

BASES NUMERICAS

Existen con el propósito de contar eventos, el análisis académico que hacemos de las mismas es con el ánimo de apoyar el conocimiento que necesitamos adquirir en el curso con la introducción de la electrónica digital y su base asociada, la “base binaria”.

Bases a analizar:

- **Base Decimal:** Compuesta por 10 símbolos diferentes de conteo de 0 a 9. Importante debido a nuestra relación nodriza y el interfazamiento con la computadora que trabaja en otra, la “binaria”
- **Base Hexadecimal:** Compuesta por 16 símbolos diferentes de conteo de los arábigos 0 a 9; y de la letra “A” a la letra “F” para acumulación de conteo de la cantidad arábica 10 a 15 respectivamente, observe que la inclusión del 0 en los símbolos que componen a todas las bases no permite que las mismas cuenten con el símbolo máximo de conteo que las identifican semánticamente. Esta base es incluida en nuestro curso, debido a que con los símbolos que la componen se pueden articular cadenas binarias, y en cantidades extensas de números en base binaria podemos poseer cantidades en base hexadecimal relativamente pequeñas que las puedan representar con total equivalencia (diferente forma igual capacidad de conteo). Basada en esta utilidad decimos que: la base hexadecimal es una base “vehículo” o “transportadora”, en computadoras con CPU’s de hasta 32 bits esta base ha sido útil, con el advenimiento de los CPU de 64 bits, la situación apunta a utilizar una base de mayor capacidad de compactación, “transporte” o manejo

En base hexadecimal cada dígito que la compone se articula con 4 dígitos en base binaria y viceversa

- **Base Octal:** Compuesta por 8 símbolos diferentes de conteo de los arábigos 0 a 7. En la actualidad ya en desuso debido a que ella compacta y transporta cadenas binarias de 3 dígitos por cada dígito en Octal y en la actualidad los CPU’s ya superaron con creces la frontera de 4 bits de longitud de datos, que fue con la cual aún trabajaron óptimamente.
- **Base Binaria:** Compuesta por los símbolos arábigos “0” & “1”. Protagonista en el apoyo de la electrónica digital, seleccionada básicamente por la necesidad de crear, mantener y optimizar circuitos eléctricos con alta inmunidad al ruido, debido a que la existencia de dos símbolos “0” & “1” se relacionan con dos rangos de tensiones que poseen una ventana de discriminación (reconocimiento) amplia y una barra de tensión que las separa, de esta forma el daño que el ruido ocasiona por corrupción es bajo en comparación con la elección de otra base.

Parámetros utilizados para analizar bases:

- **Columnas de Ponderación:** son columnas imaginarias que separan a cada uno de los dígitos que componen a una cantidad en cualquier base, decimos que la columna que enmarca al primer dígito (vista la cantidad de frente de derecha a izquierda) es la columna de menor ponderación o columna “0”; ellas son importantes en la apreciación del orden que poseen los dígitos en una cantidad
- **Peso Ponderado:** es la capacidad de conteo que posee un dígito en una cantidad por la posición que ocupa dentro de ella, o por la columna de ponderación donde se ubica, de aquí proviene la definición de unidades, decenas, centenas, etc. que se maneja en base decimal.
- **Contaje Acumulado:** es razón total de conteo de eventos que una cantidad dentro de una base posee de acuerdo con los dígitos que la componen y la ubicación de estos dentro de la misma

Aparentemente estos parámetros no son importantes, pero resultan ser fundamentales cuando ejecutamos traslaciones entre bases, la traslación entre bases es necesaria debido al interfazamiento final y se debe entender que, aunque diferentes cantidades en una u otra base sean diferentes ellas pueden mostrar el mismo contaje acumulado, a esto lo llamamos equivalencia.

NOTA: Comúnmente el dígito en base binaria ubicado en la columna de ponderación cero recibe el nombre de “LSB” y el ubicado en la columna de mayor ponderación “MSB”

Traslación entre bases o equivalencia entre bases:

La manipulación de información en lógica o electrónica digitales nos obliga a conocer métodos para traslación entre cantidades de diferentes bases.

En el libro de texto y en general en cualquier libro de electrónica digital se puede encontrar los métodos a utilizar para lograr la traslación

A.1) Ejemplo de traslación de binario a decimal:

Trasladar el número binario: 11101 a decimal.

- Separamos los diferentes dígitos de la cantidad en binario, buscando el peso ponderado de cada uno respecto de su columna de ponderación y sumando peso ponderado de acuerdo con coeficiente logramos el decimal, así:

	1	1	1	0	1
Columna ponderación	4	3	2	1	0
	2^4	2^3	2^2	2^1	2^0
Peso ponderado	16	8	4	2	1
Aporte a conteo acumulado	1x16	1x8	1x4	0x2	1x1
Equivalente decimal:	16 +	8 +	4 +	0 +	1 = 29 (decimal)

Interpretación: la cantidad “29” en decimal me indica 29 eventos contados acumulados uno a uno, IGUAL la cantidad “11101” en binario me indica 29 eventos contados acumulados uno a uno

A.2) Ejemplo de traslación de decimal a binario:

Trasladar el decimal 39 a binario

- Describir si el decimal es par o impar, esto decidirá si el primer dígito de la cantidad equivalente en binario “el de la columna de ponderación 0” es uno o cero, luego buscamos uno a uno yendo de derecha a izquierda, el peso ponderado por columna de ponderación, hasta superar a la cantidad en decimal, ese dígito ya no formara parte de la cantidad buscada en binario, por fin en función de sumas que arrojen el decimal exacto definimos los coeficientes “0’s” o “1’s”
 - 39 es impar, entonces: el dígito en la columna de ponderación 0 es “1”
 - Avanzamos de derecha a izquierda verificando peso ponderado y encontrar así el número de dígitos:

64 X	32	16	8	4	2	1
------	----	----	---	---	---	---

 Nuestra cantidad tendrá 6 dígitos (el de peso ponderado 64 ya no cuenta pues es superior a 39)
 - Los coeficientes que arrojan de acuerdo con su peso ponderado y su respectiva suma el decimal buscado son:

1	0	0	1	1	1
---	---	---	---	---	---

NOTA:

- Para números mayores, utilizar el método ilustrado en el libro de texto, capítulo 1, “divisiones sucesivas”

EJERCICIOS DE TRASLACION ENTRE BASES

- 1) Encontrar el hexadecimal equivalente del número: 463_8
- 2) Trasladar el numero 11100110_2 a decimal
- 3) Trasladar el numero $0XAA8$ a octal
- 4) Encontrar el binario equivalente del número: 89_{10}
- 5) Trasladar el numero 52_{16} a decimal

1. $463_8 \rightarrow$ Hexadecimal

$\begin{array}{ccc} 4 & 6 & 3 \\ 100 & 110 & 011 \\ & \underbrace{100110011} & \\ & 0001 & 0011 & 0011 \\ & 1 & 3 & 3 \end{array}$

R// 133_{16}

2. $11100110_2 \rightarrow$ Decimal

$\begin{array}{ccccccc} 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 2^7 & 2^6 & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\ 128 & 64 & 32 & 0 & 0 & 4 & 2 & 0 \end{array} = 230$

R// 230

3. $0XAA8 \rightarrow$ Octal

$\begin{array}{ccc} A & A & 8 \\ 1010 & 1010 & 1000 \\ & \underbrace{101010101000} & \\ & 5 & 2 & 5 & 0 \end{array}$

R// 5250_8

4. $89_{10} \rightarrow$ Binario

$\begin{array}{rcl} 89 & = & 64 + 25 \\ & & 1000000 + 11001 \\ & & 1000000 \\ & & 11001 \\ \hline & & 1011001 \end{array}$

R// 1011001_2

5. $52_{16} \rightarrow$ Decimal

$\begin{array}{cc} 5 & 2 \\ \underbrace{16^1} & \underbrace{16^0} \\ 80 & 2 \\ \hline & 82 \end{array}$

R// 82

LOGICA BINARIA

Es toda la estructura académica necesaria para conocer, entender y aplicar la lógica digital a la solución de problemas reales

La lógica binaria, esta soportada en dos grandes pilares:

a) Variables binarias:

Importantísimas para representar el comportamiento de los valores binarios en tiempo y espacio.

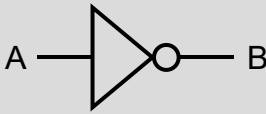

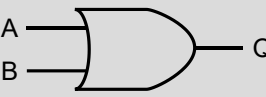
Ejemplo en espacio: cuando tenemos varios conductores con señales o valores digitales, y necesitamos diferenciar uno del otro

Ejemplo en tiempo, cuando tenemos un solo conductor con una señal binaria que cambia su valor en el tiempo, y correlacionamos el valor digital con la variable tiempo a través de una ecuación que lógicamente involucra a la o las variables binarias en cuestión.

Nexos lógicos: que no son más que algoritmos de comportamiento humano llevados a un nivel de estudio y análisis extrapolados a la lógica digital, ej.: Nexo de inclusión de recursos para lograr una meta “Y”. el nexo de elección dentro de varias alternativas para lograr un objetivo “O” y finalmente el Si y el No que responden justo a la no negación y negación de un proceso “inversión”. En lógica digital simplemente a cada uno de los anteriores nexos se le asocio un “kit” consistente en un nombre, una correlación de variables entrada versus variables de salida, un símbolo y una tabla de verdad, y así nació la lógica digital con sus diferentes compuertas.

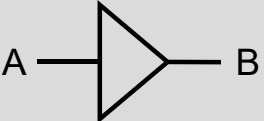
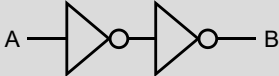
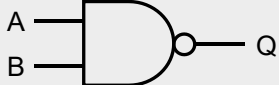

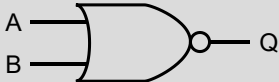

Compuertas Lógicas

Compuertas Lógicas Básicas


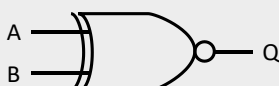

Nombre	Nexo	Símbolo	Ecuación Booleana	Tabla de Verdad																		
Inversor	Si/No		$B = \bar{A}$	<table><tr><th>Entrada</th><th>Salida</th></tr><tr><th>A</th><th>B</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	Entrada	Salida	A	B	0	1	1	0										
Entrada	Salida																					
A	B																					
0	1																					
1	0																					
AND (Multiplicación Lógica)	Inclusión		$Q = A \cdot B$	<table><tr><th colspan="2">Entrada</th><th>Salida</th></tr><tr><th>A</th><th>B</th><th>Q</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	Entrada		Salida	A	B	Q	0	0	0	0	1	0	1	0	0	1	1	1
Entrada		Salida																				
A	B	Q																				
0	0	0																				
0	1	0																				
1	0	0																				
1	1	1																				
OR (Suma Lógica) o OR Inclusiva	Selección		$Q = A + B$	<table><tr><th colspan="2">Entrada</th><th>Salida</th></tr><tr><th>A</th><th>B</th><th>Q</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	Entrada		Salida	A	B	Q	0	0	0	0	1	1	1	0	1	1	1	1
Entrada		Salida																				
A	B	Q																				
0	0	0																				
0	1	1																				
1	0	1																				
1	1	1																				

Nota: las compuertas con dos entradas mostradas son una posibilidad

Compuertas Lógicas Derivadas 1

Nombre	Símbolo	Ecuación Booleana	Tabla de Verdad																		
BUFFER	<div></div> <div></div>	$B = A$	<table><tr><th>Entrada</th><th>Salida</th></tr><tr><th>A</th><th>B</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table>	Entrada	Salida	A	B	0	0	1	1										
Entrada	Salida																				
A	B																				
0	0																				
1	1																				
NAND o NOT-AND	<div></div> <div></div>	$Q = \bar{A} + \bar{B}$	<table><tr><th colspan="2">Entrada</th><th>Salida</th></tr><tr><th>A</th><th>B</th><th>Q</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	Entrada		Salida	A	B	Q	0	0	1	0	1	1	1	0	1	1	1	0
Entrada		Salida																			
A	B	Q																			
0	0	1																			
0	1	1																			
1	0	1																			
1	1	0																			
NOR o NOT-OR	<div></div> <div></div>	$Q = \bar{A} \cdot \bar{B}$	<table><tr><th colspan="2">Entrada</th><th>Salida</th></tr><tr><th>A</th><th>B</th><th>Q</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	Entrada		Salida	A	B	Q	0	0	1	0	1	0	1	0	0	1	1	0
Entrada		Salida																			
A	B	Q																			
0	0	1																			
0	1	0																			
1	0	0																			
1	1	0																			

Compuertas Lógicas Derivadas 2

Nombre	Símbolo	Ecuación Booleana	Tabla de Verdad		
XOR o OR Exclusiva		$Q = A \oplus B$ $Q = \bar{A}.B + A.\bar{B}$	Entrada		Salida
			A	B	Q
			0	0	0
			0	1	1
			1	0	1
			1	1	0
XNOR o Equivalencia	 	$Q = A \odot B$ $Q = \bar{A} \oplus \bar{B}$ $Q = \bar{A}.\bar{B} + A.B$	Entrada		Salida
			A	B	Q
			0	0	1
			0	1	0
			1	0	0
			1	1	1

XOR o XNOR: Solo puede tener 2 puertas lógicas (Compuertas de solo 2 entradas, no podría funcionar el producto cruz)

P// Será que se puede obtener una ecuación booleana que diga $Z = X \text{ XOR } Y \text{ XOR } W$ ¿Se puede obtener una ecuación con más de 2 variables con XOR?

R// Si se puede, operando un lado & dicho resultado operarlo con el otro XOR (Haciéndolo en parejas)

Álgebra de BOOLE o Álgebra BI-VALENTE

Los axiomas, postulados, teoremas y leyes asociadas a esta herramienta de manejo de variables binarias en el entorno también binario y su verificación en la base binaria le confieren a esta el título de “campo numérico”, con ello la base binaria adquiere características importantísimas que luego veremos reflejadas en el diseño digital.

TABLA: Postulados & Teoremas del Álgebra de Boole

Postulado 2	(a) $x + 0 = x$	(b) $x \cdot 1 = x$
Postulado 5	(a) $x + x' = 1$	(b) $x \cdot x' = 0$
Teorema 1	(a) $x + x = x$	(b) $x \cdot x = x$
Teorema 2	(a) $x + 1 = 1$	(b) $x \cdot 0 = 0$
Teorema 3 (Involución)	$(x')' = x$	
Postulado 3 (Conmutativo)	(a) $x + y = y + x$	(b) $xy = yx$
Teorema 4 (Asociativo)	(a) $x + (y + z) = (x + y) + z$	(b) $x(yz) = (xy)z$
Postulado 4 (Distributivo)	(a) $x(y + z) = xy + xz$	(b) $x + yz = (x + y)(x + z)$
Teorema 5 (DeMorgan)	(a) $(x + y)' = x'y'$	(b) $(xy)' = x' + y'$
Teorema 6 (Absorción)	(a) $x + xy = x$	(b) $x(x + y) = x$

EJ: Utilizando Teoremas, Axiomas & Leyes del Álgebra Booleana, proceda a agrupar por términos & común & luego simplificar la siguiente expresión. (Obtenga la mayor simplificación)

$$z'y'x' + z'y'x + z'yx'$$

$$z'y'x' + z'y'x + z'yx'$$

Teorema 1 (a)

$$z'y'x' + z'y'x' + z'y'x + z'yx'$$

Postulado 4 (a)

$$z'y'(1) + z'x'(1)$$

Postulado 2 (b)

$$z'y' + z'x'$$

Postulado 4 (a)

$$z'(x' + y')$$

$$\text{R// } z'(x' + y')$$

El resultado tiene una compuerta OR que concatena a y' & a x' & una compuerta AND que concatena lo que está en paréntesis con z' .

EJ: Se requiere que, utilizando alguna herramienta booleana, usted obtenga una forma equivalente de la estructura anterior, pero con concatenaciones diferentes entre variables.

$$z'(x' + y')$$

$$z'(x' + y')$$

Teorema 5 (DeMorgan)

$$[[z'(x' + y')]]'$$

$$[z + (x' + y')]'$$

$$[z + (x \cdot y)]'$$

$$\text{R// } [z + (x \cdot y)]'$$

Métodos de Agrupación

Para analizar este tema y que el estudiante entienda su trascendencia, se necesita que se observe el dibujo a bloques adjunto:



Existen 2 métodos de agrupación (técnica para lograr la articulación algebraica que correlacionara a “variables independientes” versus “variables dependientes” en un entorno digital con vistas a la solución de un problema).

Método de Agrupación por Términos MINIMOS “MINTERM”

Que se basa para su articulación algebraica del término y la posterior agrupación de términos en lo que se conoce como “convención de lógica positiva”, la cual se basa en las siguientes reglas básicas:

- El dígito “1” se considera un evento lógico verdadero.
- El dígito “0” se considera un evento lógico falso.
- Cualquier variable que se involucre en una articulación algebraica en pro de la obtención de una función booleana y represente a un evento lógico verdadero, deberá aparecer NO negada.
- Por el contrario, cualquier variable que se involucre en una articulación algebraica en pro de la obtención de una función booleana y represente a un evento lógico falso, deberá aparecer negada.

