ETSI Telecomunicación

DCSE
(Diseño de Circuitos y Sistemas Electrónicos)

Apuntes de Pak

Diseño de Circuitos y Sistemas Electrónicos (DCSE)

Apuntes de Pak (Fco. J. Rodríguez Fortuño)
ETSI Telecomunicación. Universidad Politécnica de Valencia.
Segundo cuatrimestre de 3er curso
Curso 2005/2006

Contenido:

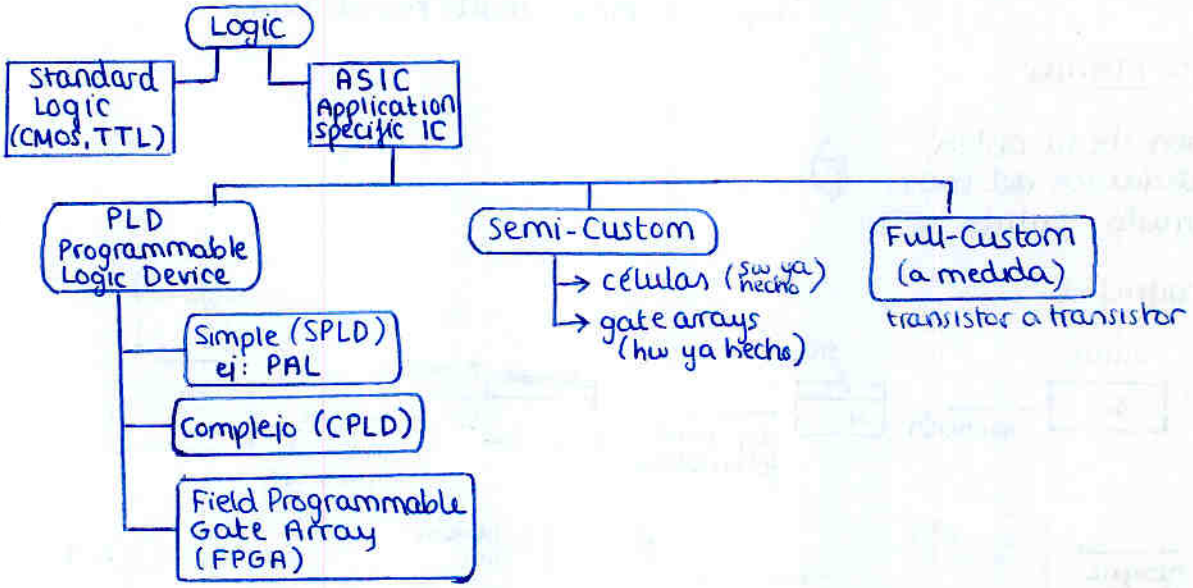
- Resumen de algunos temas de teoría: advierto que es un resumen basado en las notas que tomé de forma rápida sobre las transparencias en clase, así que es muy posible que contengan datos incorrectos.
- Resumen de algunas prácticas: escritos a toda prisa en las propias prácticas

Fecha de última actualización: 30 Septiembre 2007

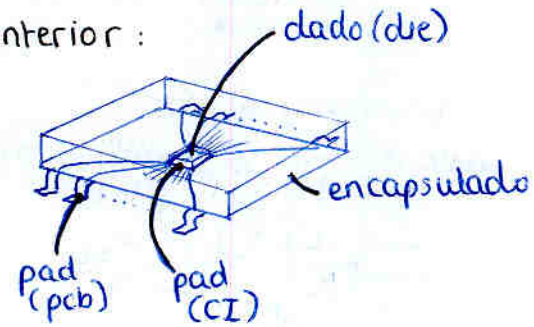
TEMA 1. Introducción a la Fabricación y a las Tecnologías de Circuitos Integrados

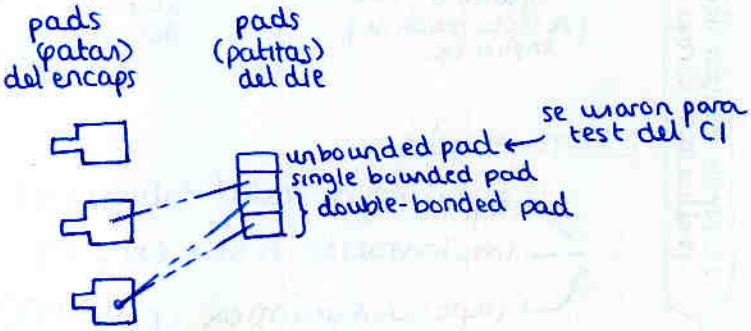
1. Introducción

Tipos de circuitos integrados:



El interior :



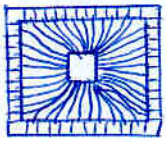


Dado vs Layout :

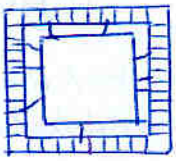
Dado : diseño físico en silicio
Layout : aspecto final del dado, representación gráfica

Limitación del diseño

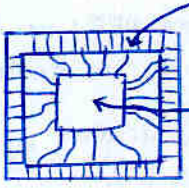
Pad limited



Core limited



Floorplan



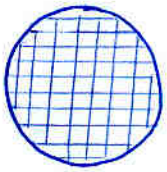
- Linea de distribución de CLK
- Buses de PWR
  - → limpios : alimentar células dimensiones definidas
  - → sucios : mas gruesos apantallamiento

2. Fabricación de circuitos integrados. Proceso Planar

Se realiza en la Fundición de Silicio

Oblea (wafer) - Disco de silicio sobre el que crecerán los transistores

En una oblea se hacen varios chips (los q quepan)



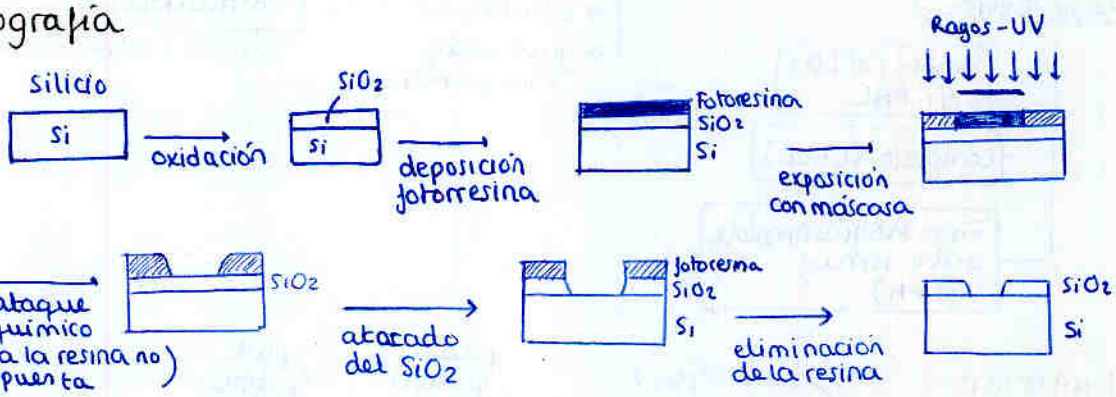
Siempre mismo chip { SPC – Single Project Chip
MPC – Multi Project Chip
↳ cada chip tiene varios proyectos q decidieron fabricarse juntos para abaratar costes

Distintos chips : MPW – Multi Project Wafer

El proceso Planar

A. Preparación de la oblea
- Fabricación del tocho
- serrado, pulido

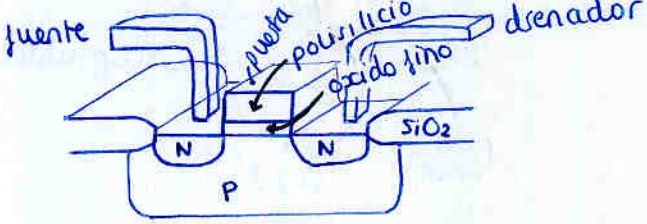
B. Litografía



se repite tantas veces como máscaras tenga

C. Procesado
- Deposición de metal (aluminio)
- Implantación iónica (N^- o P^+)
- Deposicion química (polisilicio)

ej: transistor MOS



D. Post procesado

· Test → 1º sobre PEDs → Super Transistores
↳ Luego de los dados: con punta de test

Rendimiento (Yield (%)) $Y(\%) = 100 \cdot \frac{N^\circ \text{dados buenos}}{N^\circ \text{total dados oblea}}$

siempre MOS mejor que Bipolar

$D \equiv$ defectos $/cm^2$

$A \equiv$ area del dado (cm^2) → $A \uparrow \Rightarrow Y(\%) \downarrow$

3. Tecnologías digitales sobre silicio en la era VLSI

¿Por qué Silicio?

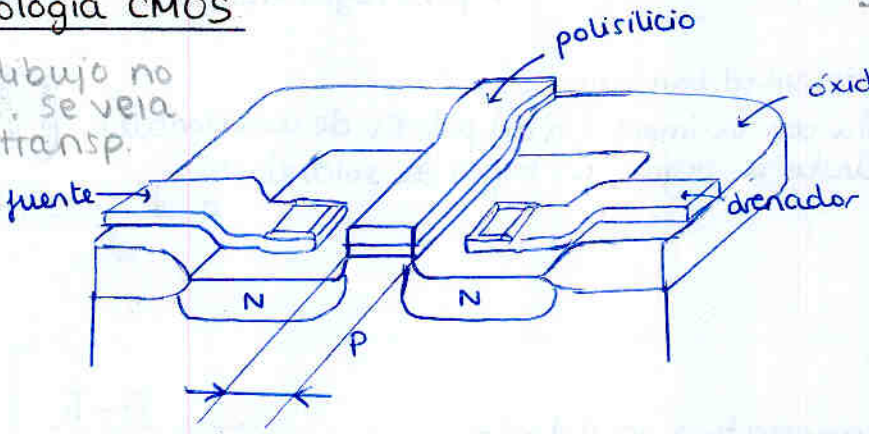
Germanio: - mejor movilidad
- PERO el oxido no crece bien → sin aislante entre capas no hay chip

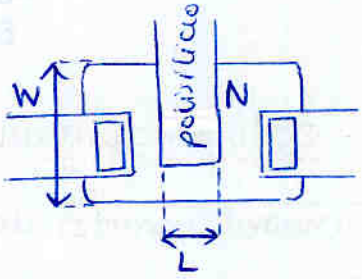
Arsénico de galio: - consume poco y va muy rápido
- caro y escaso

Aplicaciones:
- VHSICS: very high speed IC
- Microondas
- Fotónica

• Tecnología CMOS

Nota: dibujo no fiable, se veía mal en transp.



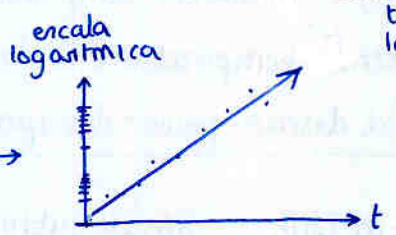


$L = 2\lambda$ ↳ resolución del dibujo ≈ 1'00 a 0'2 μm

$I_{ds} = f(W/L)$
↳ i.e. no importa el tamaño sino la relación !!!

Evolución:

nº transistores:
tamaño del dado:
frecuencia:
} crece exponencialmente →



• Tecnología bipolar

- $I_{ES} = f(W_B)$ (↑ anchura de base) : El tamaño SI importa en este caso

Aplicaciones: electrónica analógica y radiofrecuencia

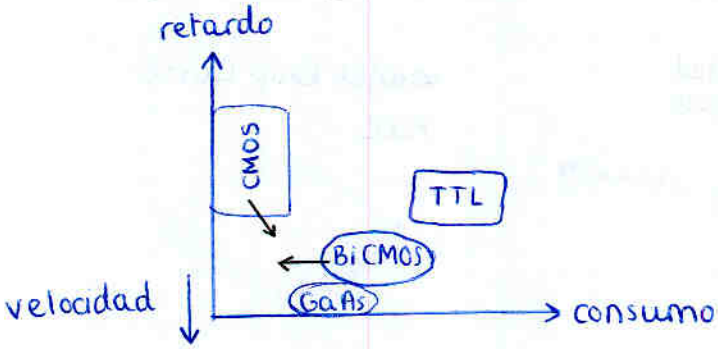
- Fabricación más compleja que CMOS

• Tecnología BiCMOS

- combinación bipolar y CMOS
- En general: - 'interior' CMOS
  - 'exterior' = PLL's, A/D's, etc... Bipolar

Más costoso pero más aplicaciones

• comparación y tendencia



4. Técnicas y tipos de encapsulado

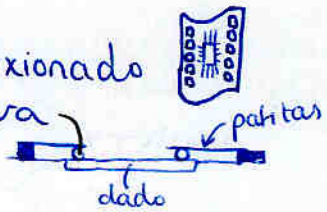
- Dado DIRECTAMENTE sobre placa (DCA)
  - Smart Cards (móviles, VISA)

  - Chip on board (COB) → gota negra de plástico para proteger)

Técnicas: • Wire bonding

+ gota negra encima

  - Tape-automated bonding
    - Cinta con la impresión del patrón de conexionado
    - El dado se pega con topes de soldadura



- Dado encapsulado

Thermal Characterisation

T_J : junction : máxima temperatura en el dado

T_C : package : máxima temperatura en encapsulado

T_A : ambient : temperatura ambiente

P_d : total device power dissipation (W)

$$\theta_{JC} = \frac{T_J - T_C}{P_d}$$

$$\theta_{JA} = \frac{T_J - T_A}{P_d}$$

$$T_{Amax} = T_{Jmax} - P_d \cdot \theta_{JA}$$

$$P_{dmax} = \frac{T_{Jmax} - T_A}{\theta_{JA}}$$

Dual In Line

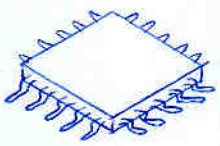
DIP

Small Outline Package

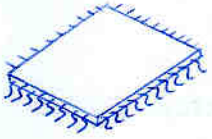
SOIC

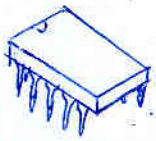
Quad Flat Pack

VQFP
TQFP
HTQFP



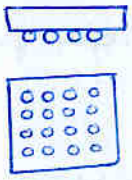
PQHQFP





Ball Chip Scale Package

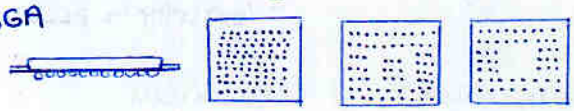
CS 48



CS 144

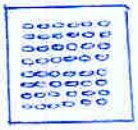
Ball Grid Array

BGA



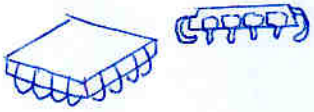
Pin Grid Array

PGA



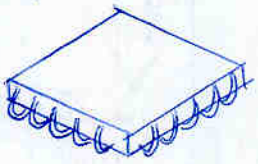
Ceramic Leaded Chip Package

CC



Leadless Chip Carrier

PLCC



Resumen conceptos tema 3

Arquitecturas básicas: →

PROM: AND fijo (cada uno de los miterminos = direcciones), OR programable

PLA: AND progr, OR progr → lógica multisalida → difícil

PAL: AND progr, OR fijo → despilfarro de P.T. → ideal en nuestro tiempo

Tecnologías:

