Versión 2025.01

Carrera: INGENIERIA EN INFORMATICA

Asignatura: 3631-Fundamentos de sistemas embebidos.

Tema: Álgebra de Boole y circuitos combinatorios

Unidad: 2.0 y 2.1

Objetivo: Comprender el diseño de circuitos combinatorios, lógica de dos niveles, simplificaciones y circuitos típicos de una ALU.

Competencia/s a desarrollar:

- Concepción, diseño y desarrollo de proyectos de ingeniería en sistemas de información / informática.
- Gestión, planificación, ejecución y control de proyectos de ingeniería en sistemas de información / informática.
- Utilización de técnicas y herramientas de aplicación en la ingeniería en sistemas de información / informática.
- Generación de desarrollos tecnológicos y/o innovaciones tecnológicas.
- Desarrollo de una actitud profesional emprendedora.
- Aprendizaje continuo.
- Actuación profesional ética y responsable.
- Comunicación efectiva.
- Desempeño en equipos de trabajo.
- Identificación, formulación y resolución de problemas de ingeniería en sistemas de información/informática.

Descripción de la actividad:

1-Tiempo estimado de resolución: 2 semana
2-Metodología: Escrito, utilizando logisim-evolution para verificar
3-Forma de entrega: No obligatoria
4:Metodología de corrección y feedback al alumno: Presencial y por Miel.

D- Lógica de dos niveles

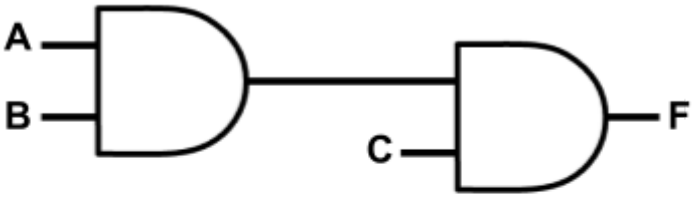
D.01 *Utilizando los postulados del álgebra de boole simplifique las siguientes funciones.*

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

Verifique que las simplificaciones sean válidas comparando la tabla de verdad de la función original contra la tabla de verdad de la función simplificada.

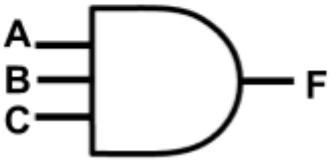
D.02 *El producto lógico (AND) es asociativo, es decir*

$$ABC=(AB)C=A(BC)$$



Esto permite , por ejemplo, implementar una compuerta AND de 3 entradas utilizando dos compuertas AND de 2 entradas cada una (aumentando el tiempo de propagación).

La operación ***NAND NO es asociativa****. Eso quiere decir que*



$$\overline{ABC}\neq\overline{\overline{AB}\,C}\neq\overline{A\,\overline{BC}}$$

Por ende, ***NO*** *se puede hacer (A NAND B) NAND C y esperar que resulte A NAND B NAND C. Escriba una tabla de verdad con todos los valores posibles para A,B y C. Verifique la desigualdad planteada arriba encontrando los valores para*

$$\overline{ABC}\ ;\ \overline{\overline{AB}\,C}\ ;\ \overline{A\,\overline{BC}}$$

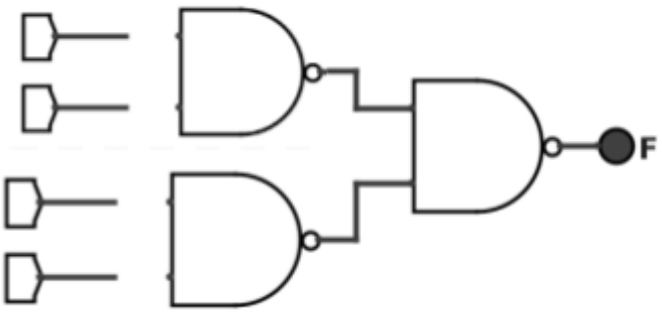
Encuentre una forma de implementar la operación NAND de 3 entradas utilizando ***SOLAMENTE*** *compuertas NAND de 2 entradas.*
Tenga en cuenta la siguiente ***igualdad***

$$\overline{ABC}=\overline{\overline{AB}\,C}$$

D.03 *Dada la siguiente tabla de verdad, simplifique a su mínima expresión la función F utilizando el mapa de Karnaugh (como suma de productos). Luego realice la implementación de la misma utilizando un único tipo de compuertas (NAND). Complete los nombres de las variables correspondientes en los túneles , agregando los negadores necesarios.*