Método de Agrupación por Términos MAXIMOS o “MAXTERM”

Que se basa para su articulación algebraica del término y la posterior agrupación de términos en lo que se conoce como “convención de lógica NEGATIVA”, la cual se basa en las siguientes reglas básicas:

- El dígito “1” se considera un evento lógico falso.
- El dígito “0” se considera un evento lógico verdadero.
- Cualquier variable que se involucre en una articulación algebraica en pro de la obtención de una función booleana y represente a un evento lógico verdadero, deberá aparecer NO negada.
- Por el contrario, cualquier variable que se involucre en una articulación algebraica en pro de la obtención de una función booleana y represente a un evento lógico falso, deberá aparecer negada.

OBSERVEMOS DEL LIBRO DE TEXTO

En el Capítulo 2, la tabla 2-3; titulada “términos mínimos y máximos para 3 variables” Pág. 61

Tabla 2-3: Términos Mínimos & Máximos para 3 variables.

Términos Mínimos					Términos Máximos	
X	Y	Z	Término	Designación	Término	Designación
0	0	0	$x'y'z'$	m_0	$x + y + z$	M_0
0	0	1	$x'y'z$	m_1	$x + y + z'$	M_1
0	1	0	$x'yz'$	m_2	$x + y' + z$	M_2
0	1	1	$x'yz$	m_3	$x + y' + z'$	M_3
1	0	0	$xy'z'$	m_4	$x' + y + z$	M_4
1	0	1	$xy'z$	m_5	$x' + y + z'$	M_5
1	1	0	xyz'	m_6	$x' + y' + z$	M_6
1	1	1	xyz	m_7	$x' + y' + z'$	M_7

Concluimos que:

Agrupar por Mínimos: Conlleva dos etapas:

- **Primera Etapa:** cuando creamos al término en sí, que en este caso es la concatenación de las variables independientes involucradas por nexos “&”.
- **Segunda Etapa:** que es la concatenación de los términos mínimos que hacen verdadera a la función que buscamos, concatenados por nexos “o”, así:

$$F_n = \sum m_j$$

Donde:

- m_j Son los términos mínimos que hacen verdadera a la función Booleana F_n

Agrupar por Máximos: Conlleva dos etapas:

- **Primera Etapa:** cuando creamos al término en sí, que en este caso es la concatenación de las variables independientes involucradas por nexos “o”.
- **Segunda Etapa:** que es la concatenación de los términos máximos que hacen verdadera a la función que buscamos, concatenados por nexos “&”, así:

$$F_n = \prod M_j$$

Donde:

- M_j Son los términos mínimos que hacen verdadera a la función Booleana F_n

EJERCICIOS DE METODOS DE AGRUPACIÓN

Observe la tabla de verdad adjunta y proceda de acuerdo con criterios de optimización, la correlación entre variables independientes y dependientes (la función booleana)

Variables Independientes				Variables Dependientes			
Ponderado Decimal	C	B	A	F1	F2	F3	F4
0	0	0	0	1	1	0	1
1	0	0	1	1	0	0	1
2	0	1	0	1	0	0	1
3	0	1	1	0	1	0	1
4	1	0	0	0	0	0	1
5	1	0	1	0	1	0	1
6	1	1	0	0	1	0	0
7	1	1	1	0	0	1	1

Desarrollo de F1:

Vemos de la columna de F1 que la cantidad de 1's es menor que de ceros y optamos por agrupar por mínimos, así:

$$F1 = \sum m_j = \sum m_0, m_1, m_2 = m_0 + m_1 + m_2 = C'B'A' + C'B'A + C'B A'$$

$$F1 = \text{multiplicativa } M3, M4, M5, M6, M7 = M3 \cdot M4 \cdot M5 \cdot M6 \cdot M7 \\ = (C+B'+A')(C'+B+A)(C'+B+A')(C'+B'+A)(C'+B'+A')$$

simplificación. A este tipo de estructura booleana se le llama "FORMA CANONICA" (ver libro de texto, cap. 2)

Desarrollo de F2:

$$F2 = \text{sumatoria } m_0, m_3, m_5, m_6 = m_0 + m_3 + m_5 + m_6$$

$$F2 = \text{multiplicativa } M1, M2, M4, M7 = M1 \cdot M2 \cdot M4 \cdot M7$$

Desarrollo de F3:

Vemos de la columna que tiene como nombre F3; que existen ceros en todas las filas, con excepción de la última, razón por la cual se hace imperativo trabajar con mínimos, así:

$$F3 = m_3 = C B A$$

Desarrollo de F4: Igual que la anterior vemos que todas las filas poseen un "1" en la columna de "F3" con excepción de la penúltima, por razón de optimización de tarea utilizaremos máximos, así:

$$F4 = M6 = A' + B' + C$$

METODOS DE SIMPLIFICACIÓN

Simplificar: Simplificar significa en la manipulación de estructuras algebraicas en base binaria, NO solo factorizar y lograr en “apariencia” una estructura algebraica más pequeña. Simplificar en lógica digital va más allá, simplificar significa eliminar variables y tantos términos como se pueda y/o eliminar términos completamente, si no logramos esto, poco hemos hecho, lo más seguro que solo habremos trasladado nuestras estructuras algebraicas de forma de suma de productos a producto de sumas o al revés.

Todos los métodos de optimización están basados en algebra booleana, incluso, uno de ellos como tal es el álgebra booleana; citemos algunos de estos métodos:

- Algebra Booleana
- Mapas de Karnaugh
- Primeros Implicados

Algebra Booleana: es obvio que contiene todos los lineamientos a través de los cuales debemos y podemos manipular estructuras algebraicas en la base binaria, pero conlleva el inconveniente que debemos tener un alto nivel de habilidad, para poder “simplificar óptimamente” una estructura booleana. Note, NO basta con simplificar o tratar de simplificar, lo debemos hacer de una forma óptima, y repito para ello se necesita una gran habilidad al utilizar el álgebra booleana (explicita)

Mapa de Karnaugh: es un método de índole grafico-matricial utilizado para lograr simplificar funciones booleanas (la óptima si se aplica cuidadosamente el método). Este método es susceptible de sufrir apreciaciones subjetivas de parte del usuario, sobre todo si el sujeto que lo usa no observa todas las reglas y las concatena adecuadamente. Por esta razón se dice que de una persona a otra utilizando este método para simplificar la misma función se pueden obtener estructuras algebraicas con diferente grado de simplificación (más o menos optimas). ES OBLIGACION del usuario del método obtener LA OPTIMA.

Método de Primeros Implicados: Es un método de índole tabular por excelencia totalmente mecánico en su dinámica, de tal forma que es difícil obtener una estructura algebraica booleana simplificada que no sea la óptima.

Veamos el libro de texto en el capítulo 3; de la página 75 en adelante

EJEMPLO DE APLICACIÓN DEL ALGEBRA DE BOOLE (EXPLICITA) EN LA TAREA DE OBTENER UNA FUNCIÓN BOOLEANA SIMPLIFICADA

Procedamos a simplificar la siguiente función booleana dependiente de 3 variables independientes, así:

$$\text{SIMPLIFICAR } F1(C, B, A) = C'B'A' + C'B'A + C'BA'$$

APLICACIÓN ERRONEA O CON FALTA DE HABILIDAD:

- 1) Factorizamos por elemento común el primer y segundo término (izquierda a derecha)

$$F1(C, B, A) = C'B'(A + A') + C'BA'$$

$$2) \quad = C'B'(1) + C'BA' \quad (\text{aplico } x + x' = 1)$$

$$3) \quad = C'B' + C'BA' \quad (\text{aplico } x \cdot 1 = x)$$

$$\text{RESPUESTA NO OPTIMA (AUNQUE SIMPLIFICADA): } F1(C, B, A) = C'B' + C'BA'$$

APLICACIÓN CORRECTA O CON HABILIDAD:

- 1) Factorizamos por elemento común el primero con el segundo y tercero (aplico $x + x = x$)

$$F1(C, B, A) = C'B'A' + C'B'A + C'B'A' + C'BA' \quad (\text{se clono el primero})$$

$$2) \quad = C'B'(A' + A) + C'A'(B' + B) \quad (\text{se aplica } x + x' = 1)$$

$$3) \quad = C'B'(1) + C'A'(1) \quad (\text{se aplica } x \cdot 1 = x)$$

$$4) \quad = C'B' + C'A'$$

$$\text{RESPUESTA SIMPLIFICADA Y OPTIMA: } F1(C, B, A) = C'B' + C'A'$$

EJEMPLO DE SIMPLIFICACION DE UNA FUNCION DE 4 VARIABLES POR MEDIO DEL MAPA DE KARNAUGHT

TABLA DE VERDAD

Variables Independientes					Variables Dependientes		
Ponderado Decimal	MSB 2^3	2^2	2^1	2^0 LSB	F1	F2	F3
J	D	C	B	A			
0	0	0	0	0	1	1	1
1	0	0	0	1	0	0	1
2	0	0	1	0	1	0	1
3	0	0	1	1	1	1	1
4	0	1	0	0	0	0	1
5	0	1	0	1	1	1	1
6	0	1	1	0	1	1	1
7	0	1	1	1	1	0	1
8	1	0	0	0	0	0	1
9	1	0	0	1	1	1	1
10	1	0	1	0	0	1	1
11	1	0	1	1	0	0	1
12	1	1	0	0	1	1	1
13	1	1	0	1	1	0	1
14	1	1	1	0	0	0	1
15	1	1	1	1	1	1	0

La matriz correspondiente se estructuraría así: F1

BA →		00	01	11	10
DC ↓	00		0	1	1
	01	0	1	1	1
	11	1	1	1	0
	10	0	1	0	0
		m_0	m_4	m_{12}	m_8
			m_1	m_5	m_9
				m_3	m_7
					m_6
				m_{15}	m_{14}
					m_{10}
				m_{11}	

NOTE LOS SIGUIENTE:

- En la matriz que generamos con su rotulación de denotación de filas y columnas y con la colocación de los unos y ceros que corresponden a la función F1, hemos hecho que este mapa sea el mapa para “simplificar a F1”
- Hemos seleccionado agrupar por mínimos, igual el mapa soporta igual de bien mínimos o máximos. Al escoger mínimos debemos respetar los lineamientos de lógica positiva
- Hemos cubierto con figuras de contorno cerrado todos los “1’s” existentes en la matriz, la mayor cantidad de ellos por figura, la menor cantidad de figuras y evitando la redundancia de agrupación de “1’s” por figura.

Articulación Algebraica simplificada (conclusión de agrupación por figuras de contorno cerrado)

$$F1 = \sum \text{fig1, fig2, fig3, fig4, fig5}$$

Donde: se debe entender que la designación Fig. “n”, corresponde a “el fruto de la reducción de la estructura, producto de la agrupación producida por la figura “n” “

$$F1 = D'B + D'C'A' + CA + DCB' + DB'A$$

La aplicación correcta del mapa nos daría esta estructura como la forma algebraica simplifica (la óptima). A esta forma algebraica el libro de texto le llama FORMA ALGEBRAICA NORMALIZADA

PROBLEMAS PROPUESTOS:

Obtener las estructuras algebraicas simplificadas (las óptimas) correspondientes a las funciones booleanas siguientes:

1) F2

2) F3

Ambas de la tabla de verdad de 4 variables que se ve página arriba

3) F4 (Z, Y, X) = $\sum m_0, m_3, m_6, m_7$

1) F2

BA \ DC	00	01	11	10
00	1 m_0	0 m_1	1 m_3	0 m_2
01	0 m_4	1 m_5	0 m_7	1 m_6
11	1 m_{12}	0 m_{13}	1 m_{15}	0 m_{14}
10	0 m_8	1 m_9	0 m_{11}	1 m_{10}

$$\begin{aligned}
 F2 &= D'C'B'A' + D'C'BA + D'CB'A + D'CBA' + DCB'A' + DCBA + DC'B'A + DC'BA' \\
 &= D'C'(B'A' + BA) + D'C(B'A + BA') + DC(B'A' + BA) + DC'(B'A + BA') \\
 &= D'C'(A \text{ XOR } B)' + D'C(A \text{ XOR } B) + DC(A \text{ XOR } B)' + DC'(A \text{ XOR } B) \\
 &= (D'C' + DC)(A \text{ XOR } B)' + (D'C + DC')(A \text{ XOR } B) \\
 &= (D \text{ XOR } C)'(A \text{ XOR } B)' + (D \text{ XOR } C)(A \text{ XOR } B)
 \end{aligned}$$

Cambio de Variable:

$$W = D \text{ XOR } C$$

$$Z = A \text{ XOR } B$$

Cambiando Variables:

$$W'Z' + WZ = (W \text{ XOR } Z)'$$

Regreso de variables:

$$[(D \text{ XOR } C) \text{ XOR } (A \text{ XOR } B)]'$$

2) F3

<div> <div>BA →</div> <div>DC ↓</div> </div>		00	01	11	10
		00	01	11	10
00		1 m_0	1 m_1	1 m_3	1 m_2
01		1 m_4	1 m_5	1 m_7	1 m_6
11		1 m_{12}	1 m_{13}	0 m_{15}	1 m_{14}
10		1 m_8	1 m_9	1 m_{11}	1 m_{10}

Fig. 1: Grouping $m_0, m_1, m_4, m_5, m_8, m_9$

Fig. 2: Grouping m_3, m_2, m_7, m_6

Fig. 3: Grouping m_{11}, m_{10}

Fig. 4: Grouping m_{14}, m_{10}

$$F3 = B' + D'B + DC'B + DBA'$$

3) F4 (Z, Y, X) = $\sum m_0, m_3, m_6, m_7$

<div> <div>YX →</div> <div>Z ↓</div> </div>		00	01	11	10
		00	01	11	10
0		1 m_0	0 m_1	1 m_3	0 m_2
1		0 m_4	0 m_5	1 m_7	1 m_6

Grouping m_0, m_3, m_7, m_6

$$F4 = Z'Y'X' + YX + ZY$$

APARICION DE LOS VALORES DE “NO IMPORTA”, DIAGRAMACIÓN DIGITAL Y TRASLACIÓN DE DIAGRAMAS DE COMPUERTAS MISCELANEAS A “NOR” O “NAND”

APARICIÓN DE VALORES DE “NO IMPORTA” “DON’T CARE” Y SU MANIPULACIÓN:

La denotación de la existencia de un valor de no importa comúnmente se hace utilizando la letra “X”, el origen de la aparición de estos valores puede ser variada, y tiene sus orígenes en los análisis de campo del enunciado de un problema que se solventara por electrónica digital. El tratamiento o manipulación de este valor de “no importa” es importante, pues la base binaria como tal solo soporta dos valores “el 1 lógico” y “el 0 lógico”; la aparición de un tercer valor “X” podría causar serias confusiones, pues si nos encontramos en base binaria el entorno de elementos representativo se debería mantener en “dos elementos”; pero el estudiante debe comprender: Que el “valor de no importa” no es un tercer elemento, si no la incertidumbre o libertad según el punto de vista de donde se analice, de que este elemento “X” puede adoptar el valor “1” o el valor “0”, realmente cual valor adopte queda a criterio del diseñador, según le convenga, por esta razón muchos libros de texto le llaman “comodín”, esta palabra y su significado se hacen patentes cuando el “valor de no importa” se involucra en un proceso de simplificación. Como se esperaba, el hecho que un diseñador utilice como “comodín” al “valor de no importa” en pro de lograr una simplificación optima de una función booleana, también tendrá sus repercusiones en el circuito digital que se obtenga y la forma de trabajar del mismo.

