Apuntes de *Fundamentos de los computadores*

Eduardo Espuch

Resumen

Documento que se ira ampliando que agrupa los distintos conocimientos explicados en la asignatura de *Fundamentos de los computadores*, que por ahora consisten en los sistemas numéricos, la codificación binaria, el álgebra de Boole y el uso adecuado de puertas lógicas básicas en sistemas digitales.

Índice

1. Cuatro conjuntos numéricos ya las equivalencias entre sí

1.1. Sistemas de numeración

Los sistemas de numeración en base b son conjuntos que utilizan un alfabeto con b elementos para representar una gran variedad de números. Principalmente trataremos con los siguientes:

1. Sistema binario(b=2), $\mathbb{B} = \{0, 1\}$
2. Sistema octal(b=8), $\mathbb{O} = \{0, 1, 2, 3, 4, 5, 6, 7\}$
3. Sistema decimal(b=10), $\mathbb{D} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
4. Sistema hexadecimal(b=16), $\mathbb{H} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$

Para representar estos conjuntos como un número, se aplican diversos métodos de conversión . Por lo general el número que se representa equivale al aportado por el sistema decimal, aunque esto depende respecto al tipo de codificación usada (se vera más adelante):

Ejemplo de equivalencia de los 16 primeros términos:

Binario	Octal	Decimal	Hexadecimal
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	10	8	8
1001	11	9	9
1010	12	10	A
1011	13	11	B
1100	14	12	C
1101	15	13	D
1110	16	14	E
1111	17	15	F

(1)

Notese que el entre el binario, el octal y el hexadecimal, existe una correlacion con el número de caracteres tal que $2^1 x 2^3 = 2^4$ o lo que es lo mismo 2x8 = 16, siendo esto de ayuda para realizar conversiones de forma directa (un binario de 3 caracteres puede representar 8 números y un binario de 4, 16).

1.2. Sistemas de codificación y representación básicos de números

Los siguientes mecanismos de cambio de base o codificación están conectados, es decir, existe una correlación entre todos ellos con lo cual es posible que un valor tenga varias representaciones en los diversos sistemas.

Binario: Dado un número codificado en sistema binario(b=2), con $\mathbb{B} = \{0, 1\}$, observamos que la conversión a los distintos sistemas se produce de la siguiente forma:

A. A sistema octal(b=8): se agrupan en términos de 3 caracteres de derecha a izquierda en la parte entera y de izquierda a derecha la fraccionaria, rellenando con ceros los espacios necesarios para cerrar un grupo y se cambian por su equivalente. Fijarse en la tabla (1)

B. A sistema decimal(b=10): se entiende al número en binario como una sucesión de términos ordenados de derecha a izquierda con origen en el primer entero($i = 0$) y se considera el siguiente sumatorio $\sum_i x_i \cdot b^i$ donde, si consideramos el numero en binario $x = 1010,011$, podemos descomponerlo y obtener $x_3x_2x_1x_0, x_{-1}x_{-2}x_{-3}$ o, lo que es lo mismo, la sucesión de $x_i \in \mathbb{B}$ con $i = \{-3, 3\}$. Sabiendo esto aplicamos el sumatorio considerando que b es la base con la que hemos estado operando ($b = 2$) y obtenemos un número en representación decimal.

C. A sistema hexadecimal(b=16): se agrupan en términos de 4 caracteres de derecha a izquierda en la parte entera y de izquierda a derecha la fraccionaria, rellenando con ceros los espacios necesarios para cerrar un grupo y se cambian por su equivalente. Fijarse en la tabla (1)

Octal: Dado un número codificado en sistema octal(b=8), con $\mathbb{O} = \{0, 1, 2, 3, 4, 5, 6, 7\}$, observamos que la conversión a los distintos sistemas se produce de la siguiente forma:

A. A sistema binario(b=2): se descompone un término en 3 caracteres respetando el orden de cada término. Fijarse en la tabla (1)

B. A sistema decimal(b=10): Considerando el como descomponer un número en un determinado sistema a términos ordenados de una sucesión (visto en la conversión de binario a decimal) se aplica el siguiente sumatorio $\sum_i x_i \cdot b^i$, considerando que $b = 8$.

C. A sistema hexadecimal(b=16): se obtiene el binario y se aplica la conversión de binario a hexadecimal.

Decimal: Dado un número codificado en sistema decimal(b=10), con $\mathbb{D} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, observamos que la conversión a los distintos sistemas se produce de la misma forma pero cambiando un valor:

A. Parte entera: tomamos la parte entera del número en decimal y lo dividimos por b, el cociente obtenido lo volvemos a dividir por b y asi sucesivamente hasta que el cociente no pueda dividirse más. Tomaremos el número equivalente en base b del número en decimal a aquel que se forma con los restos de las divisiones y el ultimo cociente calculado, siendo el primer resto el valor mas a la derecha y el ultimo cociente el valor mas a la izquierda. El valor usado en b define a que sistema se esta convirtiendo y, usando la tabla 1 se sabe para los casos en el que el resto es un valor superior a 9.

B. Parte fraccionaria: tomando únicamente la parte fraccionaria, la multiplicamos por b, volviendo a multiplicar por b ahora el resultado. La parte entera que se obtenga con el orden usual en los resultados será la parte fraccionaria que corresponderá al número al cual tratamos de obtener representado en el sistema de base b

Hexadecimal: Dado un número codificado en sistema hexadecimal(b=16), con $\mathbb{H} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$, observamos que la conversión a los distintos sistemas se produce de la siguiente forma:

A. A sistema binario(b=2): se descompone un término en 4 caracteres respetando el orden de cada término. Fijarse en la tabla (1)

B. A sistema octal(b=8): se obtiene el binario y se aplica la conversión de binario a octal.

C. A sistema decimal(b=10): Considerando el como descomponer un número en un determinado sistema a términos ordenados de una sucesión (visto en la conversión de binario a decimal) se aplica el siguiente sumatorio $\sum_i x_i \cdot b^i$, considerando que $b = 16$.

Para referirnos al conjunto de valores representable por un sistema numérico de base b hablaremos de rangos de representacion, los cuales tienen en cuenta la cantidad de valores/cifras/bits (n) empleados en representar el total de combinaciones la base en la que se trabaje, obteniendo b^n combinaciones y abordando el conjunto $[0, \ldots, b^n - 1]$.

1.3. Aritmética binaria

Con este apartado, consideramos un cuerpo $\mathcal{B} = \{0, 1\}$ en el cual se define una operación asociativa no usual ($0001 \oplus 0001 = 10$ y $0001 \ominus 0011 = -0010$) y la operación multiplicativa usual y dejamos con la siguiente tabla las distintas operaciones definidas en él.

A	B	A+B	A	B	A-B	A	B	AxB	A	B	A/B
0	0	0	0	0	0	0	0	0	10	10	1
0	1	1	0	1	(1)1	0	1	0	10	11	0,111...
1	0	1	1	0	1	1	0	0	11	10	1,1
1	1	(1)0	1	1	0	1	1	1	11	01	11

En la aritmética binaria usual (con binario natural) existe un mecanismo llamado acarreamiento, lo que implica que se suma o resta un 1 en el caso de que este quede fuera de la operación (trabajas con 5 bits pero la operación realizada te saca un 1 para el sexto, dicho 1 suma o resta al número de nuevo).

1.4. Sistemas de codificación y representación avanzados de números

Veamos por encima los tres códigos basados en sistemas binarios con los que podremos trabajar:

1. Binario natural: el usado hasta ahora, donde para convertir un binario en decimal se aplican las conversion anteriormente explicadas.
2. Código Gray: similar al binario natural, éste es no ponderado, continuo y ciclico donde dos números sucesivos sólo varían en un bit tal que

2 bits	4 bits	Decimal	Binario natural
00	0000	0	0000
01	0001	1	0001
11	0011	2	0010
10	0010	3	0011
	0110	4	0100
	0111	5	0101
	0101	6	0110
	0100	7	0111
	1100	8	1000
	1101	9	1001
	1111	10	1010
	1110	11	1011
	1010	12	1100
	1011	13	1101
	1001	14	1110
	1000	15	1111

3. BCD (Binary Coded Decimal): de este existen varias ramas pero nos bastara con saber la natural (y la exceso X donde es sumar X a cada valor de forma individual). La codificación en BCD natural consiste en entender codificar cada dígito decimal con una combinación de 4 dígitos en binario natural.

Veamos la representación de números por partes, primero veamos los enteros y después los reales:

1.4.1. Representación de enteros

El conjunto de números enteros $\mathbb{Z} = \{-\infty, \ldots, 0, \ldots, +\infty\}$ se caracteriza por la existencia del término negativo, por ende, debes considerarse que trabajando con los números de los distintos sistemas numéricos con términos positivos y negativos.

Antes de empezar, debemos tener claro que trabajaremos con un numero limitado de cifras o bits junto con un signo. Por lo general, usaremos en este curso el sistema binario considerando que b=2.

A. Signo y Magnitud (SM)

   Método que reserva el bit en la posición más alta (a la izquierda) al signo (0 si es positivo y 1 si es negativo) y deja el resto al valor numérico en valor absoluto. Existe doble representación del 0 y su rango de representación abarca $\mathbb{SM} = \{-(b^{n-1}-1), \ldots, 0, \ldots, b^{n-1}-1\}$ con b la base con la que se trabaja y n el número de bits usados. Esta explicación considerarla para el sistema binario, con el resto considerar el uso del signo.

   El volver al sistema numérico natural consiste en fijarse en el primer bit.

B. Complemento a base menos 1($\mathbb{C}_{b-1}$)

   Método que combina el SM y operaciones aritméticas. Los valores positivos se representan en SM, los negativos se obtienen de restar $b^n - 1$ con b la base en la que trabajamos y n el total de digitos que se hacen uso (el $-$ se cuenta). Esto permite representar en positivo los valores negativos para convertir restas en sumas (existe acarreamiento). En particular, con $\mathbb{C}_1$ basta con intercambiar 1 por 0 y viceversa. Su rango de representación es $\mathbb{C}_{b-1} = \{-(b^{n-1}-1), \ldots, 0, \ldots, b^{n-1}-1\}$

   Para volver al sistema numérico natural basta con despejarlo de la operación $C_{b-1} = nt - b^n - 1 \Leftrightarrow C_{b-1} + b^n + 1 = nt$

C. Complemento a base($\mathbb{C}_b$)

   Partiendo del $\mathbb{C}_{b-1}$, basta con saber que con los valores positivos es igual pero con los negativos se suma 1 al valor obtenido, es decir, para positivos el sistema en base b natural, el $\mathbb{C}_{b-1}$ y $\mathbb{C}_b$ coinciden(considerando un 0 delante por el término positivos) y los negativos cumplen que, asumiendo que nt es el número en el sistema en base b natural, $\mathbb{C}_{b-1} = nt - b^n - 1$ y $\mathbb{C}_b = C_{b-1} + 1 = (nt - b^n - 1) + 1$ siendo importante el no simplificar esta operación. Su rango de representación es $\mathbb{C}_b = \{-(b^{n-1}), \ldots, 0, \ldots, b^{n-1}-1\}$.

