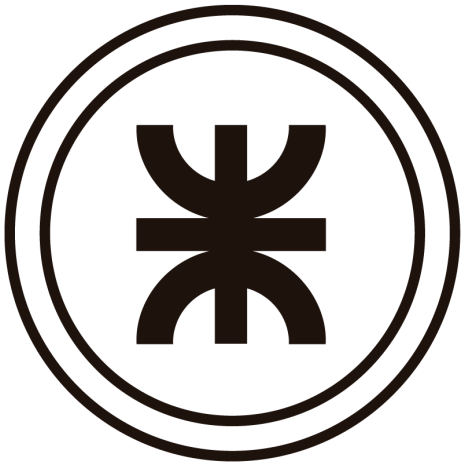
Universidad Tecnológica Nacional

Facultad Regional Buenos Aires



Técnicas Digitales I

Guía de Ejercicios

Versión 2021.0

Tabla de contenidos

Intencionalmente en blanco.

1. Álgebra de conmutación

Ejercicio 1.1.

Examine la validez de las siguientes propuestas:

1. Sean A y B **variables booleanas**, entonces $BA = 0 \wedge (B + A) = 1 \Rightarrow A = \overline{B}$.
2. Sean Z_1 y Z_2 **expresiones booleanas**, entonces $Z_2 Z_1 = 0 \wedge (Z_2 + Z_1) = 1 \Rightarrow Z_1 = \overline{Z_2}$

Ejercicio 1.2.

Haciendo uso del principio de dualidad, obtenga los teoremas duales de los siguientes teoremas:

1. $A \cdot 0 = 0$
2. $A \cdot 1 = A$
3. $A \cdot A = A$
4. $A \cdot \overline{A} = 0$
5. $(A \cdot B) \cdot C = A \cdot (B \cdot C)$
6. $A + A \cdot B = A$
7. $A + \overline{A} \cdot B = (A + B)$
8. $A \cdot B + A \cdot \overline{B} = A$
9. $\overline{A \cdot B \cdot C} = \overline{A} + \overline{B} + \overline{C}$
10. $A \cdot B + \overline{B} \cdot C + A \cdot C = A \cdot B + \overline{B} \cdot C$

Ejercicio 1.3.

Usando los teoremas del álgebra de Boole, simplifique en forma algebraica las siguientes expresiones hasta obtener formas estándar mínimas.

$$F_1 = A + B \cdot \overline{A} + C \cdot \overline{(B + A)} + D \cdot \overline{(C + B + A)}$$
$$F_2 = \overline{B} \cdot A + C \cdot A + D \cdot C \cdot B + \overline{D}$$
$$F_3 = \overline{A} + \overline{B} \cdot \overline{A} + \overline{D} \cdot C \cdot B + \overline{D} \cdot B$$
$$F_4 = C \cdot \overline{B} \cdot A + (\overline{C} + \overline{B}) \cdot (\overline{D} + \overline{B}) + \overline{(C + B + A)}$$

Ejercicio 1.4.

Muestre a través de un ejemplo sencillo las siguientes equivalencias:

$$\overline{F^D(A, B, C, ...)} = F(\overline{A}, \overline{B}, \overline{C}, ...)$$
$$F^D(A, B, C, ...) = \overline{F(\overline{A}, \overline{B}, \overline{C}, ...)}$$

*El **Teorema de Shannon** establece que el complemento de una expresión, se obtiene complementando todas las variables del dual de dicha expresión.*

Ejercicio 1.5.

El teorema de expansión de Shannon establece que cualquier función lógica puede expresarse como:

$$f(x_1, x_2, x_3, ..., x_n) = x_1 \cdot f(1, x_2, x_3, ..., x_n) + \overline{x_1} \cdot f(0, x_2, x_3, ..., x_n)$$

a) Aplíquelo en forma reiterada a la expresión de la función: $F(C, B, A) = \overline{CB} + CA$, a fin de obtener la expresión canónica SPF.

b) Escriba la expresión dual del teorema de expansión indicado arriba.

Ejercicio 1.6.

Para cada una de las siguientes funciones:

$$Z_1(D, C, B, A) = \sum_4 (0, 3, 6, 9, 12, 15)$$
$$Z_2(D, C, B, A) = \prod_4 (0, 5, 7, 13, 14, 15)$$

a) Confeccione las tablas de verdad y los mapas de Karnaugh.

b) Obtenga las expresiones canónicas SPF y PSF en sus formas numérica y literal.

c) Obtenga las expresiones mínimas SPm y PSm simplificando por el método de Karnaugh.

Ejercicio 1.7.

Una caja de caudales tiene cinco cerraduras, C1, C2, C3, C4 y C5 de modo que para tener acceso a ella, las cinco cerraduras deben estar abiertas. Las llaves para las cerraduras están asignadas a cinco personas de la siguiente forma:

- A tiene las llaves para C1 y C3.
- B tiene las llaves para C1 y C4.
- C tiene las llaves para C2 y C4.
- D tiene las llaves para C3 y C5.
- E tiene las llaves para C1 y C5.

a) Determine cuál es la cantidad mínima de personas requerida para abrir la caja.

b) Encuentre todas las combinaciones de personas que pueden abrir la caja y escriba una expresión que especifique las condiciones para poder abrirla en función de aquellas personas que estén presentes.

c) ¿Quién es la persona esencial sin la cual, la caja no se puede abrir?

Sugerencia: Ordene las variables de entrada de la siguiente manera ***E,D,C,B,A****.*

Ejercicio 1.8.

Un técnico de laboratorio químico dispone de cuatro productos (D, C, B, A), cada uno de los cuales puede encontrarlos en uno cualquiera de dos depósitos de almacenamiento. De vez en cuando, él cree conveniente cambiar uno o más productos de un depósito a otro. La naturaleza de los productos es tal que es peligroso guardar C y B juntos a menos que A esté en el mismo depósito; también es peligroso almacenar D y C juntos a menos que A esté presente.

a) Escriba una expresión suma de productos para la señal P, de modo que sea P=1 para cada situación peligrosa de almacenamiento.

b) Ahora escriba una expresión suma de productos para la señal S, tal que S=1 cuando el almacenamiento sea seguro.

Ejercicio 1.9.

Un dispositivo de cuatro señales de entrada ($I_3\ I_2\ I_1\ I_0$) denominado detector de mayoría, es un circuito combinacional cuya salida es $M = 1$ si la mayoría de las entradas son 1; de otra forma, la salida es $M = 0$.

a) Mediante una tabla de verdad, encuentre la función mayoría $M = F(I_3, I_2, I_1, I_0)$.

b) Obtenga las expresiones canónicas **SPF** y **PSF** para la función **M**.

c) Obtenga las expresiones mínimas **SPm** y **PSm** para la función **M**.

d) Repita lo solicitado en a), b) y c) para la función $\overline{M}$.

e) Repita lo solicitado en a), b) y c) para la función M^D.

2. Diseño de circuitos combinacionales con compuertas

Ejercicio 2.1.

Dé el nombre del dispositivo lógico con la cantidad de entradas indicada, cuya salida está en **H** si y sólo sí:

1. Sus dos entradas están en **L**.
2. Al menos una de sus dos entradas está en **L**.
3. Ninguna de sus dos entradas está en **L**, o ambas lo están.
4. Su única entrada está en **L**.
5. Su única entrada está en **H**.
6. Sólo una de sus dos entradas está en **H**.
7. Al menos una de sus dos entradas está en **H**.
8. Sus dos entradas están en **H**.

Ejercicio 2.2.

Verifique las siguientes propiedades de la función **O exclusiva**.

1. $A \oplus 0 = A$
2. $A \oplus 1 = \overline{A}$
3. $A \oplus A = 0$
4. $A \oplus \overline{A} = 1$
5. $\overline{B} \oplus A = B \oplus \overline{A} = \overline{B \oplus A}$
6. $Dual(B \oplus A) = (B \oplus A)^D = \overline{B \oplus A}$
7. $C \cdot (B \oplus A) = (C \cdot B) \oplus (C \cdot A)$

Ejercicio 2.3.

Diseñe un circuito que disminuya en uno una palabra binaria de 4 bits, para el cual $B = b(3)b(2)b(1)b(0)$ sea la palabra de entrada y $D = d(3)d(2)d(1)d(0)$ la palabra de salida.

1. Sintetice las salidas para obtener expresiones tipo producto de sumas empleando mapas de Karnaugh.
2. Sintetice las salidas para obtener expresiones tipo suma de productos empleando mapas de Karnaugh.

Ejercicio 2.4.

Diseñe un decodificador de Código BCD a Código 7 segmentos utilizando mapas de Karnaugh, para el cual $B = b(3)b(2)b(1)b(0)$ sea la palabra de entrada y a, b, c, d, e, f, g sean cada una de las salidas.

1. Sintetice las salidas para obtener expresiones tipo producto de sumas empleando mapas de Karnaugh.
2. Sintetice las salidas para obtener expresiones tipo suma de productos empleando mapas de Karnaugh.
3. Analice las expresiones de cada salida y determine para cada una si la implementación *AND/OR* es más simple que la *OR/AND*.

Ejercicio 2.5.

Diseñe un circuito reciba una palabra BCD como entrada y que a su salida duplique el valor recibido. Recuerde que una palabra BCD tiene una longitud de 4 bits y representa los dígitos decimales del 0 al 9 (**"0000"** al **"1001"**). El resto de las combinaciones binarias del 10 al 15 (**"1010"** al **"1111"**) no representan dígitos válidos y por lo tanto no producirán resultados válidos.

Encontrará casos en los que al duplicar un dígito BCD no podrá representarlos con un único dígito, por ejemplo $0110_{BCD} \times 2 = 0001_{BCD}\ 0010_{BCD}$ $(6 \times 2 = 12)$. Para estos casos el circuito solo generará una palabra de 4 bits (**"0010"** en este ejemplo) y activará una salida de acarreo **Co**.

Para el diseño utilice la siguiente nomenclatura:

1. $B = b(3)b(2)b(1)b(0)$ debe ser la palabra de entrada,
2. $D = d(3)d(2)d(1)d(0)$ debe ser la palabra de salida y
3. Co debe ser la salida de acarreo.

Ejercicio 2.6.

1. Diseñe utilizando mapas de Karnaugh un circuito que convierta una palabra en Código binario natural a Código Gray de 4 bits. Utilice $B = b(3)b(2)b(1)b(0)$ para identificar la palabra de entrada y $G = g(3)g(2)g(1)g(0)$.
2. Diseñe utilizando mapas de Karnaugh un circuito que convierta una palabra en Código Gray de 4 bits a Código binario natural. Utilice $G = g(3)g(2)g(1)g(0)$ para identificar la palabra de entrada y $B = b(3)b(2)b(1)b(0)$.

Ejercicio 2.7.

Diseñe un circuito que multiplique por 3 una palabra binaria de 3 bits. Obtenga una implementación *OR/AND*.

Utilice $B = b(2)b(1)b(0)$ para indicar la palabra de entrada y determine la longitud de la palabra T de salida.

Ejercicio 2.8.

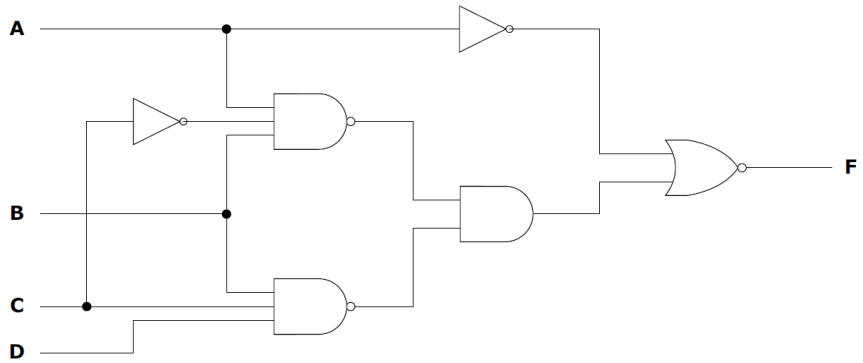
Diseñe un multiplicador combinacional de dos números enteros positivos de dos bits **b** y **a**. La salida **p** del circuito será entonces el producto **b x a**.

1. Obtenga la tabla de verdad del multiplicador combinacional (formato: **b1 b0 a1 a0 p**).
2. Realice la síntesis empleando mapas de Karnaugh, a fin de obtener las expresiones mínimas como suma de productos que representen el vector de salida p.

Ejercicio 2.9.

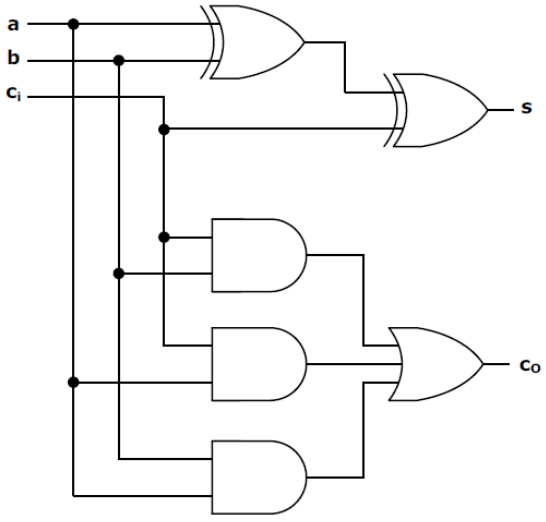
Para el circuito de la figura:

1. Obtenga la expresión de salida empleando el método de análisis algebraico.
2. Obtenga nuevamente la expresión de salida transformando el circuito y empleando al mínimo las leyes del álgebra de conmutación.



Ejercicio 2.10.

Demuestre que el circuito de la figura es una posible implementación de una celda sumadora completa.



Ejercicio 2.11.

1. Para cada circuito obtenido en el ejercicio 2.3, calcule el costo por literal (C_l) y el costo por entrada (C_{gi}). Luego determine cual de ellos resulta ser la mejor implementación.
2. Repita el punto anterior, pero aplicado ahora a todos los circuitos obtenidos en el ejercicio 2.4.

Ejercicio 2.12.

Reduzca las siguientes expresiones booleanas al número de literales indicados.

1. $\overline{X}.\overline{Y} + X.Y.Z + \overline{X}.Y \quad (3\ literales)$
2. $X + Y.(Z + \overline{X + Z}) \quad (2\ literales)$
3. $\overline{W}.X.(\overline{Z} + \overline{Y}.Z) + X.(W + \overline{W}.Y.Z) \quad (1\ literal)$
4. $(A.B + \overline{A}.\overline{B}).(C.D + \overline{C}.\overline{D}) + \overline{A}.\overline{C} \quad (4\ literales)$

Ejercicio 2.13. *Resolución opcional.*

Aplique descomposición para encontrar el menor costo por entrada para las siguientes funciones. La implementación final sólo debe constar de compuertas AND, OR y NOT.

1. $F(A, B, C, D) = A.\overline{B}.C + \overline{A}.B.C + A.\overline{B}.D + \overline{A}.B.D$
2. $G(W, X, Y, Z) = W.Y + X.Y + \overline{W}.X.Z + W.\overline{X}.Z$

Ejercicio 2.14. *Resolución opcional.*

Aplique extracción para encontrar el menor costo por entrada para las siguientes funciones. La implementación final sólo debe constar de compuertas AND, OR y NOT.

1. $F(A, B, C, D) = \sum m(0, 5, 11, 14, 15), \quad d(A, B, C, D) = \sum m(10)$
2. $G(A, B, C, D) = \sum m(2, 7, 10, 11, 14), \quad d(A, B, C, D) = \sum m(15)$

3. Tecnologías y estándares del hardware digital

Ejercicio 3.1. Tecnología CMOS

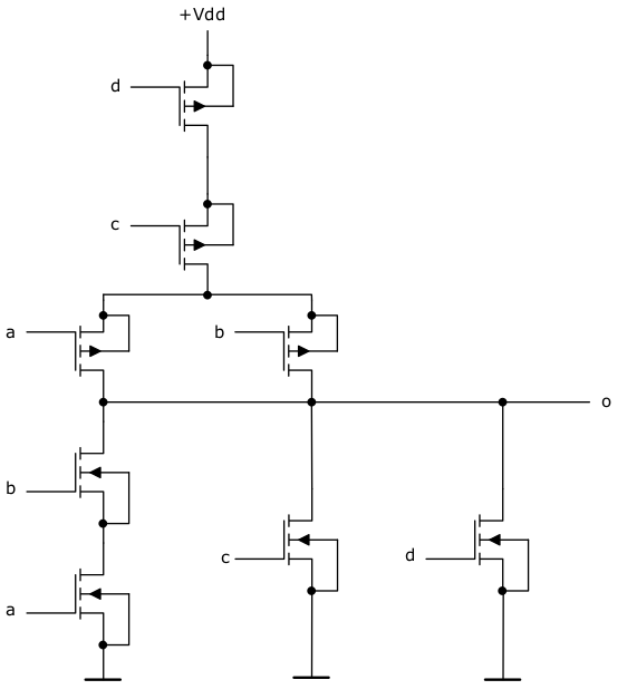
Dibuje las siguientes compuertas a nivel transistor con tecnología CMOS

- NOT
- NAND
- NOR
- AO21
- OA21
- AOI22
- OAI22

Ejercicio 3.2. Tecnología CMOS

Para la celda construida con MOSFET

- Construya su tabla de verdad expresada en niveles de tensión (H y L).
- Construya su tabla de verdad expresada con 1s y 0s.
- Obtenga la expresión lógica de la salida.



Ejercicio 3.3. Tecnología CMOS

Estudie la **compatibilidad** entre los siguientes estándares digitales. Para cada caso en particular, considere que dispone de dos dispositivos **A** y **B**, de modo tal que **A** está conectado con **B**, y **A** pertenece a los estándares *fila* y **B** pertenece a los estándares *columna*.

	TTL	CMOS	LVTTL	LVCMOS	2.5CMOS	1.8CMOS
TTL						
CMOS						
LVTTL						
LVCMOS						
2.5CMOS						
1.8CMOS						

Ejercicio 3.4. Consumo de energía

Calcule el **consumo estático**, bajo las condiciones especificadas a continuación, de un **inversor NMOS** si V_{DD} es $+5V$ y su resistor de *pull-up* tiene una resistencia equivalente de 40 $k\Omega$.

- La entrada del inversor es un nivel bajo.
- La entrada del inversor es un nivel alto.
- La entrada del inversor es una señal rectangular con un *duty cycle* del 50 %.
- La entrada del inversor es una señal rectangular con un *duty cycle* del 10 %.
- La entrada del inversor es una señal rectangular con un *duty cycle* del 90 %.

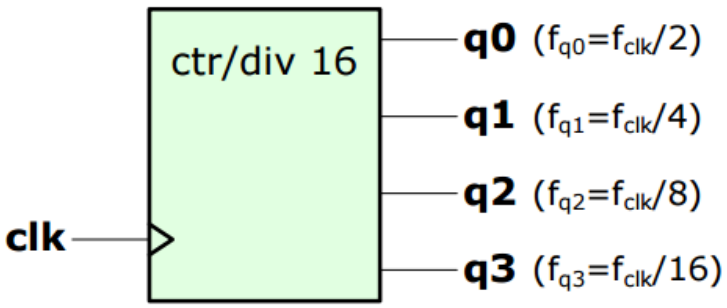
Ejercicio 3.5. Consumo de energía

Considere que la corriente de *leakage* en una tecnología 1.8CMOS es aproximadamente 1 nA por transistor. Estos transistores tienen en promedio una conmutación cada 100 ns y manejan una carga de salida de FO4 (*fanout of 4*). Calcule el consumo total, dinámico mas estático, tomando $R_P = R_N = 5{,}5\ k\Omega$ y $C_P = 10\ fF$.

Compruebe que si la frecuencia de las conmutaciones disminuye a 1000 ns por conmutación el consumo total se reduce. ¿A cuánto debe bajar la frecuencia de conmutación para que el consumo dinámico sea igual al estático?

Ejercicio 3.6. Consumo de energía

La figura representa a un contador binario CMOS, con una entrada **clk** y cuatro salidas **q0 q1 q2 q3**. Estime la potencia disipada total **PD** teniendo en cuenta que la tensión de alimentación es **VCC = 5V**, la capacidad de disipación de potencia es **Cpd = 45 pF**, la corriente de reposo es **Icc = 40 uA** y la frecuencia de la señal de entrada es de **f = 10 MHz** con ciclo de actividad del **50 %**. Además suponga que cada salida tiene asociada una capacidad de carga **Cl = 40 pF**.



Ejercicio 3.7. Riesgos

Responda a las siguientes preguntas

1. Nombre qué tipos de riesgos lógicos pueden existir en un circuito combinacional.
2. Explique qué produce su presencia a la salida de un circuito combinacional.
3. Ilustre, por medio de formas de ondas, los 4 tipos de riesgos lógicos posibles.
4. Explique la diferencia entre riesgo lógico y *glitch*.
5. ¿Pueden eliminarse los riesgos lógicos de un circuito combinacional? Explique cómo.

Ejercicio 3.8. Riesgos

Considere la función $f = \overline{C}A + CB$

1. Implemente f como suma de productos.
2. Asumiendo que las variables del circuito en el instante 0 adoptan los siguientes valores $C = B = A = 1$, analice la existencia de un riesgo estático cuando la variable C cambia de 1 a 0.
3. Realice un diagrama de las formas de onda para cada señal del circuito. Incluya las variables de entrada A, B y C, la función f, como así también las señales intermedias $\overline{C}$, (CB) y $(\overline{C}A)$. Considere que el tiempo de propagación de cada compuerta es unitario.
4. En caso de haber encontrado un riesgo estático, explique como solucionaría el problema encontrado y demuestre por medio de un diagrama formas de onda que el problema es resuelto.

Ejercicio 3.9. Riesgos

Considere la función $f = (C + \overline{A}) \cdot (\overline{C} + \overline{B})$

1. Implemente f como producto de sumas.
2. Asumiendo que las variables del circuito en el instante 0 adoptan los siguientes valores $C = 0$ y $B = A = 1$, analice la existencia de un riesgo estático cuando la variable C cambia de 0 a 1.
3. Realice un diagrama de las formas de onda para cada señal del circuito. Incluya las variables de entrada A, B y C, la función f, como así también las señales intermedias $\overline{A}$, $\overline{B}$, $\overline{C}$, $(C + \overline{A})$ y $(\overline{C} + \overline{B})$. Considere que el tiempo de propagación de cada compuerta es unitario.
4. En caso de haber encontrado un riesgo estático, explique como solucionaría el problema encontrado y demuestre por medio de un diagrama formas de onda que el problema es resuelto.

4. Aritmética binaria y circuitos aritméticos

Ejercicio 4.1. Operación de suma con operandos no signados

Realice cada una de las sumas e indique en cada caso

- el resultado de la suma en n-bits, donde n es el ancho de palabra de los operandos
- si hubo o no acarreo de salida

a) $11_2 + 01_2$
b) $1010_2 + 1011_2$
c) $11111111_2 + 00000001_2$
d) $10101010_2 + 10111011_2$

Ejercicio 4.2. Operación de suma con operandos no signados

a) Diseñe un circuito combinacional en dos niveles de compuerta, que en su entrada reciba una magnitud binaria de cuatro bits $B = B(3)B(2)B(1)B(0)$ y en su salida genere la suma aritmética $O = B + 1$.

b) Tomando como base las expresiones obtenidas en el punto anterior, determine si es posible implementar una celda incrementadora para un bit, que permita realizar un incrementador combinacional iterativo para magnitudes de cualquier número de bits. (Sugerencia: transforme de manera algebraica las expresiones para descubrir si existe un patrón regular de generación).

c) En caso afirmativo, obtenga el circuito interno de la celda y muestre cómo conectarlas a fin de obtener un incrementador para magnitudes de cuatro bits.

d) ¿Qué ventajas y desventajas tiene la realización iterativa respecto a la implementación en dos niveles de compuerta? Justifique su respuesta.

Ejercicio 4.3. Operación de suma con operandos no signados

Partiendo de un sumador de 4 bits con acarreo (ripple carry adder), diseñe un circuito que incremente en 2 un número binario no signado de 4 bits. La función a implementar debe ser entonces $S = A + 2$. Demuestre en forma algebraica la reducción del circuito.

Nota: para comparar las implementaciones puede utilizar el concepto de costo por entrada.

Ejercicio 4.4. Operación de resta con operandos no signados

Realice cada una de las restas e indique en cada caso

- el resultado de la resta en n-bits, donde n es el ancho de palabra de los operandos
- si hubo o no préstamo en la étapa más significativa

a) $11111_2 - 10000_2$
b) $10110_2 - 11111_2$
c) $1011110_2 - 1011110_2$
d) $101_2 - 101000_2$

Ejercicio 4.5. Operación de resta con operandos no signados

Obtenga los complemento a 1 y a 2 de los siguientes números binarios no signados:

a) 10011100
b) 10011101
c) 10101000
d) 01111111
e) 00000000
f) 10000000

Ejercicio 4.6. Operación de resta con operandos no signados

Diseñe dos versiones de un circuito combinacional cuya entrada **X** sea un número de 4 bits y cuya salida **Y** sea el complemento a 2 del número de entrada, para cada uno de los siguientes casos:

a) El circuito debe ser un circuito simplificado a dos niveles, más los inversores necesarios para las variables de entrada.

b) El circuito debe estar formado por cuatro celdas idénticas de dos entradas y dos salidas, una para cada bit del número de entrada. Las celdas deben conectarse en cascada, con conexiones entre ellas similares a un acarreo. Utilice la letra **P** (propagación) para representar las señales que realizan estas conexiones. Determine cual es la condición de frontera del circuito.

Ejercicio 4.7. Operación de resta con operandos no signados

Realice cada una de las restas tomando el complemento a 2 del sustraendo e indique en cada caso

- el resultado de la resta en n-bits, donde n es el ancho de palabra de los operandos
- si hubo o no préstamo en la étapa más significativa

a) $11111_2 - 10000_2$
b) $10110_2 - 11111_2$
c) $1011110_2 - 1011110_2$
d) $101_2 - 101000_2$

Ejercicio 4.8. Operación de multiplicación con operandos no signados

Diseñe un multiplicador binario que multiplique dos números **A** y **B** sin signo de 4 bits. Para ello emplee compuertas AND y sumadores binarios. Llame **P** al producto de **A**por **B**.

Ejercicio 4.9. Operación de multiplicación con operandos no signados

Partiendo del circuito resultante del ejercicio anterior, diseñe un circuito que multiplique un número de 4 bits por la constante 10_{10}. La función a implementar debe ser entonces $P = 10 \cdot A$. Demuestre en forma algebraica la reducción del circuito.