EJEMPLO:

Simplificar utilizando el mapa de Karnaugh, la función booleana F1

Variables Independientes					Variables Dependientes	
Ponderado Decimal	MSB 2^3	2^2	2^1	2^0 LSB	F1	F2
J	D	C	B	A		
0	0	0	0	0	X	1
1	0	0	0	1	X	1
2	0	0	1	0	1	1
3	0	0	1	1	0	1
4	0	1	0	0	X	1
5	0	1	0	1	1	1
6	0	1	1	0	1	1
7	0	1	1	1	1	1
8	1	0	0	0	0	1
9	1	0	0	1	1	1
10	1	0	1	0	Xc	1
11	1	0	1	1	0	1
12	1	1	0	0	1	X
13	1	1	0	1	1	X
14	1	1	1	0	X	X
15	1	1	1	1	1	1

BA \ DC	00	01	11	10
00	X=0 <i>m</i> ₀	X=1 <i>m</i> ₁	1 <i>m</i> ₃	0 <i>m</i> ₂
01	X=1 <i>m</i> ₄	1 <i>m</i> ₅	1 <i>m</i> ₇	1 <i>m</i> ₆
11	1 <i>m</i> ₁₂	1 <i>m</i> ₁₃	1 <i>m</i> ₁₅	X=1 <i>m</i> ₁₄
10	0 <i>m</i> ₈	1 <i>m</i> ₉	X=1 <i>m</i> ₁₁	0 <i>m</i> ₁₀

Se han marcado con figuras de contorno cerrado en color rojo la mejor distribución de ceros, esto, si elegimos agrupar por máximos, y con figuras de contorno cerrado en línea de color verde la mejor disposición geométrica de unos según Karnaugh, esto, si agrupamos por mínimos, favor analizar las siguientes observaciones:

- Observar cómo, con o sin “valores de no importa” la reglas que rigen la obtención de funciones booleanas simplificadas se aplican igual
- Observar como a criterio, se cambian las “X” por unos o ceros, según convenga a la mejor agrupación, ya sea por mínimos o máximos. Pero NO indiscriminadamente
- Debemos saber que el cambiar “X” por unos o ceros, nos proveerá una mejor simplificación que si no lo hacemos, PERO, se deberán investigar las repercusiones que ello tendrá en el circuito final

SIMPLIFICACIÓN:

1) Por Mínimos: $F1 = \sum \text{fig. 1, fig. 2} = \text{fig 1} + \text{fig 2} = A + C$

2) Por Máximos: $F1 = \prod \text{figura única} = A + C$

DIAGRAMACION DIGITAL:

Con esta presentación NO se pretenda uniformizar una técnica de diagramación, más bien un consejo de cómo hacerlo, igual si el estudiante ya sabe diagramar y lo hace bien, puede utilizar su propia técnica. La única regla que podemos mencionar es la siguiente:

- Que los diagramas que ejecutemos sean claros, entendibles por los demás y nosotros mismos. Esto podría requerir que la estética este presente en los mismos

Técnica aconsejada para diagramar:

- En un espacio destinado para generar el diagrama, dibujar en el lado extremo izquierdo, de arriba- abajo a las líneas digitales que representen a las variables de entrada que intervengan en el diagrama, en sus formas negadas y no negadas
- Dividir el espacio disponible, partiendo del lugar donde se dibujaron las líneas anteriores, hasta llegar al extremo derecho, con columnas imaginarias que alberguen los diferentes grupos de variables o términos correlacionados por nexos lógicos, siendo la primera columna, la dibujada de izquierda a derecha la que albergaría a los nexos lógicos (compuertas) que correlacionan a las variables independientes, luego yendo de la correlación de variables, hacia la correlación de términos, hacia la correlación de grupos de términos etc, hasta terminar de analizar la estructura algebraica en su totalidad
- Si fuera necesario, se pueden identificar compuertas y/o nexos con un numero para facilidad de su dibujo individual en esta estructura aconsejada

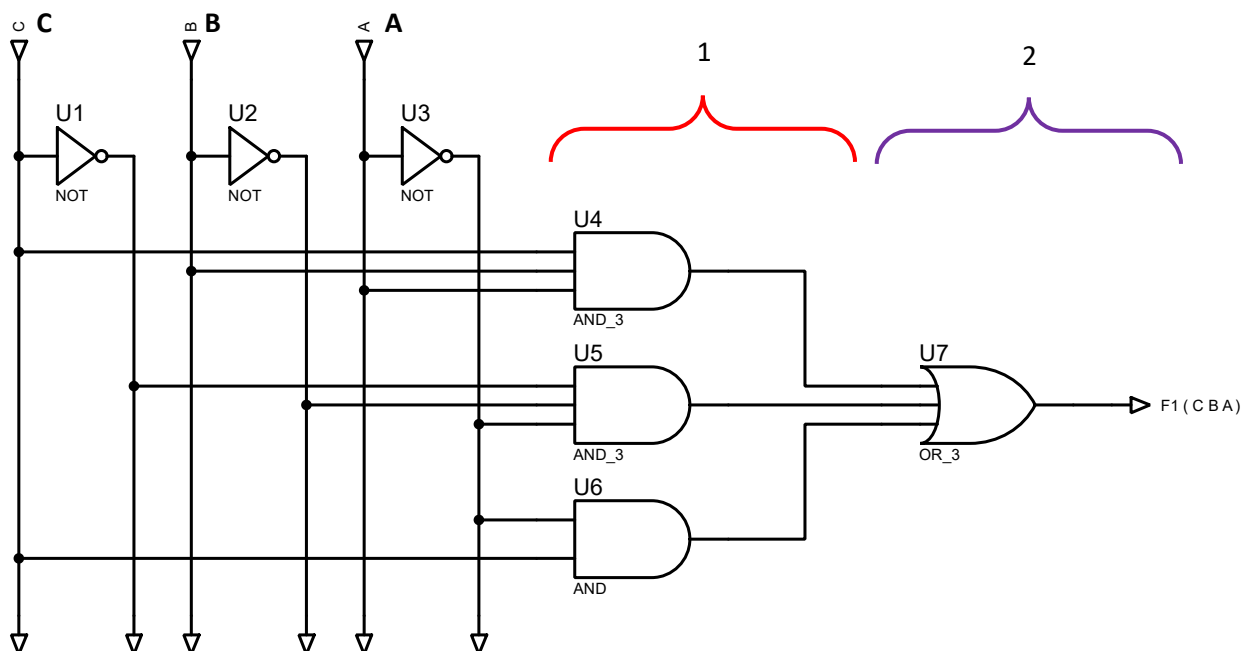
EJEMPLO: Dibujar el diagrama digital correspondiente a la función booleana F1; donde

$$F1(C, B, A) = \underbrace{\frac{A B C}{1.a} + \frac{A' B' C'}{1.b}}_2 + \frac{A' C}{1.c}$$

Observe:

- La función booleana F1; tiene tres términos formados por las variables independientes “C, B, A” articuladas por nexos lógicos & (compuertas and) a ese nivel le llamamos “1” y sub-indicamos las compuertas para propósitos posteriores; luego detectamos un segundo nivel de concatenación o correlación entre los anteriores “términos” nada más que articulados por el nexo lógico “o”, a eso otro nivel le llamamos “2”; como solo hay uno, no le adjuntamos sub-índice.
- Observamos que las variables “A, B, C” se encuentran en sus versiones negadas y no negadas

Aplicando la técnica:

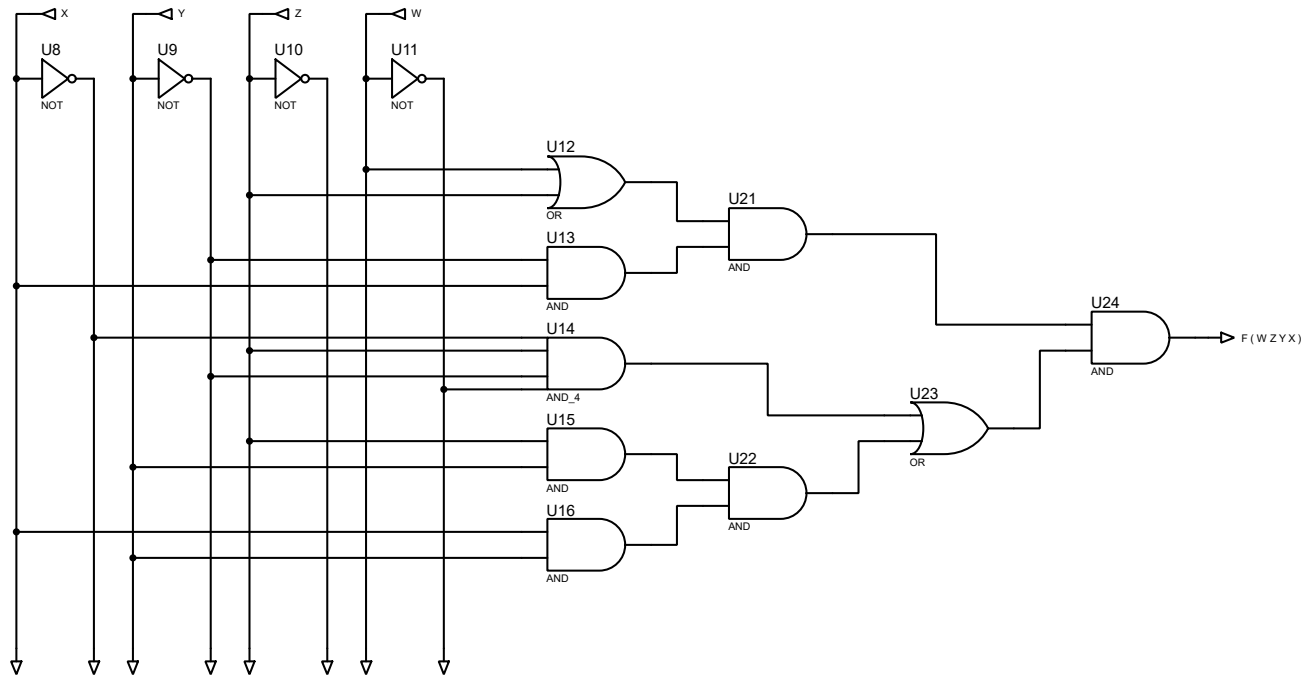


Ejercicio de Diagramación:

- Diagramar la siguiente expresión algebraica.

$$F(W, Z, Y, X) = [(W + Z)(XY')][(W'ZY'X') + (ZY)(YX)]$$

1.a 1.b 1.c 1.d 1.e
2.a 2.b
3.a
4.a



Convenciones existentes de conexión eléctrica:

CONVENCION "A":

NO CONEXIÓN:

CONEXIÓN:

CONVENCION "B":

TRASLACION DE CUALQUIER TIPO DE COMPUERTAS HACIA “NOR o NAND”:

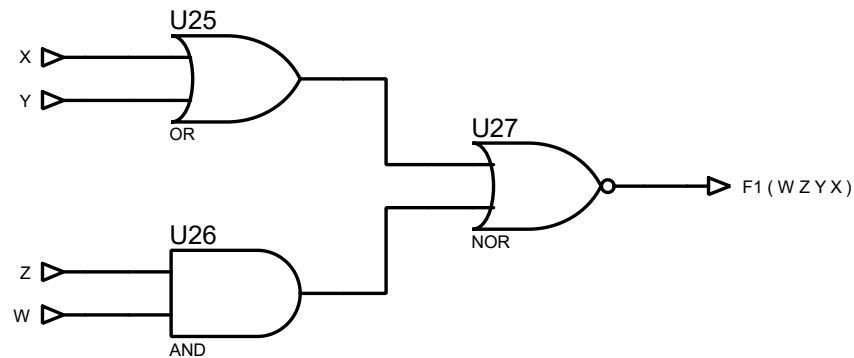
El sustento de este tipo de traslación es la aplicación de:

- El teorema de Morgan
- Tabla de verdad de las compuertas en si

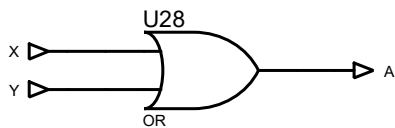
Ejemplo:

Trasladar a compuertas NAND, el siguiente diagrama:

$$F1(W, Z, Y, X) = ([X + Y] + [ZW])'$$



1) Tratamiento de la compuerta OR:

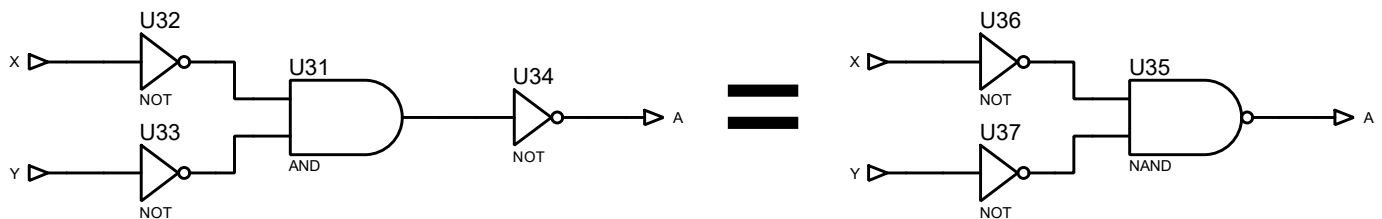


por Morgan: $A = X + Y$ Negando ambos lados

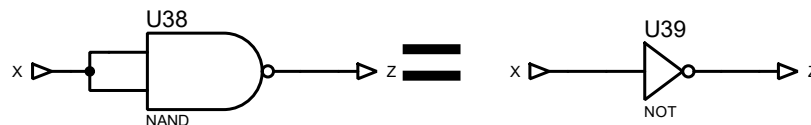
$A' = (X + Y)' = X'Y'$ regresando a la variable “A”
que el lo mismo que negarla de nuevo

$$A = (X'Y')'$$

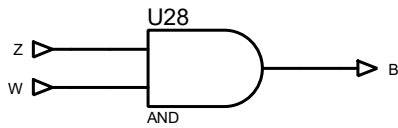
la cual se vería así:



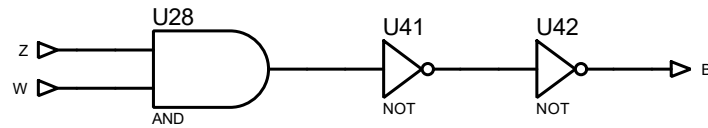
De la tabla de verdad de la compuerta NAND: $Z = (X.Y)'$ y si $X = Y$ (pensando en unir la dos entradas a un solo punto) entonces $Z = (X.X)'$ del algebra de Boole $X.X = X$ entonces $Z = X'$ en diagramación se vería:



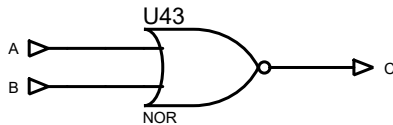
2) Tratamiento de la compuerta AND:



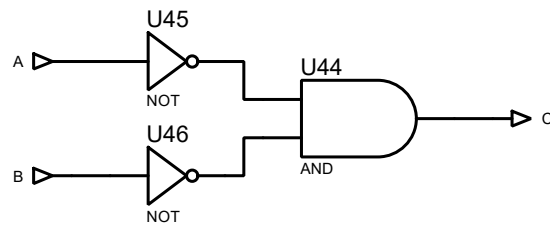
por el algebra de Boole $B = \{ (Z \cdot W)' \}'$ conversión que si vería



3) Tratamiento de la compuerta NOR:

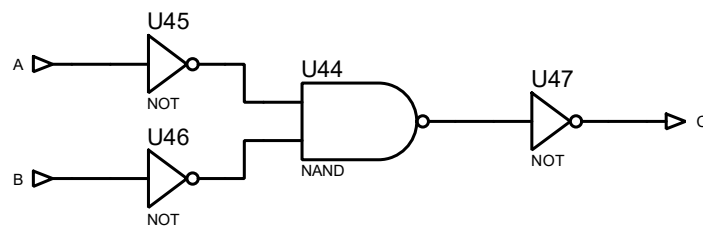


$C = (A + B)'$ aplicando Morgan $C = A'B'$
Dando como resultado:

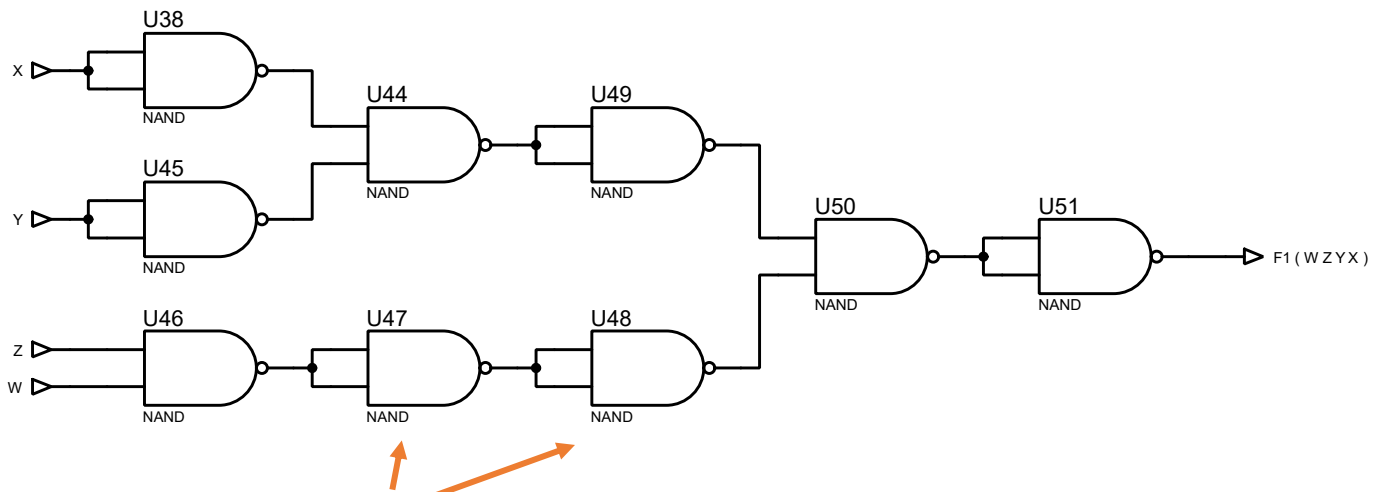


Para llegar a la forma NAND, negamos la salida dos veces, ya que por algebra de Boole:

