**Tema 2: Sistemas de numeración, aritmética y códigos**

**Operaciones en binario**

* **Suma:** Comenzando por la derecha y propagando el acarreo hacia la izquierda (1+1) = (10)
* **Resta:** Exige un prestamo en la columna de bits a la izquierda a la restada (en el caso de 0-1)
* **Producto:** Como en decimal, desplazando cada producto parcial una posición a la izquierda.
* Desbordamiento: Se produce cuando el resultado de una operación necesita un número de bits mayor que el de los operandos

**Codificación de números binarios con signo**

* **Signo-magnitud:** El bit más a la izquierda es el bit del signo, 1 negativo y 0 positivo.
  + Los bits de magnitud son el número sin signo.
  + Con n bits se pueden representar números en el intervalo {-(2n-1 -1), (2n-1 -1)}
  + Ej.: -16 = 110000
* **Complemento a 1:** El bit más a la izquierda es el bit del signo. Además, los números negativos son el complemento a 1 del correspondiente número positivo: intercambia los 0 y los 1.
  + Mismo intervalo que signo-magnitud
  + Dos representaciones posibles para el 0: 0000(+0) o 1111(-0)
  + Ej.: -16 = 101111
* **Complemento a 2:** El bit a la izquierda es el signo, los números negativos son el complemento a 2 del positivo:
  + Para calcular el complemento a 2 de un binario N de n bits, se calcula la diferencia 2^n-N y se toman los n bits más a la derecha.
    - Para los números que no son los extremos del intervalo de representación, se puede hacer el complemento a 1 y sumar 1.
    - Para los extremos del intervalo (ej. -16) se hace el complemento a 1, se suma 1 y sobra un bit a la izquierda que se elimina.
    - Ej.: -16 = ~~1~~**10000**
  + Intervalo: {-(2n-1), (2n-1 -1)} (**1 número negativo más** que signomagnitud)
  + 0 sólo tiene una representación, 0. Non existe o -0.
* **NOTA:** Para C1 y C2, los positivos son iguales que en signo magnitud.

**Operaciones en distintas notaciones**

* **Complemento a 2**
  + **Suma**: Se expresa cada operando en notación C2 y se suma igual que en binario, **suprimiendo** el acarreo final.
    - NOTA: Se deben expandir los sumandos (con 0s a la izquierda de ser positivos, o con 1s de ser negativos) para que tengan el mismo nº de bits que requiere la solución. Esos bits expandidos NO se suprimen.
  + De ser una resta, se opera como una suma con distintos signos.
  + **Producto:** Se expresan los números en positivo, se multiplica como si fuesen decimales y luego se cambia el signo del resultado para que coincida. De ser positivo se añade un 0 a la izquierda.
* **Complemento a 1:**
  + Suma: Se suma igual que en binario. El acarreo final se **suma** en el bit de la derecha (pero no el desbordamiento).
* **Signo-magnitud:**
  + Suma: Si son del mismo signo, se suman sólo los bits de magnitud. Si son de distinto signo, se coloca el número mayor (valor absoluto) encima y se restam. Luego se añade el bit del signo.
    - Como alternativa, se puede convertir a C1 los sumandos y luego volver a SM.
    - Ejemplo: 7-8 = 11000-00111 = 10001 = -1

**Sistema de numeración octal**

* Sistema de numeración posiciónal de base 8, con 8 dígitos.
  + Para convertir de octal a decimal, se multiplica cada dígito por 8^su posición.
  + De decimal a octal: convertir el decimal a binario, y de binario ao octal
  + Para convertir de octal a binario: convertir cada cifra individualmente, luego unirlas.
  + De binario a octal: Agrupar las cifras de 3 en 3 y convertirlas a octal.
    - Se comienza a agrupar desde el punto, tanto si hay decimales como no

**Sistema de numeración hexadecimal**

* Sistema de numeración posiciónal de base 16, con 16 dígitos: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
  + Para convertir de hexadecimal a binario, se sustituye cada dígito por los 4 bits correspondientes.
  + Para convertir de binario a hexadecimal, se agrupan los bits en 4 y se convierten a hexadecimal.

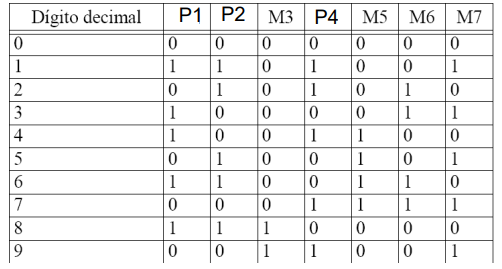
**Codificación en BCD**

* Paso directo de un número decimal a BCD: se analiza cada cifra del número decimal, se convierte a binario usando 4 bits y luego se vuelven a unir.
  + Ejemplo (51): 5 = 0101, 1=0001, 51=01010001
* Para sumar en BCD, se suman las cifras por separado.
  + Si la suma de los dígitos decimales es <=9 la suma es correcta.
  + Si es >9, habrá desbordamiento. Para corregirlo se suma **6** en binario al resultado final.
    - Ejemplo: 5+5 => 0101 + 0101 = 1010 + **0110** =>0001 0000=> 10

**Códigos con paridad**

* Códigos redundantes con un bit que indica si el número es par o impar. Se pueden usar para encontrar errores de 1 bit de distancia.
* Distancia mínima: número de bits que diferencian dos palabras distintas. Con distancia mínima=2 se detectan errores de 1 bit.

**Códigos Hamming**

* Código con distancia mínima 3 con paridad **par** (si se indica paridad impar, las combinaciones deberán tener un número de 1s impar).
* Ejemplo: Codificar las cifras del 0 al 9 en binario, pero con las posiciones 1,2 y 4 del número siendo bits de paridad.
* P1 es la paridad de M3,M5 y M7
* P2 es la paridad de M3,M6 y M7
* P4 es la paridad de M5,M6 y M7
* Si el código es correcto, el número de 1s en las combinaciones [P1,M3,M5,M7], [P2,M3,M6 y M7], [P3,M5,M6 y M7] debe ser **par**. De no serlo, conocemos que hay un error.
  + Los números de 1s en los conjuntos [P1,M3,M5,M7] y los otros conjuntos son los **bits de comprobación de paridad** C1, C2 y C4. Ci=1 si es impar, Ci=0 si es par.
  + Para un mensaje de tamaño m, se requieren k+m<=2k-1 bits de paridad. (en este caso, k=3 y m=4, lo cual cumple que 3+4<=7)

**Códigos Hamming SEC/DED**

* Código de distancia mínima = 4 que permite corrección de errores simples (en 1 bit) y detección de errores dobles.
* Para 4 bits de mensaje añade 3 bits de paridad y 1 bit de paridad total PT, que es la paridad de todos los demás bits del código.
* Situaciones posibles:
  + No hay error si PT=0 y todos los Ci=0
  + Error doble si PT=0 pero algún Ci=1
  + Error simple si PT=1 y algún Ci=1 (en ese caso, se puede localizar donde está el error por el Ci) o si CT=1 y todos los Ci=0 (error en el CT)

**Otras codificaciones**

* Codificación en exceso a 3: Se obtiene sumando 3 al número en BCD
* Código Grey: Código usado en Karnaugh, dígitos con distancia 1, sin peso
  + De binario a Grey:A cada bit se le suma el de su izquierda, sin acarreos
  + De Grey a binario: El primer bit queda igual, luego a cada bit de Grey se le suma el de su izquierda en binario.

**Tema 3: Análisis y síntesis de circuitos lógicos combinacionales**

**Fundamentos del álgebra de Boole**

* Conjunto A con dos operaciones binarias (+, \*), equivale a (OR, AND) que verifica los postulados siguientes:
  + Es un conjunto cerrado
  + Elementos neutros (a+0=a, a\*1=a)
  + Elemento complementario (a+ā=0, a\*ā=1)
  + Conmutatividad (a+b=b+a, a\*b=b\*a)
  + Asociatividad (a+[b+c]=[a+b]+c, a\*[b\*c]=[a\*b] \*c, )
  + Propiedad distributiva (a\*(b+c) = ab+ac)
* **Teoremas:**
  + a+a=a, a\*a=a
  + a+1=1, a\*0=0
  + a+ab=a, a(a+b)=a
  + a+āb=a+b, a(ā+b) = ab
  + ab+aЂ=a, (a+b)(a+Ђ)=a ¬
  + ¬(a+b) = ¬a\*¬b, ¬(a\*b) = ¬a+¬b
  + ab+aЂc = ab+ac
  + (a+b)(a+Ђ+c) = (a+b)(a+c)

**Funciones booleanas**

* Funciones de n variables que toman valor 0 o 1 y cuyas salidas toman también valores 0 o 1
* Se representan mediantes tablas de verdad o formas algebraicas
  + Forma algebraica: F(X2,X1,X0) = X2\*X0 + X2\*X1
* Función **incompletamente especificada:** Si para alguna entrada la salida es de valor desconocido o irrelevante.
  + Ejemplo: una función cuya entrada es la tirada de un dado de 6 caras requiere 3 variables, sin embargo, para el valor 1112 = 710, la salida es irrelevante.
* **Expresiones equivalentes:** Expresiones algebraicas que representan la misma función
* **Formas suma de productos (SOP)**: Expresión donde se realiza el OR de términos AND de la función.
  + **Mintérmino** (mi): número binario representado por los dígitos del término producto que contiene todas las variables de la función SOP
* **Forma POS:** AND de términos OR, producto de sumas.
  + **Maxtérmino** (Mi): Lo mismo aplicado a los sumandos de la función POS.
  + **¬mi = Mi , ¬Mi = mi**
* **Formas canónicas:**
  + Forma SOP o POS única para la función.
  + Cada término AND o OR incluye a todas las variables de la función una sola vez negadas o sin negar
  + Cada término AND o OR aparece una única vez en la expresión
* **Literales:** Cada una de las variables que aparece en un término suma o término producto
* Obtener función SOP a partir de tabla:

| a | b | c | f |
| --- | --- | --- | --- |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

Forma SOP: f(a,b,c) = m0 + m2 = ∑m(0,2) = ¬a\*¬b\*¬c + ¬a+b+¬c

Donde hay 0s se escribe ¬a, donde hay 1 se escribe a.

