,max} = -0'4\,mA$	$I_{IH\,max} = 5\,nA$
$I_{OL\,max} = 16\,mA$	$I_{IL\,max} = -5\,nA$
$V_{OH\,min} = 2'4\,V$	$V_{IL\,max} = 1'5\,V$
$V_{OL\,max} = 0'5\,V$	$V_{IH\,min} = 3'5\,V$

Vamos a probar de tres formas:

a) Directo



"0":

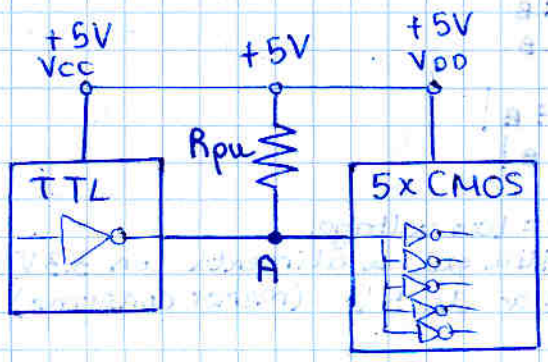
$V_{OL\,max} < V_{IL\,max}$ SI
$I_{OL\,max} > 5 \cdot I_{IL\,max}$ SI

"1":

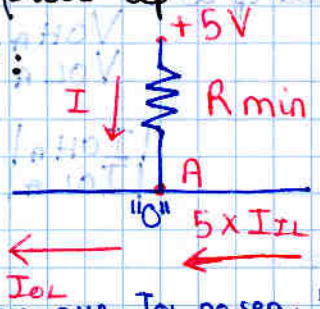
$V_{OH\,min} > V_{IH\,min}$ NO!
$I_{OH\,max} > 5 \cdot I_{IH\,max}$ SI

No es posible la interconexión directa. Si lo fuera, deberíamos comprobar los márgenes de ruido.

b) Mediante una resistencia pull-up



"0":



Rmin para que I_{OL} no sea tan grande que saca al transistor de corte

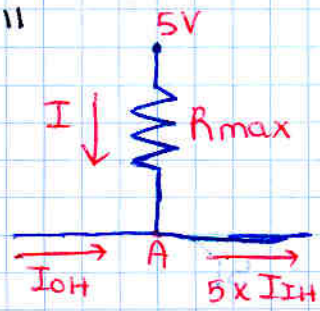
$$I + nI_L \leq I_{OL\,max}$$

$$\frac{5 - V_A}{R_{min}} + 5I_L \leq I_{OL\,max}$$

$$R_{min} \geq \frac{5 - V_A}{I_{OL\,max} - 5I_L}$$

$$R_{min} = 0'28\,k\Omega$$

"1"



$$V_A \geq V_{IH\,min} + \Delta 1$$

$$5 - IR_{max} \geq V_{IH\,min} + \Delta 1$$

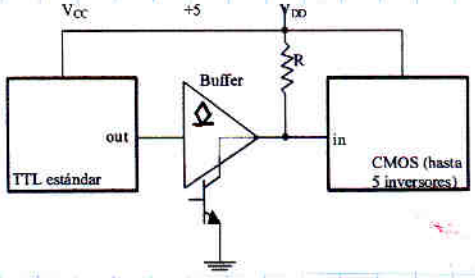
$$R_{max} \leq \frac{5 - (V_{IH\,min} + \Delta 1)}{n \cdot I_{IH} - I_{OH}}$$

$$R_{max} = 20000\,k\Omega$$

c) con un buffer TTL open-collector

$I_{OH\,max} = 0'25\,mA$ (fugas)

$I_{OL\,max} = 16\,mA$



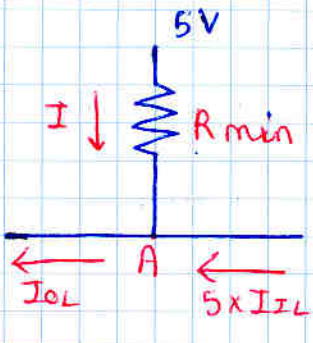
Conexión TTL → TTL-OC

- los niveles de tensión se cumplen
- los niveles de corriente se cumplen (con fanout 1 no hay problema)

Conexión Buffer TTL (open collector) → Cmos

En este caso Rpu es obligatorio (como en toda puerta open-collector)

"0"



para el caso "0", es exactamente igual que en el apartado anterior, pues en el caso "0" una TTL-OC es exactamente igual que una TTL-totem pole.

$$I + nI_{IL} \leq I_{OL\,max}$$

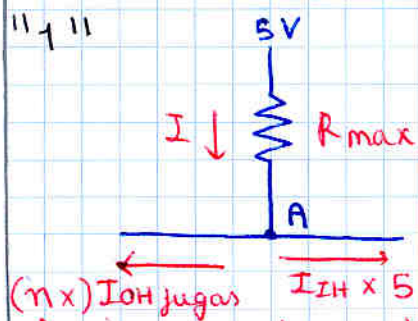
$$\frac{5 - V_A}{R_{min}} + nI_{IL} \leq I_{OL\,max}$$

$$R_{min} = 0'28\,k\Omega$$

Si estuviéramos haciendo uso del cableado lógico de open collector (varias salidas OC) lo cual es un caso típico de problema de examen; la ecuación sería la misma, pues el peor caso es que una única salida open collector esté en "0" y tenga que tragarse toda la corriente (sólo cambiaría por las fugas de las demas salidas)

NOTA: la corriente de fugas $I_{OH\,max}$ es: es logico si recuerdas el funcionamiento
entrante: salidas open-collector
saliente: salidas totem-pole

"1"



↑ Si se conectaran varias salidas OC (cableado lógico)

$$V_A \geq V_{IH\,min} + \Delta 1$$

$$5 - R_{max} I \geq V_{IH\,min} + \Delta 1$$

$$I = (n)I_{OH\,fugas} + 5I_{IH}$$

$$R_{max} \leq \frac{5 - (V_{IH\,min} + \Delta 1)}{nI_{IH} + I_{OH}}$$

$$R_{max} = 2\,k\Omega$$

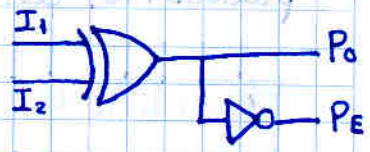
TEMA 4. CIRCUITOS SUBSISTEMAS COMBINACIONALES

· Generador de paridad elemental

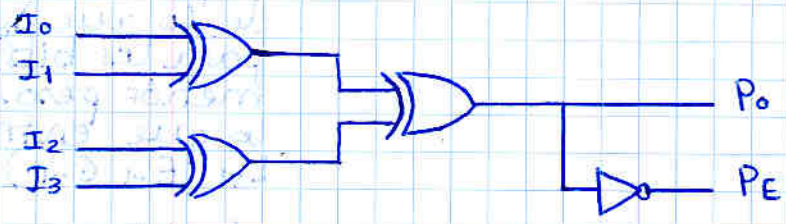
Indica si hay un número par de 1's

- Para 2 bits:

I_0	I_1	P_{EVEN}	P_{ODD}
0	0	1	0
0	1	0	1
1	0	0	1
1	1	1	0



Para 4 bits



· Comparadores (greater, equal, less)

- Palabras de un bit

A B	L E G
0 0	0 1 0
0 1	1 0 0
1 0	0 0 1
1 1	0 1 0

$L = [A < B]$

$E = [A = B]$

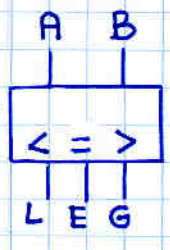
$G = [A > B]$

por minitérminos:

$L = \bar{A}B$

$E = A\bar{B} + \bar{A}\bar{B}$

$G = A\bar{B}$



- Palabras de 4 bits

$a_3a_2a_1a_0 \lessgtr b_3b_2b_1b_0$ por tabla de verdad? NO! 2^8

se hace lógicamente:

$$G(A>B) = (a_3>b_3) + (a_3=b_3)(a_2>b_2) + (a_3=b_3)(a_2=b_2)(a_1>b_1) + (a_3=b_3)(a_2=b_2)(a_1=b_1)(a_0>b_0)$$

Con la notación: $(a_3 > b_3) = G_3$, $(a_2 = b_2) = E_2$ etc...

$$G(A>B) = G_3 + E_3G_2 + E_3E_2G_1 + E_3E_2E_1G_0$$
$$E(A=B) = E_3E_2E_1E_0$$
$$L(A<B) = L_3 + E_3L_2 + E_3E_2L_1 + E_3E_2E_1L_0$$

} se pueden implementar a partir de comparadores de un bit

A veces es más interesante repetir bloques en el dibujo que calcular toda la función para implementar directamente.

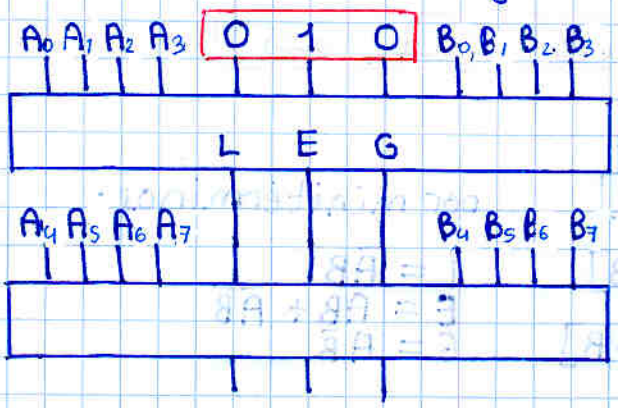
NOTA: Se podría hacer $L = \overline{G} \cdot \overline{E}$, pero entonces dependería de G y de E y el circuito sería más lento (aunq sería más barato)

Comparador de palabras de 8 bits

se construye a partir del siguiente comparador de palabras de 4 bits.

Si los 4 bits de mayor peso:
- $A > B$ → entonces $A > B$ para los 8 bits
- $A < B$ → entonces $A < B$ para los 8 bits
- $A = B$ → dependerá sólo de lo que salga para los bits de menor peso. (Justo lo que entra por L_{-1} E_{-1} G_{-1})

Veamos cómo se construye:



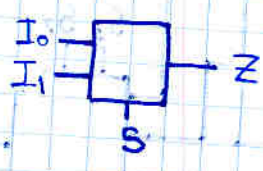
es decir, cuando $A = B$ la salida coincide con L_{-1}, E_{-1}, G_{-1}

Por eso en el comparador inicial $(L_{-1}, E_{-1}, G_{-1}) = (0, 1, 0)$

Multiplexores

A partir de n datos de entrada seleccionar uno de ellos como salida segun unos bits de selección

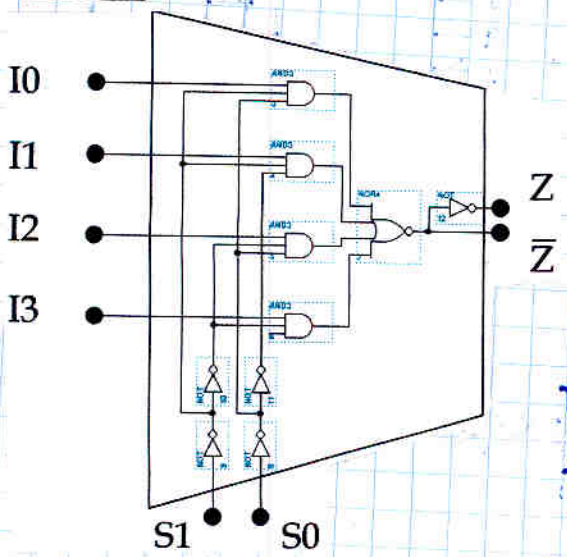
2 a 1



S	Z
0	I_0
1	I_1

$$Z = \bar{S} I_0 + S I_1$$

4 a 1



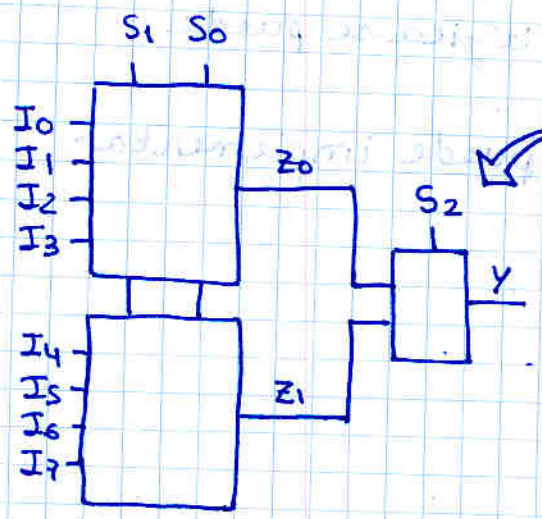
$$Z = (\bar{S_1}\bar{S_0})I_0 + (\bar{S_1}S_0)I_1 + (S_1\bar{S_0})I_2 + (S_1S_0)I_3$$

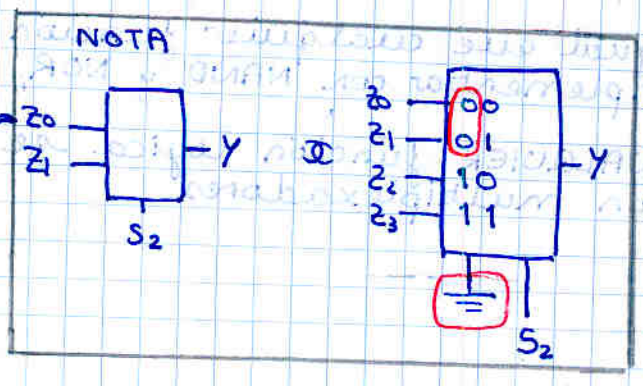
esos diodos que aparentemente son un malgasto, sirven para que desde fuera, el diseñador que meta un bit por esas entradas no tenga que preocuparse del fanout interno.

Nada mas entrar, el inversor "refresca" el chip, para elevarlo a una tensión y corriente aptas

8 a 1

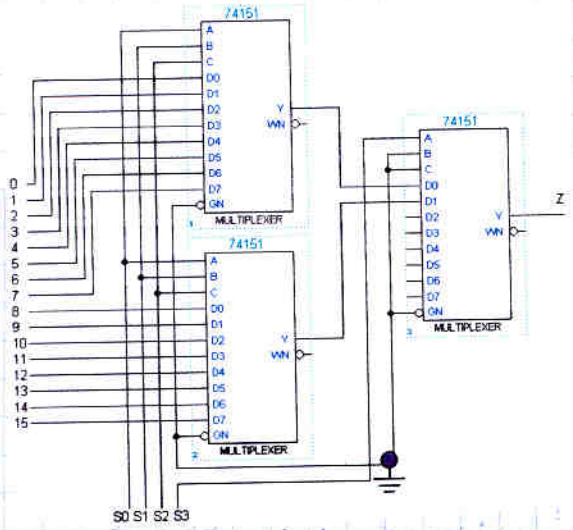
a partir de 4 a 1's



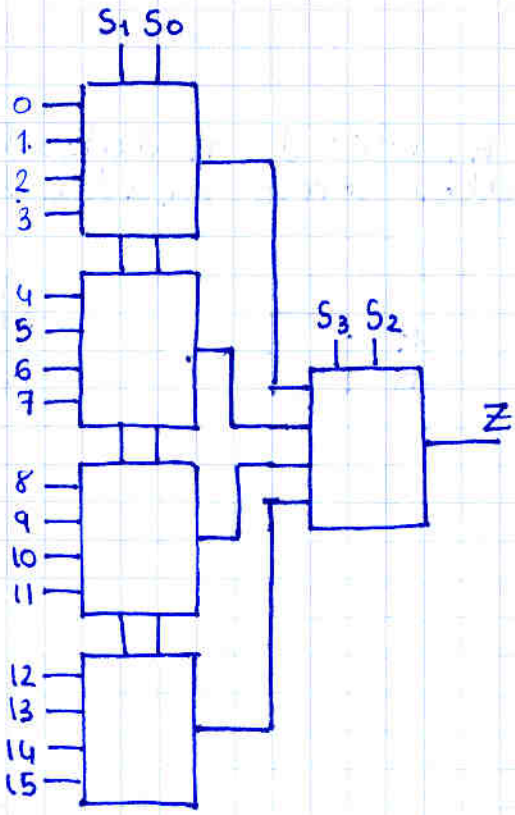


A veces los multiplexadores tienen una linea de control que dice si debe o no funcionar.
Hay veces que los 8 a 1 tienen 2 entradas de control, una para cada 4 a 1.
Todo depende del fabricante.

16 a 1 (a partir de 8 a 1's)



(a partir de 4 a 1's)



Debemos ser capaces de generalizar a n bits utilizando multiplexadores de m bits.

Utilidad de los multiplexadores

- Encaminación de datos:
  - multiplexación en el tiempo
  - conversión paralelo-serie

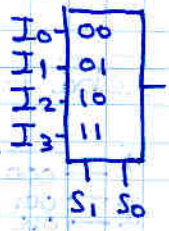
- Implementación de funciones lógicas:

  Igual que cualquier función lógica se puede implementar con NAND y NOR,

  CUALQUIER función lógica se puede implementar con multiplexadores

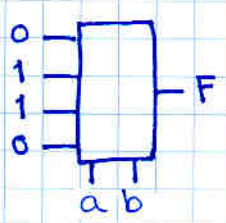
Funciones Lógicas con multiplexores

La F de un multiplexor:

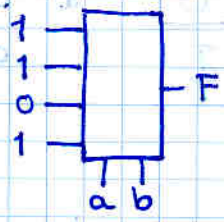


$$F = I_0\bar{S}_1\bar{S}_0 + I_1\bar{S}_1S_0 + I_2S_1\bar{S}_0 + I_3S_1S_0$$

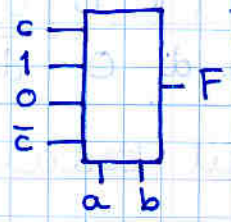
Aplicándolo a casos particulares:



$F = \bar{a}b + a\bar{b}$



$F = \bar{a}\bar{b} + \bar{a}b + ab$



$F = c\bar{a}\bar{b} + \bar{a}b + \bar{c}ab$

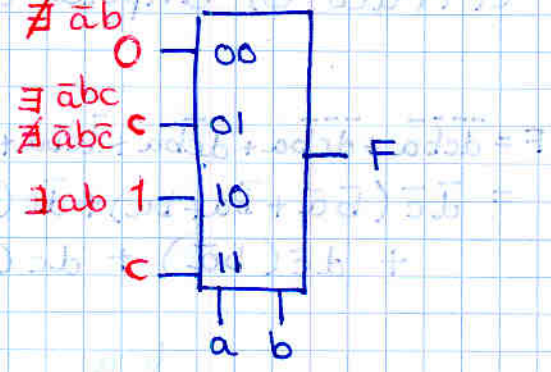
Lo mismo se puede hacer, pero a la inversa, dada una función, saber "su multiplexor"

$F = \bar{a}bc + cb + a\bar{b}$

para no perdernos

$F = \bar{a}bc + \underbrace{cba + cb\bar{a}}_{cb} + \underbrace{a\bar{b}c + a\bar{b}\bar{c}}_{a\bar{b}(c+\bar{c}) = a\bar{b}}$

repe



En general;

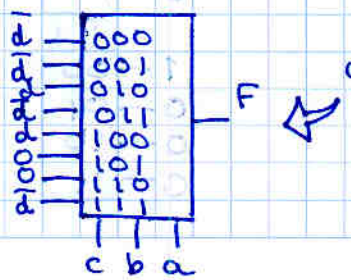
> Con multiplexor de n variables de control se puede implementar cualquier función lógica de n+1 variables

Un truco mejor que expresar F como miniterminos canónicos es hacerse una tabla: (no es Karnaugh)

ej $F(d,c,b,a) = \sum_4 m(0,1,3,7,10,12)$

$$F = \bar{d}\bar{c}\bar{b}\bar{a} + \bar{d}\bar{c}\bar{b}a + \bar{d}\bar{c}ba + \bar{d}cba + d\bar{c}b\bar{a} + dc\bar{b}\bar{a}$$

d \ cba	000	001	010	011	100	101	110	111
0	0 1	1 1	2 0	3 1	4 0	5 0	6 0	7 1
1	8 0	9 0	10 1	11 0	12 1	13 0	14 0	15 0
	↑ d̄	↑ d̄	d	d̄	d	0	0	d̄



compruébalo

En realidad se puede hacer con cualquier orden de las variables (cuidado al poner los miniterminos en la tabla)

$$F = \sum m(0, 1, 3, 7, 10, 12)$$

$$F = \bar{d}\bar{c}\bar{b}\bar{a} + \bar{d}\bar{c}\bar{b}a + \bar{d}c\bar{b}a + \bar{d}cba + d\bar{c}\bar{b}a + dc\bar{b}\bar{a}$$

d \ a c b	000	001	010	011	100	101	110	111
0	0: 1	2: 0	4: 0	6: 0	1: 1	3: 1	5: 0	7: 1
1	8: 0	10: 1	12: 1	14: 0	9: 0	11: 0	13: 0	15: 0
	$\bar{d}$	d	d	0	$\bar{d}$	$\bar{d}$	0	$\bar{d}$

Compruebalo con la expresión de F

Más difícil todavía

Supongamos que nos piden implementar

$F(d, c, b, a) = \sum m(0, 1, 3, 7, 10, 12)$

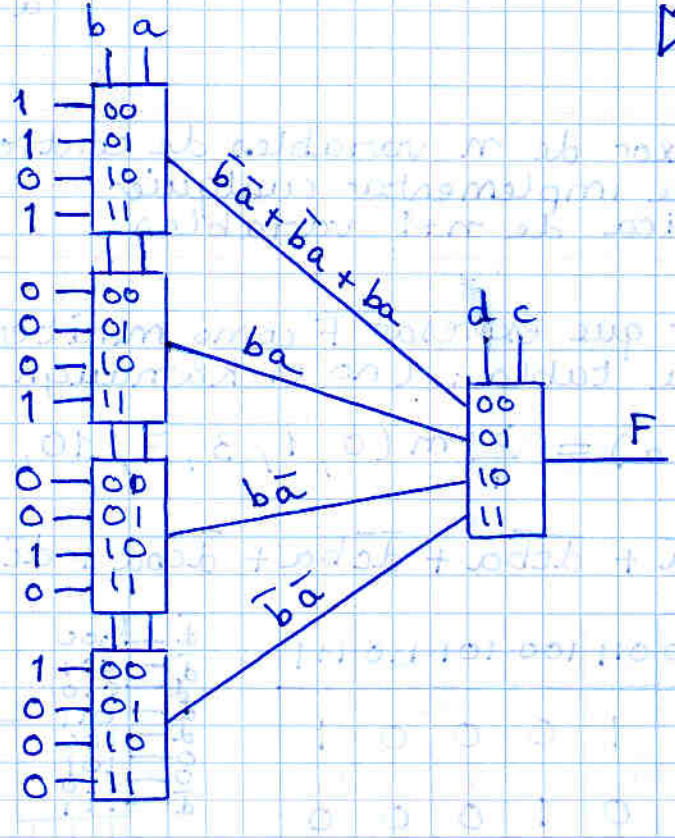
sólo con multiplexadores 4 a 1.

El truco es empezar desde el final

$$F = \bar{d}\bar{c}\bar{b}\bar{a} + \bar{d}\bar{c}\bar{b}a + \bar{d}\bar{c}b\bar{a} + \bar{d}cba + d\bar{c}\bar{b}a + dc\bar{b}\bar{a}$$

$$= \bar{d}\bar{c}(\bar{b}\bar{a} + \bar{b}a + ba) + \bar{d}c(ba) + d\bar{c}(b\bar{a}) + dc(\bar{b}\bar{a})$$

dc
- 00 → $\bar{b}\bar{a} + \bar{b}a + ba$
- 01 → ba
- 10 → $b\bar{a}$
- 11 → $\bar{b}\bar{a}$

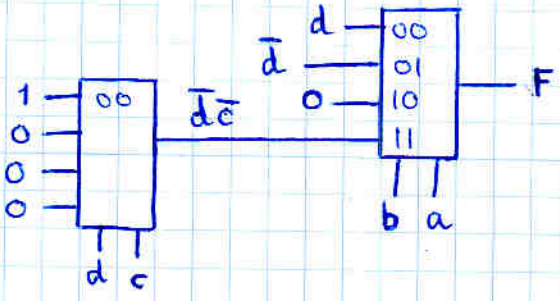


mas ejemplos: funciones lógicas con MUX

ej: $F(d,c,b,a) = \sum_4 m(1,3,5,8,12)$ con 4 a 1's

$= \bar{d}\bar{c}\bar{b}a + \bar{d}c\bar{b}a + d\bar{c}\bar{b}a + d\bar{c}\bar{b}\bar{a} + dc\bar{b}\bar{a}$

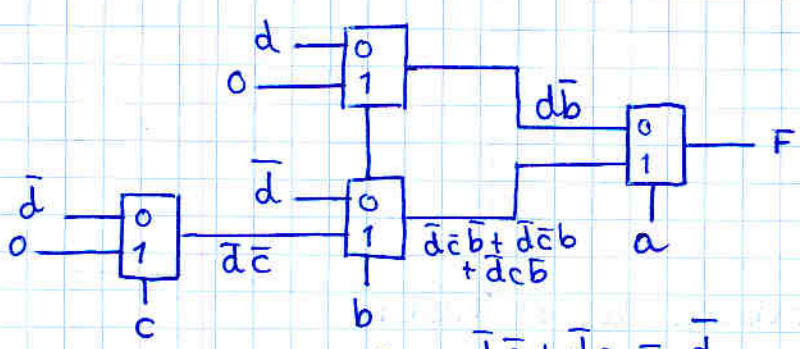
$ba \begin{cases} 00 \to d\bar{c} + dc = d \\ 01 \to \bar{d}\bar{c} + \bar{d}c = \bar{d} \\ 10 \to 0 \\ 11 \to \bar{d}\bar{c} \end{cases}$



ej con 2 a 1's order lineas de control: d c b a

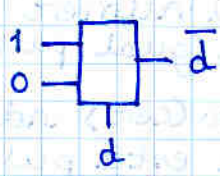
$b \begin{cases} 0 \to d \\ 1 \to 0 \end{cases}$ $a \begin{cases} 0 \to d\bar{c}\bar{b} + dc\bar{b} = d\bar{b} \\ 1 \to \bar{d}\bar{c}\bar{b} + \bar{d}\bar{c}b + \bar{d}cb \end{cases}$

no merece la pena simplificar porque no eliminamos variables.