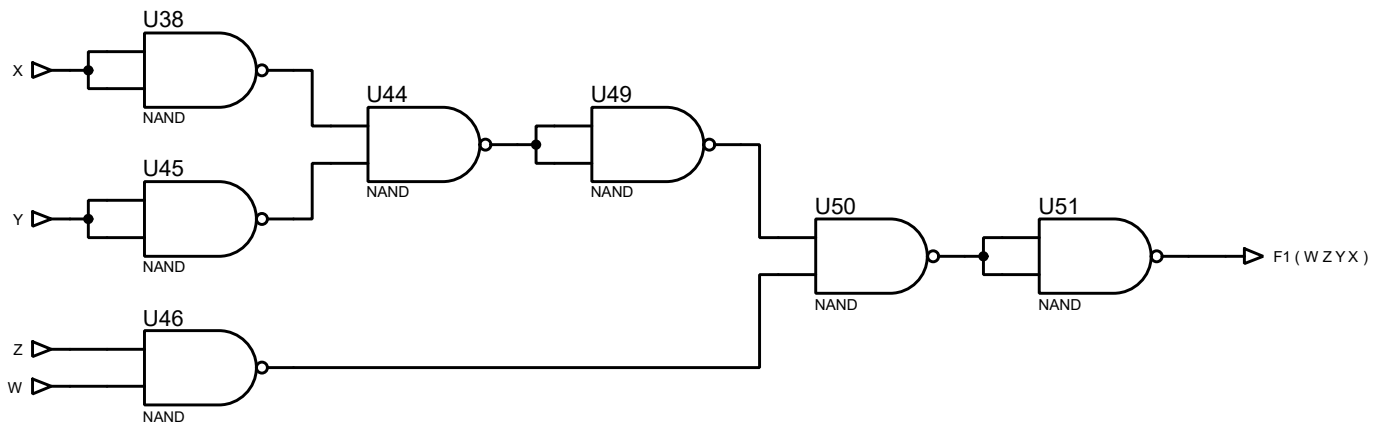
$$((X)')' = X$$



DIBUJANDO EL NUEVO CIRCUITO:



Y eliminando las dos compuertas **NAND** en serie, por el postulado $((X)') = X$

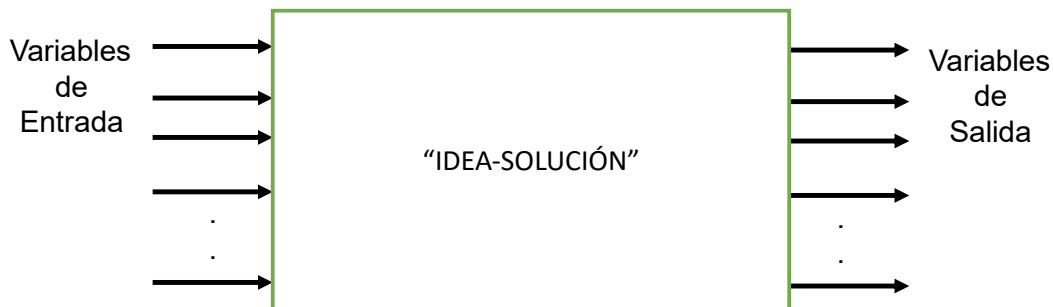


TODO EN FUNCION DE COMPUERTAS NAND

PROCEDIMIENTO DE DISEÑO CON LOGICA COMBINACIONAL**Pasos a seguir:**

- 1) Poseer un enunciado claro, esto es:
 - a. Saber si por lógica digital se le puede dar solución al problema
 - b. Determinar cuántas y cuáles variables están involucradas en el problema
 - c. Denotar adecuadamente las variables tanto las independientes como las dependientes
 - d. Conocer la o las técnicas afines asociadas a nuestro entorno científico, que nos ayuden a generar el algoritmo-solución o “idea solución”

Todo esto se puede resumir en el **“BLACK-BOX”**

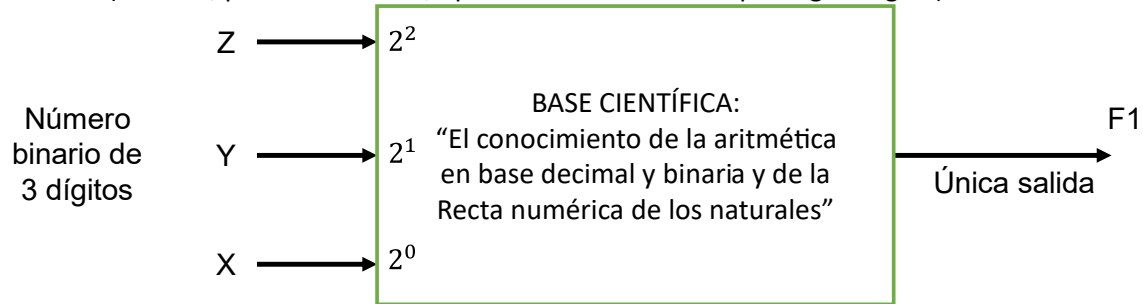


- 2) Encontrar la técnica a través de la cual académicamente logremos la correlación entre las variables de entrada y las variables de salida
 - Tabla de Verdad (“práctico” para trabajar con 4 variables independientes a lo sumo)
 - Extrapolación de algoritmos deducidos e inducidos de la base decimal (más allá de 4 variables independientes)
- 3) La optimización y obtención de una estructura booleana que algebraicamente correlacione dominio versus contra-dominio (se le puede aplicar un bypass)
 - Algebra booleana
 - Mapa de Karnaught
- 4) Generación del diagrama digital
- 5) Implementación física

EJEMPLO:

Diseñar un circuito digital que por medio de una única salida nos alerte de cuando el equivalente decimal del número binario de 3 dígitos que tiene a su entrada es múltiplo entero de 3

- 1) BLACK BOX: (el obvio, por el enunciado, que la solución se busca por lógica digital)



- 2) CORRELACION ENTRE ENTRADAS Y SALIDAS = $F1(Z,Y,X) = ?$

Debido a que el número de variables de entrada es menor a cuatro, utilizo tabla de verdad

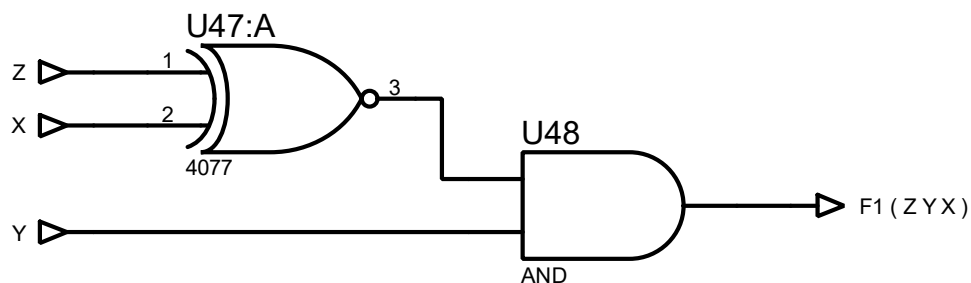
Ponderado Decimal	MSB 2^3	2^2	2^1 LSB	Función de Boole
J	Z	Y	X	F1
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	0

- 3) AGRUPACION POR MINIMOS Y SU RESPECTIVA OPTIMIZACION

Debido a lo sencillo de la distribución de “unos” que hacen verdadera a F1. Utilizare Algebra de Boole:

$$\begin{aligned}
 F1(Z,Y,X) &= \sum m_3, m_6 = m_3 + m_6 \\
 &= Z'YX + ZYX' = Y(Z'X + ZX') \\
 F1(Z,Y,X) &= Y(X \text{ xor } Z)
 \end{aligned}$$

- 4) GENERACION DEL DIAGRAMA DIGITAL



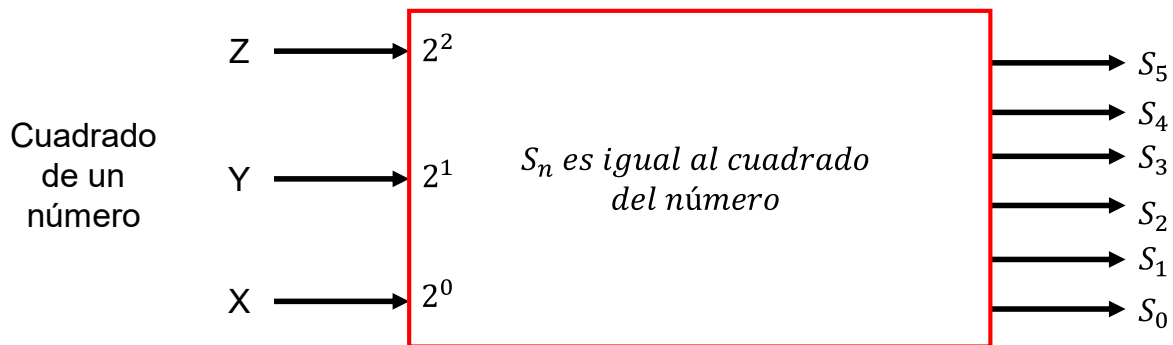
Ejercicios propuestos:

- 1) Diseñar un circuito digital que teniendo a su entrada un número binario de 3 bits, genere a su salida el cuadrado de este, exprese el resultado en binario también
- 2) Diseñar un circuito digital que teniendo a su entrada dos números binarios de 3 dígitos cada uno "A" & "B", nos indique por medio de una única salida "Z", de cuando "A" es mayor o que "B", así:

$Z = 1$ cuando $A > B$
 $Z = 0$ en cualquier otro caso

Ejercicio 1:

Paso 1: Black Box



Paso 2: Correlación entre Entradas

Ponderado Decimal	$MSB 2^2$	2^1	$2^0 LSB$	Función de Boole					
J	Z	Y	X	S_5	S_4	S_3	S_2	S_1	S_0
0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	1
2	0	1	0	0	0	0	1	0	0
3	0	1	1	0	0	1	0	0	1
4	1	0	0	0	1	0	0	0	0
5	1	0	1	0	1	1	0	0	1
6	1	1	0	1	0	0	1	0	0
7	1	1	1	1	1	0	0	0	1

Paso 3: Agrupación
 S_5

$\begin{array}{c c} & YX \\ \hline Z & \end{array}$	00	01	11	10
0	0	0	0	0
1	0		1	1

$$S_5 = ZY$$

 S_4

$\begin{array}{c c} & YX \\ \hline Z & \end{array}$	00	01	11	10
0	0	0	0	0
1	1	1	1	0

$$S_4 = ZY' + ZX$$

 S_3

$\begin{array}{c c} & YX \\ \hline Z & \end{array}$	00	01	11	10
0	0	0	1	0
1	0	1	0	0

$$S_3 = Z'YX + ZY'X$$

 S_2

$\begin{array}{c c} & YX \\ \hline Z & \end{array}$	00	01	11	10
0	0	0	0	1
1	0	0	0	1

$$S_2 = YX'$$

 S_1

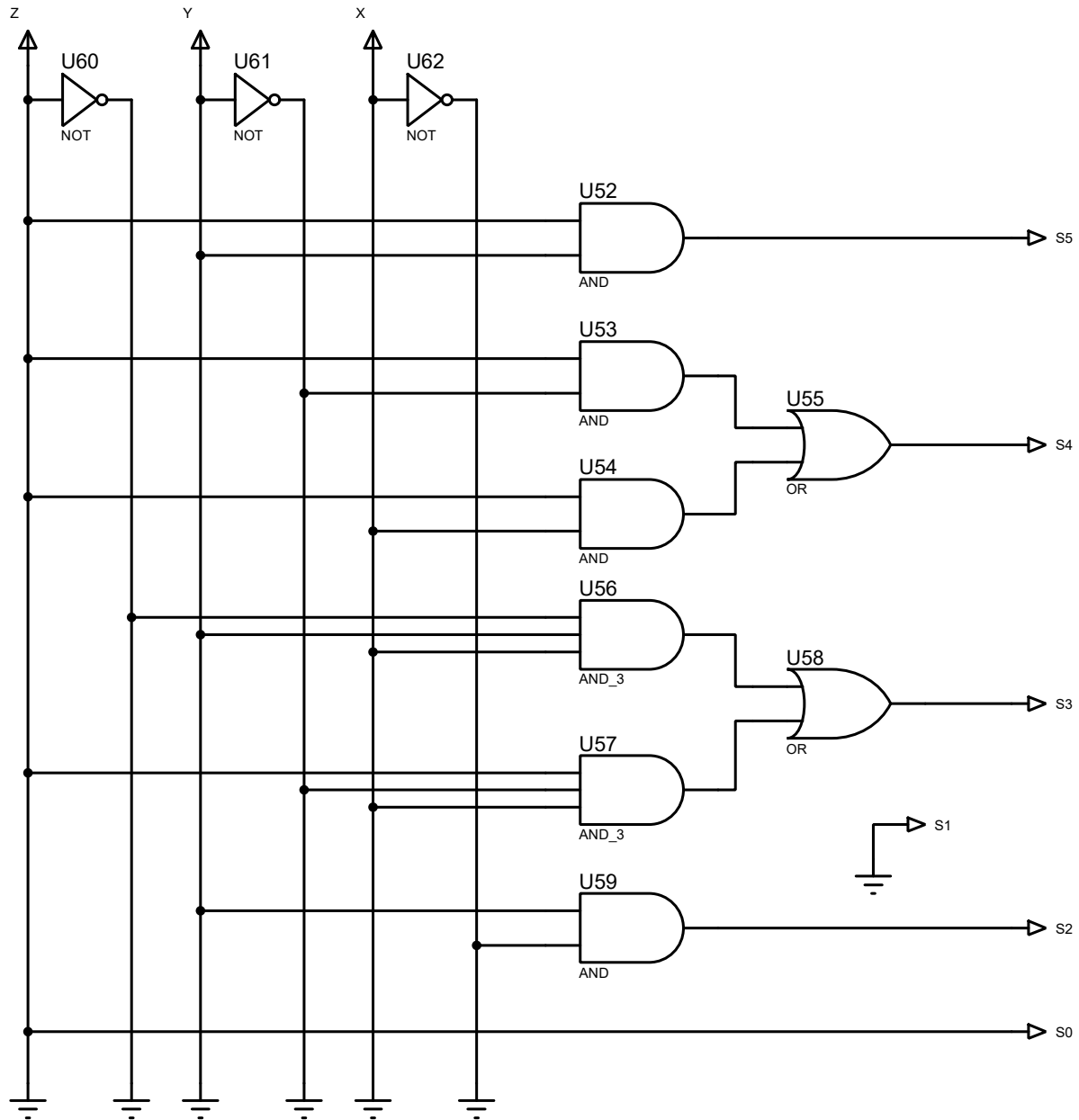
$\begin{array}{c c} & YX \\ \hline Z & \end{array}$	00	01	11	10
0	0	0	0	0
1	0	0	0	0

$$S_1 = 0$$

 S_0

$\begin{array}{c c} & YX \\ \hline Z & \end{array}$	00	01	11	10
0	0	1	1	0
1	0	1	1	0

$$S_0 = X$$

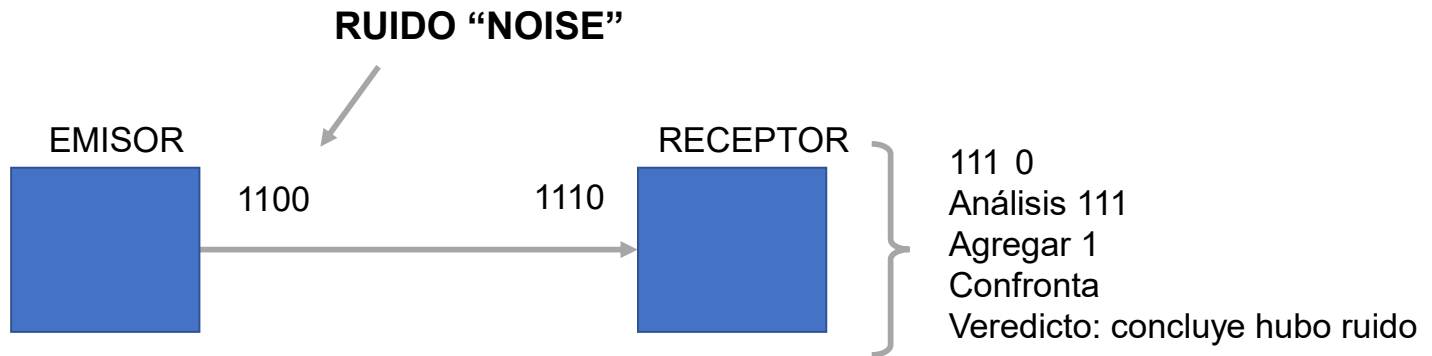
Paso 4: Generación de Diagrama

CÓDIGOS DE ERROR

Codificación por Paridad Par de UNOS

Cadena original: 110 quiero generar el bit de código: 0

Por tanto la cadena resguardada por la calificación será: 1100



Nota: Ambos, tanto emisor como receptor manejan el mismo tipo de código

DISTANCIA DE MILES DE KILOMETROS

¿Cuándo un número es mayor que otro? $23 < 25$?