- Bipolar
- Antifusible BIP
- CMOS (pseudo CMOS)
- EPROM
- EEPROM
- Flash

Primeras PLD

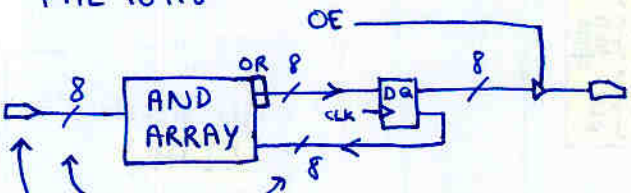
PAL

ej: 16 R 8 ← salidas

↑ entradas dedicadas (i.e. sólo entradas) + entradas realimentadas

16 H 8 → salidas nivel alto
16 L 8 → " " bajo
16 R 8 → " registradas

PAL 16R8



8 + 8 = 16 entradas al AND array

8 entradas dedicadas

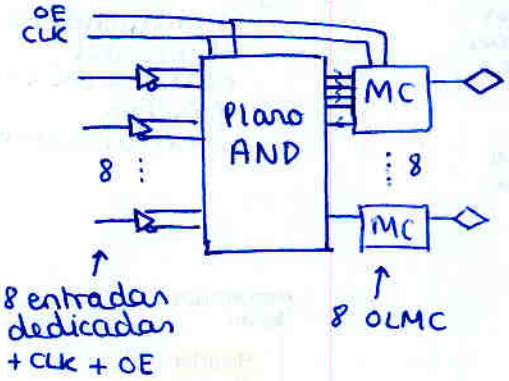
SPLD (PAL universales) (GAL)

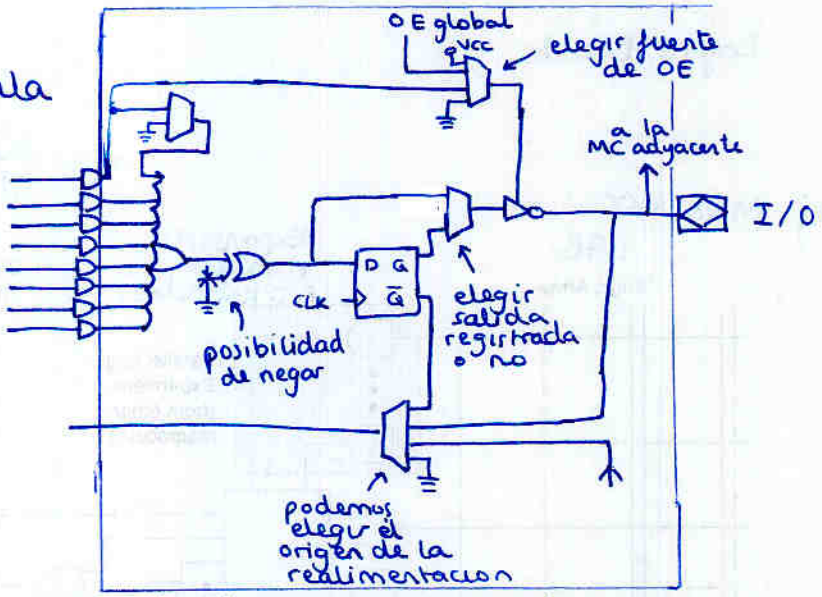
emula TODO lo anterior

pueden configurar su salida para 'imitar' a cualquier PAL anterior i.e. salida nivel alto, bajo, registrada, etc...

ej GAL 16V8

introduce concepto de macrocélula





Output Logic Macrocell

ej PAL 22V10 ↑ 10 macroceldas

- Incluía preset y reset global por 1ª vez

Crecieron hasta lo absurdo (PAL 64R32 → fracasó → imposible mantener velocidad y baja penalización de área con única matriz)

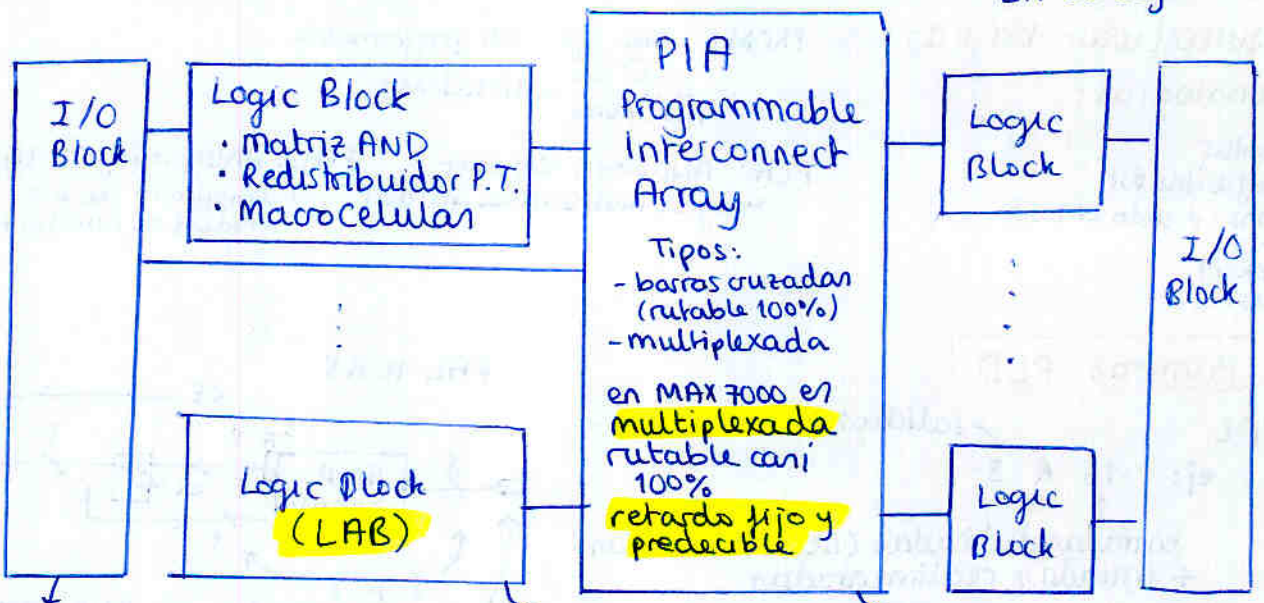
⇓

solución: esquema multinivel

CPLDs: Familia MAX 7000

Bits especiales
- Bit Turbo
- Bit de seguridad

nomenclatura: EPM 7 128 S LC 84 - 7
- EPM 7 → pld fam 7000
- 128 → nº LC's
- S → vel
- LC 84 → nº pines



desde CPLDs con macroceldas rígidamente asociadas a una célula de E/S hasta otras donde las MC llevan salidas a la PIA para redirigirlas a I/O

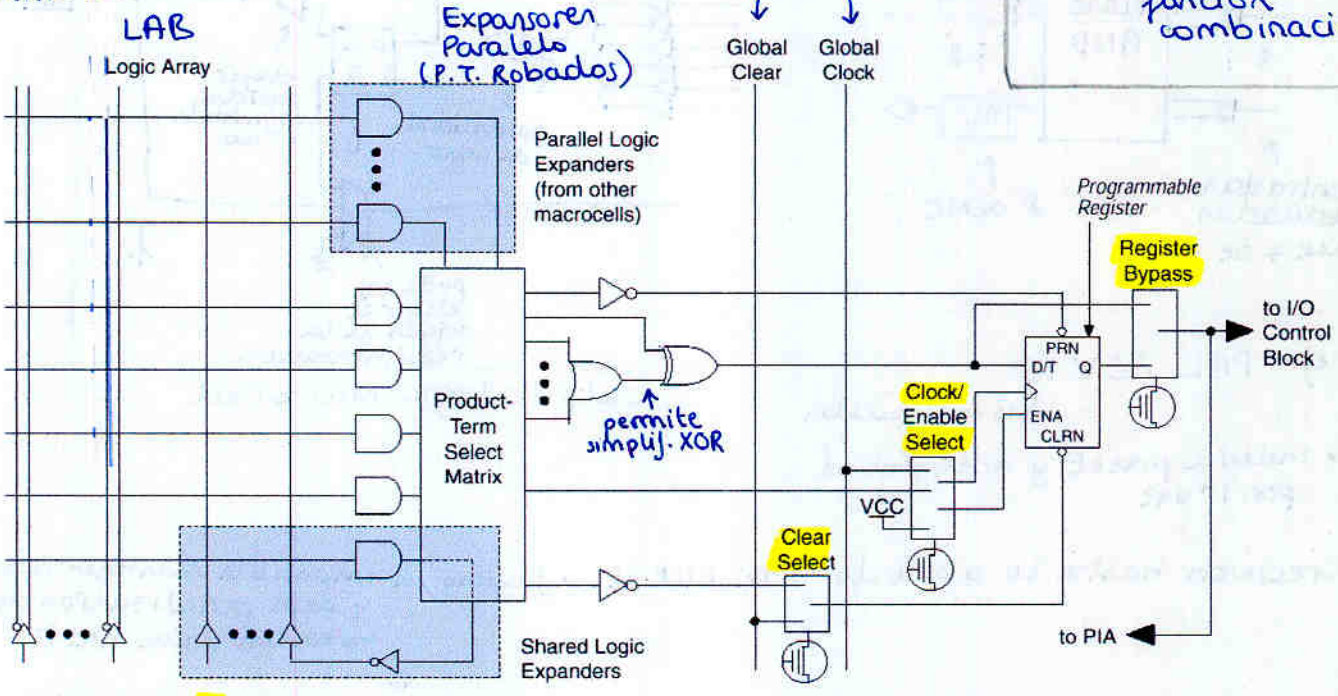
- lo MAX 7000 es sin redireccionamiento de E/S.
- E/S se configuran
  - input / output / bidir
  - slew rate control
  - open drain control (MAX 7000S)

rutabilidad permite pin locking capability
↳ colocar las señales en el pin que queramos

Logic Block

en general CPLD's carece de Register Packing
↳ independizar el uso del registro de la función combinacional