   Para el caso del sistema binario, podemos buscar el primer bit de derecha a izquierda con valor 1 y intercambiar los valores a continuación por la izquierda de 1 a 0 y viceversa. La conversión al sistema numérico natural consistiría en despejar la operación. Además, al operar con números en complemento a base no se considera acarreamiento pero si convertimos restas en sumas.

D. Representación sesgada ($\mathcal{S}$)

   Considerando que los términos negativos los podemos hacer positivos aplicando aplicando cualquiera de los 3 métodos anteriores y que lo usaremos básicamente sobre el sistema binario, consideremos que la representación sesgada consiste en sumar un sesgo (2^{n-1} con n el total de dígitos en uso) al número en cuestion, sin importar su signo, de tal manera que siendo n el número ya sea en natural, $\mathbb{SM}$,$\mathbb{C}_1$ y $\mathbb{C}_2$, se cumple $\mathcal{S} = n + 2^{n-1}$ pudiendo de esta operación despejar el número natural equivalente.

   Los sesgos con 1 al principio suelen ser positivos y los que tienen 0, negativos. Su rango de representación abarca $\mathcal{S} = \{-2^{n-1}, \ldots, 0, \ldots, 2^{n-1}-1\}$.

1.4.2. Representación de reales

Con la representación de los números reales incluimos la parte fraccionaria (a la derecha de la coma) a la parte entera, permitiéndonos obtener un número mas especifico que con los enteros. Distinguiremos la representación en coma fija y en coma flotante, pero haremos mayor hincapié ya que en coma fija basta con saber que del total de bit disponibles, una cantidad esta reservada a la parte entera y la otra a la parte fraccionaria, teniendo un rango de $\{-(2^{n-1}-1),\ldots,0,\ldots,2^{n-1}-1\}$.

Con la representación en coma flotante debemos distinguir 3 partes, las cuales son:

1. Mantisa o coeficiente

   Representado con $(s)M$ siendo s el signo y M la mantisa, es un número formado por un único dígito significativo en la parte entera seguido del resto de dígitos en la parte fraccionaria.

2. Base

   Representado por b, es la base sobre la que trabajamos

3. Exponente

   De valor estrictamente entero y representado por e, eleva la base obteniendo una potencia.

El resultado final quedaría $\mathcal{N} \equiv (s)M \cdot b^e$.

Esta es la forma general y se nos puede pedir el representar el sesgo y la mantisa con distintos métodos de representación, pero se ha establecido un estándar en el cual la forma de representar cada elemento es única, veamos lo:

- Representación estándar IEEE745 (IE^3).

  Utilizando el modelo SEM (Signo Exponente Mantisa) con la base fijada en 2, la forma mas comun, la de precisión simple, utiliza 32 bits distribuidos de una forma en particular para representar un real en coma flotante. Considerando N el total de bits, nS los bits dedicados al signo, nE los dedicados al exponente y nM los dedicados a la mantisa, vemos que $N = nS + nE + nM$. Debe considerarse que existe la versión con la mantisa normalizada y desnormalizada (pero realmente la diferencia es en el exponente con lo cual lo estudiaremos ahi).

+ Signo

  El primer bit esta destinado a indicar si es positivo (0) o negativo (1).

+ Mantisa

  Son los últimos 23 bits y se entiende que el número dado en el IE^3 es únicamente la parte fraccionaria (nos referiremos a ella como m) pero se considera oculta una parte entera con lo cual $M = [1.m]$ cumpliéndose $1 < M < 2$. m por lo general se representa en binario natural.

+ Exponetne

  Los 8 bits restante situados entre el signo y la mantisa están reservados al exponente, que se representa con el modelo sesgado, de tal manera que $E = e - \mathcal{S}$ siendo E el valor numérico del exponente, e el número representado en sesgado y $\mathcal{S}$ el sesgo, que en formato de mantisa normalizada es $\mathcal{S} = 2^{nE} - 1$.

  Hablaremos de mantisa desnormalizada cuando $e = 0\ldots0$, en esta caso tomaremos $\mathcal{S}_d = 2^{nE} - 2$ haciendo que el valor del exponente sea ahora $E = e - \mathcal{S}_d$ produciendose que $E = 0\ldots0 - (2^{nE} - 2) = -2^{nE} + 2$. Se utiliza para números próximos al 0.

Recordar que dada $m = 0\ldots0$, dará 0 si $e = 0\ldots0$ y ∞ si $e = 1\ldots1$, que para redondear un número en IE^3 lo habitual es el redondeo al par (si hay dos 1 consecutivos, se suma 1 en la posción siguiente, si no sucede ésto se truncan) y para representar el rango que abarca veremos:

Nº normalizados		Nº desnormalizados	
$\|b\| = M_{max}2^{E_{max}}$	$M_{max} = 2 - 2^{-nM}$		
	$E_{max} = 2^{nE-1} - 1$		
$\|a\| = M_{min}2^{E_{min}}$	$M_{min} = 1$	$\|a'\| = M'_{min}2^{E'_{min}}$	$M'_{min} = 2^{-nM}$
	$E_{min} = -(2^{nE-1} - 2)$		$E'_{min} = -(2^{nE-1} - 2)$

donde

$M = \{1, \ldots, 1{,}111\ldots\}$ con $1 < M < 2$ y $nM = 23$ para IE^3

$E = \{-126, \ldots, +127\}$ con $nE = 8$ para IE^3

siendo el rango $\mathcal{N} = [-b, -a'] \cup [a', b]$ y hablaremos de OVERFLOW cuando $|\mathcal{N}| > |b|$ (desborda en dirección de $\pm\infty$) y de UNDERFLOW cuando $|\mathcal{N}| < |a'|$ (desborda en dirección de 0).

2. Álgebra de Boole. Fundamentos de los sistemas digitales.

Trabajando sobre un cuerpo en el que predomina la aritmética binaria, definimos un conjunto de propiedades que dará lugar al Álgebra de Boole, permitiéndonos estudiar de forma teórica el funcionamiento de los transistores, un componente esencial en los sistemas digitales que actúa como llave electrónica o conmutador entre dos estados: encendido (1) o apagado(0).

2.1. Álgebra de Boole

El Álgebra de Boole es una gran herramienta matemática para analizar y diseñar circuitos digitales, considerando el conjunto con el que trabaja ($\mathbb{B} = \{0, 1\}$) y la operaciones descritas en él (aritmética binaria). Considerando ésto, veamos algunas propiedades básicas.

2.1.1. Conocimientos básicos

RECORDATORIO:

Consideramos un cuerpo $(\mathbb{B}, \oplus, \cdot)$ con $\mathbb{B} = \{0, 1\}$ en el cual se define una operación asociativa no usual ($0001 \oplus 0001 = 0010$ y $0001 \ominus 0011 = -0010$) y la operación multiplicativa usual y dejamos con la siguiente tabla las distintas operaciones definidas en él.

A	B	A+B	A	B	A-B	A	B	AxB	A	B	A/B
0	0	0	0	0	0	0	0	0	10	10	1
0	1	1	0	1	(1)1	0	1	0	10	11	0,111...
1	0	1	1	0	1	1	0	0	11	10	1,1
1	1	(1)0	1	1	0	1	1	1	11	01	11

Las propiedades que obtenemos directamente de definir el cuerpo $(\mathbb{B}, \oplus, \cdot)$ son llamadas axiomas, y las que desarrollamos a partir de estos los llamaremos leyes o teoremas, veamos los esenciales y demostremos algunos de ellos:

Identificador	Nombre/Propiedad	Igualdad
Ax.1	Axioma 1 Asociativa	$(a+b)+c = a+(b+c)$
Ax.2	Axioma 2 Conmutativa	$a+b = b+a$ $a{\cdot}b = b{\cdot}a$
Ax.3	Axioma 3 Complementación	$a+\overline{a} = 1$ $a{\cdot}\overline{a} = 0$
Ax.4	Axioma 4 Absorción	$a+(a{\cdot}b) = a$ $a{\cdot}(a+b) = a$
Ax.5	Axioma 5 Idempotencia	$a+a = a$ $a{\cdot}a = a$
Ax.6	Axioma 6 Distributiva	$a{\cdot}(b+c) = a{\cdot}b + a{\cdot}c$ $a+(b{\cdot}c) = (a+b){\cdot}(a+c)$
PD	Teorema 1 Principio de dualidad	Dado un enunciado que es válido, su enunciado dual ($1 \Leftrightarrow 0$ y $+ \Leftrightarrow \cdot$) también lo sera
EN	Teorema 2 Elementos neutros	$a+0 = a$, 0 es el neutro en sumas $a{\cdot}1 = a$, 1 es el neutro en productos
EA	Teorema 3 Elementos absorbentes	$a+1 = 1$, 1 es el absorbente en sumas $a{\cdot}0 = 0$, 0 es el absorbente en productos
LsM	Teorema 4 Leyes de Morgan	$\overline{a+b} = \overline{a}{\cdot}\overline{b}$ $\overline{a{\cdot}b} = \overline{a}+\overline{b}$
TI	Teorema 5 Teorema de involución	$\overline{\overline{a}} = a$
TCON	Teorema 6 Teorema del consenso	$a{\cdot}b + \overline{a}{\cdot}c = a{\cdot}b + \overline{a}{\cdot}c + b{\cdot}c$ $(a+b){\cdot}(\overline{a}+c) = (a+b){\cdot}(\overline{a}+c){\cdot}(b+c)$
TCAN	Teorema 7 Teorema de cancelación	$a+\overline{a}{\cdot}b = a+b$
TSH	Teorema 8 Teorema de Shannon	Toda función representada en suma de productos tiene un equivalente en producto de sumas
Considerando $\mathbb{B} = \{0,1\}$, las propiedades se cumplen		$\forall a,b,c \in \mathbb{B}$

Muchos de los teoremas definidos se basan en el principio de dualidad (Leyes de Morgan, involución, Shannon (este ademas considerando la distributiva),...), los de la existencia de elementos neutros y absorbentes es trivial considerando la complementación y la absorción, el de cancelación se obtiene aplicando distributiva y después complementación y el del consenso es aplicar de forma correcta distributiva y complementación para simplificar o ampliar la función. No son complicados de realizar pero debe tenerse claro que se quiere obtener para poder encaminar la operación de forma correcta.