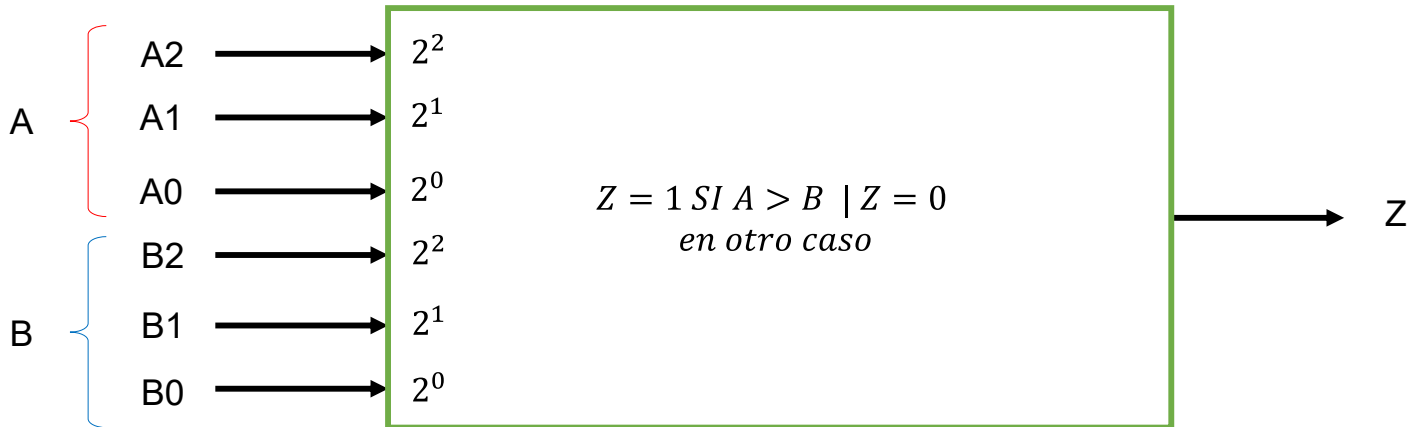
- Homogenizar el número de dígitos en cada cantidad a comparar.
- Comparamos los dígitos más significativos de ambas cantidades.
- Comparamos sucesivamente hasta el menos significativo si los anteriores comparados son iguales, hasta llegar al último y emitir criterio.

EJEMPLO DE APLICACIÓN DE ALGORITMOS DEDUCIDOS DE LA BASE DECIMAL:

Diseñar un circuito digital que tenga la capacidad de alértanos por medio de una salida “Z” de cuando “A > B” donde A & B son números binarios de 3 bits cada uno.

Así: $Z = 1$ si $A > B$; $Z=0$ en cualquier otro caso

1) Black Box



Algoritmo: Utilizaremos la misma forma de analizar en binario a como lo hacemos en la base decimal, en la cual, para comparar dos cantidades, Primero: ecualizamos el número de dígitos de ambas cantidades, Segundo: luego comparamos dígito por dígito empezando por los dígitos más ponderados, hasta llegar al menos ponderado, teniendo siempre en cuenta la comparación acumulada.

2) Correlación entre “A” & “B” Versus “Z”

Debido a que tenemos 6 variables binarias o booleanas, la tabla de verdad no es práctica y tendremos, de acuerdo con el algoritmo arriba bosquejado, aplicar algoritmos deducidos de base 10. En base binaria quedaría así:

Si A está compuesta por los dígitos:

2^2	2^1	2^0
A2	A1	A0
B2	B1	B0

Y B está compuesta por los dígitos:

Z es verdadera si

- $(A2 > B2)$ sea verdadera
- $[(A2 = B2) \& (A1 > B1)]$ sea verdadera
- $[(A2 = B2) \& (A1 = B1) \& (A0 > B0)]$ sea verdadera

• Sustituimos:

- “o” = compuerta OR
 - “&” = compuerta AND
 - “verdadera = 1
 - “falso” = 0
- } lógica positiva

$$Z = (A2 > B2) + [(A2 = B2) (A1 > B1)] + [(A2 = B2) (A1=B1) (A0 > B0)]$$

- Para poder obtener una forma algebraica con compuertas conocidas, debemos primero averiguar cuál es la función Booleana que concatena la relación “>, mayor que” Y “=, igual a”. Esta vez tenemos este problema, pero para dos variables, procedemos así:

X	Y	X > Y = F1
0	0	0
0	1	0
1	0	1
1	1	0

$$X > Y = F1 = \sum m2 = XY'$$

(trabajamos lógica positiva)

Producto de las tablas anteriores, obtenemos:

$$X > Y: XY'$$

$$X = Y: (X \text{ xor } Y)'$$

X	Y	X = Y = F2
0	0	1
0	1	0
1	0	0
1	1	1

$$F2 = \sum m0, m3 = m0 + m3 = X'Y' + XY$$

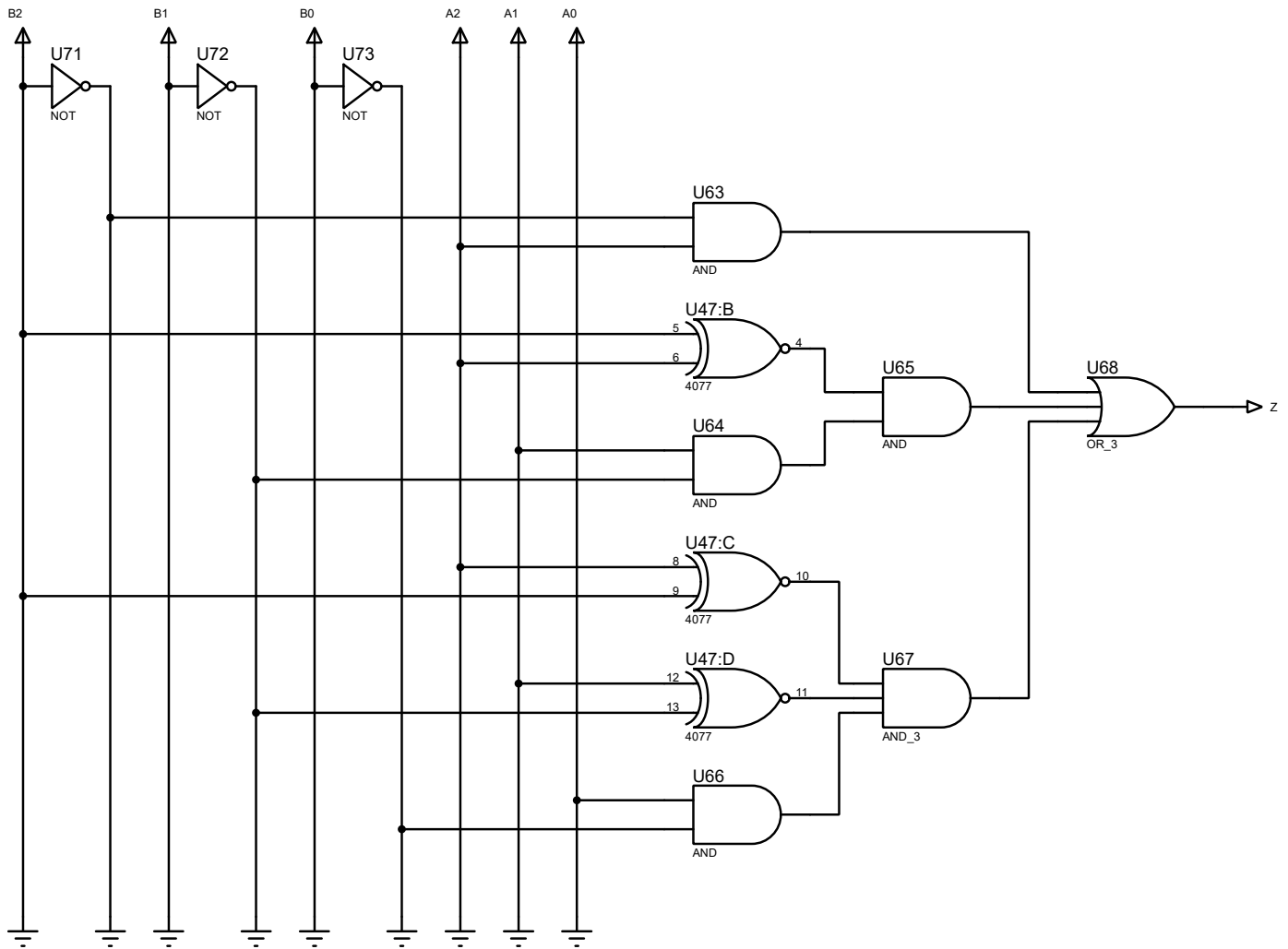
Sustituyendo en la última expresión de “Z”

$$Z = (A2 > B2) + [(A2 = B2) (A1 > B1)] + [(A2 = B2) (A1 = B1) (A0 > B0)]$$

$$Z = (A2 B2') + [(A2 \text{ xor } B2)' (A1 B1')] + [(A2 \text{ xor } B2)' (A1 \text{ xor } B1)' (A0 B0')]$$

3) Optimización de la función BOOLEANA “Z”: (Bypass)

4) Diagrama Digital



EJERCICIO: Diseña un circuito digital que teniendo a su entrada un dígito decimal codificado BCD, nos alerte por medio de una única salida “Z” de cuando hay un error en la representación del símbolo arábigo, así:

$z = 1$ si hay un error & 0 en cualquier otro caso

1) Black Box



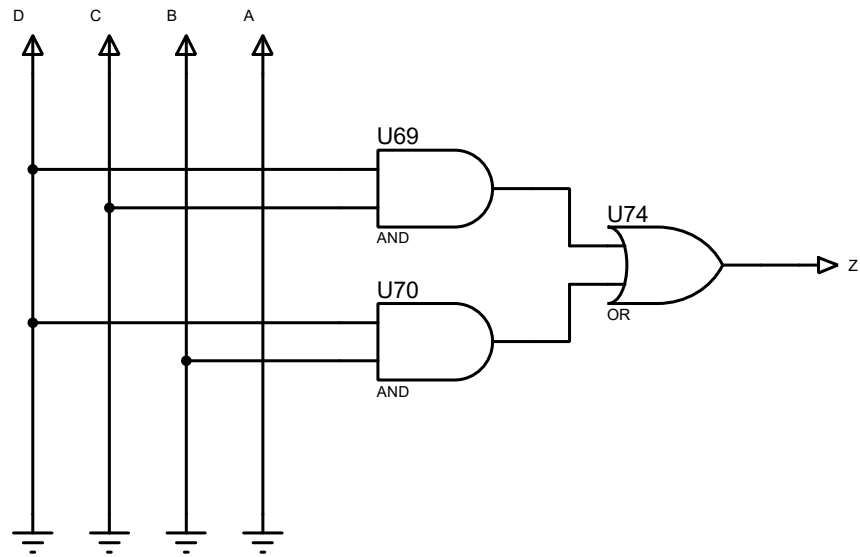
2) Corrección entre Entradas & Salidas

Ponderado Decimal	<i>MSB</i> 2^3	2^2	2^1	2^0 <i>LSB</i>	<i>Función de Boole</i>
J	D	C	B	A	Z
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	1

BA \ DC	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	0	0	1	1

$$Z = DC + DB$$

3) Diagrama Digital



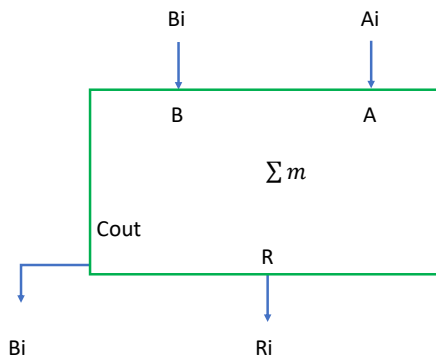
SUMADOR MEDIO (Variables de 1 dígito C/U)

Tabla de Verdad

VARIABLES INDEPENDIENTES			FUNCIONES DE BOOLE		
J	Bi	Ai	J	Cout	Ri
0	0	0	0	0	0
1	0	1	1	0	1
2	1	0	1	0	1
3	1	1	2	1	0

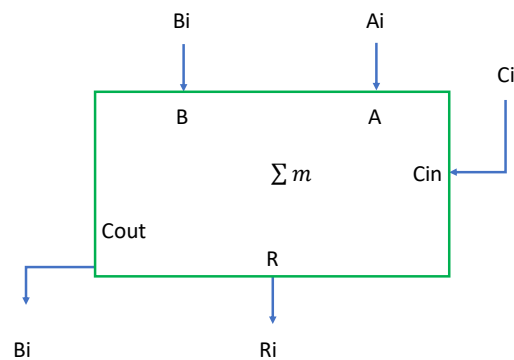
SUMADOR COMPLETO (Variables de 1 dígito C/U)

Tabla de Verdad

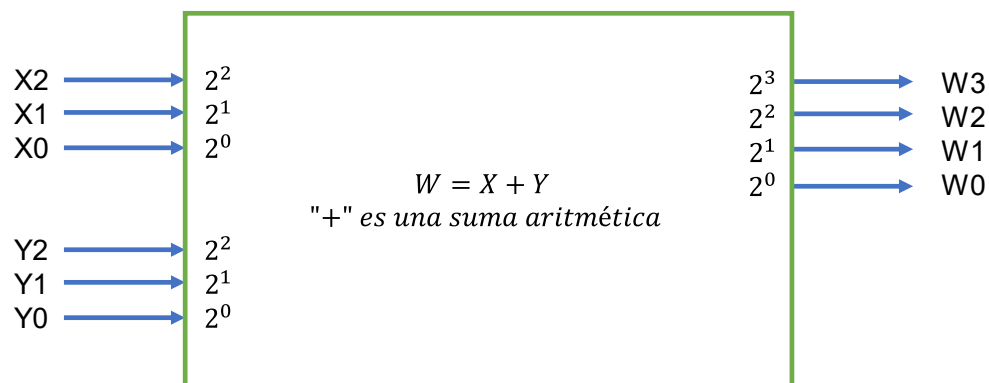
VARIABLES INDEPENDIENTES				FUNCIONES DE BOOLE		
J	Cin <i>MSB</i>	Bi	Ai <i>LSB</i>	J	Cout <i>MSB</i>	Ri <i>LSB</i>
0	0	0	0	0	0	0
1	0	0	1	1	0	1
2	0	1	0	1	0	1
3	0	1	1	2	1	0
4	1	0	0	1	0	1
5	1	0	1	2	1	0
6	1	1	0	2	1	0
7	1	1	1	3	1	1

EJEMPLO DE APLICACIÓN DE ALGORITMOS DEDUCIDOS DE BASE 10

Diseñar un circuito digital que teniendo a su entrada dos números binarios "X & Y" de 3 dígitos cada uno, proceda este a generar a su salida la suma aritmética de estos. Expresa el resultado en binario también.

1) Black Box

- El diseño se desarrolla por lógica digital.
- Poseemos 6 variables booleanas de entrada (tres por número)
- Poseeremos 4 variables binarias de salida ¿Cómo lo sé?
 - Sumo los dos números en sus combinaciones que arroja el mayor conteo acumulado, esto es:
 - ❖ $X = 111$ $Y = 111$
 - ❖ Al sumar el equivalente decimal de ambos me da $7+7 = 14$; número que al trasladarse de decimal a binario necesita 4 dígitos.
- Denotaremos con suscritos.
- Algoritmo, pues la copia de la forma en la que sumamos en decimal. Pero afrontamos un problema, y este es: que en este momento no tenemos una compuerta o función booleana articulada que nos concatene la suma de dos dígitos binarios con un acarreo anterior y un acarreo posterior.