Nota: para comparar las implementaciones puede utilizar el concepto de costo por entrada.

Ejercicio 4.10. Operación de multiplicación con operandos no signados

Diseñe un circuito que multiplique un dato de 4 bits por la constante $1000_2 = 8_{10}$. Llame **A** al operando de entrada y **P** al producto resultante.

Ejercicio 4.11. Operación de suma y resta con operandos signados

Calcule las siguientes operaciones aritméticas en el sistema de numeración binario.

a) $(+36)_{10} + (24)_{10}$

b) $(+36)_{10} + (-24)_{10}$

c) $(+36)_{10} - (24)_{10}$

d) $(+36)_{10} - (-24)_{10}$

- Tenga presente que, en cada caso, el sufijo indica la base del sistema de numeración.
- Convierta los números en base decimal a base binaria.
- Indique en cada caso si se produce desbordamiento.

Ejercicio 4.12. Operación de suma y resta con operandos signados

Diseñe un circuito que reciba una palabra de entrada de 8 bits y en su salida la expanda a 16 bits. Considere que la palabra de entrada representa un número y determine que diferencias existen si el número en cuestión es signado o no. Llame **X** al número de entrada al circuito y **Y** a la salida del mismo.

Ejercicio 4.13. Operación de suma y resta con operandos signados

Si un sumador binario de cuatro bits se emplea con operandos enteros representados en código de complemento a dos, la etapa más significativa del sumador corresponderá a los bits de signo.

a) Realice una tabla de verdad para el sumador de bits de signo usando nombres de señales asociados a esa etapa, que incluya una salida adicional para la señal oVerflow (**V**). Llame **A** y **B** a los operandos, y **C** a la señal que propaga el acarreo.

b) Obtenga las expresiones de V en función de los signos de los operandos y del signo del resultado.

c) Obtenga las expresiones de V en función de los signos de los operandos y del acarreo de entrada a la etapa.

d) Obtenga las expresiones de V en función de los signos de los operandos y de los acarreos de entrada y salida a la etapa.

Ejercicio 4.14. Operación de suma y resta con operandos signados

Diseñe un circuito sumador cuyos operandos (**A** y **B**) sean de 8 bits y que disponga de una señal de modo **M**, de forma tal que:

1. Si **modo = '0'** los operandos sean considerados como magnitudes.
2. Si **modo = '1'** los operandos sean considerados como signados.

El circuito debe ser capaz de detectar las condiciones de error en cada una de las operaciones. Explique además la diferencia entre *carry* (**C**) y *overflow* (**V**), y proponga ejemplos ilustrativos que demuestran las diferencias.

Ejercicio 4.15. Comparador por contracción

Diseñe un circuito combinacional que compare dos números **A** y **B** sin signo de 4 bits y que determine si **A** es mayor que **B**. El circuito deberá tener una salida **X**, de modo tal que $\mathbf{X = 1}$ si **A** es mayor que **B**, y $\mathbf{X = 0}$ en cualquier otro caso.

Para ello utilice un restador de 4 bits. Analice como debe manipular las entradas y salidas del restador para lograr el mismo funcionamiento que el circuito anterior.

Ejercicio 4.16. Comparador combinacional

a) Diseñe un comparador de dos números enteros **P(1:0)** y **Q(1:0)** de dos bits **representados en código binario desplazado**. El comparador deberá generar tres salidas:

   (1) **PMAQ** $(p > q)$
   (2) **PIGQ** $(p = q)$
   (3) **PMEQ** $(p < q)$

b) Repita el diseño para el caso que los números **P(1:0)** y **Q(1:0)** sean enteros **representados en código de complemento a dos**.

Ejercicio 4.17. Asignación, transferencia y complemento

a) Dibuje un diagrama que implemente la función

$$G = (G(7), G(6), G(5), G(4), G(3), G(2), G(1), G(0)) = (A, \overline{A}, 0, 1, \overline{A}, A, 1, 1)$$

   usando los símbolos de masa, tensión de alimentación e inversores.

b) Dibuje un diagrama que implemente la función

$$F = (F(7), F(6), F(5), F(4), F(3), F(2), F(1), F(0)) = (1, 0, A, \overline{A}, \overline{A}, A, 0, 1)$$

   usando los símbolos de masa, tensión de alimentación e inversores.

Ejercicio 4.18. Habilitación

Un sistema de seguridad doméstico tiene un interruptor principal que se usa para habilitar una alarma, luces, cámaras de video, y una llamada a la policía local en el caso de que uno o más de seis juegos de sensores detecte a un intruso. Las entradas, salidas, y las operaciones de habilitación lógicas se especifican a continuación:

Entradas:

- $S_i,\ i = 0, 1, 2, 3, 4, 5$, señales de sensores (1 - detectado intruso, 0 - ningún intruso).
- **P**, el interruptor principal (1 - sistema de seguridad encendido, 0 - sistema de seguridad apagado).

Salidas:

- **A** - alarma (1 - alarma encendida, 0 - alarma apagada)
- **L** - luces (1 - luces encendidas, 0 - luces apagadas)
- **V** - cámaras de video (0 - cámaras apagadas, 1 - cámaras encendidas)

- **M** - llamada a la policía (0 - no llamar, 1 - llamar)

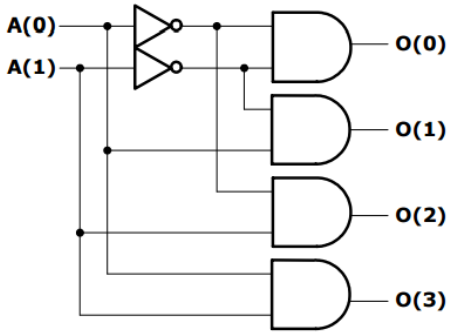
Funcionamiento:
Si uno o más de los juegos de sensores descubre un intruso y el sistema de seguridad esta encendido, entonces todas las salidas están activadas. De lo contrario, todas las salidas estarán apagadas. Encuentre la implementación con puertas AND, OR e inversores de este circuito de habilitación que minimiza el número total de entradas de puertas.

Ejercicio 4.19. Decodificación I

---

a) Analice el circuito combinacional de la figura, a fin de obtener las expresiones de las salidas $O(3)O(2)O(1)O(0)$ en función de las entradas $A(1)A(0)$.

b) Describa con sus propias palabras la operación del circuito.



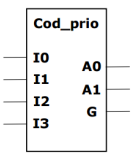
Ejercicio 4.20. Decodificación II

Se pretende a diseñar un decodificador especial 3 a 6 con habilitación, tal que los códigos de la entrada sean desde 000 hasta 101. Para un código dado, la salida D_i, con i igual al equivalente decimal del código, es 1 y todas las demás salidas son 0. Diseñe el decodificador empleando un decodificador 2 a 4, un decodificador 1 a 2, y compuertas AND de 2-entradas, tal que las salidas de todos los decodificadores se usen por lo menos una vez.

Ejercicio 4.21. Codificación

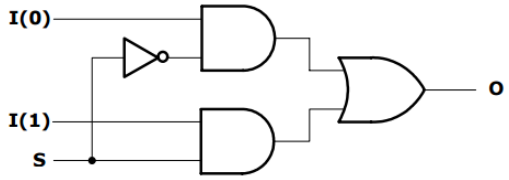
Diseñe un codificador binario con prioridad, cuya entrada sea una palabra binaria de 4 bits y cuya salida incluya, además de la palabra codificada en 2 bits, la señal de grupo G. Dicha señal debe activarse siempre que a la entrada del codificador exista al menos una entrada activa. Explique por qué es relevante la inclusión de la señal de grupo.

I3	I2	I1	I0	A1	A0	G
0	0	0	0	0	0	0
1	–	–	–	1	1	1
0	1	–	–	1	0	1
0	0	1	–	0	1	1
0	0	0	1	0	0	1



Ejercicio 4.22. Selección I

Analice el circuito combinacional de la figura, a fin de obtener la expresión de la salida O en función de las entradas S, $I(1)$ e $I(0)$. Describa con sus propias palabras la operación del circuito.



Ejercicio 4.23. Selección II

a) Diseñe un circuito con cuatro entradas de datos de un bit $I(3)I(2)I(1)I(0)$ y una salida O, de modo que con dos entradas de control adicionales $S(1)S(0)$ se pueda seleccionar que sólo una entrada de datos sea copiada en la salida.

b) Agregue al circuito obtenido una entrada de habilitación EN que cuando se encuentre activa ($EN = 1$) el circuito opere como se describió anteriormente y cuando no se encuentre activa ($EN = 0$) imponga $O = 0$ independientemente del estado lógico de las otras entradas.

Ejercicio 4.24. Selección III

Diseñe un multiplexor 8 a 1 a partir de compuertas de transmisión. Llame **X** al vector de entradas, **S** a la señal de selección y **Y** a la señal de salida.

Ejercicio 4.25. Selección IV

Demuestre que un demultiplexor es un decodificador con habilitación.
Nota: Si le facilita el trabajo, considere un decodificador binario de 2 a 4 bits.

Ejercicio 4.26. *Opcional:* Descomposición funcional - Teorema de Shannon

Dada la función de cuatro variables $F = (D + C + A) \cdot (C + B + A) \cdot (\bar{C} + B + \bar{A}) \cdot (\bar{D} + \bar{B} + \bar{A})$, impleméntela con:

a) Un multiplexor de 8 canales.

b) Un multiplexor de 4 canales.

c) Un multiplexor de 2 canales.

En todos los casos, debe emplear un solo multiplexor y mínima lógica adicional.

Ejercicio 4.27. *Opcional:* Descomposición funcional de Ashenhurst

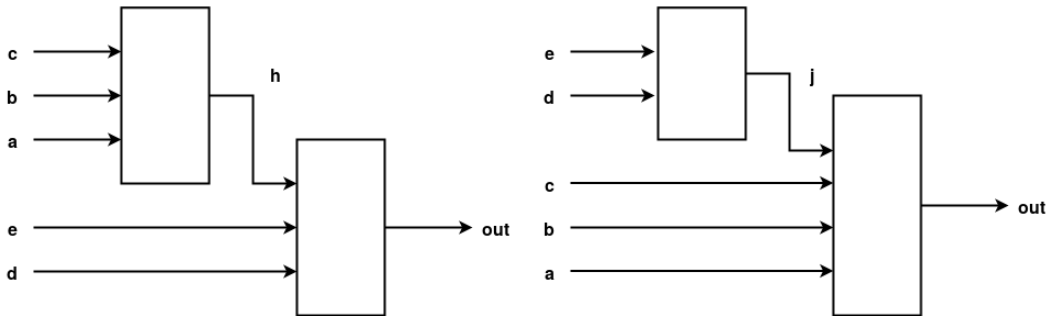
El circuito de la figura (a) está formado por 2 LUTs de 3 entradas y una salida y está basado en una descomposición disyunta de Ashenhurst. Sabiendo que las funciones **h** y **out** responden a las siguientes tablas de verdad,

h	e	d	**out**	c	b	a	**h**
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	1
0	1	0	1	0	1	0	0
0	1	1	1	0	1	1	1
1	0	0	1	1	0	0	0
1	0	1	1	1	0	1	1
1	1	0	0	1	1	0	1
1	1	1	0	1	1	1	1

a) Construya la tabla de verdad del circuito (salida **out**).

b) Construya el mapa de Karnaugh de **out**.

c) Determine si es posible hacer la descomposición que sugiere el circuito de la figura (b).



(a) (b)

Ejercicio 4.28. *Opcional:* Implementación alternativa de funciones lógicas

a) Implemente un sumador completo con dos multiplexores 4 a 1 y un único inversor.

b) Implemente la función booleana siguiente con un multiplexor 8 a 1 y un único inversor con la variable D como entrada: $F(A,B,C,D) = \sum m(2,4,6,9,10,11,15)$.

c) Implemente la función booleana $F(A,B,C,D) = \sum m(1,3,4,11,12,13,14,15)$ con un multiplexor 4 a1 y compuertas. Conecte las entradas A y B a las líneas de selección. Las entradas necesarias para las cuatro líneas de datos serán función de las variables C y D. Los valores de estas variables se obtienen expresando F como una función de C y D para cada uno de los cuatro casos cuando AB igual 00, 01, 10, y 11.

d) Dada la función $q = \overline{e}.\overline{d} + c.b.a$, impleméntela usando la menor cantidad posible de multiplexores de dos canales y un sólo inversor.

e) Dada la función $q = \overline{e}.\overline{d} + c.b.a$, impleméntela usando sólo dos LUTs de tres entradas. Confeccione la tabla de verdad de ambas LUTs indicando en cada caso la salida de la LUT en función de sus señales de entrada.

Ejercicio 4.29. Síntesis RTL

Diseñe un circuito combinacional tal que su entrada $i(2:0)$ sean números enteros representados en el código signo y magnitud y su salida $o(2:0)$, represente a los mismos enteros pero en el código de complemento a dos.

a) Confeccione la tabla de verdad del conversor.

b) Implemente las funciones del vector de salida o usando bloques multiplexores de dos canales y mínima lógica adicional.

c) Dibuje el circuito resultante.

Ejercicio 4.30. Síntesis RTL

Se desea diseñar un circuito combinacional que reciba en su entrada un número entero representado en el código de complemento a 2, $e(3:0)$, y genera en su salida $m(3:0)$ el valor absoluto del número de entrada.

a) Obtenga la tabla de verdad del dispositivo y analícela.

b) Implemente el circuito usando la cantidad mínima de celdas incrementadoras de 1 bit, multiplexores de 2 canales e inversores.

Ejercicio 4.31. Síntesis RTL

Considere el siguiente circuito de ruteo. El mismo posee dos puertos de entrada, **x(0)** y **x(1)**, una señal de control **ctrl** de 2 bits. Las señales de entradas se rutean a las salidas **y(0)** e **y(1)** en función de la señal **ctrl**, de acuerdo con la siguiente tabla de funcionamiento:

ctrl	**y(1)**	**y(0)**
00	x(1)	x(0)
01	x(0)	x(1)
10	x(0)	x(0)
11	x(1)	x(1)

a) Dibuje un diagrama RTL del circuito.

b) Expanda el diagrama del punto anterior a fin de obtener un circuito a nivel de compuertas. Luego derive la expresión lógica simplificada del mismo en forma de suma de productos.

Ejercicio 4.32. Síntesis RTL

a) Construya un circuito que recibe 2 palabras de W bits y entrega en su salida la menor de ellas.

b) Construya un circuito que recibe 2 palabras de W bits y entrega en su salida la mayor de ellas.

c) Construya un circuito que en función en señal de control **ctrl**, entregue en su salida la mayor de dos palabras de entradas de W bits siempre que **ctrl** sea igual a 1, en tanto que si **ctrl** es igual a 0, el circuito deberá entregar en su salida la menor de las dos palabras de entrada.

Ejercicio 4.33. Síntesis RTL

a) Diseñe un circuito que realice la siguiente suma

$$res = a + b \; si \; s = 0$$
$$res = a + c \; si \; s = 1$$

cuyos operandos dependen de la entrada de selección s. Solo dispone de un único circuito sumador para construirlo.

b) Diseñe un circuito que realice la siguiente suma

$$res = a + b \; si \; s = 0$$
$$res = c + d \; si \; s = 1$$

cuyos operandos dependen de la entrada de selección s. Solo dispone de un único circuito sumador para construirlo.

Ejercicio 4.34. Síntesis RTL

a) Diseñe un circuito combinacional que produzca una salida **O = abs(A-B)**. Considere que los vectores de entrada al circuito son de longitud **W** y que los mismos representan **números no signados**. Determine además la longitud del vector de salida **O**.

b) Repita el punto anterior, pero ahora considere que los vectores de entrada representan **números signados en código de complemento a 2**.

Ejercicio 4.35. Síntesis RTL

Implemente un circuito que permita calcular el promedio de 4 números binarios signados en CA2 de 4 bits cada uno. Justifique la cantidad de bits necesarios para operar y calcular correctamente el promedio. Trunque la parte decimal del número obtenido.

Ejercicio 4.36. Síntesis RTL

Se pretende construir un circuito combinacional cuya interfaz es la siguiente

Nombre del puerto	Tipo	Longitud de palabra
bin	Entrada	8
acc_curr	Entrada	16
mode	Entrada	1
acc_next	Salida	16
sat	Salida	1

El funcionamiento del mismo es tal que si la señal **mode es igual a 0** la salida **acc_next** debe ser igual a **acc_curr + bin**, en tanto que si **mode es igual a 1** la salida será **acc_curr - bin**. Tenga en cuenta que las operaciones aritméticas se realizan sobre operandos signados, y que además el circuito debe ser capaz de saturar el resultado siempre que sea pertinente, en cuyo caso la señal **sat** deberá indicar, en lógica positiva, dicha condición.

Ejercicio 4.37. Síntesis RTL

Se pretende construir un circuito combinacional cuya interfaz es la siguiente

Nombre del puerto	Tipo	Longitud de palabra
X	Entrada	4
Y	Entrada	2
R	Salida	A determinar

El funcionamiento del mismo es tal que para las palabras no signadas de entrada **X** e **Y**, el circuito produce en su salida **R** el **resto** de la división **X/Y**.

a) Dimensione la cantidad de bits de la palabra **R**.

b) Adopte un criterio de diseño para el caso en que la entrada **Y** sea igual **"00"**. Justifique su decisión.

c) Dibuje un diagrama RTL del circuito.

Ejercicio 4.38. Síntesis RTL

Diseñe un circuito que calcule la raíz cuadrada (sin la parte fraccionaria) y su resto para una palabra entrante de 4 bits. Llame **X** a la palabra de entrada, **SQRT** a la raíz cuadrada de **X** y **REM** al resto.

a) Dimensione la cantidad de bits de las palabras de salida **SQRT** y **REM**.

b) Implemente usando multiplexores de 3 entradas de selección y compuertas NOT.

c) Proponga una implementación alternativa a la utilizada en el punto anterior.

Ejercicio 4.39. Desplazamientos

Diseñe un circuito que reciba una palabra de entrada de 8 bits y a su salida la desplace lógicamente una posición a la derecha. Además determine,

a) Si la palabra de entrada representa un número no signado, ¿a que operación aritmética corresponde este desplazamiento?

b) ¿Cómo conectaría múltiples desplazadores en cascada?

c) Si la palabra de entrada representa un número signado, es de esperarse que la operación aritmética se conserve. Investigue si esto es cierto, caso contrario, diseñe un circuito que permita realizar la misma operación cuando la palabra de entrada represente un número signado.

d) ¿Existe alguna condición indeseada para el circuito que ha diseñado?

Ejercicio 4.40. Desplazamientos

Diseñe un circuito que reciba una palabra de entrada de 8 bits y a su salida la desplace lógicamente una posición a la izquierda. Además determine,

(a) Si la palabra de entrada representa un número no signado, ¿a que operación aritmética corresponde este desplazamiento?

(b) ¿Cómo conectaría múltiples desplazadores en cascada?

(c) En caso que para una misma palabra, efectúe un desplazamiento múltiple, ¿puede producirse una condición de error? ¿En qué casos? ¿Cómo lo detecta?

(d) Si la palabra de entrada representa un número signado, es de esperarse que la operación aritmética se conserve. Investigue si esto es cierto, caso contrario, diseñe un circuito que permita realizar la misma operación cuando la palabra de entrada represente un número signado.

(e) ¿Existe alguna condición indeseada para el circuito que ha diseñado?

Ejercicio 4.41. Desplazamientos

Diseñe un circuito que divida un dato de 8 bits entre la constante $1000_2 = 8_{10}$ dando un cociente de 8 bits y un resto de 8 bits. Llame **A** al dividendo, **C** al cociente y **R** al resto.

Ejercicio 4.42. Rotación

a) Diseñe un circuito que reciba una palabra de entrada de 4 bits y a su salida la rote una posición a la izquierda.

b) Diseñe un circuito que reciba una palabra de entrada de 4 bits y a su salida la rote una posición a la derecha.

c) Diseñe un circuito que reciba una palabra de entrada de 4 bits y que en función de una señal de selección $R/\bar{L}$, a su salida la rote a izquierda ($R/\bar{L} = 0$) o derecha una posición ($R/\bar{L} = 1$).

Ejercicio 4.43. Rotación

Se busca implementar un circuito combinacional con cuatro entradas de datos $D(3)D(2)D(1)D(0)$ y cuatro salidas $O(3)O(2)O(1)O(0)$ y dos entradas de control $C(1)C(0)$, de modo tal que la palabra de salida corresponda a la palabra de entrada rotada en la cantidad de posiciones dada por las entradas de control. Por ejemplo, si la palabra de entrada es $D(3)D(2)D(1)D(0)$ y control es $C(1)C(0) =$ "01", la palabra de salida debe ser $O(3)O(2)O(1)O(0) = D(2)D(1)D(0)D(3)$.

a) Confeccione la tabla de verdad reducida del dispositivo.

b) Diseñe el circuito en base a multiplexores de cuatro canales.

c) Agregue al diseño una entrada de control adicional llamada $R/\bar{L}$, de modo tal que si $R/\bar{L} = 1$ el circuito rota la palabra hacia la derecha, en tanto que si $R/\bar{L} = 0$ el circuito rota la palabra hacia la izquierda.

Ejercicio 4.44. Saturación

a) Implemente un circuito que reciba un número no signado de 16 bits y sature el mismo a 8 bits.

b) Implemente un circuito que reciba un número signado en convenio complemento a dos de 16 bits y sature el mismo a 8 bits.

En ambos casos llame **X** al número de entrada al circuito y **Y** a la salida del mismo.

Ejercicio 4.45. Saturación

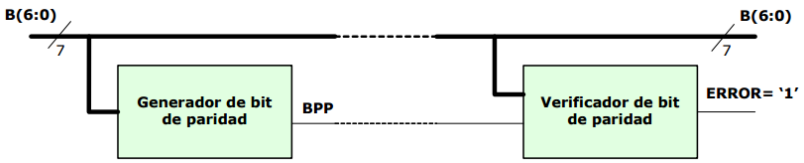
(a) Implemente un circuito que realice la suma de dos números signados en CA2 de N bits con resultado de N bits. Si la operación produce *overflow* sature el resultado. Indique con una salida que el resultado está saturado.

(b) Implemente un circuito que realice la resta de dos números signados en CA2 de N bits con resultado de N bits. Si la operación produce *overflow* sature el resultado. Indique con una salida que el resultado está saturado.

En ambos casos llame **A** y **B** a los operandos de entrada, **R** al resultado obtenido y **SAT** a la señal que indica si el resultado está o no saturado.

Ejercicio 4.46. Paridad

Diseñe los circuitos correspondientes a un generador y a un verificador de bits de paridad para el código ASCII con siete bits B(6:0), que utilicen bit de paridad par (BPP).



5. Diseño de circuitos secuenciales sincrónicos

Ejercicio 5.1. Registro SISO

Construya un registro de desplazamiento de 4 bits SISO en base a flip-flops D disparados por flanco ascendente.

a) Dibuje el circuito de un registro que desplace a la derecha y otro a la izquierda. Escriba las ecuaciones de transición de ambos circuitos.

b) Integre los 2 registros del punto anterior de manera que pueda desplazar a derecha o a izquierda. Escriba la ecuación de transición del dispositivo resultante. ¿Necesita entradas adicionales?

c) Incorpore al registro anterior una entrada de habilitación y otra de reset sincrónico. El reset sincrónico debe tener la mayor prioridad. Escriba las ecuaciones de transición.

d) A partir de un diagrama temporal muestre como puede conocer el estado interno de un registro SISO aplicando pulsos de reloj y leyendo su salida serie sin olvidar de activar su entrada de habilitación.

e) Muestre con el diagrama del punto anterior que la lectura del estado del registro SISO es destructiva, es decir, que luego de la aplicación de los 4 pulsos de reloj el estado interno del registro se destruye. Incorpore una entrada **rd** que al activarse conserve el estado del registro luego de 4 pulsos de reloj.

f) Generalice el circuito anterior para un registro SISO de una longitud de palabra arbitraria W.

Ejercicio 5.2. Registro PIPO

Construya un registro *Parallel Input - Parallel Ouptut* de W bits (longitud arbitraria) de forma tal que cumpla con las siguientes especificaciones:

Interfaz de entrada/salida

Nombre del puerto	Tipo	Ancho	Descripción
rst_n	Entrada	1	Reset asincrónico
clk	Entrada	1	Reloj
srst	Entrada	1	Reset sincrónico
din	Entrada	W	Dato de entrada
load	Entrada	1	Señal de carga
clr	Entrada	1	Señal de limpieza
dout	Salida	W	Dato de salida
new	Salida	1	Señal de dato nuevo

Tabla de comportamiento

srst	load	clr	new	dout*	new*
1	-	-	-	'd0	0
0	1	-	0	din	1
0	1	-	1	dout	1
0	0	1	-	dout	0
0	0	0	-	dout	new

- Las columnas *dout** y *new** denotan el estado futuro de los registros *dout* y *new*.
- La expresión *'d0* indica que el valor del registro debe ser 0 (en numeración decimal).

Ejercicio 5.3. Registro PISO

Construya un registro *Parallel Input - Serial Output*. Para ello siga los siguientes pasos:

a) Determine el sentido de desplazamiento del registro.

b) Encuentre las ecuaciones de transición del registro.

c) Dibuje el circuito resultante.

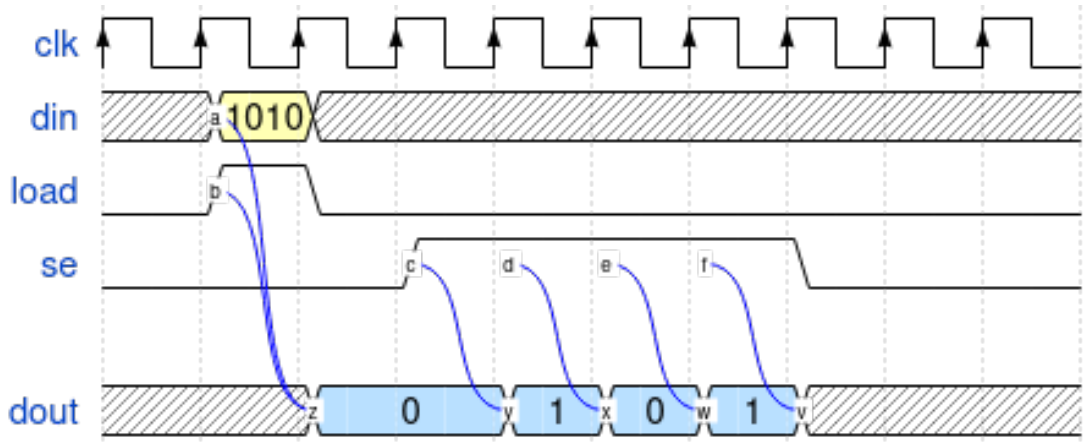
d) Incorpore una señal de reset sincrónico y obtenga las nuevas ecuaciones de transición (o tabla reducida de comportamiento).

e) Generalice el circuito anterior para un registro PISO de una longitud de palabra arbitraria W.