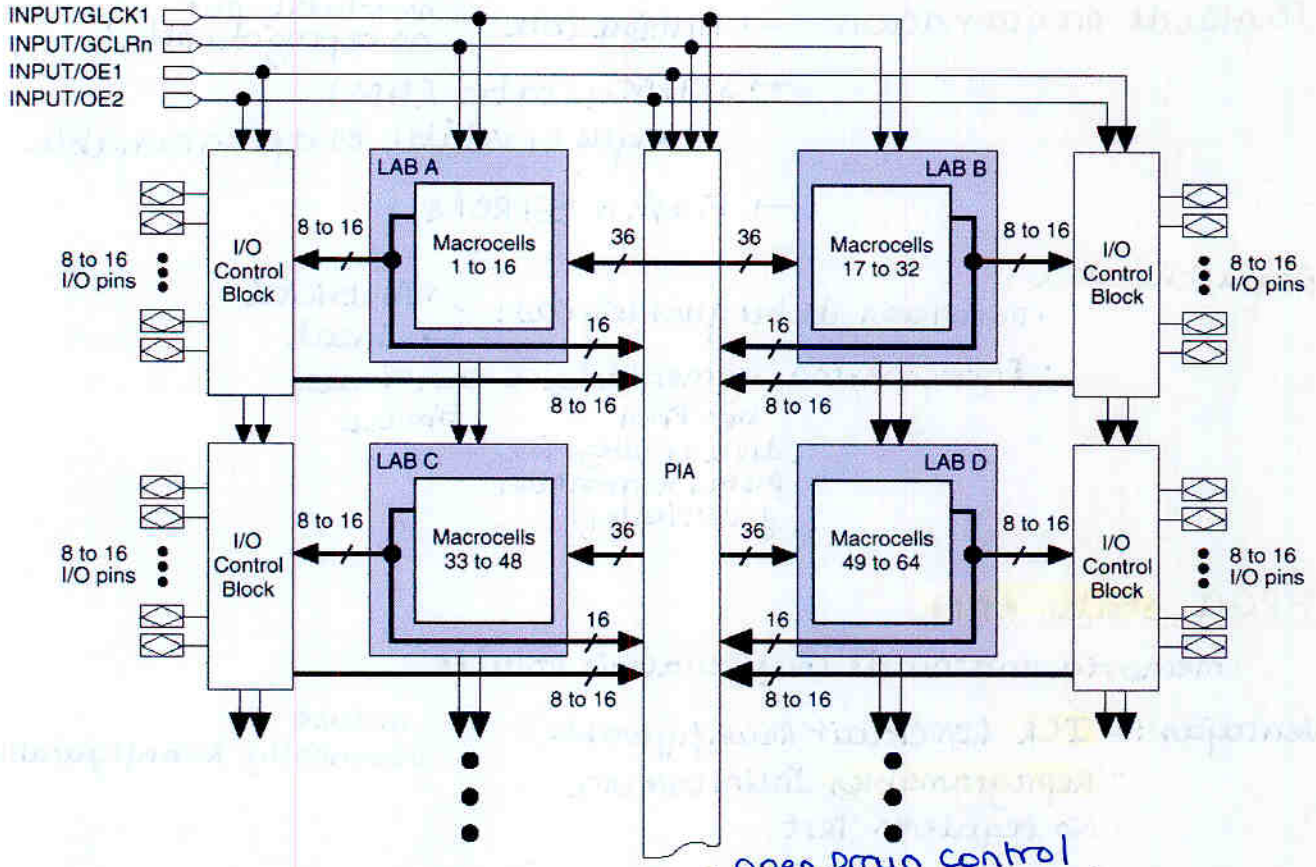
ej: MAX 7000



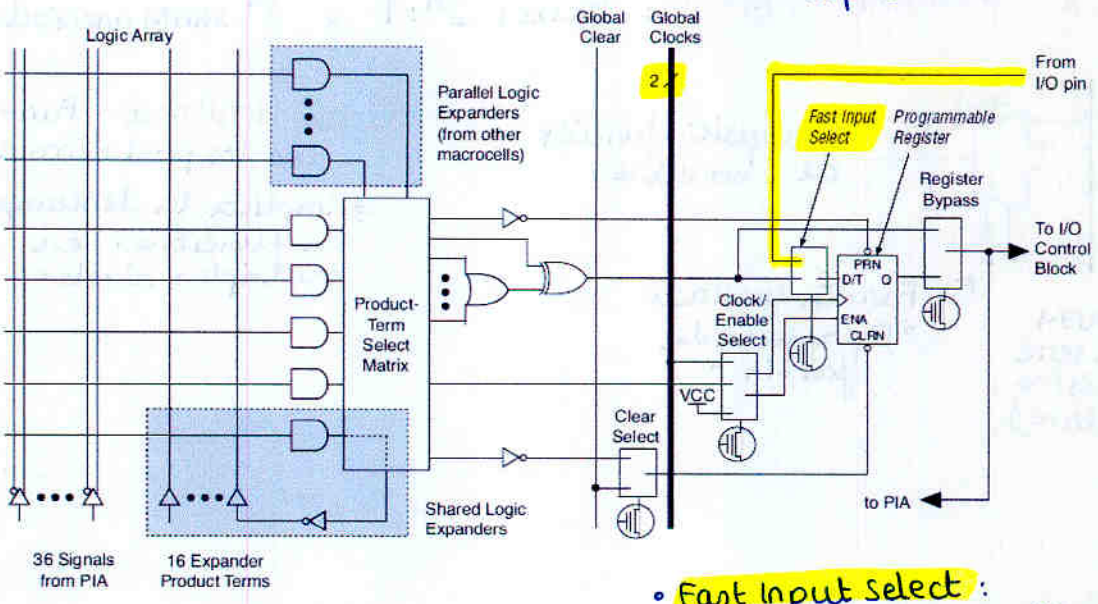
Expansores Compartidos.
De los 5 P.T. puedes usar uno compartido (si quieres) que se realimenta a matriz local

Limitado en P.T.
- 5 P.T. propios
- Hasta 32 P.T.s mediante expansores

esquema de la MAX7000

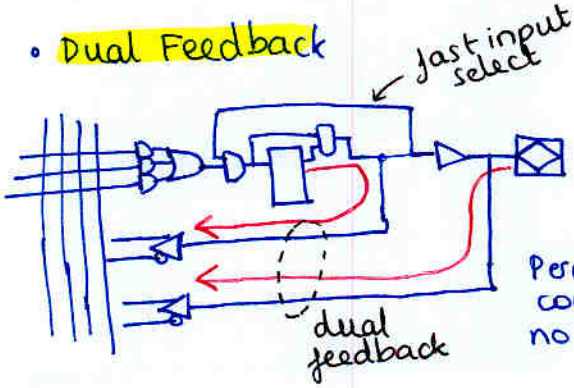


MAX 7000S: versión mejorada
- open Drain control
- MultiVolt VCCINT/VCCIO
- ISP (In System Programmable)
- Dual Feedback
- Fast Input Select



- **Fast Input Select**: Que la entrada entre directamente a la macrocélula sin pasar por la PIA

- **Dual Feedback**



Permite independizar un pin I/O (usado como entrada) de la macrocélula, para no tener que sacrificarla

FPGA's

Clasificación:

Técnica de programación:
- → Antifusible (no volatil, pero no reprogramable)
- → SRAM (Static RAM) aunque es volátil, es reprogramable
- → Flash o EEPROM

Arquitectura:
- Disposición de bloques lógicos: → Matricial / → Lineal
- Interconexión segmentada o continua
  - segmentada: tipo FPGA (diversas categorías pistas horizontales y verticales)
  - continua: tipo CLD

FPGA Static RAM

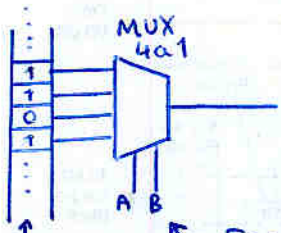
- memoria interna de configuración volatil

Ventajas:
- ICR (In Circuit Reconfigurable) — incluso Dinamically Reconfigurable
- Reprogramables Infinitamente
- No requieren Test

Inconvenientes:
- (FPGA + memoria de configuración) en la PCB
- Sin confidencialidad

Look-Up-Tables (LUT): MUXes $2^n:1$ y 2^n SRAM asociadas

ej n=2



memoria SRAM (se carga en serie como un registro desplazamiento)

Fan-In limitado "El problema del fan-in"

Cualquier función de 2 variables

Factor limitante: Fan-In (no los prod término)

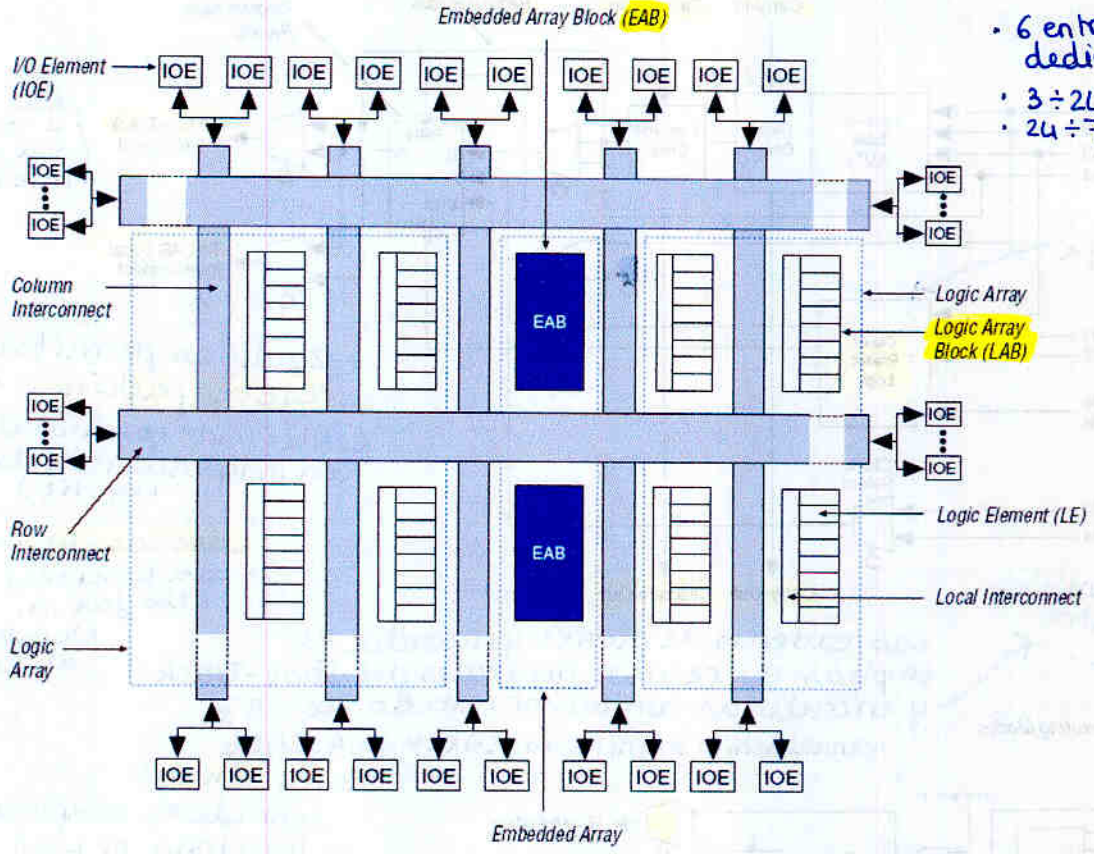
⇒ Implica la descomposición de funciones en múltiples niveles

FPGA : Familia FLEX 10K

de 576 a 9984 MC's (LE's)

- SRAM
- CMOS
- MultiVolt
- ClockLock y ClockBoost

EPF10K20RC240-4
Flex 10K · 20 000 puertas equiv. · pines · vel



- 6 entradas dedicadas/globales
- 3 ÷ 24 rows
- 24 ÷ 76 columns

Figure 5. FLEX 10K LAB → agrupación de 8 macrocélulas (logic elements = LE)

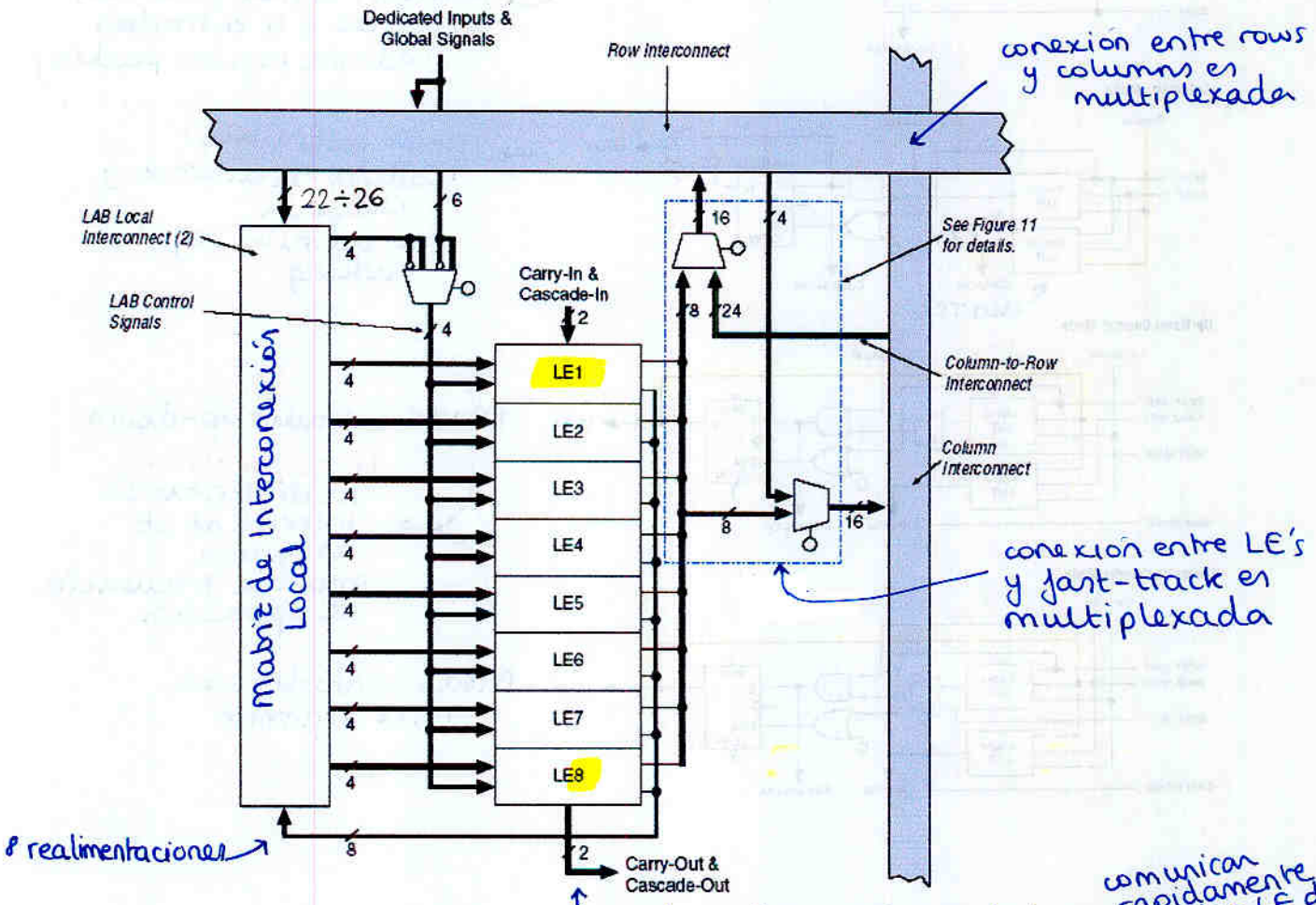
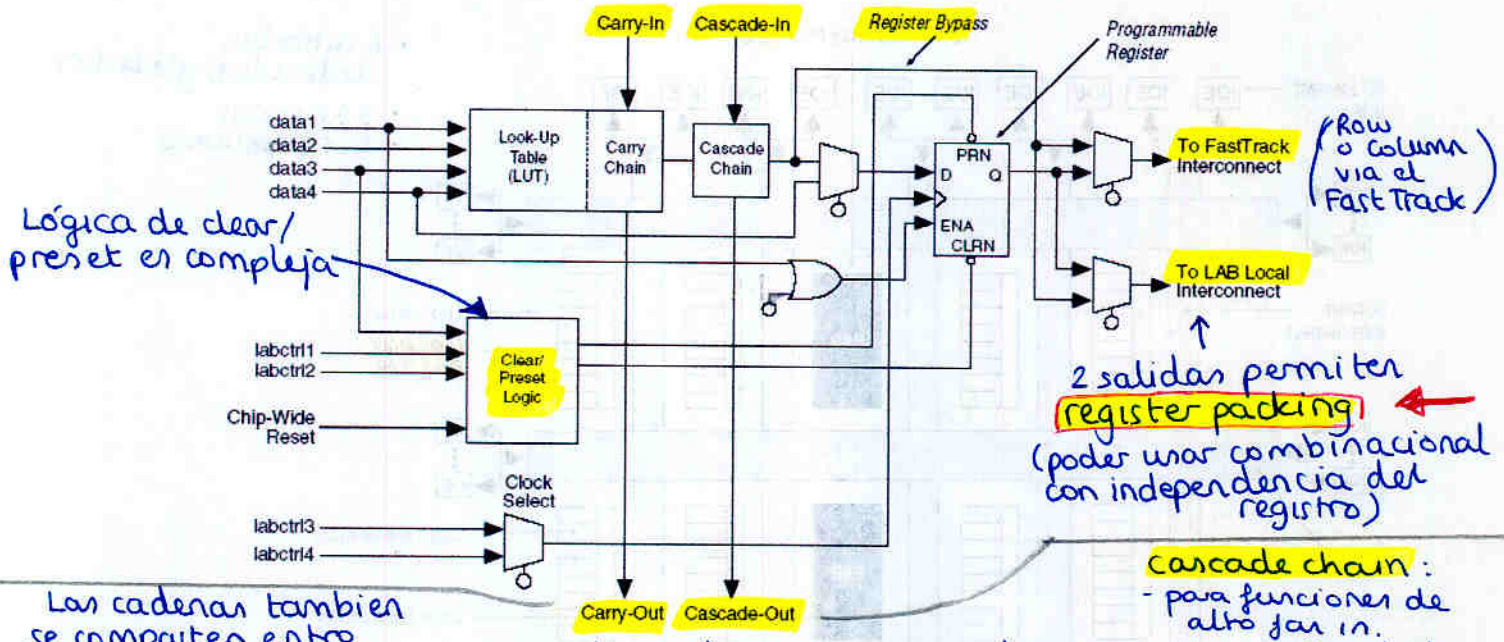