$c \begin{cases} 0 \to \bar{d} \\ 1 \to 0 \end{cases}$ $b \begin{cases} 0 \to \bar{d}\bar{c} + \bar{d}c = \bar{d} \\ 1 \to \bar{d}\bar{c} \end{cases}$

Para negar variables



ej. Uso del enable

a 0 → multiplexor multiplexa

a 1 → saca algo fijo según fabricante

Si una variable sale como factor común de toda una función, se puede utilizar el enable.

ej: $F(edcba) = \sum m(0,8,13,14,15)$

menor que 15 → 'e' vale siempre cero → e en enable

suponiendo que con enable=1 saca cero

$F(edcba) = \sum m(1,7,9,17)$

todos impares → 'a' vale siempre 1 → $\bar{a}$ en enable

Codificadores

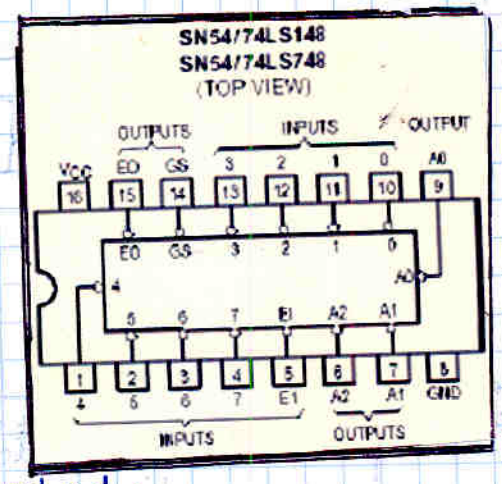
Reducen m entradas a n salidas
ej: decimal → BCD

Sin prioridad: si se activa >1 entrada, no funciona (salen cosas absurdas)

$I_9 I_8 I_7 I_6 I_5 I_4 I_3 I_2 I_1 I_0$	D C B A
0 0 0 0 0 0 0 0 0 1	0 0 0 0
0 0 0 0 0 0 0 0 1 0	0 0 0 1 $D = I_8 + I_9$
0 0 0 0 0 0 0 1 0 0	0 0 1 0
0 0 0 0 0 0 1 0 0 0	0 0 1 1 $C = I_4 + I_5 + I_6 + I_7$
0 0 0 0 0 1 0 0 0 0	0 1 0 0
0 0 0 0 1 0 0 0 0 0	0 1 0 1 $B = I_2 + I_3 + I_6 + I_7$
0 0 0 1 0 0 0 0 0 0	0 1 1 0
0 0 1 0 0 0 0 0 0 0	0 1 1 1 $A = I_1 + I_3 + I_5 + I_7 + I_9$
0 1 0 0 0 0 0 0 0 0	1 0 0 0
1 0 0 0 0 0 0 0 0 0	1 0 0 1

Con prioridad

INPUTS									OUTPUTS				
EI	0	1	2	3	4	5	6	7	A2	A1	A0	GS	EO
H	X	X	X	X	X	X	X	X	H	H	H	H	H
L	H	H	H	H	H	H	H	H	H	H	H	H	L
L	X	X	X	X	X	X	X	L	L	L	L	L	H
L	X	X	X	X	X	X	L	H	L	L	H	L	H
L	X	X	X	X	X	L	H	H	L	H	L	L	H
L	X	X	X	X	L	H	H	H	L	H	H	L	H
L	X	X	X	L	H	H	H	H	H	L	L	L	H
L	X	X	L	H	H	H	H	H	H	L	H	L	H
L	X	L	H	H	H	H	H	H	H	H	L	L	H
L	L	H	H	H	H	H	H	H	H	H	H	L	H

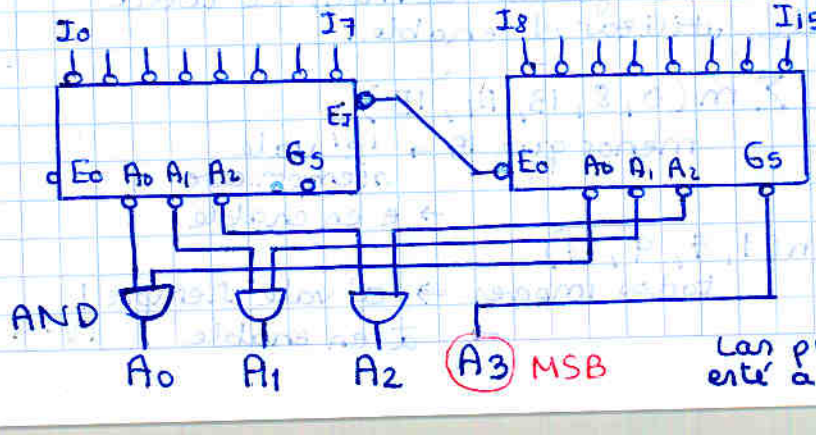


Analicemos su tabla de verdad:

- Salidas activas a nivel bajo
- Entradas activas a nivel bajo
- Es más prioritario el número cuanto más grande sea (si se pulsa el 4, da igual pulsar el 0, 1, 2, 3)

Salidas: EO → activo (cero) indica que, estando encendido, no se está pulsando nada
GS → activo (cero) cuando se está codificando
inactivo: deshabilitado o nada pulsado

- Aumento nº bits: codif. 16 a 4



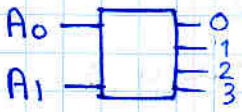
Como queremos que haya prioridad: conectamos EO → EI
Así sólo hacemos caso al de la izq. cuando nada pulsado en el de la derecha

El MSB diferencia entre entradas $I_0 \to I_7$ y $I_8 \to I_{15}$
Es decir, es GS

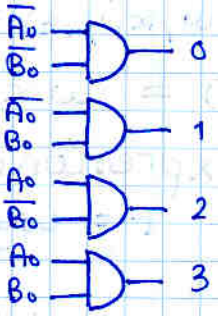
Las puertas AND hacen caso al que esté activo (marca cero)

Decodificadores

ej. decodificador 2 a 4 (binario → decimal)

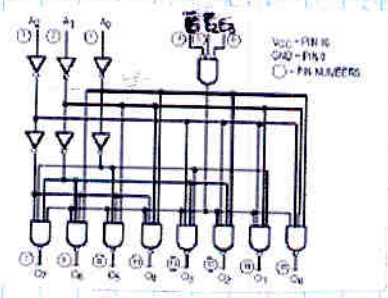


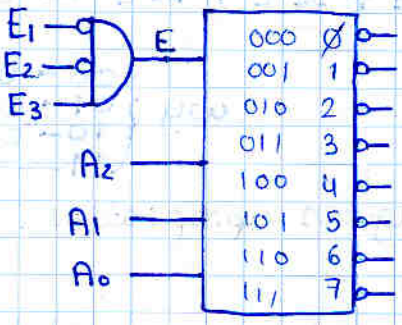
$0 = \overline{A_0}\,\overline{B_0}$
$1 = \overline{A_0}\,B_0$
$2 = A_0\,\overline{B_0}$
$3 = A_0\,B_0$



Son muy sencillos.
Cada salida es una función de un mintérmino canónico.

ej. decodificador 3 a 8 (74138)



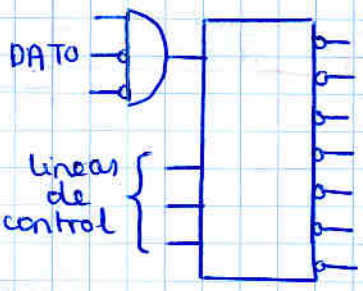


el Enable $E = \overline{E_1}\,\overline{E_2}\,E_3$ es decir, se activa con 2 a nivel bajo y 1 a nivel alto.

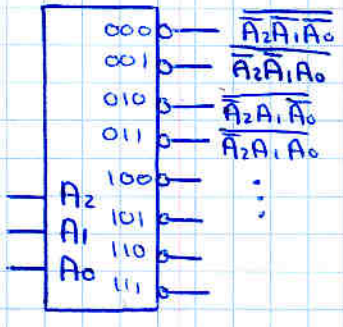
Aplicaciones:

- Demultiplexor:

La única dificultad es pensar dónde hay que poner el dato.
Razonando es sencillo
se mete en un enable negado.
Si vale 0 → sale 0 por el correspondiente
si vale 1 → sale 1 por todos



- Generación de funciones lógicas:



Como las salidas son todo mintérminos canónicos negados, se puede aprovechar para hacer funciones lógicas.

Ideal para conectarlos a MUX cuyas entradas requieren mintérminos negados.

Problema: (el más difícil de la asignatura)

Implementar mediante multiplexor doble 4 a 1 (74153) y decodificador 3 a 8 (74138) las funciones

$$F_1(w,x,y,z) = \sum m(10,14,15)$$
$$F_2(w,x,y,z) = \prod M(1,2,8,10)$$

a) $F_1(w,x,y,z) = w\bar{x}y\bar{z} + wxy\bar{z} + wxyz$

1. Sacamos expresiones del multiplexor.

$$F = wy(\bar{x}\bar{z} + x\bar{z} + xz)$$

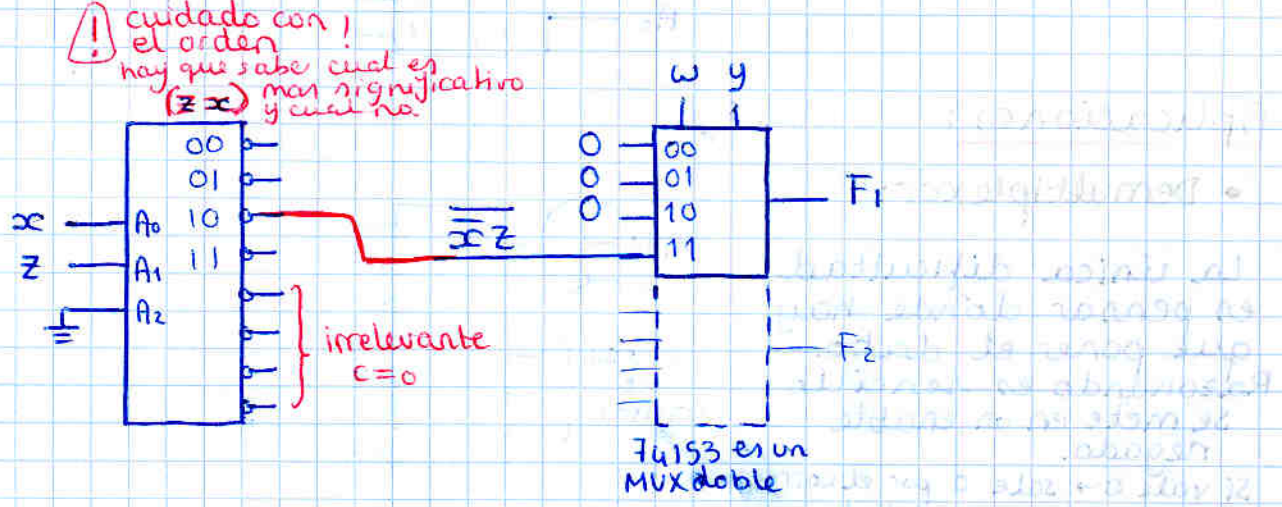
2. Necesitamos miniterminos negados

wy { 11 → $\bar{x}\bar{z} + x\bar{z} + xz$; 10 → 0 ; 01 → 0 ; 00 → 0

TRUCO 1: $\bar{x}\bar{z} + x\bar{z} + xz = \overline{\bar{x}z}$ es la combinación q falta

wy { 00 → 0 ; 01 → 0 ; 10 → 0 ; 11 → $\overline{\bar{x}z}$

3. Conectar en lugares apropiados



b) $F_2(w,x,y,z) = \Pi M(1,2,8,10)$

1. Expresarlo como minitérminos (para el MUX)

$F_2 = \sum m(0,3,4,5,6,7,9,11,12,13,14,15)$

no caben en el mux! (sólo caben 4 minitérminos)

Truco 2: $\overline{F_2}(w,x,y,z) = \sum m(1,2,8,10)$

2. Sacar expresiones del MUX

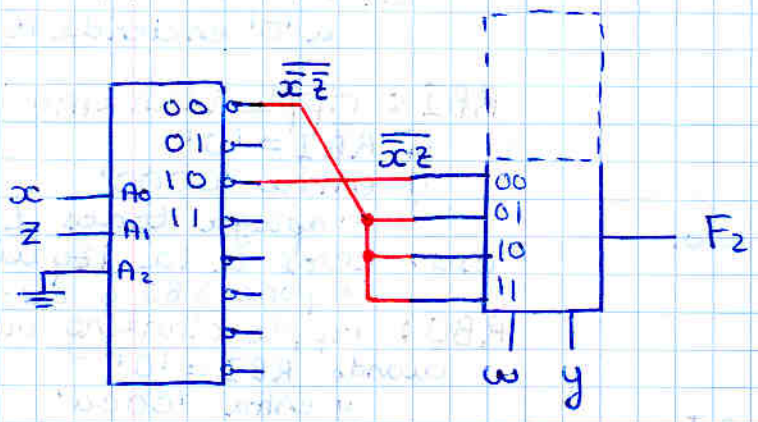
$\overline{F_2} = \bar{w}\bar{x}\bar{y}z + \bar{w}\bar{x}y\bar{z} + w\bar{x}\bar{y}\bar{z} + w\bar{x}y\bar{z}$

$\overline{F_2}$ wy $\begin{cases} 00 \rightarrow \bar{x}z \\ 01 \rightarrow \bar{x}\bar{z} \\ 10 \rightarrow \bar{x}\bar{z} \\ 11 \rightarrow \bar{x}\bar{z} \end{cases}$

Truco 3: Si un multiplexor saca F, sacará $\overline{F}$ sin más que negar cada entrada

F_2 wy $\begin{cases} 00 \rightarrow \overline{\bar{x}z} \\ 01 \rightarrow \overline{\bar{x}\bar{z}} \\ 10 \rightarrow \overline{\bar{x}\bar{z}} \\ 11 \rightarrow \overline{\bar{x}\bar{z}} \end{cases}$

es perfecto para implementar con el decodificador, que saca minitérminos negados.

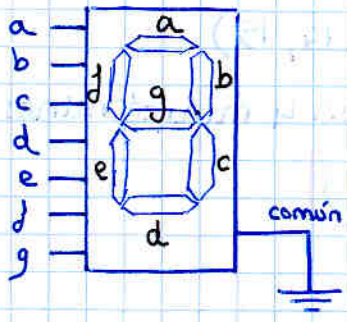


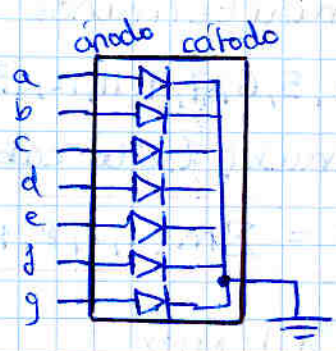
Se puede implementar F_1 y F_2 a la vez (el multiplexor es doble)

Decodificador y display BCD → 7segmentos

Tipos de display 7 segmentos:

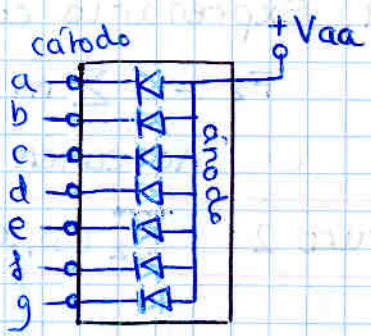
- Cátodo común





lo que se ilumina son simplemente LED's numerados de la a a la g.

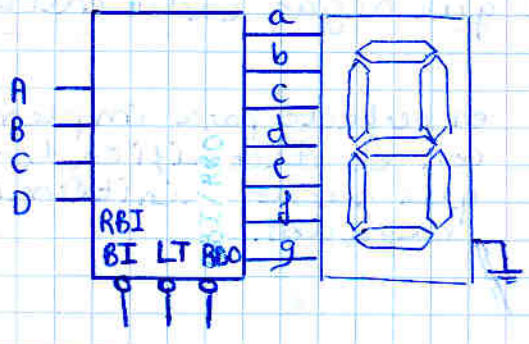
- Ánodo común



activo a nivel bajo (el led x se enciende con '0' en la entrada x)

- Decodificador BCD → 7 segmentos (7448)

activo a nivel alto → se une a cátodo común



tiene tres entradas extra (a nivel bajo)

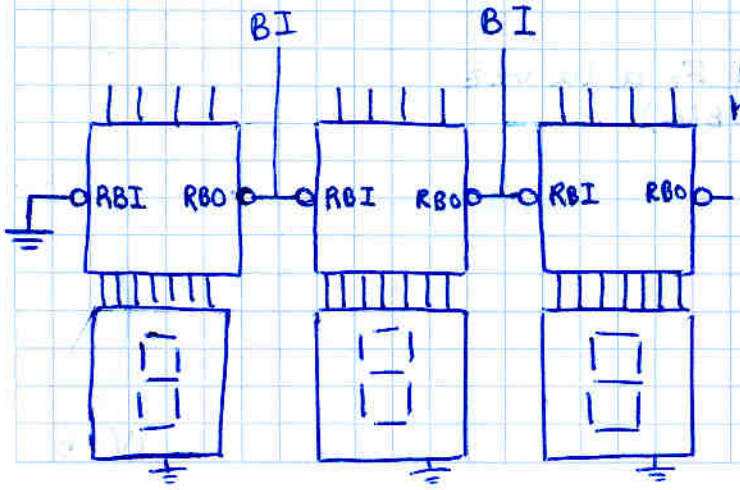
BI: a '0' apaga todos LED
↳ Blanking input

LT: lamp test
a '0' enciende todos LED

Detalle:
misma entrada RBI y BI. No nos preocupemos por eso.

RBI: ripple blanking input
si RBI = '0'
y entra '0000'
↳ apaga todos LED's
uso: ceros a la izquierda
↳ pone RBO a cero

RBO: ripple blanking output
cuando RBI = '0'
y entra '0000'
entonces RBO = '0'
(para poder 'propagar' el hecho de que haya cero a la izquierda)

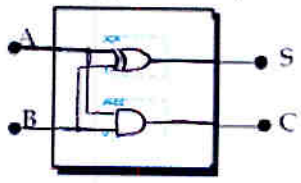


Sumadores

Semisumador : Half adder

DISEÑO: Semisumador (Half adder)

A B	S C
00	0 0
01	1 0
10	1 0
11	0 1



$$S = A \oplus B = \overline{A}B + A\overline{B}$$

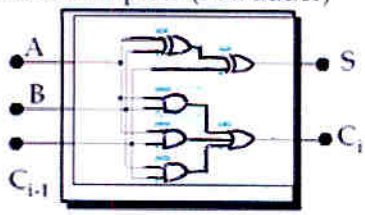
$$C = AB$$

Sólo puede sumar 2 bits ya que no tiene entrada de acarreo

Sumador Completo Full adder

DISEÑO: Sumador completo (Full adder)

A B C_{i-1}	S C_i
0 0 0	0 0
0 0 1	1 0
0 1 0	1 0
0 1 1	0 1
1 0 0	1 0
1 0 1	0 1
1 1 0	0 1
1 1 1	1 1

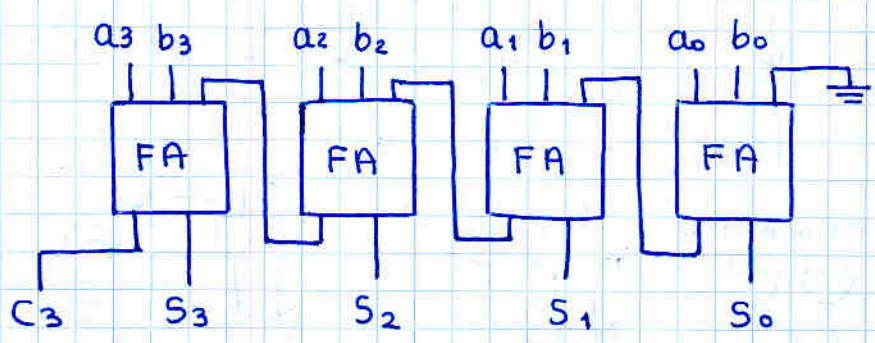


$$S = A \oplus B \oplus C_{i-1}$$

$$C_i = AB + AC_{i-1} + BC_{i-1}$$

Sumador de dos palabras en paralelo con acarreo serie.

Permite sumar n-bits mediante un circuito muy facil de dibujar y entender.



Este método imita el procero que hacemos nosotros a mano.

Pegas: supongamos t_{pd} = 5 ns (en cada puerta)

en cada FA, 2 puertas : retraso 10 ns

Para una suma de palabras de n bits

retardo = 10n ns

¡El sumador es la parte de la CPU que mas se utiliza!
¡Debe ser rapidísimo!

Soluciones:

- ↳ tecnológica: la cadena de acarreo con los mejores cables y puertas
- ↳ diseño: que la suma S_n no dependa de C_{n-1}
  - ↳ acarreo anticipado

Acarreo anticipado: Concepto

Al sumar cada 2 bits de cada una de las palabras, que calcule en ese instante el acarreo sin esperar a que lo saquen los bits anteriores.

Sumador paralelo acarreo paralelo: Carry Look Ahead

» Sumador paralelo acarreo paralelo: "Carry Look-Ahead"

A_i	B_i	C_{i-1}	S_i	C_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$S_i = A_i \oplus B_i \oplus C_{i-1}$$

$$C_i = A_i B_i + A_i C_{i-1} + B_i C_{i-1} =$$

$$\boxed{C_i = G_i + P_i \cdot C_{i-1}}$$

P_i=función propagación → si el acarreo venía de la suma anterior y se ha propagado

$$P_i = A_i \oplus B_i$$

G_i=función generación → si el acarreo se ha generado en la propia suma actual

$$G_i = A_i \cdot B_i$$

Básicamente, si desarrollamos C_i de forma 'recursiva' se obtiene

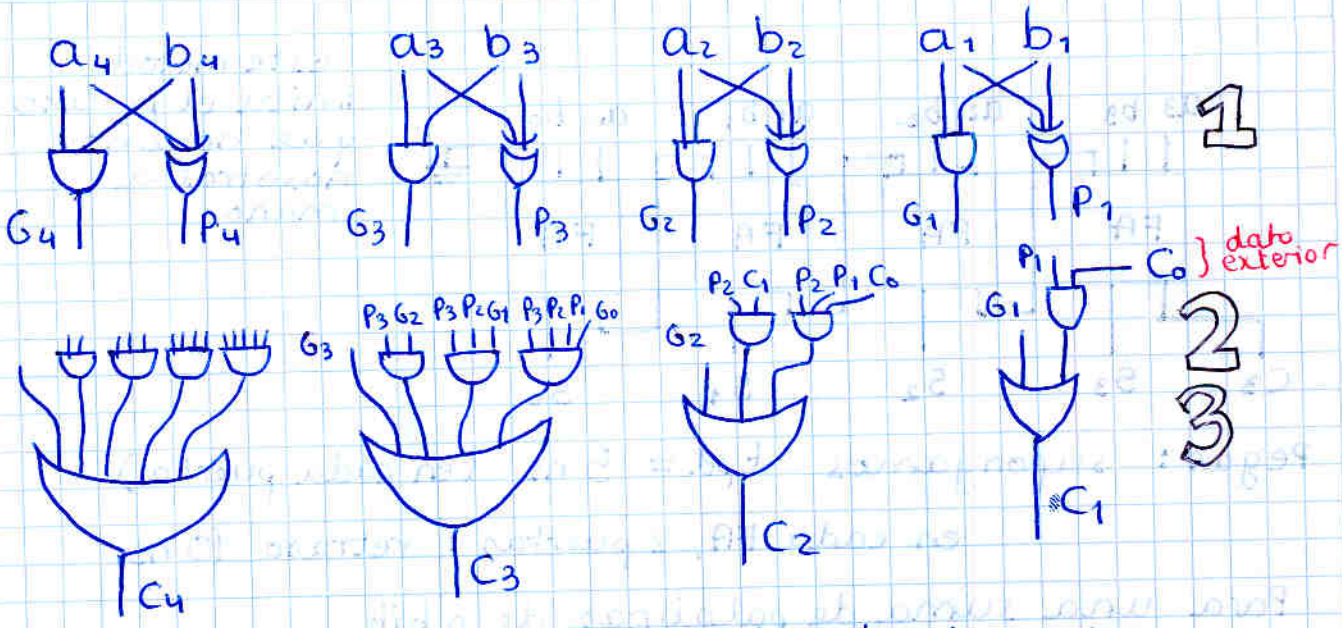
$C_1 = G_1 + P_1 C_0$

$C_2 = G_2 + P_2 C_1 = G_2 + P_2 G_1 + P_2 P_1 C_0$

$C_3 = G_3 + P_3 C_2 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_0$

$C_4 = G_4 + P_4 C_3 = G_4 + P_4 G_3 + P_4 P_3 G_2 + P_4 P_3 P_2 G_1 + P_4 P_3 P_2 P_1 C_0$

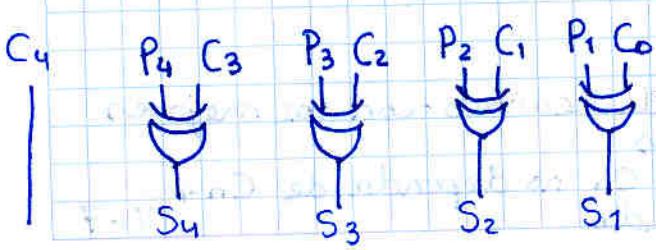
Así podemos calcular todos los acarreos en paralelo y sin depender unos de otros



Obtenemos los acarreos con tan sólo 3 niveles de puertas lógicas.

Ahora, la suma $\boxed{S_i = P_i \oplus C_{i-1}}$

$P_i \oplus C_{i-1} = \overline{A_i \oplus B_i}$



C_0 ≡ acarreo del exterior (útil para restar)

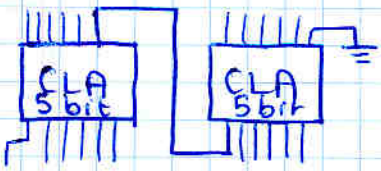
Obtenemos la suma en 4 niveles

Ventajas:
- Con 4 niveles de puertas tenemos la suma de dos palabras de cualquier nº de bits.
  - ↳ Sólo tarda 4tpd

Desventajas:
- Necesita puertas de n entradas para palabras de n bits (aunque son igual de rápidas (casi) son más caras)
- más caro
- más grande
- consume más

Sumador mixto

Para no tener carry-look-ahead's (CLA) enormes. se hacen CLA's menores con acarreo en serie.
Es llegar a un compromiso entre velocidad y tamaño.
Si 1 CLA tarda 4 tpd (tpd ≡ tiempo de propagación de una puerta)
n CLA tardan 4n tpd



Restador

Nota importante:

Una suma de palabras de n bits da como resultado n bits más uno de acarreo.