Interfaz de entrada/salida

Nombre del puerto	Tipo	Ancho	Descripción
rst_n	Entrada	1	Reset asincrónico
clk	Entrada	1	Reloj
din	Entrada	4	Dato de entrada
load	Entrada	1	Señal de carga
se	Entrada	1	Shift Enable
dout	Salida	1	Dato de salida serie

Diagrama temporal



Ejercicio 5.4. Registro SIPO

Construya un registro *Serial Input - Parallel Output*. Para ello siga los siguientes pasos:

a) Determine el sentido de desplazamiento del registro.

b) Encuentre las ecuaciones de transición del registro.

c) Determine el valor de la salida dout en los ciclos identificados con '?'.

d) Dibuje el circuito resultante.

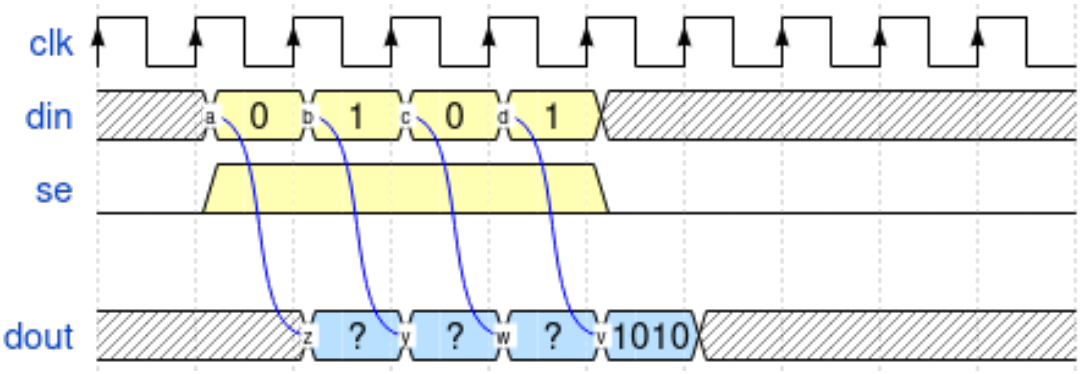
e) Incorpore una señal de reset sincrónico y obtenga las nuevas ecuaciones de transición (o tabla reducida de comportamiento).

f) Generalice el circuito anterior para un registro SIPO de una longitud de palabra arbitraria W.

Interfaz de entrada/salida

Nombre del puerto	Tipo	Ancho	**Descripción**
rst_n	Entrada	1	Reset asincrónico
clk	Entrada	1	Reloj
din	Entrada	1	Dato de entrada serie
se	Entrada	1	Shift Enable
dout	Salida	4	Dato de salida

Diagrama temporal



Ejercicio 5.5. Contador universal de longitud arbitraria W

Diseñe un contador cuya palabra de salida, de longitud arbitraria W, esté codificada en binario natural. Para ello tenga en cuenta las siguientes consideraciones de diseño

a) El contador debe ser capaz de contar en forma ascendente y descendente. Para ello incorpore una señal $U/\overline{D}$ que permita seleccionar la dirección de conteo.

b) El contador debe contar con una entrada de habilitación *en* y una entrada de reset sincrónico $srst$.

c) Además del estado de cuenta, el contador debe tener una señal de salida que indique el final de la cuenta. Diseñe convenientemente esta indicación y llama a la señal *tc* (terminal count).

d) Establezca un esquema de prioridades para las operaciones sincrónicas y explique el por qué de sus elecciones.

Ejercicio 5.6. Contador de módulo arbitrario M

Diseñe un contador de módulo arbitrario y seleccionable M, es decir, el contador debe ser capaz de efectuar la siguiente secuencia de conteo:

$$cuenta = \{0, 1, 2, \ldots, M-2, M-1, 0\}$$

Asuma que la palabra de salida del contador está codificada en binario natural. Para el diseño del mismo tenga en cuenta las siguientes consideraciones de diseño

a) Encuentre una relación entre la longitud de palabra del contador y el módulo de su cuenta.

b) El contador sólo deberá contar en forma ascendente.

c) El contador debe contar con una entrada de habilitación *en* y una entrada de reset sincrónico $srst$.

d) Para cargar el valor arbitrario del módulo, el contador deberá tener una señal de carga paralela *ld* y un vector de entrada *din* cuya longitud deberá determinar.

e) Además del estado de cuenta, el contador deberá tener una señal de salida que indique el final de la cuenta. Diseñe convenientemente esta indicación y llama a la señal *tc* (terminal count).

f) Establezca un esquema de prioridades para las operaciones sincrónicas y explique el por qué de sus elecciones.

Ejercicio 5.7. Contador signado

Diseñe un contador cuya palabra de salida, de longitud arbitraria W, sea un entero codificado en código de complemento a 2. Para ello tenga en cuenta las siguientes consideraciones de diseño

a) El contador debe ser capaz de contar en forma ascendente y descendente. Para ello incorpore una señal $U/\overline{D}$ que permita seleccionar la dirección de conteo.

b) El contador debe contar con una entrada de habilitación *en* y una entrada de reset sincrónico *srst*.

c) Además del estado de cuenta, el contador debe tener una señal de salida que indique el final de la cuenta. Diseñe convenientemente esta indicación y llama a la señal *tc* (terminal count).

d) Establezca un esquema de prioridades para las operaciones sincrónicas y explique el por qué de sus elecciones.

Ejercicio 5.8. Toggle habilitado

Un circuito temporizado por un reloj de 50MHz tiene por finalidad hacer destellar un led (la salida del circuito)

(a) Implemente el circuito usando un contador ascendente de la longitud de palabra que considere necesaria y un comparador de magnitudes.

(b) Implemente el circuito con un contador ascendente y un flip-flop habilitado por la señal de fin de cuenta.

Ejercicio 5.9. Esquemas de expansión

(a) Construya un contador binario ascendente de módulo 1024.

(b) Determine la cantidad de flip-flops necesarios para construir dicho contador y calcule el costo por entrada de la lógica de estado futuro.

(c) Construya ahora un contador binario ascendente de módulo 32.

(d) Evalúe la posibilidad de construir un circuito equivalente al del punto a) pero a partir de dos contadores de módulo 32.

(e) Determine la cantidad de flip-flops necesarios para construir dicho contador y calcule el costo por entrada de la lógica de estado futuro.

(f) Construya ahora un contador binario ascendente de módulo 16 y otro de módulo 64.

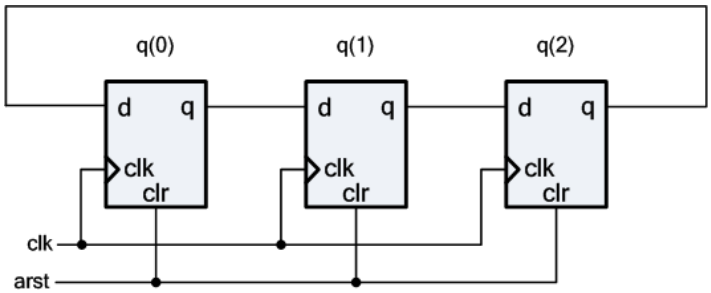
(g) Evalúe la posibilidad de construir un circuito equivalente al del punto a) pero a partir de estos dos nuevos contadores.

(h) Determine la cantidad de flip-flops necesarios para construir dicho contador y calcule el costo por entrada de la lógica de estado futuro.

(i) Compare las tres implementaciones y obtenga conclusiones.

Ejercicio 5.10. Contador en anillo

Dado el siguiente registro de desplazamiento de 3 bits



(a) ¿Qué sucede si los flip-flops son inicializados con la señal de reset asincrónico?

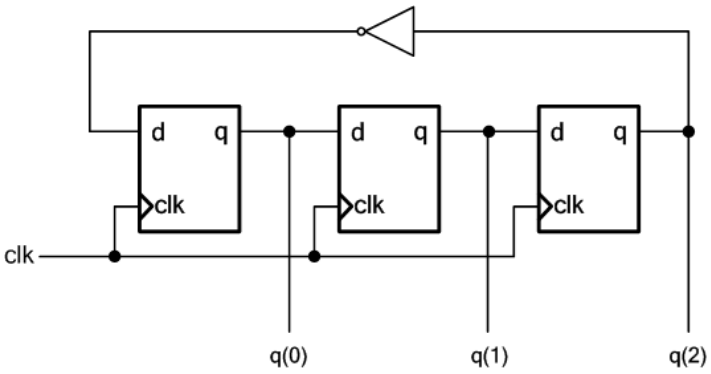
(b) ¿Cómo inicializaría al registro para que no entre en un situación de deadlock? En este dicho caso ¿por cuántos estados transita? Derive una señal de fin de cuenta.

(c) ¿Existen otros estados de inicio que lleven a un deadlock? ¿Existen otros que no? Si es así, ¿por cuantos estados transita el shift register?

(d) Generalice sus conclusiones para un contador en anillo de una longitud de palabra W.

Ejercicio 5.11. Contador Johnson

Una variante del contador en anillo es el contador Johnson. En la figura vemos uno de 3 bits



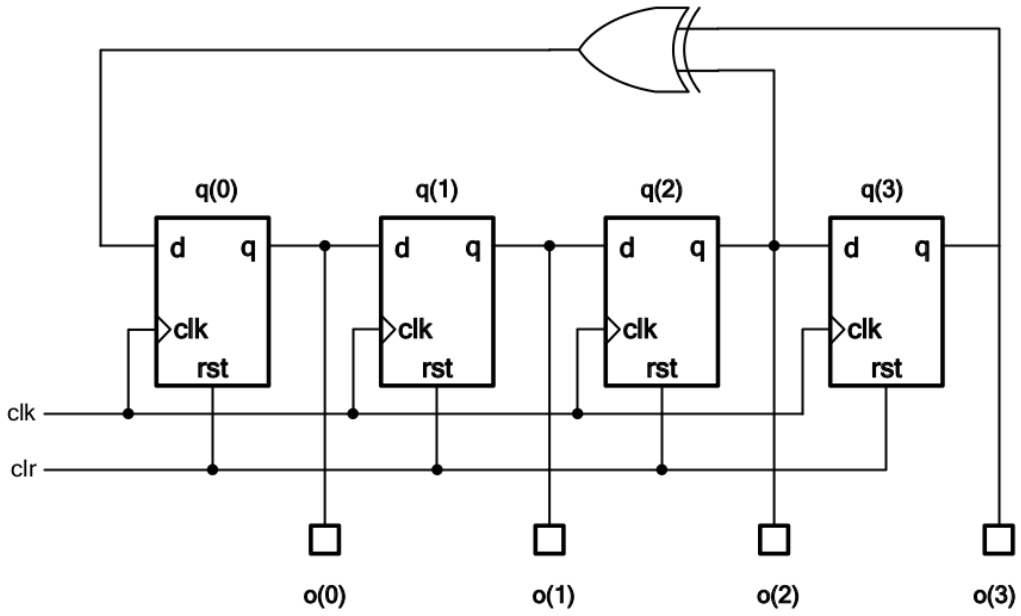
(a) Si las salidas de todos los flip-flops están en '0' ¿por cuantos estados transita el contador?

(b) En el caso que el registro de desplazamiento inicie en un estado diferente al "000"¿cuál es el módulo del contador?

(c) Generalice sus conclusiones para un contador Johnson de una longitud de palabra W.

Ejercicio 5.12. LFSR - Linear Feedback Shift Register

Un sistema de ruleta necesita de un generador de números aleatorios (en realidad pseudo-aleatorios) del 0 al 36. En primer lugar se propuso el circuito de la figura como solución al problema.



a) Determine por simple inspección si este circuito es capaz de resolver la generación de números en el rango solicitado.

b) Obtenga las ecuaciones de transición.

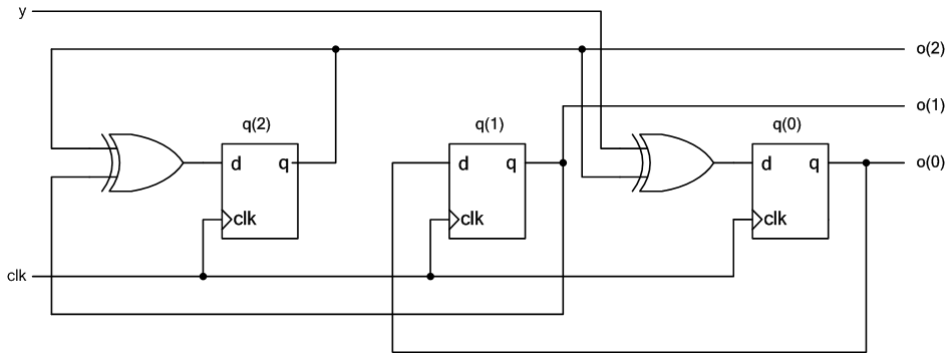
c) Confeccione el diagrama de estados y compruebe que la secuencia que recorre es difícil de predecir.

d) Indique cual es el estado que no forma parte de la secuencia principal. Muestre como modificaría el circuito de forma tal que si se encuentra en este estado fuera de secuencia, reingrese a la secuencia buscada en el próximo ciclo de reloj. Debe realizar un cambio simple.

e) Evalúe como podría rediseñar el circuito para resolver el problema inicial. Para ello se sugiere la siguiente lectura: *Efficient Shift Registers, LFSR Counters, and Long Pseudo-Random Sequence Generators*, que puede encontrar en el siguiente enlace `https://www.xilinx.com/support/documentation/application_notes/xapp052.pdf`

6. Máquinas de estados finitos (FSM – Finite State Machine)

Ejercicio 6.1. Análisis de una máquina de estados



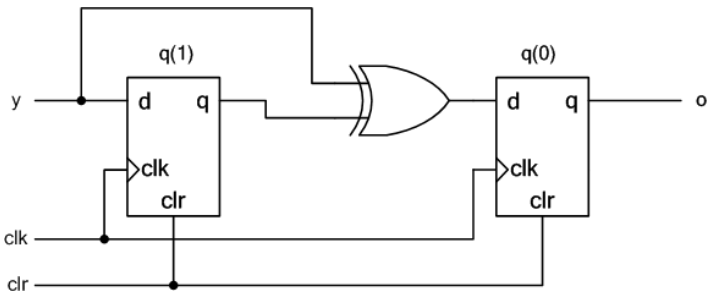
Dado el siguiente circuito secuencial sincrónico, analícelo a fin de obtener

(a) Las expresiones de excitación y salida.

(b) La tabla de transiciones / salidas.

(c) El diagrama de estados debidamente documentado de acuerdo al tipo de máquina.

Ejercicio 6.2. Análisis de una máquina de estados



Dado el siguiente circuito secuencial sincrónico, analícelo a fin de obtener

(a) Las expresiones de excitación y salida.

(b) La tabla de transiciones / salidas.

(c) El diagrama de estados debidamente documentado de acuerdo al tipo de máquina.

(d) Asumiendo que los parámetros temporales de los flip-flops y las compuertas son

$$5ns \leqslant t_{XOR} \leqslant 15ns \quad \wedge \quad t_{SU} = 10ns \quad \wedge \quad t_h = 5ns$$

Determine la máxima frecuencia a la que puede operar correctamente el circuito dando la debida justificación.

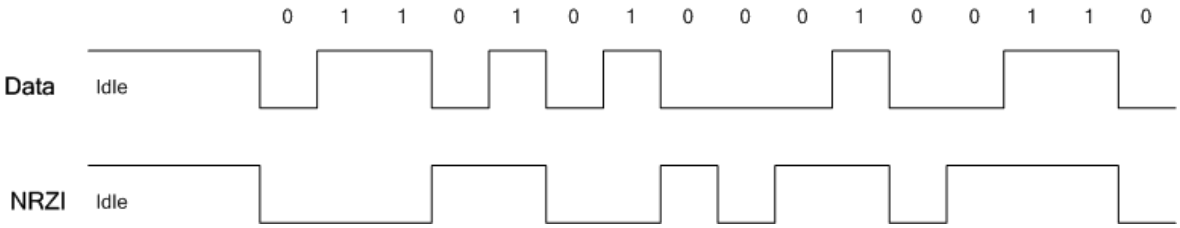
(e) Determine el máximo desplazamiento temporal positivo (clock skew) admisible en la entradas de clk para garantizar la correcta operación.

Ejercicio 6.3. Síntesis de una máquina de estados: Decodificador NRZI

En el protocolo USB la información que se transmite o recibe entre el Host Controller y los Devices está codificada en **NRZI**. Esta codificación consiste en representar a:

- el **cero** con una transición (de **'1' a '0'** ó de **'0' a '1'**)
- el **uno** con una no transición.

Por ejemplo la siguiente secuencia de bits ***"00000001"*** se codifica en **NRZI** como ***"01010100"***, que corresponde al campo Sync (Sync Pattern). A continuación se muestra una trama a modo de ejemplo extraída de la norma USB 2.0



Teniendo en cuenta que los datos se muestrean sincrónicamente en los flancos ascendentes del reloj, diseñe una máquina de estados que decodifique NRZI.

(a) Confeccione el diagrama de estados.

(b) Realice una asignación de código de estados buscando minimizar la lógica necesaria.

(c) Implemente usando flip-flops D y mínima lógica adicional.

Ejercicio 6.4. Síntesis de una máquina de estados:

Se requiere implementar un contador sincrónico decimal con entrada de inicialización asincrónica **reset**, que evolucione acorde a la siguiente secuencia en las salidas de sus flip-flops:

$$q3\ q2\ q1\ q0 \Rightarrow [0000\text{–}0001\text{–}0010\text{–}0011\text{–}0100\text{–}1000\text{–}1001\text{–}1010\text{–}1011\text{–}1100\text{–}0000\text{–}...]$$

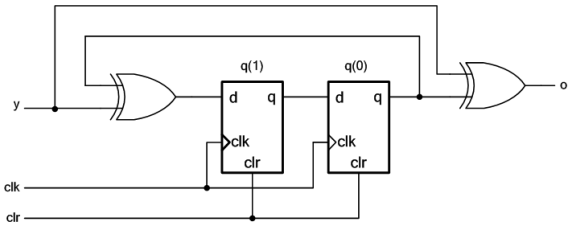
a) Confeccione el diagrama de estados.

b) Realice una asignación de código de estados garantizando que todas las salidas de la máquina de estado resultante sean tipo Moore.

c) Implemente usando flip-flops D y mínima lógica adicional. Obtenga las expresiones suma de productos mínima (SPm) de las entradas preparatorias de cada flip-flop.

Ejercicio 6.5. Análisis de una máquina de estados

Dado el siguiente circuito secuencial sincrónico, analícelo a fin de obtener

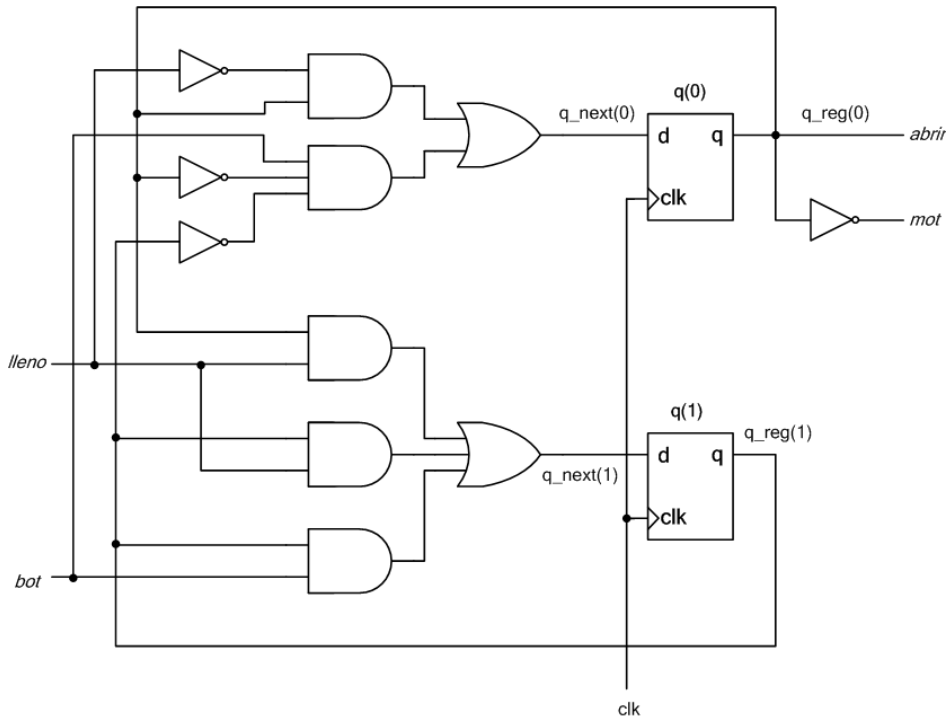


(a) Las expresiones de excitación y salida.

(b) La tabla de transiciones / salidas.

(c) El diagrama de estados debidamente documentado de acuerdo al tipo de máquina.

Ejercicio 6.6. Análisis de una máquina de estados



Dado el siguiente circuito secuencial sincrónico, analícelo a fin de obtener

(a) Las expresiones de excitación y salida.

(b) La tabla de transiciones / salidas.

(c) El diagrama de estados debidamente documentado de acuerdo al tipo de máquina.

(d) Asumiendo que los parámetros temporales de los flip-flops y las compuertas son

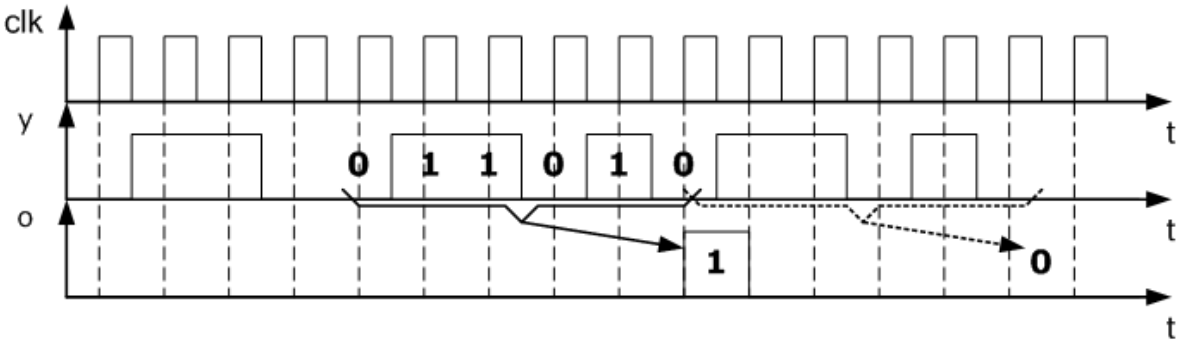
$$5ns \leqslant t_{XOR} \leqslant 15ns \quad \wedge \quad t_{SU} = 10ns \quad \wedge \quad t_h = 5ns$$

Determine la máxima frecuencia a la que puede operar correctamente el circuito dando la debida justificación.

(e) Determine el máximo desplazamiento temporal positivo (clock skew) admisible en la entradas de clk para garantizar la correcta operación.

Ejercicio 6.7. Síntesis de una máquina de estados: Detector de un patrón de bits

Implemente una máquina de estados sincrónica que con una entrada **Y** y una salida **O** permita detectar dentro de un cadena de bits el patrón de 6 bits ***"011010"*** sin solapamiento poniendo su salida **O = '1'** durante un ciclo de clk a partir del instante desde que muestrea el último bit de la secuencia buscada. Tome como ejemplo el siguiente diagrama temporal:

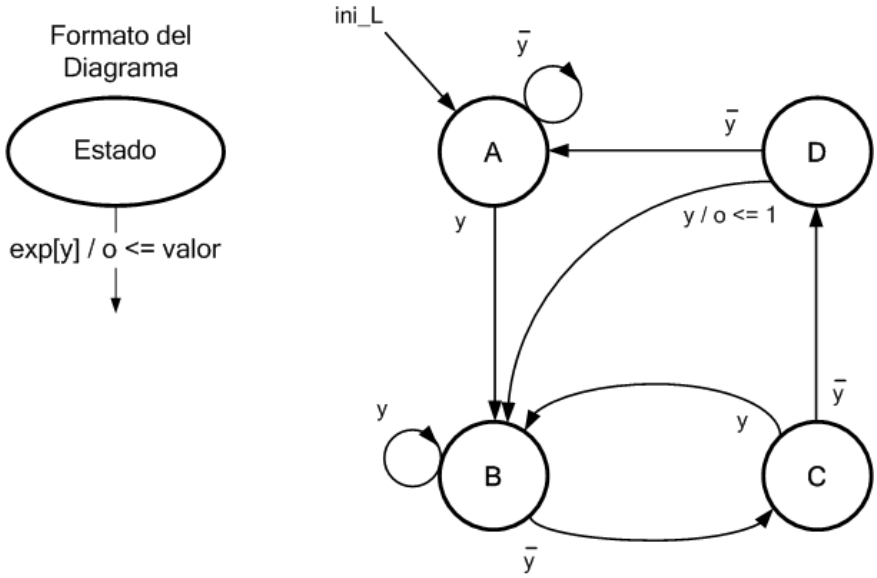


(a) Confeccione el diagrama de estados.

(b) Realice una asignación de códigos de estados usando asignación Casi One-Hot.

(c) Implemente el circuito usando flip-flops D y mínima lógica adicional.

Ejercicio 6.8. Síntesis de una máquina de estados



Dado el diagrama de estados de la figura, sintetice utilizando flip-flops y mínima lógica adicional. Utilice para la síntesis una asignación de códigos de estado casi one-hot. Para ello obtenga

(a) La tabla de Transición / Excitación / Salida.

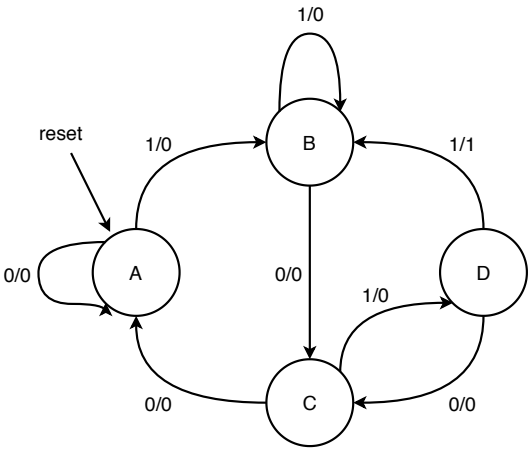
(b) Obtenga las expresiones mínima de la lógica de estados futuros y de la salida o.