Figure 6. FLEX 10K Logic Element



Lógica de clear/ preset es compleja

(Row o Column via el FastTrack)

2 salidas permiten register packing (poder usar combinacional con independencia del registro)

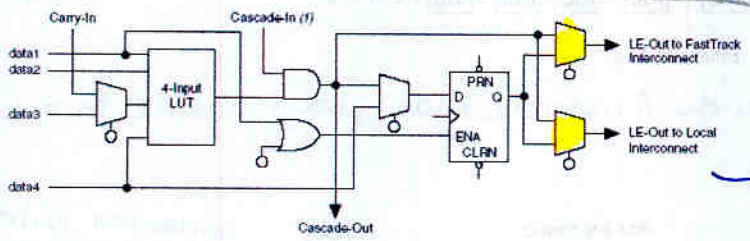
cascade chain:
- para funciones de alto fan in. No afectan al LUT

Las cadenas también se comparten entre todas las LE de LABS alternos

Las cadenas de acarreo y cascada no emplean los recursos generales del Fast-Track y encadenan con menor retardo los LE's manualmente: primitivas CARRY y CASCADE

Figure 9. FLEX 10K LE Operating Modes

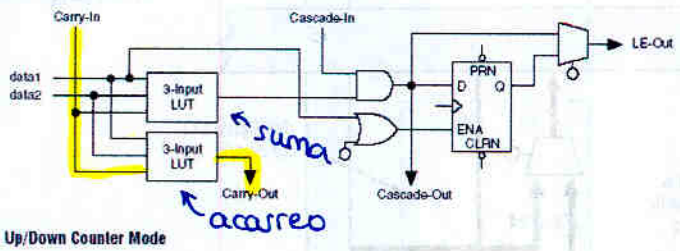
Normal Mode



Carry chain:
- contadores, comparadores
- fracciona el LUT

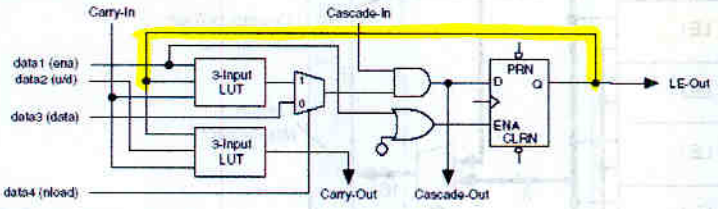
Normal: lógica genérica de ≤ 4 entradas
· admite register packing

Arithmetic Mode



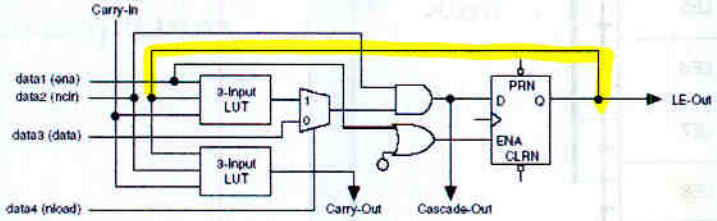
Modo aritmético:
· cadena de acarreo y cascada
· no admite register packing

Up/Down Counter Mode



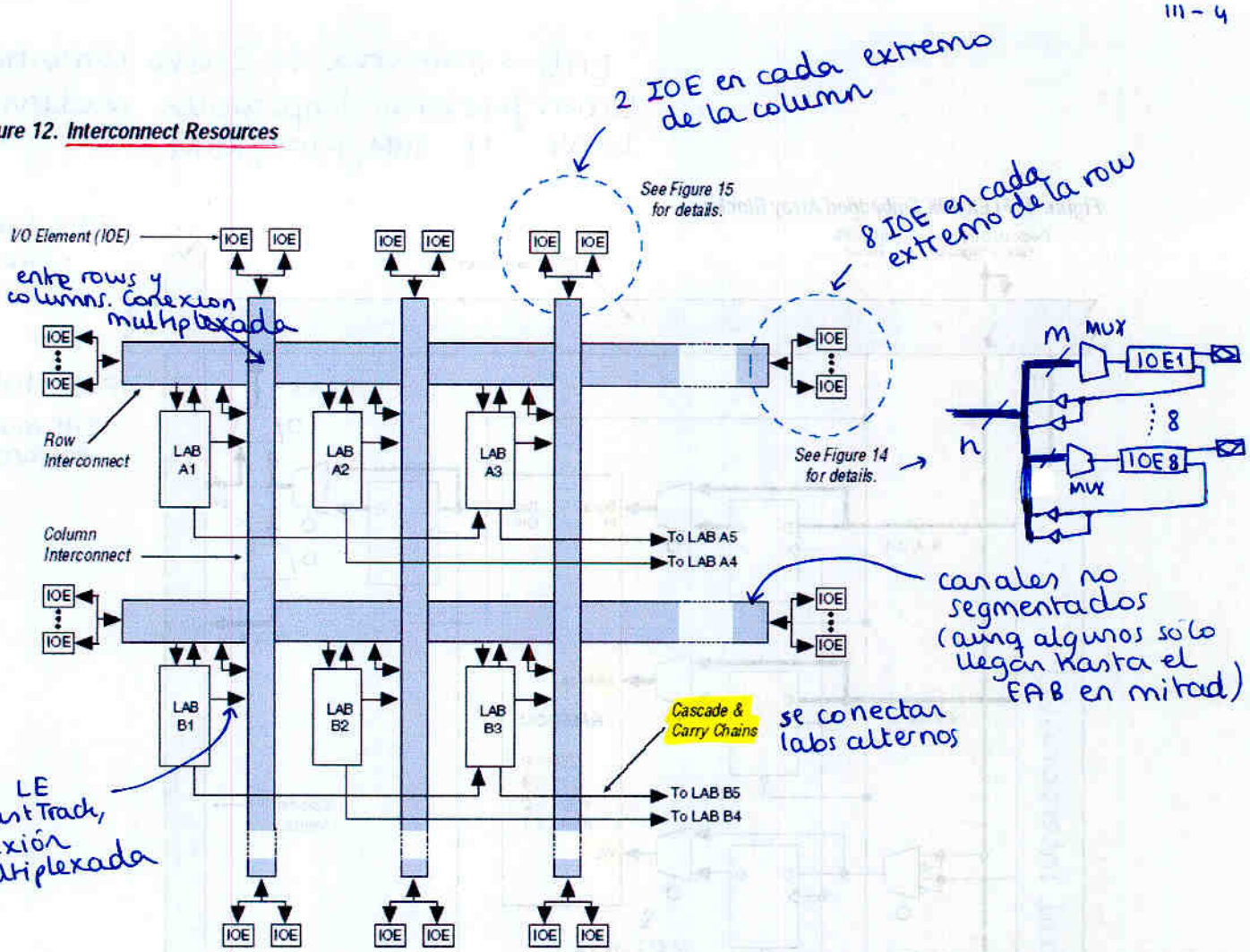
Modo contador up-down (Local feedback) Realimentación interna al LE asegura máxima frecuencia de operación

Clearable Counter Mode



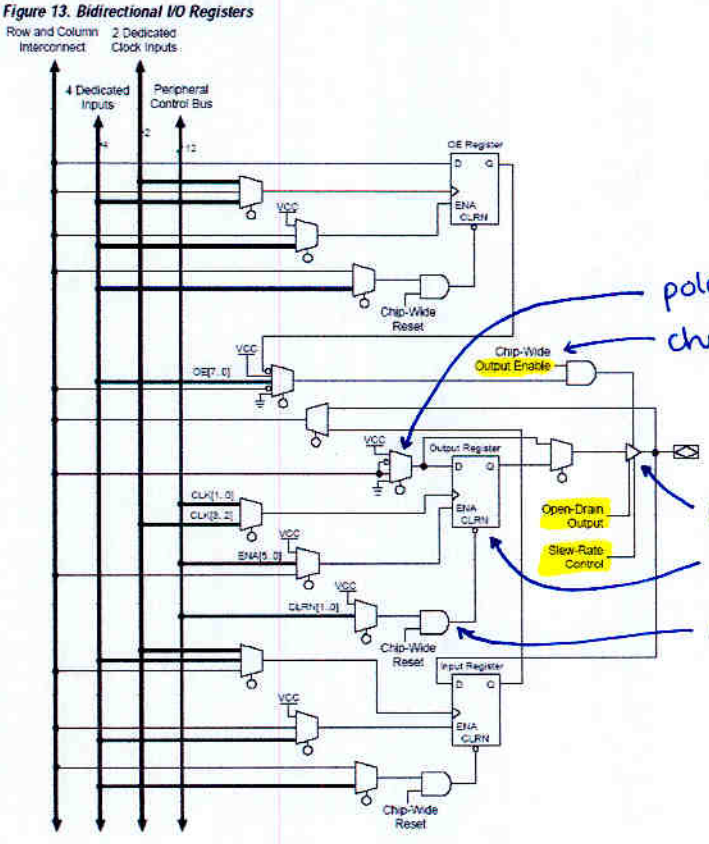
Modo contador con reset síncrono

Figure 12. Interconnect Resources



Input/Output Element

Figure 13. Bidirectional I/O Registers



polaridad programable

chip-wide output enable

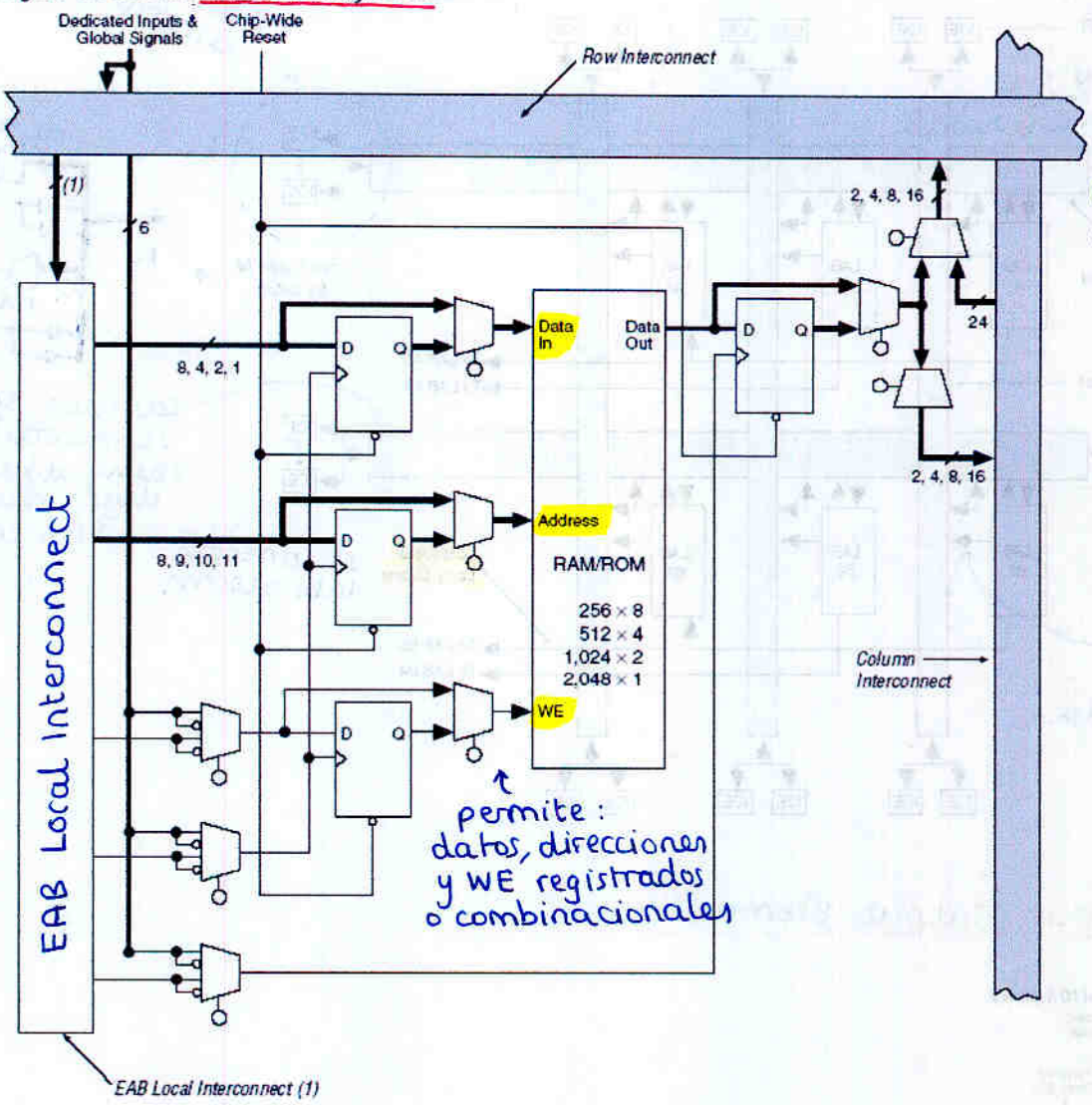
Buffer IO con open drain y slow slew-rate

FF periférico bypasseable: tsu y tco muy bajo

chip-wide reset

EAB → memoria de 2-4kb concentrada
tienen funciones disponibles mediante
LPM's ej: RAM, FIFO, ROM

Figure 4. FLEX 10K Embedded Array Block

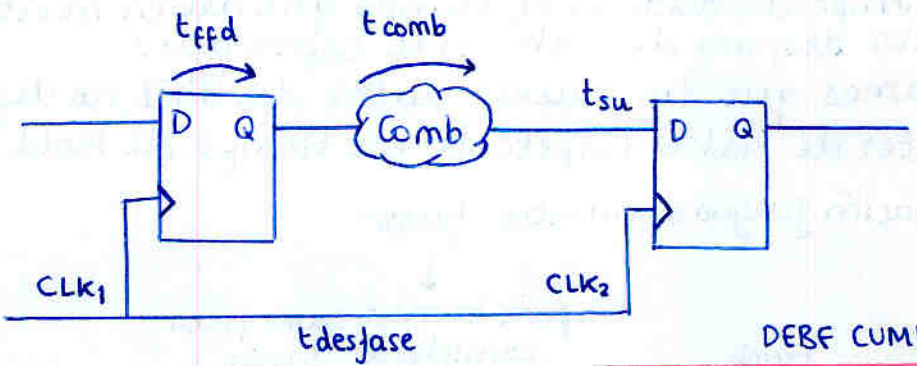


permite:
datos, direcciones
y WE registrados
o combinacionales

FLEX 10k/10kA
· 2kb

FLEX 10KE
· EAB muy mejorado

Tiempos de activación y retención — Tema 5



Margen de activación = slack

DEBE CUMPLIRSE

$$T_{clk} > t_{ffd_{max}} + t_{comb_{max}} + t_{su}$$

¿cuanto mayor?