2) Correlaciones entre "X", "Y" & "W"

Trabajar tabla de verdad NO es práctico, utilizaremos un algoritmo deducido de base 10

Ejemplo (no es una demostración)

Supongamos la siguiente suma en binario entre dos números cualquiera de "X" "Y"

Columna de
Ponderación

	3	2	1	0
X =	1	0	0	1
Y =	1	1	0	0
W	1	0	1	1

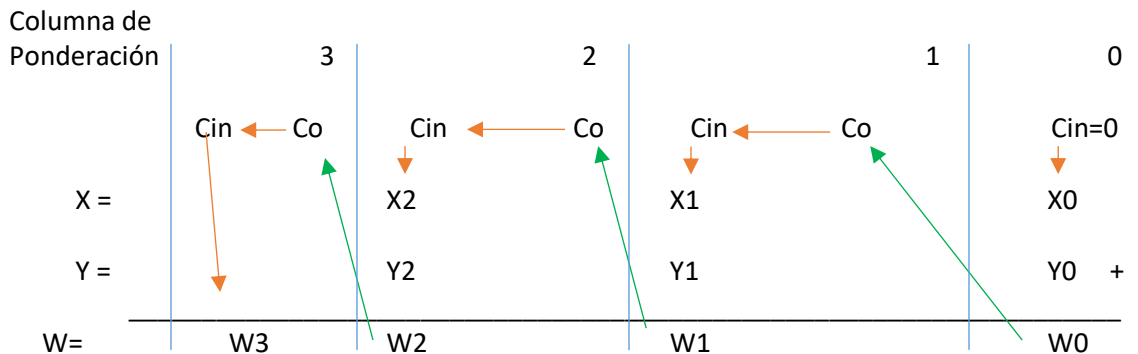
Diagrama de la suma binaria con acarreo:

- Columna 0: $1 + 0 = 1$, $Ci=0$, $Co=0$
- Columna 1: $0 + 0 = 0$, $Ci=0$, $Co=0$
- Columna 2: $0 + 1 = 1$, $Ci=0$, $Co=0$
- Columna 3: $1 + 1 = 10$, $Ci=1$, $Co=1$

La ilustración anterior nos da información valiosa acerca de cómo desarrollar una suma de varios dígitos en binario, de lo importante que es la concatenación entre parejas de dígitos en la misma columna de ponderación, esta concatenación gracias al concepto de ACARREOS tanto de entrada como de salida (Cin & Cout). Concluimos, por tanto: que necesitamos un módulo (función) booleano que sea capaz de ejecutar la suma entre dos números binarios de 1 bit cada uno y que pueda concatenar acarreo anterior y acarreo posterior.

Ese modulo lo encontramos en el libro de texto **pagina 123; Sección 4-3**. Con el nombre “sumador medio” y “sumador Completo” ambos aritméticos

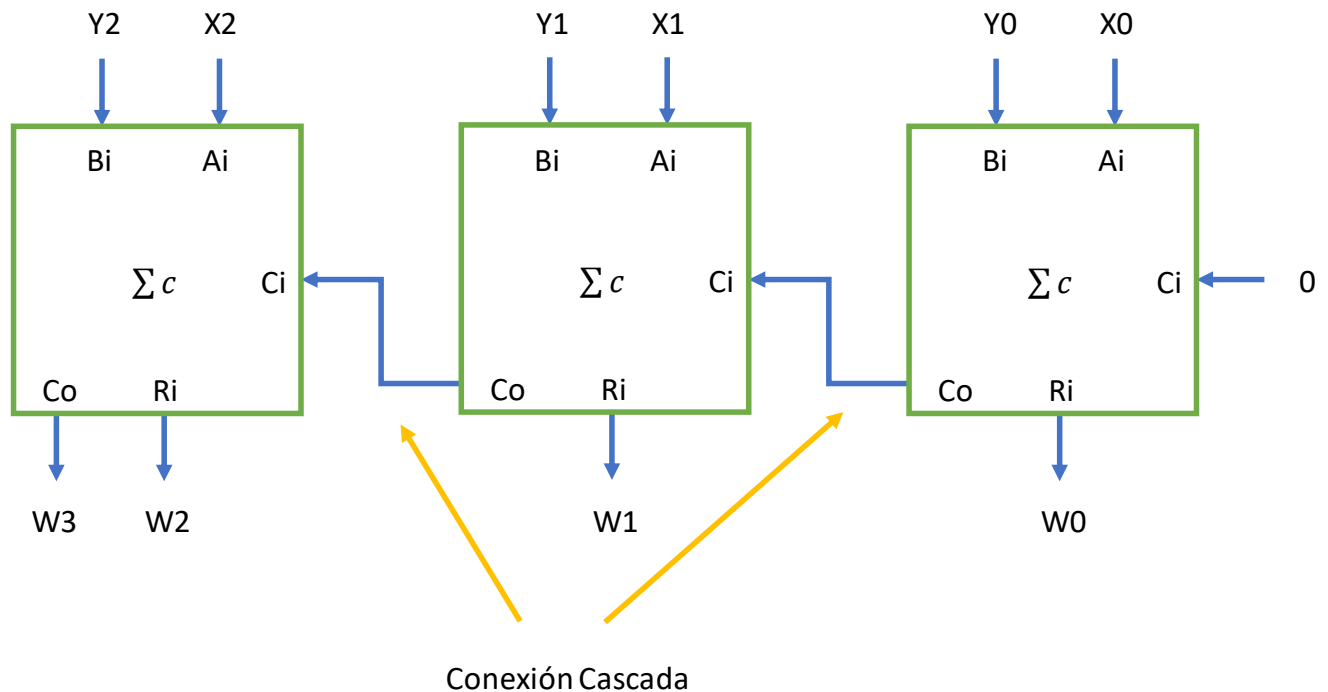
Generalizando y sabiendo que tanto los enteros en base 10 como la base binaria son campos numéricos, procedemos:



3) Optimización (Bypass)

4) Diagrama Digital

Aplicando los módulos SUMADORES COMPLETOS en cada columna de ponderación en parejas de bits, queda:



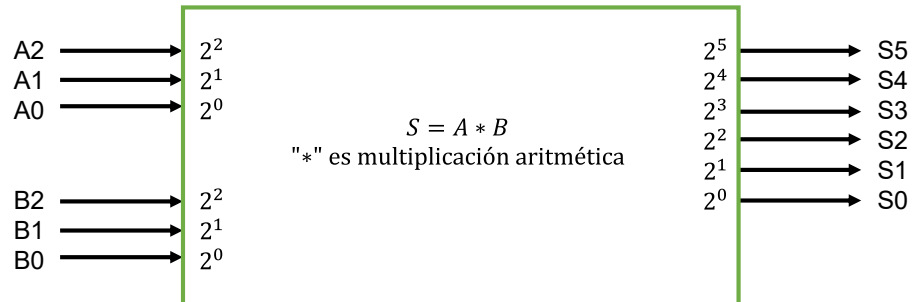
Ejercicios Para Resolver sobre Diseño con Lógica Combinacional

Ejercicio No. 1: Diseñe un circuito digital que teniendo a su entrada un número decimal de un dígito codificado BCD, sea capaz de alertarnos por medio de una única salida booleana "Z", de cuando exista un error en la representación BCD del número arriba citado así:

$$Z = 1 \text{ si hay un error}$$

$$Z = 0 \text{ en cualquier otro caso}$$

1) Black Box



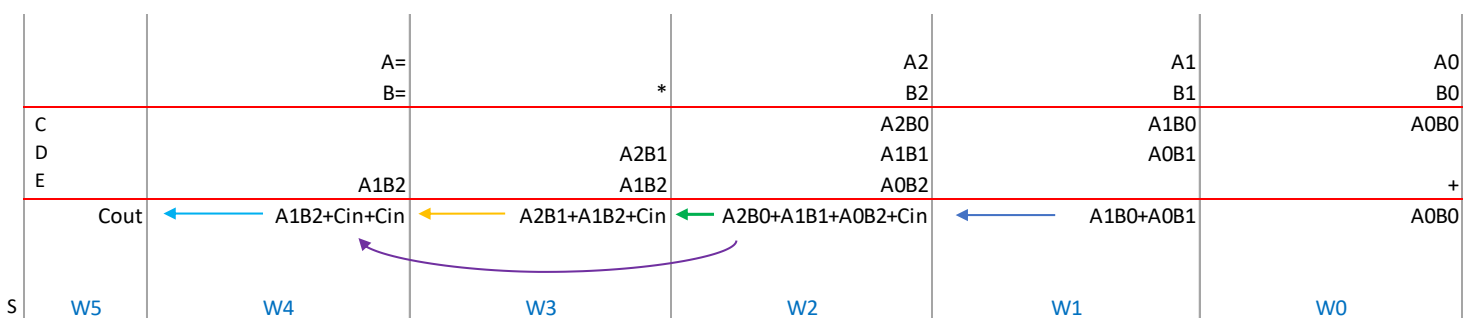
El número más grande en A & B sería 7; por lo tanto $7 \times 7 = 49$ (Forma con 6 dígitos en binario).

2) Correlación entre Dominio y Contra dominio

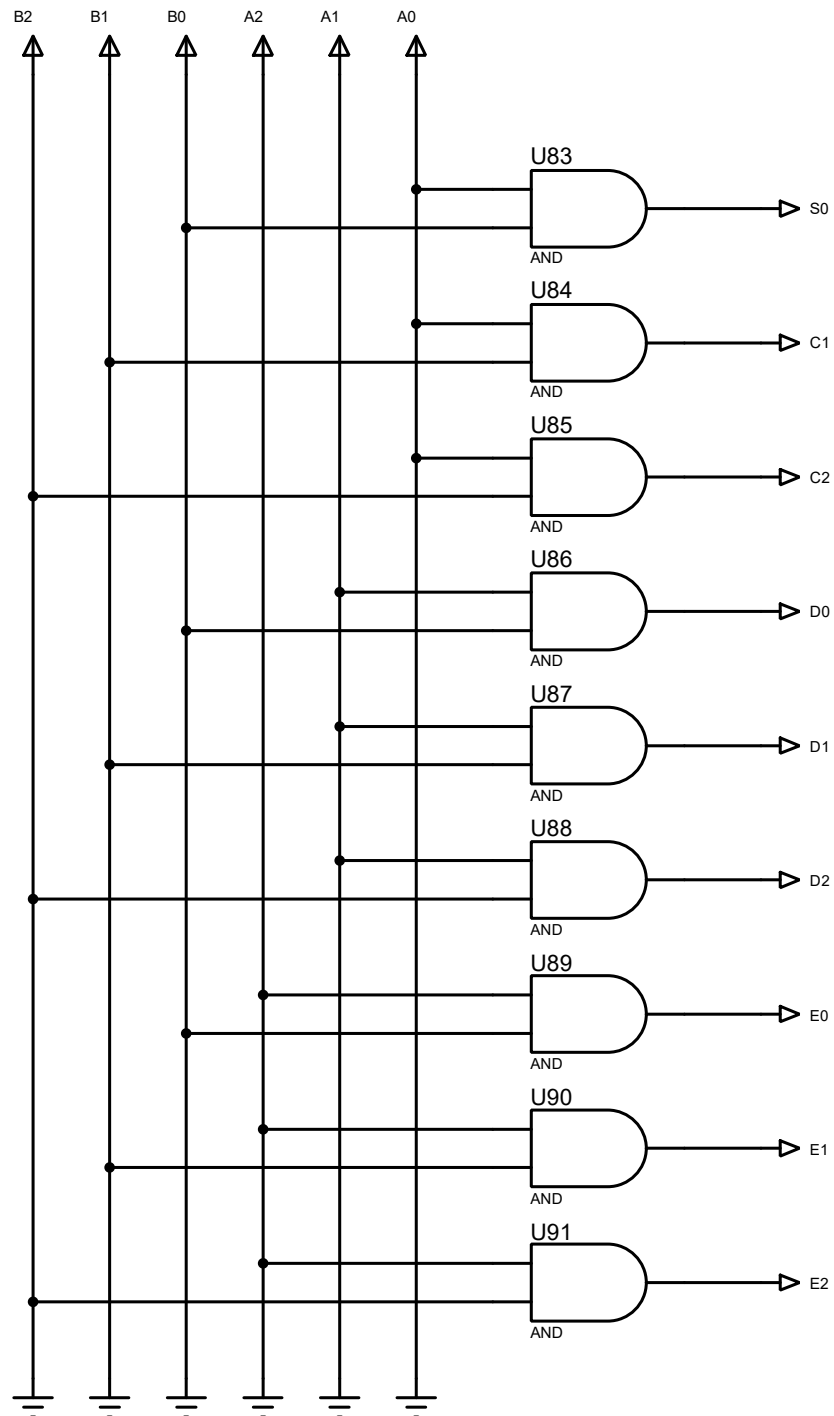
Tabla de verdad "no práctica" pues tenemos 6 variables independientes, NOS AUXILIAMOS de algoritmos inducidos de base 10.

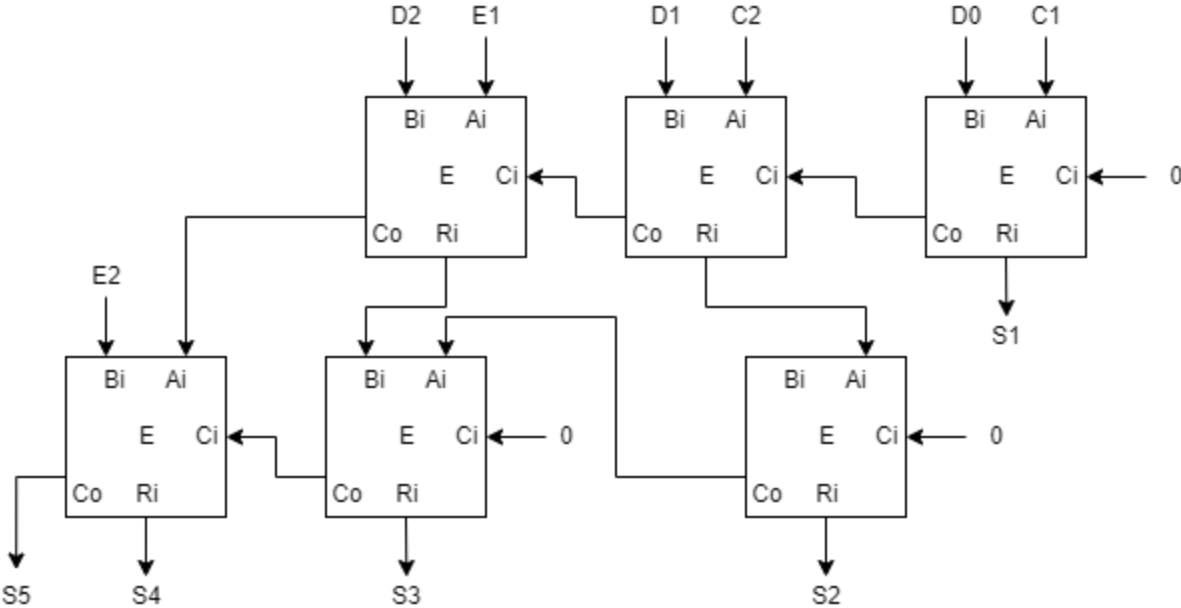
Ejemplo de multiplicación en binario usando el algoritmo de multiplicación en decimal por tándem.

	CP=5	CP=4	CP=3	CP=2	CP=1	CP=0	
A=				1	0	1	5(10)
B=			x	1	1	1	7(10)
				1	1	1	
			1	1	1		
+		1	1	1			
	1	1	0	0	0	1	35(10)



3) Diagrama Digital



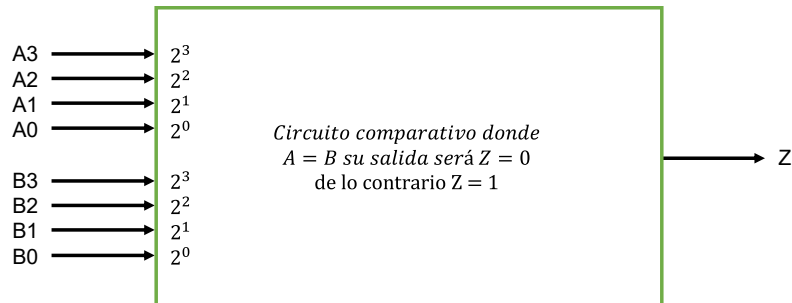


Ejercicio No. 2: Diseñar un Circuito digital que teniendo a su entrada dos números binarios “A” & “B” de 3 dígitos cada uno, genere a su salida la multiplicación aritmética de estos, exprese el resultado en binario también.

Sugerencia:

- Utilice algoritmos deducidos de base 10.
- Puede utilizar bloques sumadores medios o completos.

1) Black Box



2) Correlación entre A & B

Si A está compuesta por los dígitos:

Y B está compuesta por los dígitos:

2^2	2^1	2^0
A2	A1	A0
B2	B1	B0

Z es verdadera si

- (A3 > B3) sea verdadera
- (A2 = B2) sea verdadera
- (A2 = B1) sea verdadera
- (A0 = B0) sea verdadera

$$Z = (A3 = B3) \& (A2 = B2) \& (A1 = B1) \& (A0 = B0)$$

X	Y	$X = Y = F1$
0	0	0
0	1	1
1	0	1
1	1	0

$$Z = (X' + Y')(X + Y)$$

Entonces:

$$Z = (A3 \text{ xnor } B3)(A2 \text{ xnor } B2)(A1 \text{ xnor } B1)(A0 \text{ xnor } B0)$$

Ejercicio No. 3: Diseñar un circuito digital que teniendo a su entrada un número binario de 5 dígitos, nos indique por medio de una salida booleana “W”, de cuando este número sea impar, así:

$W = 1$ si el número de entrada es impar

$W = 0$ en cualquier otro caso

Ejercicio No. 4: Diseñe un circuito digital que, teniendo a su entrada de 3 bits, genere a su salida el bit producto de la codificación por error impar de unos del número de entrada.