A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	X
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

AB\CD	00	01	11	10
00				
01				
11				
10				



D.04 *Utilizando la función F del punto D.03, simplifique a su mínima expresión la función F con el mapa de Karnaugh (como producto de sumas). Luego realice la implementación con un único tipo de compuertas (NOR).*

D.05 *Utilizando la función F del punto D.03, implemente la función utilizando un multiplexor de 4 entradas de selección (16 entradas de datos). Verifique que en todos los casos (D.03, D.04 y D.05) el resultado que sale por F es el mismo para todos los valores posibles de A,B,C y D.*

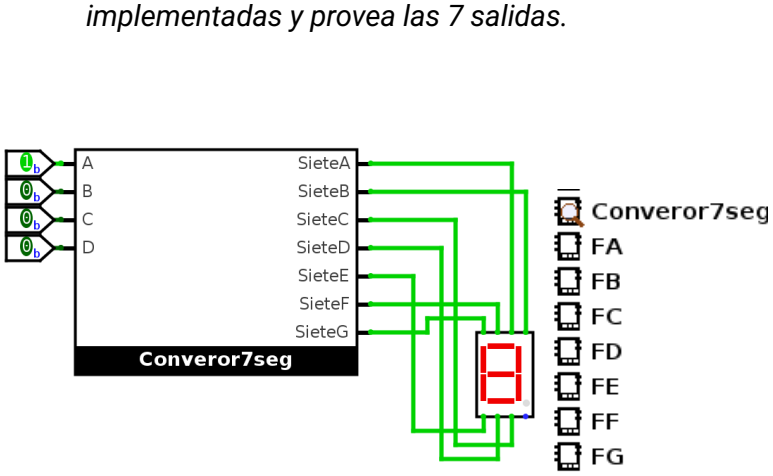
D.06 *Escriba las siguientes funciones en forma canónica.*

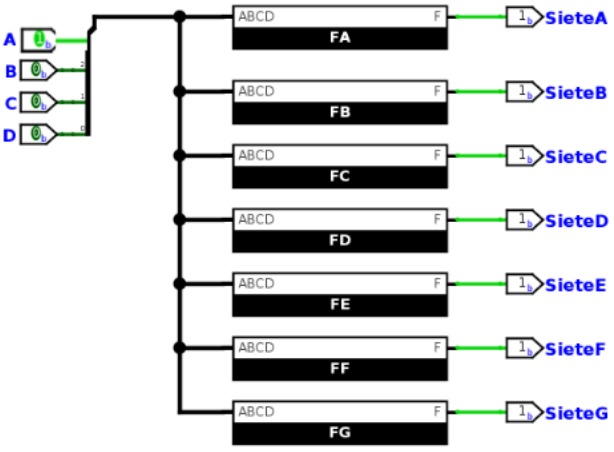
$$F(A,B,C,D)=\overline{A}\,B+A\,B\,\overline{D}$$
$$F(A,B,C,D)=B\,\overline{D}+\overline{A}\,B\,D$$

D.07 *Dadas 4 variables de entrada (A,B,C,D) , utilizando logisim-evolution, haga la implementación de las 7 funciones de salida (FA,FB,FC,FD,FE,FF,FG). En el caso de FF y FG utilice producto de sumas (**Maxitérminos**), en el caso de FE implemente utilizando multiplexor (**MUX4**), mientras que en el resto utilice sumas de productos (**minitérminos**). Debe simplificar por Karnaugh a la menor expresión sin términos redundantes. Implemente utilizando un único tipo de compuerta (**NAND** o **NOR** según el caso o MUX4 en el caso de FE).*