- Sin desfase: $T_{clk} = t_{ffd} + t_{comb} + t_{su} +$ margen activación

  tiempo que le sobra al ciclo de reloj

$$\text{margen activación} = t_{clk} - t_{ffd_{max}} - t_{comb_{max}} - t_{su_{max}}$$

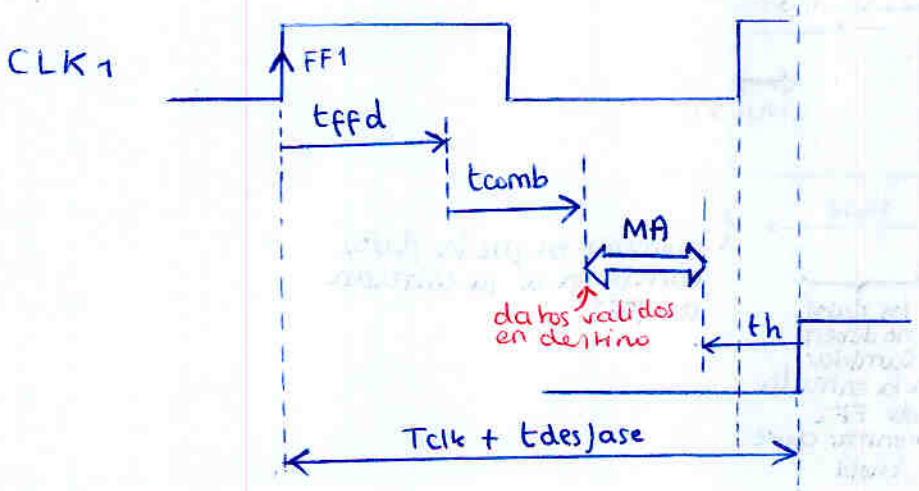
(mínimo en el caso peor) → ser pesimistas

- Con desfase: el desfase juega a favor del margen de activación, ya que 'alarga' el ciclo del reloj (salvo si el reloj llega antes al FF2 que al FF1)

  para unificar expresiones: $t_{desfase} = \text{reloj 2} - \text{reloj 1}$ $\begin{cases} >0 & \text{si } CLK_1 \text{ antes q } CLK_2 \\ <0 & \text{si } CLK_2 \text{ antes q } CLK_1 \end{cases}$

$$\text{margen activación} = t_{clk} + \underbrace{t_{desfase_{min}}}_{\text{el desfase "alarga" el ciclo de reloj}} - t_{ffd_{max}} - t_{comb_{max}} - t_{su_{max}}$$

el caso peor es $t_{desfase_{min}}$ → si el desfase es negativo tomamos lo más negativo posible lo cual juega a nuestra contra



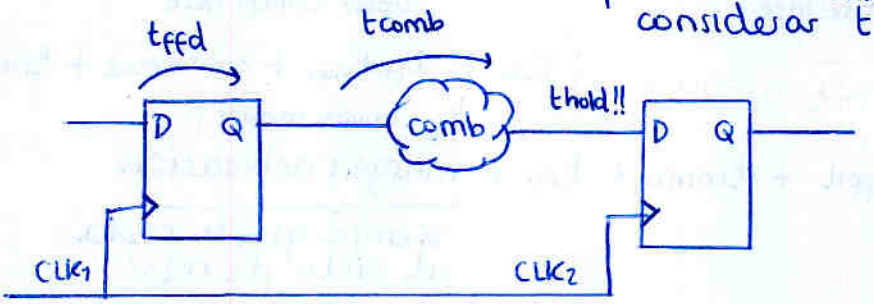
se puede obtener f_{max} haciendo MA = 0

Margen de retención

Sabemos que el tiempo de hold es el tiempo que deben mantenerse los datos estables después de haber sido capturados.
¿Cómo garantizamos que los nuevos datos del FF1 no lleguen a D del FF2 antes de haber respetado ese tiempo de hold?
Ésta vez la propagación juega a nuestro favor

↓

para tomar caso peor considerar t_{min}



DEBE CUMPLIRSE

$$t_{ffd_{min}} + t_{comb_{min}} > t_{hold}$$

¿cuánto mayor?

de hecho

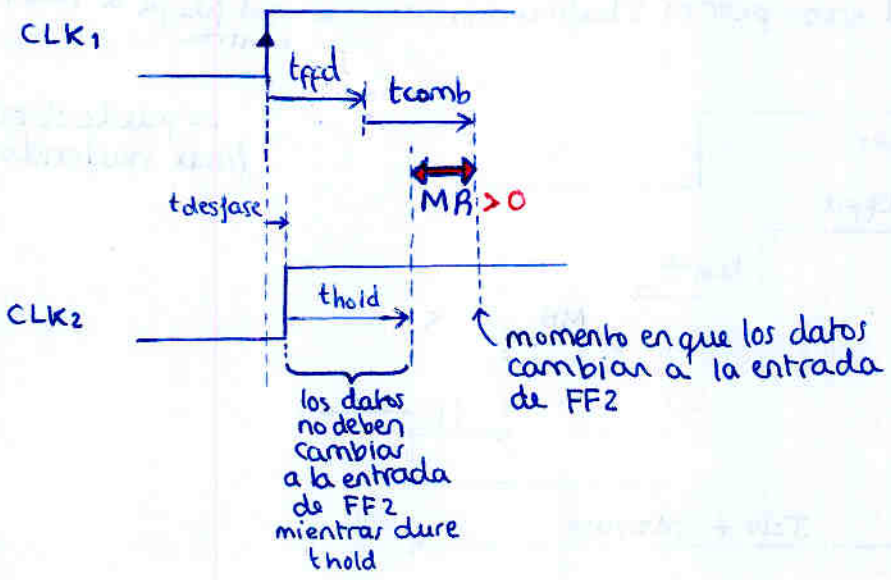
$$t_{ffd} + t_{comb} = t_{hold} + \text{margen retencion}$$

⇩

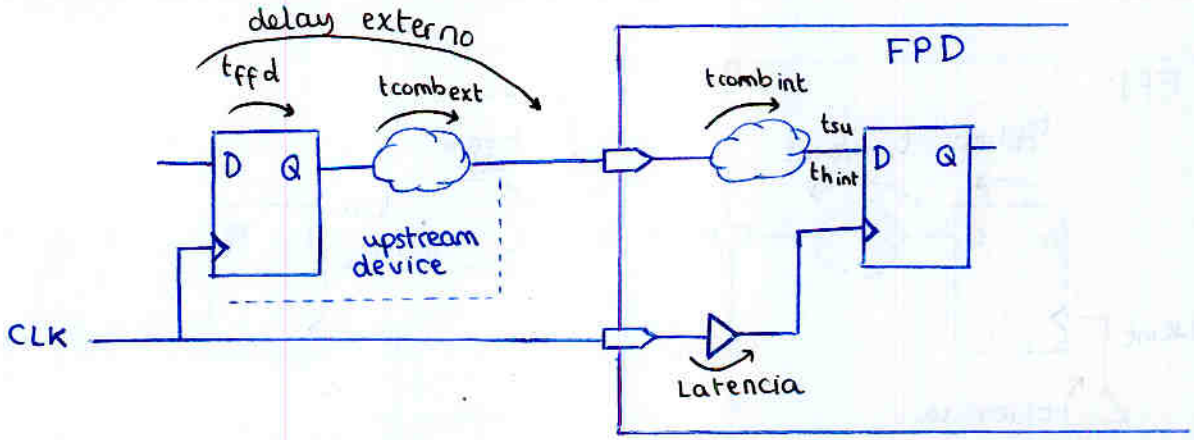
$$\text{margen retencion} = t_{ffd\,min} + t_{comb_{min}} - t_{hold}$$

Y esta vez parece obvio que un desfase de reloj positivo juega en nuestra contra, ya que es como si redujera $t_{ffd} + t_{comb}$

$$\text{margen retencion} = -t_{desfase} + t_{ffd\,min} + t_{comb\,min} - t_{hold}$$



Path registro exterior a registro interior FPD



Margen activación (slack):

$$\text{delay externo} = t_{ffd} + t_{comb\,ext}$$

$$\text{offset in} = t_{su\,int} + (t_{comb\,max\,int} - t_{latencia})$$

A veces offset in se llama $t_{su\,ext}$

$$t_{su\,ext} = t_{su\,int} + (t_{comb\,max\,int} - t_{latencia})$$

tiene lógica ya que desde el punto de vista exterior, $t_{setup\,externo}$ es justo el tiempo que deben estar los datos antes del flanco.

Debe cumplirse:

$$t_{CLK} > \text{delay externo}_{max} + \text{offset in}$$

t_b llamado ↑ input max delay

offset in = $t_{setup\,ext}$

Se calcula $t_{setup\,ext}$ para todas las entradas a la FPD que llevan a un FF

Margen retención

$$\text{delay externo} = t_{ffd} + t_{comb\,ext}$$

$$t_{h\,ext} = t_{h\,int} - (t_{comb\,min\,interno} - t_{latencia})$$

de nuevo tiene sentido pensar en thold externo, ya que es el tiempo durante el cual no debe cambiar la entrada a la FPD tras CLK externo para asegurar $t_{hold\,int}$

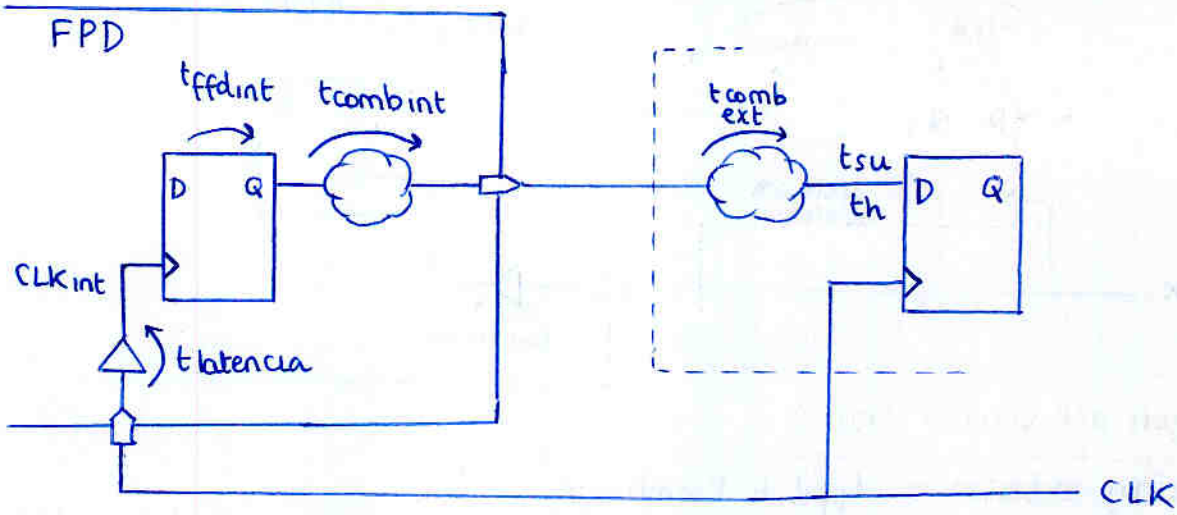
Debe cumplirse:

$$MR = \text{delay externo}_{min} - t_{h\,ext} > 0$$

t_b llamado ↑ input min delay

El análisis calcula t h-e para todas las entradas a la FPD que llevan a FF

Path registro interior FPD a registro exterior



Margen de activación:

delay externo = $t_{combext} + t_{su}$
offset out = t_{co} = $t_{ffdint} + t_{combint} + t_{latencia}$

⇔ Tiempo de propagación de las salidas registradas

Debe cumplirse:

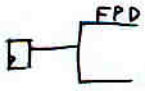
$$t_{CLK} > t_{co\,max} + \text{delay externo max}$$

Margen de retención:

Debe cumplirse:

$$MR = t_{comin} - \underbrace{(t_h - t_{combext\,min})}_{\text{output minimum delay}} > 0$$

Lo importante:



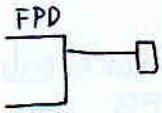
MA:

$t_{su\,ext} = t_{su\,int} + (t_{comb} - t_{latencia})$

offset-in

MR

$t_{h\,ext} = t_{h\,int} - (t_{comb} - t_{latencia})$



$t_{co} = t_{ffdint} + t_{combint} + t_{latencia}$

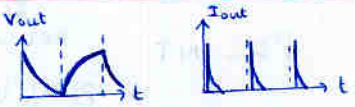
offset out

↕

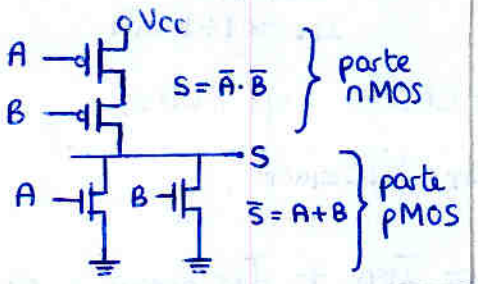
tiempo de propagación de las salidas registradas