(c) Determine la máxima frecuencia de clk a la que puede operar el circuito considerando que

$$t_{CQ} = 24ns \quad \wedge \quad t_{SU} = 19ns \quad \wedge \quad t_h = 3ns$$

$$t_{gate} = 18ns \quad \wedge \quad t_{NOT} = 18ns$$

Ejercicio 6.9. Integrador #1



a) ¿Cuál es el número mínimo de flip-flops necesarios para implementar la máquina de estados de la figura?

b) Confeccione la tabla de transición de estados.

c) Obtenga las expresiones mínimas de las entradas preparatorias para todos los flip-flops D del circuito.

d) Obtenga las expresiones mínimas para todas las salidas del circuito.

Ejercicio 6.10. Integrador #2

Se requiere implementar un sistema sincrónico (**clk**) que lleve la cuenta de los vehículos que ingresan y egresan de una cochera. El lugar tiene la puerta de ingreso separada de la puerta de egreso y en ambas hay sensores para determinar si un vehículo entra (**car_in=1**) o sale (**car_out=1**) – ambas señales están sincronizadas con **clk**. Se admite la presencia simultánea de hasta tres autos dentro del recinto. Además, en un mismo ciclo de **clk** un vehículo puede entrar y otro puede salir del lugar.

a) Confeccione el diagrama de estados del sistema, que genere en la salida **avail** la cantidad de lugares vacantes y una salida **full** (completo) que debe activarse inmediatamente cuando esa condición sea verdadera. Debe proveerse una entrada de **reset asincrónico (arst)** para inicializar el circuito.

b) Realice una conveniente asignación de código de estado y confeccione la tabla de transición/salidas

c) En base a la tabla obtenga las expresiones mínimas de excitación de los FF-D empleados y las expresiones mínimas de las salidas.

Ejercicio 6.11. Integrador #3

En una comunicación digital, se emplea un patrón de sincronización especial (preámbulo) para indicar el comienzo de un paquete de datos. En el estándar IEEE 802.3 el preámbulo está formado por siete bytes, cada uno de los cuales incluye el patrón de bits “10101010”. Se busca diseñar una máquina de estados que genere el patrón “10101010”. El circuito deberá contar con una señal de entrada **start**, de modo que cuando **start=1** durante un ciclo de **clk**, el circuito genere el patrón descripto en la señal de salida **data_out** durante los próximos ocho ciclos de **clk**. Además, el circuito debe contar también con una entrada de reset asincrónica **arst**.

a) Confeccione el diagrama de estados.

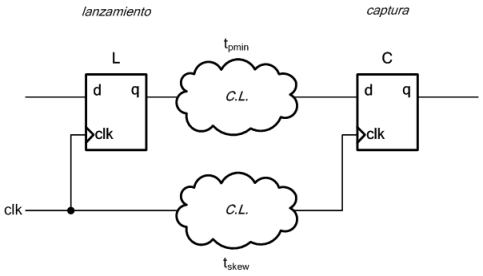
b) Confeccione la tabla de transición de estados.

c) Obtenga las expresiones mínimas de las entradas preparatorias para todos los flip-flops D del circuito.

d) Obtenga las expresiones mínimas para todas las salidas del circuito.

7. Análisis temporal estático

Ejercicio 7.1. Clock skew negativo



El clock skew negativo refiere a que la señal de reloj llega demorada al flip-flop de lanzamiento respecto al de captura. Para esta situación determine a partir de los siguientes parámetros temporales

$$t_{SU} = 200ps \qquad t_h = 100ps \qquad t_{CQ} = 200ps$$

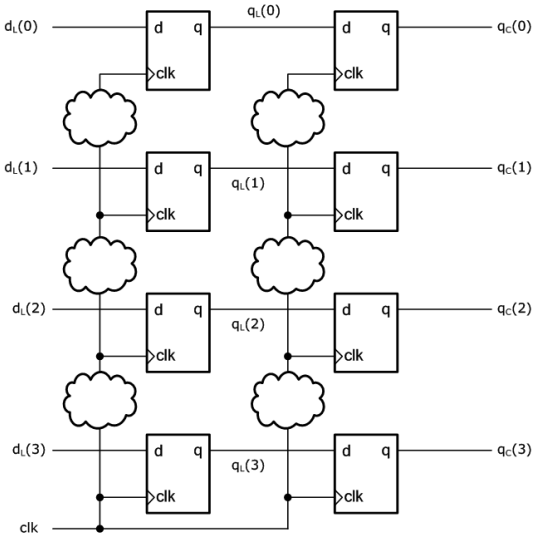
$$t_{skew} = -50ps \qquad T_{CLK} = 1ns$$

(a) Determine si hay alguna violación de timing. Calcule el hold slack.

(b) Repita el procedimiento para un $t_{skew} = -100ps$.

(c) Si no encuentra problemas de timing averigüe para que valor de t_{skew} comienzan los problemas. Plasme en una ecuación las conclusiones.

Ejercicio 7.2. Clock skew encadenado



Dados 2 registros PIPO como los de la figura con

$$t_{SU} = 200ps$$

$$t_h = 100ps$$

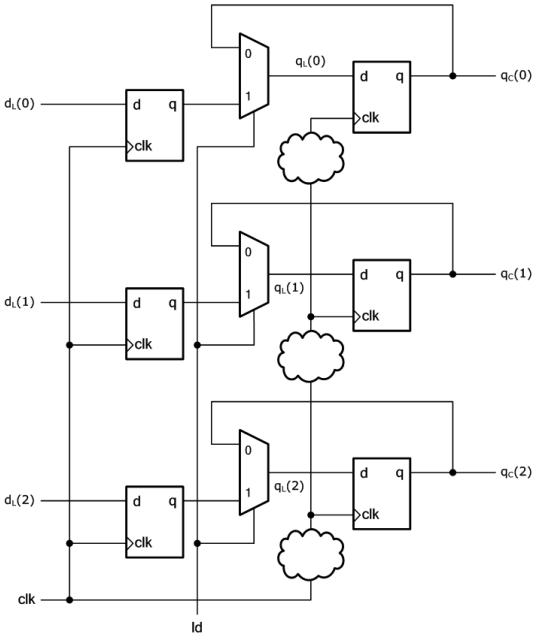
$$t_{CQ} = 200ps$$

$$t_{skew} = +50ps$$

$$T_{CLK} = 1ns$$

- Si todos los t_{skew} son iguales determine si hay alguna posibilidad de violación de timing.
- Si hay un variación en t_{skew} de $\pm 10\,\%$ entre ellos evalúe la posibilidad de una violación de timing.

Ejercicio 7.3. Carreras entre datos y reloj

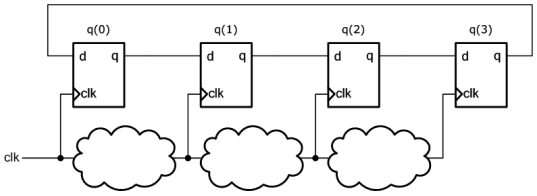


Analice el siguiente registro PIPO con los siguientes parámetros temporales

$$t_{SU} = 200ps$$
$$t_h = 100ps$$
$$t_{CQ} = 200ps$$
$$t_{skew} = +50ps$$
$$T_{CLK} = 1ns$$

y el t_{skew} entre flip-flops es de +50ps. Los mux tienen una demora de propagación tmux de 50ps. Determine si se producen violaciones temporales o carreras entre data y clock que den lugar a capturas fuera de tiempo.

Ejercicio 7.4. Caminos R2R interdependientes



Para el siguiente circuito con clock skew positivo, determine hasta que valor puede crecer el clock skew sin afectar el correcto funcionamiento del circuito. Los parámetros temporales de los flip-flops son:

$$t_{SU} = 150ps$$
$$t_h = 75ps$$
$$t_{CQ} = 180ps$$
$$T_{CLK} = 500ps$$

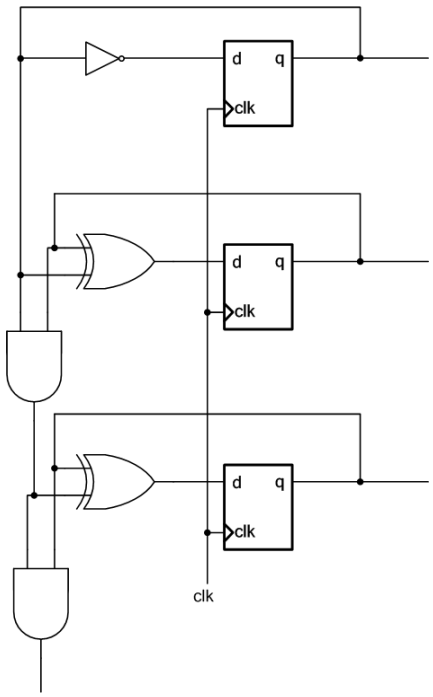
Ejercicio 7.5. Máxima frecuencia de operación de un contador en anillo

Dado un contado en anillo de 5 bits en el cual sus flip-flops tienen los siguientes parámetros temporales

$$t_{CQ} = 400ps$$
$$t_{SU} = 300ps$$
$$T_h = 200ps$$

Determine la máxima frecuencia de operación del contador.

Ejercicio 7.6. Máxima frecuencia de operación de un contador binario



Determine para el circuito de la figura

(a) Si puede funcionar a una frecuencia de reloj de 500 MHz.

(b) Si no es posible, utilice clock skew para lograrlo.

Considere los siguientes parámetros temporales:

$$t_{CQ} = 275ps$$

$$t_{SU} = 250ps$$

$$t_h = 100ps$$

$$t_{AND} = 600ps$$

$$t_{NOT} = 100ps$$

$$t_{XOR} = 500ps$$

Ejercicio 7.7. Integrador

Bajo la suposición de que todas las compuertas del mismo tipo que forman un sumador iterativo tienen el mismo tiempo de propagación (todas las compuertas and tienen el mismo t AND, todas compuertas xor tienen el mismo t XOR, etc.), determine

(a) El camino crítico del sumador.

(b) Como aumenta el tiempo de propagación del camino crítico en la medida que aumenta la cantidad de bits del sumador.

(c) Es sabido que el tiempo de propagación de una compuerta depende en buena medida de la carga capacitiva de salida que debe manejar. No obstante la asunción de que el tiempo de propagación del mismo tipo de compuertas es idéntico en un sumador iterativo, es aceptable ¿Por qué?

Ejercicio 7.8. Integrador

Dada una suma multioperando donde la cantidad de operandos es una potencia de 2. Compare 2 alternativas de implementación, una basada en una estructura en cascada y otra en árbol. Ejemplifique con 8 operandos y suponga que el tiempo de propagación de todas las compuertas del circuito es el mismo.

(a) Determine el tiempo de propagación de ambas soluciones.

(b) Determine la cantidad de elementos (área en el chip) que requieren ambas alternativas.

Ejercicio 7.9. Integrador

El circuito de la figura corresponde a un contador de módulo programable de 16 bits, cuya señal **terminal count** es registrada mediante un flip-flop y empleado por el resto del sistema. En función de los siguientes tiempos de propagación:

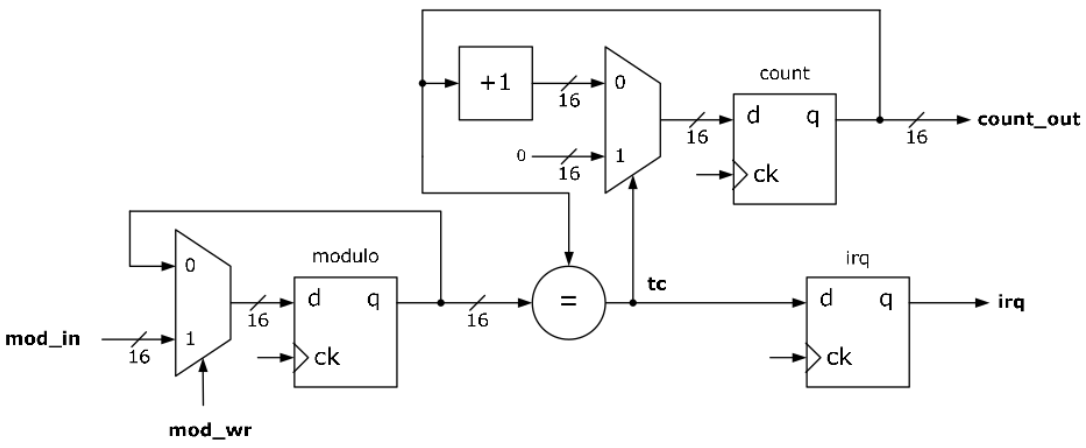
- $t_{MUX} = 500ps$
- $t_{comp} = 2ns$
- $t_{inc} = 16ns$
- $t_{SU} = 100ps$
- $t_{CQ} = 250ps$
- $t_{hold} = 80ps$

Determine

a) Indique cual de los caminos combinacionales entre dos registros tiene el mayor tiempo de propagación. ¿Cuál es el motivo por el qué es mayor al resto?

b) Calcule la máxima frecuencia a la que puede operar el circuito en forma confiable.

c) Si la frecuencia de operación fuera 50 MHz, recalcule el timing del punto a) y determine el margen de tiempo respecto al a) (time slack).



8. HDLs

8.1. Parte 1: Operaciones lógicas y circuitos de ruteo

Ejercicio 8.1. Función XOR

- Diseñe un módulo digital cuya salida sea igual a la suma exclusiva de sus entradas.
- La implementación debe estar basada en suma de productos.
- Respete los nombres de los puertos de entrada y salida propuestos para la interfaz del módulo.
- Obtenga su diagrama RTL.
- Simule el diseño y verifique su correcto funcionamiento.

```
entity xor_gate is
  port (
      x1  : in    std_logic;
      x2  : in    std_logic;
      y   : out   std_logic
  );
end xor_gate;
```

Ejercicio 8.2. Función XOR

- Diseñe un módulo digital cuya salida sea igual a la suma exclusiva de sus entradas.
- La implementación debe estar basada en el uso del operador **xor**.
- Respete los nombres de los puertos de entrada y salida propuestos para la interfaz del módulo.
- Obtenga su diagrama RTL.
- Simule el diseño y verifique su correcto funcionamiento.

```
entity xor_gate is
  port (
      x1  : in    std_logic;
      x2  : in    std_logic;
      y   : out   std_logic
  );
end xor_gate;
```

Ejercicio 8.3. Multiplexor 2 a 1

- Diseñe un multiplexor 2 a 1, dos entradas, una palabra de selección y una salida.
- La implementación debe estar basada en suma de productos.
- Respete los nombres de los puertos de entrada y salida propuestos para la interfaz del módulo.
- Obtenga su diagrama RTL.
- Simule el diseño y verifique su correcto funcionamiento.

```
entity mux2to1 is
  port (
      x1  : in    std_logic;
      x2  : in    std_logic;
      s   : in    std_logic;
      y   : out   std_logic
  );
end mux2to1;
```

Ejercicio 8.4. Multiplexor 2 a 1

- Diseñe un multiplexor 2 a 1, dos entradas, una palabra de selección y una salida.
- La implementación debe estar basada en el uso de la sentencia **when ... else**.
- Respete los nombres de los puertos de entrada y salida propuestos para la interfaz del módulo.
- Obtenga su diagrama RTL.
- Simule el diseño y verifique su correcto funcionamiento.

```
entity mux2to1 is
  port (
       x1   : in     std_logic;
       x2   : in     std_logic;
       s    : in     std_logic;
       y    : out    std_logic
  );
end mux2to1;
```

Ejercicio 8.5. Multiplexor 2 a 1

- Diseñe un multiplexor 2 a 1, dos entradas, una palabra de selección y una salida.
- La implementación debe estar basada en el uso de la sentencia **with ... select**.
- Respete los nombres de los puertos de entrada y salida propuestos para la interfaz del módulo.
- Obtenga su diagrama RTL.
- Simule el diseño y verifique su correcto funcionamiento.

```
entity mux2to1 is
  port (
       x1   : in     std_logic;
       x2   : in     std_logic;
       s    : in     std_logic;
       y    : out    std_logic
  );
end mux2to1;
```

Ejercicio 8.6. Multiplexor 2 a 1

- Diseñe un multiplexor 2 a 1, capaz de multiplexar palabras de longitud arbitraria W.
- Utilice el criterio que considere más conveniente para la implementación del mismo.
- Respete los nombres de los puertos de entrada y salida propuestos para la interfaz del módulo.
- Obtenga su diagrama RTL.
- Simule el diseño y verifique su correcto funcionamiento.

```
entity mux2to1 is
  generic (
      W: natural := 8
  );
  port (
       x1   : in     std_logic_vector(W-1 downto 0);
       x2   : in     std_logic_vector(W-1 downto 0);
       s    : in     std_logic;
       y    : out    std_logic_vector(W-1 downto 0)
  );
end mux2to1;
```

Ejercicio 8.7. Circuito de ruteo con prioridad

- Diseñe un circuito de ruteo con prioridad.
- El circuito debe funcionar de la siguiente manera
  - Si c1 está activa, la salida debe ser igual a x1.
  - Si c2 está activa pero c1 no, la salida debe ser igual a x2.
  - Si c3 está activa pero c1 no y c2 tampoco, la salida debe ser igual a x3.
  - En cualquier otro caso, la salida debe ser igual a x4.
- Respete los nombres de los puertos de entrada y salida propuestos para la interfaz del módulo.
- Obtenga su diagrama RTL.
- Simule el diseño y verifique su correcto funcionamiento.

```
entity prior_route is
    generic (
        W: natural := 8
    );
    port (
        x1   : in     std_logic_vector(W-1 downto 0);
        x2   : in     std_logic_vector(W-1 downto 0);
        x3   : in     std_logic_vector(W-1 downto 0);
        x4   : in     std_logic_vector(W-1 downto 0);
        c1   : in     std_logic;
        c2   : in     std_logic;
        c3   : in     std_logic;
        y    : out    std_logic_vector(W-1 downto 0)
    ) ;
end prior_route;
```

Ejercicio 8.8. Circuito de ruteo sin prioridad

- Diseñe un circuito de ruteo sin prioridad.
- El circuito debe funcionar de la siguiente manera
  - La salida debe ser igual a x1 cuando la palabra de selección sea igual a 0.
  - La salida debe ser igual a x2 cuando la palabra de selección sea igual a 1.
  - La salida debe ser igual a x3 cuando la palabra de selección sea igual a 2.
  - La salida debe ser igual a x4 cuando la palabra de selección sea igual a 3.
- Respete los nombres de los puertos de entrada y salida propuestos para la interfaz del módulo.
- Obtenga su diagrama RTL.
- Simule el diseño y verifique su correcto funcionamiento.

```
entity non_prior_route is
    generic (
        W: natural := 8
    );
    port (
        x1   : in     std_logic_vector(W-1 downto 0);
        x2   : in     std_logic_vector(W-1 downto 0);
        x3   : in     std_logic_vector(W-1 downto 0);
        x4   : in     std_logic_vector(W-1 downto 0);
        s    : in     std_logic_vector(  1 downto 0);
        y    : out    std_logic_vector(W-1 downto 0)
    ) ;
end non_prior_route;
```

Ejercicio 8.9. Integrador: Unidad de operaciones lógicas

- Diseñe una unidad de operaciones lógicas.
- El circuito debe funcionar de la siguiente manera
  - La palabra de selección **sel**, determina cuales son los operandos, de forma tal que si **sel** es igual a 0, las entradas pares (x2 y x4) son sometidas a la operación lógica, en tanto que si **sel** es igual a 1, las entradas impares (x1 y x3) son sometidas a la operación lógica.
  - Cuando la palabra de operación sea igual a 0 (decimal), la salida será igual al operando cuyo índice resulte el menor de entre los dos operandos seleccionados.
  - Cuando la palabra de operación sea igual a 1 (decimal), la salida será igual al producto lógico (**and**) de los operandos seleccionados.
  - Cuando la palabra de operación sea igual a 2 (decimal), la salida será igual a la suma lógica (**or**) de los operandos seleccionados.
  - Cuando la palabra de operación sea igual a 3 (decimal), la salida será igual a la suma exclusiva (**xor**) de los operandos seleccionados.
  - Cuando la palabra de operación sea igual a 4 (decimal), la salida será igual al complemento del operando cuyo índice resulte el menor de entre los dos operandos seleccionados.
  - Cuando la palabra de operación sea igual a 5 (decimal), la salida será igual al complemento del producto lógico (**nand**) de los operandos seleccionados
  - Cuando la palabra de operación sea igual a 6 (decimal), la salida será igual al complemento de la suma lógica (**nor**) de los operandos seleccionados
  - Cuando la palabra de operación sea igual a 7 (decimal), la salida será igual al complemento de la suma exclusiva (**xnor**) de los operandos seleccionados.
- Respete los nombres de los puertos de entrada y salida propuestos para la interfaz del módulo.
- Obtenga su diagrama RTL.
- Simule el diseño y verifique su correcto funcionamiento.

```
1  entity logic_unit is
2    generic (
3        W: natural := 8
4    );
5    port (
6        x1      : in     std_logic_vector(W-1 downto 0);
7        x2      : in     std_logic_vector(W-1 downto 0);
8        x3      : in     std_logic_vector(W-1 downto 0);
9        x4      : in     std_logic_vector(W-1 downto 0);
10       sel     : in     std_logic;
11       opcode  : in     std_logic_vector(  2 downto 0);
12       y       : out    std_logic_vector(W-1 downto 0)
13   ) ;
14 end logic_unit;
```

8.2. Parte 2: Circuitos aritméticos

Ejercicio 8.10. Incrementador/Decrementador genérico

- Diseñe un circuito incrementador/decrementador.
- El circuito debe funcionar de la siguiente manera
  - La señal de control **m** determina la operación a realizar, 0 (cero) para incrementos y 1 (uno) para decrementos.
  - La palabra de salida **y** es el incremento/decremento en 1 (uno), de la palabra de entrada **x**.
  - La señal de salida **c** representa el carry/borrow de la operación. En ambas operaciones esta señal debe operar con lógica positiva.
- Respete los nombres de los puertos de entrada y salida propuestos para la interfaz del módulo.
- Obtenga su diagrama RTL.
- Simule el diseño y verifique su correcto funcionamiento.

```
entity inc_dec_generic is
    generic (
        W: natural := 8;
        K: integer := 1
    );
    port (
        x : in  std_logic_vector(W-1 downto 0);
        m : in  std_logic;
        y : out std_logic_vector(W-1 downto 0);
        c : out std_logic
    ) ;
end inc_dec_generic;
```

Ejercicio 8.11. Complementos

- Diseñe un circuito capaz de calcular el complemento a 1 (uno) o complemento a 2 (dos) de una palabra de entrada.
- El circuito debe funcionar de la siguiente manera: cuando la señal de control **ctrl** es igual a **0**, la salida es el complemento a 1 (uno) de la señal de entrada, caso contrario es igual al complemento a 2 (dos).
- Respete los nombres de los puertos de entrada y salida propuestos para la interfaz del módulo.
- Obtenga su diagrama RTL.
- Simule el diseño y verifique su correcto funcionamiento.

```
entity complement is
    generic (
        W    : natural := 8
    );
    port (
        x    : in    std_logic_vector(W-1 downto 0);
        ctrl : in    std_logic;
        y    : out   std_logic_vector(W-1 downto 0)
    ) ;
end complement;
```

Ejercicio 8.12. Valor absoluto

- Diseñe un circuito capaz de calcular el valor absoluto de la señal de entrada.
- El circuito debe funcionar de la siguiente manera: cuando la señal de control **ctrl** es igual a **0**, la señal de entrada debe ser interpretada como un número codificado en **signo y magnitud**, caso contrario debe considerarse como un número en **complemento a 2**.
- Respete los nombres de los puertos de entrada y salida propuestos para la interfaz del módulo.
- Obtenga su diagrama RTL.
- Simule el diseño y verifique su correcto funcionamiento.

```
entity abs_value is
    port(
        x    : in  std_logic_vector(7 downto 0);
        ctrl : in  std_logic;
        y    : out std_logic_vector(7 downto 0)
    );
end abs_value;
```

Ejercicio 8.13. Desplazamientos bidireccionales

- Diseñe un circuito capaz de desplazar bidireccionalmente una palabra de entrada.
- El circuito debe funcionar de la siguiente manera:
  - Cuando la señal de dirección **dir** es igual a 0 (cero), el circuito desplazará a izquierda, caso contrario, el circuito desplazará a derecha.
  - La palabra **amt** (*del inglés amount: cantidad*) determina la cantidad de desplazamientos a realizar, de forma tal que si **amt** es igual a 3 (tres), la palabra de entrada se desplaza 3 (tres) posiciones.
- Respete los nombres de los puertos de entrada y salida propuestos para la interfaz del módulo.
- Obtenga su diagrama RTL.
- Simule el diseño y verifique su correcto funcionamiento.

```
-------------------------------------------
-- si dir = '0' => desplaza a la izquierda
-- si dir = '1' => desplaza a la derecha
-- amt determina el # de desplazamientos
-------------------------------------------
entity barrel_shifter is
    port(
        x   : in  std_logic_vector(7 downto 0);
        dir : in  std_logic;
        amt : in  std_logic_vector(2 downto 0);
        y   : out std_logic_vector(7 downto 0)
    );
end barrel_shifter;
```

Ejercicio 8.14. Promedio

- Diseñe un circuito combinacional capaz de calcular el promedio de cuatro palabras signadas en CA2.
- Respete los nombres de los puertos de entrada y salida propuestos para la interfaz del módulo.
- Obtenga su diagrama RTL.
- Simule el diseño y verifique su correcto funcionamiento.

```
entity ma4 is
    generic (
        W  : natural := 8
    );
    port(
        x1 : in  std_logic_vector(7 downto 0);
        x2 : in  std_logic_vector(7 downto 0);
        x3 : in  std_logic_vector(7 downto 0);
        x4 : in  std_logic_vector(7 downto 0);
        y  : out std_logic_vector(7 downto 0)
    );
end ma4;
```

Ejercicio 8.15. Sumador con saturación

- Diseñe un circuito sumador de palabras signadas con capacidad de saturación.
- El circuito debe funcionar de la siguiente manera:
  - Si la señal **mode** es igual a 0 (cero) la salida **acc_next** debe ser igual a **acc_curr + bin**, en tanto que si mode es igual a 1 la salida será **acc_curr - bin**.
  - Tenga en cuenta que las operaciones aritméticas se realizan sobre operandos no signados, y que además el circuito debe ser capaz de saturar el resultado siempre que sea pertinente, en cuyo caso la señal **sat** deberá indicar, en lógica positiva, dicha condición.
- Respete los nombres de los puertos de entrada y salida propuestos para la interfaz del módulo.
- Obtenga su diagrama RTL.
- Simule el diseño y verifique su correcto funcionamiento.

```
entity acc is
    port(
        bin      : in  std_logic_vector( 7 downto 0);
        acc_curr : in  std_logic_vector(15 downto 0);
        mode     : in  std_logic;
        acc_next : out std_logic_vector(15 downto 0);
        sat      : out std_logic
    );
end acc;
```

Ejercicio 8.16. Multiplicador/Divisor combinacional

- Diseñe un circuito combinacional capaz de realizar multiplicaciones y divisiones con operandos constantes.
- El circuito debe funcionar de la siguiente manera:
  - X representa el operando de entrada e Y representa la palabra de salida.
  - Si op es igual a 0 (cero) el circuito hará operaciones de multiplicación, en cambio si vale 1 (uno) hará operaciones de división.
  - Factor determina el segundo operando de la operación a realizar, el cual es igual a 2^{factor}, es decir, en el caso de la multiplicación la operación será igual a $y = x \cdot 2^{factor}$, y en el caso de la división la operación será igual a $y = x/2^{factor}$.
- Respete los nombres de los puertos de entrada y salida propuestos para la interfaz del módulo.
- Obtenga su diagrama RTL.
- Simule el diseño y verifique su correcto funcionamiento.

```
entity mult_div is
    port(
        x      : in  std_logic_vector( 7 downto 0);
        factor : in  std_logic_vector( 1 downto 0);
        op     : in  std_logic;
        y      : out std_logic_vector(10 downto 0)
    );
end mult_div;
```

Ejercicio 8.17. Raíz cuadrada combinacional

- Diseñe un circuito combincacional que calcule raices cuadradas.
- El circuito debe funcionar de la siguiente manera:
  - Llame X a la palabra de entrada.
  - SQRT es la raíz cuadrada de X, sin tener en cuenta la parte fraccionaria.
  - REM es el resto de la operación, o bien, la parte fraccionaria.
- Respete los nombres de los puertos de entrada y salida propuestos para la interfaz del módulo.
- Obtenga su diagrama RTL.
- Simule el diseño y verifique su correcto funcionamiento.

```
entity sqrt_comb is
    port(
        x      : in  std_logic_vector(3 downto 0);
        o_sqrt : out std_logic_vector(3 downto 0);
        o_rem  : out std_logic_vector(3 downto 0)
    );
end sqrt_comb;
```

8.3. Parte 3: Flips-Flops y Registros Sincrónicos

Ejercicio 8.18. Registro de estado

- Diseñe un registro de estado. El nombre de la entidad deberá ser **reg_status**.
- Su funcionamiento debe respetar los siguientes lineamientos
  - **i_clk** es la entrada de reloj.
  - **i_rst** es la entrada de reset asincrónico.
  - **i_en** es la entrada de habilitación del circuito.
  - Siempre que el circuito esté habilitado, si **i_srst** también lo está, todos los registros deberán reiniciarse sincrónicamente, siendo 0 su valor de reset.
  - Siempre que el circuito esté habilitado, si la entrada **i_ld_en** se encuentra activa, la salida **o_flags** tomará el valor de la entrada **i_flags**.
- Respete los nombres de los puertos de entrada y salida propuestos para la interfaz del módulo, a saber

nombres	Tipo	Tamaño
i_clk	in	1
i_rst	in	1
i_en	in	1
i_srst	in	1
i_flags	in	4
i_ld_en	in	1
o_flags	out	4

- Obtenga su diagrama RTL.
- Simule el diseño y verifique su correcto funcionamiento.