2.1.2. Representación de funciones booleanas

Veremos 3 métodos de los que disponemos para representar funciones booleanas:

- Tablas de verdad:

  Estructuras de $2^n + 1$ filas (primera fila para las variables y el resultado y las 2^n restantes para las distintas combinaciones) y $n+1$ columnas, siendo n el total de variables y la ultima columna la solución respecto a cada combinación posible. Son un método sencillo para estudiar los resultados obtenidos para cada combinación y tienen la siguiente forma:

x_1	x_2	...	x_i	$f(x_1, x_2, \ldots, x_i)$
0	0	...	0	$f(0,0,\ldots,0)$
0	0	...	1	$f(0,0,\ldots,1)$
⋮	⋮	⋮	⋮	⋮
1	1	...	1	$f(1,1,\ldots,1)$

- Representación algebraica:

  Definimos a las expresiones booleanas como variables booleanas ($a, b, c, \ldots$ a la que podremos asignar un valor en concreto dentro del conjunto $\mathbb{B}$), valores constantes (0 y 1) o la combinación de los anteriores con un operador, y serán dichas expresiones con las que representaremos algebraicamente a las funciones booleanas. Según la combinación de valores dados por las expresiones mínimas (las variables) obtenemos un valor equivalente a la función, pero cabe destacar que una misma función booleana tiene varias representaciones algebraicas, salvo que ducha función sea una expresión mínima.

  Debemos destacar dos componentes importantes en las representaciones algebraicas:

  · Maxitérmino o término suma canónico (M_i): término suma en el que intervienen todas las variables disponibles, de forma directa o negada/inversa disponibles en la función. $f(a,b,c) = \overline{a}b + \underline{a + \overline{b} + c}$

  · Minitérmino o término producto canónico(m_i): término producto en el que intervienen todas las variables disponibles, de forma directa o negada/inversa disponibles en la función. $f(a,b,c) = \overline{a} + b + \underline{a\overline{b}c}$

- Representación numérica:

  Para entender como se hace la representación numérica hay que entender que son las funciones estándar o canónica, las funciones incompletas, la conversión a binario para términos productos y términos suma y el funcionamiento del sumatorio y productorio no usuales, comencemos:

  · Función estándar o canónica: funciones formadas exclusivamente por términos canónicos (suma o producto). Podemos definir una función en la que interviene una función canónica (considerando que se mantienen el numero de variables en ambas).

  · Función incompleta: funciones cuyo resultado para las combinaciones dadas es indeterminada, es decir, una incógnita.

  · Conversión a binario: podemos interpretar los términos productos y suma como un numero binario y este a su vez como un numero decimal, veamos como:

a	b	c	d	Binario	Decimal	Equivalencia
0	0	0	0	0000_2	0_{10}	$\overline{a+b+c+d} = \overline{a}\overline{b}\overline{c}\overline{d} = 0$
0	0	0	1	0001_2	1_{10}	$\overline{a+b+c+\overline{d}} = \overline{a}\overline{b}\overline{c}d = 0$
0	0	1	0	0010_2	2_{10}	$\overline{a+b+\overline{c}+d} = \overline{a}\overline{b}c\overline{d} = 0$
⋮	⋮	⋮	⋮	⋮	⋮	⋮
1	1	1	0	1110_2	14_{10}	$\overline{a}+\overline{b}+\overline{c}+d = \overline{abc\overline{d}} = 0$
1	1	1	1	1111_2	15_{10}	$\overline{a}+\overline{b}+\overline{c}+\overline{d} = \overline{abcd} = 0$

  La columna de equivalencias esta partida por una separación de puntos para dar a entender no solo la simplificacion de las numerosas interpretaciones disponibles, también el observar que para leer las variables las consideramos siempre con valor 1 y la igualdad la obtenemos aplicando el principio de dualidad. Por ese motivo tenemos que $\overline{a+b+c+d} = \overline{a}\overline{b}\overline{c}\overline{d} \Leftrightarrow 0000_2$, considerando que las variables valen 1 en todo momento.

  · Funcionamiento del sumatorio y productorio no usuales: ya que estamos en el cuerpo $(\mathbb{B}, \oplus, \cdot)$ con $\mathbb{B} = \{0,1\}$, es normal que el sumatorio y prodcutorio no funcionen de la misma forma que en la álgebra usual. Las explicaciones pueden variar y la forma de como representarlas también (en algunos casos se usan los maxitérminos en el productorio (que correspondería al producto de términos suma canónicos o POS) y minitérminos en el sumatorio (que correspondería a la suma de términos producto canónicos o SOP) pero veremos lo más básico.

  Para simplificar el proceso de explicación, definiremos $\Xi_i := \{\sum_i, \prod_i\}$, es decir, con Ξ_i nos referiremos a ambos casos pero sabiendo que el resultado del sumatorio dará 1 y el del productorio dará 0. La expresión básica es $\Xi_i(x_1, x_2, \ldots, x_n)$ donde i sera el total de

variables que intervienen o un 0 o el vacío $\varnothing$ (para funciones incompletas, se explicara que significa a continuación) y los valores $(x_1, x_2, \ldots, x_n)$ son las representaciones en decimal de los términos producto o suma canónicos en los que se cumple que da el resultado correspondiente (términos suma $\Leftrightarrow$ productorio con resultado igual a 0 y términos producto $\Leftrightarrow$ sumatorio con resultado igual a 1). Para las funciones incompletas, tanto el productorio como sumatorio, el resultado es una incógnita. Veamos ahora como se representarían y, un ejemplo rápido de resolver una función incompleta:
$\sum_3 m(1,2,3) = m\overline{a}\overline{b}c + m\overline{a}b\overline{c} + m\overline{a}bc = 1$ con m=1
$\prod_3 M(0,4) = M(a+b+c)M(\overline{a}+b+c) = 0$ con M=0
$\sum_{\varnothing} x_i(5,6) = x_1a\overline{b}c + x_2ab\overline{c} = x$ con $x = x_1 + x_2$ y $x_i \in \{0,1\}$
$\prod_{\varnothing} x_i(5,6) = x_1(\overline{a}+b+\overline{c})x_2(\overline{a}+\overline{b}+c) = x$ con $x = x_1 + x_2$ y $x_i \in \{0,1\}$
en los casos donde coexistan funciones incompletas con estándares, habrá que despejar las x_i teniendo como premisas las funciones estándares.

2.1.3. Simplificación de funciones booleanas

1. Simplificación algebraica:

   Consiste en aplicar los axiomas y teoremas de forma correcta para obtener una expresión booleana con el menor número de términos posible, cabe destacar que puede que no sea única con lo cual, lo más cómodo, es comprobar su veracidad haciendo uso de tablas de verdad o, la mejor opción, con las tablas o mapas de Karnaugh.

2. Tablas de Karnaugh:

   Las tablas de Karnaugh nos permiten simplificar las expresiones booleanas estudiando la distribución sistemática de los valores sobre ésta. Están constituidas por $2^n + 1$ filas y $2^m + 1$ columnas, donde $n + m =$ total de variables. Por ahora aplicaremos estas tablas a expresiones con 3 ($n = 1$ y $m = 2$), 4 ($n = 2$ y $m = 2$) y 5 ($n = 2$ y $m = 3$) variables.

   La primera fila y columna están reservadas a las posibles combinaciones entre las variables ordenados según el Código Gray (dos números consecutivos difieren en una sola variable), pero la celda asignada a cada posible combinación se le asigna un decimal según el valor en binario natural (mirar imagen de ejemplo). Dicha representación decimal nos sera de ayuda para asignar los valores de las representaciones numéricas donde corresponda.

   Veamos la tabla para 5 variables con los representantes numéricos incluidos para poner un ejemplo:

ab \ cde	000	001	011	010	110	111	101	100
00	0	1	3	2	6	7	5	4
01	8	9	11	10	14	15	13	12
11	24	25	27	26	30	31	29	28
10	16	17	19	18	22	23	21	20

   Según los datos iniciales que tengamos (una tabla de verdad, una representación numérica en POS o en SOP o una representación algebraica), introducimos 1 o 0 donde corresponda. Si se nos da una tabla, bastaría con introducir el resultado respecto a los valores de la variable con las que se obtiene.

Con las SOP $\sum_i(x_1, x_2, \ldots, x_m) = x_1 + x_2 + \cdots + x_m = 1$, las celdas con valor numérico $x_1, x_2, \ldots, x_m$ serán donde se cumple la SOP y, por lo tanto, hay situado el valor 1 (si i =total de variables) o una incógnita (si $i = \varnothing$ o $i = 0$, siendo una función incompleta). Cuando se da una función incompleta, la función cambiara para despejar los posible valores de las incógnitas pero respetando las premisas y rellenando las celdas restantes con 0.

Por otro lado, las POS $\prod_i(x_1, x_2, \ldots, x_m) = x_1 \cdot x_2 \cdot \ldots \cdot x_m = 0$ podríamos considerarlo el enunciado dual de SOP, es decir, si i =total de variables las celdas con valor numérico $x_1, x_2, \ldots, x_m$ serán donde introduciremos un 0, si es $i = \varnothing$ o $i = 0$ pasaría igual. Se despejará la función incompleta, si hay, y se rellena de 1 las celdas restantes.

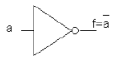
A la hora de introducir una función algebraica, lo que realmente haremos será introducir una función para que la simplifique. La tabla agrupara elementos del mismo valor en grupos, respetando una simetría, de 2^n con $n \in \mathbb{N} \cup \{0\}$. Formaremos términos producto si los grupos son respecto al valor 1 y suma si el valor es 0.