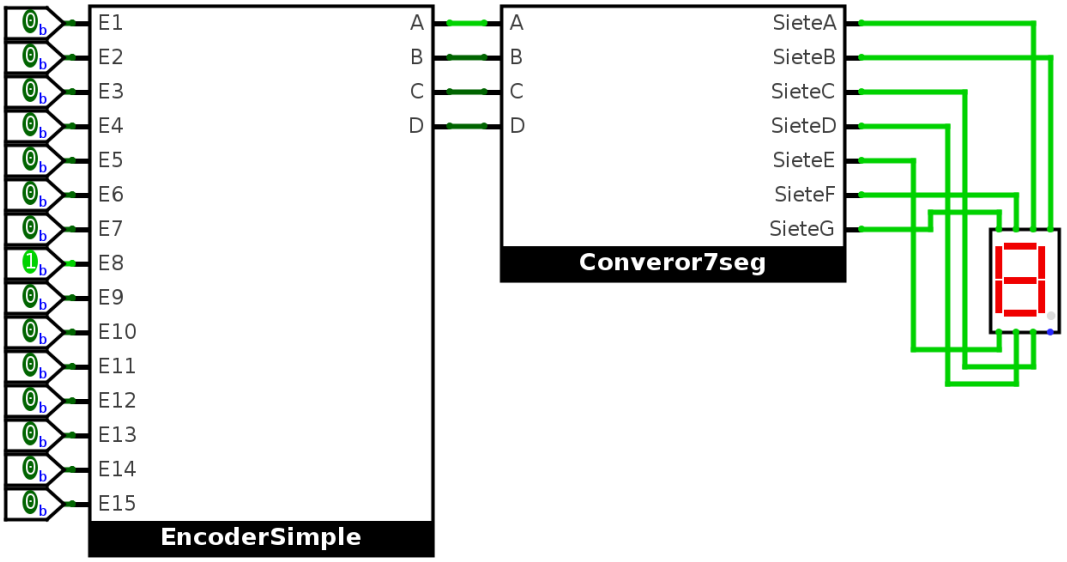
Entradas				Salidas						
A	B	C	D	FA	FB	FC	FD	FE	FF	FG
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	1	1	1	0	1	1	1
1	0	1	1	0	0	1	1	1	1	1
1	1	0	0	1	0	0	1	1	1	0
1	1	0	1	0	1	1	1	1	0	1
1	1	1	0	1	0	0	1	1	1	1
1	1	1	1	1	0	0	0	1	1	1

D.08 *Utilizando logisim-evolution, cree un circuito por cada función del punto anterior. Luego cree un nuevo circuito llamado Conversor7Seg que incluya cada una de las funciones implementadas y provea las 7 salidas.*

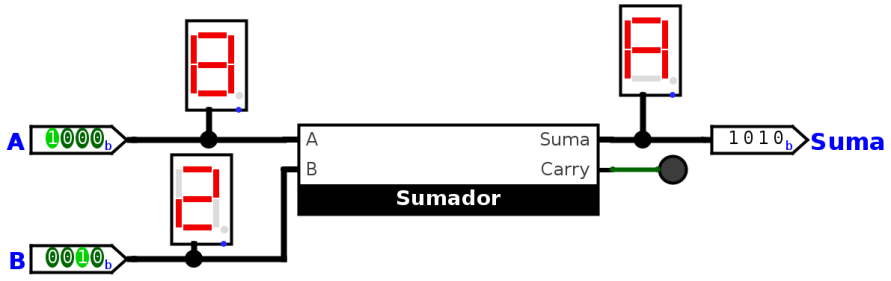




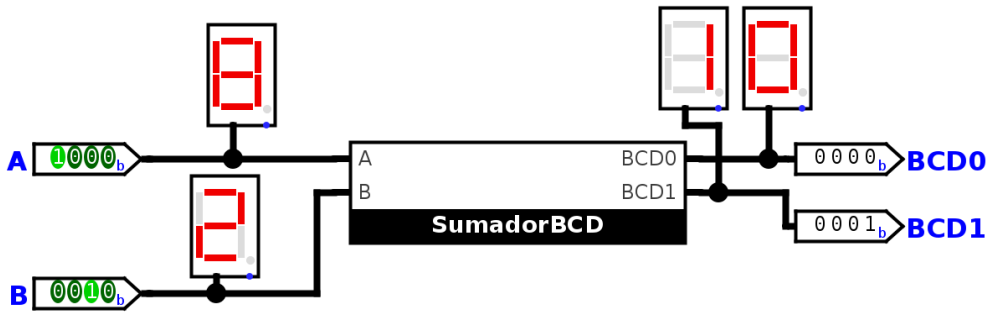
D.09 *Implemente un nuevo circuito llamado EncoderSimple. El mismo se comporta como un codificador simple con 15 entradas (15 pulsadores). Posee 4 salidas que identifican cuál de los pulsadores fue presionado. Ejemplo: Si se presiona el pulsador 3 (y el resto en 0) entonces la salida será 0011. Si se presiona el 9 (y el resto en 0) entonces la salida será 1001. Conecte esta salida al circuito anterior de forma que en el 7 segmentos se represente que botón está apretado (de 1 a F).*