Forma POS: f(a,b,c) = M1\*M3\*M4\*M5\*M6\*M7 = ∏M(1,3,4,5,6,7)

= (a+b+¬c)\*(a+¬b+¬c)........

Donde hay 0s se escribe a, donde hay 1 se escribe ¬a.

Donde hay una indiferencia, se puede expresar como mintérmino o maxtérmino.

**Simplificación de expresiones lógicas**

* Aplicar álgebra de Boole y leyes de De Morgan
  + Ejemplo: F(A,B,C) = B\*(A+C)+ABC = B\*(A+C)
* Analizar las adyacencias mediante diagramas de Karnaugh
  + Dos estados de entrada son adyacentes cuando entre ellos solo cambia una de las variables.

| b \ a | 0 | 1 |
| --- | --- | --- |
| 0 | 0 | 1 |
| 1 | 0 | 1 |

* + Ejemplo: tabla de Karnaugh de ab + a¬b:

Dos valores adyacentes se agrupan, resultando en que **ab + a¬b = a**

* Obtener función a partir de tabla de Karnaugh:

| ab \ c | 0 | 1 |
| --- | --- | --- |
| 00 | 0 | X |
| 01 | 0 | 1 |
| 11 | 0 | 1 |
| 10 | 0 | X |

Para obtener la fórmula SOP, nos fijamos en los implicantes[[1]](#footnote-0) de 1s:

f(a,b,c) = b\*¬c + a\*b

Para obtener la fórmula POS, nos fijamos en los implicantes de 0s y escribimos la inversa de la fórmula correspondiente.

f(a,b,c) = (b) \* (a + ¬c)

* En una tabla de Karnaugh, las indeterminaciones (representadas con X) se pueden considerar como 0s o 1s para poder expandir los implicantes todo lo posible.

**Puertas lógicas**

* Circuitos que realizan funciones lógicas básicas.

| Puerta | Símbolo | Puerta | Símbolo |
| --- | --- | --- | --- |
| NOT |  |  | |
| AND |  | NAND |  |
| OR |  | NOR |  |
| EXOR |  | NEXOR |  |

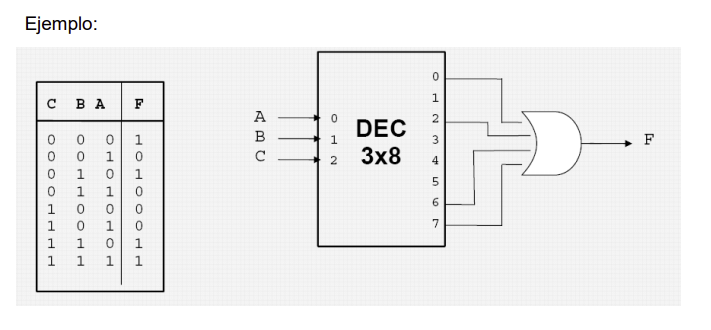
* XOR: a⊕b = ¬a\*b + a\*¬b (a ou b, pero non ambas).
  + ¬(a⊕b) = ¬a⊕b = a⊕¬b
  + a⊕0 = a
  + a\*(b⊕c) = (a\*b)⊕(a\*c)
* Las leyes de De Morgan demuestran que ¬(A+B) = ¬A\*¬B, por lo que una puerta NOR es equivalente a una AND con las entradas complementadas.
* Además, ¬(A\*B) = ¬A+¬B, por lo que una puerta NAND equivale a una OR con las entradas complementadas.

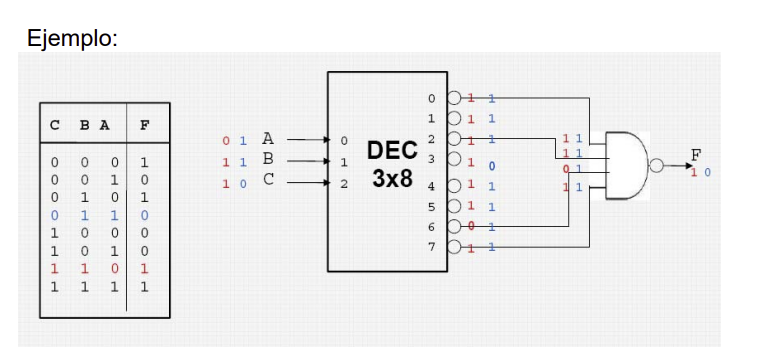
**Conjuntos completos de puertas**

* Aquel con el que se puede implementar cualquier función lógica.
  + AND y NOT
  + OR y NOT
  + NAND
  + NOR
* Con una puerta NAND, se puede crear cualquier otra puerta lógica:
  + AND: negando la NAND
  + OR: Conectando dos NAND a otra NAND
  + NOT: conectando las dos entradas de la NAND entre sí para crear una única
* Lo mismo con una NOR:
  + AND: Conectando dos NOR a otra NOR
  + OR: Negando la NOR
  + NOT: mismo método previo

**Tema 4: Lógica combinacional modular**

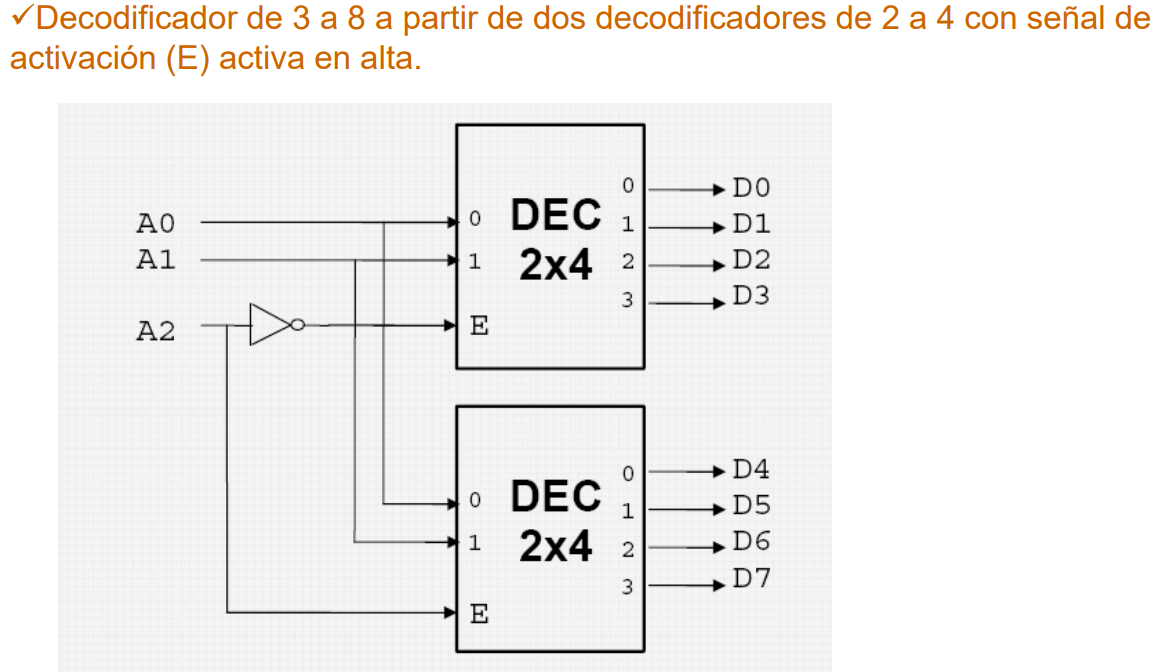
**Decodificadores**

* Activa **una** salida a 1 en función del valor binario de las entradas.
  + Para n entradas puede tener 2^n salidas.
* Se puede incluír una entrada adicional de validación (**ENA**) tal que el programa solo funciona si esta entrada está en alta o en baja según se defina.
* Un decodificador de n entradas se puede considerar un generador de mintérminos de n variables. Entonces, se puede realizar una función mediante decodificadores.
* También se puede considerar un generador de maxtérminos. Se niegan las salidas y se reemplaza el OR con un NAND-



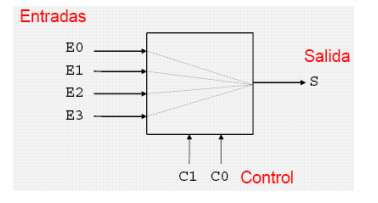
**Redes de decodificadores**

* Mediante el uso de las entradas de ENA, se puede crear una entrada que elija entre distintos decodificadores. De esta forma, se puede crear, por ejemplo, un decodificador 3 a 8 utilizando dos decodificadores 2 a 4.
* También es posible utilizar un decodificador que seleccione la entrada ENA que activar de varios decodificadores, permitiendo crear redes más grandes.