2.2. Sistemas digitales

2.2.1. Puertas lógicas digitales e implementación de funciones booleanas

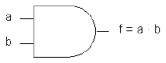
Veamos las puertas básicas junto con la expresión booleana asociada a éstas:

- Operació negació: porta NOT, inversor o negador



a	f
0	1
1	0

- Operació producte: porta AND

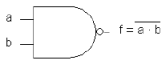


a	b	f
0	0	0
0	1	0
1	0	0
1	1	1

- Operació suma: porta OR

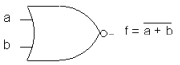
a	b	f
0	0	0
0	1	1
1	0	1
1	1	1

- Porta NAND



a	b	f
0	0	1
0	1	1
1	0	1
1	1	0

- Porta NOR

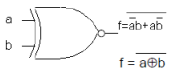


a	b	f
0	0	1
0	1	0
1	0	0
1	1	0

- Porta OR exclusiva o XOR

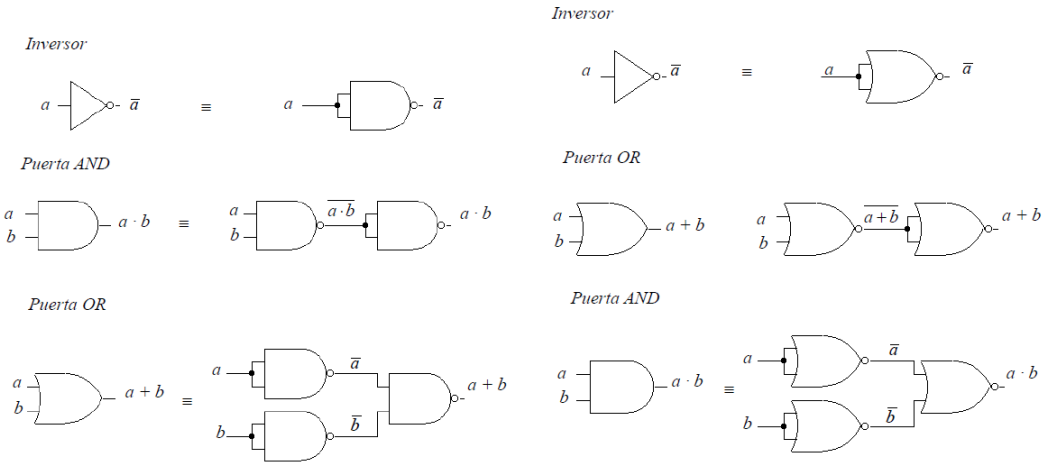
a	b	f
0	0	0
0	1	1
1	0	1
1	1	0

- Porta NOR exclusiva o XNOR



a	b	f
0	0	1
0	1	0
1	0	0
1	1	1

Con éstas definidas podemos observar como definir una misma puerta haciendo uso de otras (la puerta NAND y NOR están representadas a continuación):



Llamaremos conjunto de puertas completo al conjunto compuesto por un tipo de puerta mínimo que nos permite implementar cualquier función booleana en un circuito digital.

3. Circuitos combinacionales

Los sistemas digitales se calsifican en dos grandes grupos, diferenciados por como interacciona el tiempo sobre los valores de las variables de salida. En este tema nos centraremos en el circuito combinacional y en el siguiente veremos los sistemas secuenciales.

Lllamaremos circuito o sistema combinacional al conjunto de dispositivos lógicos en el que las salidas dependen exclusivamente del valor actual de las entradas. Podemos asignarle una función lógica y se puede formar un sistema combinación compuesto por otros, llamandose sistema combinacional integrado. Veamos algunos tipos:

3.1. Codificadores

Los codificadores pueden venir con 3 elementos adicionales ademas de las entradas y salidas, siendo estos elementos el Enable Input(EI), Enable Output (EO) y el Group Selection (GS), luego veremos su funciones pero entendamos el funcionamiento basico del codificador:

- Llamados codificadores m:n con m entradas y n salidas, clasificados en con y sin prioridad (que depende POR LO QUE PARECE de cuando varias entradas activan una misma salida, en este caso tomara la de mayor peso).
- Se tiene que cumplir que $m \leq 2^n$, si se da que $m = 2^n$, sera un codificador completo, si no se cumple sera incompleto.
- El circuito se puede construir haciendo uso de puertas OR cuyas entradas son los componentes con los que se activaria la salida.
- EI vale 1 de forma predeterminada. Si valiese 0, la salida, aunque las entradas fueran el codigo correcto, estaria desactivada. Actua como un interruptor ON/OFF del circuito.
- EO y GS muestran si hay o no una combinacion correcta que activa la salida. EO sera 1 y GS 0 si ningun codigo es correcto, y viceversa. La unica ocasion donde ambos valdran o es cuando EI es 0.

- Deben entenderse las entradas y salidas como componentes de un numero binario, es decir, el codificador leera un numero de m bits y si el las secuencia correcta te devolvera un numero de n bits correspondiente a dicho codigo.
- La prioridad, recalcando la diferencia otra vez, sera cuando un conjunto de entradas con valores similares o que se pueden agrupar (1XXX por ejemplo) denota al mismo codigo.

3.2. Decodificadores

Un decodificador es lo opuesto al codificador, dado un valor en binario en particular habra una salida dedicada a dicho valor. Existe un unico elemento adicional, un EI que actua como con el codificador.

- Dicho de otra manera, los decodificador actuan como una criba en la que para cada combinacion de las entradas, existe una unica salida (con el codificador varias entradas podian hacer referencia a la misma salida debido a que hay mas entradas que salidas).
- Llamados decodificadores m:n, se debe de cumplir que $m \geq 2^n$, y recordando que EI es un habilitador o interrumptor del circuito.
- Se puede construir un circuito decodificador usando puertas AND donde las entradas seras la secuencia correcta(p.e. $a\bar{b}$ correspondiente a S_2 con un habilitador en el circuito, las entradas de la pueta AND para esta salida serian a, (NOT)b y EI).
- Al igual que con el codificador, tratamos con los componentes de numeros binarios de distintos bits, para resolverlo tener claro cuales son las entradas y que combinacion activa que salida. La forma mas sencilla para implementarlo seria para una funcion booleana en SOP, donde los valores representado numericamente serian las posiciones o el valor correspondiente donde se deben poner las salidas.

Un ejemplo sencillo de 2 bits donde la salida corresponde directamente al valor por orden seria:

a	b	S_0	S_1	S_2	S_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

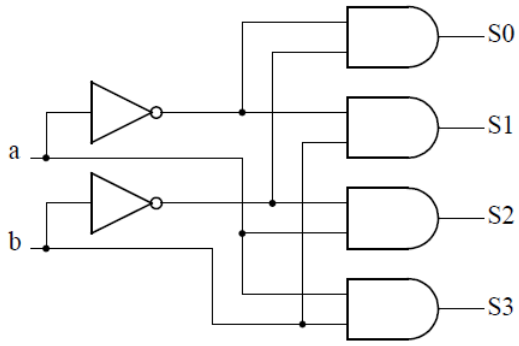
$S_0 = \bar{a}\bar{b}$

$S_1 = \bar{a}b$

$S_2 = a\bar{b}$

$S_3 = ab$

Esquema:



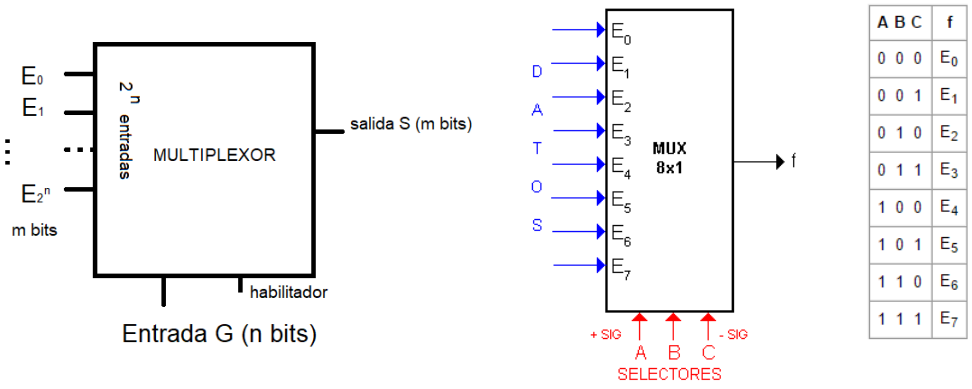
3.3. Multiplexores

El multiplexor es un circuito que permite simplificar tablas de verdad y viene caracterizado por 2^n entradas, n señales de controles o entradas de seleccion y una unica salida, ademas un habilitador EI, pero podemos entender a todo el circuito como un problema de n+1 entradas para realizar la resolucion.

- Llamados multiplexor 2^n:1, la salida tendra el valor de la entrada si se da que la señal de control para dicha posicion esta activa. La tabla a continuacion es un ejemplo de como se simplificaria un tabla de verdad de 3 valores (abc) dejandola en funcion de una de ellas.

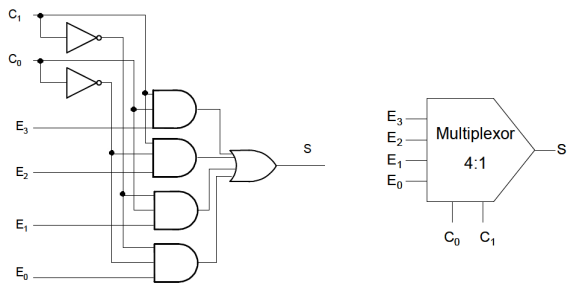
Señales de control		f(c)		f simplificada
a	b	c		f
		1	0	
0	0	0	0	0
0	1	1	0	c
1	0	0	1	$\overline{c}$
1	1	1	1	1

- Las entradas del multiplexor serian los valores de f en las posciones correspondientes a las señales de control (0 o menor peso arriba, 2^n o de mayor peso abajo). El dispositivo se veria de la siguiente forma:



A B C	f
0 0 0	E_0
0 0 1	E_1
0 1 0	E_2
0 1 1	E_3
1 0 0	E_4
1 0 1	E_5
1 1 0	E_6
1 1 1	E_7

- Se puede construir como un conjunto de codificador cuyas salidas estan conectadas a una puerta OR y las entradas de estas serian las señales de control y los habilitadores serian la entrada del multiplexor. En esta imagen se puede ver facilmente:



- Sera util para simplificar las tablas de verdad de circuitos con muchas entradas de bits. Recordar el uso del habilitador, puedes aprovechar la salida de un multiplexor para usarlo como habilitador de otro.

3.4. Desmultiplexores

El desmultiplexor es el opuesto al multiplexor, dispone de una unica entrada y n señales de controles para seleccion cual de las 2^n salidas se activara. Practicamente la idea del desmultiplexor o distribuidor es un decodificador donde el habilitador que corresponderia a la entrada del desmultiplexor y las entradas de por si serian las señales de control. Se podria plantear como un multiplexor para desarrollarlo. No creo que se pregunte pero es practicamente un decodifcador.