En la suma binaria, para ver si hay 'overflow' habría que mirar el bit de acarreo.

Normalmente no se usa la suma binaria, sino el complemento a dos.

Complemento a dos:

(+5)	0	1	0	1	(-5)	1	0	1	1
+ (+2)	0	0	1	0	+ (+2)	0	0	1	0
(+7)	0	1	1	1	(-3)	1	1	0	1

En la suma con Ca-2, el bit de acarreo se ignora por completo. Ni siquiera sirve para comprobar 'overflow'

(+5)	0	1	0	1	(-5)	1	0	1	1
+ (-2)	1	1	1	0	+ (-2)	1	1	1	0
(+3)	1	0	0	1	1 (-7)	1	1	0	0	1

ignorar ↑ ignorar ↑

Overflow ⇔ sumar 2 positivos da negativo
sumar 2 negativos da positivo

Para comprobar 'overflow' en una operación Ca-2 se mira el bit de mayor peso de la suma.

(7)	0	1	1	1	overflow!
+ (6)	0	1	1	0	suma 1101 en
(-3)	1	1	0	1	binario SI es 13

para restar se utiliza el propio sumador pero con los números en Ca-2

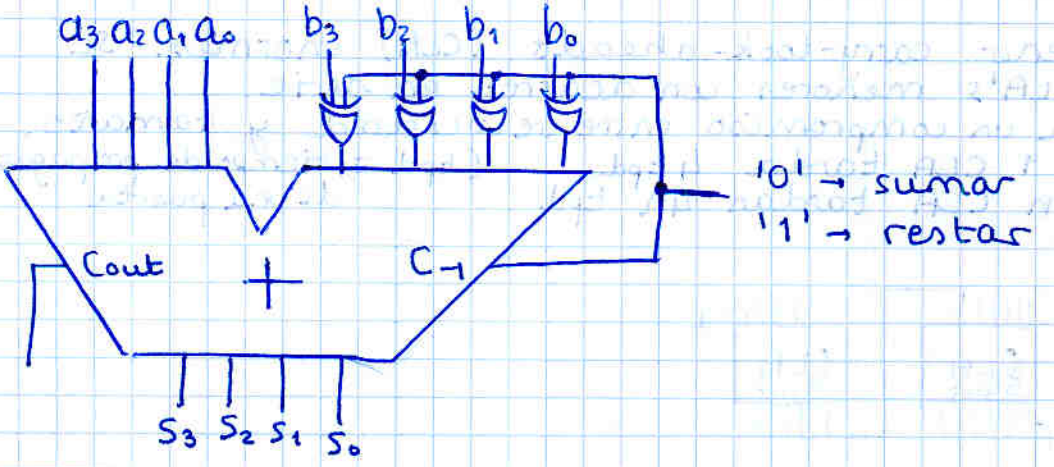
Para conseguir un número negativo se hace uso de la

transformación C2

$2 \equiv 0010$ → (se cambia cada bit) 1101 (está en Complemento a 1) → (se suma uno) $1110 \equiv -2$

con puerta XOR

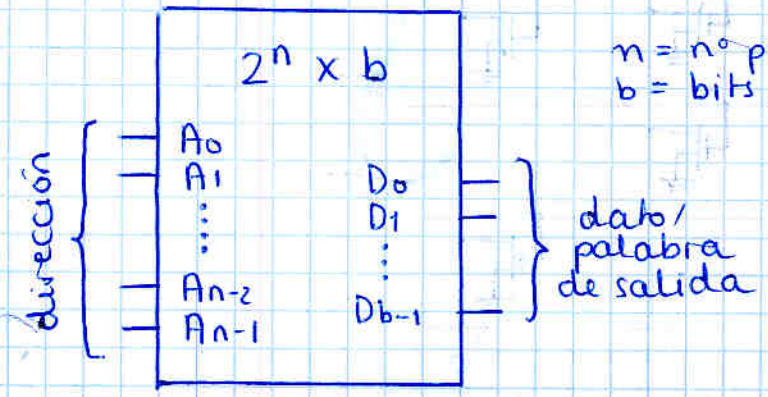
con bit de acarreo C_{-1}



Memorias de sólo lectura (ROM)

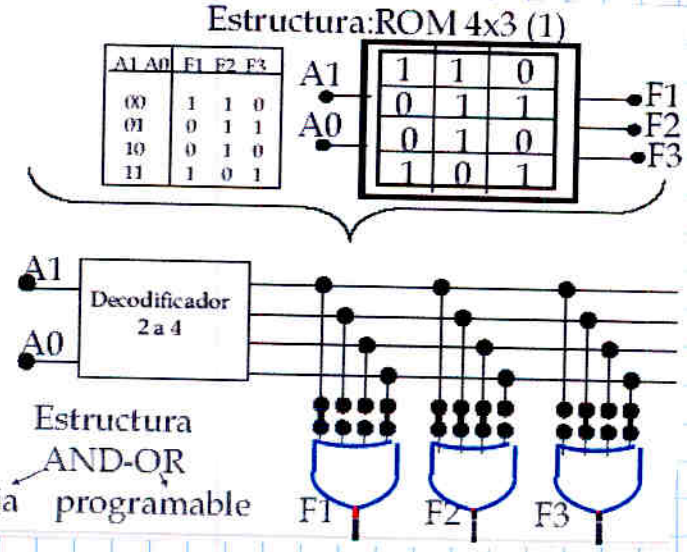
Almacena información estática (para siempre)

Utilidad:
- Boot ROM del PC
- Almacenamiento de una aplicación completa
- Generación de funciones lógicas



n = nº palabras
b = bits/palabra

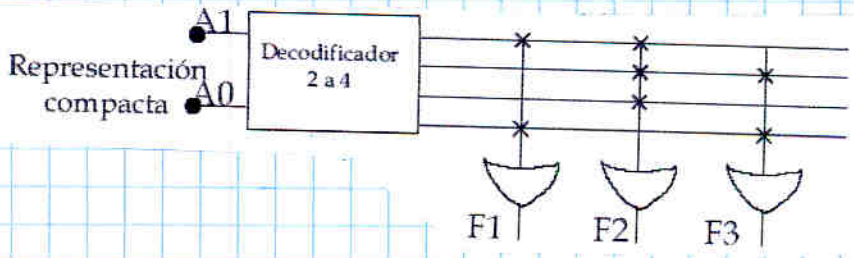
ejemplo: ROM 4 palabras de 3 bits
básicamente son 3 funciones de salida (una para cada bit)



se llama AND — fija → el decodif. está hecho con puertas AND

OR programable → cada bit de salida se consigue con una puerta OR de n entradas. Las patillas están conectadas o no según el dato en memoria

una mejor representación es la siguiente:

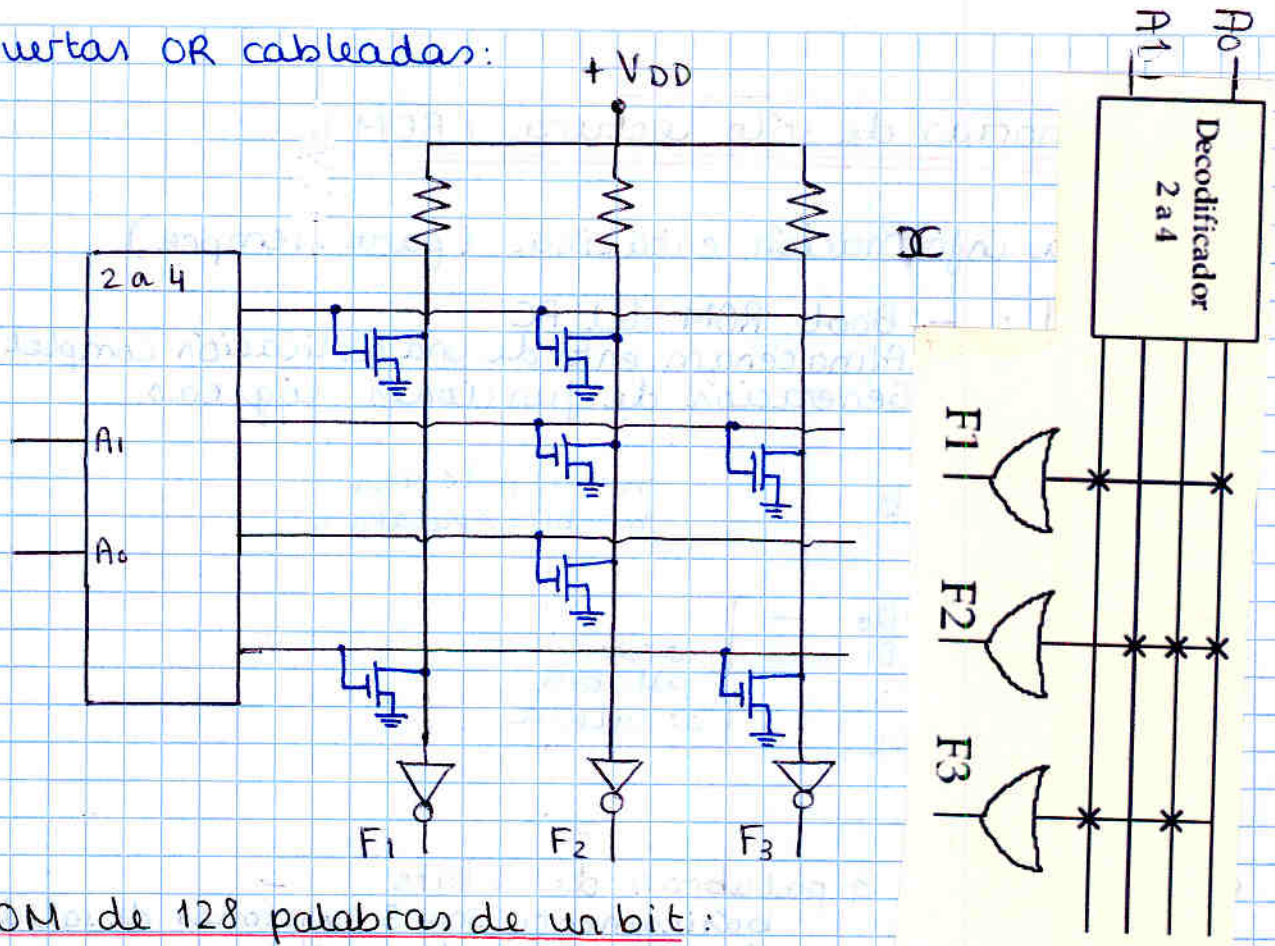


{F1, F2, F3} es la palabra de 3 bits almacenada en la dirección A1 A0

Vemos una clara pega: puertas OR de n entradas!
n puede ser 512, 1024, ...

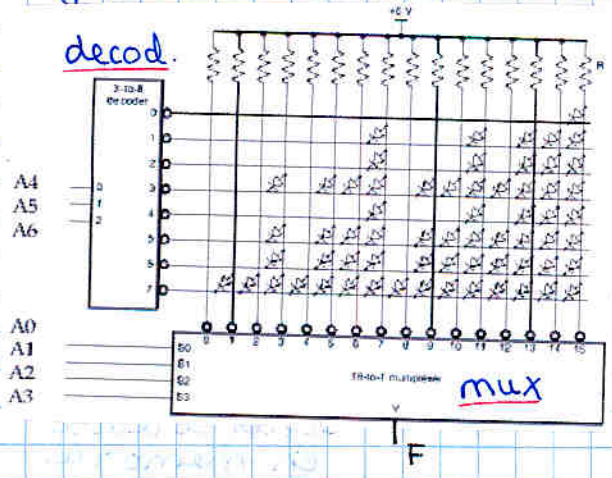
Solución: puertas OR cableadas

puertas OR cableadas:



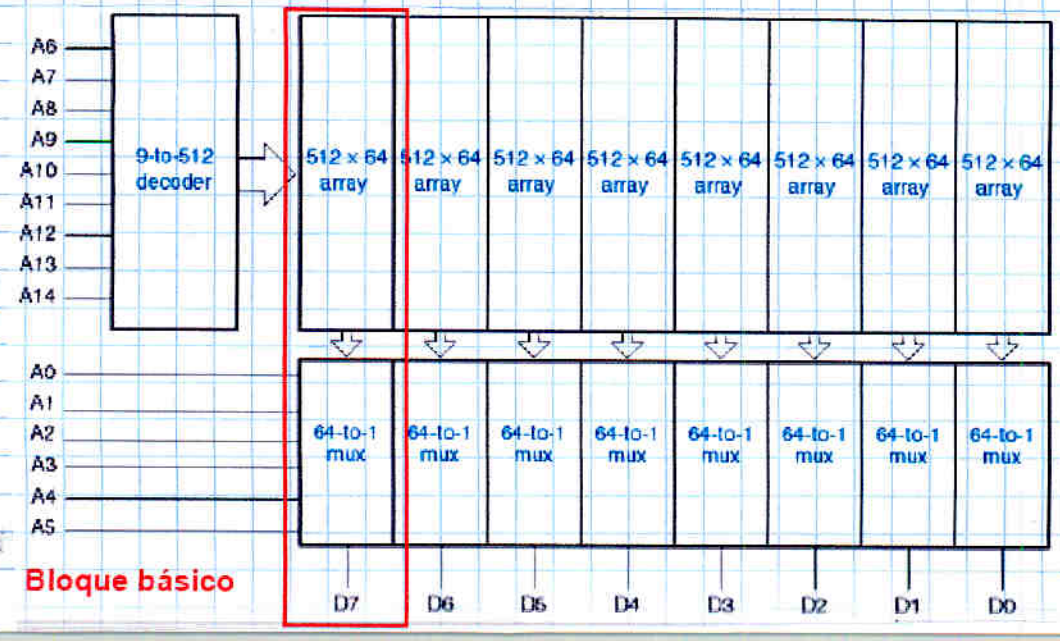
ROM de 128 palabras de un bit:

No se hace con decodificador 7 a 128 porque es un cho rizo. En digital les gusta lo cuadrado.



los diodos actuan similar a un transistor.
Para comprender funcionamiento, ponte ejemplos.
(Recuerda que lo de abajo es un MUX con ENTRADAS a nivel bajo)

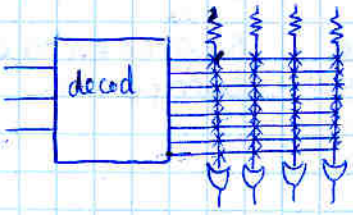
ROM de 2^{15} palabras de 8 bits. → (ROM 32 kB)



Tipos de ROM

Cuando la ROM se vende, están todos los transistores.

Lo que se hace al programarla es deshabilitar los que no se necesitan.



OTP / PROM: se programan quemando fusibles. Programable una única vez

EPROM: Tensión al transistor
Programable Electricamente
↳ se forma barrera de portadores que lo inhabilita
↳ se recuperan con luz ultravioleta

EEPROM: Programable y borrable electricamente
→ ocupa mucho + menos capacidad

MASKROM: se encarga a 'Taiwan' al 'por mayor' ya programada

Entradas de EEPROM comerciales:
- CS: chip select: como el enable
- OE: output enable: sin activar, salidas alta impedancia
- VPP: tensión de programación: tb es la alimentación.

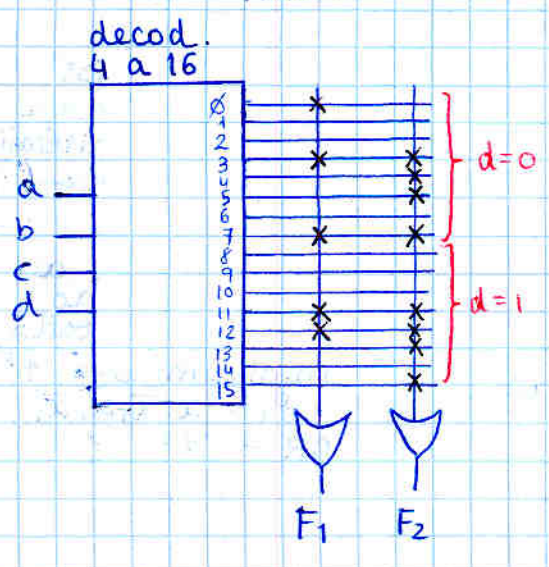
Creación de funciones lógicas

MUY simple:

ej: $F_1(dcba) = \Sigma m(0, 3, 7, 11, 12)$

$F_2(cba) = \Sigma m(3, 4, 5, 7)$ cuidado! debe funcionar para d='0' y para d='1'

F(dcba)
↑ 1 salida
↖ 4 entradas



Es simplemente hacer la conexión en el minitérmino correspondiente

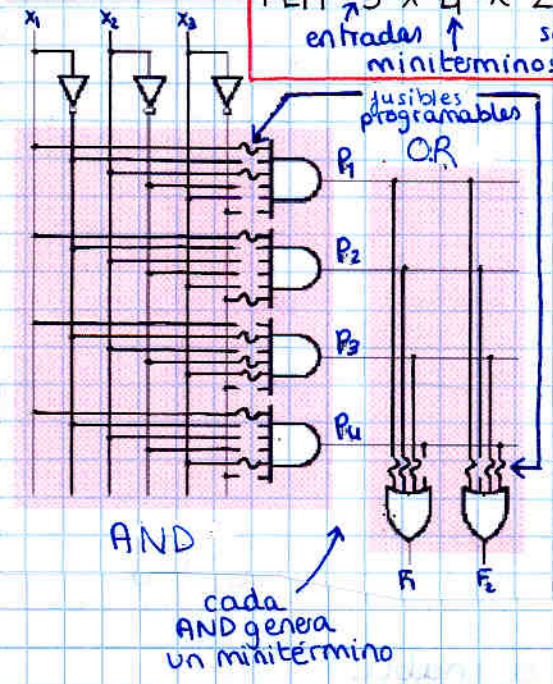
Hacer funciones lógicas con ROM's es malgastar circuito y espacio.

Solución: PLA

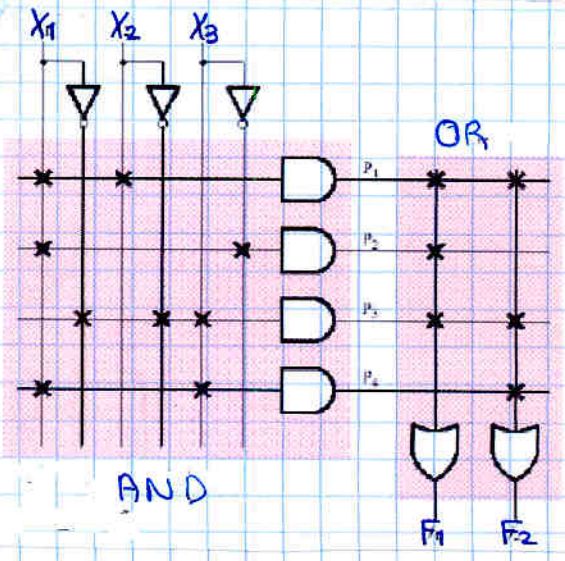
PLA: Programable Logic Array

Muy similar a la ROM, salvo que la parte AND que antes era un decodificador ahora es programable para sacar sólo los minitérminos que interesan.

PLA 3 x 4 x 2
(3 entradas, 4 minitérminos, 2 salidas)



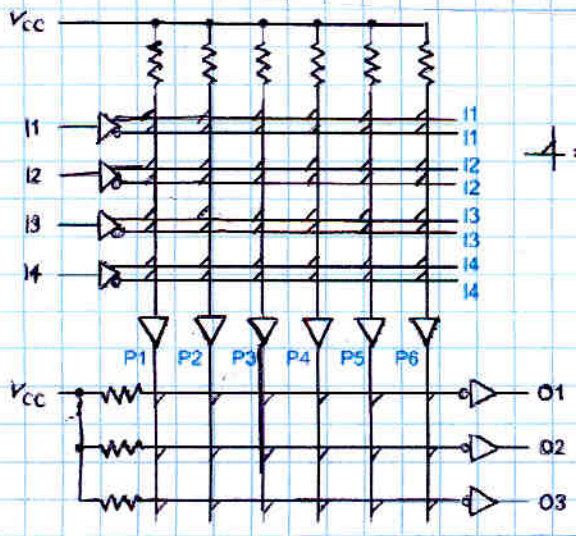
Representación compacta

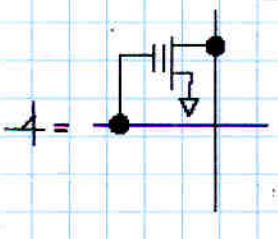


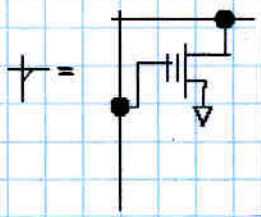
$F_1 = P_1 + P_2 + P_3$
$F_2 = P_1 + P_4$

$P_1 = x_1 \cdot x_2$
$P_2 = x_1 \cdot \overline{x_3}$
$P_3 = \overline{x_1} \cdot \overline{x_2} \cdot x_3$
$P_4 = x_1 \cdot x_3$

Estructura (AND y OR cableada) real a nivel transistores





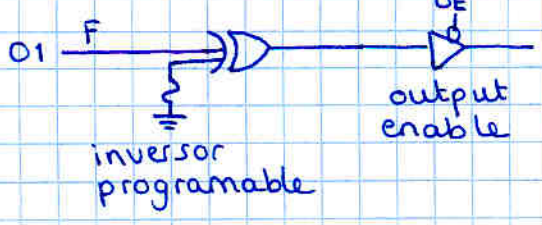


Casos 'extraños':

→ OR no conectada a ningun minitérmino = Vcc — 'O'

→ OR conectada a un minitérmino conectado a todo minitérmino → '1'
$P_i = x_1 \overline{x_1} x_i \overline{x_i} \ldots = 0$
OR → '1'

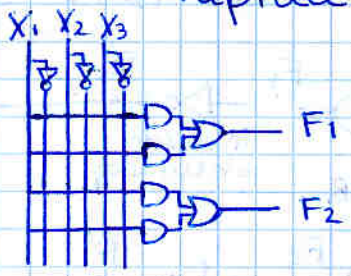
A la salida pueden tener mas cosas



PAL (programmable Array Logic)

AND – OR → fija rápida
↓ programable

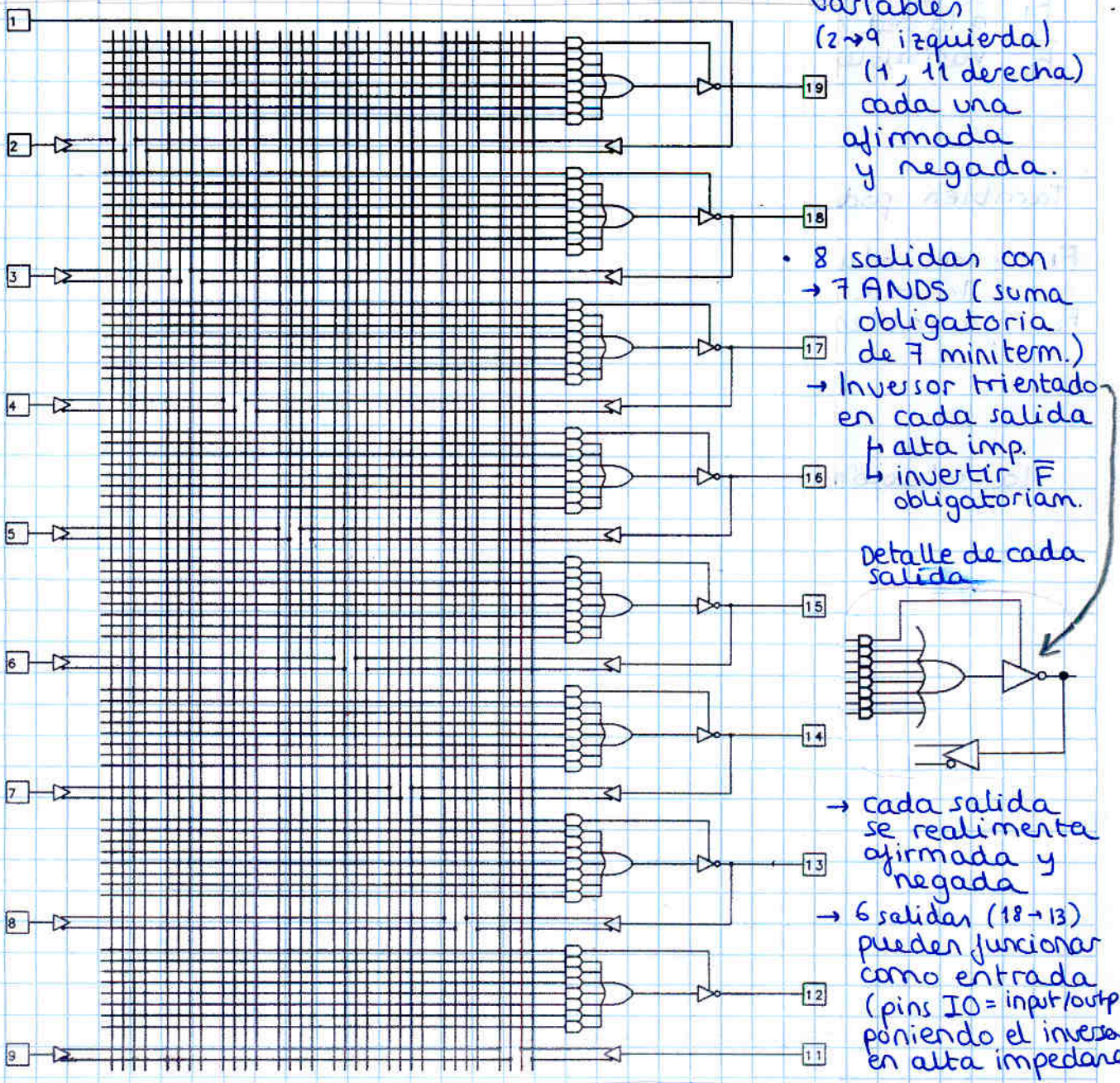
ventaja: la parte OR es mucho más rápida



desventaja: sólo funcionan de pocos minitérminos.

solución: veamos la PAL comercial

PAL16L8



- 10 entradas de variables (2→9 izquierda) (1, 11 derecha) cada una afirmada y negada.
- 8 salidas con
  - → 7 ANDS (suma obligatoria de 7 miniterm.)
  - → Inversor triestado en cada salida
    - ↳ alta imp.
    - ↳ invertir $\bar{F}$ obligatorian.