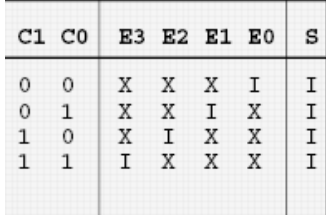
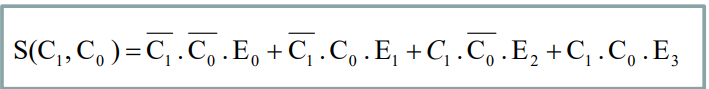


**Codificadores**

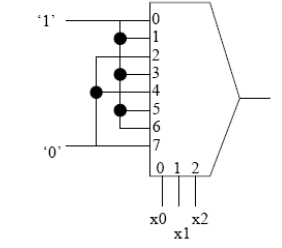
* En cada momento sólo hay una entrada activa. Asigna un código de salida a cada entrada.
  + Al hacer la tabla de valores, en los lugares con varias entradas activas la salida será una indeterminación.
* **Codificadores con prioridad**: Permite que haya más de una entrada activa simultáneamente. La salida será el código asociado a la entrada de mayor prioridad.
  + Se añade a la tabla de verdad una salida ‘GS’ que se activa si cualquiera entrada es 1 y una salida ‘E0’ que se activa sólo en el caso 0000.

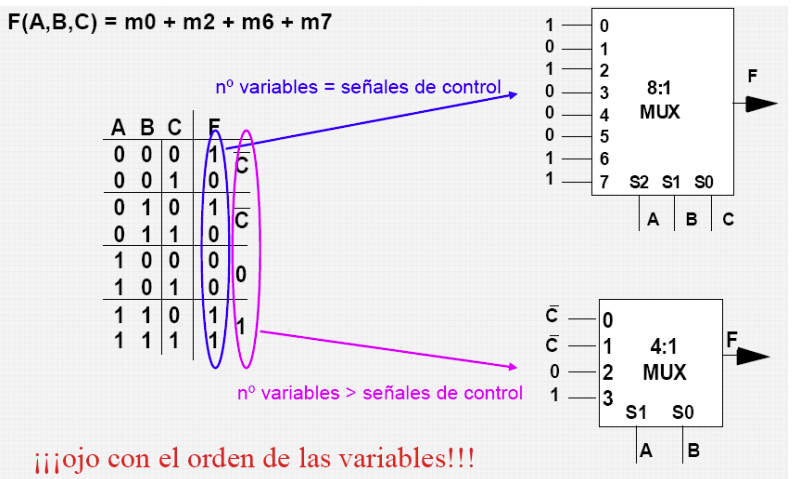


**Multiplexores**

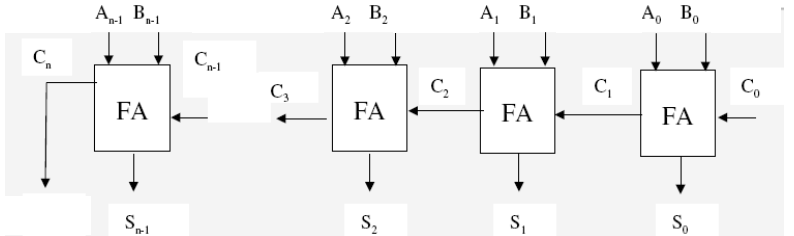
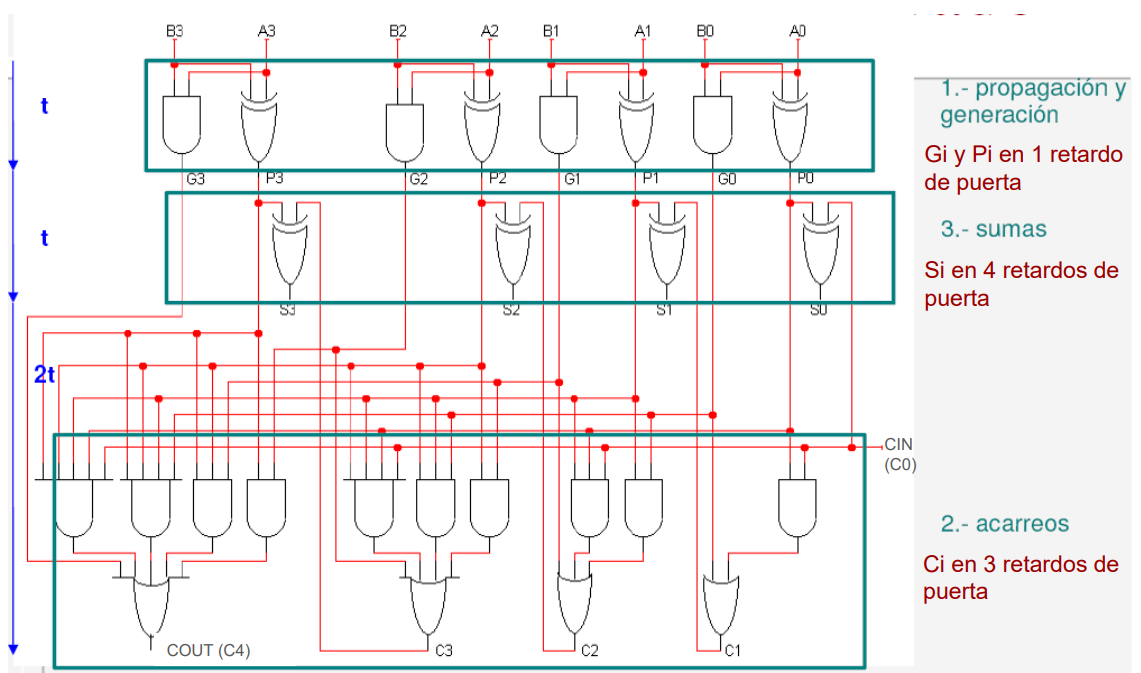
* **Multiplexor:** Dispone de 2^n entradas, n señales de control y 1 salida.
* Permite seleccionar la entrada que se utiliza mediante las señales de control.
* Na tabla de verdad, as entradas que non están seleccionadas actualmente polo interruptor márcanse con X, indeterminación.
* Logo, a saída depenerá só do valor que non ten unha indeterminación, aquel que está seleccionado polo multiplexor.
* 
* **Demultiplexor:** envía el valor binario de su entrada a la salida seleccionada con la línea del selección. Idéntico al multiplexor pero realiza la función inversa, al permitir seleccionar la salida.
  + Os valores das salidas serán, por exemplo, Q0 = S1’ \* S0’ \* A,donde S son as lineas de selección e Q as salidas.

**Funciones mediante multiplexores**

* Cualquier función de conmutación de n variables se puede representar con un multiplexor de 2^n a 1.
* Las entradas de la función se conectan a las entradas de control del MUX
* Cada entrada de datos xi del multiplexor recibe valor 0 si ‘i’ no es un mintérmino de la función, 1 si sí lo es.
* Ejemplo: F= m(0,1,3,5,6)
* Si el multiplexor tiene menos señales de control que variables la función, se expresa la salida en función de la variable.



**Circuitos aritméticos**

* **Semisumador:** Permite sumar dos bits de entrada y produce 2 bits de salida: suma(S) y acarreo (C)
  + Se pueden incluír más bits de entrada y más de salida para sumar números de varios bits. El bit de acarreo siempre será 1.
* **Sumador completo:** 3 entradas: permite sumar 2 bits de entrada y uno de acarreo.
  + **Sumador de acarreo enlazado:** Cada sumador realiza la suma en paralelo. El acarreo se propaga en serie de un sumador a otro.
    - Cada sumador completo realiza una suma y genera un acarreo que se transmite al siguiente.
    - Los tiempos de retardo se acumulan.
    - 
  + **Sumador de acarreo anticipado:** Calculanse en paralelo os valores Pi = AiXORBi e Gi = Ai^Bi,. Circuito máis complexo pero máis eficiente.

**Comparadores**

* Compraran dos números activando la salida que corresponda (<, >, =)
  + No se expresa el = como salida, pues este es el caso en el que no se cumple ninguna de las salidas
  + (A>B) = (A3>B3) + (A3=B3)\*(A2>B2) + (A3=B3) \* (A2=B2) \* (A1>B1)
    - A3>B3 = A3\*¬B3
    - A3=B3 =
* En logisim existen de complemento a 2 o en binario, hasta 8 bits.
* Donde un dígito más significativo es mayor o menor, los demás bits menos significativos ya no se consideran y se marcan como indeterminaciones.