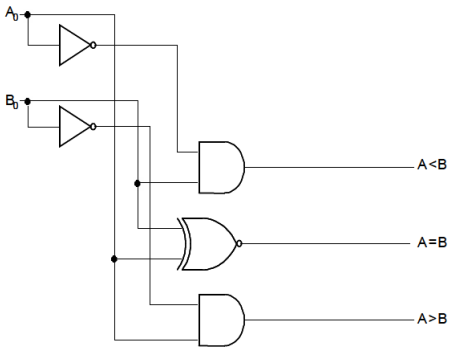
3.5. Circuitos comparadores

Un comparador es un circuito en el que se introducen dos numeros binarios de n bits ambos e ira probando si tienen igual valor los bits en la misma posicion o si uno es mayor que otro. Para probar la realacion entre los valores en la siguiente posicion bastaria con encadenarlos con el estado anterior que permita seguir avanzando (cuando valgan los mismo).

Seria complicado de dibujar pero bastaria con hacer tres ramas donde la central fuera la cadena que comprueba la igualdad de los bits en una posicion y los extremos de cada subrama la desigualdad entre estos. Puede ser abstracto de primeras pero con el ejemplo para un bit creo que sera suficiente:

A_0	B_0	$S_{A<B}$	$S_{A=B}$	$S_{A>B}$
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

$$S_{A=B}=\overline{A_0 \oplus B_0}$$
$$S_{A<B}=\overline{A}_0 \cdot B_0$$
$$S_{A>B}=A_0 \cdot \overline{B}_0$$



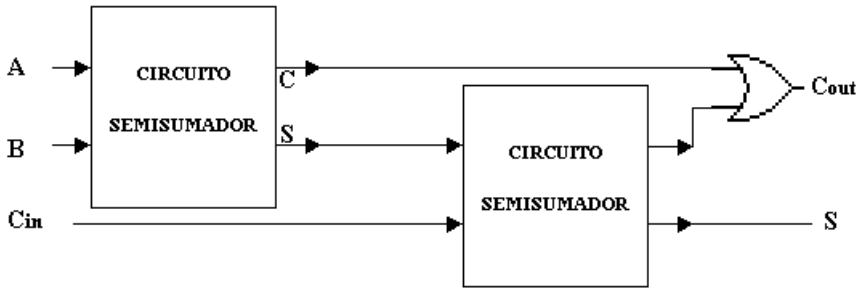
3.6. Circuitos aritmeticos

3.6.1. Sumadores

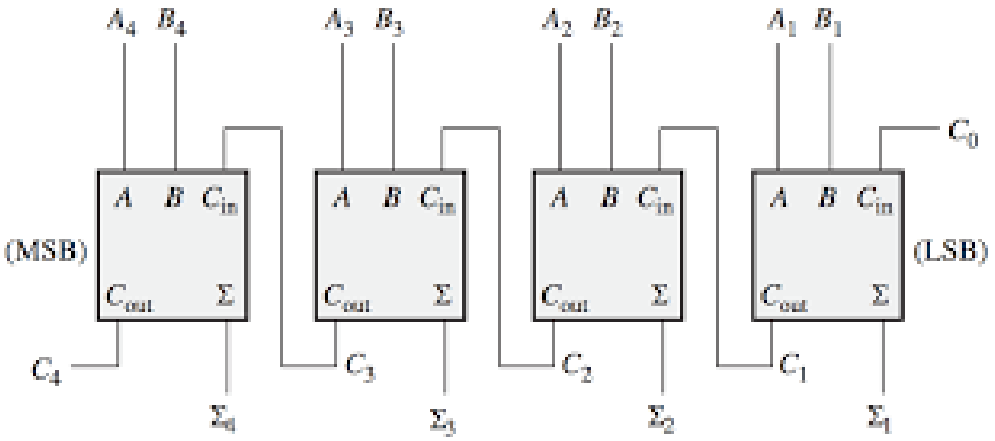
El sumador completo(considera acarreo anterior) y el semisumador(no considera directamente un acarreo anterior, solo suma dos elementos que se le de) son consecuencia directa del implementar las puertas XOR y AND (la puerta XOR equivale a $\overline{a}b + a\overline{b} = a \oplus b$ y la puerta AND a $a \cdot b$ con $a, b \in \mathcal{B}, \mathcal{B} = [0, 1]$). Con la puerta XOR realizamos la suma y con la puerta AND obtenemos el posible acarreo.

Son estas operaciones las que realizan ambos circuitos pero, en especial, el sumador completo que podemos entenderlo como dos semisumadores.

$$S = A \oplus B \oplus C_{in} = (\overline{A}B + A\overline{B}) \oplus C_{in} = \overline{A}\overline{B}C_{in} + \overline{A}B\overline{C}_{in} + A\overline{B}\overline{C}_{in} + ABC_{in}$$
$$C_{out} = AB + AC_{in} + BC_{in}$$



En la imagen anterior observamos como la suma $A+B$ acompañada por un posible acarreo (C_{in})de un etapa anterior aporta como resultado final un valor S y otro C_{out}, pero debemos comprender que trabaja con 1 bit, si se desease con más lo que realmente haria sería una sucesión de sumadores completos. Veamos un ejemplo con 4 bits.

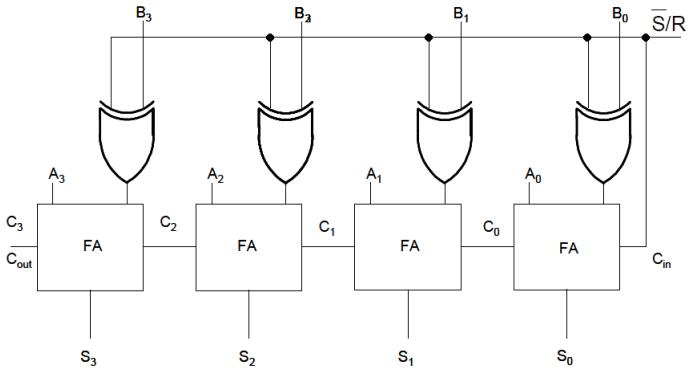


En este ejemplo vemos las suma de $A + B$ acompañado por un posible acarreo C_0, siendo estas nuestras entradas, cuya salida nos dará el acarreo de la ultima etapa (C_4) y la suma, $\sum$.

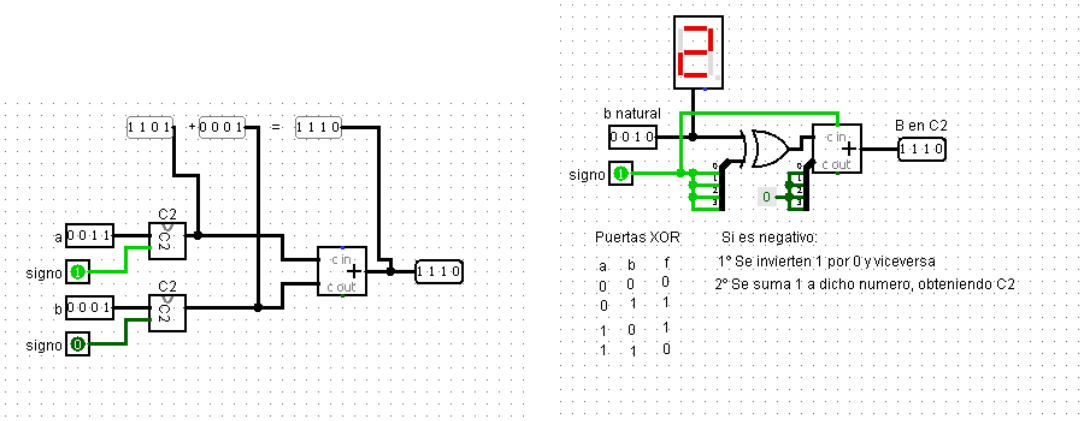
3.6.2. Restadores

Un restador completo y un semirestador, cuya diferencia es la misma que la de los sumadores, siguen los mismos principios que el sumador, lo unico que cambia es el acarreo que en lugar de ser $C_{out+} = ab$ como en el sumador (situacion en la que la suma genera escedente), sera $C_{out-} = \overline{a}b$ (situacion en la que el positivo es menor que el negativo en valores de la misma posicion y se aplica la resta al valor de la siguiente posicion.

Tambien, podemos considerar las propiedas de los complemento a 1 y a 2 para entender la resta como una suma. Recordad que un numero en C1 es el opuesto del binario natural negativo y el C2 seria el complemento a 1 +1. Con el sumador se veria como las entradas del numero negativo con puertas NOT y un acarreo con valor a 1 si fuese con C2. Es mas, en el siguiente ejemplo se observa como se utilizan puertas XOR y un activador S/R para indicar que funcion se realiza (usado en C2, solo considera el segundo numero posible de ser negativo):



Este seria un ejemplo realizado por logisim de un sumador que considera si un numero es negativo o positivo y los suma.



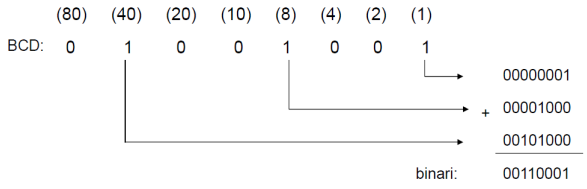
3.7. Más sistemas combinacionles (dado en valenciano)

Estos conceptos se explicara por encima ya que solo los ha introducido una profesora sustituta y no esta en los apuntes que se nos dio al principio, son faciles de sacar con tablas de verdad y los circuitos anteriormente vistos ademas de ciertas propiedades, veamoslos:

3.7.1. Conversores de código

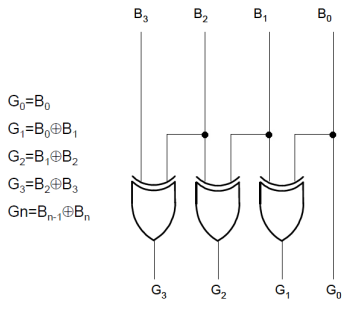
Dsipositivo que permite pasar de un tipo de binario a otro (recordemos que esta el binario natural, el BCD, el codigo Gray.. vistos en el 1.4).

