DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
ESCUELA DE INGENIERÍA
PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

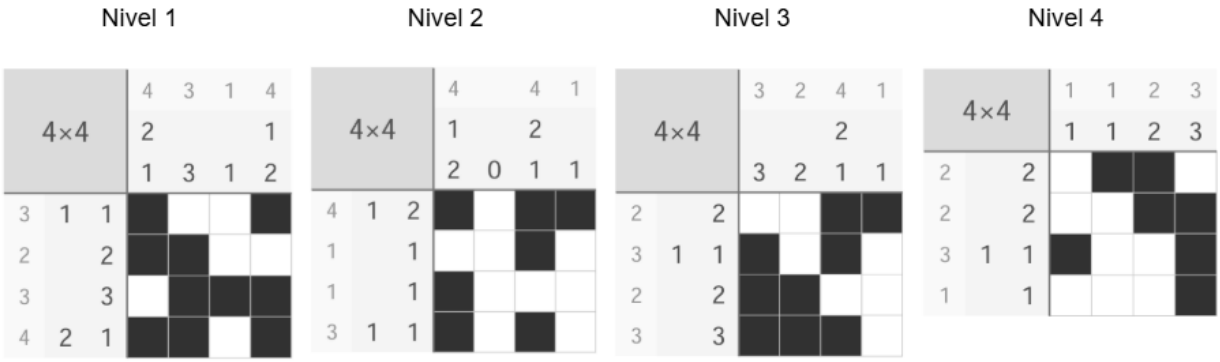
IIC2343 - Arquitectura de Computadores (I/2026)

Ayudantía 2

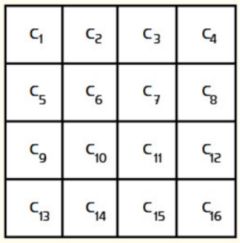
Ayudantes: Alberto Maturana (alberto.maturana@uc.cl), Mario Rojas (mario.denzel@estudiante.uc.cl) , Fernanda Escobar (fer_jez2002uc.cl)

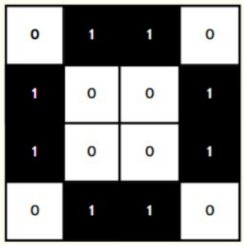
Pregunta 1: Diseñar Circuito

Navegando en la *Playstore* descubres el juego *Nonogram*, un juego de lógica donde debes descubrir una imagen oculta rellenando celdas en una cuadrícula usando números como pistas.
Luego de jugar un rato tienes la brillante idea de crear tus propios niveles para poner a prueba a tus amigos así que diseñas 4 niveles con su respectiva solución:



Han pasado un par de horas y tus amigos ya resolvieron tus niveles y se mueren de ganas de saber si lo hicieron correctamente, pero tu, con tus conocimientos del curso de Arquitectura de computadores en lugar de solo decirles la respuesta decides armar un circuito utilizando compuertas lógicas básicas (AND, OR, NOT y XOR) que les permita ingresar el nivel y obtener la solución correspondiente.
Para diseñar el circuito decides la siguiente distribución de celdas, donde usarás un 1 para representar una casilla coloreada y un 0 para una no coloreada:





Solución: Primero, escribimos la tabla de verdad. Como tenemos cuatro posibles niveles, aplicamos logaritmo en base 2 para saber cuántos bits necesitamos para representarlos.

$$log_2(4) = 2$$

De esta manera, usaremos $00b$, $01b$, $10b$ y $11b$ para representar los niveles 1, 2, 3 y 4 respectivamente.