Ejercicio No. 5: Observe la figura que se detalla abajo, ella muestra un horno eléctrico armado con un sensor de temperatura digital alojado en su interior, mismo que envía hacia afuera de la cavidad por medio de 5 conductores, un número digital de 5 dígitos, los cuales representan el valor de la temperatura en grados Celsius que hay en la cavidad, observe la tabla adjunta.

Usted deberá diseñar un indicador de rangos de temperatura hecho a base de LED's, observe la tabla de rangos versus color de luz indicadora.



Sensor de Temperatura de 5 Hilos

Tabla que muestra la dependencia entre temperatura en grados Celsius y el valor digital de salida del sensor.

Temperatura en Grados	Salida Digital				
$T^{\circ}C$	<i>E MSB</i>	<i>D</i>	<i>C</i>	<i>B</i>	<i>A LSB</i>
0	0	0	0	0	0
.
.
40	1	1	1	1	1

Nota: Asuma linealidad en la interrelación valores no mostrados.

Relación entre Rangos de Temperatura en Grados Celsius Versus Color de Indicador Luminoso

<i>Rango $T^{\circ}C$</i>		<i>LED VERDE</i>	<i>LED AMARILLO</i>	<i>LED ROJO</i>
0	8	ON	OFF	OFF
9	25	OFF	ON	OFF
26	40	OFF	OFF	ON

Nota: Asuma ON = 1 | OFF = 0

$$y = mx + b$$

T = Temperatura en Grados C

V = Valor Digital

$$V = mT + b \quad b = 0$$

$$V = mT$$

$$31 = m40 \text{ por tanto } m = 31/40$$

$$8^{\circ} = V = 6.2$$

$$25^{\circ} = V = 19.3$$

COLUMNA DE PONDERACIÓN	RANGO TEMPERATURA DIGITAL					ACCIÓN EN LEDS		
J	D	E	C	B	A	VERDE	AMARILLO	ROJO
1	0	0	0	0	0	ON	OFF	OFF
6	0	0	1	1	0	ON	OFF	OFF
7	0	0	1	1	1	OFF	ON	OFF
19	1	0	0	1	1	OFF	ON	OFF
20	1	0	1	0	0	OFF	OFF	ON
31	1	1	1	1	1	OFF	OFF	ON

BLOQUES DIGITALES COMBINACIONALES DE MEDIANA ESCALA DE INTEGRACION

Existen una gran variedad de bloques digitales que se podrían considerar dentro de la clasificación arriba mencionada, pero nos concentraremos en 4 bloques de estos, el primero lo citamos abajo

DECODIFICADOR:

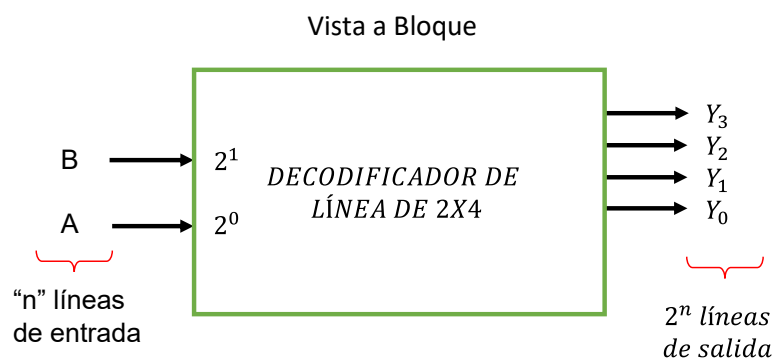
Es un bloque digital combinacional de mediana escala de integración, construido con el objetivo de generar la acción inversa a la de codificar, comúnmente estos tipos de bloques digitales poseen “n” entradas y un máximo de 2^n salidas.

Existe una gran variedad de decodificadores, todos ellos derivados de la necesidad de aplicaciones específicas, uno de ellos, diríamos el más ampliamente usado y flexible de adaptarse a las múltiples y personalizadas necesidades de diseño es el “decodificador de línea”

DECODIFICADOR DE LINEA:

Es el único decodificador que cumple con poseer a su entrada “n” líneas digitales y a su salida 2^n salidas digitales, a continuación, se muestra su tabla de verdad, y en ella se puede distinguir una característica importante de este decodificador y es: la capacidad, a través de su forma de trabajar, de poder identificar a cada una de sus salidas como la existencia de un término mínimo “mj” en función de la combinación que la genere. Es por ello que Morris Mano en su libro “Lógica digital y diseño de computadores” lo presenta como una alternativa más para poder

Ejemplo de un decodificador de línea de “2 X 4”; implementado en lógica positiva



Ponderado Decimal	Variables Entrada		Término Mínimo	Variables de Salida			
J	B	A	m_j	Y_3	Y_2	Y_1	Y_0
0	0	0	m_0	0	0	0	1
1	0	1	m_1	0	0	1	0
2	1	0	m_2	0	1	0	0
3	1	1	m_3	1	0	0	0

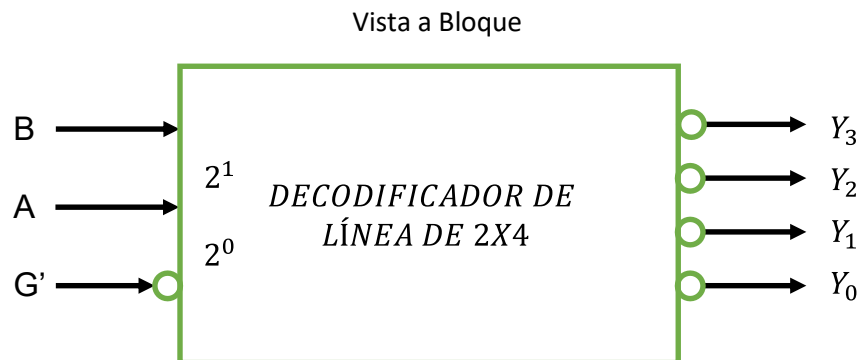
Nota: Como Y, podría interpretarse como m.

PUNTO ANEXO:

ENTRADAS DE HABILITACION

Las entradas de habilitación son entradas digitales asociadas a ciertos módulos y/o compuertas digitales que permiten o no (según el valor digital conectadas a ellas) que estos módulos o compuertas funcionen tal y como fueron proyectadas.

Ejemplo de un decodificador de línea de “2 X 4”; implementado en lógica negativa, además con entrada de habilitación “G” activa en “0”.



PONDERADO	VARIABLES DE ENTRADA			VARIABLES DE SALIDA			
J	G	B	A	Y_3	Y_2	Y_1	Y_0
0	0	0	0	1	1	1	0
1	0	0	1	1	1	0	1
2	0	1	0	1	0	1	1
3	0	1	1	0	1	1	1
N.A.	1	X	X	1	1	1	1

Nota: Cuando $G = 1$; el dispositivo está “Des-Habilitado” y por tanto no funciona como fue proyectado.

APLICACIÓN DEL DECODIFICADOR DE LINEA EN LOGICA POSITIVA A LA IMPLEMENTACION DE FUNCIONES BOOLEANAS (METODO No. 2 de diseño)

Directriz: “Cualquier función Booleana dependiente de “n” variables independientes, puede ser implementada por medio de un decodificador de línea en lógica positiva de “n X 2^n ”

Pasos:

- 1) Elegir el decodificador adecuado.
- 2) Conectar las “n” variables independientes a las “n” variables de entrada del decodificador, guardando el grado de ponderación idéntico entre ambas.
- 3) Elegir tantas compuertas OR como funciones booleanas se deban implementar. Las compuertas OR deberán tener tantas entradas como términos mínimos hagan verdadera a la función en cuestión.
- 4) Las entradas de las compuertas OR arriba citadas, se deben conectar a las salidas del decodificador “Y” con el suscrito coincidente al suscrito del término mínimo que hace verdadera a la función en cuestión.

Rotular la salida de cada compuerta OR con el nombre de la función booleana que implemento.

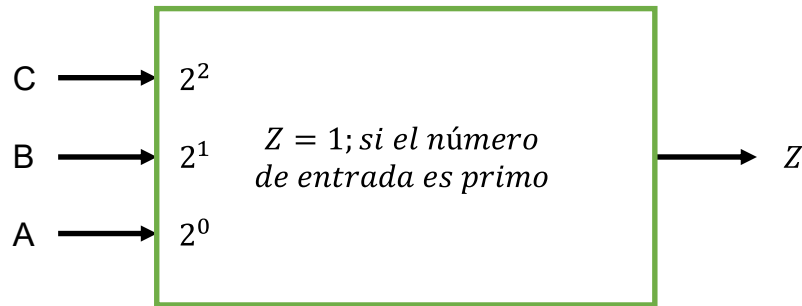
Ejemplo: Diseñar un circuito digital que teniendo a su entrada un número binario de 3 dígitos, nos indique por medio de una única salida "Z", de cuando se presente una combinación de la cual su ponderado sea primo, así:

$$Z = 1 \quad \text{si el número de entrada es primo}$$

$$Z = 0 \quad \text{en cualquier otro caso}$$

Utilice el método del decodificador de línea

- 1) BLACK BOX: El problema se puede resolver por lógica digital



- 2) CORRELACION "Z" versus "C, B, A"

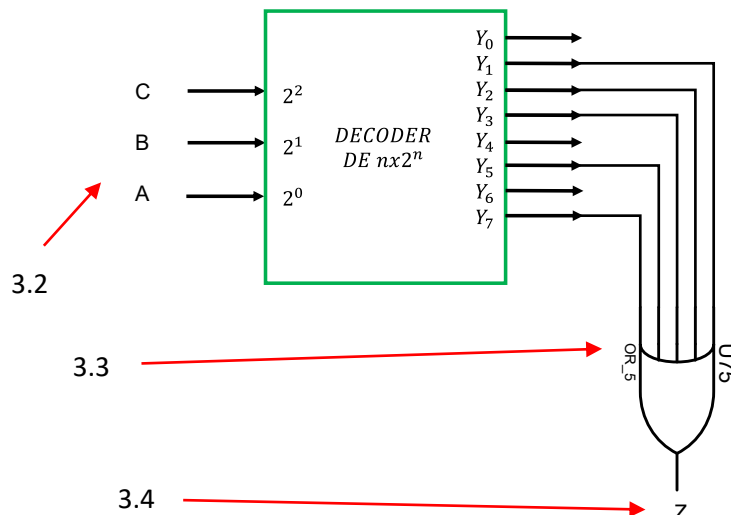
Elegimos tabla de verdad, pues el número de variables de entrada es inferior a 4, además necesitamos encontrar cuales términos mínimos hacen verdadera a "Z"

PONDERADO	VARIABLES DE ENTRADA			FUNCION DE BOOLE
J	C	B	A	Z
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

Obtenemos que $Z(C, B, A) = \sum m_1, m_2, m_3, m_5, m_7$

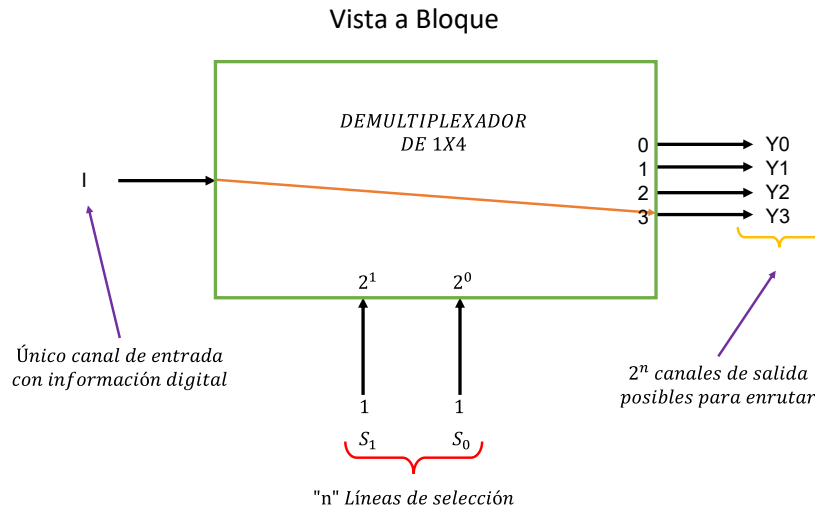
$$Z(C, B, A) = \prod m_0, m_4, m_6$$

- 3) APLICANDO EL METODO DE IMPLEMENTACIÓN POR DECODIFICADOR EN LÍNEA, LÓGICA POSITIVA



DEMULTIPLEXADOR:

Es un bloque digital combinacional de mediana escala de integración construido para enrutar la información digital existente en su único canal de entrada hacia sus posibles 2^n canales de salida (uno de ellos a la vez), donde existe un grupo de “n” líneas digitales de entrada a este bloque, llamadas líneas de selección, las que por el peso ponderado existente en ellas se establece que canal de salida es el que recibe la información y la enruta.

Ejemplo de un Demultiplexador de 1 X 4**Tabla de Función**

PONDERADO DECIMAL	LÍNEAS DE SELECCIÓN		CANALES DE SALIDA			
J	S ₁	S ₀	Y3	Y2	Y1	Y0
0	0	0	0	0	0	1
1	0	1	0	0	1	0
2	1	0	0	1	0	0
3	1	1	1	0	0	0

Ejercicio: Usando la técnica de diseño combinacional por DECODIFICADOR DE LÍNEA positivo, proceda generar la estructura electrónica correspondiente al sumador completo.

1) Black Box



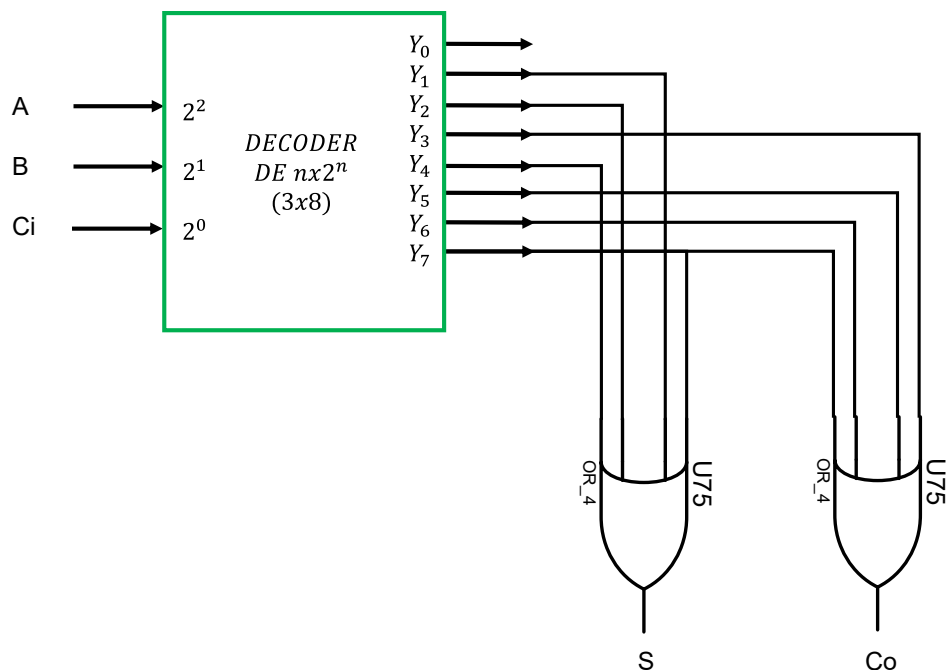
2) Correlación

J	A	B	Ci	Co	S
0	0	0	0	0	0
1	0	0	1	0	1
2	0	1	0	0	1
3	0	1	1	1	0
4	1	0	0	0	1
5	1	0	1	1	0
6	1	1	0	1	0
7	1	1	1	1	1

$$\text{Obtenemos que } S(A, B, Ci) = \sum m_1, m_2, m_4, m_7$$

$$\text{Obtenemos que } C_0(A, B, Ci) = \sum m_3, m_5, m_6, m_7$$

3) Método de implementación por decodificador de línea, Lógica positiva.



Ejemplo: Se desea elaborar un circuito digital que controle el toldo de una terraza. El toldo tiene la función de dar sombra como de proteger del viento y de la lluvia.

Por lo tanto, en un toldo resistente al viento y a la lluvia, que mantiene la terraza seca en los días que llueve. Para la elaboración del circuito digital debemos tomar en cuenta las siguientes señales:

- Señal S: Indica si hay sol.
- Señal L: Indica si llueve.
- Señal V: Indica si hay mucho viento.
- Señal F: Indica si hace frío en el interior de la casa.

Según los valores de estas señales se bajará o subirá el toldo.

Se debe tomar en cuenta que nuestro circuito digital trabaja con lógica positiva por lo tanto asumiremos que si deseamos tener el toldo extendido (BAJADO) el valor tendría que ser 1 & si lo deseamos tener recogido (SUBIDO) el valor tendría que ser 0.