Ejercicio 8.19. Sincronizador

- Escriba y sintetice un código HDL para la entidad **synchronizer**.
- ¿Por qué es importante el uso de este tipo de circuitos en los circuitos secuenciales sincrónicos?
- ¿Qué relevancia tiene la cantidad de flip-flops que conforman el circuito?
- Respete los nombres de los puertos de entrada y salida propuestos para la interfaz del módulo.
- Obtenga su diagrama RTL.
- Simule el diseño y verifique su correcto funcionamiento.

```
1  entity synchronizer is
2      generic(
3          W: natural:= 3
4      ); -- W es el # de FFs del sincronizador
5      port(
6          clk : in  std_logic;
7          rst : in  std_logic; -- async reset
8          da  : in  std_logic; -- async input (or data)
9          ds  : out std_logic  -- sync output (or data)
10     );
11 end synchronizer;
```

Ejercicio 8.20. Mapa de registros

- Diseñe un mapa de registros. El nombre de la entidad deberá ser **reg_map**.
- Su funcionamiento debe respetar los siguientes lineamientos
  - **i_clk** es la entrada de reloj.
  - **i_rst** es la entrada de reset asincrónico.
  - **i_en** es la entrada de habilitación del circuito.
  - Siempre que el circuito esté habilitado, si **i_srst** también lo está, todos los registros deberán reiniciarse sincrónicamente, siendo 0 su valor de reset.
  - La palabra **i_address** indica la dirección a la cual se desea escribir. Siempre que el circuito esté habilitado, la salida **o_ri** tomará el valor de la entrada **i_data**, donde **i** representa la dirección que se ingresa a través de **i_address**.
- Respete los nombres de los puertos de entrada y salida propuestos para la interfaz del módulo, a saber

Nombres	Tipo	Tamaño
i_clk	in	1
i_rst	in	1
i_en	in	1
i_srst	in	1
i_data	in	DATA_WIDTH
i_address	in	3
o_r0	out	DATA_WIDTH
o_r1	out	DATA_WIDTH
o_r2	out	DATA_WIDTH
o_r3	out	DATA_WIDTH
o_r4	out	DATA_WIDTH
o_r5	out	DATA_WIDTH
o_r6	out	DATA_WIDTH
o_r7	out	DATA_WIDTH

- Obtenga su diagrama RTL.
- Simule el diseño y verifique su correcto funcionamiento.

Ejercicio 8.21. FIFO

- Escriba y sintetice un código HDL para la entidad **fifo**.
- El parámetro B determina el ancho de las palabras que **fifo** almacena.
- El parámetro W determina la cantidad de palabras que **fifo** almacena. Por ejemplo, si W=4, entonces **fifo** puede almacenar 4 palabras.
- **fifo** cuenta con dos operaciones, lectura y escritura, siendo la lectura la operación más prioritaria.
- Para leer un dato de **fifo**, se debe activar la entrada **rd**. Siempre que **fifo** no esté vacía (**empty = 1**), el dato podrá leerse desde la salida **r_data**. En caso que **fifo** esté vacía **r_data** deberá ser igual a 0.
- Respete los nombres de los puertos de entrada y salida propuestos para la interfaz del módulo.
- Obtenga su diagrama RTL.
- Simule el diseño y verifique su correcto funcionamiento.

```
entity fifo is
    generic(
        B       : natural := 8;                         -- number of bits
        W       : natural := 2                          -- number of address bits
    );
    port(
        clk     : in  std_logic;
        rst     : in  std_logic;                        -- async reset
        rd      : in  std_logic;                        -- read
        wr      : in  std_logic;                        -- write
        w_data  : in  std_logic_vector(B-1 downto 0); -- data to write
        empty   : out std_logic;                        -- empty fifo flag
        full    : out std_logic;                        -- full fifo flag
        r_data  : out std_logic_vector(B-1 downto 0)  -- read data
    );
end fifo;
```

8.4. Parte 4: Contadores sincrónicos

Ejercicio 8.22. Contador binario de longitud de palabra arbitraria

- Escriba y sintetice un código HDL para la entidad **bin_counter**.
- Considere a **d** como la entrada paralela de datos.
- Considere a **q** como la salida del contador.
- La señal de entrada **ld** carga el valor **d** en el contador si y sólo si éste se encuentra habilitado (**en = 1**). Esto permite cargar un valor inicial en el contador.
- La cuenta avanzará solamente si el contador se encuentra habilitado.
- La salida **tc** es el indicador de final de cuenta.
- Respete los nombres de los puertos de entrada y salida propuestos para la interfaz del módulo.
- Obtenga su diagrama RTL.
- Simule el diseño y verifique su correcto funcionamiento.

```
entity bin_counter is
    generic(
        W: natural:= 8
    ); -- W es el ancho del registro (o de la palabra)
    port(
        clk : in  std_logic;
        rst : in  std_logic; -- async reset
        en  : in  std_logic; -- enable
        ld  : in  std_logic; -- load
        d   : in  std_logic_vector(W-1 downto 0); -- dato para cargar
        tc  : out std_logic; -- terminal count
        q   : out std_logic_vector(W-1 downto 0)
    );
end bin_counter;
```

Ejercicio 8.23. Contador binario de módulo arbitrario

- Escriba y sintetice un código HDL para la entidad **bin_mod_counter**.
- Considere a **d** como la entrada paralela de datos.
- Considere a **q** como la salida del contador.
- La cuenta avanzará solamente si el contador se encuentra habilitado.
- La entrada **max** indica el módulo del contador.
- La salida **tc** es el indicador de final de cuenta.
- Respete los nombres de los puertos de entrada y salida propuestos para la interfaz del módulo.
- Obtenga su diagrama RTL.
- Simule el diseño y verifique su correcto funcionamiento.

```
entity bin_mod_counter is
    generic(
        W: natural:= 8
    ); -- W es el ancho del registro (o de la palabra)
    port(
        clk : in  std_logic;
        rst : in  std_logic; -- async reset
        en  : in  std_logic; -- enable
        max : in  std_logic_vector(W-1 downto 0); -- modulo del contador
        tc  : out std_logic; -- terminal count
        q   : out std_logic_vector(W-1 downto 0)
    );
end bin_mod_counter;
```

Ejercicio 8.24. Contador LFSR (linear feedback shift register)

- Escriba y sintetice un código HDL para la entidad **lfsr_counter**.
- Considere a **q** como la salida del contador.
- La cuenta avanzará solamente si el contador se encuentra habilitado.
- Determine cual sería el valor de inicio apropiado para que el contador genere la secuencia deseada.
- Investigue cual sería la función de realimentación necesaria para maximizar la secuencia pseudoaleatoria que genera este tipo de contador.
- La salida **tc** es el indicador de final de cuenta.
- Respete los nombres de los puertos de entrada y salida propuestos para la interfaz del módulo.
- Obtenga su diagrama RTL.
- Simule el diseño y verifique su correcto funcionamiento.

```
entity lfsr_counter is
    generic(
        W   : natural:= 8
    ); -- W es el ancho del registro (o de la palabra)
    port(
        clk : in  std_logic;
        rst : in  std_logic; -- async reset
        en  : in  std_logic; -- enable
        q   : out std_logic_vector(W-1 downto 0);
        tc  : out std_logic
    );
end lfsr_counter;
```

Ejercicio 8.25. Inversor trifásico

- Escriba y sintetice un código HDL para la entidad **inverter**.
- Considere a **q** como la salida del inversor.
- La sistema evolucionará solamente si se encuentra habilitado.
- Determine cual sería el valor de inicio apropiado para que el sistema genere la secuencia deseada.
- Respete los nombres de los puertos de entrada y salida propuestos para la interfaz del módulo.
- Obtenga su diagrama RTL.
- Simule el diseño y verifique su correcto funcionamiento.

```
entity inverter is
    port(
        clk : in  std_logic;                      -- reloj
        rst : in  std_logic;                      -- reset asincronico
        en  : in  std_logic;                      -- habilitacion
        q   : out std_logic_vector(5 downto 0); -- salida de control
    );
end inverter;
```

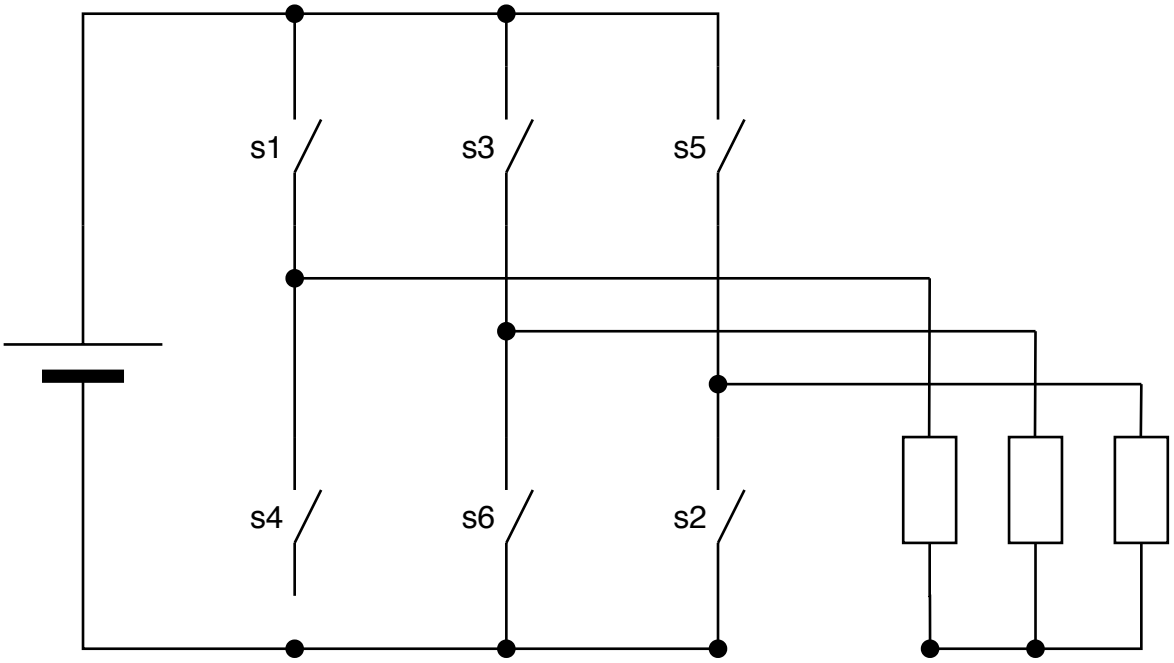


Figura 1: Circuito esquemático de un invsersor trifásico.

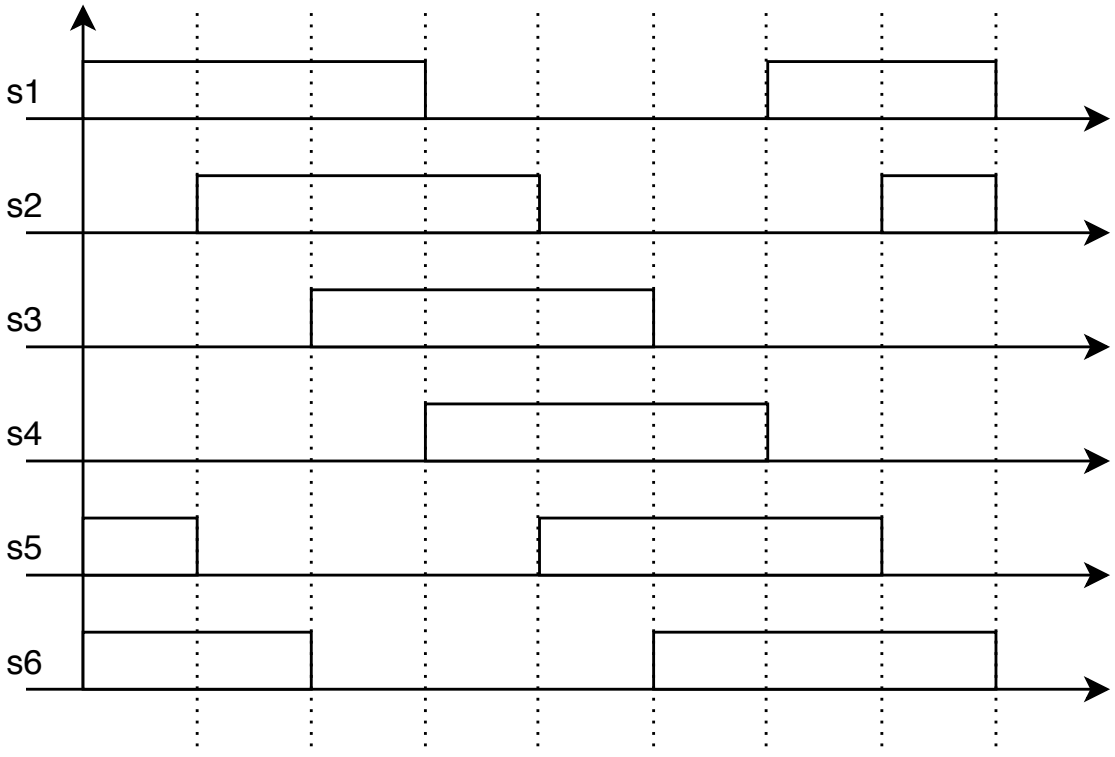


Figura 2: Secuencia de control para el inversor trifásico.

8.5. Parte 5: Máquinas de estados

Ejercicio 8.26. Máquinas de estados en VHDL

Para abordar esta sección se sugiere repasar los ejercicios propuestos durante el capítulo 6. En especial se recomienda retomar los siguientes ejercicios:

- Ejercicio 6.3: Decodificador NZRI
- Ejercicio 6.7: Detector de un patrón de bits

Proceda de la siguiente manera:

- Omita todos los pasos de la resolución que involucren los métodos de síntesis tradicional, tales como la síntesis por mapas de Karnuagh o en forma algebráica.
- Confeccione el diagrama de estados (o tablas de transición equivalente).
- Identifique claramente las entradas y las salidas del circuito.
- Identifique claramente los estados del sistema.
- Identifique claramente como la máquina de estados evoluciona, en función de las entradas del sistema.
- Describa, en lenguaje VHDL, la máquina de estados que ha diseñado.
- Delegue la síntesis del circuito al sintetizador.
- En caso de haber resuelto el problema previamente, compare sus resultados previos con los que ha obtenido del sintetizador.
- Compare los métodos de síntesis empleados en ambas situaciones y extraiga conclusiones.
- Respete los nombres de los puertos de entrada y salida propuestos para la interfaz del módulo.
- Simule el diseño y verifique su correcto funcionamiento.

```
1 entity fsm_moore is
2     port(
3         clk    : in  std_logic;                    -- reloj (flanco ascendente)
4         rst    : in  std_logic;                    -- reset asincronico
5         nrzi   : in  std_logic;                    -- entrada NRZI
6         data   : out std_logic                     -- salida DATA
7     );
8 end fsm_moore;
```

```
1 entity fsm_mealy is
2     port(
3         clk    : in  std_logic;                    -- reloj (flanco ascendente)
4         rst    : in  std_logic;                    -- reset asincronico
5         y      : in  std_logic;                    -- entrada y
6         o      : out std_logic                     -- salida  o
7     );
8 end fsm_mealy;
```

8.6. Parte 6: Máquinas de estados con camino de datos

Ejercicio 8.27. Multiplicador por *sumas sucesivas* y por *sumas y desplazamientos*

El objetivo de este ejercicio es analizar dos arquitecturas diferentes para resolver un mismo cómputo, en particular se pretende multiplicar dos operandos, es decir, se pretende resolver el producto **p = a x b**. Para ello, se propone evaluar dos algoritmos simples, el algoritmo por **sumas sucesivas** y el algoritmo por **sumas y desplazamientos**. Habiendo reconocido las características de cada uno de ellos, proceda al diseño de los circuitos. Se sugiere completar los siguientes pasos:

- Identifique los registros del camino de datos.
- Identifique los diferentes estados del algoritmo (carga de valores iniciales, cálculo, control de flujo, fin del cálculo).
- Diseñe la unidad de control (modelada por una FSM o ASM).
- Confeccione una tabla de transiciones para cada registro del camino de datos. La misma deberá describir como se actualizan los registros en cada estado.
- Dibuje un diagrama RTL del camino de datos.
- Confeccione una tabla de transiciones para el registro de estados. Esta representación deberá ser equivalente al diagrama FSM o ASM previo.
- Escriba el código VHDL que implemente el circuito diseñado. Puede adoptar la metodología de **4 bloques** o bien la de **2 bloques**.
- En la metodología de **4 bloques** se describen en forma separada los siguientes componentes del circuito:
  - Registro de estado.
  - Registros del camino de datos.
  - Lógica de estado futuro del registro de estado.
  - Lógica de estados futuros de los registros del camino de datos.
- En la metodología de **2 bloques** se describen en forma separada los siguientes componentes del circuito
  - Todos los registros del circuito.
  - Toda la lógica de estados futuros de cada uno de los registros del circuito.
- Sintetice el circuito descripto y analice los resultados de síntesis.
- Simule el circuito y determine la cantidad de ciclos de reloj que tarda el circuito en calcular el algoritmo.
- Analice ambas implementaciones y extraiga conclusiones.
- Para ambas implementaciones, respeta la entidad sugerida.

```vhdl
entity multiplier is
  generic(
    W: natural:= 8
  );
  port(
    clk    : in  std_logic;                          -- reloj (flanco ascendente)
    rst    : in  std_logic;                          -- reset asincronico
    en     : in  std_logic;                          -- habilitacion del circuito
    srst   : in  std_logic;                          -- reset sincronico
    start  : in  std_logic;                          -- para iniciar el calculo
    a      : in  std_logic_vector(  W-1 downto 0); -- operando a
    b      : in  std_logic_vector(  W-1 downto 0); -- operando b
    p      : out std_logic_vector(2*W-1 downto 0); -- producto (p = a*b)
    p_end  : out std_logic                           -- indicacion de fin
  );
end multiplier;
```

Ejercicio 8.28. Divisor por *restas sucesivas* y por *restas y desplazamientos*

El objetivo de este ejercicio es analizar dos arquitecturas diferentes para resolver un mismo cómputo, en particular se pretende calcular una división. Recuerde que en una división el **dividendo = divisor * cociente + resto**. Para ello, se propone evaluar dos algoritmos simples, el algoritmo por **restas sucesivas** y el algoritmo por **restas y desplazamientos**. Habiendo reconocido las características de cada uno de ellos, proceda al diseño de los circuitos. Se sugiere completar los siguientes pasos:

- Identifique los registros del camino de datos.
- Identifique los diferentes estados del algoritmo (carga de valores iniciales, cálculo, control de flujo, fin del cálculo).
- Diseñe la unidad de control (modelada por una FSM o ASM).
- Confeccione una tabla de transiciones para cada registro del camino de datos. La misma deberá describir como se actualizan los registros en cada estado.
- Dibuje un diagrama RTL del camino de datos.
- Confeccione una tabla de transiciones para el registro de estados. Esta representación deberá ser equivalente al diagrama FSM o ASM previo.
- Escriba el código VHDL que implemente el circuito diseñado. Puede adoptar la metodología de **4 bloques** o bien la de **2 bloques**.
- En la metodología de **4 bloques** se describen en forma separada los siguientes componentes del circuito:
  - Registro de estado.
  - Registros del camino de datos.
  - Lógica de estado futuro del registro de estado.
  - Lógica de estados futuros de los registros del camino de datos.
- En la metodología de **2 bloques** se describen en forma separada los siguientes componentes del circuito
  - Todos los registros del circuito.
  - Toda la lógica de estados futuros de cada uno de los registros del circuito.
- Sintetice el circuito descripto y analice los resultados de síntesis.
- Simule el circuito y determine la cantidad de ciclos de reloj que tarda el circuito en calcular el algoritmo.
- Analice ambas implementaciones y extraiga conclusiones.
- Para ambas implementaciones, respeta la entidad sugerida.

```
1  entity divider is
2    generic(
3      W: natural:= 8
4    );
5    port(
6      clk    : in  std_logic;                          -- reloj (flanco ascendente)
7      rst    : in  std_logic;                          -- reset asincronico
8      en     : in  std_logic;                          -- habilitacion del circuito
9      srst   : in  std_logic;                          -- reset sincronico
10     start  : in  std_logic;                          -- para iniciar el calculo
11     dvnd   : in  std_logic_vector(W-1 downto 0); -- Dividendo : Recordar que
12     dvsr   : in  std_logic_vector(W-1 downto 0); -- Divisor   : dvnd = q*dvsr + r
13     r      : out std_logic_vector(W-1 downto 0); -- Resto
14     q      : out std_logic_vector(W-1 downto 0); -- Cociente
15     q_end  : out std_logic;                          -- indicacion de fin
16     z      : out std_logic                           -- flag de zero (dvsr = 0)
17   );
18 end divider;
```

8.7. Parte 7: Diseño paramétrico y Diseño estructural

Ejercicio 8.29. Incrementador

- Escriba y sintetice un código HDL que describa un incrementador de N bits.
- Escriba y sintetice un código HDL que describa una celda incrementadora de 1 bit.
- Escriba y sintetice un código HDL que describa un incrementador de N bits en forma estructural, haciendo uso de la celda descripta en el punto anterior.
- En cada caso, respete los nombres de los puertos de entrada y salida propuestos para la interfaz del módulo.
- En cada caso, obtenga su diagrama RTL.
- En cada caso, simule el diseño y verifique su correcto funcionamiento.

```
entity inc_generic is
  generic(
    N: natural:= 16
  );
  port(
    a : in  std_logic_vector(N-1 downto 0);
    ci: in  std_logic;
    o : out std_logic_vector(N-1 downto 0);
    co: out std_logic
  );
end inc_generic;
```

```
entity inc_cell is
  port(
    a : in  std_logic;
    ci: in  std_logic;
    o : out std_logic;
    co: out std_logic
  );
end inc_cell;
```

Ejercicio 8.30. Sumador completo

- Escriba y sintetice un código HDL que describa un sumador completo de N bits.
- Escriba y sintetice un código HDL que describa una celda sumadora de 1 bit.
- Escriba y sintetice un código HDL que describa un sumador de N bits en forma estructural, haciendo uso de la celda descripta en el punto anterior.
- En cada caso, respete los nombres de los puertos de entrada y salida propuestos para la interfaz del módulo.
- En cada caso, obtenga su diagrama RTL.
- En cada caso, simule el diseño y verifique su correcto funcionamiento.

```
entity full_adder_generic is
  generic(
      N  : natural:= 16
    );
  port(
    a  : in  std_logic_vector(N-1 downto 0);
    b  : in  std_logic_vector(N-1 downto 0);
    ci : in  std_logic;
    o  : out std_logic_vector(N-1 downto 0);
    co : out std_logic
  );
end full_adder_generic;
```

```
entity full_adder_cell is
  port(
    a  : in  std_logic;
    b  : in  std_logic;
    ci : in  std_logic;
    o  : out std_logic;
    co : out std_logic
  );
end full_adder_cell;
```

Intencionalmente en blanco.