Detalle de cada salida

- → cada salida se realimenta afirmada y negada
- → 6 salidas (18→13) pueden funcionar como entrada (pins IO = input/output) poniendo el inversor en alta impedancia

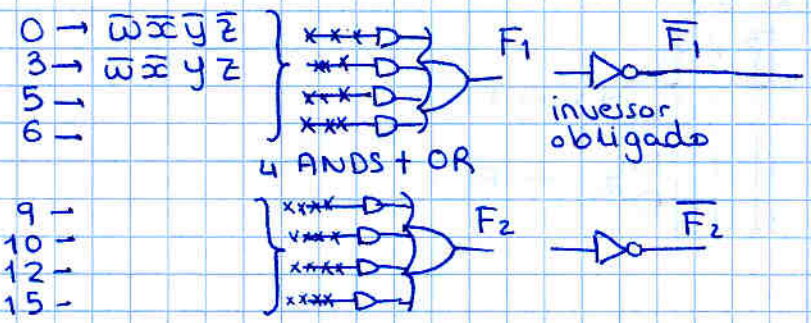
El hecho de que cada salida se realimente hace que el nº de minitérminos no esté limitado a 7.

Las PAL han desbancado a las PLA del mercado, ya obsoletas

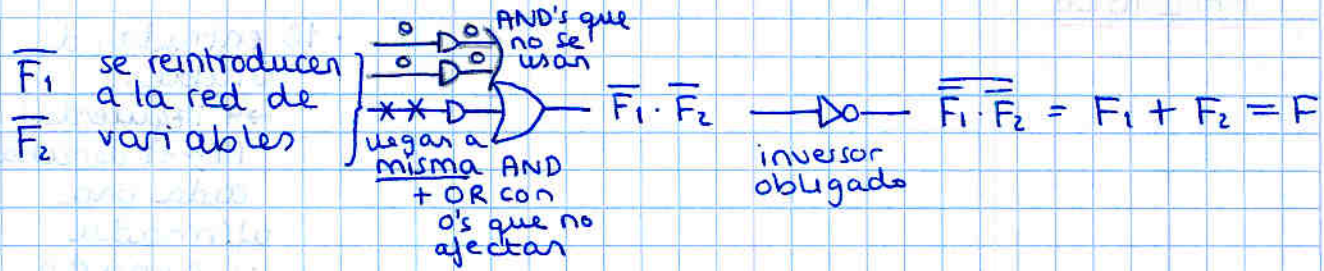
ejemplo: Implementar $F(w, x, y, z) = \Sigma m(0,3,5,6,9,10,12,15)$

↑ 1 salida ↑ 4 entradas

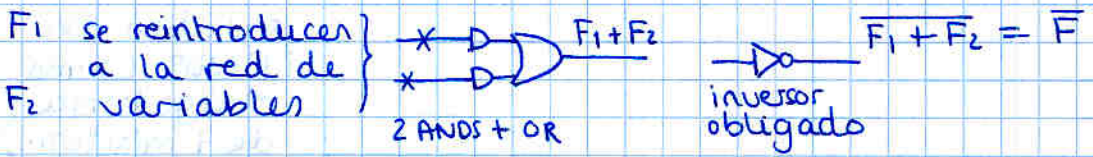
Necesitamos suma de 8 miniterminos. En cada salida sólo caben 7. Necesitaremos dos salidas y otra para sumarlas. $F = F_1 + F_2 = (\emptyset + 3 + 5 + 6) + (9 + 10 + 12 + 15)$



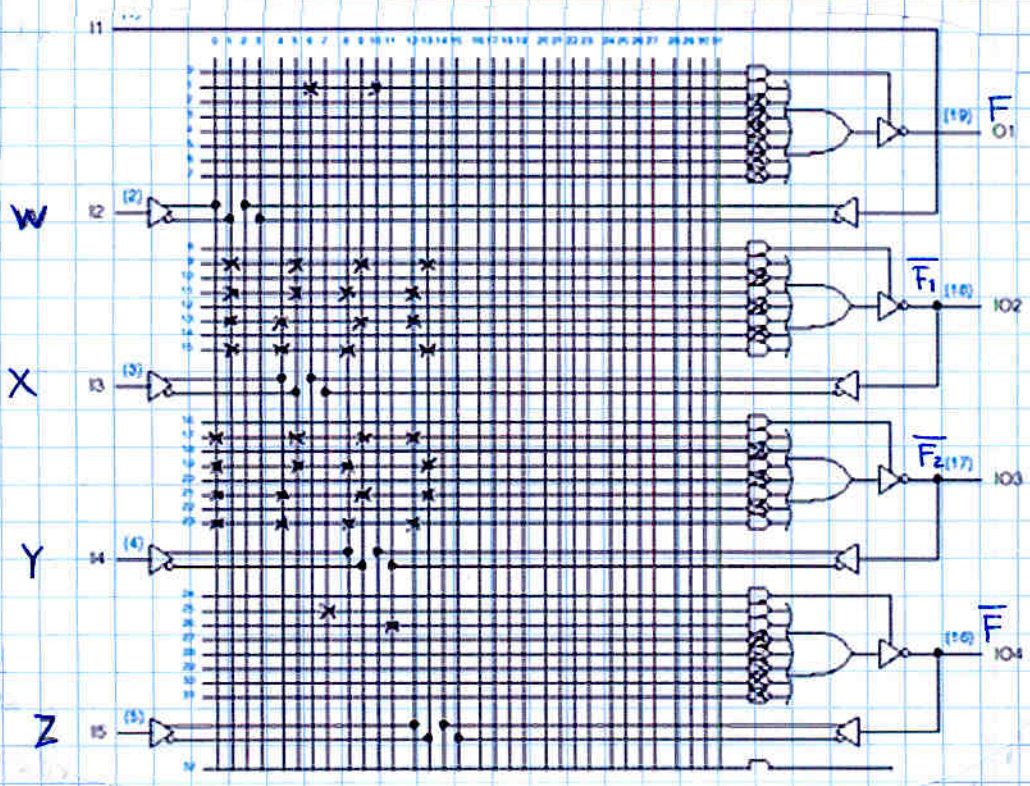
TRUCO $F = F_1 + F_2 = \overline{\bar{F}_1 \cdot \bar{F}_2}$



También podemos facilmente sacar la negada

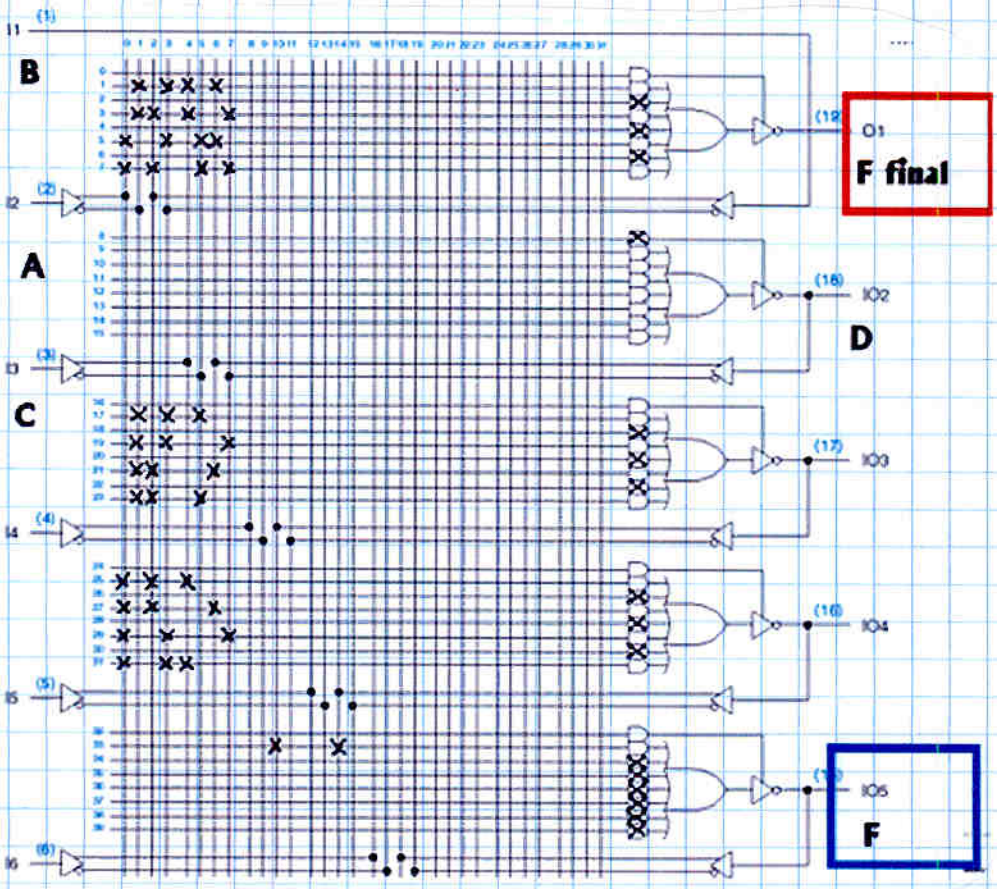


La solución final es:



ejercicio:
Dada la función implementada F (recuadro azul) volver a implementarla de la forma mas eficiente

Primero hay que conocer F



F es la negada de dos cosas multiplicadas $\overline{F_1}$ y $\overline{F_2}$

$$F = \overline{\overline{F_1} \cdot \overline{F_2}}$$

$\overline{F_1}$ viene de F_1

F_1 es la suma de cuatro minitérminos

$$F_1 = \bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}\bar{d} + \bar{a}bd + \bar{a}b\bar{c}$$

$$\bar{a}\bar{b}\bar{c}\bar{d} \quad \bar{a}\bar{b}c\bar{d} \quad \bar{a}bcd \quad \bar{a}bc\bar{d}$$
$$\bar{a}\bar{b}\bar{c}d \quad \bar{a}\bar{b}cd \quad \bar{a}b\bar{c}d \quad \bar{a}b\bar{c}\bar{d}$$

$$F_1 = \Sigma m(0, 1, 2, 5, 7, 4)$$

OJO: las variables que no 'juegan' HAY que ponerlas afirmadas y negadas

$\overline{F_2}$ viene de F_2

F_2 es la suma de cuatro minitérminos

$$F_2 = abc + abd + a\bar{b}\bar{d} + \bar{a}\bar{b}c$$

$$abcd \quad abcd \quad a\bar{b}c\bar{d} \quad \bar{a}\bar{b}cd$$
$$abc\bar{d} \quad ab\bar{c}d \quad a\bar{b}\bar{c}\bar{d} \quad \bar{a}\bar{b}c\bar{d}$$

$$F_2 = \Sigma m(14, 15, 13, 10, 8, 11)$$

$$F = \overline{\overline{F_1} \cdot \overline{F_2}} = F_1 + F_2 = \Sigma m(0, 1, 2, 4, 5, 7, 8, 10, 11, 13, 14, 15)$$

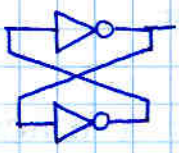
$\overline{F} = \Sigma m(3, 6, 9, 12)$ se implementa directamente en un salida que despues se niega para crear F.

TEMA 5. BIESTABLES Y TEMPORIZACIÓN

circuito { combinacional → no almacena / secuencial → sí }

- Aestable  oscilador
- Biestable  Dos estados estables
- Monoestable  Ante un pulso genera un pulso de periodo amplio y controlado

Biestable elemental



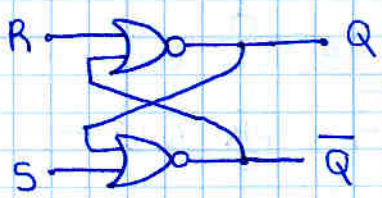
Según el estado inicial, aparecerá en Q un estado estable '1' o '0'

Estado metaestable: al cambiar de un estado estable a otro

Necesitamos poder controlarlo

→ **Biestables RS**

Biestable R-S-NOR

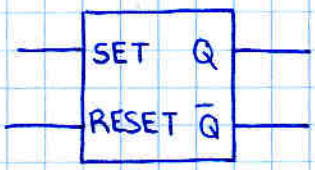


S: **Set**: pone Q = 1
R: **Reset**: pone Q = 0

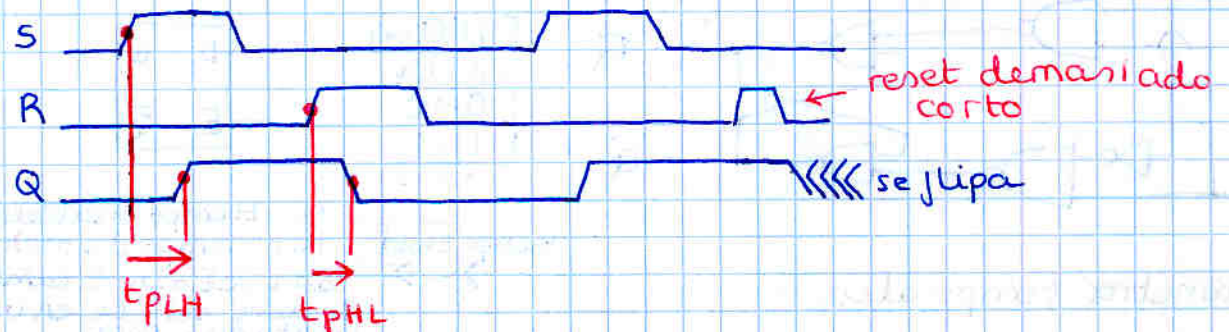
R	S	Q_{t+1}
0	0	Q_t
0	1	1
1	0	0
1	1	no deseable

S=0, R=0, Memoria: mantiene Q

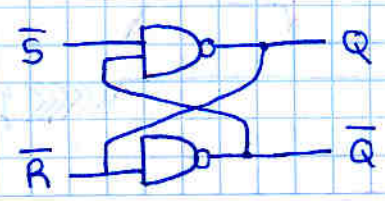
S=1, R=1, No deseable
- pone $Q = \bar{Q} = 0$ → no es lógico
- si después lo pones en memoria, no llega a estado estable y SE FLIPA



Parámetros temporales:
- Retardo de propagación (t_p)
- mínimo ancho de pulso



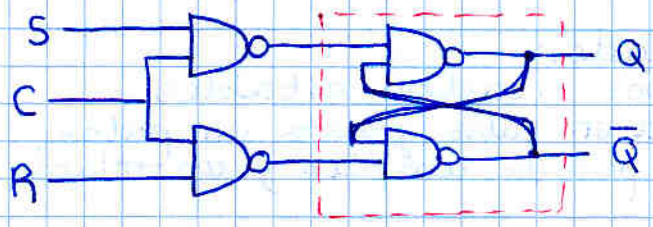
Biestable R-S-NAND



Funciona igual, pero R y S activos a nivel bajo

CRS → más Control

C predomina sobre Set y Reset, es como un cerrojo → si C = 0 → estado MEMORIA



C	R	S	Qt+1	
0	X	X	Qt	→ LOCKED
1	0	0	Qt	biestable RS
1	0	1	1	set
1	1	0	0	reset
1	1	1	no deseable	

R y S pasan a ser activos por nivel alto

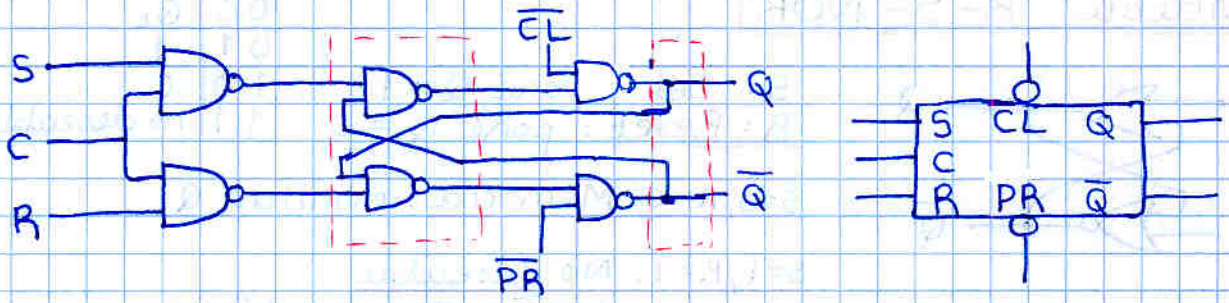
Entradas asíncronas

un control aún mayor que C
directo al usuario → no van con temporizador → 'asíncronas'

$\overline{PR}$	preset	pone Q = 1
$\overline{CL}$	clear	pone Q = 0

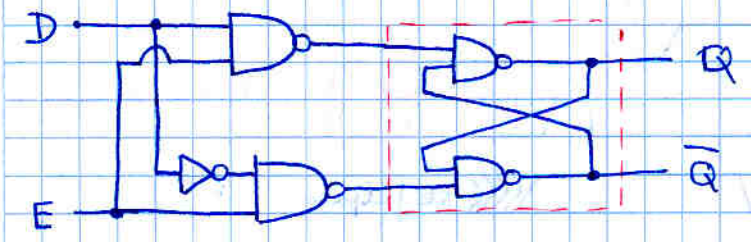
independientemente de CRS

las dos a cero (activas) → no deseable

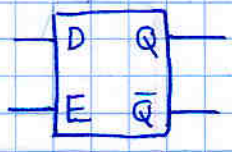


→ Biestables D el más utilizado

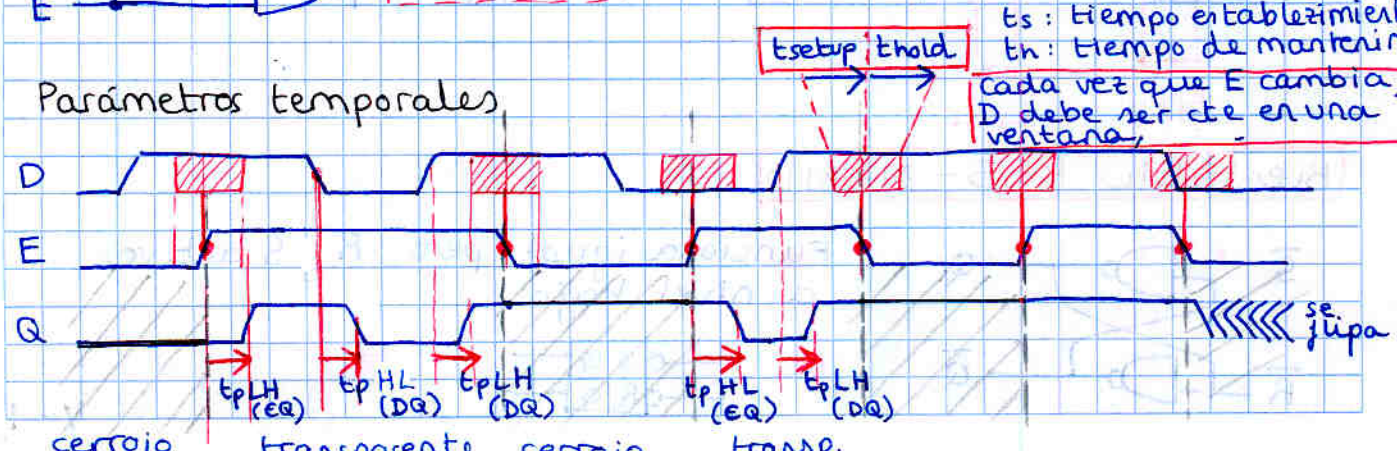
E = 0 → memoriza → modo cerrojo
E = 1 → almacena D (data) → modo transparente



E	D	Qt+1	
0	X	Qt	
1	0	0	transparente
1	1	1	transparente



Parámetros temporales



Biestable D - Edge-Triggered

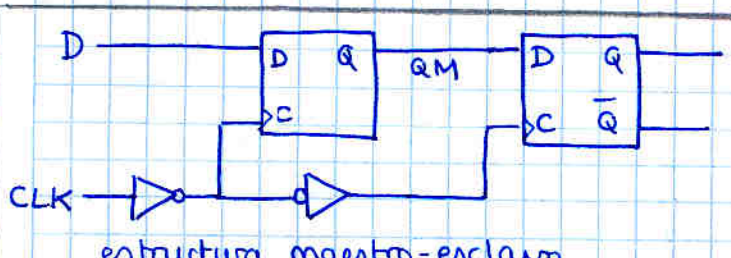
NOTA → activo por flanco de subida ↑
activo por flanco de bajada ↓

mismo concepto E=0 cerrojo
E=1 transparente
pero ahora E es un reloj CLK y se activa por FLANCO

	CLK	D	Q_{t+1}
lock	1	X	Q_t
lock	0	X	Q_t
transp.	↑	0	0
transp.	↑	1	1

CLK = ↑ = transparente → Q = D

CLK = {1, 0} = cerrojo → Q memoriza



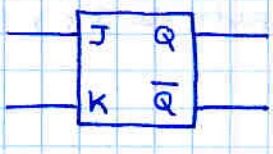
estructura maestro-esclavo

Diagrama temporal

elimina variaciones rápidas en D

→ Biestables JK

J	K	Q_{t+1}	
0	0	Q_t	memoria
0	1	0	K pone Q=0
1	0	1	J pone Q=1
1	1	$\overline{Q_t}$	inversión (toggle)

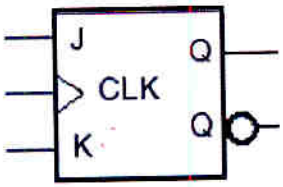


Biestables J-K Edge triggered

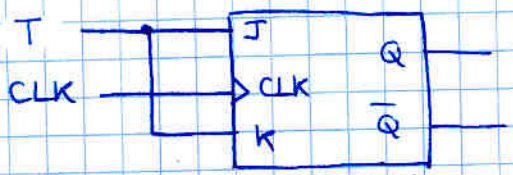
se añade CLK → si CLK 1 ó 0 → MEMORIA

si CLK ↓ → JK anterior

	J	K	CLK	Q	QN
memo	x	x	0	last Q	last QN
	x	x	1	last Q	last QN
JK anterior	0	0	↓	last Q	last QN
	0	1	↓	0	1
	1	0	↓	1	0
	1	1	↓	last QN	last Q



→ **Biestables T** (con "Toggle")



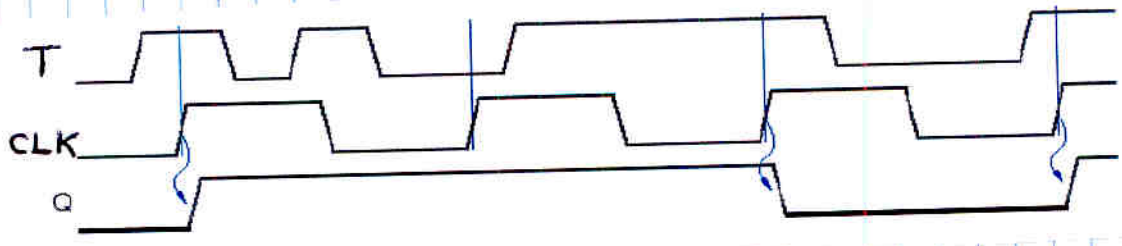
(hecho a partir de JK)

CLK = 1 ó 0 → MEMORIA

CLK = ↓ → T = 1 $Q_{t+1} = Q_t$

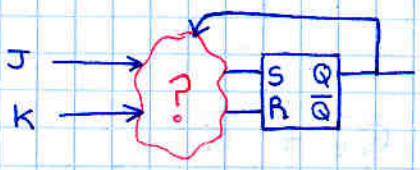
Toggle T = 0 $Q_{t+1} = \overline{Q_t}$

se usa para contadores



ejercicio:
Diseñar un JK a partir de RS-NAND

Truco para pensarlo — importante incluir la realimentación



① Hay que saber de memoria las tablas de verdad del

J	K	Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	$\overline{Q_t}$ ← conmut.

Objetivo

S	R	Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	—

RS-NAND tiene RS activo a nivel bajo.

S	R	Q_{t+1}
0	0	—
0	1	1
1	0	0
1	1	Q_t

Dato

JK
SR
D
T

② Diseñar la tabla de excitación del Dato

Q_t	Q_{t+1}	S	R
0	0	1	X
0	1	0	1
1	0	1	0
1	1	X	1

se construye pensándola a partir de la de verdad

③ Tabla de funcionamiento global

Entradas				Salidas	
J	K	Q_t	Q_{t+1}	S	R
0	0	0	0	1	X
0	0	1	1	X	1
0	1	0	0	1	X
0	1	1	0	1	0
1	0	0	1	0	1
1	0	1	1	X	1
1	1	0	1	0	1
1	1	1	0	1	0

esta columna se rellena a partir de funcionamiento del JK

con tabla de excitación del RS

④ Karnaugh:

S

Q_t \ JK	00	01	11	10
0	1	1	0	0
1	X	1	1	X

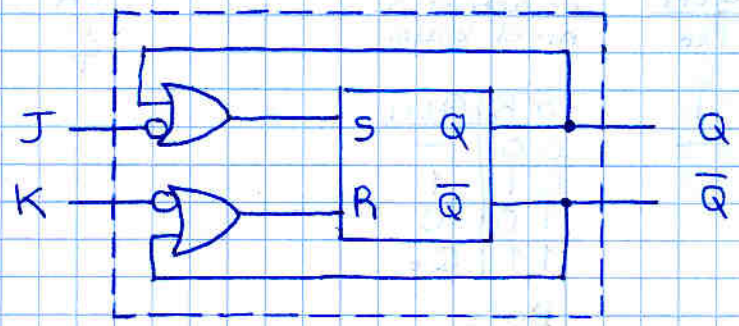
$S = \bar{J} + Q_t$

R

Q_t \ JK	00	01	11	10
0	X	X	1	1
1	1	0	0	1

$R = \overline{Q_t} + \bar{K}$

⑤ Finalizar el problema:



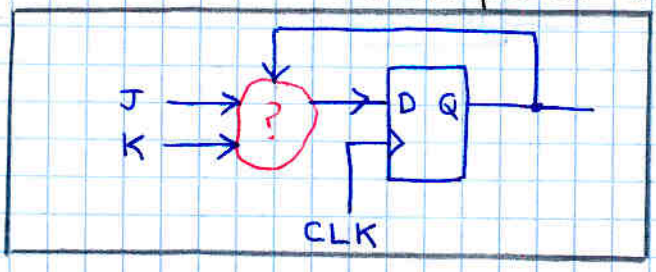
Tambien seria (obviamente)

$$S = \overline{\overline{\bar{J} + Q_t}} = \overline{J \cdot \overline{Q_t}}$$

$$J = \overline{\overline{Q_t + \bar{K}}} = \overline{Q_t \cdot K}$$

ejercicio.