**Tema 5: Sistemas secuenciales**

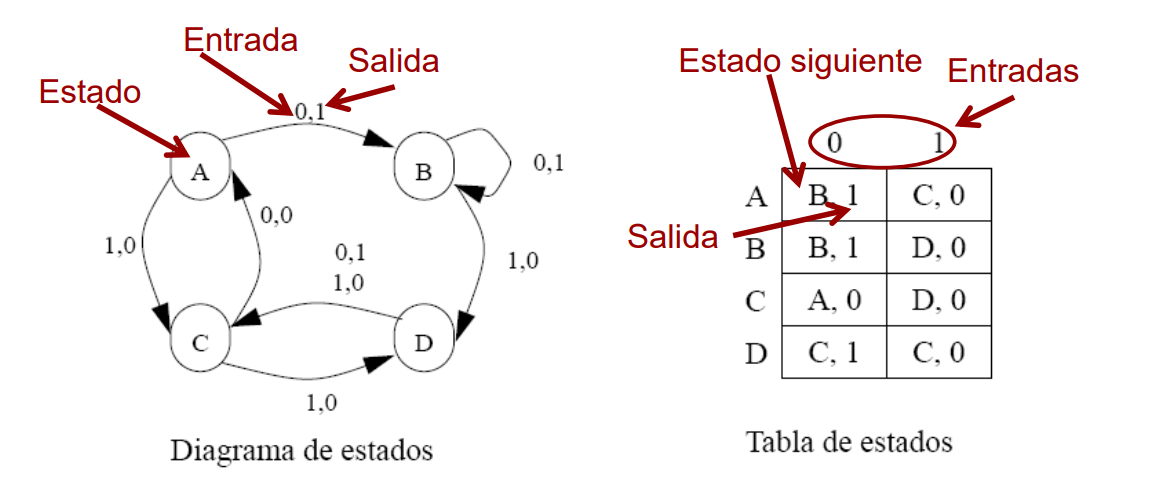
**Conceptos básicos**

* Las salidas producen una **memoria**, la cual se vuelve a conectar a las entradas e influirá en las puertas lógicas.
* La memoria almacena el estado del sistema.
  + Estado: Almacenamiento de una situación del sistema
* Utilizan una señal periódica de reloj (CLK)
  + Cuando la señal se activa se denomina ‘flanco de subida’, cuando baja es ‘flanco de bajada’.
  + Algunos sistemas se activarán solo con la CLK en alto y otros en bajo.

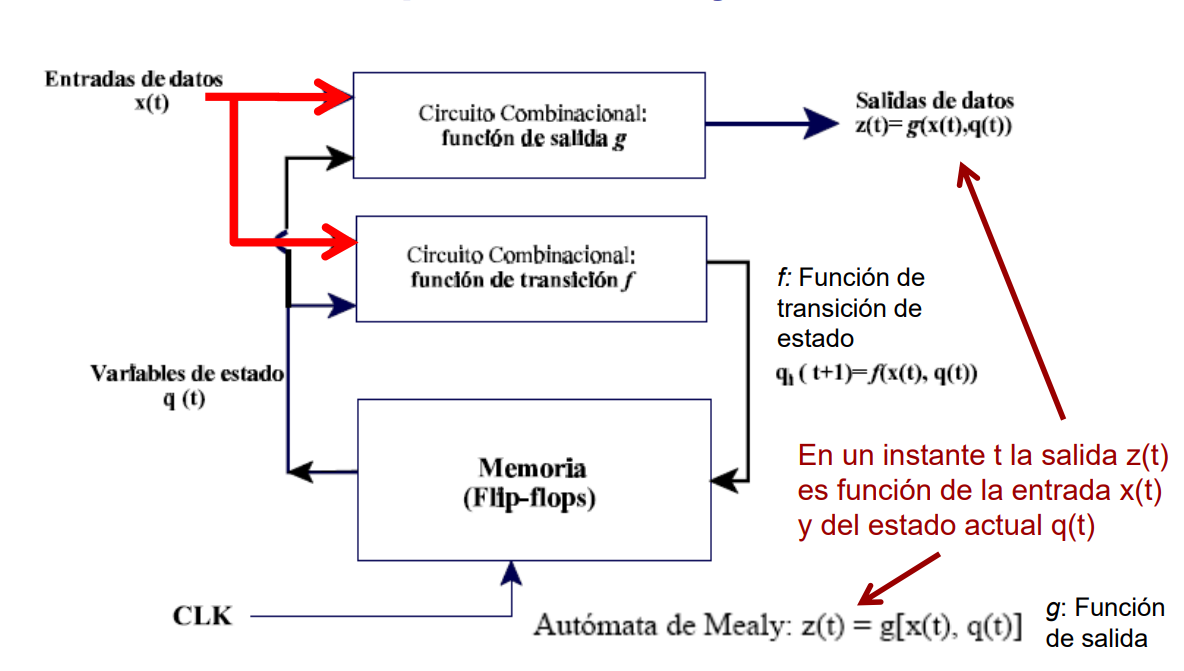
**Tipos de sistemas**

* **Asíncrono:** cambia de estado en cualquier momento en que cambien las señales de entrada
* **Síncrono:** Sólo cambia de entrada en los instantes que marca la señal de reloj.
  + Un sistema secuencial síncrono es un **autómata:** sistema electrónico capaz de tratar información
  + El modelo matemático de un sistema secuencial síncrono se llama ‘máquina de estados finitos’.

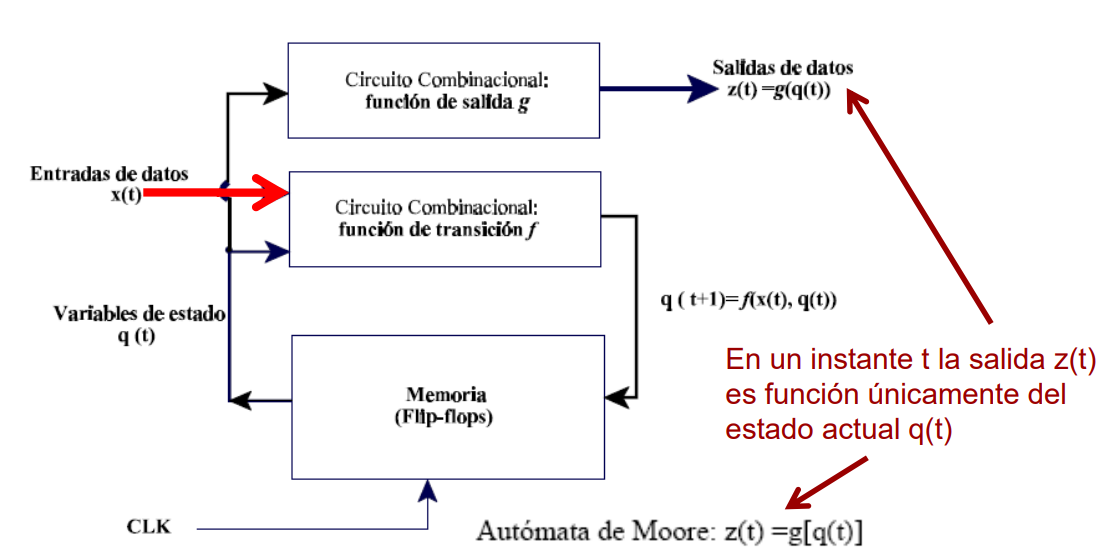
**Representaciones de sistemas**

* Se puede representar mediante un diagrama de estados o tabla de estados:
  + 
* Además hay dos formas de modelarlos: como máquina de Moore o máquina de Mealy

**Máquina de Mealy**

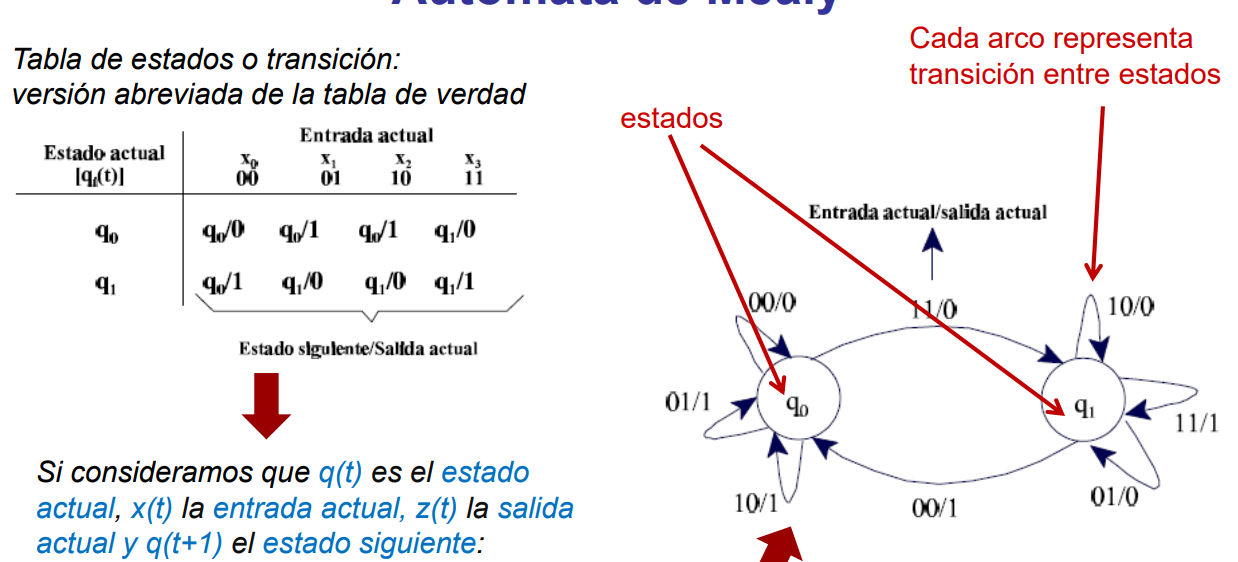
* La salida depende del estado actual y la entrada.
* 
  + Similar en funcionamiento a un ascensor
  + La función ‘f’ devuelve el próximo estado.
  + Suponiendo que el estado del sistema cambia en el flanco positivo:
    - En el resto del ciclo, la salida si cambia si cambian las entradas. Sin embargo, no cambia el estado ni se accede a la memoria.