- De natural a BCD solo consistira en tener en cuenta que el BCD son grupos de 4 bits cuyo valor numerico no supera el 9. Si realizamos un comprobador de valor superior a 9 y que este active dos cosas, un 0001 que sera el bit de mayor peso y un sumador de 6 (ya que es el equivalente de 9 en C1).
- De BCD a natural creo yo que con este ejemplo bastaria para entenderlo:

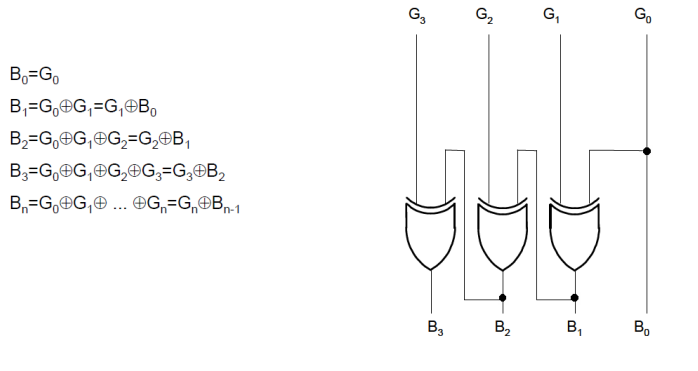


· Binario natural-Gray y Gray-Binario natural

binario-gray



gray-binario



3.7.2. Generadores/detectores de paridad

PAAAAAAAAAAAAAAASO

3.7.3. Hoja de caracteristcas (data sheet)

Es teoria, un Data sheet es un documente que explica detealladamente el funcionamiento y caracteristicas de un dispositivo.

4. Sistemas secuenciales

Los sistemas digitales cuyas salidas no dependan unicamente de las entradas en ese instante, tambien de los valores en instantes anteriores reciben el nombre de sistemas secuenciales. Se estudian dos modelos, el de Mealy y el de Moore, pero antes debemos conocer algunos dispositivos de almacenamiento de memoria:

4.1. Biestables

Circuitos lógicos en los que se distinguen dos estados estables los cuales pueden mantenerse indefinidamente aun habiendo desaparecido la señal que activa el estado. Los podemos clasificar por funcionalidad o por la forma de su activación (asíncronos, es decir, sin señal de reloj o sincrono, depende de una señal de reloj y hay por nivel o por flanco).

Diremos que es de nivel alto cuando se encuentra con valor 1, nivel bajo cuando tiene valor 0, flanco de subida en el momento exacto que se desplaza de 0 a 1 y flanco de bajada en el momento exacto que se desplaza de 1 a 0. Esta terminología también se puede aplicar a otros elementos pero sera mas frecuente con las señales de reloj. Para hacer por nivel bastaria con recibir una señal constante (las entradas dependen del valor del reloj) pero por flanco se tendria que realizar un conjunto de puertas logicas en la que la señal de reloj de una valor en una pequeña fraccion de tiempo (se vera un ejemplo mas adelante).

Por funcionalidad distinguimos los biestables RS, JK, D y T:

4.1.1. Biestable RS (reset y set)

Sin importar si es sincrono o asíncrono o sincrono con entradas asíncronas, este circuito actúa como un Reset-Set, es decir, la entrada R resetea la salida Q y la entrada S únicamente la puede activar.

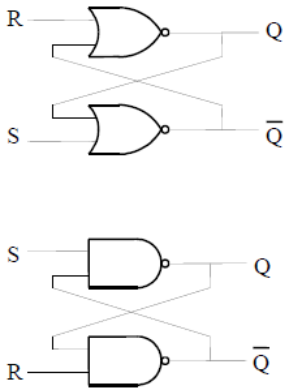
Que estén ambas encendidas no esta permitida y si están las dos apagadas la salida es la que estaba en el estado anterior. Se ha nombrado el caso de que sea sincrono con estadas asíncronas, éste es cuando se usan las entradas P y C (Preset y Clear), siendo la función del Preset la de activar la salida Q y la del Clear es despejar la salida dándole el valor de 0. La función de estos elementos es darle un valor a la salida directamente sin tener que dar una pulsación de reloj.

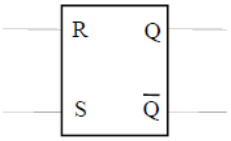
R	S	Q(t+1)	Acción
0	0	Q(t)	No cambia
0	1	1	Se activa
1	0	0	Se resetea
1	1	-	No definida

Prst	Clr	R	S	Q(t)	Acción
0	0	0	0	Q(t-1)	No cambia
0	0	0	1	1	Set
0	0	1	0	0	Reset
0	0	1	1	-	No definido
1	0	x	x	1	Activa la salida
0	1	x	x	0	Despeja la salida
1	1	x	x	0	No definida/Clear tiene preferencia

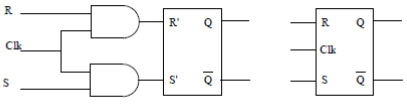
Se puede escribir de las siguiente formas: (meter fotos del tema)

· Asincrono con puertas NOR y con puertas NAND

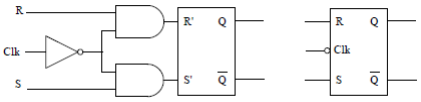




· Sincrono por nivel con puerta AND y usando asincrono.

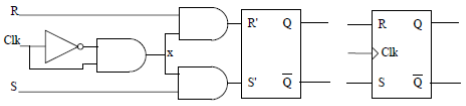


Nivel alto

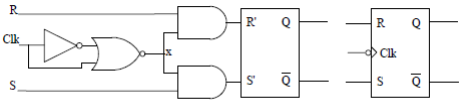


Nivel bajo

· Sincrono por flanco con puerta AND y usando asincrono.

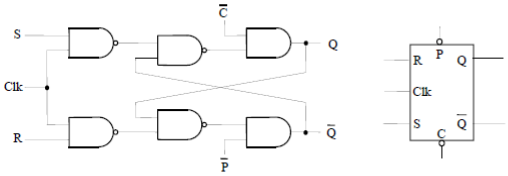


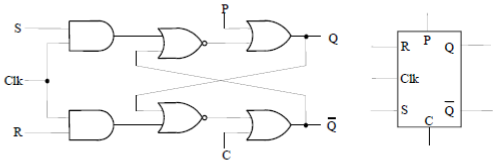
Flanco de subida



Flanco de bajada

· Con entradas asincronas (por nivel bajo y alto, respectivamente)





Se pueden darse mas formas pero estos son ejemplos .

NOTA: recordad que si usais todo puertas NAND con los sincronos, la salida de la puerta NAND sera R y S negadas para una señal de reloj de nivel alto.

4.1.2. Biestable JK

Este biestable con entradas J y K actúa de tal forma que si únicamente se activa K, Q se resetea, si solo se activa J, Q se activa, si J y K no se activan Q se mantiene en el estado anterior y si ambas entradas se activan entonces la salida sera la opuesta al estado anterior (basculación), tambien consideraremos entradas de Preset y Clear.

Aunque aparentemente utiliza señales de reloj, solo las utilizas para considerar los estados... mas adelante se vera como, construido con un biestable RS, la señal de reloj da igual si es de nivel alto o bajo que no afectara realmente a las entradas.

Debemos considerar dos tablas de verdad que hacer, el funcionamiento del biestable JK y la interacción de las entradas Preset y Clear:

J	K	Q(t+1)	Acción
0	0	Q(t)	No cambia
0	1	0	Reset
1	0	1	Set
1	1	$\overline{Q(t)}$	Basculación

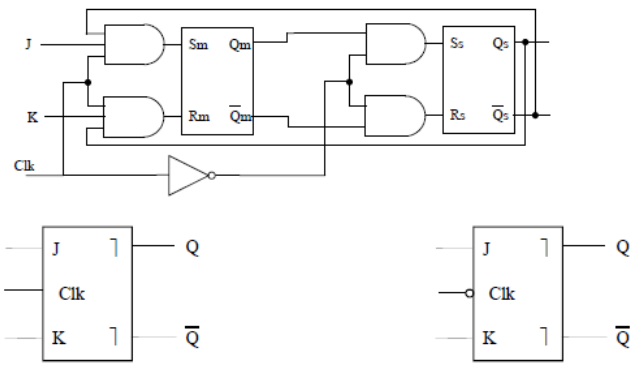
Funcionamiento del biestable JK sin uso alguno de las entradas prst y clr

Prst	Clr	Q(t)
0	0	Q(t)
1	0	1
0	1	0
1	1	0

En este caso, Q(t) se sobrescribe ya que no depende de la señal del reloj y si $Prst = Clr = 1$ entonces parece tener preferencia Clr

Prst	Clr	J	K	Q(t)	Acción
0	0	0	0	Q(t-1)	No cambia
0	0	0	1	0	Reset
0	0	1	0	1	Set
0	0	1	1	$\overline{Q(t-1)}$	Basculación
1	0	x	x	1	Activa la salida
0	1	x	x	0	Despeja la salida
1	1	x	x	0	No definida/Clear tiene preferencia

En esta imagen observamos como construimos un biestable JK a partir de uno RS, con la configuracion llamada Master-Slave (Master hace las cosas y Slave las copia).



4.1.3. Biestable D (Data)

Con una sola entrada D (Dato) ademas de la señal de reloj, la acción de este biestable es almacenar brevemente la información aportada por la entrada D. Prácticamente repite la información dada.

Se podría construir con un biestable JK siendo D la entrada J y $\overline{D}$ la entrada K.

J	K	Q(t+1)	Acción
0	0	Q(t)	No cambia
0	1	0	Reset
1	0	1	Set
1	1	$\overline{Q(t)}$	Basculación

D	$\overline{D}$	Q(t+1)	Acción
0	0	-	No se da
0	1	0	Reset
1	0	1	Set
1	1	-	No se da

D	Q(t+1)	Acción
0	0	Elimina
1	1	Guarda

4.1.4. Biestable T (Toggle)

Usando una sola entrada, T (Toggle), junto con las señales de reloj, la función de este biestable consiste en alternar la salida si la entrada T se activa, recordando el estado actual.

Usando un biestable JK, sin consideramos las entradas J y K como la entrada T, podemos construir este elemento.