Diseñar FF JK a partir de D



① Objetivo

	J	K	Q_{t+1}
↑	0	0	Q_t
↑	0	1	0
↑	1	0	1
↑	1	1	$\overline{Q_t}$

Dato

	D	Q_{t+1}
↑	0	0
↑	1	1

② Tabla de excitación de D

Q_t	Q_{t+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

$D = Q_{t+1}$

③ Tabla de funcionamiento global

Entrad.				Sal.
J	K	Q_t	Q_{t+1}	D
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	0	0

tabla del JK — tabla de excitación de D

④ D

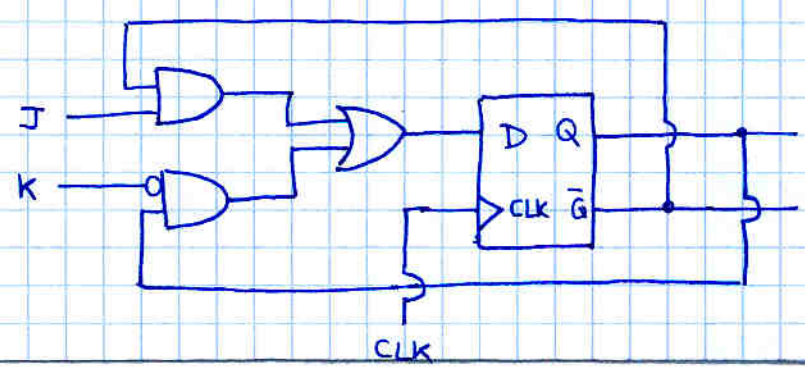
Q_t \ JK	00	01	11	10
0	0	0	1	1
1	1	0	0	1

$D = J\overline{Q}_t + \overline{K}Q_t$

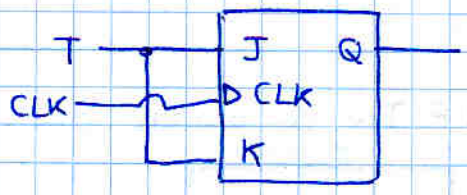
Q_t \ JK	

sólo hace falta una vez!

⑤

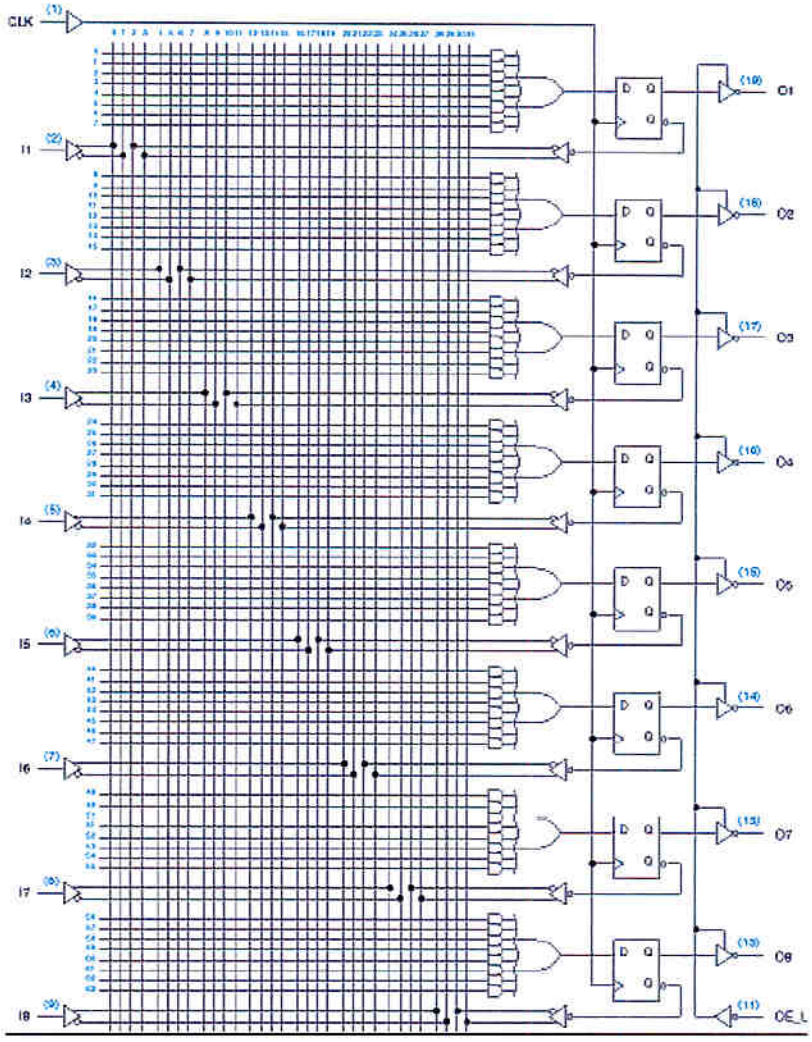


ejercicio
Toggle con JK

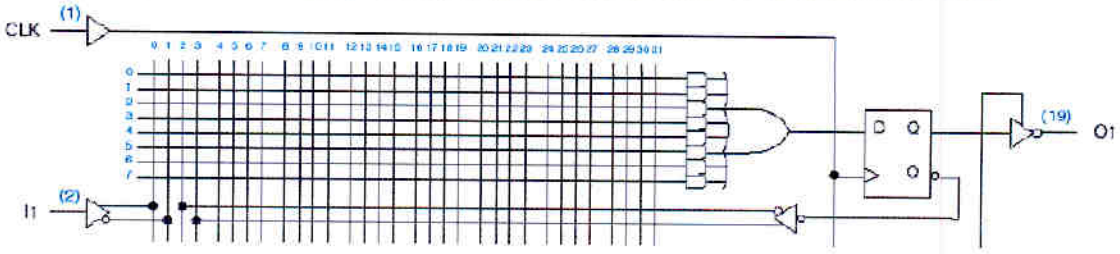


PAL
Secuenciales

- 16R8



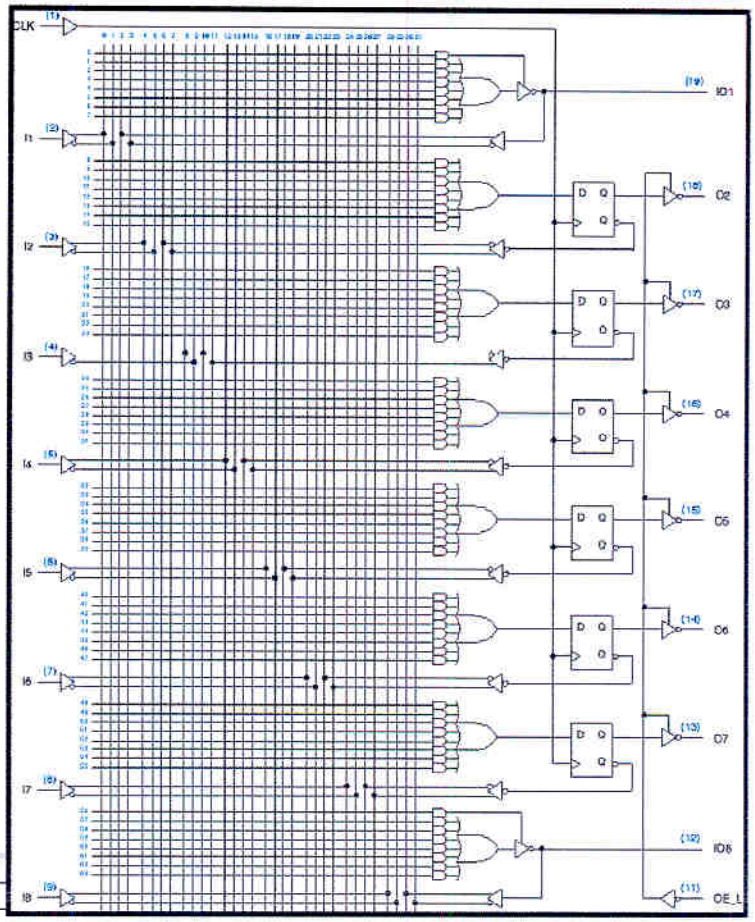
Célula de salida de la PAL16R8



- 8 productos término a la entrada D del flip-flop.
  - Flanco de subida, reloj común para todos
- Salida Q realimentada en la matriz AND.
  - Se necesita para diseñar subsistemas secuenciales como contadores y registros (tema 6) y máquinas de estados finitos (tema 7).
  - Salida Alta Impedancia con control común.

PAL16R6

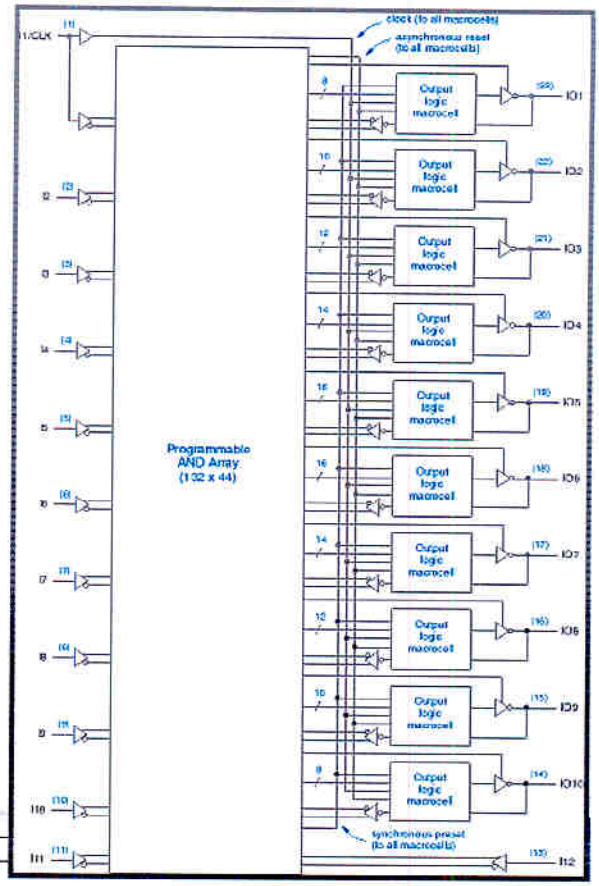
- 16 pines de entrada y/o entrada/salida.
- Seis salidas registradas.
- Dos salidas combinacionales.
- PAL16L8 : 8 salidas combinacionales



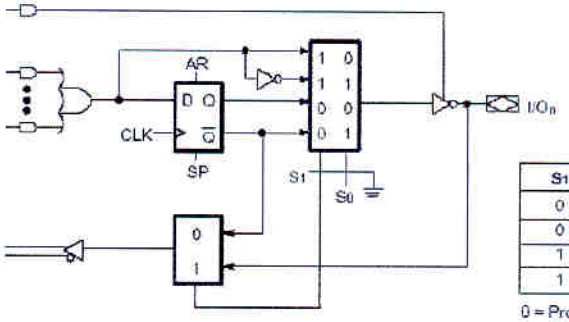
PAL22V10

- Mas entradas.
- Mas productos término.
- Reset Asíncrono.
- Preset Síncrono.
- La más utilizada.

- Fusible de seguridad.
- Firma electrónica.

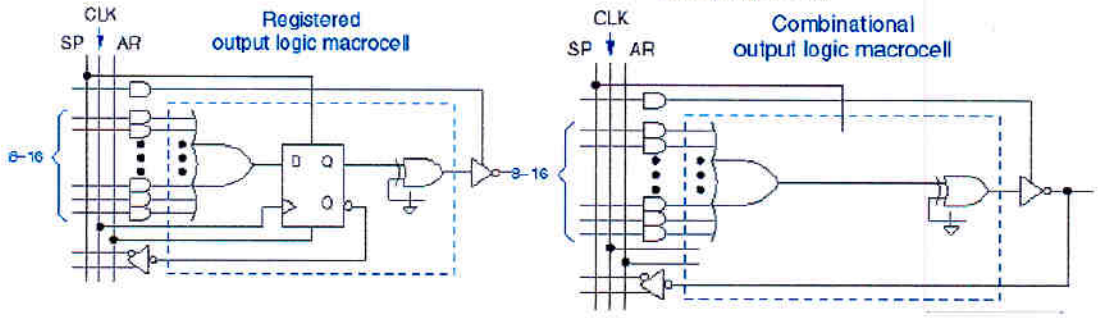


PAL22V10 output logic macrocell (OLMC)



S1	S0	Output Configuration
0	0	Registered/Active Low
0	1	Registered/Active High
1	0	Combinatorial/Active Low
1	1	Combinatorial/Active High

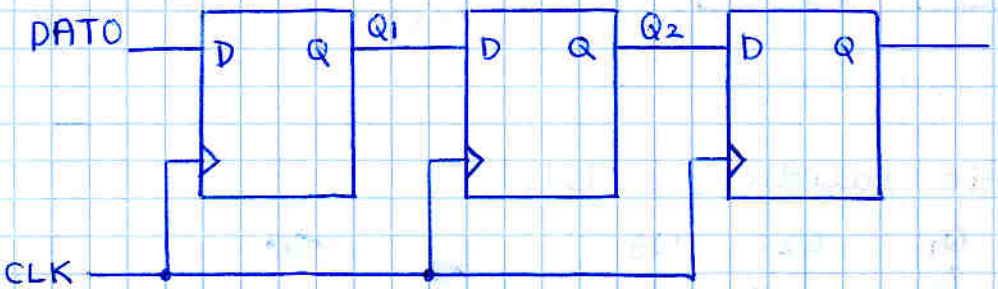
0 = Programmed EE bit
1 = Erased (charged) EE bit



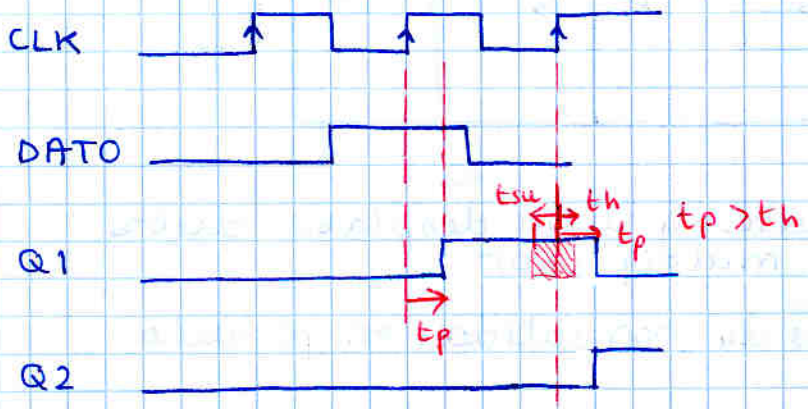
TEMA 6. SUBSISTEMAS SECUENCIALES

Registro de desplazamiento

n flip-flop D's se van "pasando" el dato en cada ciclo de reloj.



Funciona porque los flip-flop tienen un tiempo de propagación (se cumplen los tiempos de setup y de hold) y los datos no atraviesan más de un biestable



La frecuencia del reloj tiene un límite

$T > t_p$

y aun así el FF2 estaría muestreando la salida del FF1 conforme cambia

$T > t_p + t_{su}$

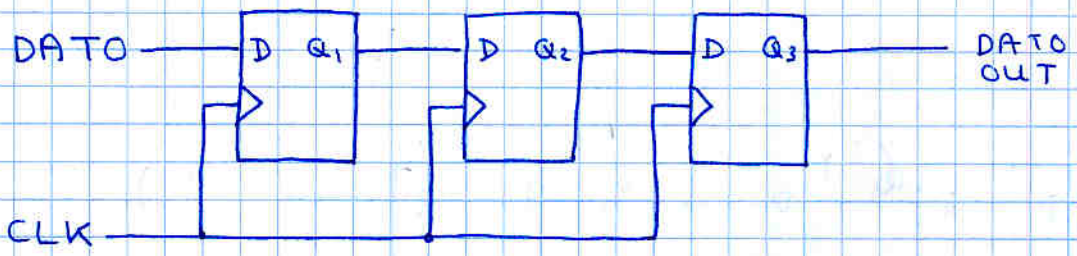
el t_h no juega porque está 'dentro' del de t_p

$$T > t_p + t_{su}$$

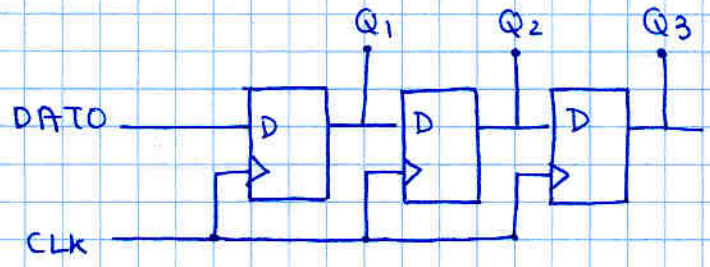
¿Que permite?

- Entrada serie
  salida serie

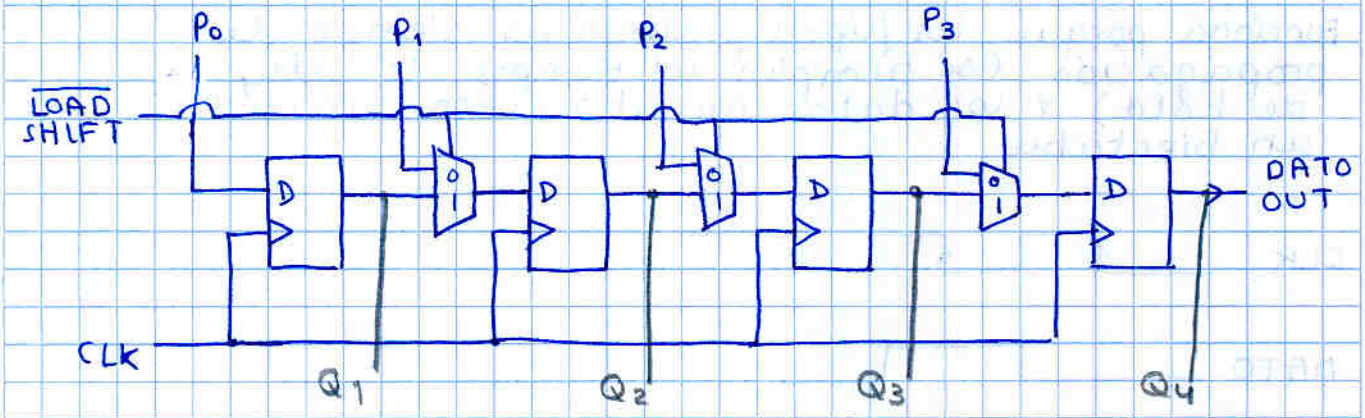
no sirve para mucho



- Entrada serie salida paralelo:



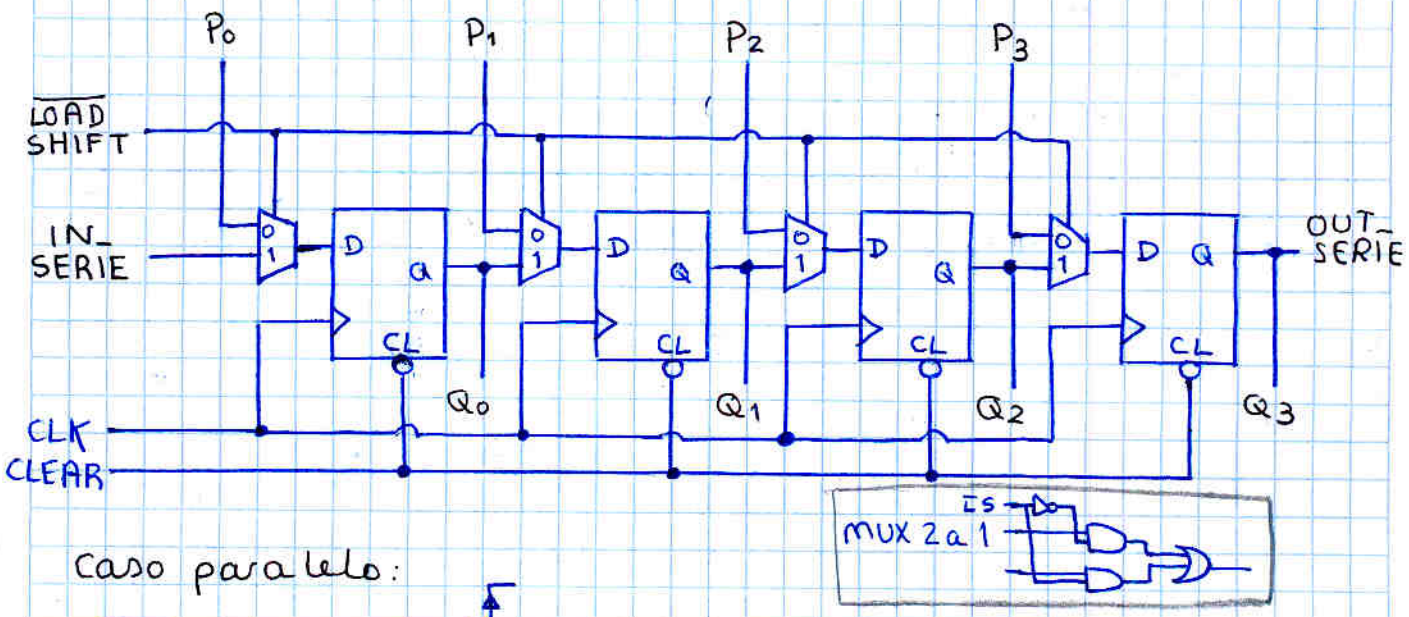
- Entrada paralelo salida serie



LOAD SHIFT permite cargar datos y luego desplazar datos mediante un multiplexor

las salidas Q_1 Q_2 Q_3 Q_4 son salidas en paralelo

Todas las configuraciones en mismo circuito



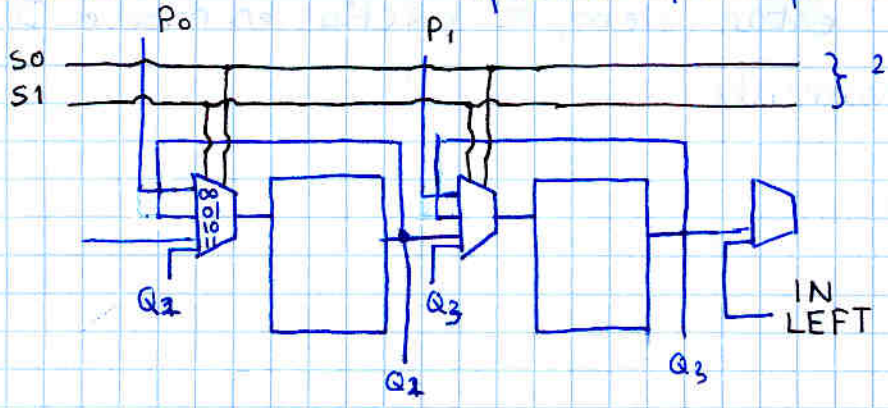
Caso paralelo:

$\overline{LOAD}/SHIFT = 0$ ⟶ $Q_0 = P_0$, $Q_1 = P_1$, $Q_2 = P_2$, $Q_3 = P_3$

Sólo mantiene durante 1 ciclo

$\overline{LOAD}/SHIFT = 1$ ⟶ se mueven hacia la derecha

Para memorizar eternamente necesitaríamos 3 entradas en el multiplexor para que haya realimentación



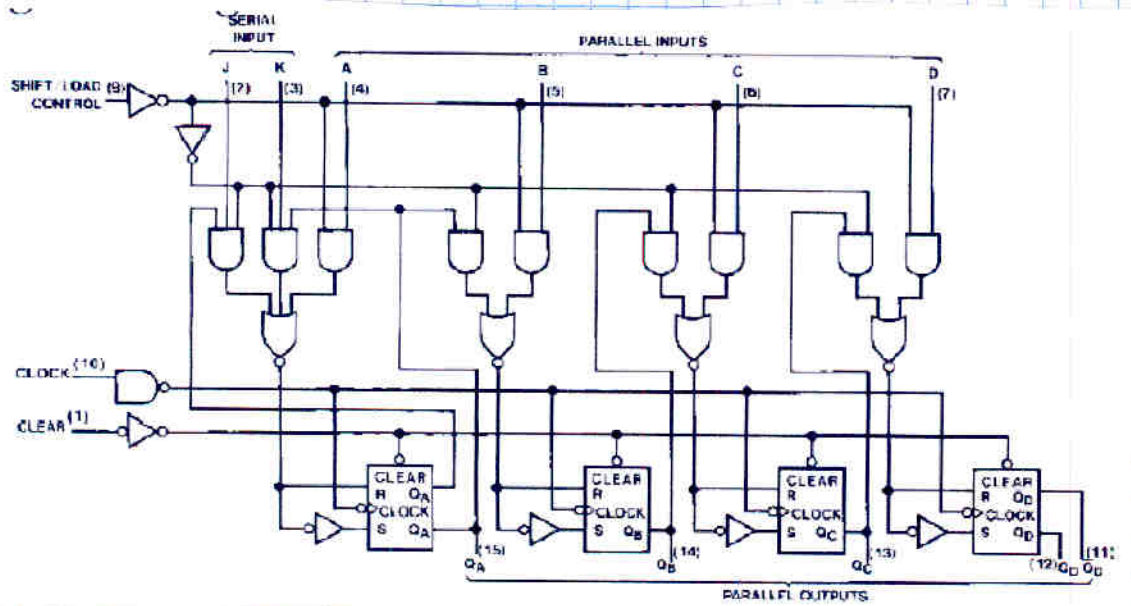
Podemos aprovechar la cuarta salida del multiplexor para permitir desplazamiento hacia la izquierda

Es un registro de desplazamiento universal

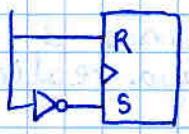
S1	S0	MODO
0	0	Carga Paralelo (parallel load)
0	1	Memoriza / Mantenimiento (hold)
1	0	Shift_Right
1	1	Shift_Left

varía según fabricante

ej. Registro de desplazamiento 74 x 195



el R-S



es como un D

Los registros de desplazamiento estan siempre hechos en modo D.