**Máquina de Moore**

* La salida sólo depende del estado actual.
  + 
  + Las entradas modifican el valor del estado, del cual depende directamente la salida.

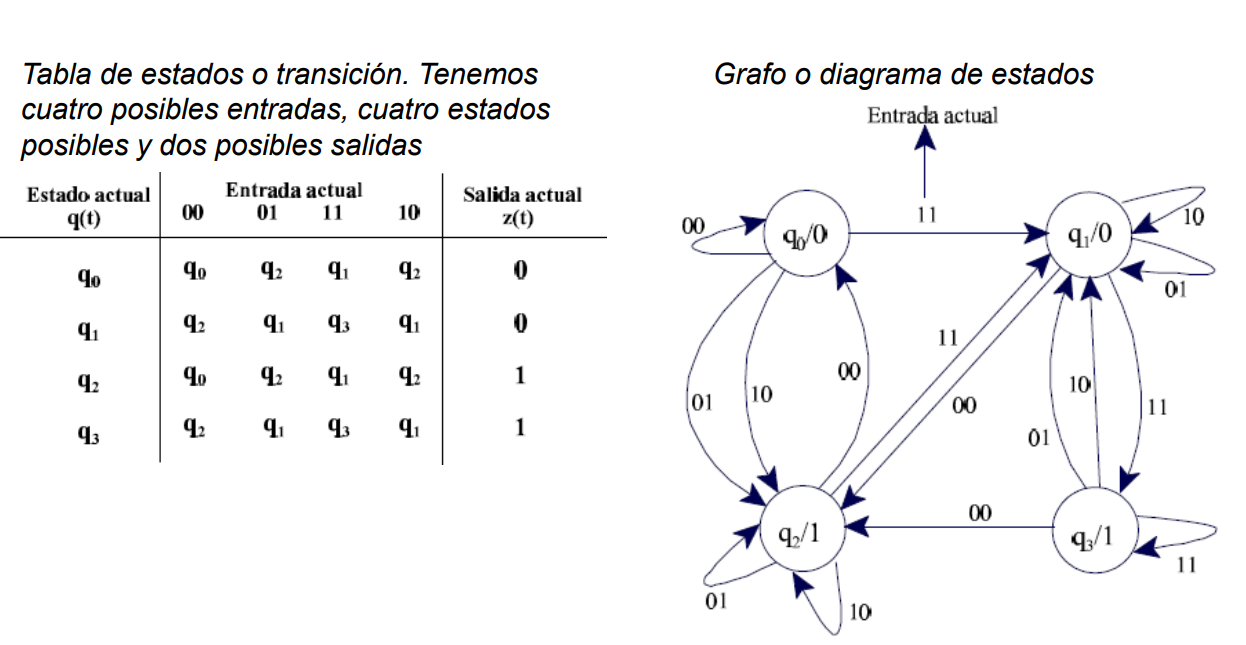
**Representaciones de un autómata de Mealy**

* Primero, se realiza la tabla de estados o transición.
* Luego, se realiza el esquema que relaciona los cambios de estado y la salida que se produce en cada movimiento.



**Representaciones de un sistema de Moore**

* La salida está dentro de la bola

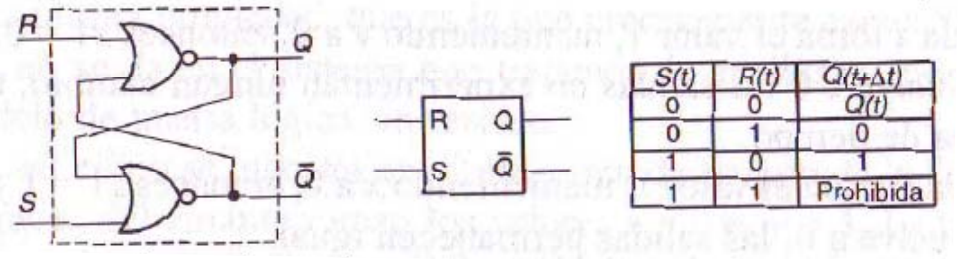


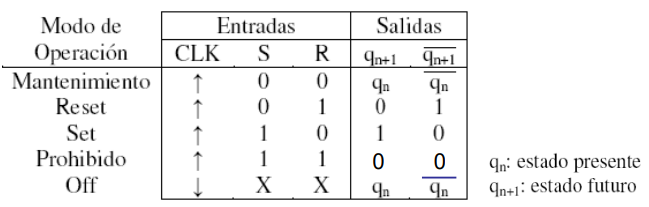
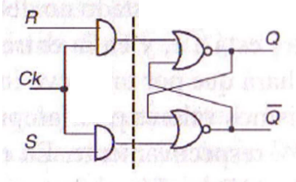
**Biestables**

* **Biestable** (tambien llamado flipflop): Circuito que consiste en un elemento de memoria capaz de almacenar un bit.
* El nombre proviene de sus dos estados posibles. En todos habrá salidas que vuelven a las entradas, creando un ciclo estable.

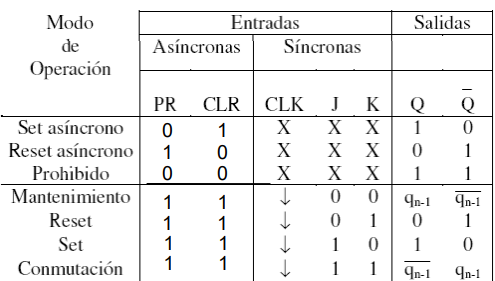
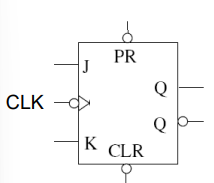
**Biestables tipo RS**

* Dos entradas R y S y dos salidas, Q y Q’ una siendo la negación de la otra. Ejemplo:

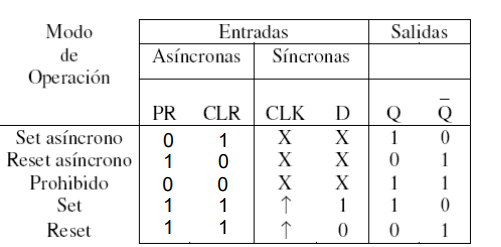
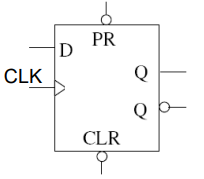


* S fija a 1, R fija a 0. Si no se activa ninguna, se preserva la salida anterior.
* Si es sincronizado por nivel, de S y R, existiría una entrada CLK (reloj) conectada con puertas AND o NAND.. El circuito funciona de forma normal cuando CLK sube. +
* 
  + Si es ‘sincronizado por flanco positivo’ se activa cuando CLK está subiendo. Si es por flanco negativo, cuando CLK baja.
* Tienen una combinación de entradas prohibida: el 1,1. Es por esto que no se suelen usar.

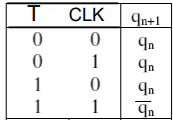
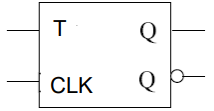
**Biestables tipo JK**

* Dos entradas de datos, J y K. Además, dos entradas de puesta a 1 (preset) y puesta a 0 (clear).
* Estas entradas son asíncronas: funcionan independientemente del reloj.
* Funcionamiento similar a un RS, pero el 1,1 invierte la salida anterior.
* Ejemplo sincronizado por flanco negativo:
* 
* Se considera que (preset) se realiza cuando PR=0. Es por esto que la combinación de entradas 0,0 está prohibida: se está realizando el preset y el clear simultáneamente.
  + Es necesario indicar que la entrada es prohibida e incorrecta. Para esto se marca que Q = Q’ = 1

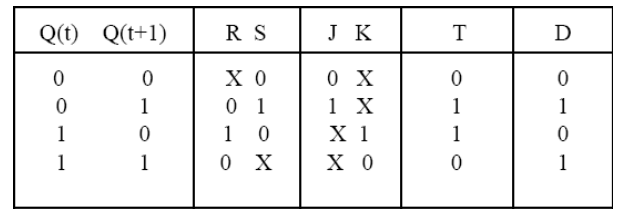
**Biestables tipo D**

* Única entrada de datos D, con las entradas de PR y CL.
* 
* En cada flanco de reloj, Q toma el valor de D.

**Biestables tipo T**

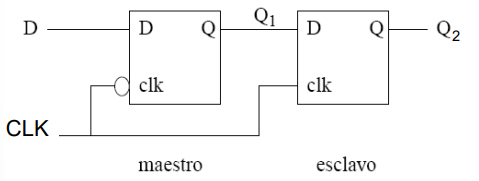
* Única entrada T además del CLK. T sólo es relevante cuando CLK=1.
* 
* T invierte la salida previa.

**Tabla de verdad de los biestables**



* **Biestable RS:** R fija la salida a 0, S fija la salida a 1. No pueden ser 1 ambas a la vez. Si ambas son 0, Q se mantiene igual.
* **Biestable JK**: J fija la salida a 1, K fija la salida a 0. Ambas a la vez invierten Q. Si ambas son 0, Q se mantiene igual.
* **Biestable D**: En cada flanco de reloj, Q toma el valor de D.
* **Biestable T**: En cada flanco de reloj, si T=1, se invierte el valor de Q.