A. Álgebra de conmutación

Solucion al ejercicio A.1.

Parte A

Puesto que para este caso B y A son variables boolenas, esta afirmación se puede comprobar analizando todos los casos posibles (4 en este caso). Para ello utilizamos la siguiente tabla de verdad:

B	A	$B \cdot A$	$B + A$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Se observa en las filas 1 y 2 de la tabla que $B \cdot A = 0$ y $B + A = 1$ y en ambos casos se comprueba que $A = \overline{B}$, y por lo tanto la afirmación es **verdadera**.

Parte B

Como esta afirmación es general es falsa, basta con mostrar un contra ejemplo.

Sean dos funciones arbitrarias $Z_1 = B \cdot A$ y $Z_2 = \overline{B} \cdot \overline{A}$ se tendrá que:

X	Y	Z_1	Z_2	$Z_1 \cdot Z_2$	$Z_1 + Z_2$
0	0	0	1	0	1
0	1	0	0	0	0
1	0	0	0	0	0
1	1	1	0	0	1

Se observa en las filas 0 y 3 de la tabla que $Z_1 \cdot Z_2 = 0$ y $Z_1 + Z_2 = 1$ pero de acuerdo a como se han definidos las funciones $Z_1 \ncong Z_2$, con lo cual queda demostrado que la afirmación propuesta es **falsa**.

Solucion al ejercicio A.2.

1. $A + 1 = 1$
2. $A + 0 = A$
3. $A + A = A$
4. $A + \overline{A} = 1$
5. $(A + B) + C = A + (B + C)$
6. $A(A + B) = A$
7. $A(\overline{A} + B) = A + B$
8. $(A + B)(A + \overline{B}) = A$
9. $\overline{A + B + C} = \overline{A} \cdot \overline{B} \cdot \overline{C}$
10. $(A + B)(\overline{B} + C)(A + C) = (A + B)(\overline{B} + C)$

Solucion al ejercicio A.3.

Las expresiones finales para cada caso son:

- $F_1 = A + B + C + D$
- $F_2 = CB + \overline{B}A + \overline{D}$
- $F_3 = \overline{A} + \overline{D}B$
- $F_4 = \overline{B} + \overline{D} \cdot \overline{C}$

Solución sugerida por Fernando, alumno del curso R3052 del ciclo lectivo 2020.

$$
\begin{aligned}
F_1 &= A + B \cdot \overline{A} + C \cdot \overline{(B+A)} + D \cdot \overline{(C+B+A)} \\
&= A + B + C \cdot \overline{(B+A)} + D \cdot \overline{(C+B+A)} \\
&\textit{tomando } X = A + B \textit{ entonces} \\
&= X + C\overline{X} + D \cdot \overline{(C+B+A)} \\
&= X + C + D \cdot \overline{(C+B+A)} \\
&= A + B + C + D \cdot \overline{(C+B+A)} \\
&\textit{tomando } Y = A + B + C \textit{ entonces} \\
&= Y + D \cdot \overline{Y} \\
&= Y + D \\
&= A + B + C + D
\end{aligned}
$$

$$
\begin{aligned}
F_2 &= \overline{B} \cdot A + C \cdot A + D \cdot C \cdot B + \overline{D} \\
&= \overline{B} \cdot A + C \cdot A + \{D \cdot C \cdot B + \overline{D}\} \\
&= \overline{B} \cdot A + C \cdot A + \{C \cdot B + \overline{D}\} \\
&= \{\overline{B} \cdot A + C \cdot A + C \cdot B\} + \overline{D} \\
&= \overline{B} \cdot A + C \cdot B + \overline{D}
\end{aligned}
$$

$$
\begin{aligned}
F_3 &= \overline{A} + \overline{B} \cdot \overline{A} + \overline{D} \cdot C \cdot B + \overline{D} \cdot B \\
&= \{\overline{A} + \overline{B} \cdot \overline{A}\} + \overline{D} \cdot C \cdot B + \overline{D} \cdot B \\
&= \overline{A} + \{\overline{D} \cdot C \cdot B + \overline{D} \cdot B\} \\
&= \overline{A} + \overline{D} \cdot B
\end{aligned}
$$

$$
\begin{aligned}
F_4 &= C \cdot \overline{B} \cdot A + (\overline{C} + \overline{B}) \cdot (\overline{D} + \overline{B}) + \overline{(C+B+A)} \\
&= C.\overline{B}.A + \overline{C}.\overline{B}.\overline{A} + (\overline{C} + \overline{B}) \cdot (\overline{D} + \overline{B}) \\
&= C.\overline{B}.A + \overline{C}.\overline{B}.\overline{A} + \overline{C}.\overline{D} + \overline{C}.\overline{B} + \overline{B}.\overline{D} + \overline{B} \\
&= \overline{C}.\overline{D} + \overline{B}(C.A + \overline{C}.\overline{A} + \overline{C} + \overline{D} + 1) \\
&= \overline{C}.\overline{D} + \overline{B}
\end{aligned}
$$

Solucion al ejercicio A.4.

Si $F(B, A) = B \cdot A$ entonces $F^D(B, A) = B + A$

$\overline{F(\overline{B}, \overline{A})} = \overline{\overline{B} \cdot \overline{A}} = \overline{\overline{B}} + \overline{\overline{A}} = B + A = F^D(B, A)$

$F(\overline{B}, \overline{A}) = \overline{B} \cdot \overline{A} = \overline{B + A} = \overline{F^D(B, A)}$

Solucion al ejercicio A.5.

Parte A

$$\begin{aligned}
F(C,B,A) &= \overline{C}\overline{B} + CA \\
&= Cf(1,B,A) + \overline{C}f(0,B,A) \\
&= C(Bf(1,1,A) + \overline{B}f(1,0,A)) + \overline{C}(Bf(0,1,A) + \overline{B}f(0,0,A)) \\
&= C.B.f(1,1,A) + C.\overline{B}.f(1,0,A) + \overline{C}.B.f(0,1,A) + \overline{C}.\overline{B}.f(0,0,A) \\
&= C.B.A.f(1,1,1) + C.B.\overline{A}.f(1,1,0) + C.\overline{B}.A.f(1,0,1) + C.\overline{B}.\overline{A}.f(1,0,0) + \\
&\quad \overline{C}.B.A.f(0,1,1) + \overline{C}.B.\overline{A}.f(0,1,0) + \overline{C}.\overline{B}.A.f(0,0,1) + \overline{C}.\overline{B}.\overline{A}.f(0,0,0) \\
&= C.B.A + C.\overline{B}.A + C.\overline{B}.A + \overline{C}.B.A + \overline{C}.B.\overline{A} + \overline{C}.\overline{B}.A + \overline{C}.\overline{B}.\overline{A} \\
&= \sum_3(0,1,2,3,4,5,7) = \prod_3(6)
\end{aligned}$$

Parte B

$f(x_1,x_2,x_3,\ldots,x_n) = (x_1 + f(0,x_2,x_3,...x_n)) \cdot (\overline{x_1} + f(1,x_2,x_3,\ldots,x_n))$

Solucion al ejercicio A.6.

Parte B

Para la función $Z_1(D,C,B,A) = \sum_4(0,3,6,9,12,15)$ se obtienen las siguientes equivalencias:

$$\begin{aligned}
Z_1 &= \overline{D}.\overline{C}.\overline{B}.\overline{A} + \overline{D}.\overline{C}.B.A + \overline{D}.C.B.\overline{A} + D.\overline{C}.\overline{B}.A + D.C.\overline{B}.\overline{A} + D.C.B.A \\
&= \prod_4(1,2,4,5,7,8,10,11,13,14) \\
&= (D+C+B+\overline{A}) \cdot (D+C+\overline{B}+A) \ldots (\overline{D}+\overline{C}+\overline{B}+A)
\end{aligned}$$

Parte C

La función no puede minimizarse por mapas de Karnaugh para obtener un expresión de suma de productos mínima.

La expresión mínima en forma de producto de sumas es:

$Z_1 = (D+\overline{C}+B)(D+\overline{C}+\overline{A})(D+B+\overline{A})(\overline{C}+B+\overline{A})(\overline{D}+C+A)(\overline{D}+C+\overline{B})(\overline{D}+\overline{B}+A)(C+\overline{B}+A)$

Solucion al ejercicio A.7.

a) Se requieren al menos 3 personas, a saber D,C y B.

b) $F = \sum_5(13,14,15,21,23,28,29,30,31)$

c) C ya que es el único que tiene la llave para la cerradura C2.

Solucion al ejercicio A.8.

Solución sugerida por Juan Pablo, alumno del curso R3052 del ciclo lectivo 2020.
Parte A

$$C \wedge B \Rightarrow Peligroso\ (P = 1)$$
$$C \wedge B \wedge A \Rightarrow No\ Peligroso\ (P = 0 \vee S = 1)$$
$$D \wedge C \Rightarrow Peligroso\ (P = 1)$$
$$D \wedge C \wedge A \Rightarrow No\ Peligroso\ (P = 0 \vee S = 1)$$

	D	C	B	A	P	S
00	0	0	0	0	0	1
01	0	0	0	1	0	1
02	0	0	1	0	0	1
03	0	0	1	1	0	1
04	0	1	0	0	0	1
05	0	1	0	1	0	1
06	0	1	1	0	1	0
07	0	1	1	1	0	1
08	1	0	0	0	0	1
09	1	0	0	1	0	1
10	1	0	1	0	0	1
11	1	0	1	1	0	1
12	1	1	0	0	1	0
13	1	1	0	1	0	1
14	1	1	1	0	1	0
15	1	1	1	1	0	1

$$P = \sum_{4}(6, 12, 14)$$
$$P = \overline{D}.C.B.\overline{A} + D.C.\overline{B}.\overline{A} + D.C.B.\overline{A}$$

$$S = \sum_{4}(0, 1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 13, 15)$$
$$\begin{aligned} S = {} & \overline{D}.\overline{C}.\overline{B}.\overline{A} + \overline{D}.\overline{C}.\overline{B}.A + \\ & \overline{D}.\overline{C}.B.\overline{A} + \overline{D}.\overline{C}.B.A + \\ & \overline{D}.C.\overline{B}.\overline{A} + \overline{D}.C.\overline{B}.A + \\ & \overline{D}.C.B.A + D.\overline{C}.\overline{B}.\overline{A} + \\ & D.\overline{C}.\overline{B}.A + D.\overline{C}.B.\overline{A} + \\ & D.\overline{C}.B.A + D.C.\overline{B}.A + \\ & D.C.B.A \end{aligned}$$

Solucion al ejercicio A.9.

Parte A

I_3	I_2	I_1	I_0	M
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Parte B

$$M = \sum_4 (7, 11, 13, 14, 15)$$
$$M = \prod_4 (0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 12)$$

Parte C

$$M_{SPm} = I_3.I_2.I_0 + I_3.I_2.I_1 + I_2.I_1.I_0 + I_3.I_1.I_0$$
$$M_{PSm} = (I_2 + I_0)(I_2 + I_1)(I_3 + I_0)(I_3 + I_1)(I_1 + I_0)(I_2 + I_3)$$

Parte D

Partiendo de la expresión SPF de la parte B, el completo de M será

$$\overline{M} = \sum_4 (0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 12)$$

Parte E

Partiendo de la expresión SPF de la parte B, el dual de M será

$$M^D = \sum_4 (3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15)$$

B. Diseño de circuitos combinacionales con compuertas

Solucion al ejercicio B.1.

1. NOR
2. NAND
3. XNOR
4. Inversor
5. Buffer
6. XOR
7. OR
8. AND

Solucion al ejercicio B.2.

Cada una de las propiedades se puede verificar en **forma algebraica**, como por ejemplo:

$$A \oplus 0 = A \cdot \overline{0} + \overline{A} \cdot 0 = A \cdot 1 + 0 = A$$
$$\therefore A \oplus 0 = A$$

O bien de **forma exhaustiva** (probando todos los casos posibles) con una tabla de verdad:

A	$f = A \oplus 0$
0	0
1	1

1. Se pretende demostrar que $A \oplus 0 = A$.

$$\begin{aligned} A \oplus 0 &= A \cdot \overline{0} + \overline{A} \cdot 0 \\ &= A \cdot 1 + 0 \\ &= A \end{aligned}$$

2. Se pretende demostrar que $A \oplus 1 = \overline{A}$

$$\begin{aligned} A \oplus 1 &= A \cdot \overline{1} + \overline{A} \cdot 1 \\ &= A \cdot 0 + \overline{A} \cdot 1 \\ &= \overline{A} \end{aligned}$$

3. Se pretende demostrar que $A \oplus A = 0$

$$\begin{aligned} A \oplus A &= A \cdot \overline{A} + \overline{A} \cdot A \\ &= 0 + 0 \\ &= 0 \end{aligned}$$

4. Se pretende demostrar que $A \oplus \overline{A} = 1$

$$\begin{aligned} A \oplus \overline{A} &= A \cdot \overline{\overline{A}} + \overline{A} \cdot \overline{A} \\ &= A + \overline{A} \\ &= 1 \end{aligned}$$

5. Se pretende demostrar que $\overline{B} \oplus A = B \oplus \overline{A} = \overline{B \oplus A}$

$$\begin{aligned}
\overline{B} \oplus A = B \oplus \overline{A} = \overline{B \oplus A} \\
\therefore \quad \overline{B} \oplus A = \overline{B \oplus A} \wedge B \oplus \overline{A} = \overline{B \oplus A} \\
\overline{B} \oplus A &= \overline{B} \cdot \overline{A} + \overline{\overline{B}} \cdot A \\
&= \overline{B} \cdot \overline{A} + B \cdot A \\
&= \overline{B \oplus A} \\
B \oplus \overline{A} &= B \cdot \overline{\overline{A}} + \overline{B} \cdot \overline{A} \\
&= B \cdot A + \overline{B} \cdot \overline{A} \\
&= \overline{B \oplus A} \\
\therefore \quad \overline{B} \oplus A = B \oplus \overline{A}
\end{aligned}$$

6. Se pretende demostrar que $Dual(B \oplus A) = (B \oplus A)^D = \overline{B \oplus A}$

$$\begin{aligned}
(B \oplus A)^D &= \overline{\overline{B} \oplus \overline{A}} \\
&= \overline{\overline{B} \cdot \overline{\overline{A}} + \overline{\overline{B}} \cdot \overline{A}} \\
&= \overline{\overline{B} \cdot A + B \cdot \overline{A}} \\
&= \overline{B \oplus A}
\end{aligned}$$

7. Se pretende demostrar que $C \cdot (B \oplus A) = (C \cdot B) \oplus (C \cdot A)$

$$\begin{aligned}
(C \cdot B) \oplus (C \cdot A) &= \overline{CB} \cdot CA + CB \cdot CA \\
&= (\overline{C} + \overline{B}) \cdot CA + CB \cdot (\overline{C} + \overline{A}) \\
&= \overline{C}CA + C\overline{B}A + C\overline{C}B + CB\overline{A} \\
&= C\overline{B}A + CB\overline{A} \\
&= C \cdot (\overline{B}A + B\overline{A}) \\
&= C \cdot (B \oplus A)
\end{aligned}$$

Solucion al ejercicio B.3.

B(3)	B(2)	B(1)	B(0)	D(3)	D(2)	D(1)	D(0)
0	0	0	0	1	1	1	1
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	1
0	0	1	1	0	0	1	0
0	1	0	0	0	0	1	1
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	1
0	1	1	1	0	1	1	0
1	0	0	0	0	1	1	1
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	1
1	0	1	1	1	0	1	0
1	1	0	0	1	0	1	1
1	1	0	1	1	1	0	0
1	1	1	0	1	1	0	1
1	1	1	1	1	1	1	0

Por simple inspección se concluye que $D(0) = \overline{B(0)}$.

A modo de ejemplo se muestra que la expresión en forma de suma de productos para **D(3)** es:

$$D(3) = \overline{B(3)}.\overline{B(2)}.\overline{B(1)}.\overline{B(0)} + B(3).B(2) + B(3).B(1) + B(3).B(0)$$

En tanto que su expresión en forma de producto de sumas es:

$$D(3) = (\overline{B(3)} + B(2) + B(1) + B(0)).(B(3) + \overline{B(2)}).(B(3) + \overline{B(1)}).(B(3) + \overline{B(0)}$$

Solucion al ejercicio B.4.

Resolución a cargo del lector.

Solucion al ejercicio B.5.

B(3)	B(2)	B(1)	B(0)	Co	D(3)	D(2)	D(1)	D(0)
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	0
0	0	1	0	0	0	1	0	0
0	0	1	1	0	0	1	1	0
0	1	0	0	0	1	0	0	0
0	1	0	1	1	0	0	0	0
0	1	1	0	1	0	0	1	0
0	1	1	1	1	0	1	0	0
1	0	0	0	1	0	1	1	0
1	0	0	1	1	1	0	0	0
1	0	1	0	-	-	-	-	-
1	0	1	1	-	-	-	-	-
1	1	0	0	-	-	-	-	-
1	1	0	1	-	-	-	-	-
1	1	1	0	-	-	-	-	-
1	1	1	1	-	-	-	-	-

Para el caso de la función Co, y considerando que todos los *don't care* se adoptan como unos lógicos, se obtiene la siguiente expresión mínima:

$$Co = B(3) + B(2).B(1) + B(2).B(0)$$

Solucion al ejercicio B.6.

Resolución a cargo del lector.

Solucion al ejercicio B.7.

Sea $B \in B^3$ la palabra de entrada, entonces $max\{B\} = 2^3 - 1 = 7$, luego $7 \times 3 = 21$ y puesto que $21 > 16 = 2^4$ y $21 < 32 = 2^5$, se requieren al menos 5 bits para contener la palabra de salida T.

B(2)	B(1)	B(0)	T(4)	T(3)	T(2)	T(1)	T(0)
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	1
0	1	0	0	0	1	1	0
0	1	1	0	1	0	0	1
1	0	0	0	1	1	0	0
1	0	1	0	1	1	1	1
1	1	0	1	0	0	1	0
1	1	1	1	0	1	0	1

$$T(0) = B(0)$$
$$T(1) = (B(1) + B(0)) \cdot (\overline{B(1)} + \overline{B(0)})$$
$$T(2) = (\overline{B(2)} + \overline{B(1)} + B(0)) \cdot (B(2) + B(1)) \cdot (B(2) + \overline{B(0)})$$
$$T(3) = (B(2) + B(1)) \cdot (\overline{B(1)} + B(0))$$
$$T(4) = B(2) \cdot B(1)$$

Solucion al ejercicio B.8.

B(1)	B(0)	A(1)	A(0)	P	P(3)	P(2)	P(1)	P(0)
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0
0	0	1	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0
0	1	0	0	0	0	0	0	0
0	1	0	1	1	0	0	0	1
0	1	1	0	2	0	0	1	0
0	1	1	1	3	0	0	1	1
1	0	0	0	0	0	0	0	0
1	0	0	1	2	0	0	1	0
1	0	1	0	4	0	1	0	0
1	0	1	1	6	0	1	1	0
1	1	0	0	0	0	0	0	0
1	1	0	1	3	0	0	1	1
1	1	1	0	6	0	1	1	0
1	1	1	1	9	1	0	0	1

$$P(0) = B(0).A(0)$$
$$P(1) = \overline{B(1)}.B(0).A(1) + B(1).\overline{B(0)}.A(0) + A(1).\overline{A(0)}.B(0) + \overline{A(1)}.A(0).B(1)$$
$$P(2) = B(1).A(1).\overline{A(0)} + B(1).\overline{B(0)}.A(1)$$
$$P(3) = B(1).B(0).A(1).A(0)$$

Solucion al ejercicio B.9.

Resolución a cargo del lector.

Solucion al ejercicio B.10.

Resolución a cargo del lector.

Solucion al ejercicio B.11.

Resolución a cargo del lector.

Solucion al ejercicio B.12.

Resolución a cargo del lector.

Solucion al ejercicio B.13.

Resolución a cargo del lector.

Nota del autor: *Lectura(s) segurida(s): [1].*

Solucion al ejercicio B.14.

Resolución a cargo del lector.

Nota del autor: *Lectura(s) segurida(s): [1].*

C. Tecnologías y estándares del hardware digital

Solucion al ejercicio C.1. Tecnología CMOS

Resolución a cargo del lector.

Solucion al ejercicio C.2. Tecnología CMOS

Resolución a cargo del lector.

Solucion al ejercicio C.3. Tecnología CMOS

Resolución a cargo del lector.

Solucion al ejercicio C.4. Consumo de energía

Resolución a cargo del lector.

Solucion al ejercicio C.5. Consumo de energía

Resolución a cargo del lector.

Solucion al ejercicio C.6. Consumo de energía

Resolución a cargo del lector.

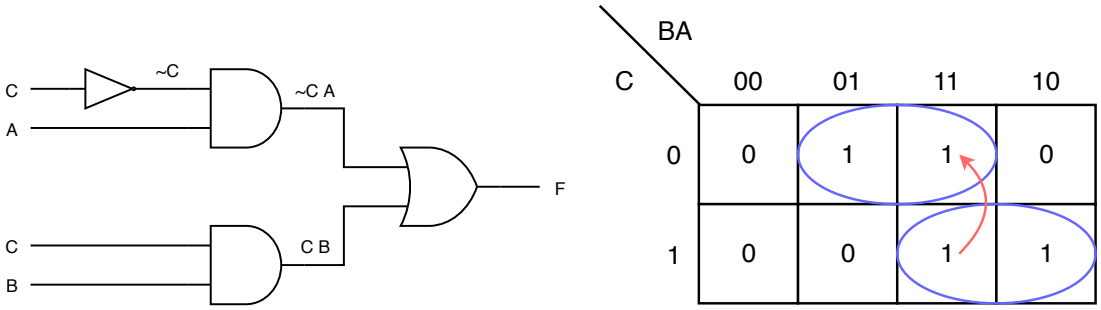
Solucion al ejercicio C.7. Riesgos

Resolución a cargo del lector.

Nota del autor: *Lectura(s) segurida(s): [2] [3].*

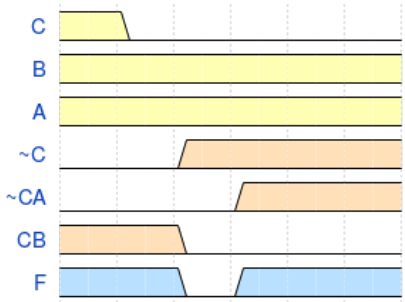
Solucion al ejercicio C.8. Riesgos

a) Implementación de F como suma de productos.



b) Considerando como estado inicial la combinación de variables $C = B = A = 1$, se observa que al momento que C cambie de $1 \rightarrow 0$, el circuito combinacional estará sometido a la combinación $C = 0 \wedge B = A = 1$. En ambos casos la función f toma un valor de salida igual a 1, tal y como puede observarse en el siguiente mapa. En **rojo** se marca la transición propuesta.

c) Para visualizar la ocurrencia de una transición espúrea a la salida, se asumirá que todas las compuertas tienen retardo unitario, exceptuando la compuerta final (OR), teniendo presente que esta compuerta sólo produce un retardo adicional en la salida, y no es causante de la transición indeseada.



d) Para eliminar el riesgo estático, se debe agregar el término de consenso en la función lógica. Del mapa de la figura del inciso a) puede observarse que el consenso corresponde al término $B \cdot A$. Agregar esta compuerta redundante permite eliminar el riesgo estático del circuito a expensas de aumentar el área en la implementación del mismo.

Solucion al ejercicio C.9. Riesgos

Resolución a cargo del lector.

D. Aritmética binaria y circuitos aritméticos

Solucion al ejercicio D.1. Operación de suma con operandos no signados

Resolución a cargo del lector.

Solucion al ejercicio D.2. Operación de suma con operandos no signados

Resolución a cargo del lector.

Solucion al ejercicio D.3. Operación de suma con operandos no signados

Recordando que para un RCA sus salidas S y Co son

$$S(k) = Ci(k) \oplus B(k) \oplus A(k)$$
$$Co(k) = B(k).A(k) + Ci(k).(B(k) \oplus A(k))$$

Para un sumador de 4 bits se tendrá

$$S(0) = Ci(0) \oplus B(0) \oplus A(0) \qquad Co(0) = B(0).A(0) + Ci(0).(B(0) \oplus A(0))$$
$$S(1) = Ci(1) \oplus B(1) \oplus A(1) \qquad Co(1) = B(1).A(1) + Ci(1).(B(1) \oplus A(1))$$
$$S(2) = Ci(2) \oplus B(2) \oplus A(2) \qquad Co(2) = B(2).A(2) + Ci(2).(B(2) \oplus A(2))$$
$$S(3) = Ci(3) \oplus B(3) \oplus A(3) \qquad Co(3) = B(3).A(3) + Ci(3).(B(3) \oplus A(3))$$

Puesto que se pretende construir un incrementador tal que $S = A + 2$, entonces el operando B será igual a 0010_2, por consiguiente $B(3) = B(2) = B(0) = 0$ y $B(1) = 1$. Así pues:

$$S(0) = Ci(0) \oplus 0 \oplus A(0) \qquad Co(0) = 0.A(0) + Ci(0).(0 \oplus A(0))$$
$$S(1) = Ci(1) \oplus 1 \oplus A(1) \qquad Co(1) = 1.A(1) + Ci(1).(1 \oplus A(1))$$
$$S(2) = Ci(2) \oplus 0 \oplus A(2) \qquad Co(2) = 0.A(2) + Ci(2).(0 \oplus A(2))$$
$$S(3) = Ci(3) \oplus 0 \oplus A(3) \qquad Co(3) = 0.A(3) + Ci(3).(0 \oplus A(3))$$

Operando algebraicamente tendremos:

$$S(0) = Ci(0) \oplus A(0) \qquad Co(0) = Ci(0).A(0)$$
$$S(1) = Ci(1) \oplus \overline{A(1)} \qquad Co(1) = A(1) + Ci(1)$$
$$S(2) = Ci(2) \oplus A(2) \qquad Co(2) = Ci(2).A(2)$$
$$S(3) = Ci(3) \oplus A(3) \qquad Co(3) = Ci(3).A(3)$$

Finalmente, al usar el sumador como incrementador la entrada $Ci(0)$ deberá ser igual a 0 y por consiguiente tendremos:

$$S(0) = A(0) \qquad Co(0) = 0$$
$$S(1) = \overline{A(1)} \qquad Co(1) = A(1)$$
$$S(2) = Ci(2) \oplus A(2) \qquad Co(2) = Ci(2).A(2)$$
$$S(3) = Ci(3) \oplus A(3) \qquad Co(3) = Ci(3).A(3)$$

Téngase presente para las expresiones de $S(1)$ y $Co(1)$ que $Co(0) = Ci(1)$.