Tabla de funcionamiento

Inputs									Outputs				
Clear	Shift/ Load	Clock	Serial		Parallel				Q_A	Q_B	Q_C	Q_D	$\overline{Q}_D$
			J	$\overline{K}$	A	B	C	D					
L	X	X	X	X	X	X	X	X	L	L	L	L	H
H	L	↑	X	X	a	b	c	d	a	b	c	d	$\overline{d}$
H	H	L	X	X	X	X	X	X	Q_{A0}	Q_{B0}	Q_{C0}	Q_{D0}	$\overline{Q}_{D0}$
H	H	↑	L	H	X	X	X	X	Q_{A0}	Q_{A0}	Q_{Bn}	Q_{Cn}	$\overline{Q}_{Cn}$
H	H	↑	L	L	X	X	X	X	L	Q_{An}	Q_{Bn}	Q_{Cn}	$\overline{Q}_{Cn}$
H	H	↑	H	H	X	X	X	X	H	Q_{An}	Q_{Bn}	Q_{Cn}	$\overline{Q}_{Cn}$
H	H	↑	H	L	X	X	X	X	$\overline{Q}_{An}$	Q_{An}	Q_{Bn}	Q_{Cn}	$\overline{Q}_{Cn}$

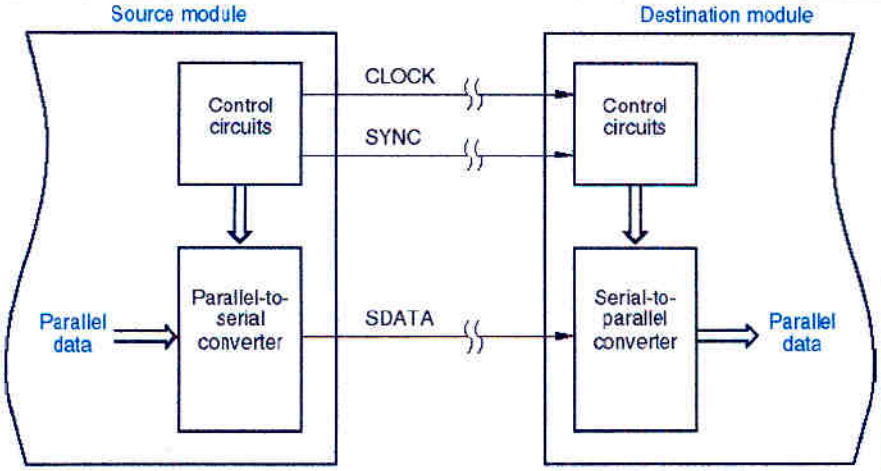
las entradas JK permiten manejar el primer flip-flop como si fuera un JK

Es muy dado a juegos de luces

Se puede aprovechar para hacer shif right conectando las salidas en paralelo a las entradas correspondientes.

Aplicaciones: Registro de desplazamiento:

Conversión serie ↔ paralelo
ej. puerto de comunicaciones



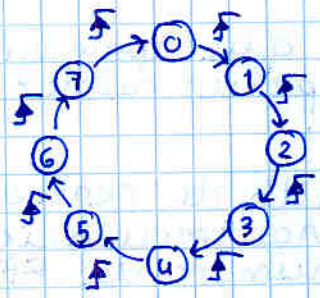
CONTADORES

Se utilizan para 'contar'

Pueden contar de cualquier forma que se nos ocurra. El módulo es el número de cuentas/estados que tiene.

Diagramas de estado

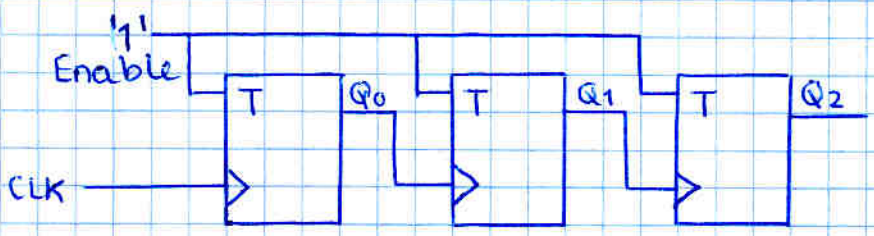
Módulo 8



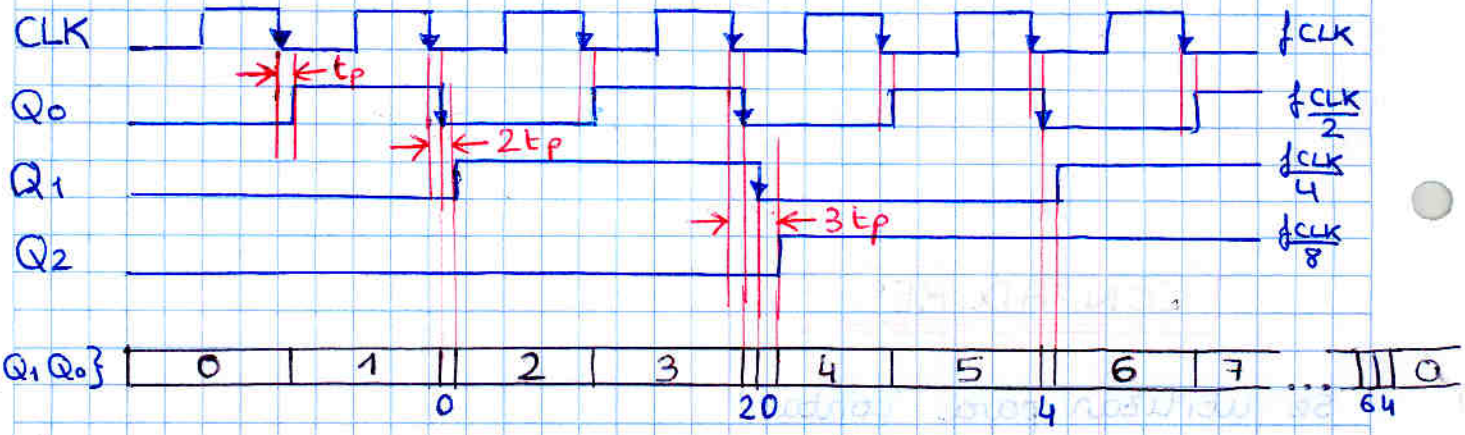
Contadores asíncronos

Los más sencillos, pero soportan menor frecuencia

Se hacen **siempre con flip-flop's T (toggle)**



Cada FF T tiene como señal de reloj la salida del anterior



¡Está contando! ocasionalmente hay cuentas 'intrusas' de duración tp.
Si la frecuencia es muy elevada, la cuenta buena (cuya duración se va acortando) puede llegar a confundirse con las cuentas espúreas (de duración tp fija)
Detalle: los números que aparecen en las cuentas espúreas son siempre un número que ya ha aparecido en la cuenta 'real'.

Ventajas: muy sencillo de realizar
Desventajas: tiene una frecuencia máxima que depende del número de FF

$$f_{max} = \frac{1}{T_{min}} = \frac{1}{n t_p}$$

Este diseño sirve para cuentas de módulo 2^n

Contadores módulo $< 2^n$

Hay que añadir una puerta lógica que haga CLEAR los FF cuando la cuenta llegue a cierto número.

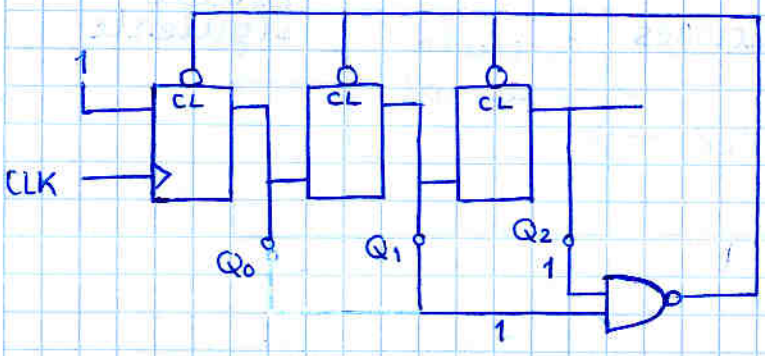
En qué estado se tiene que hacer el clear / reset?

Supongamos que queremos $0 \to 1 \to 2 \to 3 \to 4 \to 5$

CLEAR (asíncrono): hacer CL cuando llegué el 6 → se pondrá a cero
RESET (síncrono): hacer R cuando llegué a 5 → se esperará al siguiente flanco de reloj para ponerlo a cero

ejemplo:
Contador hasta 5 (módulo 6) teniendo CL

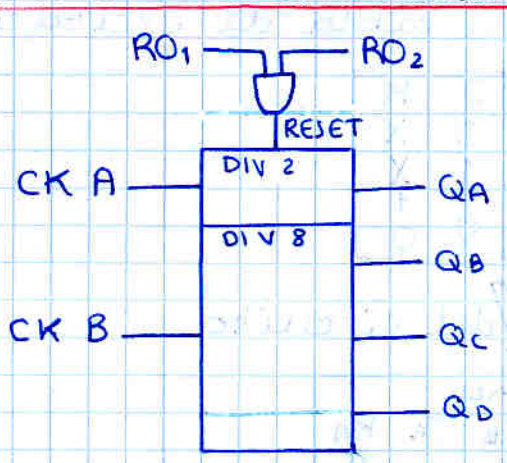
Cuando llegué al 6 (1 1 0), pasado un t_p se hará el clear, y el 6 quedará únicamente como si fuera una cuenta espúrea.



(si hubiera tenido reset (síncrono) hubiéramos hecho que la puerta se activara con 5, ya que lo mantendría todo el rato

Detalle: ¿y si la puerta se activara con un espúreo? Los espúreos aparecen después del número en cuestión

Contadores asíncronos comerciales



2 grupos de flip-flops.

DIV 2 → $f_A = \frac{f_{CK\,A}}{2}$

DIV 8 → $f_B = \frac{f_{CK\,B}}{2}$, $f_C = \frac{f_{CK\,B}}{4}$, $f_D = \frac{f_{CK\,B}}{8}$ } actúa como contador módulo 8

conectando Q_A a CKB se obtiene contador módulo 16

RO_1 y RO_2 son resets que ya tienen puerta AND para hacer cuentas con módulo $< 2^n$

Contadores Síncronos

Mismo reloj para todos los FF
(los FF pueden ser de cualquier tipo, los más fáciles son los D)
→ no tiene estados espúreos

Estructura

ejemplo:
Diseñar contador síncrono módulo 5 con FF JK

1- ¿Cuantos JK necesitamos? $2^2=4$, $2^3=8$ → 3 FF

2- Recordemos la tabla de verdad del JK y Hagamos su tabla de excitaciones

J	K	Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	$\overline{Q_t}$

⇒

Q_t	Q_{t+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

3- Hagamos la tabla de verdad del circuito

	Estado Actual			Estado Siguiente			Excitaciones					
	Q_C	Q_B	Q_A	Q_C	Q_B	Q_A	J_C	K_C	J_B	K_B	J_A	K_A
Ø	0	0	0	0	0	1	0	X	0	X	1	X
1	0	0	1	0	1	0	0	X	1	X	X	1
2	0	1	0	0	1	1	0	X	X	0	1	X
3	0	1	1	1	0	0	1	X	X	1	X	1
4	1	0	0	0	0	0	X	1	0	X	0	X
5	1	0	1	X	X	X	X	X	X	X	X	X
6	1	1	0	X	X	X	X	X	X	X	X	X
7	1	1	1	X	X	X	X	X	X	X	X	X
5	1	0	1	0	0	0	X	1	0	X	X	1
6	1	1	0	0	0	0	X	1	X	1	0	X
7	1	1	1	0	0	0	X	1	X	1	X	1

Estados que no pertenecen a la cuenta → mínimo coste / mínimo riesgo

Podemos poner cualquier estado siguiente para cualquier estado actual, de modo que la cuenta puede ser como nos salga de las narices.
Incluso más entrada y según la entrada cuente de una u otra forma.

Implementación mínimo coste:
Suponemos que nunca se entrará en los estados 5, 6 y 7 (teoricamente no se entrará). Permitirá un circuito combinacional más simple

Implementación mínimo riesgo:
Si por ruido se entrara en 5, 6, 7 se volvería al cero.

4. Elegimos mínimo coste. Hay que hacer 6 Karnaughs (dos de ellos son obvios: $K_C = 1$ $K_A = 1$

cuidado con las variables ideal $Q_C Q_B Q_A$

$Q_B Q_A$ \ Q_C	J_C 0	J_C 1	J_B 0	J_B 1	K_B 0	K_B 1	J_A 0	J_A 1
0 0	0	X	0	0	X	X	1	0
0 1	0	X	1	X	X	X	X	X
1 1	1	X	X	X	1	X	X	X
1 0	0	X	X	X	0	X	1	X

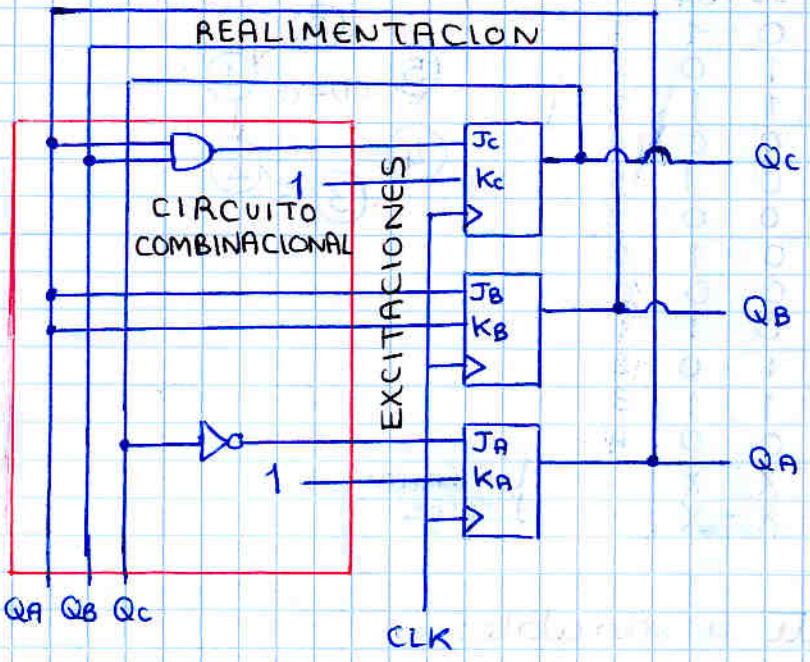
hacemos 4 karnaughs (aprovechando la misma tabla para ahorrar espacio)

$J_C = Q_A Q_B$ $J_B = Q_A$ $K_B = Q_A$ $J_A = \overline{Q_C}$

5. Lo implementamos

$J_C = Q_A Q_B$	$J_B = Q_A$	$J_A = \overline{Q_C}$
$K_C = 1$	$K_B = Q_A$	$K_A = 1$

esto puede ser útil para las siguientes preguntas



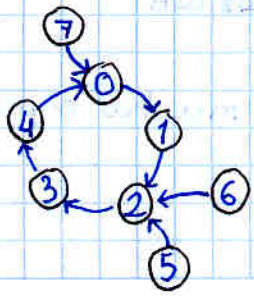
6. Habiendo implementado por mínimo coste, podrían preguntar qué pasaría si cayese en 5, 6, 7. sabiendo

$J_C = Q_A Q_B$ $J_B = Q_A$ $J_A = \overline{Q_C}$
$K_C = 1$ $K_B = Q_A$ $K_A = 1$

en 5. $Q_C Q_B Q_A = 1\,0\,1$ $J_C = 0$ $J_B = 1$ $J_A = 0$ $K_C = 1$ $K_B = 1$ $K_A = 1$ → función JK → $Q_C = 0$, $Q_B = (\text{toggle}) = 1$, $Q_A = 0$ } 2

en 6. $Q_C Q_B Q_A = 1\,1\,0$ $J_C = 0$ $J_B = 0$ $J_A = 0$ $K_C = 1$ $K_B = 0$ $K_A = 1$ → $Q_C = 0$, $Q_B = (\text{mem}) = 1$, $Q_C = 0$ } 2

en 7 $Q_C Q_B Q_A = 1\,1\,1$ $J_C = 1$ $J_B = 1$ $J_A = 0$ $K_C = 1$ $K_B = 1$ $K_A = 1$ → $Q_C = (\text{toggle}) = 0$, $Q_B = (\text{toggle}) = 0$, $Q_C = 0$ } ∅



ejemplo 2. Diseñar contador síncrono módulo 6 con entrada UP-DOWN

1. ¿FF? Como no especifican elegimos los D
   Necesitaremos 3 FF D

2. Tabla de verdad y excitaciones del D

D	Q_{t+1}
0	0
1	1

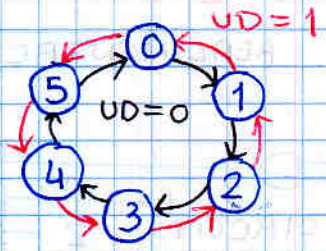
Q_t	Q_{t+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

son triviales
$D = Q_{t+1}$

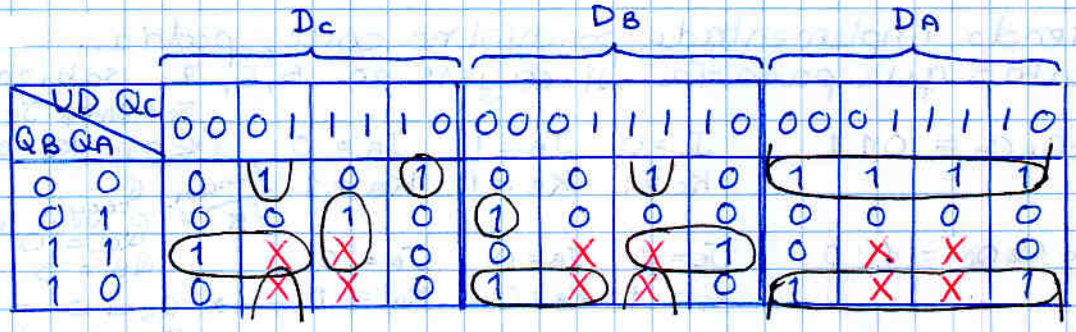
3. Tabla de verdad del circuito. Columna de excitaciones coincide con estado siguiente. Nos ahorramos una columna (por elegir FF D)

	estado actual				excitaciones/estado sig.			
	UD	Q_C	Q_B	Q_A	D_C	D_B	D_A	
Ø	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	2
2	0	0	1	0	0	1	1	3
3	0	0	1	1	1	0	0	4
4	0	1	0	0	1	0	1	5
5	0	1	0	1	0	0	0	Ø
Ø	1	0	0	0	1	0	1	5
1	1	0	0	1	0	0	0	Ø
2	1	0	1	0	0	0	1	1
3	1	0	1	1	0	1	0	2
4	1	1	0	0	0	1	1	3
5	1	1	0	1	1	0	0	4
no deseados 6	X	1	1	0	X	X	X	
no deseados 7	X	1	1	1	X	X	X	

} mínimo coste



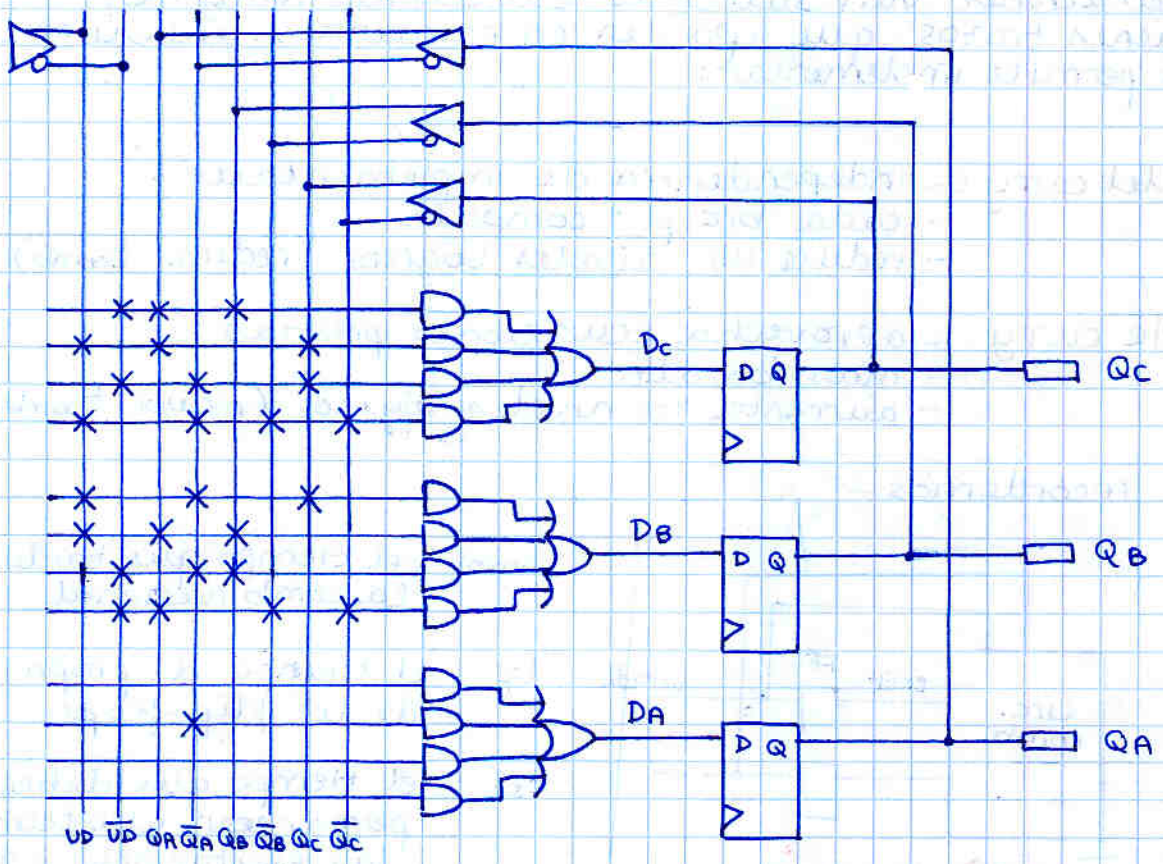
4. 3 Karnaughs de 4 variables



$D_C = \overline{UD}\,Q_B\,Q_A + UD\,Q_C\,Q_A + \overline{UD}\,Q_C\,\overline{Q_A} + UD\,\overline{Q_C}\,\overline{Q_B}\,\overline{Q_A}$
$D_B = UD\,Q_C\,\overline{Q_A} + UD\,Q_B\,Q_A + \overline{UD}\,Q_B\,\overline{Q_A} + \overline{UD}\,\overline{Q_C}\,\overline{Q_B}\,Q_A$
$D_A = \overline{Q_A}$

Implementar esto con puertas lógicas ocuparía mucho espacio (hay muchas sumas de miniterminos

Esa estructura recuerda mucho a una PAL con FF.
(PAL registrada)



ej: Contador módulo 16

nº Flipflops: 4 FF tipo D

Estado actual (Y_t)				Estado siguiente (Y_{t+1})				Excitaciones(I)			
QD	QC	QB	QA	QD	QC	QB	QA	D_D	D_C	D_B	D_A
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	1	0	0	0	1	0
0	0	1	0	0	0	1	1	0	0	1	1
0	0	1	1	0	1	0	0	0	1	0	0
0	1	0	0	0	1	0	1	0	1	0	1
0	1	0	1	0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	1	0	1	1	1
0	1	1	1	1	0	0	0	1	0	0	0
1	0	0	0	1	0	0	1	1	0	0	1
1	0	0	1	1	0	1	0	1	0	1	0
1	0	1	0	1	0	1	1	1	0	1	1
1	0	1	1	1	1	0	0	1	1	0	0
1	1	0	0	1	1	0	1	1	1	0	1
1	1	0	1	1	1	1	0	1	1	1	0
1	1	1	0	1	1	1	1	1	1	1	1
1	1	1	1	0	0	0	0	0	0	0	0

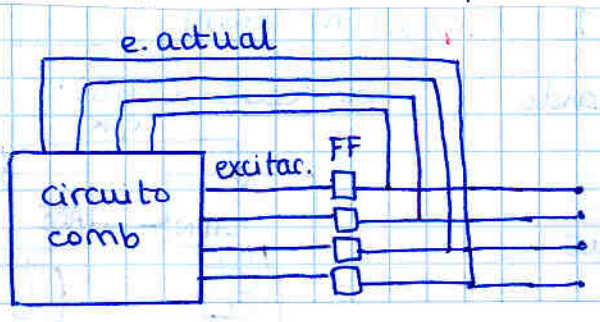
haciendo Karnaugh sale:

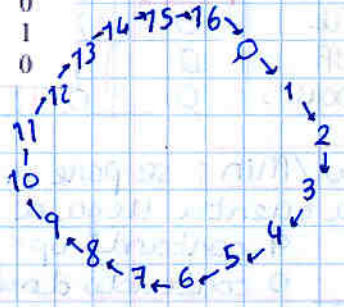
$D_A = \overline{Q_A}$

$D_B = Q_A \oplus Q_B$

$D_C = Q_C \oplus (Q_A \cdot Q_B)$

$D_D = Q_D \oplus (Q_C \cdot Q_B\, Q_A)$



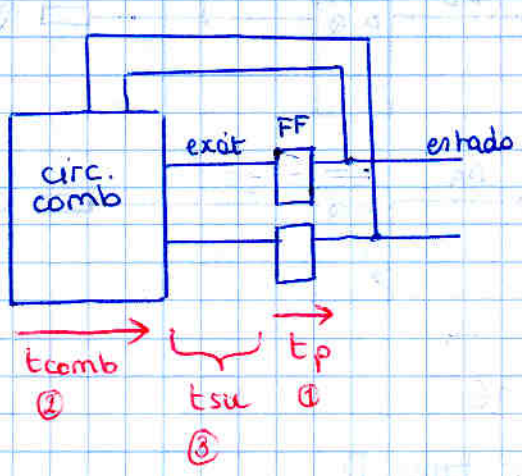


a la hora de implementar el circuito combinacional, nos podemos dar cuenta de ciertas cosas

Vemos que, casi siempre, para los bits de mayor peso de las excitaciones van saliendo ecuaciones mayores que tienen trozos que aparecen en ecuaciones anteriores. Esto permite implementar:

- Parallel carry: - independencia de implementación
  - cada bit por completo
  - reduce los niveles lógicos (reduce t_{comb})
- Ripple carry: - aprovecha ecuaciones previas
  - más barato
  - aumenta los niveles lógicos (mayor t_{comb})

recordemos



t_{comb}: el tiempo que tarda la combinacional

t_p: el tiempo de propagación de los flip-flops

t_{su}: el tiempo que deben permanecer quietecitas las excitaciones antes de que los FF las puedan agarrar con seguridad

$$f_{max} = \frac{1}{T_{min}} = \frac{1}{t_p + t_{comb} + t_{su}}$$

C.I. comerciales. Contadores síncronos

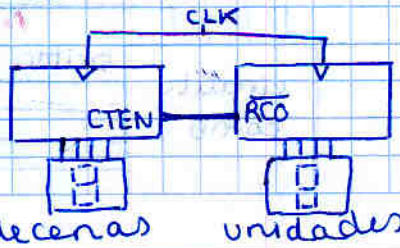
74190 – contador módulo 10 up/down con parallel load y que permite conexión en cascada

	Entradas					Salidas		
	$\overline{CTEN}$	D/$\overline{U}$	CLK	$\overline{LOAD}$	ABCD	$Q_A Q_B Q_C Q_D$	Max/min	$\overline{RCO}$
Quieto	1	X	X	X	X X X X	$Q_A Q_B Q_C Q_D$		
Load	0	X	X	0	X X X X	A B C D		
cnt. UP	0	1	↑	1	X X X X	Cnt up		
cnt. DOWN	0	0	↑	1	X X X X	Cnt down		

Max/Min: se pone a '1' cuando la cuenta llega a:
- 9 contando up
- 0 contando down

$\overline{RCO}$: Ripple Carry Output
se pone a cero
- subiendo en 9
- bajando en 0

Es 'cascadable'



Contadores 160 (MUY típicos en exámenes)

existen 4 modelos

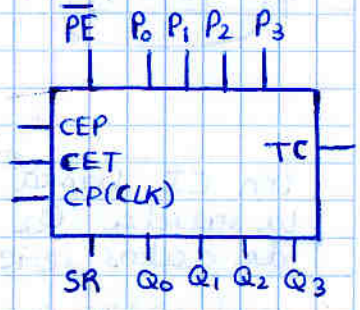
	BCD (módulo 10)	Binario (Módulo 16)
Reset Asíncrono	LS160A	LS161A
Reset Síncrono	LS162A	LS163A

Los 4 son contadores síncronos de 4 bits

Recuerda: el reset síncrono o asíncrono afecta a cuando queremos hacer un contador de menor módulo con un CI de un módulo mayor.