J	K	Q(t+1)	Acción
0	0	Q(t)	No cambia
0	1	0	Reset
1	0	1	Set
1	1	$\overline{Q(t)}$	Basculación

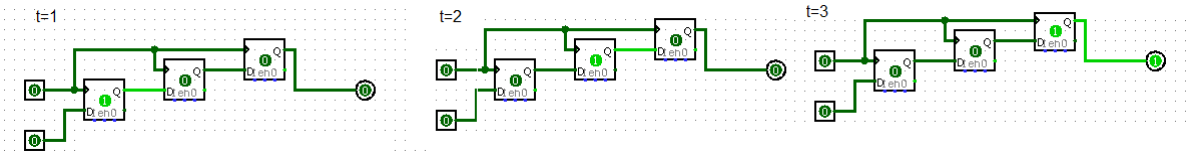
T	T	Q(t+1)	Acción
0	0	Q(t)	No cambia
0	1	-	No se da
1	0	-	No se da
1	1	$\overline{Q(t)}$	Basculación

T	Q(t+1)	Acción
0	Q(t)	No cambia
1	$\overline{Q(t)}$	Cambia

4.2. Registros y contadores

Existen otros elementos como los registros (circuitos secuenciales con funciones de almacenamiento o desplazamiento de información) y contadores (sistema secuencial que memoriza el numero de pulsaciones aplicadas a una entrada de reloj), veamoslos por encima:

- Registro: Almacenan o desplazan datos por unidad de tiempo, se pueden construit con biestables D en serie, teniendo tantos como unidades de tiempo que quieres que se mantenga el datos ya que todos ellos estaran conectados a la misma señal. En este ejemplo por logisim vemos como se almacena un dato durante tres señales de reloj



- Contadores: Son circuitos que cuentan el numero de pulsaciones de una señal de reloj, se pueden clasificar en asincronos (muy faciles) o sincronos (no veo ningun ejemplo so sudo). El asincrono utiliza biestable JK con ambas entradas con valor 1 constante conectadas en cascada, donde la señal de reloj sera la salida del biestable anterior salvo el primero, que sera la propia señal de reloj. Con n biestable podemos contar hasta 2^n ya que la salida de cada biestable seran los bits de menor a mayor peso que genere el numero de pulsaciones.

4.3. Diseño de sistemas secuenciales, modelos de Moore y de Mealy

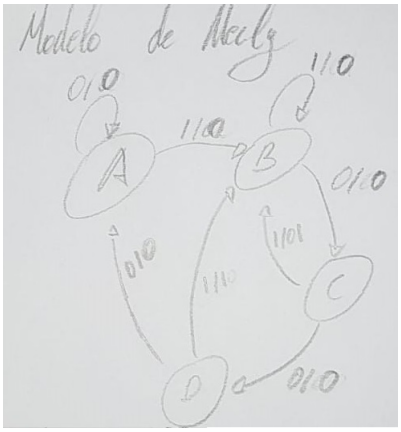
Con esto visto, podemos ver los distintos modelos de sistemas secuenciales delos que disponemos, el modelo de Mealy y el de Moore. La principal diferencia es de que depende la salida del circuito, si esta depende del estado anterior y de las entradas hablamos del modelo de Mealy, si solo depende del estado anterior entonces hablamos de Moore. Veamos como obtener la resolucion de un problema aplicando ambos modelos, ya que hay un modelo de Moore equivalente a uno de Mealy y viceversa, y siguiendo las siguientes pautas:

1. Obtener un diagrama de estado y una tabla de estados partiendo de las especificaciones del problema, es decir, distinguimos que posibles estados se crearan y que provoca que se cambie de uno a otro.
2. Codificamos los estados para obtener una tabla de transición de estados y de salida.
3. Seleccionar el biestable adecuado y realizar una tabla de excitactión del biestable a partir de la tabla de transición de estado.
4. Obtener las ecuaciones de entrada de los biestables y las ecuaciones de salida del circuito a partir de estas tablas, ésto se puede realizar usando Karnaugh.
5. Dibujar el circuito usando las entradas, salidas y biestable que hayas obtenido.

Veamos el siguiente ejercicio, donde se nos pide que se encienda una bombilla para cuando entra una secuencia de datos en particular (1001) permitiendo que exista solapamiento, es decir, aprovecha el ultimo 1 de la secuencia para iniciar la siguiente.

Veamoslo primero por Mealy y por pasos indicados anteriormente:

1. Obtener un diagrama de estado y una tabla de estados partiendo de las especificaciones del problema. Con el modelo de Mealy, habra que considerar que lo que conecta los estados no es unicamente la entrada, sino la salida.



Z_0	Accion
0	Bombilla apagada
1	Bombilla encendida

Estado	Explicacion
A	Estado inicial o cuando se cancela la secuencia con x:0
B	Se inicia la secuencia, X:1 reinicia y X:0 continua
C	Primer cero, X:1 reinicia y X:0 continua
D	Segundo cero, X:1 fin secuencia y X:0 cancela

Estado anterior	Estado nuevo en funcion de x	
	x=0	x=1
A	A,0	B,0
B	C,0	B,0
C	D,0	B,0
D	A,0	B,1

con X,Z_1Z_0, X estado nuevo y Z_i las salidas segun la entrada y X.

2. Codificamos los estados para obtener una tabla de transición de estados y de salida. Con el modelo de Mealy, estas tablas estan unidas.

Estado anterior	Estado nuevo en funcion de x	
	x=0	x=1
A=00	00,0	01,0
B=01	11,0	01,0
C=11	10,0	01,0
D=10	00,0	01,1

3. Seleccionar el biestable adecuado y realizar una tabla de excitactión del biestable a partir de la tabla de transición de estado. Lo mas comun es usar como elemento de memoria el biestable JK, ya que puede actuar como todos los demas.

   Se usaran tantos biestables como numero de bits hayan para distinguir los estados codificados, en este caso 2 bits. En este caso, es mejor hacer la tabla de excitacion del biestable considerando tambien el estado anterior tal que

J	K	q	Q	Acción
0	0	0	0	No cambia
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	1	Basculación
1	1	1	1	

$\Leftrightarrow$

J	K	q	Q	Acción
0	X	0	0	No cambia o Reset, depende de J
1	X	0	1	Set o basculacion, depende de J
X	0	1	1	No cambia o Set, depende de K
X	1	1	0	Reset o basculacion, depende de K

   Aunque parezca que dependa de la incognita, realmente depende del valor constante, ya que da igual el valor que tenga la incognita, lo que importa es el valor de la entrada que sea constante si queremos obtener Q partiendo de q. Esto es importante para entender el siguiente paso.

4. Obtener las ecuaciones de entrada de los biestables y las ecuaciones de salida del circuito a partir de estas tablas, ésto se puede realizar usando Karnaugh. Es el paso mas complicado, ya que se deben de considerar que elementos usar de entrada en un circuito combinacional para obtener la salida adecuada.

   Como hemos dicho antes, tendremos 2 biestables JK, con lo cual habra que considerar 4 entradas a realizar: J_0, K_0, J_1 y K_1. Debemos hacer que para cada una de las entradas entre la correcta combinacion de entradas y estados anteriores que de el siguiente estado, veamos como seria para este caso:

q_1q_0\X	0	1
00	0	0
01	1	0
11	X	X
10	X	X

J_1

Nos fijamos en $\overline{q_1}$

q_1q_0\X	0	1
00	X	X
01	X	X
11	0	1
10	1	1

K_1

Nos fijamos en q_1

q_1q_0\X	0	1
00	0	1
01	X	X
11	X	X
10	0	1

J_0

Nos fijamos en $\overline{q_0}$

q_1q_0\X	0	1
00	X	X
01	0	0
11	1	0
10	X	X

K_0

Nos fijamos en q_0

   Nos fijamos en las regiones en las que las entradas JK dara la salida correspondiente a su peso que deseamos, por ejemplo, estando en 01 y entra x=0, pasariamos al estado 11. La salida $Q_0 = 1$ depende de $q_0 = 1$ y x=0 (No hay cambio o se hace Set) y la salida $Q_1 = 1$ depende de $q_1 = 0$ y x=0 (se produce una basculacion o un Set). Mirando la tabla auxiliar del biestable vemos que cuan q=0, nos fijamos en la salida deseada para poner J, y si q=1 nos fijaremos en K.

En ese caso, con $q_0 = 1$ y x=0 y sabiendo por la tabla de transicion que $Q_0 = 1$, nos fijaremos en que valor de K provoca esta funcion, siendo K=0. Ahora, con $q_1 = 0$ y x=0 y sabiendo que $Q_1 = 1$, que valor de J provoca esta funcion, siendo J=1. Repitiendo este proceso obtenemos las tablas de excitacion de los biestables, que despejando por Karnaugh obtendremos las ecuaciones de entrada que son:

$$J_1 = q_0\overline{x} \quad K_1 = x + \overline{q_0 x}$$
$$J_0 = x \quad K_0 = q_1\overline{x}$$

Para obtener las de salida debemos considerar la tabla de transicion de estados al principio (en el caso de modelo de Mealy). Pero podemos si no lo vemos claro hacer una tabla para comprobar las salidas, siendo la de transicion pero omitiendo los estados nuevos y si se prefiere considerar una tabla por cada bit, tal que:

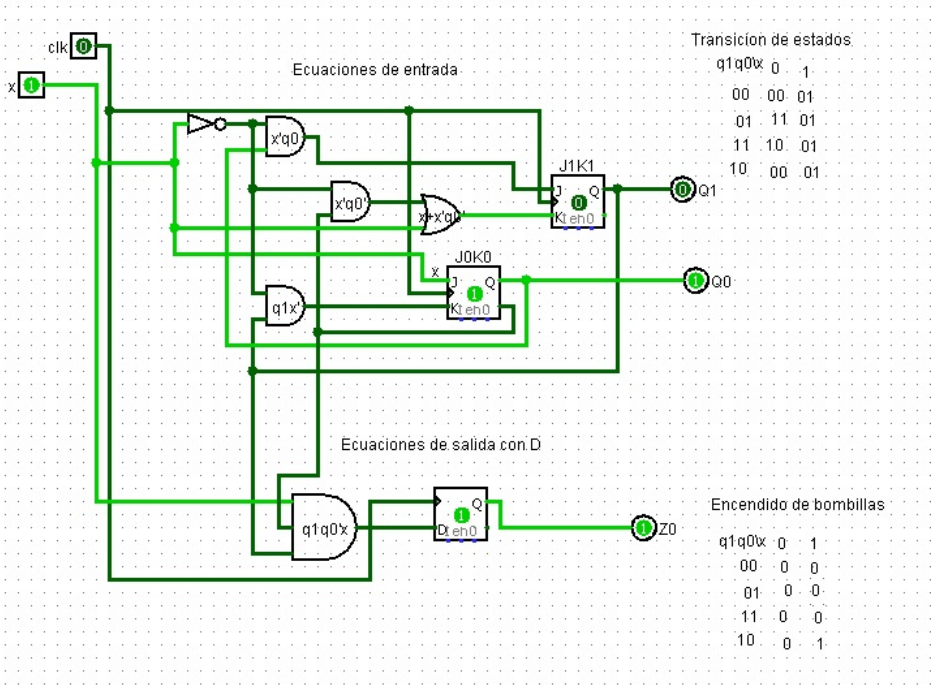
Estado anterior	Salida para Z_0	
	x=0	x=1
A=00	0	0
B=01	0	0
C=11	0	0
D=10	0	1

Y facilmente por Karnaugh despejamos: $Z_0 = q_1\overline{q_0}x$

5. Dibujar el circuito usando las entradas, salidas y biestable que hayas obtenido.

   Facil de obtener en cuanto se tienen las ecuaciones, recordad que sumas son puertas OR y multiplicaciones puertas AND, siendo estas las mas sencillas, puertas XOR de utilidad.