Solucion al ejercicio D.4. Operación de resta con operandos no signados

Resolución a cargo del lector.

Solucion al ejercicio D.5. Operación de resta con operandos no signados

Número binario	Complemento a 1	Complemento a 2
10011100	01100011	01100100
10011101	01100010	01100011
10101000	01010111	01111000
01111111	10000000	10000001
00000000	11111111	00000000
10000000	01111111	10000000

Solucion al ejercicio D.6. Operación de resta con operandos no signados

a) Implementación en dos niveles.

Sea **X(3:0)** la palabra a convertir y sea **Y(3:0)** el complemento a dos de **X(3:0)**, el funcionamiento del circuito estará dado por la siguiente tabla de verdad:

X(3)	X(2)	X(1)	X(0)	Y(3)	Y(2)	Y(1)	Y(0)
0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1
0	0	1	0	1	1	1	0
0	0	1	1	1	1	0	1
0	1	0	0	1	1	0	0
0	1	0	1	1	0	1	1
0	1	1	0	1	0	1	0
0	1	1	1	1	0	0	1
1	0	0	0	1	0	0	0
1	0	0	1	0	1	1	1
1	0	1	0	0	1	1	0
1	0	1	1	0	1	0	1
1	1	0	0	0	1	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	0	1	0
1	1	1	1	0	0	0	1

Haciendo uso de mapas de Karnaugh se /den derivas las siguientes expresiones:

$$Y(0) = X(0)$$
$$Y(1) = X(1).\overline{X(0)} + \overline{X(1)}.X(0)$$
$$Y(2) = X(2).\overline{X(1)}.\overline{X(0)} + \overline{X(2)}.X(1) + \overline{X(2)}.X(0)$$
$$Y(3) = X(3).\overline{X(2)}.\overline{X(1)}.\overline{X(0)} + \overline{X(3)}.X(2) + \overline{X(3)}.X(1) + \overline{X(3)}.X(0)$$

b) Implementación con celdas.

Se usará el resultado anterior para hallar la regularidad de las expresiones a fin de obtener la celda buscada. Partiendo de las ecuaciones de **Y(3:0)** se tendrá que:

$$\begin{aligned}
Y(1) &= X(1).\overline{X(0)} + \overline{X(1)}.X(0) \\
&= X(1) \oplus X(0) \\
Y(2) &= X(2).\overline{X(1)}.\overline{X(0)} + \overline{X(2)}.X(1) + \overline{X(2)}.X(0) \\
&= X(2).\overline{X(1) + X(0)} + \overline{X(2)}.(X(1) + X(0)) \\
&= X(2) \oplus (X(1) + X(0)) \\
Y(3) &= X(3).\overline{X(2)}.\overline{X(1)}.\overline{X(0)} + \overline{X(3)}.X(2) + \overline{X(3)}.X(1) + \overline{X(3)}.X(0) \\
&= X(3).\overline{X(2) + X(1) + X(0)} + \overline{X(3)}.(X(2) + X(1) + X(0)) \\
&= X(3) \oplus (X(2) + X(1) + X(0)) \\
Y(0) &= X(0) \\
&= X(0) \oplus 0
\end{aligned}$$

Se puede inferir entonces que, en el caso general, **Y(n)** será

$$\begin{aligned}
Y(n) &= X(n) \oplus (X(n-1) + X(n-2) + \cdots + X(1) + X(0)) \\
Y(n) &= X(n) \oplus p(n-1)
\end{aligned}$$

en donde hemos llamado $p(n-1)$ a la señal de propagación de entrada que involucra a todas las etapas previas. Llamaremos ahora $p(n)$ a la señal de propagación de salida que involucra tanto a $p(n-1)$ como a la entrada de la etapa n-ésima $x(n)$, así pues $p(n)$ será igual a:

$$\begin{aligned}
p(n) &= X(n) + X(n-1) + X(n-2) + \cdots + X(1) + X(0) \\
p(n) &= X(n) + p(n-1)
\end{aligned}$$

Finalmente tendremos la celda iterativa cuyas salidas $y(n)$ y $p(n)$ se vinculan con las entrada $x(n)$ y $p(n-1)$ de la siguiente manera:

$$\begin{aligned}
Y(n) &= X(n) \oplus p(n-1) \\
p(n) &= X(n) + p(n-1)
\end{aligned}$$

Solucion al ejercicio D.7. Operación de resta con operandos no signados

---

a)	11111 - 10000	b)	10110 - 11111	c)	1011110 - 1011110	d)	101 - 101000
	11111		10110		1011110		101
-	10000	-	11111	-	1011110	-	101000
	11111		10110		1011110		000101
+	01111	+	00000	+	0100001	+	010111
	00001		00001		0000001		000001
	101111		010111		10000000		0011101

a) Puesto que el bit de **borrow** es igual a **1**, la resta es **válida**.

b) Puesto que el bit de **borrow** es igual a **0**, la resta es **no válida**.

c) Puesto que el bit de **borrow** es igual a **1**, la resta es **válida**.

d) Puesto que el bit de **borrow** es igual a **0**, la resta es **no válida**.

Solucion al ejercicio D.8. Operación de multiplicación con operandos no signados

Resolución a cargo del lector.

Solucion al ejercicio D.9. Operación de multiplicación con operandos no signados

Resolución a cargo del lector.

Solucion al ejercicio D.10. Operación de multiplicación con operandos no signados

Resolución a cargo del lector.

Solucion al ejercicio D.11. Operación de suma y resta con operandos signados

Resolución a cargo del lector.

Solucion al ejercicio D.12. Operación de suma y resta con operandos signados

a) Números no signados.

Si X es un número no signado de 8 bits entonces el rango de representación de X es $[0, 2^8 - 1] = [0, 255]$, de donde $0_{10} = 0000\_0000_2$ y $255_{10} = 1111\_1111_2$.

Si Y es la representación extendida de X en 16 bits, se tendrá entonces que

$$Y(15:08) = 0000\_0000$$
$$Y(07:00) = X(07:00)$$

Por ejemplo, el número 255_{10}, cuya representación en **8 bits** es 1111_1111_2 tendrá una representación equivalente en **16 bits** igual a $0000\_0000\_1111\_1111_2$.

b) Números signados.

Si X es un número signado de 8 bits entonces el rango de representación de X es $[-2^{8-1}, +2^{8-1}-1] = [-2^7, +2^7-1] = [-128, +127]$, de donde $-128_{10} = 1000\_0000_2$ y $+127_{10} = 0111\_1111_2$.

Si Y es la representación extendida de X en 16 bits, para el caso de los números mayores o iguales a 0_{10}, es decir aquellos números cuyo bit más significativo sea 0_2, se tendrá que:

$$Y(15:08) = 0000\_0000$$
$$Y(07:00) = X(07:00)$$

Por ejemplo, el número $+127_{10}$, cuya representación en **8 bits** es 0111_1111_2, tendrá una representación equivalente en **16 bits** igual a $0000\_0000\_0111\_1111_2$.

Para el caso de los números menores a 0_{10}, es decir aquellos números cuyo bit menos significativo sea 0_2, se tendrá que buscar la forma de extender la longitud de la palabra sin alterar el valor que éste representa. Para estos casos se procede de la siguiente manera:

$$Y(15:08) = 1111\_1111$$
$$Y(07:00) = X(07:00)$$

Por ejemplo, el número -128_{10}, cuya representación en **8 bits** es 1000_0000_2, tendrá una representación equivalente en **16 bits** igual a $1111\_1111\_1000\_0000_2$. Para demostrar la validad de este afirmación analicemos las formas polinómicas de cada una de estas palabras.

Si $X = 1000\_0000_2$ y X se encuentra codificado en Código de Complemento a 2, su equivalente decimal se obtiene a partir de la siguiente expresión

$$X_{10} = -2^{W-1} + \sum_{i=0}^{W-2} 2^i \cdot b_i \iff b_{w-1} = 1$$
$$X_{10} = \sum_{i=0}^{W-2} 2^i \cdot b_i \iff b_{w-1} = 0$$

Así pues, se tendrá que:

$$\begin{aligned} X_{10} &= -2^{W-1} + \sum_{i=0}^{W-2} 2^i \cdot b_i \iff b_{w-1} = 1 \\ &= -2^7 + 2^6 \cdot 0 + 2^5 \cdot 0 + 2^4 \cdot 0 + 2^3 \cdot 0 + 2^2 \cdot 0 + 2^1 \cdot 0 + 2^0 \cdot 0 \\ &= -128 \end{aligned}$$

Al extender la longitud de X a 16 bits, afirmamos que su representación equivalente era $1111\_1111\_1000\_0000_2$, para la cual se tendrá que:

$$\begin{aligned} X_{10} &= -2^{W-1} + \sum_{i=0}^{W-2} 2^i \cdot b_i \iff b_{w-1} = 1 \\ &= -2^{15} + 2^{14} \cdot 1 + \cdots + 2^7 \cdot 1 + 2^6 \cdot 0 + \cdots + 2^0 \cdot 0 \\ &= -32768 + 16384 + \cdots + 128 + 0 + \cdots + 0 \\ &= -128 \end{aligned}$$

Quedando demostrada la equivalencia de las representaciones.

Solucion al ejercicio D.13. Operación de suma y resta con operandos signados

Sean

- A(3), B(3) y R(3) los bits más significativos de los operandos A y B y de la suma respectivamente,
- Ci(3) la entrada de acarreo a la celda más significativa del sumador y
- Co(3) la salida de acarreo de la celda más significativa del sumador

a) V en función de los signos de los operandos y del signo del resultado.

R(3)	B(3)	A(3)	V	Observaciones
0	0	0	0	La operación resulta exitosa.
0	0	1	0	La operación resulta exitosa.
0	1	0	0	La operación resulta exitosa.
0	1	1	1	La suma de dos números negativos no puede ser positiva.
1	0	0	1	La suma de dos números positivos no puede ser negativa.
1	0	1	0	La operación resulta exitosa.
1	1	0	0	La operación resulta exitosa.
1	1	1	0	La operación resulta exitosa.

Finalmente, la expresión de V será $V = \overline{S(3)}.B(3).A(3) + S(3).\overline{B(3)}.\overline{A(3)}$

b) V en función de los signos de los operandos y del acarreo de entrada a la etapa.

Ci(3)	B(3)	A(3)	V	Observaciones
0	0	0	0	Suma válida.
0	0	1	0	Suma válida.
0	1	0	0	Suma válida.
0	1	1	1	Suma no válida.
1	0	0	1	Suma no válida.
1	0	1	0	Suma válida.
1	1	0	0	Suma válida.
1	1	1	0	Suma válida.

Finalmente, la expresión de V será $V = \overline{Ci(3)}.B(3).A(3) + Ci(3).\overline{B(3)}.\overline{A(3)}$

c) V en función de los signos de los operandos y de los acarreos .

Co(3)	Ci(3)	B(3)	A(3)	V	Observaciones
0	0	0	0	0	Suma válida.
0	0	0	1	0	Suma válida.
0	0	1	0	0	Suma válida.
0	0	1	1	X	Do not care.
0	1	0	0	1	Suma inválida.
0	1	0	1	X	Do not care.
0	1	1	0	X	Do not care.
0	1	1	1	X	Do not care.
1	0	0	0	X	Do not care.
1	0	0	1	X	Do not care.
1	0	1	0	X	Do not care.
1	0	1	1	1	Suma inválida.
1	1	0	0	X	Do not care.
1	1	0	1	0	Suma válida.
1	1	1	0	0	Suma válida.
1	1	1	1	0	Suma válida.

Finalmente, la expresión de V será $V = \overline{Co(3)}.Ci(3) + Co(3).\overline{Ci(3)} = Co(3) \oplus Ci(3)$

Solucion al ejercicio D.14. Operación de suma y resta con operandos signados

Resolución a cargo del lector.

Solucion al ejercicio D.15. Comparador por contracción

Primer enfoque a la resolución del problema

Asumiremos que contamos con un circuito restador con la siguiente interfaz de entrada/salida

$A(3:0)$	entrada	Minuendo de 4 bits
$B(3:0)$	entrada	Sustraendo de 4 bits
$R(3:0)$	salida	Resta de 4 bits
Bw	salida	Préstamo - Señal activa en alto

Los posibles resultados para la resta serán

$$A - B > 0 \iff A > B$$
$$A - B = 0 \iff A = B$$
$$A - B < 0 \iff A < B$$

Pero el resultado de la resta a priori no dice nada respecto de la relación que existe entre A y B. Sin embargo, si observamos el estado de la señal de Préstamo para cada uno de estas casos veremos que:

	BW	$X = A > B$
$A - B > 0 \iff A > B$	0	Inconcluyente
$A - B = 0 \iff A = B$	0	Inconcluyente
$A - B < 0 \iff A < B$	1	0

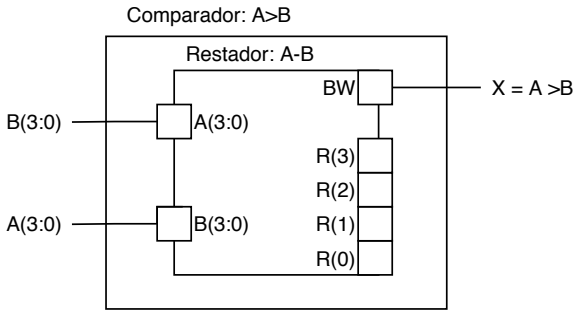
La señal de Préstamo sólo nos resulta útil para determinar cuando $A < B$ pero no para lograr la salida X deseada. Es evidentemente que necesitaremos información adicional.

Solución propuesta por Martín, alumno del curso R3052 dictado el año 2019.

La propuesta de Martín consistió en invertir los operandos del restador. A diferencia de la propuesta anterior, en esta ocasión se tendrá que:

	BW	$X = A > B$
$B - A > 0 \iff B > A$	0	0
$B - A = 0 \iff B = A$	0	0
$B - A < 0 \iff B < A$	1	1

Se puede apreciar que gracias al simple hecho de invertir el orden de los operandos en la resta, se obtiene el resultado deseado de forma simple y efectiva. El circuito resultante será entonces:



Solucion al ejercicio D.16. Comparador completo

Previo a la resolución, veamos como se relacionan las diferentes formas de representación para palabras de dos bits:

Decimal	Binario	CA2	Bin. Desp.
+3	11	–	
+2	10	–	
+1	01	01	11
+0	00	00	10
-1	–	11	01
-2	–	10	00

Parte A:

P(1)	P(0)	Q(1)	Q(0)	$p > q$	$p = q$	$p < q$
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

Cada una de las expresiones de salida se puede obtener sintetizando con el uso de mapas de Karnaugh.

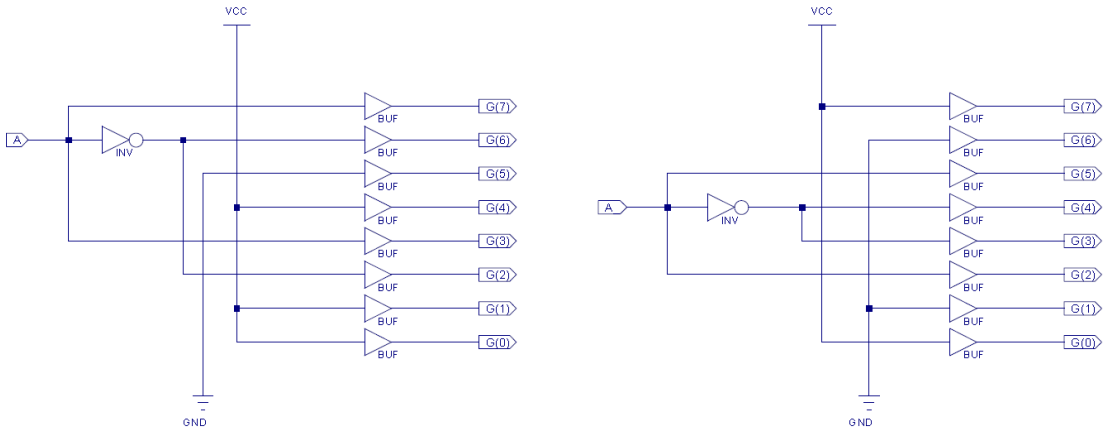
$$p < q = \overline{P(1)}.Q(1) + \overline{P(1)}.\overline{P(0)}.Q(0) + \overline{P(0)}.Q(1).Q(0)$$
$$p = q = \overline{P(1)}.\overline{P(0)}.\overline{Q(1)}.\overline{Q(0)} + \overline{P(1)}.P(0).\overline{Q(1)}.Q(0) + P(1).\overline{P(0)}.Q(1).\overline{Q(0)} + P(1).P(0).Q(1).Q(0)$$
$$p > q = P(1).\overline{Q(1)} + P(1).P(0).\overline{Q(0)} + P(0).\overline{Q(1)}.\overline{Q(0)}$$

Parte B:

Resolución a cargo del lector.

Solucion al ejercicio D.17. Asignación, transferencia y complemento

Se procede con el conexionado de las funciones solicitadas haciendo uso de buffers de salida.

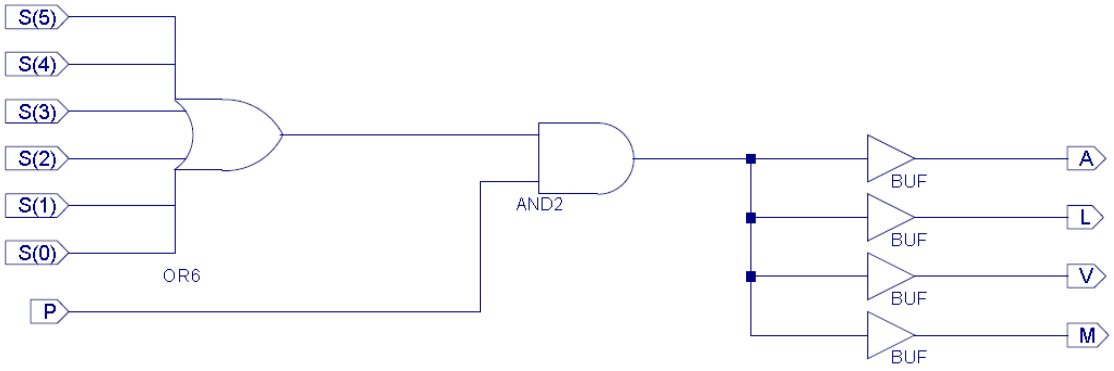


Solucion al ejercicio D.18. Habilitación

La activación de cualquier de sensor S_i debe activar las salidas del sistema de seguridad siempre y cuando la señal **P** también esté activa. Caso contrario, las salidas estarán apagadas. Este comportamiento puede modelarse con la siguiente tabla de funcionamiento:

S5	S4	S3	S2	S1	S0	P	A	L	V	M
1	-	-	-	-	-	1	1	1	1	1
-	1	-	-	-	-	1	1	1	1	1
-	-	1	-	-	-	1	1	1	1	1
-	-	-	1	-	-	1	1	1	1	1
-	-	-	-	1	-	1	1	1	1	1
-	-	-	-	-	1	1	1	1	1	1
-	-	-	-	-	-	0	0	0	0	0

Se observa que la señal **P** actúa como señal de habilitación para las restantes entradas del sistema. El circuito resultante de este análisis es entonces:



Solucion al ejercicio D.19. Decodificación I

Del análisis del circuito se obtienen las siguientes expresiones de salida:

$$O(0) = \overline{A(1)} \cdot \overline{A(0)}$$
$$O(1) = \overline{A(1)} \cdot A(0)$$
$$O(2) = A(1) \cdot \overline{A(0)}$$
$$O(3) = A(1) \cdot A(0)$$

Cuya tabla de verdad es:

A(1)	A(0)	O(3)	O(2)	O(1)	O(0)
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Se aprecia que bajo un valor binario de entrada, se activa únicamente la salida de igual índice. Este circuito se conoce como decodificador.

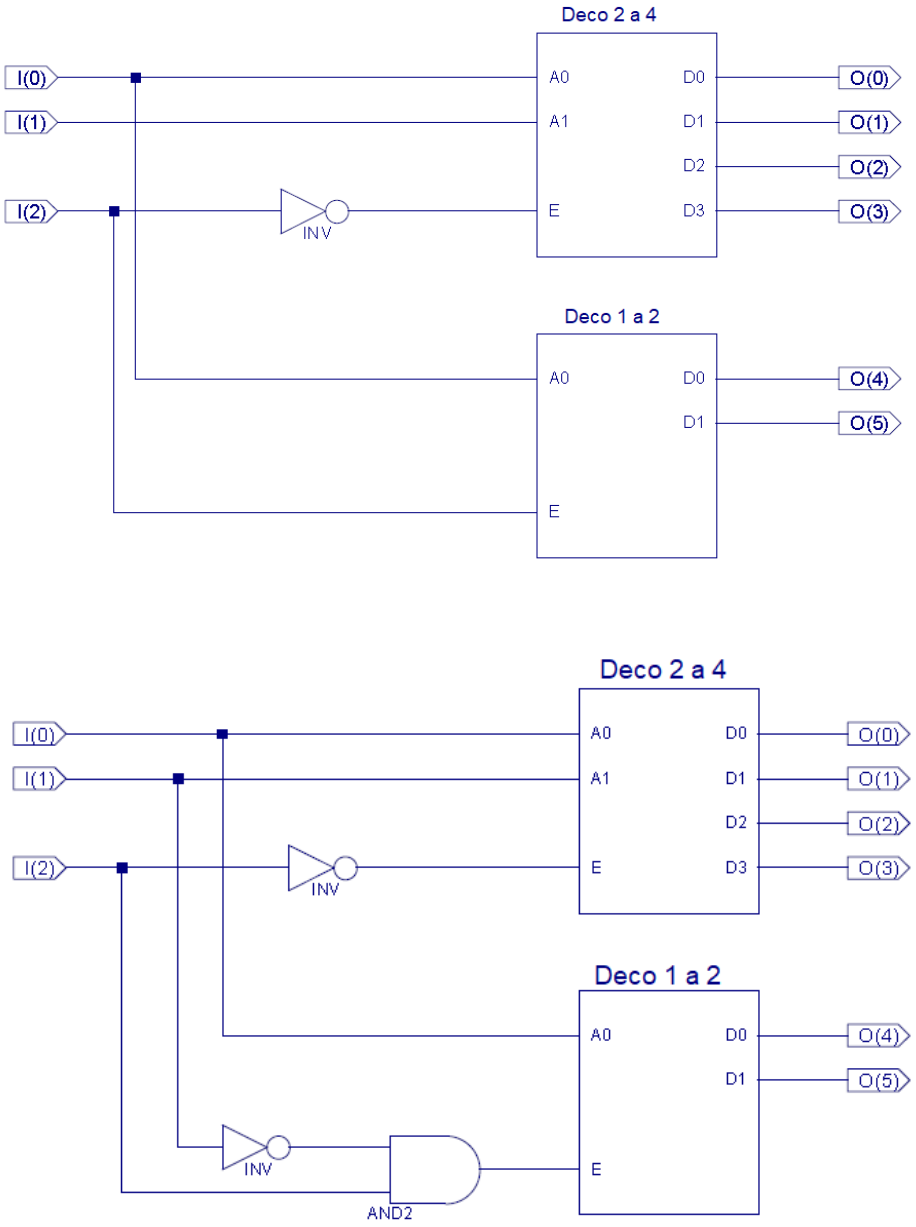
Solucion al ejercicio D.20. Decodificación II

Para realizar el decodificador de 3 a 6 usando únicamente los recursos disponibles, se divide el problema en dos partes.

a) Toda vez que la palabra de entrada I sea menor a 4, deberá activarse el decodificador 2 a 4, y este indicará cual de los números posibles ha sido ingresado. Para lograr esto se deberá conectar las entradas I(1) e I(0) a las entradas de datos del decodificador 2 a 4 (A(1) y A(0)) y la entrada I(2) complementada a la entrada de habilitación del decodificador 2 a 4.

b) Toda vez que la palabra de entrada I sea mayor o igual a 4, deberá activarse el decodificador 1 a 2, y este indicará cual de los números posibles ha sido ingresado. Para lograr esto se deberá conectar la entrada I(0) a la entrada de datos del decodificador 1 a 2 y la entrada I(2) a la entrada de habilitación del decodificador 1 a 2. Nótese que cuando I sea igual a 4_{10} (100_2) se activará la salida O(0) del decodificador 1 a 2 (cuyo nombre real será O(4)), en tanto que si I es igual a 5_{10} (101_2) se activará la salida O(1) del decodificador 1 a 2 (cuyo nombre real será O(5)).

Una posible mejora al circuito anterior se obtiene deshabilitando ambos circuitos en caso que I toma un valor inválido, es decir 6_{10} o 7_{10}. Para ello, basta con evaluar si I(1) esta activada junto con I(2). Dicha implementación será entonces:



Solucion al ejercicio D.21. Codificación

De la tabla de funcionamiento propuesta se derivan las siguientes ecuaciones:

$$\begin{aligned} A(1) &= I(3) + \overline{I(3)} \cdot I(2) \\ &= I(3) + I(2) \\ A(0) &= I(3) + \overline{I(3)} \cdot \overline{I(2)} \cdot I(1) \\ &= I(3) + \overline{I(2)} \cdot I(1) \\ G &= I(3) + I(2) + I(1) + I(0) \end{aligned}$$

La señal de grupo permite discriminar los casos donde el vector de salida $A(1:0) = 00_2$.