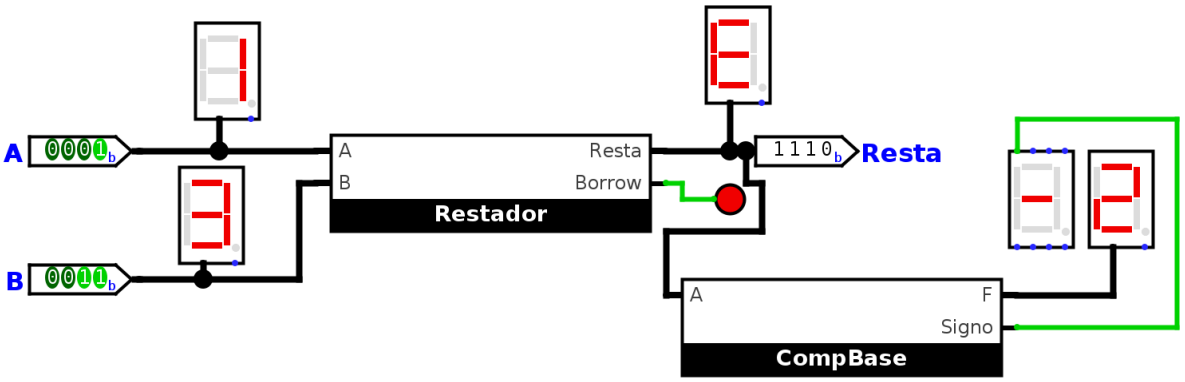
D.10 *Implementar en logisim-evolution un sumador de números de dos números de 4 bits. Como entradas tiene los números A (A3 A2 A1 A0) y B (B3 B2 B1 B0). La salida es un número de 4 bits llamado Suma más un bit de carry.*



D.11 *Modifique el sumador del ejercicio anterior sabiendo que los números de entrada (A y B) son dígitos BCD. El sumador debe tener como salida dos dígitos BCD (representando desde 0000 0000 hasta 0001 1000). Sugerimos detectar aquellos casos donde la suma excede el rango BCD (0000 ~ 1001) y corregir sumando 6. Utilice displays de 7 segmentos para representar los dígitos.*

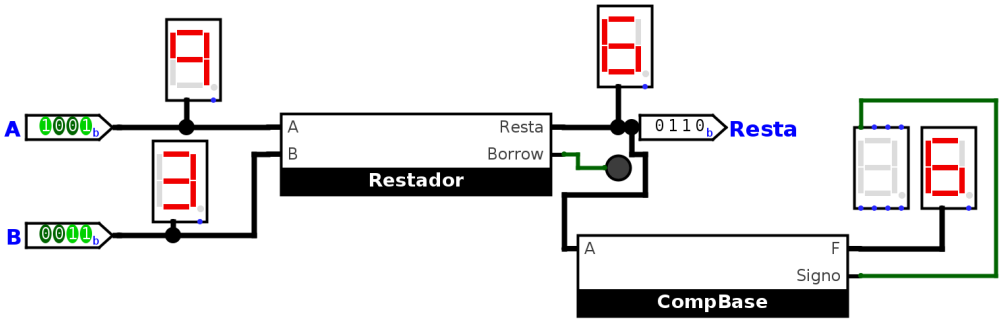


D.12 *Implementar en logisim-evolution un circuito restador de dos números de 4 bits. El mismo debe tener salida de Borrow. Compruebe el correcto funcionamiento verificando que si se resta 0001 - 0011 = 1 1110 (Note que los negativos se representan SIEMPRE en complemento a la base).*



D.13 *Implementar en logisim-evolution un circuito que tenga como entrada un número de 4 bits y calcule como salida el complemento a la base del mismo. Ej: si ingresa 0001 (1) debe salir 1111 (-1). Si ingresa 1110 (-2) debe salir 0010 (2). Razone sobre lo que ocurre cuando ingresa el número 1000 (-8).*

D.14 *Modifique el circuito del punto D.13 para que el circuito solo haga el complemento a la base si el número que ingresa (A) es negativo. En el caso de que el número que ingresa sea 0 o positivo, debe salir el mismo número. Incluya una salida llamada signo que indique con un uno cuando el número que ingresa es negativo, y cero en caso de ser positivo o cero. En la imagen del punto D.12 puede ver el circuito CompBase que cumple con esta función.*