- Si el reset es síncrono: una puerta lógica debe activar el reset al llegar a la última cuenta
- Si el reset es asíncrono: una puerta lógica debe activar el reset al llegar a la cuenta que ya no deseamos, la cual instantaneamente será reseteada y habrá sido un estado espúreo.

	$\overline{SR}$ Reset	$\overline{PE}$ Parallel Enable	2 Enable CET	2 Enable CEP
RESET	0			
LOAD	1	0		
COUNT	1	1	1	1
HOLD	1	1	0	X
HOLD	1	1	X	0



permite la carga en paralelo

con que uno de los dos esté a cero ya no cuenta

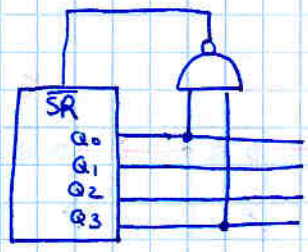
tambien tiene salida TC → terminal count output → vale 1 cuando CET vale 1 y se llega al final de la cuenta

ej: hacer contador módulo 10 con LS163: módulo 16, reset síncrono

¡Cuidado: Modulo 10 = 0 → 9

Puerta que detecte el 9 y active el reset

9 = 1001

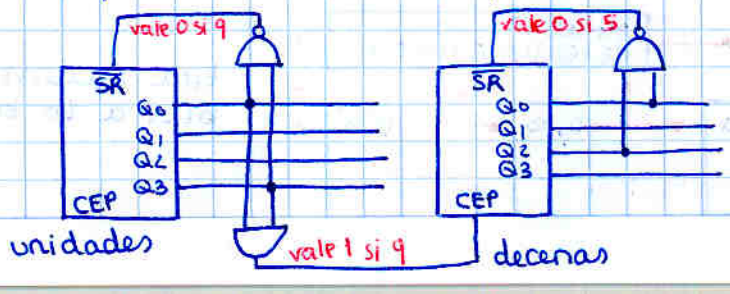


si fuera el LS161, asíncrono, la puerta debería detectar el 10

ej: contador módulo 60 (0 → 59) con LS163: módulo 16, reset síncrono

No sirve el TC para cascadarlo, hay que hacerlo manual

5 = 0101



Contadores usando registros de desplazamiento

Idea: se utiliza una codificación de los estados que sea diferente al binario natural pero más facil de implementar

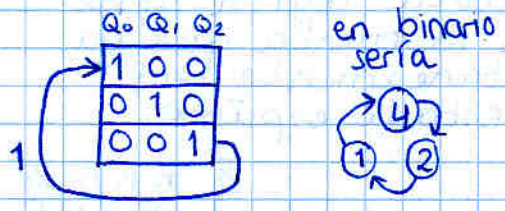
Ventajas: sencillo y muy rapido

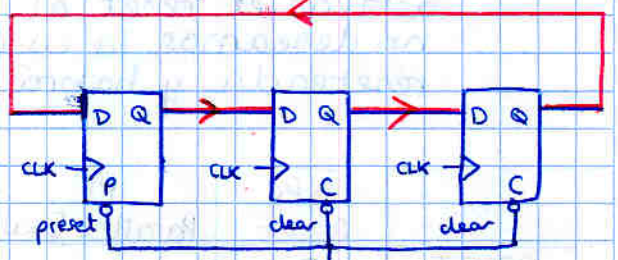
Desventajas: requiere muchos flipflops

Contador en anillo

módulo = nº de flip flops

se hace con registro de desplaz así:





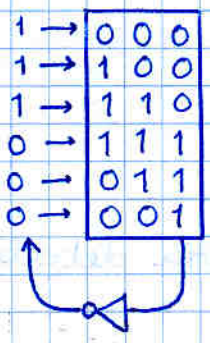
con CI, basta conectar la salida Q_2 con la entrada de datos serie

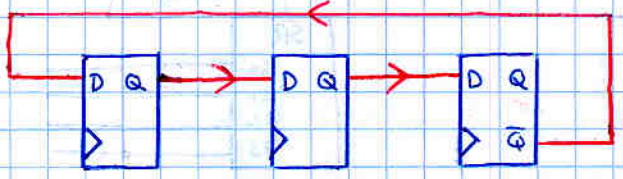
en el LS195 la entrada de datos serie es la unión de las entradas J $\overline{K}$

no hay que olvidarse de iniciar la cuenta para que el contador funcione. En un CI. se haría un parallel Load 1 0 0

Contador Johnson

aprovecha mejor los FF : **n flip-flops → 2n cuentas**



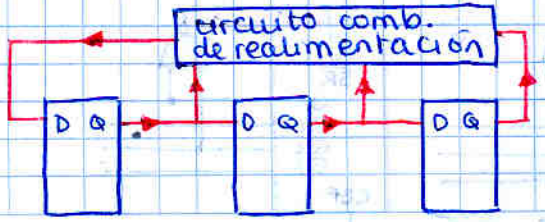


para iniciar la cuenta es un simple reset a los 3

gran ventaja de estos contadores: $t_{co} = 0$ → son rapidísimos

$$f_{max} = \frac{1}{t_p + t_{su}}$$

LFSR

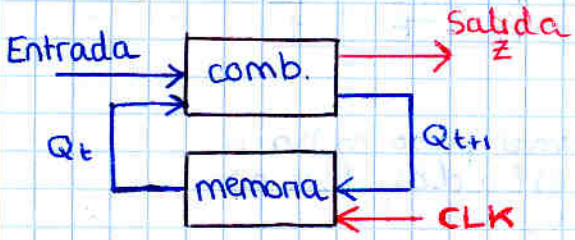


tira secuencias de bits a lo bruto

TEMA 7. ANALISIS Y SINTESIS DE CIRCUITOS SECUENCIALES SINCRONOS

Maquinas de estado o autómatas.
Tienen dos estructuras básicas.

MEALY

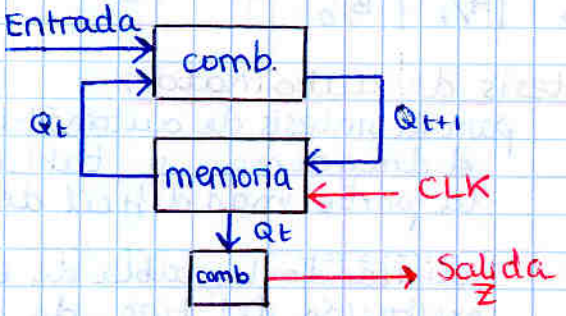


La salida cambia a la vez que la entrada (asíncrona) y depende de la entrada y Qt

$$Z = f(Q_t, E)$$

Qt: síncrono; E: asíncrono; Z: asíncrono

MOORE

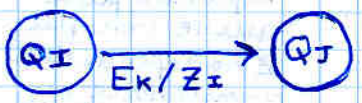


La salida cambia a la vez que el estado.
Cada estado tiene una salida.
Como los estados son síncronos la salida también lo es

$$Z = f(Q_t)$$

Z: síncrono; Qt: síncrono

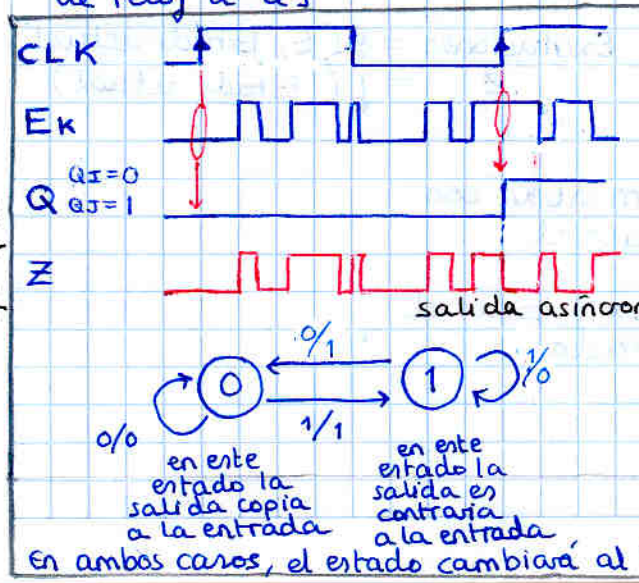
Diagrama de estados y tabla de estados

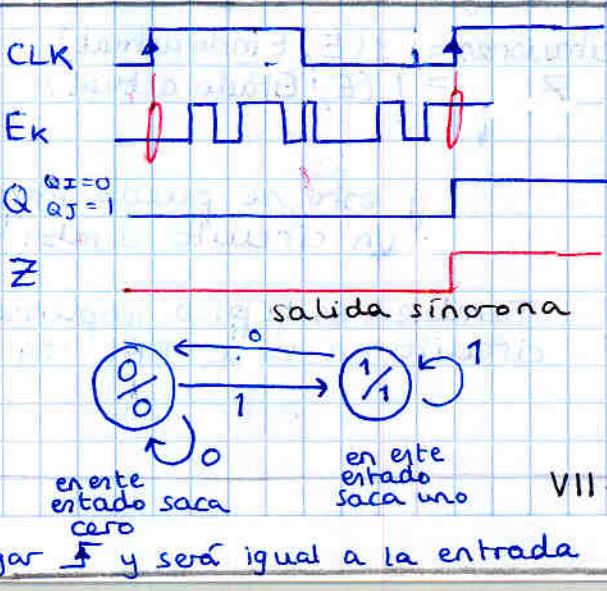


estoy en Q_I.
Si llega en este ciclo E_K, sacaré Z_I instantaneamente.
Variaré Z_I tantas veces como varíe E_K en un mismo ciclo.
Al final del ciclo, si la entrada es E_K (estaré sacando Z_K) me iré en el flanco de reloj a Q_J

Estoy en Q_I sacando su salida asociada Z_I durante todo el ciclo de reloj.
No me importan los cambios de E_K durante el ciclo.
Al llegar el flanco, si la entrada es E_K me iré a Q_J, y empezaré a sacar la salida Z_J asociada al estado Q_J

ejemplo



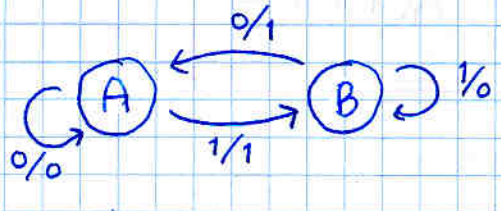


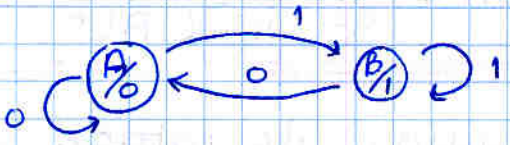
En ambos casos, el estado cambiará al llegar ↑ y será igual a la entrada

Mealy

Moore

Tabla de estados a partir del diagrama





Q(t)	E=0	E=1
A	A/0	B/1
B	A/1	B/0

se pone Q_{t+1}/Z

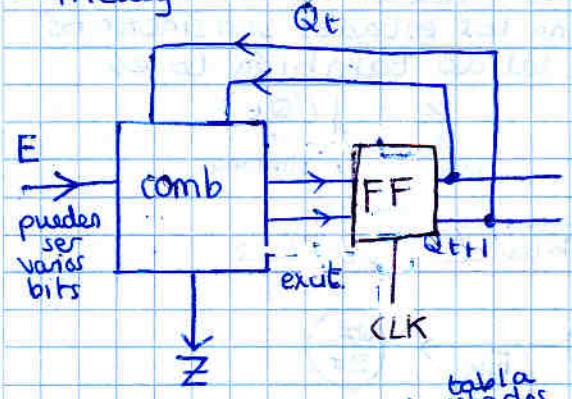
Q(t)	E=0	E=1	Z
A	A	B	0
B	A	B	1

Síntesis de autómatas

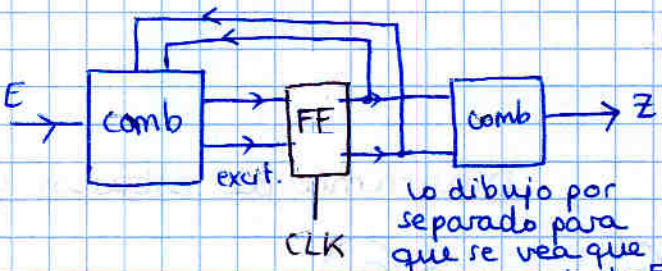
Para la síntesis de autómatas, el primer paso es hacer el diagrama y tabla de estados. Esa es la parte mas difícil de pensar.

utilizando la tabla de estados y la tabla de excitación del tipo de FF que usemos, se hace la Tabla de funcionamiento global

Mealy



Moore



lo dibujo por separado para que se vea que Z no depende de E
Normalmente es un unico circuito comb.

tabla de estados

tabla de excitaciones del tipo de FF

Entradas		Estado siguiente	Excitaciones	Z
E.	Estado actual			

por Karnaugh se saca :

en Mealy :

Excitaciones = f(E, Estado actual)

Z = f(E, Estado actual)

en Moore :

Excitaciones = f(E, Estado actual)

Z = f(Estado actual)

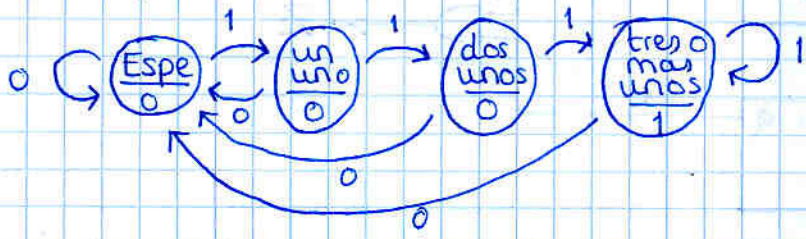
y esto se puede implementar con un circuito combinacional

Tambien es típico implementar todo el circuito en una PAL registrada

ejemplo. Detector de secuencia de tres o más unos consecutivos.

- por Moore:

1. Diagrama de estados



nº de estados: 4
nº de FF: $2^n \geq 4 \rightarrow n = 2$
tipo FF: D

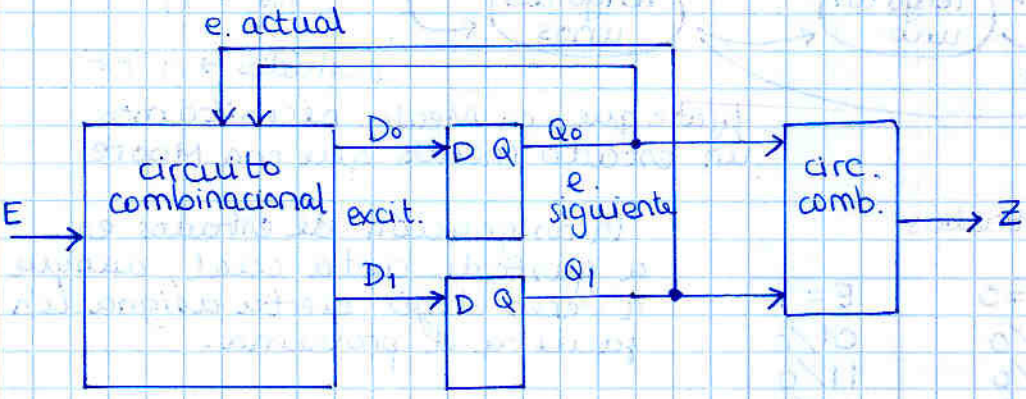
2. Asignación de estados y tabla de estados:

Espe = 00
un uno = 01
dos unos = 10
tres o + 1's = 11

siempre es bueno que el de arranque sea 00 para hacer reset.

e. actual	e. siguiente e=0	e. siguiente e=1	Z
00	00	01	0
01	00	10	0
10	00	11	0
11	00	11	1

3. Tabla de funcionamiento



Entradas E	Entradas E. actual Q_1	Entradas E. actual Q_0	E. siguiente Q_1	E. siguiente Q_0	Excitaciones D_1	Excitaciones D_0	Salida Z
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	1	0	0	0	0	0	0
0	1	1	0	0	0	0	1
1	0	0	0	1	0	1	0
1	0	1	1	0	1	0	0
1	1	0	1	1	1	1	0
1	1	1	1	1	1	1	1

las excitaciones coinciden con el estado siguiente por ser tipo D

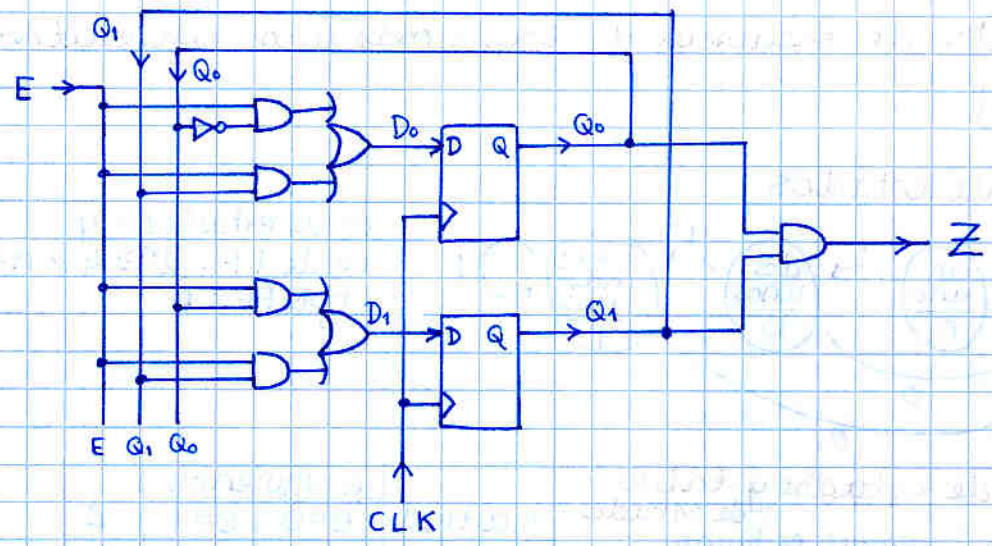
4. Karnaugh

$Q_1 Q_0$ \ E	D_1 0	D_1 1	D_0 0	D_0 1	Z 0	Z 1
0 0	0	0	0	1	0	0
0 1	0	1	0	0	0	0
1 1	0	1	0	1	1	1
1 0	0	1	0	1	0	0

forma de ahorrar espacio. 3 Karnaugh en 1

$D_1 = E \cdot Q_0 + E \cdot Q_1$
$D_0 = E \overline{Q_0} + E Q_1$
$Z = Q_1 Q_0 \rightarrow$ no depende de E ↳ es Moore

5. Implementarlo (solución Moore)



- **por MEALY**

1. Diagrama de estados

Asignación de estados
Espe = 00
1 uno = 01
2 unos = 11

3 estados → 2 FF

fijate que con Mealy necesitamos un estado menos que con Moore

2. Tabla de estados

la asignación de estados es a gusto de cada cual, aunque a veces elegir cierta asignación facilita el problema.

E.actual	E=0	E=1
00	00/0	01/0
01	00/0	11/0
11	00/0	11/1

3. Tabla de funcionamiento global

Entradas			E.sig ≡ Excitaciones		Salida
E	E.actual Q_1	E.actual Q_0	$Q_1 \equiv D_1$	$Q_0 \equiv D_0$	Z
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	X	X	X
0	1	1	0	0	0
1	0	0	0	1	0
1	0	1	1	1	0
1	1	0	X	X	X
1	1	1	1	1	1

nota; si el FF no fuera tipo D, el estado siguiente no coincidiría con las excitaciones. Habría q obtenerlas con tabla de excit.

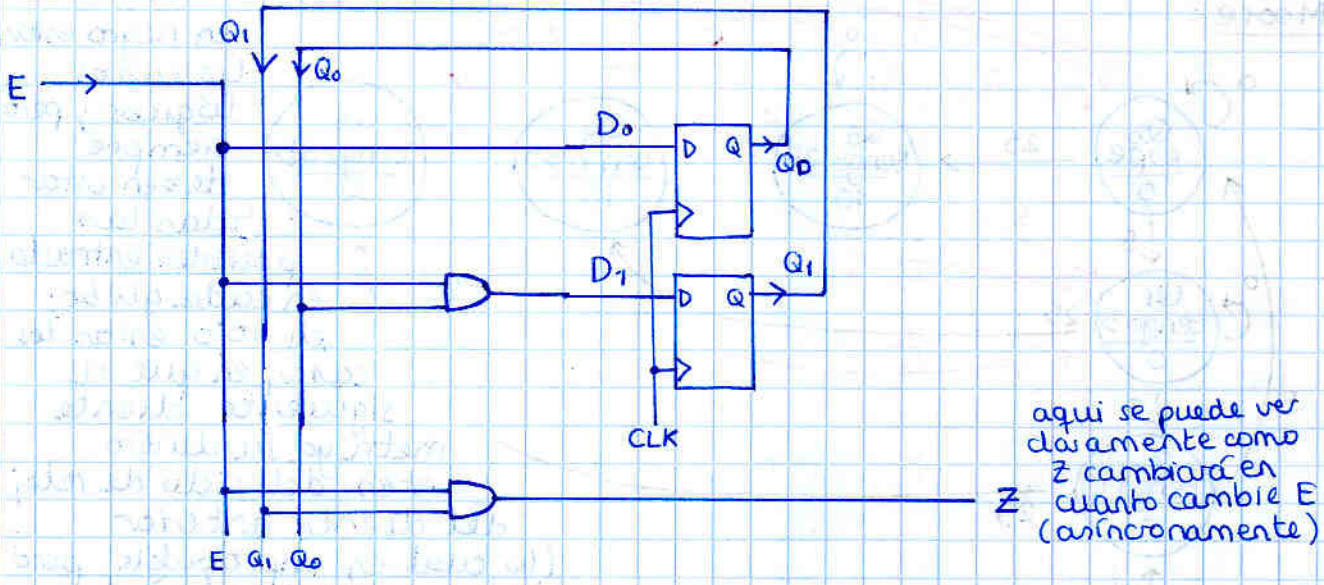
← estados no esperados, al poner X estamos haciendo implementación mínimo coste

4. Karnaugh

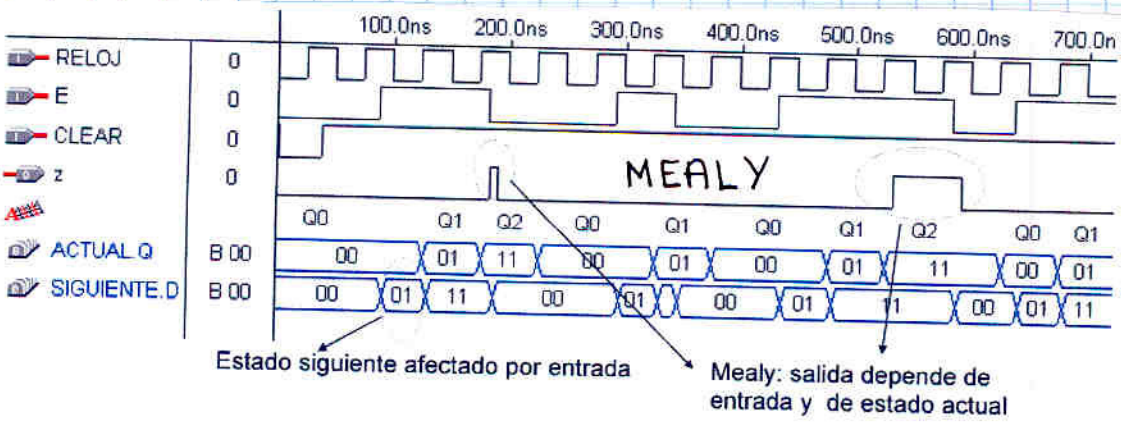
		D_1		D_0		Z	
Q_1Q_0 \ E		0	1	0	1	0	1
0	0	0	0	0	1	0	0
0	1	0	1	0	1	0	0
1	1	0	1	0	1	0	1
1	0	X	X	X	X	X	X

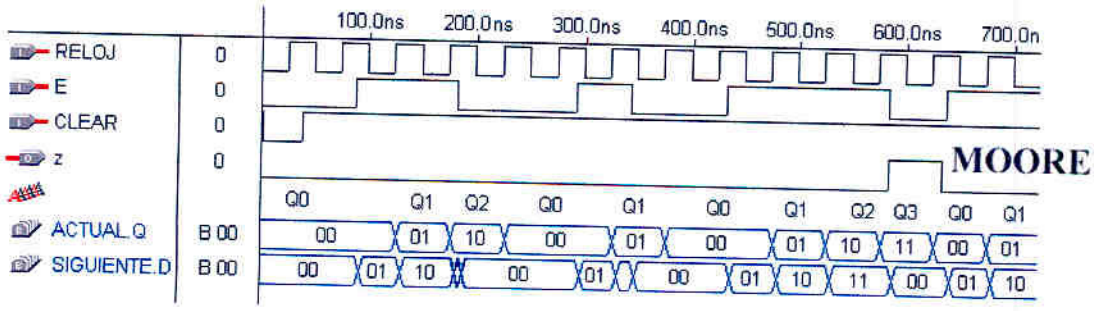
$D_1 = E \cdot Q_0$
$D_0 = E$
$Z = E \cdot Q_1$
↳ es Mealy. Z sí depende de la entrada

5. Implementarlo (solución Mealy)



• Diferencia en el funcionamiento de ambas soluciones





Vemos que Mealy hace una pequeña detección y sin embargo se va luego al estado cero; esto es porque cuando entra al estado 2 unos, en cuanto ve que hay 1 a la entrada salta la detección, pero en cuanto ve que el 1 se va dice: ¡ay no! y deja de sacar una detección, cuando llega el flanco de reloj, la entrada es cero y por ello va al estado cero.