Potencia Disipada — Tema 6

Potencia Alterna en CMOS



Recuerda:
puerta nOR CMOS



$P_{AC} = P_{ACD} + P_{ACSC}$

Potencia Dinámica: cuando pMOS o nMOS tienen que cargar/descargar C_{load}

$$P_{ACD} = C_{load} \cdot f_p \cdot V_{dd}^2$$

Potencia dinámica en cortocircuito: Imposible que nMOS y pMOS conmuten EXACTAMENTE a la vez. Hay un tiempo durante la transición en el cual ambos conducen

$$P_{ACSC} = cte \cdot f_p$$

f_p: frecuencia de conmutación (switching frequency)
Es la frecuencia de la señal a la entrada que a su salida produce dos conmutaciones. → i.e. $\frac{1}{T_p}$ → periodo que pasa a la entrada para producir 2 conmut. a salida

Potencia en un IC (CMOS)

Potencia estimada en un IC = $P_{EST} = P_{INT} + P_{IO}$
(P_{INT}: Potencia en el nucleo; P_{IO}: Potencia en periferia)

- **Potencia en el nucleo P_{INT}**

$$P_{INT} = P_{AC_{INT}} + P_{DC_{INT}}$$

$$P_{AC_{INT}} = \sum P_{ACi} \quad \text{sumatorio de las puertas del nucleo}$$

$$= \sum C_i \cdot f_{pi} \cdot V_{dd}^2$$

$$f_{pi} = \frac{1}{2} \cdot f_{CLK} \cdot tog_i$$

- T_{CLK} produce como máximo 1 conmutación
- $2T_{CLK}$ produce 2 (max) por tanto la frec de entrada que a la salida produce 2 conmutaciones es $\frac{1}{2T_{CK}} = 0'5\, f_{CLK}$
- tanto por 1 de las conmutaciones de cada registro
  - ej: bit D0 en contador bin. tiene $tog_i = 1$ ya que cambia en cada flanco
  - bit D1 en contador binario tiene $tog_i = 0'5$

$$P_{AC_{INT}} = C_{NODE} \cdot \frac{1}{2} f_{CLK} \cdot tog_{INT} \cdot V_{CC_{INT}}^2$$

- $C_{NODE} = \sum C_i = N \cdot C_{LOAD}$
- tog_{INT}: la media de tog_i del sistema. Altera sugiere 0'125 que equivale a contador binario de 16 bits

en general; contador de n bits
cada bit $tog_i = (1/2)^{i-1}$
la media $tog = \frac{2^n - 1}{n \cdot 2^{n-1}}$

$$tog = \frac{1 + 1/2 + 1/4 + \ldots}{n}$$

$P_{INT} = P_{AC_{INT}} + P_{DC_{INT}}$

$P_{DC_{INT}}$ → en continua

- para IC CMOS dual: $P_{DC_{INT}} = \underbrace{I_{DC_{INT}}}_{\text{Device leakage current}} \cdot V_{DC_{INT}} \simeq 0$

  $I_{DD} \simeq 1 \div 10\,mA$

- para IC con pseudo CMOS (ej: PLD's)

$$P_{DC_{INT}} = I_{DC_{INT}} \cdot V_{DC_{INT}}$$

$$I_{DC_{INT}} = \underbrace{I_{DD}}_{\text{leakage}} + \underbrace{I_{DC_{ARRAY}}}_{\text{NO es despreciable}} \simeq 50 \div 250\,mA$$

• Potencia en la periferia

$$P_{IO} = P_{AC_{IO}} + P_{DC_{IO}}$$

$P_{AC_{IO}}$ → despreciable en entradas pero grande en salidas con C_{out} alto o conmuten mucho

$P_{DC_{IO}}$ → en general se desprecia pero existe (pag T6-38)

$$P_{AC_O} = \sum \cdot C_{load_i} \cdot f_i \cdot V_o^2$$

f_i → frec conmutación de cada salida

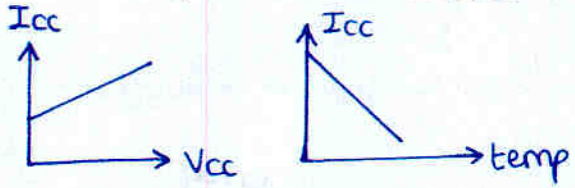
V_o^2 → $\begin{cases} V_{CC_{IO}}^2 \text{ en salida complementaria} \\ V_{OH\,min} \cdot V_{CC_{IO}} \text{ en salida totem pole} \end{cases}$

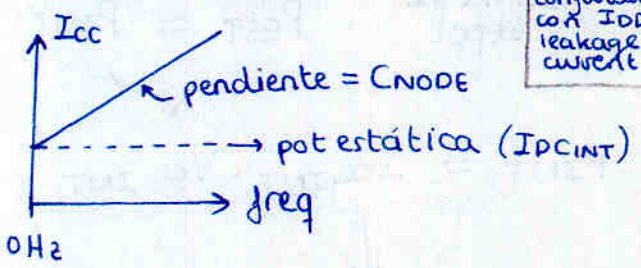
Estima de la potencia disipada en SPLD

ej GAL 16V8

se da el parámetro **Icc : Operating Power Supply Current** obtenido experimentalmente

no confundas con IDD; leakage current





$$I_{CC} = K_V \cdot K_T \cdot K_f \cdot I_{CC/typ}$$

$$P_{INT} = I_{CC} \cdot V_{CC}$$

- Con bit de turbo a OFF → puede quedarse en standby
  ej GAL 16V8Z ↑ indica bit turbo

t_{AS} : tiempo activo → standy
t_{SA} : tiempo standby → activo
t_{PD} : tiempo propagación modo activo
t_{SR} : tiempo entre transiciones a las entradas

si $t_{SR} > t_{PD} + t_{SA}$ le da tiempo a 'dormirse' en cada transición

(i) cuidado : bit de turbo a OFF no basta para ahorrar potencia ya que si $t_{SR} < t_{AS}$, no se entra en standby

$$I_{CC_{AVG}} = I_{CC} \cdot \frac{t_{AS} + t_{PD}}{t_{SR}} + I_{SB} \frac{t_{SR} - (t_{AS} + t_{PD})}{t_{SR}}$$

↑ corriente en standby ≃ 50μA despreciable

Recuerda:

Una vez calculada P_{EST}

$$P_{MAX} = \frac{T_{JMAX} - T_{AMAX}}{\theta_{JA}} > P_{EST}$$

↑ si no se cumple, se quema

nota:
uso comercial
$T_{Amax} = 70°C$

Estima de la potencia disipada en CPLD's ej MAX 7000

Estima de la pot $P_{EST} = P_{INT} + P_{IO}$

$P_{INT} = I_{CC\,INT} \cdot V_{CC\,INT}$

$I_{CC\,INT} = A \cdot MC_{T_{ON}} + B\,(MC_{DEV} - MC_{T_{ON}}) + C \cdot f_{max} \cdot tog_{LC} \cdot MC_{USED}$

- A: [mA/MC]
- $MC_{T_{ON}}$: Macro Células con turbo ON
- B: [mA/MC]
- $(MC_{DEV} - MC_{T_{ON}})$: MC's con turbo off (i.e. duermen)
- C: [mA/MHz/MC]
- f_{max}: del reloj
- MC_{USED}: MC's usadas

Estática: Incluye todas las MC del Device, por el mero hecho de ser alimentadas

Dinámica

$P_{EST} = P_{INT} + P_{IO}$

$P_{In\,out} \simeq P_{out} = P_{ACout} + P_{DCout}$

(P_{IO})

$P_{ACout} = \sum_{i}^{N} C_i \cdot f_i \cdot V_i \cdot V_{CCIO}$

$P_{ACout} = N \cdot C_{LOAD} \cdot \frac{1}{2} f_{max} \cdot tog_{IO} \cdot V_O \cdot V_{CCIO}$

$3'8\,V \leftrightarrow 5\,V$
$3'5\,V \leftrightarrow 3'3\,V$
$2'5\,V \leftrightarrow 2'5\,V$

$P_{DCout} = \sum_i P_{DC_i}$

Estima de la potencia disipada en FPGA's

$P_{EST} = P_{INT} + P_{IO}$

$P_{INT} = (I_{CCACTIVE} + I_{CCSTANDBY}) \cdot V_{CC\,INT}$

(dinámica + estática)

$I_{CCACTIVE} = K \cdot N \cdot f_{max} \cdot tog_{LC}$

- K: cte experimental (es como la C)
- N: LE usados

I_{CCO} → dato catálogo (igual q I_{CC} en SPLD)

nota: las LE (MC) no usadas no consumen potencia estática

LAB. DCSE - Referencia Rápida

Sesión 1

Max Plus II Manager → maneja sólo UN proyecto

File > Project > Name

Options > License Setup

System Info → Nº serie del disco duro

Esquemático:

File > New > Graphic Editor File

File > Saveas > accumul

Símbolos
mega_lpm → símbolos parametrizables
mf → macrofunciones (old-style)

Ayuda:
- VHDL
Help > VHDL → tiene ejemplos para todo

- Símbolos/LPM
Help > megafunctions
Son librerías de símbolos parametrizables
↳ lpm_counter
↳ lpm_add_sub

lpm_add_sub

Donde lista los puertos leer bien comments q hacen referencia a parámetros

ej: input add_sub no se utiliza si se usa el parámetro LPM_DIRECTION (add o sub para siempre)

Bot derecho > Enter Symbol > (F1) mega_lpm > add_sub (doble clic)

EDIT PORTS, PARAMETERS
sin Cin
sin Cout
con Overflow (genera un uno cuando halla overflow a no ser que pongamos inversión i.e. (lo ponemos a ALL))
LPM_DIRECTION: ADD
LPM_REPRESENTATION: UNSIGNED
LPM_WIDTH: 4
OK.

overflow y carryoutput son iguales si trabajamos sin signo

Nota:
latencia de un circuito: nº de ciclos hasta obtener el resultado.
ej sumador con registros en cada paso 'entubados' (pipeline) pueden ir realizando muchos cálculos a la vez, aunque tengan alta latencia (tanta como registros tenga) a cada golpe de reloj saca un resultado (ej. cadena de montaje)

Los pines no habilitados no tienen acceso al exterior. Los pines invertidos tienen 'bolita' junto a un número (cuidado algunos puertos tienen bolita para hacer bonito)

Librería Prim lógica combinacional.
Bot derecho > Enter Symbol > prim
ej: and, or, not, flip flops ej DFF

Recomendación: para introducir símbolos
bot derecho > Enter symbol > Escribir nombre directamente
ej: and, or, input, output, vcc, ...

VCC y GND son niveles lógicos y no nudos / etiquetas

tri: buffer triestado
bidir: puerto bidireccional.
Es un problema, debe conectarse a un tri
si trihabilitado → salida
si no → entrada
el diseñador deberá poner 0's y 1's cuando sea entrada y Z cuando sea salida
↘ es importante hacer un dibujo bonito

Etiquetar Pines
Bot derecho > Edit name ej: AccIn [3..0] (un bus)
al darle a ENTER para al siguiente

Pin Default Value: al no conectar pin ¿que valor tiene?

Cables:
Nota: util poner rubberbinding ON → cuidado; eliminar colgajos
Permite conectar simplemente acercando componentes
Se pueden dibujar cables directamente con
Bot derecho > Enter Node Name
Clic: selecciona sección >> otro clic → selecciona el nodo

→ si queremos un bus hacerla línea gorda
→ cuando el texto toca el cable, lo nombra

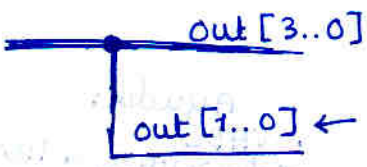
Etiquetado:

- son de ámbito local
- Los puestos tienen prioridad sobre los cables (i.e. el puesto da nombre al cable)
- Un nudo sólo puede tener un nombre
- Para hacer subBus

(ver libro ejercicios pag252 → g1[3..0], g2[3..0] → WIRE data[0][3..0], data[1][3..0] → data[1..0][3..0] ← vector de buses !!)

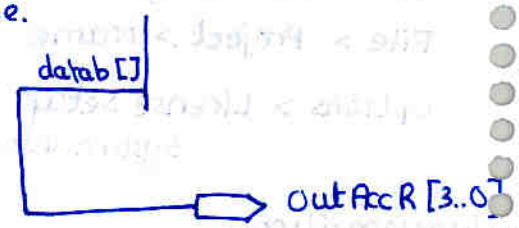
solución: primitiva WIRE

nombreA WIRE nombreB



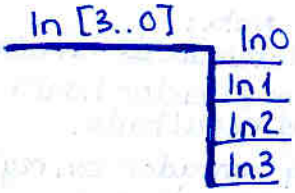
← este texto habría que arrastrarlo manualmente

i.e.



Ver: Help > Pin and node names
para ver todo lo de los buses (ver Example's, haciendo dic sale dibujito)
ver tb Help > Bus names

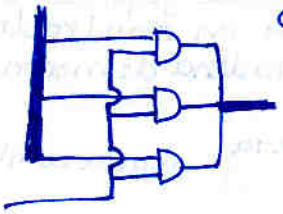
- Los buses se conectan por nombre
  SIEMPRE etiquetarlos



Nota: no hace falta etiquetar buses enterrados (no salen a un puesto exterior) si conectan pines de la misma anchura.

- cuando conectamos un bus de 4 bits a una puerta AND se convierte en un Primitive Array i.e. (se comporta como varias puertas en paralelo)

Esto es MUY util



cuidado: si una entrada es bus (i.e. hacemos un primitive array) la salida debe ser bus.