D.15 *Dados dos números de 4 bits (A3A2A1A0 y B3B2B1B0), realice un circuito que tenga 6 leds con la siguiente leyenda:*

- LED A=B se enciende si A=B
- LED A<B se enciende si A<B
- LED A>=B se enciende si A>=B
- LED A > B se enciende si A > B
- LED A <= B se enciende si A <= B
- LED A != B se enciende si A != B

Recuerde sistemas de numeración. Luego de hacer A-B se sabe que:

- A=B si A-B= 0
- A!=B si A-B != 0
- A < B si SignoResultado XOR Overflow = 1
- A >=B si SignoResultado XNOR Overflow = 1
- Overflow en la resta cuando el signo del minuendo y el sustraendo es distinto y el resultado tiene el signo opuesto al minuendo.

D.16 *Implemente un circuito en logisim-evolution llamado DesplazaDerecha. El mismo posee una entrada de 4 bits llamada Dato y otra entrada de 2 bits llamada Cantidad. La salida del circuito son 4 bits llamados Salida. El mismo se debe comportar de la siguiente forma:*

Dato	Cantidad	Salida
abcd	**00**	**abcd**
abcd	**01**	**0abc**
abcd	**10**	**00ab**
abcd	**11**	**000a**

Nota: El circuito se resuelve con un multiplexor de 4 entradas de datos (2 de selección) donde cada entrada de datos es de 4 bits. Utilice Separadores para armar los valores (0,a,b,c);(0,0,a,b);(0,0,0,a) y utilice el MUX para seleccionarlos.

D.17 *Implemente un circuito en logisim-evolution llamado DesplazaDerechaAritmetico. El mismo es similar al circuito D.16 pero con la siguiente tabla. Note que el valor de la entrada 'a' se repite.*

Dato	Cantidad	Salida
abcd	**00**	**abcd**
abcd	**01**	**aabc**
abcd	**10**	**aaab**
abcd	**11**	**aaaa**

D.18 *Implemente un circuito en logisim-evolution llamado DesplazaIzquierda. El mismo es similar a los dos anteriores pero con la siguiente tabla:*

Dato	Cantidad	Salida
abcd	**00**	**abcd**
abcd	**01**	**bcd0**
abcd	**10**	**cd00**
abcd	**11**	**d000**

D.19 *Implemente en logisim-evolution un circuito llamado ALU. El mismo tiene como entrada un número A de 4 bits, un número B de 4 bits, y una entrada de 3 bits llamada Selección. La salida de la ALU es un número de 4 bits llamado Salida , los 6 leds del punto D.12, un led de Carry y un led de Borrow.*

El valor de salida se define dependiendo de los valores de selección:

- 000 → Salida = A - B
- 001 → Salida = A + B
- 010 → Salida = A en complemento a la base
- 011 → Salida = B en complemento a la base
- 100 → Salida = A desplazado derecha (usando B1yB0 como desplazamiento)
- 101 → Salida = A desplazado aritmético (usando B1yB0 como desplazamiento)
- 110 → Salida = A desplazado izquierda (usando B1yB0 como desplazamiento)
- 111 → Salida = A AND B (bitwise A4 AND B4, A3 AND B3, etc).

D.19 *Dados dos números, A (A1,A0) y B (B1, B0) de dos bits, se sabe que A representa números sin signo mientras que B representa números signados en complemento a la base. Escriba la tabla de verdad de un sumador A+B. Tenga en cuenta que A puede valer 0,1,2 o 3, mientras que B puede valer -2, -1,0,1. Elija la cantidad de bits de resultado acorde considerando que tiene que soportar sumas como 0 + (-2) , 3 + 1, etc. Implemente el circuito de la forma que crea más conveniente para cada bit del resultado (sumas de productos o producto de sumas) utilizando siempre un único tipo de compuertas. El circuito debe estar implementado con lógica de dos niveles (no encadenado sumadores).*

D.20 *Dados los números del punto anterior, implemente un circuito multiplicador de AxB. Tenga en cuenta que debe soportar resultados como 3 x -2 por ende elija la cantidad de bits de resultado acorde a estos valores. Comience por la tabla de verdad y luego haga la implementación en logisim-evolution.*