Pasar de estado → síncrono
Salida Z → asíncrona

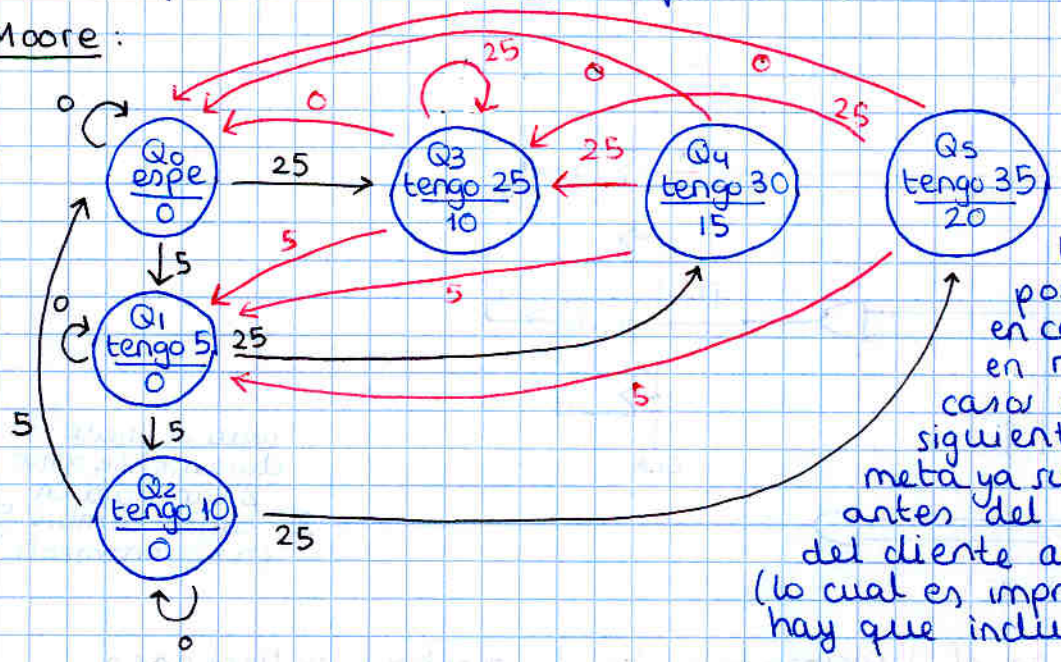
sólo detecta

En Moore sin embargo cuando hay 3 unos consecutivos durante tres flancos de reloj consecutivos, y la señal de detección dura exactamente un ciclo, Moore en general "no se anda con tonterías y le gusta lo preciso, claro y síncrono"

ejercicio: lo que devuelve el cambio en una máquina de refrescos.

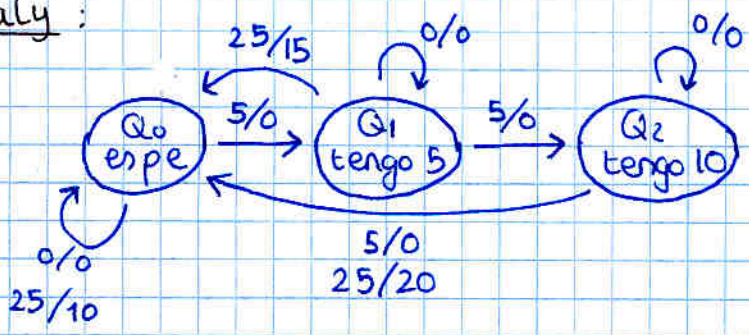
Todos los productos valen 15. Sólo acepta 5 ó 25

Moore:



en negro están los casos lógicos; pero siempre deben estar todas las posibles entradas en cada globo; en rojo están los casos en que el siguiente diente meta ya su dinero antes del ciclo de reloj del diente anterior (lo cual es improbable, pero hay que incluirlo)

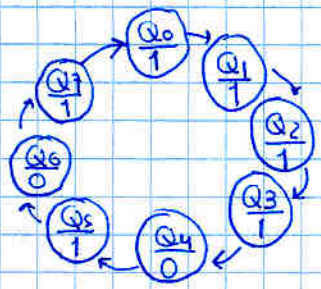
Mealy:



Vemos como en este caso Mealy es mucho más lógico

ejercicio: generador de secuencia 1 1 1 1 0 1 0 1

Moore

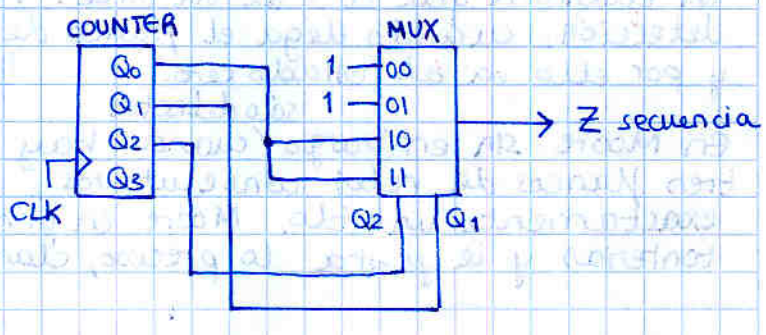


nos lo piden montar con
1 x 74F163 → contador módulo 16 reset síncrono
1 x 74F153 → 2 mux de 4 a 1.

$Z = \overline{Q_2} + Q_0$

Q_0, Q_1 y Q_2 sacarlos del contador ignorando Q_3 para tener módulo 8
Z hacerlo con el MUX
(la mejor forma es mirando la tabla)

Q_2	Q_1	Q_0	Z
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



Karnaugh

Q_2 \ Q_1Q_0	00	01	11	10
0	1	1	1	1
1	0	1	1	0

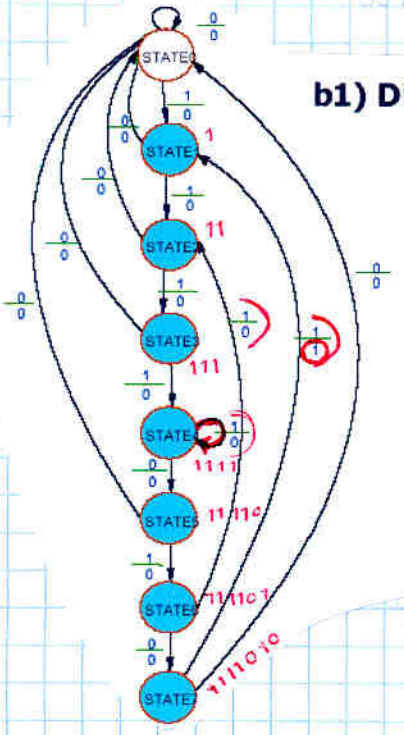
ejecicio: Detector de secuencia CON SOLAPE

1 1 1 1 0 1 0 1

por ej si llega 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 0 1 0 1

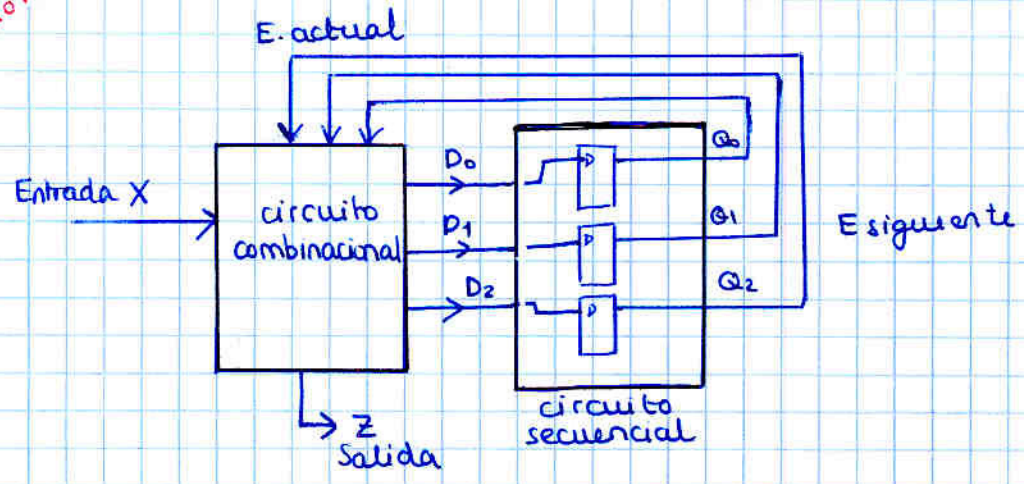
llevo un uno · llevo dos · llevo tres · llevo cuatro · llevo cuatro (por no ser cero no debe volver a empezar) · llevo 4 unos y un cero · salida 1 · salida 1

8 estados = 3 FF

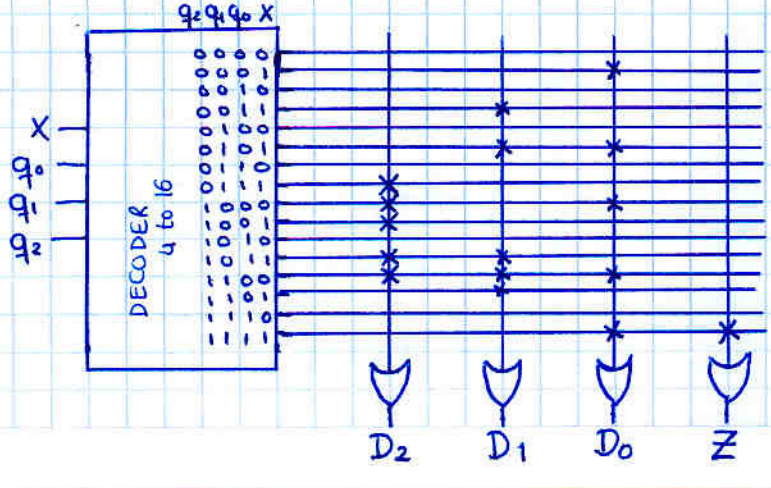


b1) Diagrama de Estados y tabla de estados

	Estado actual			Entrada	Estado siguiente			Excitación			Salida
	Q_{2t}	Q_{1t}	Q_{0t}	X	Q_{2t+1}	Q_{1t+1}	Q_{0t+1}	D_2	D_1	D_0	Z
STATE0	0	0	0	0	0	0	0	0	0	0	0
STATE0	0	0	0	1	0	0	1	0	0	1	0
STATE1	0	0	1	0	0	0	0	0	0	0	0
STATE1	0	0	1	1	0	1	0	0	1	0	0
STATE2	0	1	0	0	0	0	0	0	0	0	0
STATE2	0	1	0	1	0	1	1	0	1	1	0
STATE3	0	1	1	0	0	0	0	0	0	0	0
STATE3	0	1	1	1	1	0	0	1	0	0	0
STATE4	1	0	0	0	1	0	1	1	0	1	0
STATE4	1	0	0	1	1	0	0	1	0	0	0
STATE5	1	0	1	0	0	0	0	0	0	0	0
STATE5	1	0	1	1	1	1	0	1	1	0	0
STATE6	1	1	0	0	1	1	1	1	1	1	0
STATE6	1	1	0	1	0	1	0	0	1	0	0
STATE7	1	1	1	0	0	0	0	0	0	0	0
STATE7	1	1	1	1	0	0	1	0	0	1	1



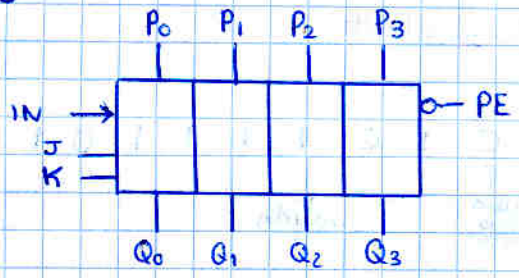
Nos piden implementar el circuito combinacional con una ROM y el circuito secuencial con un 1195 (registro de desplazamiento)



- como es tan facil implem. una función con una ROM no hace falta Karnaugh
- Solo hay que poner en cada función de salida los minitérminos correspondientes.

en cuanto al circuito secuencial; disponemos de:

1195



la JK permiten elegir qué aparecerá en Q0 cuando se desplace hacia la derecha.

Usaremos el 1195 como 4 simples FF tipo D, es decir, siempre en modo carga paralelo, sin utilizar desplazamiento.

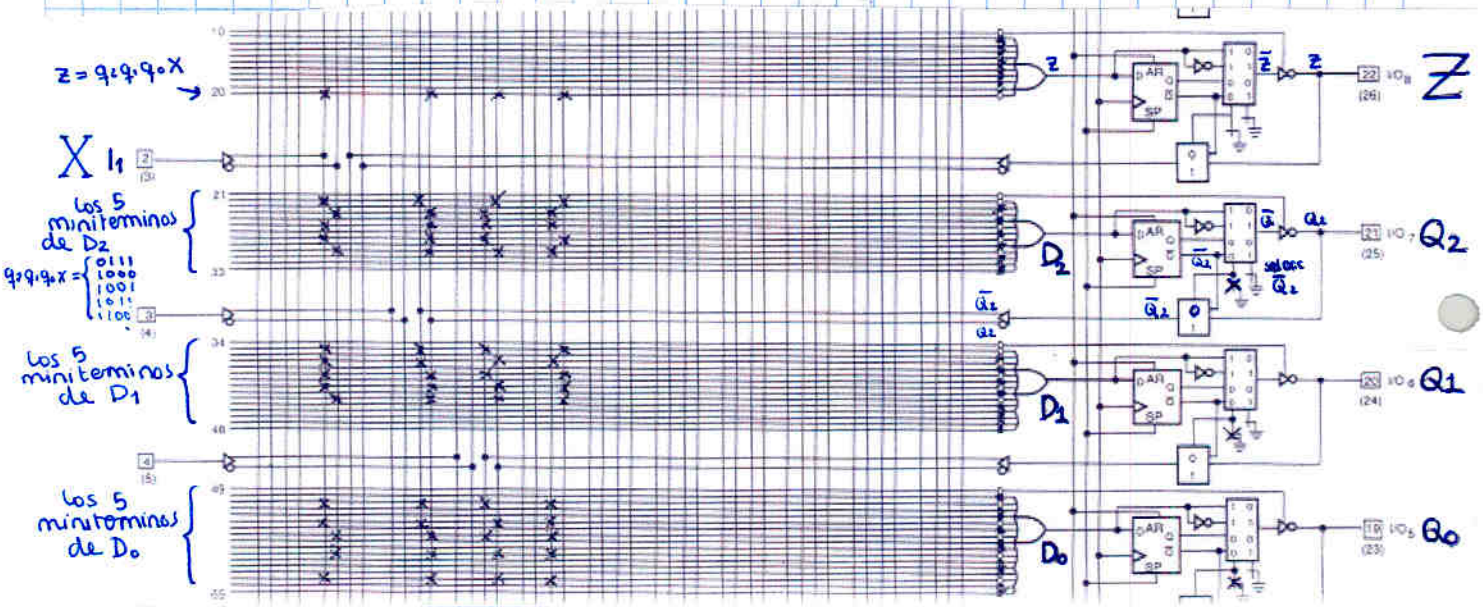
b) Implementar todo el detector sólo con una PAL registrada

de nuevo, no hace falta Karnaugh, sólo incorporar Q_2, Q_1, Q_0 con sus miniterminos

ej: $Q_2 = \begin{matrix}0111\\1000\\1001\\1011\\1100\\ \hline q_2 q_1 q_0 x\end{matrix} = \bar{q}_2 q_1 q_0 x + q_2 \bar{q}_1 \bar{q}_0 \bar{x} + q_2 \bar{q}_1 \bar{q}_0 x + q_2 \bar{q}_1 q_0 x + q_2 q_1 \bar{q}_0 \bar{x}$

Hay que asegurarse de que los MUX de las salidas realimenten bien q_2, q_1 y q_0.

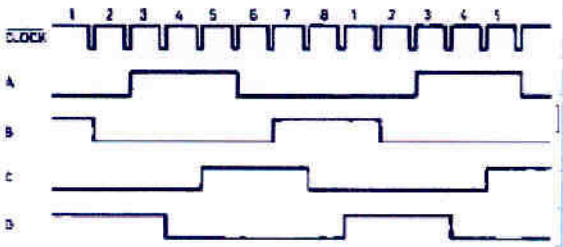
En este ejemplo se realimenta $\bar{q}_2$, $\bar{q}_1$ y $\bar{q}_0$ pero no pasa nada ya que luego se dispone de la afirmada y la negada



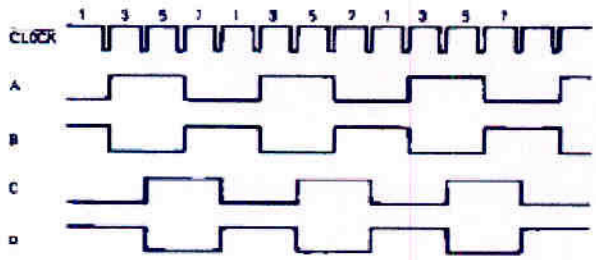
más ejemplos

Diseño motor "paso a paso" con entrada "HALF" que determina el modo

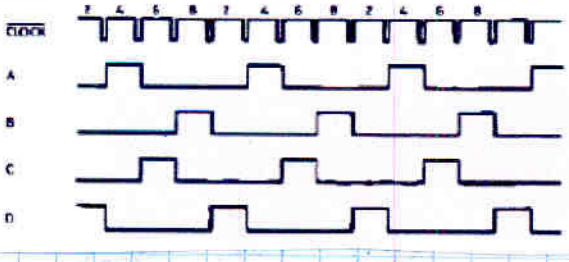
Medio paso



Dos fases



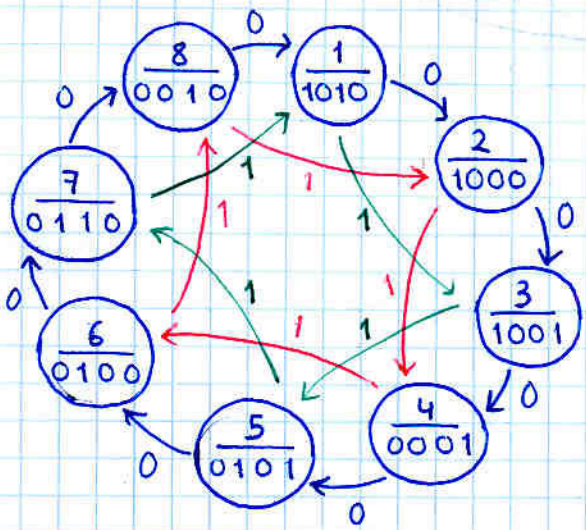
Fase única



el motor funciona en uno de sus 3 modos según la entrada HALF

HALF=0	Modo medio paso
HALF=1 y estado impar	Modo de dos fases
HALF=1 y estado par	Modo fase única

El diagrama de estados por Moore sería:



leyenda

Vemos que la 'salida' de este autómata son en realidad 4 salidas

Una posibilidad sería llamar a los estados igual que la salida; así nos ahorramos la combinacional de salida

Aunque probablemente, en este caso eso no haría más que empeorar la situación, ya que la combinacional de entrada tendría 5 variables

ej: Detector de dos secuencias

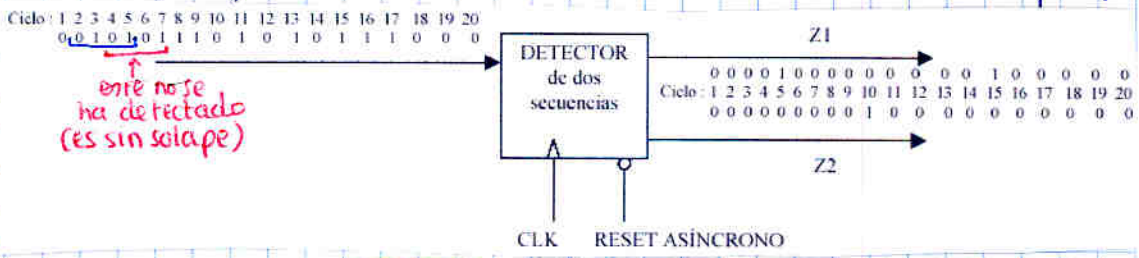
si entrada 0101 → $\{Z_1 Z_2\} = 10$

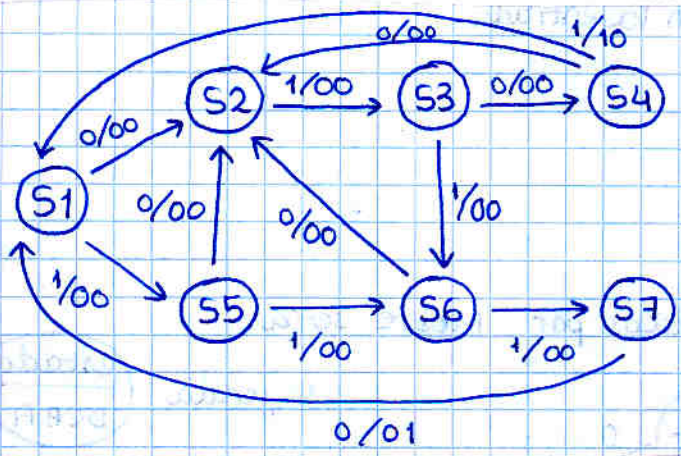
si entrada 1110 → $\{Z_1 Z_2\} = 01$

nos indica que es Mealy

Que detecte la secuencia en el mismo ciclo que ha entrado el ultimo bit

Que sea SIN SOLAPE, (en el examen habría que darse cuenta de ello mirando el ejemplo





x	Q(t)2	Q1(t)	Q0(t)	Q2(t+1)	Q1(t+1)	Q0(t+1)	D2	D1	D0	Z1	Z2
0	0	0	0	0	0	1	0	0	1	0	0
0	0	0	1	0	0	1	0	0	1	0	0
0	0	1	0	0	1	1	0	1	1	0	0
0	0	1	1	0	0	1	0	0	1	0	0
0	1	0	0	0	0	1	0	0	1	0	0
0	1	0	1	0	0	1	0	0	1	0	0
0	1	1	0	0	0	0	0	0	0	0	1
0	1	1	1	X	X	X	X	X	X	X	X
1	0	0	0	1	0	0	1	0	0	0	0
1	0	0	1	0	1	0	0	1	0	0	0
1	0	1	0	1	0	1	1	0	1	0	0
1	0	1	1	0	0	0	0	0	0	1	0
1	1	0	0	1	0	1	1	0	1	0	0
1	1	0	1	1	1	0	1	1	0	0	0
1	1	1	0	1	1	0	1	1	0	0	0
1	1	1	1	X	X	X	X	X	X	X	X

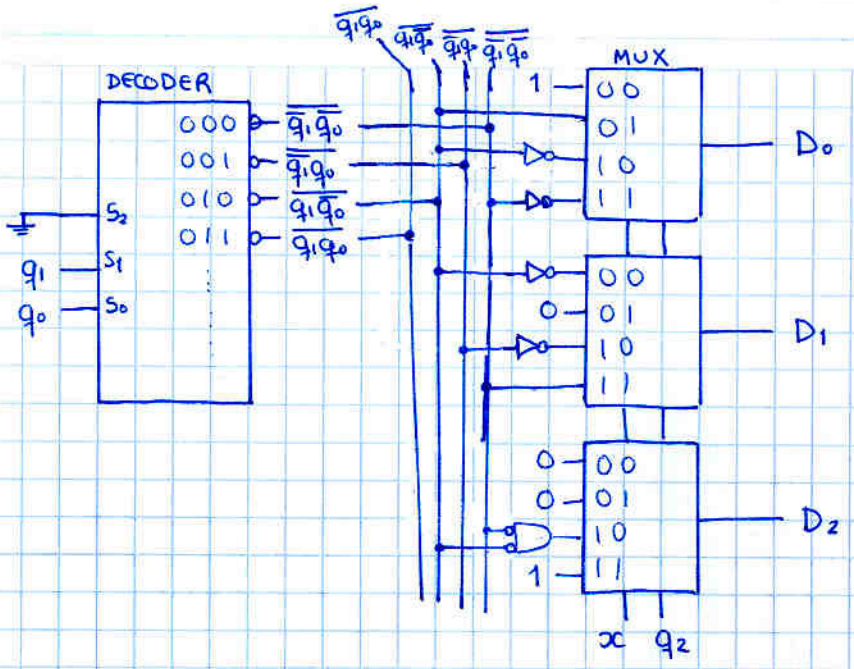
se puede obtener directo de la tabla

para implementar con un MUX y un decodificador

D_2: xq_2 { 00 → 0; 01 → 0 (tomando X=0); 10 → $\overline{q_0} = \overline{q_1}\overline{q_0} + q_1\overline{q_0}$; 11 → 1 (tomando X=1)

D_1: xq_2 { 00 → $q_1\overline{q_0}$; 01 → 0 (X=0); 10 → $\overline{q_1}q_0$; 11 → $\overline{q_0}q_1 + q_1\overline{q_0} + q_1q_0 = \overline{\overline{q_1}\,\overline{q_0}}$ } además hay que tomar X=1

D_0: xq_2 { 00 → 1; 01 → $\overline{q_1}\,\overline{q_0}$ (X=1); 10 → $q_1\overline{q_0}$; 11 → $\overline{q_1}\overline{q_0}$ }



Análisis de sistemas secuenciales síncronos

ej: Hallar diagrama de estados de:

y se da un circuito.

Hay que hallar las ecuaciones de las excitaciones y ver cuantos estados hay (nº flipflops), y la ec. de la salida

Sabiendo la función de las excitaciones en función de la entrada, podemos saber el estado correspondiente a cada entrada/s.
Igualmente podemos saber la salida de cada entrada/estado

⇒ podremos dibujar el diagrama de estados.