Para comprobar:

File > Project > Save and Check

Compile (se puede hacer dándole al botón de la fabriquita

Para que concuerde con ubs de prácticas (q. utiliza otra versión del compilador)

assign > device > MAX7000S (desmarcar show only Fastest...) > EPM7128SLC84-7

y volver a compilar con el nuevo dispositivo

Compilación:

Logic Synthesizer: da warning y elimina cosas innecesarias en caso de encontrarlas
Timing SNF: modelo lógico-temporal que hay que estimular para la simulación

Simulación:

(tras compilar)

File > New > Waveform Editor file

Bot Derecho > Enter nodes from SNF : elegir señales que queremos ver
* INTRO → ver todas

organizas señales arrastrando el iconito

Elegir eje x:

File > End Time ej: 4us (sin espacios)

View > Fit in Window

Estimular el circuito: un arte

Options > Grid Size ej: 50.0ns → 10 MHz

- Bot Derecho en Value de CLK > Overwrite > CLOCK

  Dibujar siempre el reloj pegado a la rejilla (i.e. multiplicar periodo x 1 siempre adaptar la rejilla

- los reset: Value > 1 que si se nos olvida luego no va
  activos a nivel bajo

- AccIn : Value > Overwrite > Count Value

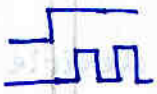
  Count Every: que cambien a golpe de reloj
  i.e. ajustar multiply by

  Asegurarse que los datos sean estables en el flanco de subida del reloj (i.e. al capturarlo)

- Hacer reset al principio de la estimulación (cargarse el reloj si es asíncrono para no liarnos)?

  Bot derecho > overwrite

  ej: RESET asíncrono
  CLK



- Para editar visualización de un bus

  Boton Derecho > Enter Group → eliges decimal, hex, ...
  > Ungroup → se ve cada señal del bus

  Para cambiar las señales en bus > boton derecho > Insert > Group Value

- Boton Timing Simulation

- MaxPlusII > Simulator

  Botón START

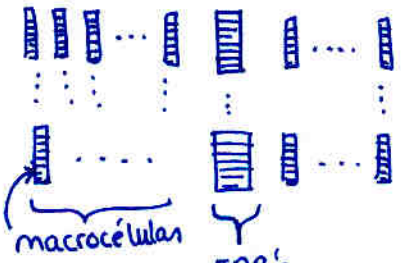
  Asegurarse de que end time sea correcto.

El archivo .scf se puede reutilizar en otros circuitos si se mantienen los nombres de entradas/salidas

~~Sesión 12. Tecnología~~

EABs Embedded Array Block

Floorplan de una flex 10k



Hay 6
EAB_A
EAB_B
⋮
EAB_F

Son como memorias ROM de 2k bits cada una que pueden tener salidas registradas, aunque no permiten realimentaciones

ej 1k palabras de 2 bits

son muy utiles para almacenar lo de una LPM_ROM

Para forzar un bloque a implementarse sobre una EAB

Boton Derecho > Assign > Logic Options > Individual Logic Options > Implement in EAB

o para decirle en cual

Tras haber elegido dispositivo:

Assign > Pin Location Chip > EAB

Sesión 2

- .acf → asignaciones y configuración
- .gdf → esquemáticos
- .scf → simulación

Al empezar nuevo proyecto

1. Set Project Name to current file
2. Assign Device EPM 7128SLC84-7 (MAX 7000 S)

Personalizar LPM de forma fácil

File > MegaWizard > Elegir LPM decir fichero destino ⇒ Nos genera descripción estructural en VHDL y genera símbolo .sym

Para introducir símbolo: botón derecho > Enter Symbol

Comparar simulaciones:

☑ Check output en la ventana de simular
Y comprueba si coincide lo que tu hayas puesto en el .scf que será lo que esperas que se obtenga

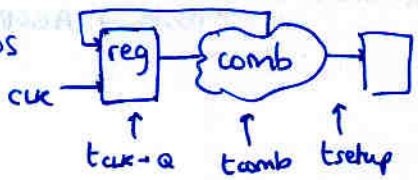
Análisis de Tiempos — Timing Analizer

- Para documentar 'temporalmente' el C.I. que uno ha creado para que otros sepan usarlo
- Salidas combinacionales: dependen sólo de entradas

calculamos tiempo de propagación de datos
hay mínimo y máximo, con incertidumbre intermedia

- Salidas registradas: A la salida de un registro desde el CLK hasta la salida (TCO) (no importan entradas)

- Tiempo de setup y de hold de las entradas respecto al flanco de reloj (se calcula diferencia de retardos desde el reloj y desde las entradas)
- Transferencia de información entre registros



el más lento se llama PATH CRITICO
↓
es el que intentaremos mejorar para que fmax sea mayor

Sistema secuencial: Hay registros de estados (i.e. realimentación entre registros)
no valen registros de salida

Se puede ver el Time Analysis de una única salida haciendo boton derecho en la salida

Para hacerlo desde nodos → ver detrás

Timing Analizer
options > Time Restrictions → show Only Longest Path
options > Autorecalculate (chorada)
> Cell width (cambiar tamaño celda ej 14)
> Cut-off .... → Para no analizar ciertas cosas que no suelen interesar
ej I/O Pin → True DPRAM → utiliza entrar y salir en mismo pin. muy "dificil de analizar

Retardos

start:

si las salidas son registradas miras la fila del CLK

Para hacerlo de nudos internos
Timing Analizer > Node > Timing Analysis Destination
como los nombres no se respetan, poner asterisco delante y detras y darle a LIST (checkeando ALL y show All Node synonims)

Para volver a analizar 'por dejecto'
Utilities > clear all timing analysis tags

Setup y Hold

Analysis > setup/Hold

- Vemos que el FF de más peso es normalmente el que mas tsu necesita

los nombres: ej ~3~4
↑ FF3 el de mas peso (el cuarto)
(que tiene un bus de entrada de 4 → i.e. FF3 son en realidad 4)

Frecuencia máxima
Analysis > Registered Performance

---

s20:
evcell.gdf : compara AN con BN
si GNA = 1 → B > A → AGEB = 0
→ resto → AGEB = 1
↑
A greater equal B → AGEB de los bits anteriores (menor peso)
si GNA = 0 → A > B → AGEB = 1
→ resto → AGEB = 0

Simular Directamente sin retardos } Nota simulador Processing > Functional SNF Extractor Para simular tal cual sin tiempos (no importa el assign y no hay retardos)

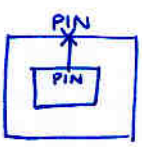
Símbolos: Jerarquía

.gdf → File > Create Default Symbol → crea .sym

Open > .sym ⇒ Editamos el símbolo

Edición de símbolo (.sym)

- Poner salidas y entradas para que quede bonito
- Unir pines interiores y exteriores con líneas
  Bot derecho en pin para editar el nombre que se verá
- File > Save



Doble clic para nuevos pines

Añadir Pin → Doble Clic

ejemplo: accumul.sym

- Está bastante arregladito
- Puertos > Difiere Full Pinstub Name / Visible Pinstub Name] tenemos la libertad de no poner nºs entre
  ↳ Fuente: Altera 7

Parámetros
Element > Enter Parameters ⇒ doble clic en la X para poner [n..0]

salida[3..0] ⟷ Salida[3..0]

Recuerda que en un símbolo, los pines no conectados tomarán su valor por defecto (default pin value)

Nota: mf → macrofunctions
las combinacionales funcionan bien
las secuenciales son chapuza → RESET, SET y CLK pasan por combinacional → son demasiado importantes para ello.

Editar símbolos / Guardar cambios

- Si se edita el archivo .sym (por ej añadiendo puertos) las instancias en los esquemáticos no cambian → Puede ser útil
  Para que cambien: Symbol > Update Symbol

- Si editamos el esquemático que creó un símbolo, se modifica el símbolo en cuanto guardemos el esquemático, incluso las instancias de éste símbolo que aparezcan en esquemáticos

- si editamos símbolo con .vhd, usar FILE > Create default symbol para actualizar el .sym

Prefijos jerárquicos ↔ sinonyms

Para referirte a algún nudo del interior de un símbolo, al nombre de este nudo se le añade un prefijo para que cuando en la compilación se 'aplaste' la jerarquía se distingan uno de otro.

Por eso para añadir nudo en la simulación hay que tickear "show All node name synonyms" y por eso hay q añadir '*' delante para buscar

compiler
Processing > Design Doctor
> Design Doctor Settings

Design Rules:
más restrictivas según bajamos

F1 → Ayuda online
↳ Project Reliability Checklist ⇒ muy importante sobre todo Clock Configuration
ej: CLK, CLEAR y RESET no admiten combinacional
↓
solución: Enable del reloj

En la asignatura sólo se admite Global Clock

Cuidado: que esté activo JUSTO al llegar el flanco
i.e. si queremos activarlo cuando un contador anterior llegue a 7, activar el enable cuando llegue a 6 para que al llegar el 7 YA ESTÉ a 1

Design Doctor Warnings se deben a los glitches

Help on Message
↓
Action:

Es posible evitar glitch con lógica que cubra ese glitch.
Esa lógica la quitará el optimizador de lógica a menos que le digamos lo contrario

Explicación glitch → ver glitch.gdf

Informe:
Compilar: doble clic rpt en fitter

El informe es MUY importante

- Auto Global signals: se ha dado cuenta de que merecen recursos especiales
- Logic for device: se ve el dispositivo
  - las patillas con almohadilla: las del bus JTAG — precio a pagar por programarlo directo en la placa
  - se ve device options → ej turbo bit, security bit, user code
    - ↳ se puede leer aún con el bit seguridad
    - Es para identificación
    - multivolt
      - ↳ si está ON es una 5V con el núcleo 3'3V a la periferia

- Resource Usage
  - Macrocélulas usadas
  - Total number flip flops
- Inputs } Pines utilizados por cada macrocélula
- Outputs }
- Equations
  - # or
  - ! not
  - & and
  - $ xor

  Utiles para saber si salidas son combinacionales o secuenciales

LCELL → célula: salida combinacional
DFFE → viene de flip-flop tipo D

ASIGNACIONES

El software ha hecho lo que ha querido con mis señales
¿NO puedo yo hacer lo que crea a nivel de chip?

Device → elejimos dispositivo → si ponemos AUTO y le damos una lista en el boton 'Auto Device' El software escoge el menor de todos en el que cabe el diseño
el nuestro
EPM7128SLC84-7

→ More options: opciones generales ej: bit Turbo
user code

Assign > Global Project Device Options
Puedes reservarle pins y células para poder ampliarlo

Elegir Pin de cada entrada: Asignaciones Rígidas
Boton derecho al pin > Assign > Pin Location Chip

NOTA: Si es un bus, hay q hacerlo individual

Asignaciones Flexibles
Recomendaciones
Bot derecho > Assign > Clique
Darle un nombre al grupo.
Se intentará que esa lógica esté juntita

También puedes agruparlos en un mismo LAB.
Hay 8 LABs
cada LAB → 16 macroceldas

Parámetros

Assign > Global Project Parameters → Escribir nombre de algun parámetro ej: macrofuncion y darle a ADD

Se usa mucho en VHDL
la mejor forma de poner parámetros

Cambiar Nombre a Nodo
Assign > Probe > Search
Para poder verlos mejor en el simulador

Al compilar, cualquier exigencia que el compilador no pueda, pregunta.

Nota: Como saber que pins utilizar
Help > Devices and Adapters > .. EPM7128 ...
Tabla muy importante

pad limited / core limited

El dado es el mismo pero los encapsulados tienen mas o menos pines (el nuestro es el JLEAD)

Columna LCell → nº de macrocelula → ver si tiene salida al exterior y en que pin.
Según encapsulado hay más o menos accesos a macrocélulas

Floor Plan

Compilar: Bot derecho en algo del esquemático > Find node in floor plan
Vemos cada macrocélula

Ver asignaciones
Layout > Last compilation → ver lo que ha hecho la compilación
Layout > Current Assignments → para poder cambiarlo

Podemos asignar pines manualmente con drag & drop → Para 'aceptar las' hay que compilar

↳ Back-Annotation:
compilar y luego ir haciendo las asignaciones convenientes
- Assign > Back-Annotate Project → te lleva las asignaciones de la compilación a la representación

a efectos prácticos es como un UNDO para volver a la

A la izquierda tenemos 4 entradas 'especiales' i.e. CLK
Podemos dar importancia

Nota: si juntamos dos pines, para separarlos, arrastrarlos hasta 'unassigned nodes & pins' arriba en blanquito

Assign > back-annotate project → coloca las señales donde la última compilación

Sintetizador de Funciones

Assign > Global Project Logic Synthesis

- Define system style > Podemos ver opciones de cada estilo
  ej: normal > minimization > Full → no sirve para eliminar glitches
  ej: WYSIWYG > minimization > Parcial → sólo elimina miniterminos REPETIDOS

Bot derecho componentes esquemática > Assign > Logic options > Style → añadir lógica redundante para eliminar glitches

- Optimize Area ☐☐☐ Speed
- Automatic Global

Eliminar glitches:
Ir por cada bloque > ver esquemático > assign to current project > compiler > doctor > compilar y ver warnings
solucionar las warnings

Agrupar lógica combinacional: símbolo LCELL
Nota: Símbolo LCELL es como si fuera un paréntesis, → Normalmente se ahorra espacio
indica que ese nodo sea salida de macrocélula
Permite que la jerarquía del esquemático en bloques se repete al meterlo en macrocélulas.
i.e. Un bloque utiliza tantas macrocélulas como necesite

III - 2

Ver lc1.gdf → lc2.gdf
→ lc3.gdf

Nota: La salida de un registro siempre es salida de macrocélula → no poner LCELL
Poner LCELL en combinacionales que ataquen a otros combinacionales

Mejorar frec maxima

Al hacer esto suele ocurrir que el retardo es mayor (más macrocelulas en cascada)

¿Que puedo hacer?