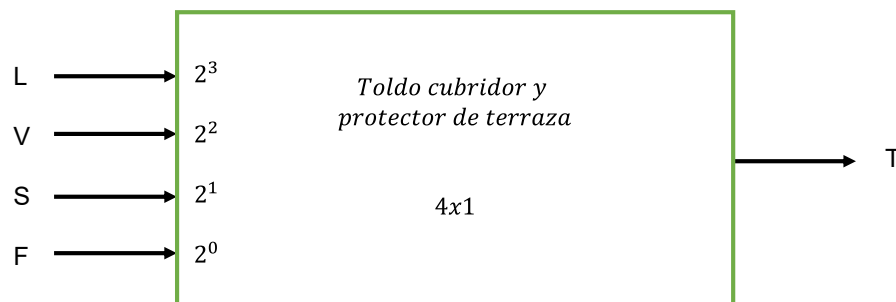
El circuito digital que controla el toldo debe funcionar según las siguientes premisas:

- Independientemente del resto de señales de entrada, siempre que llueva se debe extender el toldo para evitar que se moje la terraza. No se considerará posible que simultáneamente llueva y haga sol.
- Si hace viento se debe extender el toldo para evitar que el viento moleste. Sin embargo, hay una excepción aún cuando haya viento, si el día está soleado y hace frío en la casa, se recogerá el toldo para que el sol caliente la casa.
- Por último, si no hace viento ni llueve, solo se bajará el toldo en los días de sol y cuando haga calor en el interior, para evitar que se caliente mucho la casa.

Al documento agregar:

- Tabla de verdad.
- Mapas de Karnaugh.
- Función simplificada.
- Diagrama del circuito final en el PDF.
- Circuito final con extensión (.lvwa/.pdsprj)

1) Black Box



0 → Bajado

1 → Subido

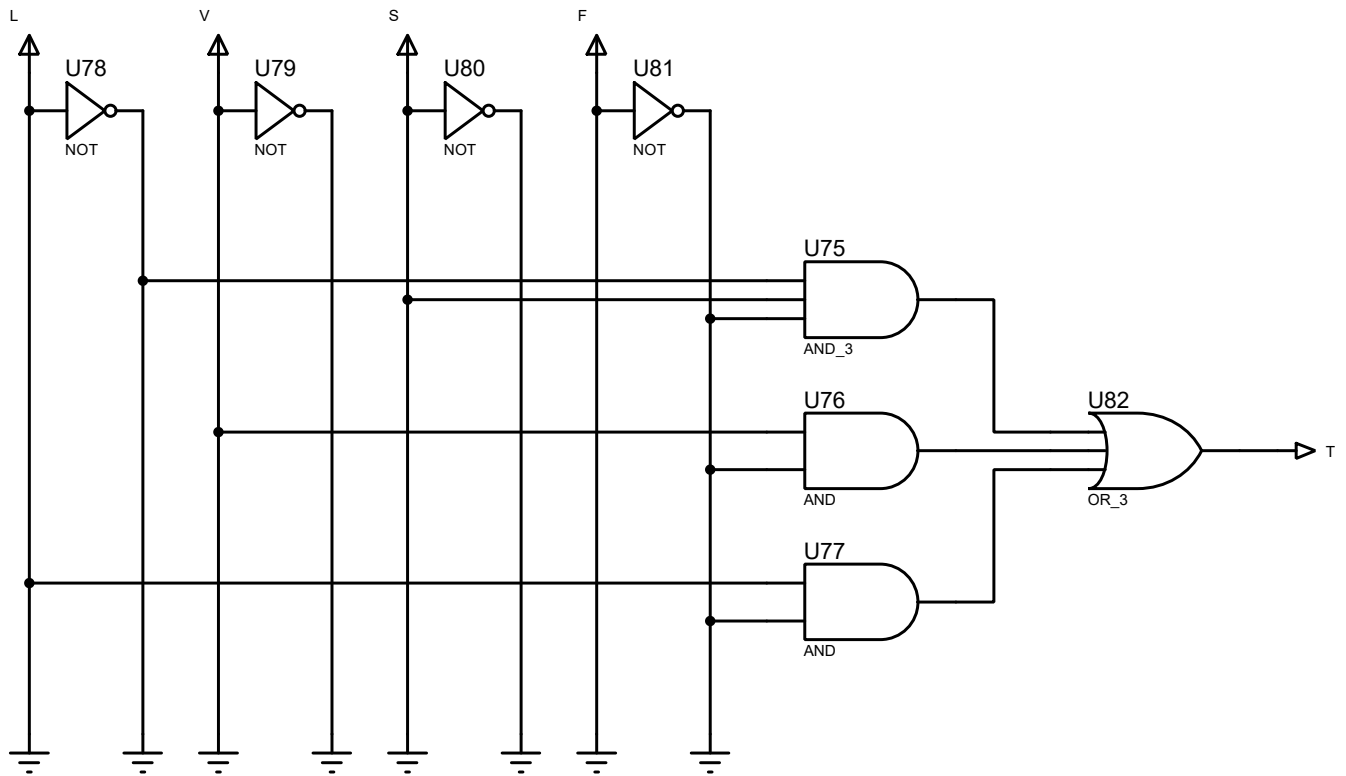
2) Correlación

J	LLUVIA MTB	VIENTO	SOL	FRIO LSB	SALIDA TOLDO
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	X (NO EXISTE)
11	1	0	1	1	X
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	X
15	1	1	1	1	X

LV \ SF	00	01	11	10
00	0	0	0	1
01	1	1	0	1
11	1	1	X	X
10	1	1	X	X

$$T = L'SF' + VS' + LS'$$

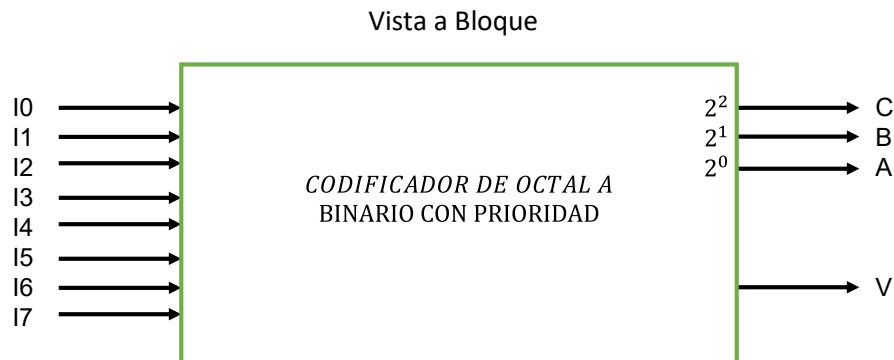
3) Diagrama Digital



El Codificador:

Es un bloque digital combinacional de MSI construido con el objetivo de producir la acción de codificar, en general ellos poseen un máximo de 2^n líneas de entrada y “n” líneas de salida.

Ejemplo de un codificador de Octal a binario con nivel de jerarquía.



NOTA:

- Que el codificador tenga nivel de jerarquía o prioridad, significa que en el hecho real que todas las entradas pudieran ser verdaderas, el codificador siempre obtendrá el binario de la entrada I0, que comúnmente es la más jerárquica o nivel más alto de prioridad, por otro lado, la entrada I7 posee el nivel más bajo en prioridad de codificación o jerarquía más baja.
- La salida “V” llamada salida de “validación” ayuda a que el sistema digital que desee tomar valores binarios codificados en las salidas “C, B, A”, pueda saber cuándo un valor en estas salidas es producto de una codificación real o un valor “ficticio” (vea la tabla de verdad).

Tabla de Verdad

VARIABLES DE ENTRADA								VARIABLES DE SALIDA				POND.
I0	I1	I2	I3	I4	I5	I6	I7	V	C	B	A	J
1	X	X	X	X	X	X	X	1	0	0	0	0
0	1	X	X	X	X	X	X	1	0	0	1	1
0	0	1	X	X	X	X	X	1	0	1	0	2
0	0	0	1	X	X	X	X	1	0	1	1	3
0	0	0	0	1	X	X	X	1	1	0	0	4
0	0	0	0	0	1	X	X	1	1	0	1	5
0	0	0	0	0	0	1	X	1	1	1	0	6
0	0	0	0	0	0	0	1	1	1	1	1	7
0	0	0	0	0	0	0	0	0	X	X	X	NA.

Nota:

- Observe como la jerarquía o prioridad asignada a las entradas cobra sentido en la codificación de ellas en las salidas “C, B, A”
- Note la tarea tan importante de la salida “V” “validación”, cuando en la última fila no existe ninguna entrada con un valor lógico verdadero “1”; la salida “V” nos dice que no importa que valor digital exista en “C, B, A”, NO debe tomarse, pues no es representativo

EL MULTIPLEXADOR:

Es un bloque digital combinacional de MSI construido con el objetivo de dirigir la información binaria existente en cualquiera de sus 2^n canales de entrada hacia su único canal de salida "Y". La elección de que canal de entrada (uno a la vez) dirigirá su información hacia el único canal de salida, se hace a través de un grupo de "n" líneas digitales conocidas como Líneas de Selección.

Ejemplo de un Multiplexador de $2^n \times 1$ (aquí " n " = 2) o sea un Multiplexador de 4 X 1

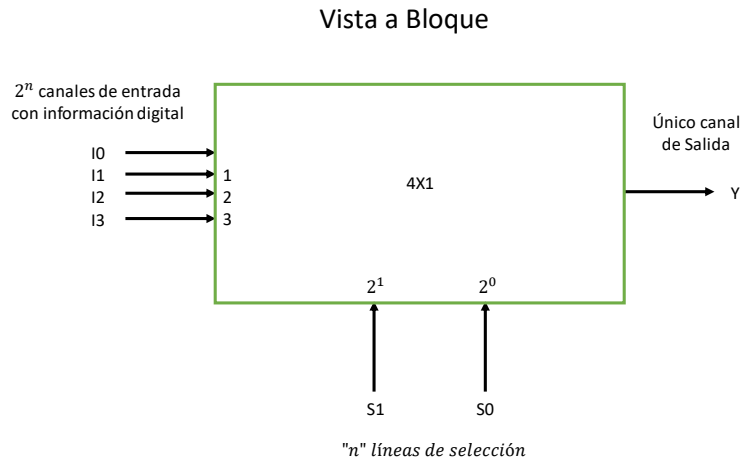


Tabla de Función

PONDERADO DECIMAL	LÍNEAS DE SELECCIÓN		CANAL DE SALIDA
J	S1	S0	Y
0	0	0	I0
1	0	1	I1
2	1	0	I2
3	1	1	I3

APLICACIÓN DEL MULTIPLEXADOR A LA IMPLEMENTACIÓN DE FUNCIONES BOOLEANAS (Método No. 3 para diseñar lógica digital combinacional)

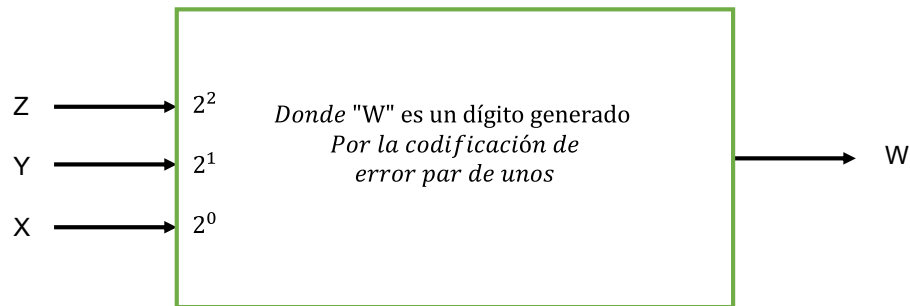
Directriz: Cualquier función booleana dependiente de " $n+1$ " variables independientes puede ser implementada por medio de un Multiplexador de $2^n \times 1$ (Un multiplexador por función)

- 1) Elegir el multiplexador adecuado para la aplicación.
- 2) Conectar las " n " variables independientes menos ponderadas a las " n " líneas de selección, guardando una correspondencia de ponderación total.
- 3) Conectar la variable de entrada más ponderada a los 2^n canales de entrada del multiplexador, todo ello de acuerdo a lo arrojado por la tabulación conocida como "tabla de configuración".
- 4) Rotular el canal único de salida del Multiplexador "Y" con el nombre de la función booleana implementada.

Ejemplo:

Aplicando el método de diseño por Multiplexador. Diseñe un circuito digital que teniendo a su entrada un numero binario de 3 bits, genere a su salida el dígito correspondiente al código de error por paridad par de unos.

- 1) BLACK BOX: La solución es obvia que se trabaja por lógica digital



- 2) ENCONTRAR LA CORRELACION de "W" versus "Z, Y, X"

Se usa tabla de verdad pues el número de variables de entrada es menor a cuatro, además la tabla de configuración solicita los términos mínimos m_j que hacen verdadera a "W"

PONDERADO DECIMAL	LÍNEAS DE SELECCIÓN			CANAL DE SALIDA
J	Z	Y	X	W
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

Concluimos que "W" (Z, Y, X) = $\sum m_1, m_2, m_4, m_7$

- 3) APLICACIÓN DEL METODO DEL MULTIPLEXADOR

3.1) Elección del Multiplexador: $n + 1 = 3$; entonces $n = 2$ por tanto el Multiplexador tiene 4 canales de entrada y un único de salida, así como 2 líneas de selección.

3.2) Tabla de configuración para la variable independiente más ponderada "Z".

	I0	I1	I2	I3
Z'	m0	m1	m2	m3
Z	m4	m5	m6	m7

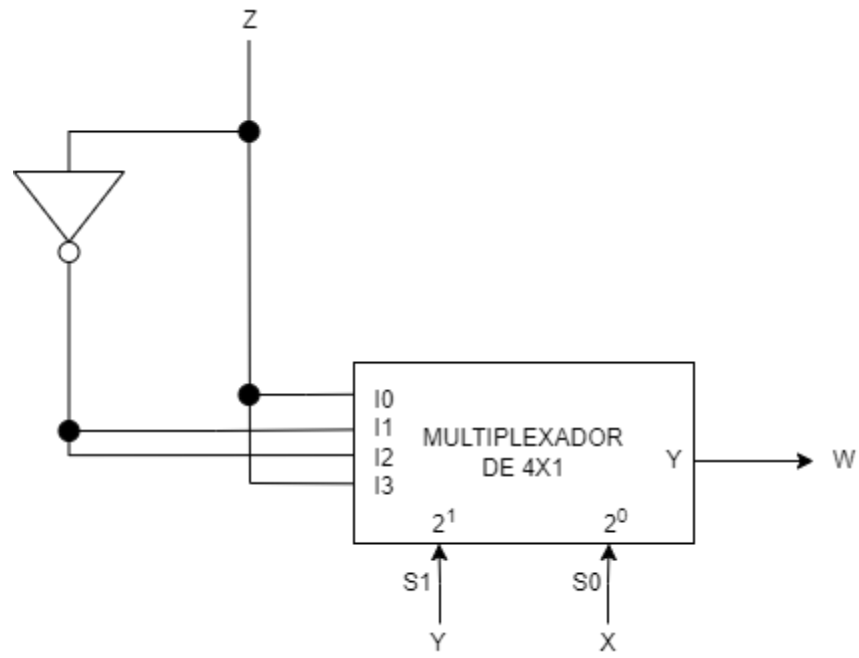
Recordando que:

$$W(Z, Y, X) = \sum m_0, m_2, m_4, m_7$$

Conclusión:

z z' z' z

3.3) El Diagrama final con Conexiones.

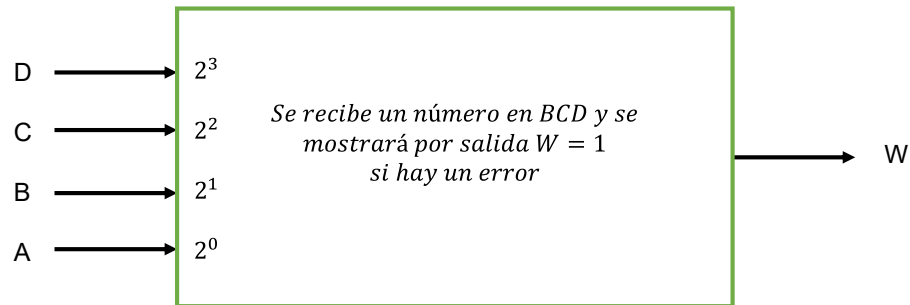


Ejercicio de Diseño con MULTIPLEXADORES

Utilizando el método de diseño con MULTIPLEXADORES, implemente la función Booleana correspondiente al siguiente problema.

Diseñar un circuito digital que teniendo a su entrada un dígito decimal codificado en BCD, nos alerte por medio de una única salida “W” de cuando exista un error en la respectiva representación.

1) Black Box



2) Tabla de Función

J	D	C	B	A	W
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	1

Se encuentra por mínimos $W = \sum m_{10}, m_{11}, m_{12}, m_{13}, m_{14}, m_{15}$

D'	m0	m1	m2	m3	m4	m5	m6	m7
D	m8	m9	m10	m11	m12	m13	m14	m15

Conclusión: 0 0 D D D D D D

MULTIPLEXADOR 8X1

3) Diagrama Final con Conexiones