Solucion al ejercicio D.22. Selección I

La tabla de verdad del circuito es:

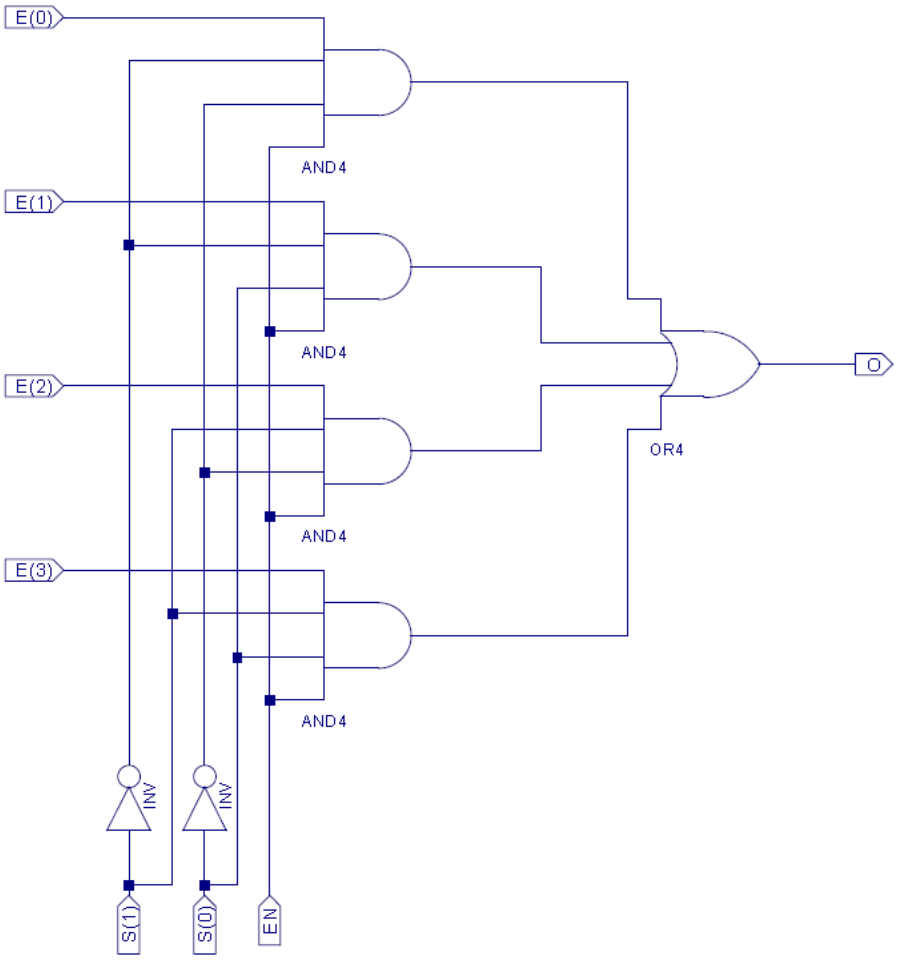
S	I(1)	I(0)	O
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

$$O = \overline{S} \cdot I(0) + S \cdot I(1)$$

Se aprecia que la señal S selecciona si la salida toma el valor de la entrada I(0) o de la entrada I(1), por lo cual es un circuito selector o multiplexor.

Solucion al ejercicio D.23. Selección II

b) Multiplexor con habilitación



Solucion al ejercicio D.24. Selección III

Resolución a cargo del lector.

Solucion al ejercicio D.25. Selección IV

Resolución a cargo del lector.

Solucion al ejercicio D.26. Descomposición funcional - Teorema de Shannon

Resolución a cargo del lector.

Solucion al ejercicio D.27. Descomposición funcional de Ashenhurst

Resolución a cargo del lector.

Solucion al ejercicio D.28. Implementación alternativa de funciones lógicas

Resolución a cargo del lector.

Solucion al ejercicio D.29. Síntesis RTL

Resolución a cargo del lector.

Solucion al ejercicio D.30. Síntesis RTL

Resolución a cargo del lector.

Solucion al ejercicio D.31. Síntesis RTL

Resolución a cargo del lector.

Solucion al ejercicio D.32. Síntesis RTL

Resolución a cargo del lector.

Solucion al ejercicio D.33. Síntesis RTL

Resolución a cargo del lector.

Solucion al ejercicio D.34. Síntesis RTL

Resolución a cargo del lector.

Solucion al ejercicio D.35. Síntesis RTL

Resolución a cargo del lector.

Solucion al ejercicio D.36. Síntesis RTL

Resolución a cargo del lector.

Solucion al ejercicio D.37. Síntesis RTL

Resolución a cargo del lector.

Solucion al ejercicio D.38. Síntesis RTL

Resolución a cargo del lector.

Solucion al ejercicio D.39. Desplazamientos

Resolución a cargo del lector.

Solucion al ejercicio D.40. Desplazamientos

Resolución a cargo del lector.

Solucion al ejercicio D.41. Desplazamientos

Resolución a cargo del lector.

Solucion al ejercicio D.42. Rotación

Resolución a cargo del lector.

Solucion al ejercicio D.43. Rotación

Parte A:

C(1)	C(0)	O(3)	O(2)	O(1)	O(0)
0	0	D(3)	D(2)	D(1)	D(0)
0	1	D(2)	D(1)	D(0)	D(3)
1	0	D(1)	D(0)	D(3)	D(2)
1	1	D(0)	D(3)	D(2)	D(1)

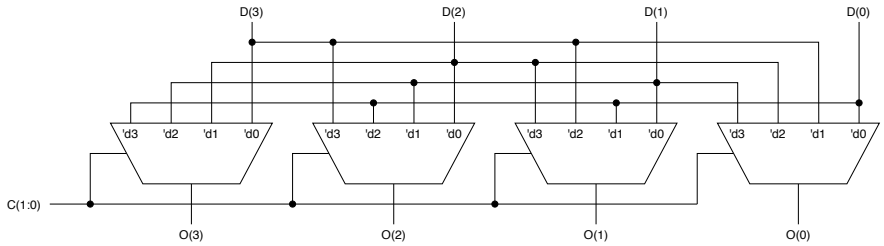
Parte B:

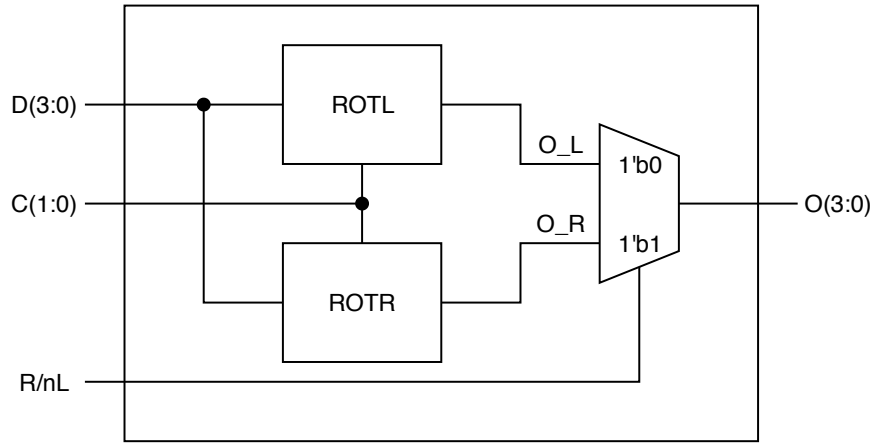
Parte C:

Sea O_L la palabra que resulta de rotar la palabra de entrada D hacia la izquierda, y sea O_R la palabra que resulta de rotar la entrada D hacia la derecha, se tendrá entonces que la salida O de este circuito se puede obtener según la siguiente tabla:

R/nL	$O(3:0)$
0	$O\_L(3:0)$
1	$O\_R(3:0)$

de donde O_L es equivalente a la salida del circuito obtenido en los puntos A y B, y O_R es la salida de un circuito combinacional que puede construirse de forma análoga al anterior.





Solucion al ejercicio D.44. Saturación

Resolución a cargo del lector.

Solucion al ejercicio D.45. Saturación

Resolución a cargo del lector.

Solucion al ejercicio D.46. Paridad

Resolución a cargo del lector.

E. Diseño de circuitos secuenciales sincrónicos

Solucion al ejercicio E.1. Registro SISO

Resolución a cargo del lector.

Solucion al ejercicio E.2. Registro PIPO

Resolución a cargo del lector.

Solucion al ejercicio E.3. Registro PISO

Resolución a cargo del lector.

Solucion al ejercicio E.4. Registro SIPO

Resolución a cargo del lector.

Solucion al ejercicio E.5. Contador universal de longitud arbitraria W

Resolución a cargo del lector.

Solucion al ejercicio E.6. Contador de módulo arbitrario M

Resolución a cargo del lector.

Solucion al ejercicio E.7. Contador signado

Resolución a cargo del lector.

Solucion al ejercicio E.8. Toggle habilitado

Resolución a cargo del lector.

Solucion al ejercicio E.9. Esquemas de expansión

Resolución a cargo del lector.

Solucion al ejercicio E.10. Contador en anillo

Resolución a cargo del lector.

Solucion al ejercicio E.11. Contador Johnson

Resolución a cargo del lector.

Solucion al ejercicio E.12. LFSR - Linear Feedback Shift Register

Resolución a cargo del lector.

F. Máquinas de estados finitos

Solucion al ejercicio F.1. Análisis de una máquina de estados

Resolución a cargo del lector.

Solucion al ejercicio F.2. Análisis de una máquina de estados

Resolución a cargo del lector.

Solucion al ejercicio F.3. Síntesis de una máquina de estados: Decodificador NRZI

Resolución a cargo del lector.

Solucion al ejercicio F.4. Síntesis de una máquina de estados

Resolución a cargo del lector.

Solucion al ejercicio F.5. Análisis de una máquina de estados

Resolución a cargo del lector.

Solucion al ejercicio F.6. Análisis de una máquina de estados

Resolución a cargo del lector.

Solucion al ejercicio F.7. Síntesis de una máquina de estados: Detector de un patrón de bits

Resolución a cargo del lector.

Solucion al ejercicio F.8. Síntesis de una máquina de estados

Resolución a cargo del lector.

Ejercicio F.1. Integrador #1

Resolución a cargo del lector.

Ejercicio F.2. Integrador #2

Resolución a cargo del lector.

Ejercicio F.3. Integrador #3

Resolución a cargo del lector.

G. Análisis temporal estático

Solucion al ejercicio G.1. Clock skew negativo

a) **Análisis de restricciones temporales para** $tskew = -50ps$

- **Hold Check**

$$tcq_L + tnext > tskew + th_C$$
$$200\ ps + 0\ ps > (-50)\ ps + 100\ ps$$
$$200\ ps > 50\ ps$$

Por lo tanto, no hay violación de tiempo de hold.

- **Hold Slack**

$$tcq_L + tnext = tskew + th_C + hold\_slack$$
$$200\ ps + 0\ ps = (-50)\ ps + 100\ ps + hold\_slack$$
$$150\ ps = hold\_slack$$

- **Setup Check**

$$tcq_L + tnext + tsu_C < tskew + tclk$$
$$200\ ps + 0\ ps + 200\ ps < (-50)\ ps + 1000\ ps$$
$$400\ ps < 950\ ps$$

Por lo tanto, no hay violación de tiempo de setup.

- **Setup Slack**

$$setup\_slack + tcq_L + tnext + tsu_C = tskew + tclk$$
$$setup\_slack + 200\ ps + 0\ ps + 200\ ps = (-50)\ ps + 1000\ ps$$
$$setup\_slack = 550\ ps$$

b) **Análisis de restricciones temporales para** $tskew = -100ps$

- **Hold Check**

$$tcq_L + tnext > tskew + th_C$$
$$200\ ps + 0\ ps > (-100)\ ps + 100\ ps$$
$$200\ ps > 0\ ps$$

Por lo tanto, no hay violación de tiempo de hold.

- **Hold Slack**

$$tcq_L + tnext = tskew + th_C + hold\_slack$$
$$200\ ps + 0\ ps = (-100)\ ps + 100\ ps + hold\_slack$$
$$200\ ps = hold\_slack$$

- **Setup Check**

$$tcq_L + tnext + tsu_C < tskew + tclk$$
$$200\ ps + 0\ ps + 200\ ps < (-100)\ ps + 1000\ ps$$
$$400\ ps < 900\ ps$$

Por lo tanto, no hay violación de tiempo de setup.

- **Setup Slack**

$$setup\_slack + tcq_L + tnext + tsu_C = tskew + tclk$$
$$setup\_slack + 200\ ps + 0\ ps + 200\ ps = (-100)\ ps + 1000\ ps$$
$$setup\_slack = 500\ ps$$

c) **¿Para qué valor de** $tskew$ **existen violaciones temporales?**
Primero, se buscará el intervalo de tiempos para el cual no existen violaciones temporales

$$tcq_L + tnext + tsu_C - tclk < tskew$$
$$200\ ps + 0\ ps + 200\ ps - 1000\ ps < tskew$$
$$-600\ ps < tskew$$

$$tcq_L + tnext - th_C > tskew$$
$$200\ ps + 0\ ps - 100\ ps > tskew$$
$$100\ ps > tskew$$
$$\therefore -600\ ps < tskew < 100\ ps$$

Finalmente, habrá violaciones temporales siempre que

$$tskew < -600\ ps \quad \vee \quad tskew > 100\ ps$$

Solucion al ejercicio G.2. Clock skew encadenado

Resolución a cargo del lector.

Solucion al ejercicio G.3. Carreras entre datos y reloj

Resolución a cargo del lector.

Solucion al ejercicio G.4. Caminos R2R interdependientes

Recordemos que para que no haya violación de tiempo de hold se debe cumplir:

$$tcq_L + tnext > tskew + th_C$$

y para que no haya violación de tiempo de setup se debe cumplir:

$$tcq_L + tnext + tsu_C < tskew + tclk$$

Del circuito se observa la existencia de los siguientes caminos R2R

$$\begin{array}{lcl} Path_{01} & \wedge & launching = 0 \quad \wedge \quad capturing = 1 \\ Path_{12} & \wedge & launching = 1 \quad \wedge \quad capturing = 2 \\ Path_{23} & \wedge & launching = 2 \quad \wedge \quad capturing = 3 \\ Path_{30} & \wedge & launching = 3 \quad \wedge \quad capturing = 0 \end{array}$$

Los caminos $Path_{01}$, $Path_{12}$ y $Path23$ son equivalentes, en lo que análisis temporal respecta. Para estos casos tendremos entonces que:

$$\begin{aligned} tcq_0 + tsu_1 &< tskew + tclk \\ 180\ ps + 150\ ps &< tskew + 500\ ps \\ 330\ ps &< tskew + 500\ ps \\ \therefore si\ tskew > 0\ &entonces\ es\ siempre\ verdadera. \end{aligned}$$

$$\begin{aligned} tcq_0 &> tskew + th_1 \\ 180\ ps &< tskew + 75\ ps \\ 105\ ps &< tskew \end{aligned}$$

Por lo tanto, para cumplir con ambas restricciones se requiere que

$$0 < tskew < 105ps$$

Finalmente, para el camino $Path_{30}$, si consideramos que la demora entre dos flip flops cualesquiera es igual a ΔT, tendremos que:

$$tskew = -3\Delta T$$

Así entonces

$$\begin{aligned} tcq_3 + tsu_0 &< tskew + tclk \\ 180\ ps + 150\ ps &< -\Delta T + 500\ ps \\ \Delta T &< \frac{170}{3}\ ps \end{aligned}$$

Por lo tanto, para cumplir con ambas restricciones se requiere que

$$0 < tskew < \frac{170}{3}\ ps$$

Solucion al ejercicio G.5. Máxima frecuencia de operación de un contador en anillo

Verificamos primero que no haya violación de tiempo de hold se debe cumplir:

$$\begin{aligned} tcq_L &> th_C \\ 400\ ps &> 200\ ps \\ \therefore No\ hay\ violacion\ de\ tiempo\ de\ hold. & \end{aligned}$$

Luego para calcular la máxima frecuencia de operación del circuito se tendrá que:

$$\begin{aligned} Tclk_{min} &= tcq + tsu \\ Tclk_{min} &= (400 + 300)\ ps \\ \therefore fclk_{max} &= \frac{1}{Tclk_{min}} = 1,428GHz \end{aligned}$$

Solucion al ejercicio G.6. Máxima frecuencia de operación de un contador binario

En la siguiente tabla se exhiben las demoras combinacionales para cada camino R22. Considérese que la intersección de la fila *i-ésima* con la columna *j-ésima* representa la demora combinacional en el camino R2R entre el regitro i y el registro j

	R0	R1	R2
R0	100 ps	500 ps	1100 ps
R1	0 ps	500 ps	1100 ps
R2	0 ps	0 ps	500 ps

Si $fclk = 500MHz$, entonces $tclk = 2ns$, luego para que no ocurra violaciones de tiempo de setup se debe cumplir que

$$\begin{aligned} tclk = 2ns &\geq tcq + tnext_{max} + tsu \\ 2ns &\geq (275 + 1100 + 250)\ ps \\ 2ns &\geq 1625\ ps = 1,625\ ns \end{aligned}$$

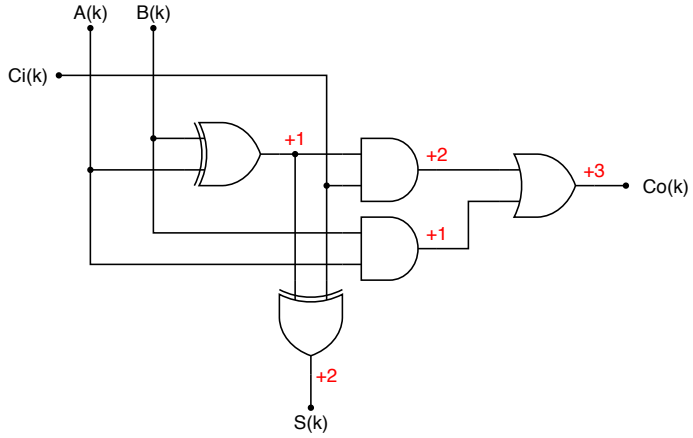
Por lo tanto, no habrá violación de tiempo de setup. Finalmente, como $tcq > th$ podemos afirmar que tampoco habrá violaciónd de tiempo de hold, y por lo tanto, se cumplen todas las restricciones temporales.

Solucion al ejercicio G.7. Integrador

Consideremos la siguiente implementación para el sumador RCA

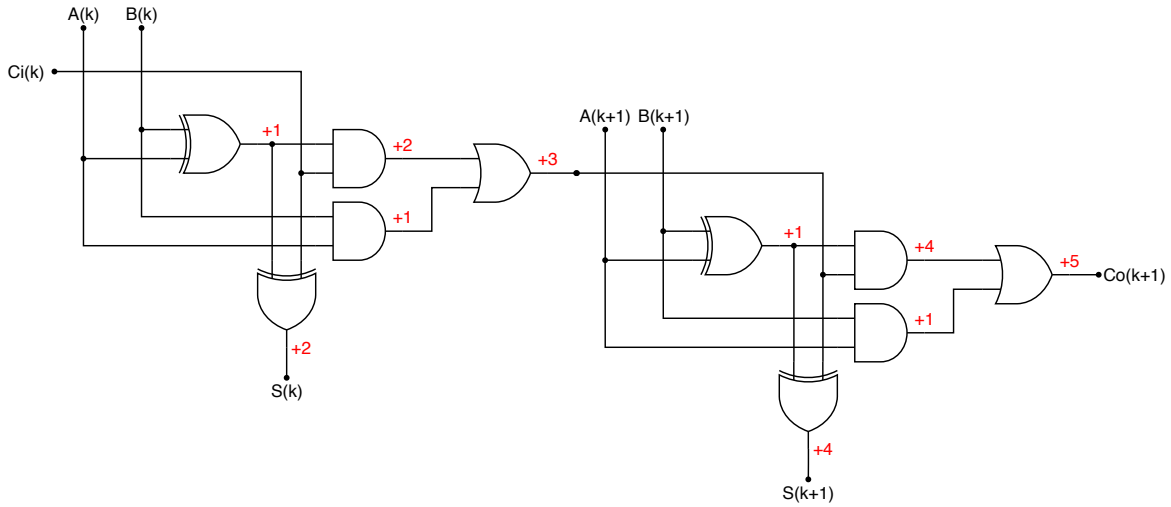
$$S(k) = Ci(k) + B(k) + A(k)$$
$$Co(k) = B(k)A(k) + Ci(k)(B(k) \oplus A(k))$$

Cuya implementación a nivel de compuertas es



Para el siguiente análisis se considerará que el tiempo de propagación de las compuertas es unitario y el mismo para todas las compuertas. Así pues, al considerar tan sólo una celda, se observa que el tiempo de propagación requerido para que las señales $Co(k) y S(k)$ se estabilicen es igual igual a $3UT$ y $2UT$ (unidades de tiempo).

Sin embargo, al considerar dos celdas consecutivas, se observa, que hay nodos intermedios en la celda $k+1$ que estabilizan previo a la señal de propagación de $Co(k)$ y gracias a ello, para la segunda celda en cuestión, el tiempo final de propagación es menor al obtenido previamente.



Bajo estas condiciones se puede afirmar que para un sumador de 4 bits, el retardo será igual a

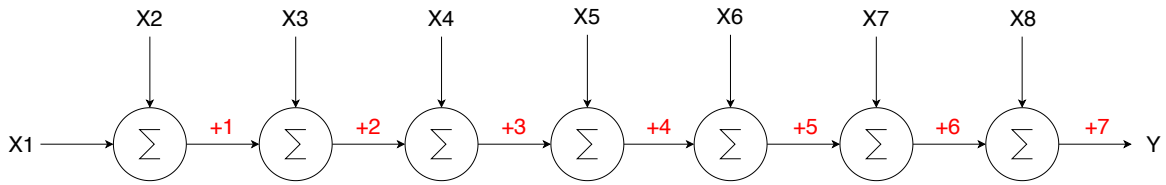
$$3\ UT + 2\ UT \cdot 3 = 9\ UT$$

y en general, para un sumador de N bits de estas características se tendrá

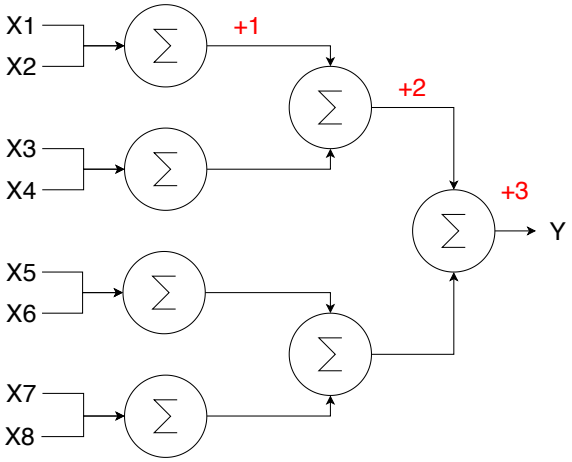
$$3\ UT + 2\ UT \cdot (N-1)$$

Solucion al ejercicio G.8. Integrador

Considérese un sumador cuyo tiempo total de propagación es equivalente a 1 UT (unidad de tiempo) y cuya implementación insume un área equivalente a 1 UA (unidad de área). Bajo estas condiciones, una implementación en cascada insumiríaun total de 7 UT y 7 UA, tal y como lo muestra la siguiente figura.



Al considerar una estructura en árbol, observamos que el total de área necesaria para la implementación es la misma que la anterior, es decir, 7 UA, ya que en ambas implementaciones se requieren hacer 7 sumas en total. Sin embargo, se puede apreciar que con este nueva estructura el tiempo necesario para completar la operación es igual a 3 UT. Es evidente que un simple cambio en la arquitectura del circuito redujo en más de la mitad el tiempo requerido para la operación



En general, para sumar N operandos, siendo N una potencia de 2, se tendrá que el tiempo requerido en la estructura de árbol es igual a $log_2(N)$.

Solucion al ejercicio G.9. Integrador

Resolución a cargo del lector.

H. HDLs

- Las soluciones a los problemas de este capítulo se encuentran a disposición de los alumnos en los repositorios de la facultad destinados a tal fin.
- Los alumnos cuentan con el acceso a un banco de pruebas (*test-bench*), escrito en código VHDL, que le permite simular y verificar el correcto funcionamiento de cada uno de los diseños solicitados en los ejercicios de este capítulo.

Intencionalmente en blanco.

I. Libros recomendados

- *Título:* **Diseno digital: principios y practicas**.
  - *Autor:* John F. Wakerly.
  - *Disponibilidad en Medrano:* 2 (dos) copias.
  - *Disponibilidad en Campus:* 2 (dos) copias.
  - *Disponibilidad en Dpto. de Electrónica:* 1 (una) copias.
- *Título:* **Fundamentos de diseño lógico y computadoras**.
  - *Autor:* M. Morris Mano y Charles R. Kime.
  - *Disponibilidad en Medrano:* 4 (cuatro) copias.
  - *Disponibilidad en Campus:* 0 (cero) copias.
  - *Disponibilidad en Dpto. de Electrónica:* 0 (cero) copias.
- *Título:* **Fundamentos de lógica digital con diseño VHDL**.
  - *Autor:* Stephen Brown y Zvonko Vranesic.
  - *Disponibilidad en Medrano:* 2 (dos) copias.
  - *Disponibilidad en Campus:* 1 (una) copias.
  - *Disponibilidad en Dpto. de Electrónica:* 0 (cero) copias.
- *Título:* **RTL Hardware Design using VHDL**.
  - *Autor:* Pong P. Chu.
  - *Disponibilidad en Medrano:* 0 (cero) copias.
  - *Disponibilidad en Campus:* 0 (cero) copias.
  - *Disponibilidad en Dpto. de Electrónica:* 1 (una) copias.
- *Título:* **Digital Logic and Microprocessor Design with VHDL**.
  - *Autor:* Enoch O. Hwang.
  - *Disponibilidad en Medrano:* 0 (cero) copias.
  - *Disponibilidad en Campus:* 0 (cero) copias.
  - *Disponibilidad en Dpto. de Electrónica:* 0 (cero) copias.

Nota: La información acerca de la disponibilidad de los libros en la facultad fue extraída del siguiente enlace **Biblioteca UTN FRBA**, el día 24 de marzo de 2019. El mismo se encuentra a disposición de todos los alumnos de la facultad para ser consultado cuando lo deseen. El orden exhibido en los libros está sujeto exclusivamente a la disponibilidad de copias en las bibliotecas de la UTN FRBA.

Referencias

[1] Morris Mano y Charles Kime. *Fundamentos de diseño lógico y de computadoras*. Pearson - Prentice Hall, 3ra edición edition, 2005. Optimización de circuitos multinivel, Capítulo 2 Sección 6.

[2] Enoch O. Hwang. *Digital Logic and Microprocessor Design With VHDL*. 2005. Timing Hazards and Glitches, Capítulo 3 Sección 5.

[3] Stephen Brown y Zvonko Vranesic. *Fundamentos de Lógica Digital con diseño VHDL*. Mc Graw Hill, 2da edición edition, 2006. Riesgos, Capítulo 9 Sección 6.