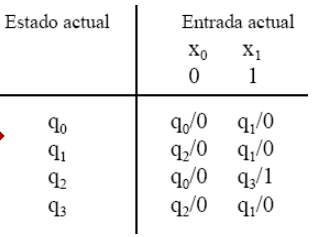
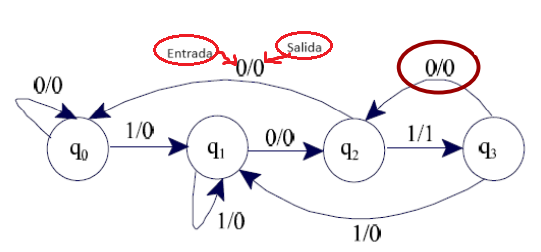
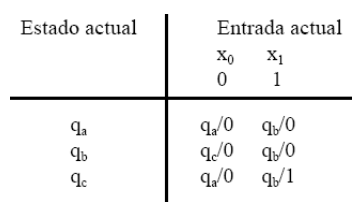
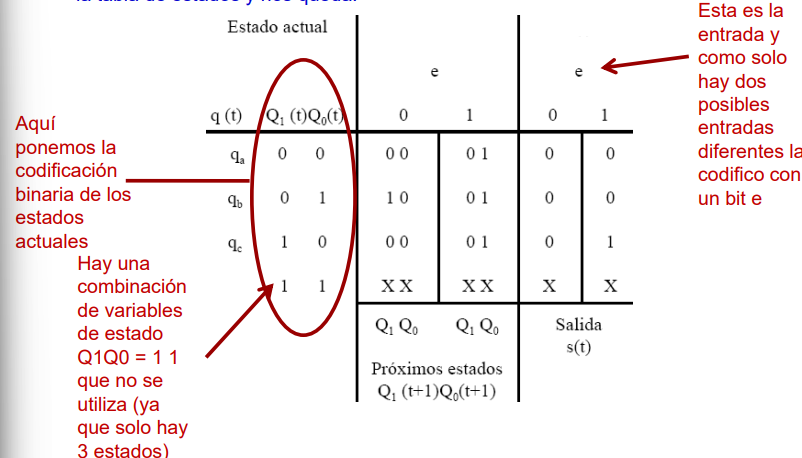
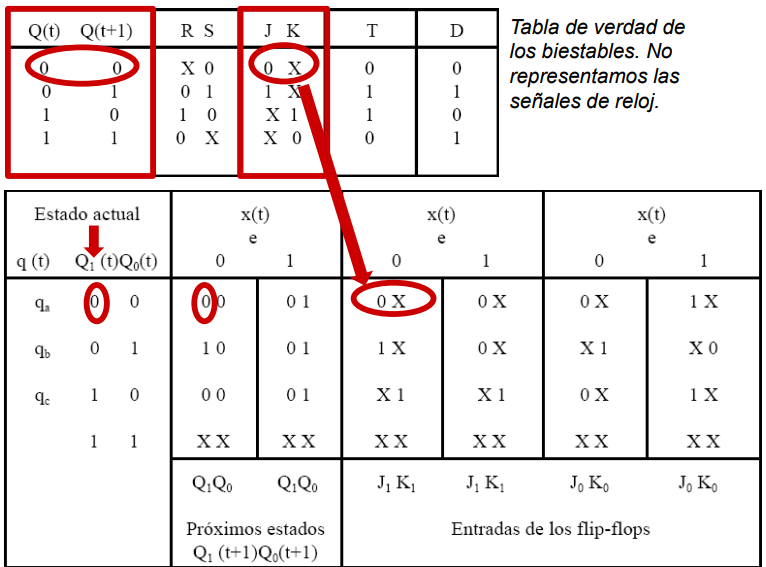
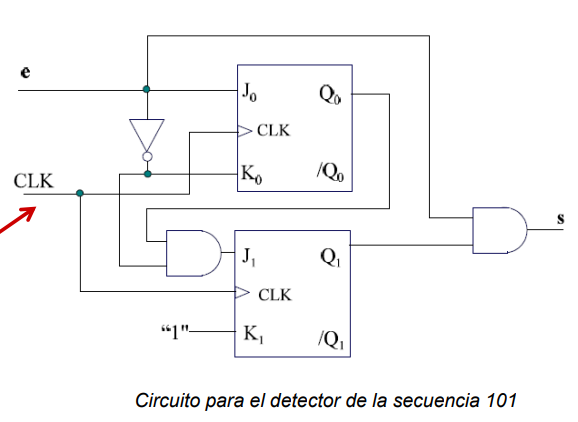
**Biestables maestro-esclavo tipo D**

* Aprovechan todo el ciclo de la señal del reloj. Combinamos dos biestables activos por nivel. Dependiendo de la CLK, estará activo uno o el otro, pero no ambos.
* 

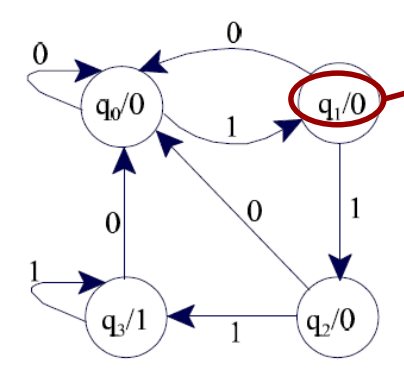
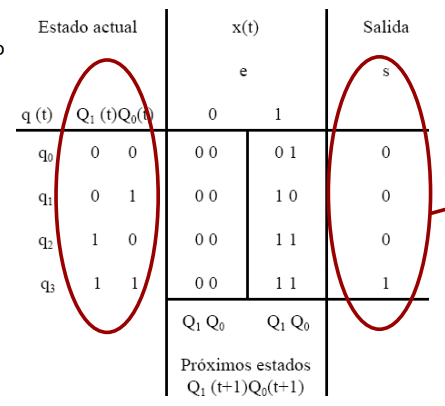
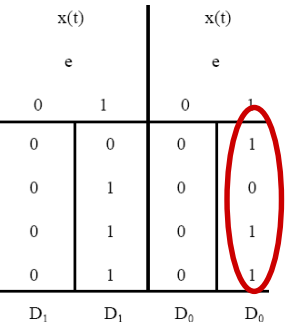
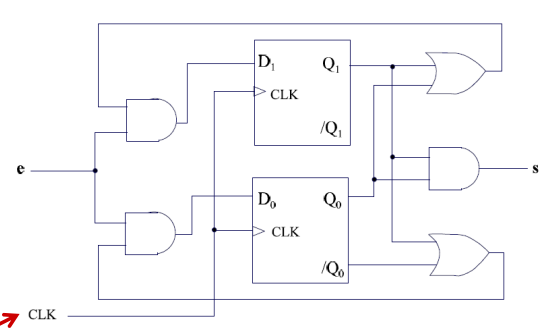
**Síntesis de circuitos secuenciales síncronos**

* A partir de la descripción de la función del circuito, obtener su esquema más sencillo.
* Pasos a seguir:
  + Planteamiento del problema
  + Obtención del grafo (diagrama de estados).
  + Obtenció y simplificación de la tabla de estados inicial
  + Asignación de codificación binaria a los estados y obtención de la tabla de transición
  + Obtención de las ecuaciones de salida del sistema y ecuaciones de entrada de cada biestable.
  + Realización del circuito.

**Ejemplo autómata de Mealy:** Secuencia 101

* Diseñar mediante un autómata de Mealy un circuito que roporcione en su salida un valor alto cuando en su línea se haya presentado 101.
* Se definen los estados como q1->’1’, q2->’10’ y q3, -> ‘101’. q0 es el estado base.
  + 
  + La salida sólo es 1 cuando han acabado las comprobaciones y se sabe que el número contiene 101.
  + Se realiza una tabla de estados, relacionando cada combinación de estado actual+entrada con la salida y estado final que produce.
  + Viendo la tabla se sabe que q1 y q3 son idénticos. Entonces se puede eliminar q3 y cambiar todas las referencias a él por qa. Repetimos la tabla, ahora con 3 estados:
* Obtenemos la tabla de estados inicial a partir de la tabla de estados simplificada.
* 
* Esta tabla contiene la misma información que la previa, pero definiendo los estados como números de 2 bits almacenados en dos variables.
* A partir de esta tabla se obtienen las expresiones de las salidas mediante Karnaugh.
  + Cada salida física se implementa en función de las entradas (e) y los estados (Q1 y Q2).
* Decidir el tipo de biestables y calcular sus funciones de entrada. Habrá tantos biestables por variables de estado, en este caso 2 (Q0  y Q1).
  + Para obtener las funciones partimos de la tabla previa.
  + Observamos la tabla de verdad de los biestables (ver arriba). A partir de esta tabla, realizamos las tablas de Karnaugh de cada uno de los biestables.
  + En este ejemplo elegimos los biestables JK.
* Observamos la tabla del biestable y la tabla de estados simplificada.
* Comprobamos el cambio de estado necesario realizar para cada entrada, y la combinación de entradas del biestable JK que causa este cambio.
  + Por ejemplo, en nuestro sistema, para la entrada 0, el estado 0 0 se transforma en 0 0. Para realizar el cambio 0→0 se utiliza la combinación de entradas 0 X en el biestable 0 y también 0 X en el biestable 1.
  + Para la entrada 0, el estado 0 1 se transforma en 1 0. Para el cambio 0→1 el biestable toma 1 X, y para el 1→0 toma X 1. Entonces, el biestable 0 toma 1 X y el biestable 1 toma X 1.
  + Entonces, obtenemos todos los valores de los biestables para cada combinación de estado actual y próximo, y entrada.
  + 
  + Para simplificar, realizamos la tabla de Karnaugh de cada una de las partes del biestable (J1, J0, K1, K0). Obtenemos que K1=1,K0=e’, J1=Q0\*e’, J0=e.
* Una vez tenemos las entradas de cada biestable, las conectamos en el circuito.
* 