Assign > Global Timing Requirements → pedirle frec minima
o individualmente en esquemático
Bot dcho > Assign > Timing Requirements

Hay que darle libertad

ej: • Assign > Global Project Logic synthesis > Multinivel synthesis
• Compilador → Procesing > Total recompile → desde el principio

ej • Assign > Ignore Assignments

Para ver frecuencia
Max Plus - Tyming Analizer

↓
si usamos Smart Assignement
↓
Aprovecha los buenos resultados de antes
ej. decir que optimice puertas pero seguirá sin glitches porq vió q no molestaban los miniterminos redundantes
↓
la compilación es un proceso 'recursivo'

Configurar la FPGA

· Device > Elegir Device
· Compilar
· Doble clic en pof (bajo Assembler)
· JTAG > Multi-Device JTAG chain ok
> " " " " Setup

Select Programming File > Seleccionar SRAM object files
> detcnthw.sof

Add
Detect JTAG Chain Info
Ok
Configure

Sesión 4
Diseño de Subsistemas Combinacionales con VHDL

Recuerda VHDL → No es programar, es DISEÑAR

Combinacional → no reloj
↳ como lista de sensibilidad: TODAS LAS ENTRADAS

Nota: en cada rama del IF hay que asignar TODAS las salidas (sino, tendrá el valor anterior, i.e. pondrá FF)

Ficheros VHDL:

- extensión .vhd
- mismo nombre que entidad (no más de 8 caracteres)
- al compilar crea un .sym. Para actualizarlo File > Create Default Symbol
- al insertar símbolo en .gdf ↳ NOMBRE distinto

LPM_ROM

Bloques EAB (embedded array block) son memorias configurables (usar como RAM, ROM, doble puerto, ...) en las FPGA

Para trabajar con ellos se usan las LPM's:

- instanciar LPM_ROM → megasímbolo → configurable
  Doble clic
  Edit Ports / Parameters:
  - Ports
  - Parameters
    LPM_WITAD: nº líneas direcciones
    LPM_WIDTH: nº líneas datos
    LPM_NUMWORDS: tamaño de la memoria
    LPM_FILE: fichero con el contenido inicial → fichero .mif (memory initialization file)
    LPM_ADDRESS_CONTROL: → registered / → unregistered

Utiliza placa

- Diseñar
- Assign | Device EPF10K20RC240-4
- Assign | Pin Location Chip
- Recompilar el diseño
- MAX Plus II | Programmer
  JTAG | Multi Device JTAG Chain setup
  Fichero .sof
- Configure

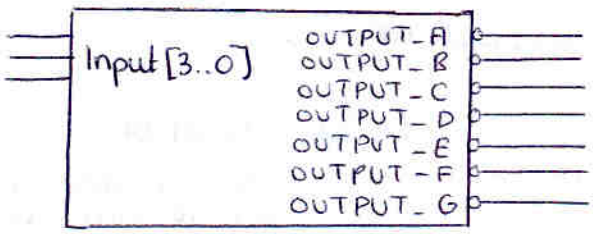
Poner parámetros en VHDL

```
entity
  generic (constant fin_cuenta : integer := 50);   -- por defecto
ports
```

además podemos hacer

```
architecture
  constant mitad_fin_cuenta : integer := fin_cuenta/2;
begin
```

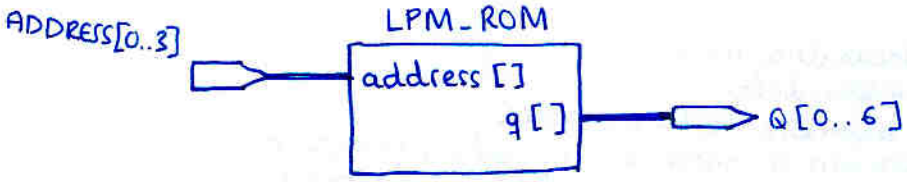
Decodificador hexadecimal a 7 segmentos



Interruptor	Pin
I1	48
I2	50
I3	53
I4	55

segmento	Pin
a	17
b	18
c	19
d	20
e	21
f	23
g	24

LPM_ADDRESS_CONTROL = "UNREGISTERED"
LPM_FILE= "w:\dsce\clase\inimem.mif"
LPM_NUMWORDS =
LPM_OUTDATA="UNREGISTERED"
LPM_WIDTH = 7
LPM_WIDHAD= 4

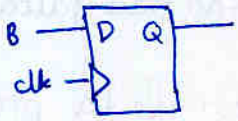



```
DEPTH = 16;
WIDTH = 7;

ADDRESS_RADIX = HEX;
DATA_RADIX = BIN;

CONTENT
   BEGIN
      0 : 0000001;
      1 : 1001111;
        :

      F : 0111000;
END;
```

Procesos secuenciales:

```
process (clk)
begin
  if (clk'event and clk='1') then
     SAL <= B;
  end if
end process;
```

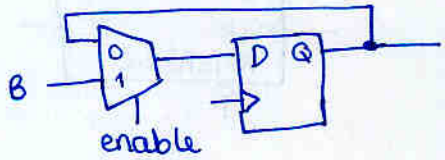
FF tipo D



podría ser

```
if (enable='1') then
    SAL <= B;
end if;
```

← cuando ve un IF, pone un MUX



Si queremos con clear asíncrono

```
process (clk, reset)
  begin
    if (reset='0') then          } parte asíncrona
        sal <= '0';
    elsif (clk'event and clk='1') then   } parte síncrona
        :
    end if;
  end if;
```

cuidado si el reset es síncrono o asíncrono.

Variables vs señales

- variables actualizan valor de forma inmediata
- señales actualizan su valor al acabar ← representa hardware

- Asignaciones de <u>señales</u> en la <u>parte síncrona</u> ⇒ 1FF por cada asignación de bit (si asignas bus 8 bit ⇒ 8FF)

- Asignación de <u>variables</u>
  el orden SI importa; las variables se asignan inmediatamente

```
C := B;
B := A;      } crea 3FF
A := IN;
```

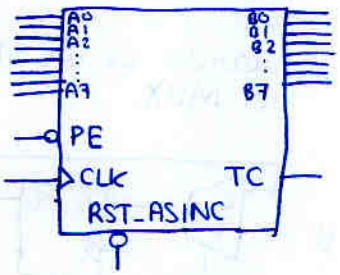
```
A := IN
B := A       } sólo 1 FF
C := B
```

Jerarquía VHDL

- Declarar
- Instanciar
- configurar ← no hace falta en Altera

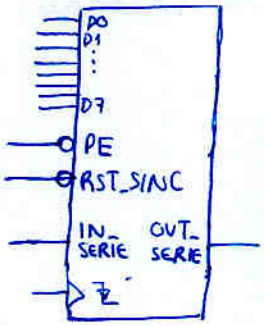
Tareas de la práctica

contador



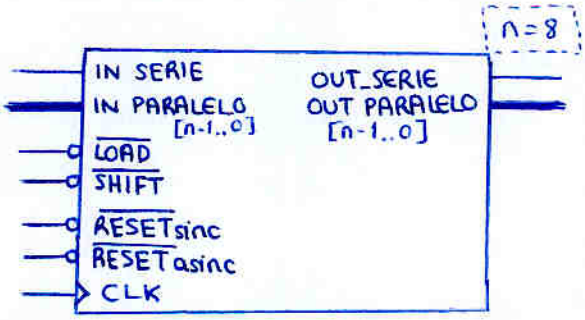
con std_logic_vector
y con integer

Registro de desplazamiento



LIFO

Registro de desplazamiento

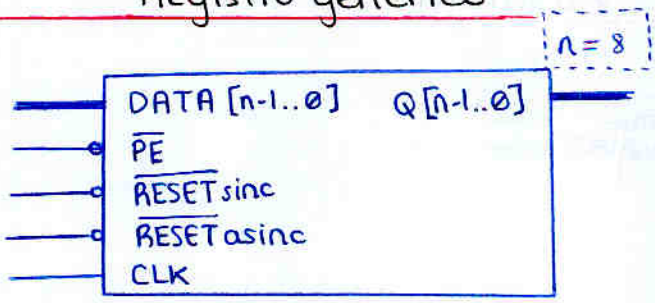



```
entity regist_desp is
        generic (constant n : integer := 8);
        port (
                IN_SERIE        : in std_logic;
                IN_PARALELO     : in std_logic_vector (n-1 downto 0);
                LOADn           : in std_logic;
                SHIFTn          : in std_logic;
                RESETn_asinc    : in std_logic;
                RESETn_sinc     : in std_logic;
                CLK             : in std_logic;
                OUT_SERIE       : out std_logic;
                OUT_PARALELO    : out std_logic_vector(n-1 downto 0));
end regist_desp;

architecture a of regist_desp is
        signal datos : std_logic_vector (n-1 downto 0);
begin
        process (CLK, RESETn_asinc) ← CLK y señales asíncronas
        begin
                if (RESETn_asinc = '0') then
                        datos <= (others => '0');
                elsif (CLK'event and CLK = '1') then
                        if (RESETn_sinc = '0') then
RESET →                         datos <= (others => '0');
                        else
                                if (LOADn = '1') then
                                        if (SHIFn = '1') then -- shift y load desactivados
HOLD →                                          datos <= datos;      ↳ hold
                                        else -- shift
SHIFT →                                         datos <= IN_SERIE & datos(n-1 downto 1);
                                        end if;
                                else -- load (prioridad sobre shift)
LOAD →                                  datos <= IN_PARALELO;
                                end if;
                        end if;
                end if;
        end process;

        OUT_SERIE <= datos(0);
        OUT_PARALELO <= datos;
end a;
```

parte asíncrona

Registro genérico

tb sirve LPM_FF




```
entity registro is
    generic ( constant n : integer := 8 );
    port (
          DATA          : in std_logic_vector (n-1 downto 0);
          PEn           : in std_logic;
          RESETn_sinc   : in std_logic;
          RESETn_asinc  : in std_logic;
          CLK           : in std_logic;
          Q             : out std_logic_vector (n-1 downto 0)
    );
end registro;

architecture a of registro is
    signal q_s : std_logic_vector (n-1 downto 0);
begin
    process (CLK, RESETn_asinc)
    begin
        if ( RESETn_asinc = '0') then
            q_s <= (others => '0');
        elsif ( CLK'event and CLK = '1') then
            if ( RESETn_sinc = '0') then
                q_s <= (others => '0');
            else
                if ( PEn = '0') then
                    q_s <= data;
                else
                    q_s <= q_s;
                end if;
            end if;
        end if;
    end process;

    Q <= q_s;

end a;
```

en gdf se puede concatenar

BUS_A[3..0], BUS_B[3..0] | BYTE [7..0]

Manejo de buses en VHDL

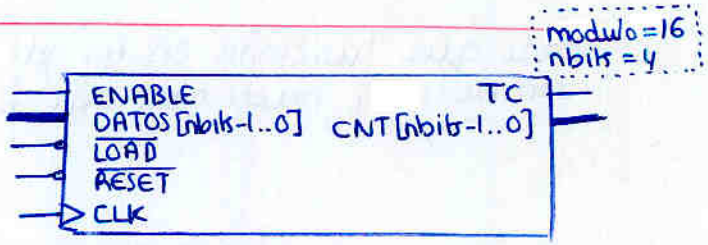
```
signal BUS_A : std_logic_vector (3 downto 0)
signal BUS_B : std_logic_vector (3 downto 0)
signal BYTE : std_logic_vector (7 downto 0)
```

```
BUS_A(3) <= '1';
BUS_B <= (3 => '1', 2 downto 1 => '0', others => '0');
BUS_A <= BUS_B;
BUS_B <= "1010";
BYTE <= BUS_A(3 downto 1) & BUS_A(0) & BUS_B
if ( BUS_B = "1010") then
```

cuidado al comparar

```
C <= "111"
D <= "1011"
if (C > D) then   -- true
```

Contador

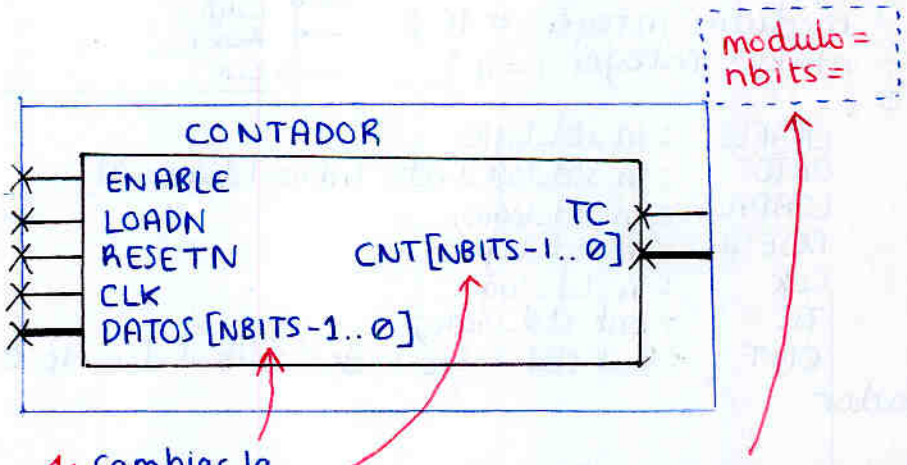



```
entity contador is
    generic (
            modulo: integer := 16;
            nbits : integer := 4 );
    ports (
            ENABLE  : in std_logic;
            DATOS   : in std_logic_vector (nbits-1 downto 0);
            LOADn   : in std_logic;
            RESETn  : in std_logic;
            CLK     : in std_logic;
            TC      : out std_logic;
            CNT     : out std_logic_vector(nbits-1 downto 0);
end contador

architecture a of contador is
    signal   cuenta : std_logic_vector (nbits-1 downto 0);
    constant modulo_menos_uno : integer := modulo - 1;
begin
    process (CLK, RESETn)
    begin
        if (RESETn = '0') then
            cuenta <=(others => '0');
        elsif (CLK'event and CLK = '1') then
            if (ENABLE = '0')then
                cuenta <= cuenta;
            elsif (LOADn = '0') then
                cuenta <= DATOS;
            else
                if (cuenta = modulo_menos_uno) then
                    cuenta <= (others => '0')
                    -- se puede hacer, para cambiar cuenta inicial:
                    -- cuenta <= CONV_STD_LOGIC_VECTOR (inicial, nbits)
                    -- siendo inicial un parámetro o constante integer
                else
                    cuenta <= cuenta + 1;
                end if;
            end if;
        end if;
    end process;

    CNT <= cuenta;

    TC <= '1' when cuenta = modulo_menos_uno else
          '0';

    -- se puede activar una salida ante cualquier nº integer de cuenta
    SALIDA <= '1' when cuenta = constante_tipo_integer else
              '0';
                                  ej: modulo_partido_dos
                                  definido junto a modulo_menos_uno

end a;
```

(i)

(i)

Para que funcione en un gdf primero hay que editar el símbolo y hacer que sea éste:



1: Cambiar la etiqueta haciendo doble clic

2: Element > Enter Parameters
Poner sus nombres dando a Add, pueden dejar Value en <none>.
Ya los pedirá al poner el símbolo en el gdf.