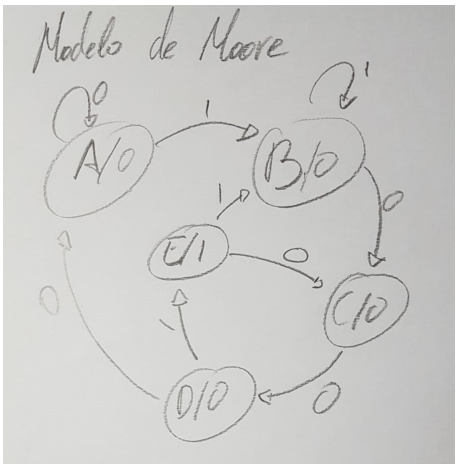
   En ocasiones se pueden dar problemas de sincronizacion debido a que depende de la entrada y esta no va a la par con el pulso, con lo cual es posible que se necesita añadir un biestable D para desplazar la x. Sinceramente, esto cuesta verlo y mas por escrito, con lo cual no creo que os vayan a matar mucho por ello.



La bombilla permanecera encendida cuando, estando en el estado D=10, recibe un x=1, no se de la siguiente señal de reloj. Si no estuviese el biestable D, se encenderia un instante que seria justo el instante en el que recibe las condiciones necesarias.

Veamoslo ahora para el modelo de Moore:

1. Obtener un diagrama de estado y una tabla de estados partiendo de las especificaciones del problema. Con el modelo de Moore, cada estado tiene una salida asociada.



Z_0	Accion
0	Bombilla apagada
1	Bombilla encendida

Estado	Explicacion
A	Estado inicial o cuando se cancela la secuencia con x:0
B	Se inicia la secuencia, X:1 reinicia y X:0 continua
C	Primer cero, X:1 reinicia y X:0 continua
D	Segundo cero, X:1 fin secuencia y X:0 cancela
E	Fin secuencia, X:1 reinicia y X:0 continua

Estado anterior	Salida correspondiente
A	0
B	0
C	0
D	0
E	1

Estado nuevo en funcion de x	
x=0	x=1
A	B
C	B
D	B
A	E
C	B

2. Codificamos los estados para obtener una tabla de transición de estados y de salida. Con el modelo de Moore, seran dos tablas distintas que parten de la tabla anterior.

Estado anterior	Salida correspondiente
A=000	0
B=001	0
C=011	0
D=010	0
E=110	1

	Estado nuevo en funcion de x	
	x=0	x=1
000	000	001
001	011	001
011	010	001
010	000	110
110	011	001

Con Mealy teniamos un estado menos que era justo $2^2 = 4$ estados, los que podiamos clasificar con las combinaciones de 2 bits, pero en este caso, con Moore, tenemos 5 con lo cual tendremos que coger 3 bits, que nos dan hasta $2^3 = 8$ combinaciones.

3. Seleccionar el biestable adecuado y realizar una tabla de excitactión del biestable a partir de la tabla de transición de estado. Lo mas comun es usar como elemento de memoria el biestable JK, ya que puede actuar como todos los demas.

   Como se ha dicho durante la explicacion de Mealy, usaremos tantos biestables como bits hayan para distinguir los estados, es decir, 3 bits y por lo tanto usamos 3 biestables, cada uno para los bits de entrada del mismo peso o que estan en la misma posicion que los de salida.

   Recordemos de nuevo la tabla del biestable JK (que puede actuar como todos) considerando q y constantes:

J	K	q	Q	Acción
0	0	0	0	No cambia
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	1	Basculación
1	1	1	1	

$\Leftrightarrow$

J	K	q	Q	Acción
0	X	0	0	No cambia o Reset, depende de J
1	X	0	1	Set o basculacion, depende de J
X	0	1	1	No cambia o Set, depende de K
X	1	1	0	Reset o basculacion, depende de K

Sera comodo escribir dos tablas para facilitar el siguiente paso:

J	q	Q	Acción
0	0	0	No cambia o Reset
1	0	1	Set o basculacion

K	q	Q	Acción
0	1	1	No cambia o Set
1	1	0	Reset o basculacion

4. Obtener las ecuaciones de entrada de los biestables y las ecuaciones de salida del circuito a partir de estas tablas, ésto se puede realizar usando Karnaugh. Es el paso mas complicado, ya que se deben de considerar que elementos usar de entrada en un circuito combinacional para obtener la salida adecuada.

   Veamos ahora las ecuaciones de entradas para los tres biestables JK que usaremos, teniendo en cuenta que los estados 111, 101 y 100 no estan definidos:

$q_2q_1q_0$\X	0	1
000	0	0
001	0	0
011	0	0
010	0	1
110	X	X
111	X	X
101	X	X
100	X	X

J_2

Nos fijamos en $\overline{q_2}$

$q_2q_1q_0$\X	0	1
000	X	X
001	X	X
011	X	X
010	X	X
110	1	1
111	X	X
101	X	X
100	X	X

K_2

Nos fijamos en q_2

$q_2q_1q_0$\X	0	1
000	0	0
001	1	0
011	X	X
010	X	X
110	X	X
111	X	X
101	X	X
100	X	X

J_1

Nos fijamos en $\overline{q_1}$

$q_2q_1q_0$\X	0	1
000	X	X
001	X	X
011	0	1
010	1	0
110	0	1
111	X	X
101	X	X
100	X	X

K_1

Nos fijamos en q_1

$q_2q_1q_0$\X	0	1
000	0	1
001	X	X
011	X	X
010	0	0
110	1	1
111	X	X
101	X	X
100	X	X

J_0

Nos fijamos en $\overline{q_0}$

$q_2q_1q_0$\X	0	1
000	X	X
001	0	0
011	1	0
010	X	X
110	X	X
111	X	X
101	X	X
100	X	X

K_0

Nos fijamos en q_0

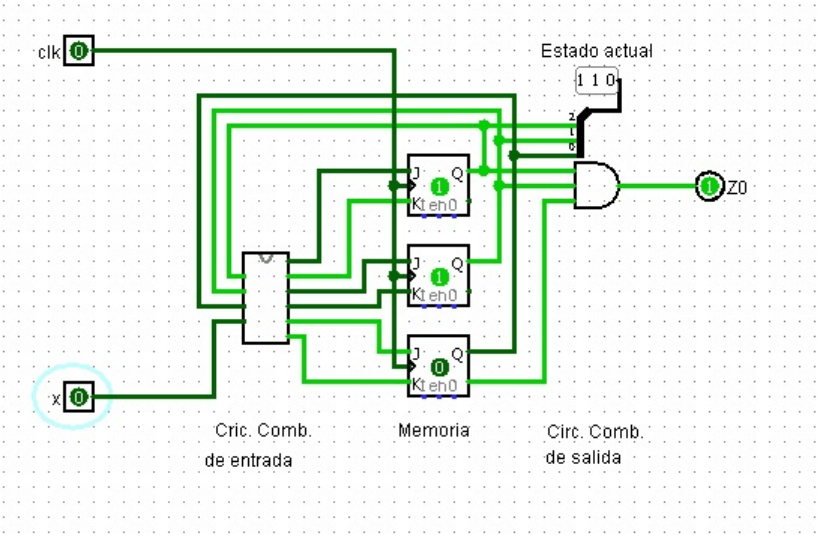
   Veamos las ecuaciones de entrada que se obtienen de realizar Karnaugh(que por cierto, la tabla de Karnaugh no corresponde a las mostradas pero se sacaria facilmente tanto la tabla como las operaciones en estas usando el mismo principio) sobre estas tablas:

$$J_2 = q_1\overline{q_0}x \qquad J_1 = q_0\overline{x} \qquad J_0 = q_2 + \overline{q_1}x$$
$$K_2 = 1 \qquad K_1 = \overline{q_2q_0x} + q_0x + q_2x \qquad K_0 = q_1\overline{x}$$

   Para obtener las salidas nos fijaremos en la tabla anteriormente de salidas para cada estado, pero en este caso en sencillo ya que hay un estado exclusivo para activar la salida, siendo $Z_0 = E = q_2q_1\overline{q_0}$

5. Dibujar el circuito usando las entradas, salidas y biestable que hayas obtenido.

   Al igual que con Mealy, usamos las ecuaciones para definir puertas logicas para cada ecuacion, en este caso al ser mas numeros resumimos el circuito combinacional pero el contenido es el mismo.



   En este caso, no hace un biestable para controlar la salida ya que este viene dado por los estados que se obtiene al dar una señal de reloj.

5. Recomendaciones

Pues ya mas no se que poner, si a alguno se le ocurre alguna cosa de ultima hora la anoto:

· Recordad los trucos para pasar de dinario natural a los complementos y SM, la representacion de numero reales era algo mas complicada con el sesgo pero en el tema 1 esta bien explicado.

· Una resta se puede convertir en suma usando complementos.

· Codigo Gray lo usamos para las tablas de karnaugh, BCD es agrupar de 9 en 9(en grupos de 4 bits que no superar 1001).

· Revisar las reglas logicas del tema 2, viene muy bien para simplificar cuando por Karnaugh no lo ves

· Memorizar JK y tener claro que J=D=T y K=$\overline{D}$=T, todos los biestables estan relacionados con el, RS es un componente del JK.

· Las tablas de excitacion estan relacionadas directamente con la posicion del bit de salida y de entrada y su valor, si q=0 trabajas con J, si q=1 con K y segun la salida que quieras obtener para Q, J y K tomaran valores distintos (ej. en desarrollo por Moore).

Si se quieren los archivos de logisim habladme @eduespuch (y seguirme por insta y todo eso)