**Ejemplo autómata de Moore:** Secuencia 111

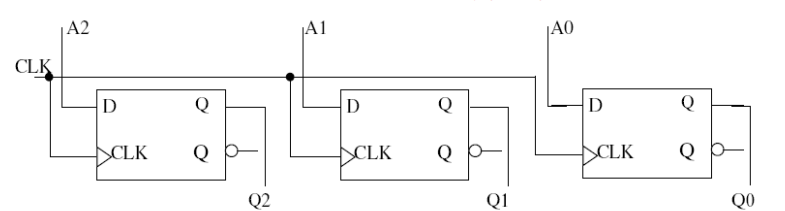
* Enunciado: Circuito que recibe por entrada datos en serie y detecta secuencias de **3 o más 1s** consecutivos (detecta 111)
* En este caso existe una sóla entrada y una sola salida.
* q0 es el estado inicial, q1 es al recibir 1, q2 es al recibir 11 y q3 es al recibir 111.
* Escribimos la tabla de estados. A diferencia de en Mealy, la salida actual y el estado siguiente se representan en columnas distintas. En este caso no es posible simplificar la tabla, por lo que la representamos con los 4 estados.
* Al ser un autómata de Moore, la salida no varía con la entrada, sólo con el estado actual.
* Obtenemos las ecuaciones de salida del sistema. Escribimos la función de salida en función del estado actual y la simplificamos con Karnaugh. En este caso obtenemos que s = Q1Q0.
* Obtenemos las funciones de entrada de los biestables. En este caso escogemos biestables tipo D, y necesitaremos 2. Sean D1 y D0.
* El valor del biestable D es 1 cuando el estado siguiente es 1, sin importar el estado anterior. Entonces, D1 y D0 copiarán los valores de Q1(t+1) y Q0(t+1), respectivamente.
* A partir de la tabla anterior realizamos los diagramas de Karnaugh. Las funciones simplificadas serán D0 = (Q0’ + Q1)\*e, D1 = (Q1 + Q0) + e.
* A partir de esta función realizamos el circuito:
* El CLK no se tiene en cuenta como variable durante la elaboración del ejercicio pero se añade después.

**Simplificación de tabla de estados**

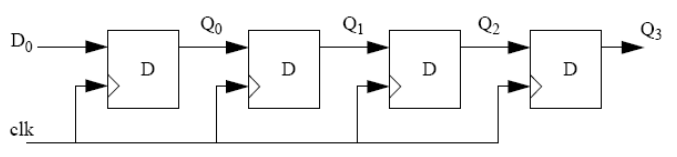
* Dos estados qi y qj se dice que son **equivalentes (qi/qj)** si y sólo si para cada combinación de entradas se cumple:
  + Las señales de salida correspodnientes son las mismas
  + Los siguientes estados correspondientes también son equivalentes.
* Pasos a seguir:
  + Dividir la tabla en **grupos de estados actuales** tales que estados actuales tengan las mismas salidas ante las mismas entradas.
    - Nombramos cada grupo con una letra (A,B,C)
  + Para cada estado actual se escribe el grupo de los estados próximos.
  + Si en algún grupo hay algún estado que no lleva a los mismos grupos en el próximo estado, los estados del grupo no son equivalentes entre sí. Entonces, dividimos el grupo en otros grupos más pequeños.
  + Repetimos el proceso hasta conseguir una tabla en la que todos los estados del mismo grupo tengan los mismos ‘grupos de estados próximos’.
  + Reescribimos la tabla de transición, tomando como nuevos estados los grupos que hemos definido.

**Tema 6: Lógica secuencial modular**

**Registros**

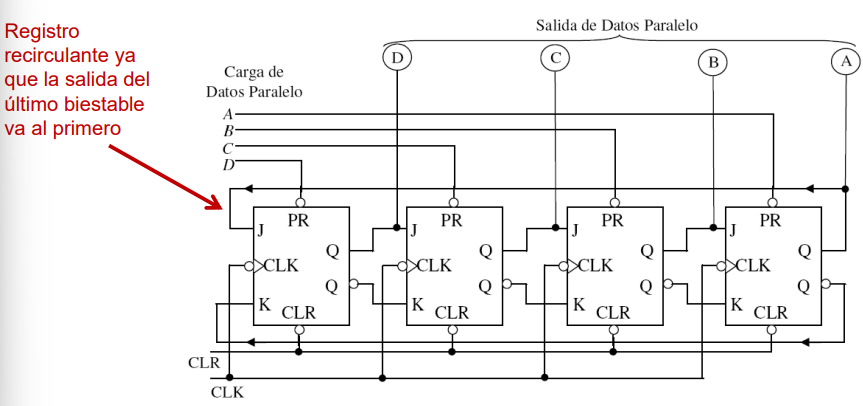
* Los **registros** se usan para almacenar una serie de bits.
* Se pueden construir como una secuencia de biestables.
* 
  + En este caso, los biestables no están conectados entre sí, pero todos usan la misma señal de reloj.
  + Cada biestable almacena un bit, este sistema almacena 3. Estos bits se devuelven por las salidas Q2, Q1 y Q0.

**Registros de desplazamiento**

* En un **registro de desplazamiento**, la salida de cada biestable está conectada como entrada del siguiente.
* Ejemplo:registro de desplazamiento a la derecha con entrada serie con biestables tipo D.
* 
  + Para realizar este biestable con desplazamiento a la izquierda, se conectaría la salida de cada biestable a la entrada del biestable de su izquierda.
  + Para realizar un biestable que permita **cambiar** entre estos dos sentidos, se utiliza un multiplexor que permita cambiar entre la salida del biestable de la izquierda o el de la derecha.
* Estos circuitos funcionan de la siguiente forma (siendo desplazamiento a la derecha y en serie): con cada señal de reloj, la entrada D0 actual se almacena en el biestable más de la izquierda, y las señales ya existentes se pasan una posición a la derecha.
  + Por ejemplo, si los valores actuales son 1010 y la entrada es un 1, pasan a ser 1101.

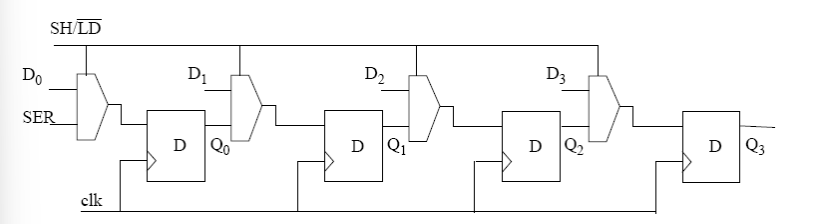
**Registros con carga en paralelo**

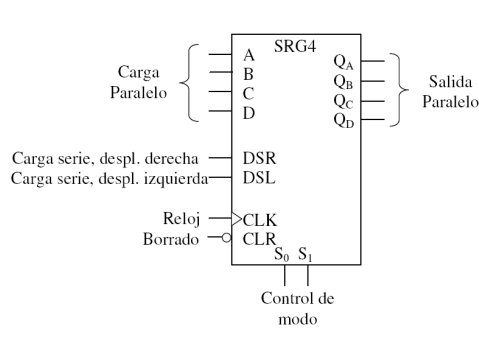
* A diferencia de los ejemplos previos, en este caso se cargan todos los bits al mismo tiempo a través de las entradas **Preset (PR)**, que son asíncronas y activas en baja.

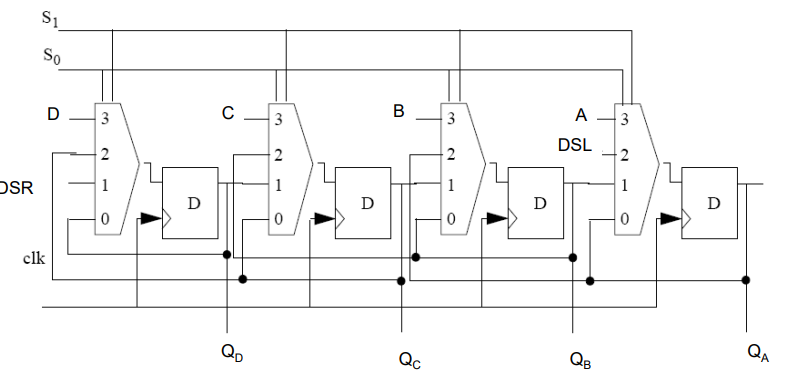


* (en logisim la entrada PRESET está situada en la parte inferior del biestable)
* La señal de borrado CLR es también asíncrona y activa en baja.
* Cada biestable tiene su propia entrada que permite asignarle su valor directamente. Por lo demás es idéntico, con cada señal de reloj avanzando los dígitos una posición a la derecha.

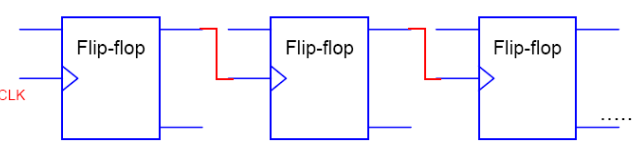
**Registros con carga en serie o paralelo**

* A la entrada de cada biestable hay un **multiplexor** de dos entradas:
  + Si SH\LD=1 se activa el desplazamiento a la derecha en serie. A cada biestable entra la salida del anterior y al primero el que haya en la entrada SER.
  + Si SH\LD = 0 se activa la carga en paralelo. En el siguiente flanco positivo del reloj, cada biestable toma su respectivo valor D0, D1, D2, D3.

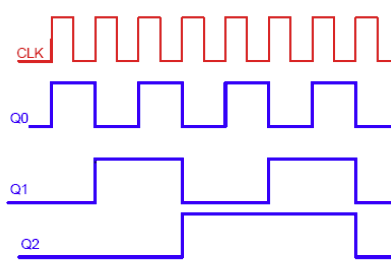
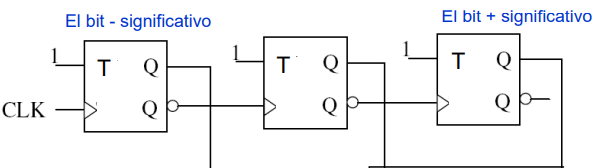
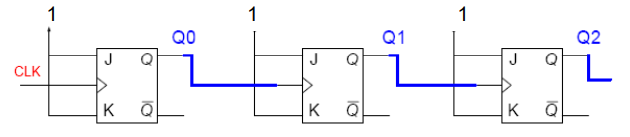
**Registro de desplazamiento universal**

* Permite realizar carga en serie o en paralelo, y desplazamiento a la izquierda o a la derecha, así como mantenimiento de los valores.
  + Las operaciones se seleccionan mediante las señales de control S1 y S0:
* **Esquema**
* Valores de (S0,S1): 0(0,0) mantiene, 1(0,1) izquierda, 2(2,0) derecha y 3(1,1) carga en paralelo.

**Contadores**

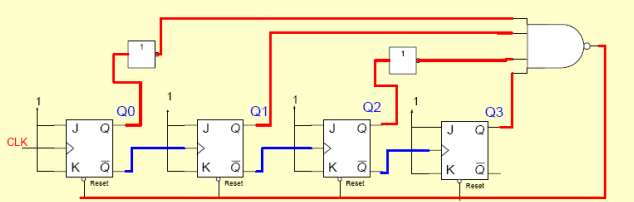
* Un **contador** es un circuito secuencial que cuenta el número de estados por los que pasa.
* El número de estados por los que pasa se denomina **módulo**. Por ejemplo, al existir 8 números binarios con 3 bits, un contador binario de 3 bits es de módulo 8.
* Si la cuenta es en binario, se trata de un contador binario.
* Si la misma señal del reloj va a todos los biestables, se trata de un contador síncrono.
  + De lo contrario, es un contador asíncrono, como el siguiente:
  + 

**Contadores binarios asíncronos**

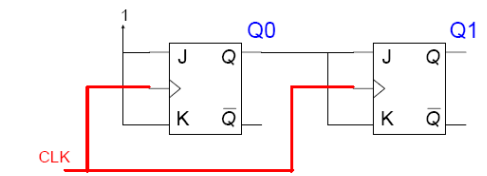
* No va la misma señal de reloj a todos los biestables, sino que están conectados en cadena, por lo que no todos cambian de estado simultáneamente. **Ejemplo:**
* 
  + (ascendientes con biestables T, módulo 8)
* Con cada pulso de reloj se avanza un número: 000, 001, 010, 011, 100…, 111, 000…
* La señal de reloj es la salida negada del anterior: cada bit se pone a 1 cuando el bit de su derecha se pone a 0. Ejemplo: 001 → 0**1**0.
* Debido a esto, se trata de un **biestable ascendente**.
* De estar conectada la señal de reloj con la salida sin negar, sería un **biestable descendiente**.
  + - (descendiente con biestables JK)

**Contadores módulo 10 asíncronos**

* Para contar 10 estados son necesarios 4 bits.
* Además, para resetear el contador después del 9, utilizamos una puerta NAND con las 4 salidas de los biestables que vaya conectada a los bits de Reset.

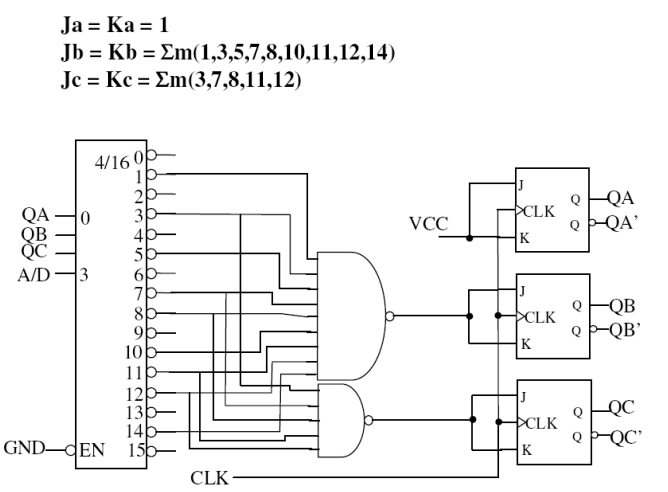
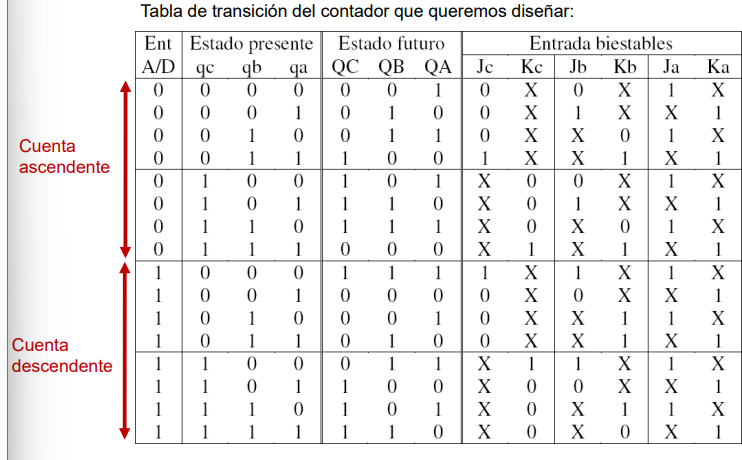


**Contadores binarios síncronos**

* Al ser síncronos, la misma señal de reloj va a todos los biestables, y todos cambian de estado a la vez.
* **Ejemplo:** Contador síncrono binario ascendiente de módulo 4 (2 bits)
* 
* Al ser un biestable JK, el segundo biestable cambia de estado (avanza 1) si se activa la señal de reloj **y** el previo bit es un 1. Entonces, 01 → **10**.

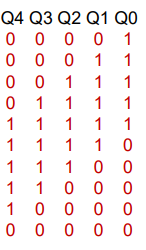
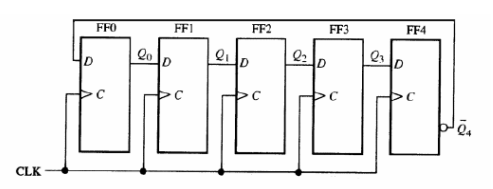
**Diseño de contadores binarios síncronos**

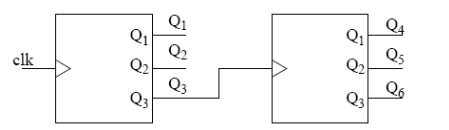
* Diseñamos un contador binario síncrono módulo 8 que sea ascendente y descendente. Creamos una entrada que indica si es ascendente o descendente.

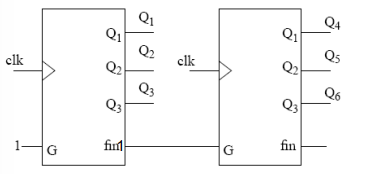


* Las entradas de los biestables se encuentran mediante su tabla de verdad, de la misma forma que con cualquier circuito secuencial.
* A partir de esta tablas obtenemos las entradas simplificadas de los biestables. Al resultar complejas, realizamos el circuito mediante un decodificador.

**Contador Johnson**

* Contador síncrono de desplazamiento, pues los biestables están unidos como en un registro de desplazamiento, pero la salida negada del último va conectada a la entrada del primero.
* Con 5 biestables se cuentan 10 estados, pero no es binario (no cuenta números binarios ordenados, sino la siguiente sucesión

**Conexión de contadores en cascada**

* Para conseguir contadores de módulo alto se pueden conectar varios pequeños.
* De ser **asíncronos**, se econectan del mismo modo que estén conectados entre sí los biestables de dentro de los contadores.
* De ser **síncronos**, se usa la señal ‘fin de cuenta’ como señal de habilitación del contador siguiente.
  + Al ser síncronos, tienen todos la misma señal de reloj.

1. Implicante: conjunto formado por celdas adyacentes

   Implicante primo: aquel que no está contenido en otro implicante

   **Implicante primo esencial:** aquel que además cubre un mintérmino que no puede cubrir ningún otro. [↑](#footnote-ref-0)