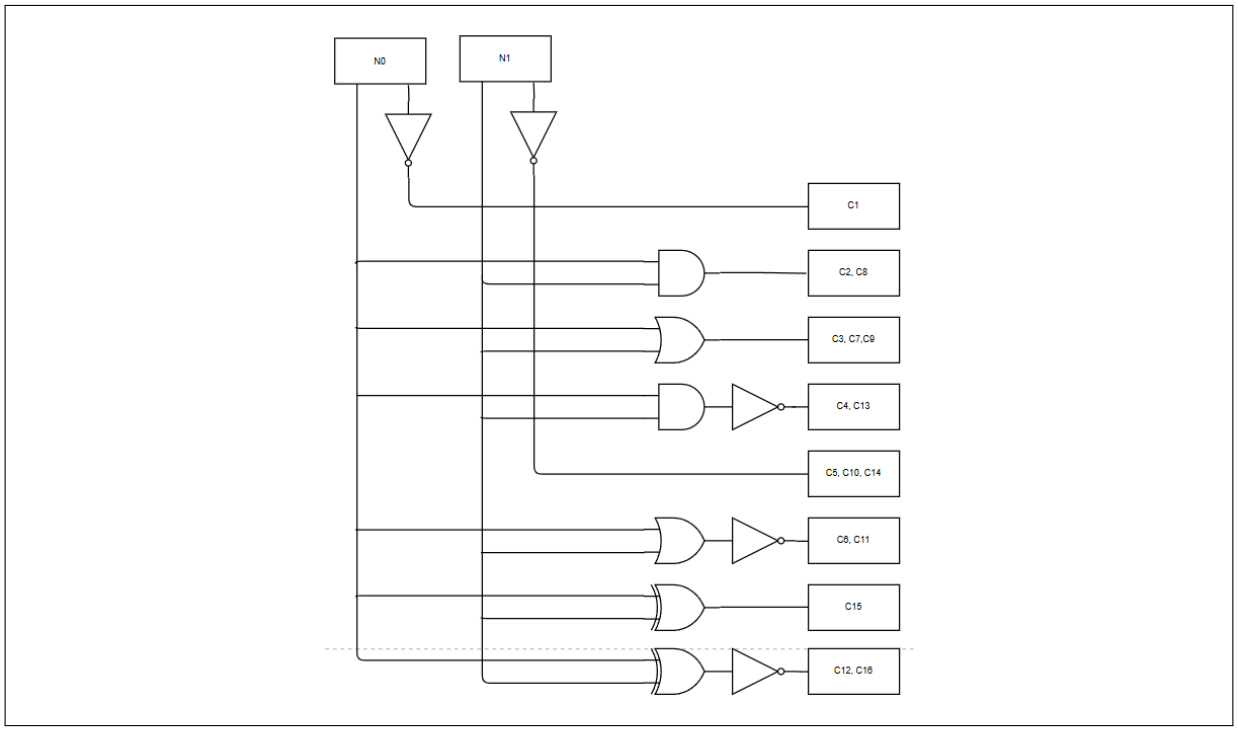
N_0	N_1	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}
0	0	1	0	0	1	1	1	0	0	0	1	1	1
0	1	1	0	1	1	0	0	1	0	1	0	0	0
1	0	0	0	1	1	1	0	1	0	1	1	0	0
1	1	0	1	1	0	0	0	1	1	1	0	0	1

C_{13}	C_{14}	C_{15}	C_{16}
1	1	0	1
1	0	1	0
1	1	1	0
0	0	0	1

Luego, a partir de la tabla, usamos compuertas lógicas para representar cada casilla.

$$\begin{aligned}
C_1 &= \neg N_0 \\
C_2 &= N_0 \wedge N_1 = C_8 \\
C_3 &= N_0 \vee N_1 = C_7, C_9 \\
C_4 &= \neg(N_0 \wedge N_1) = C_{13} \\
C_5 &= \neg N_1 = C_{10} = C_{14} \\
C_6 &= \neg(N_0 \vee N_1) = C_{11} \\
C_{15} &= N_0 \oplus N_1 \\
C_{12} &= \neg(N_0 \oplus N_1) = C_{16}
\end{aligned}$$

Recordar que el símbolo $\oplus$ representa la compuerta XOR.
Finalmente, diseñamos un circuito a partir de estas expresiones.



Pregunta 2: Obtener tabla de verdad a partir de un circuito

Obtenga la tabla de verdad que represente cada uno de los siguientes circuitos ¿Qué se puede concluir a partir de los resultados?

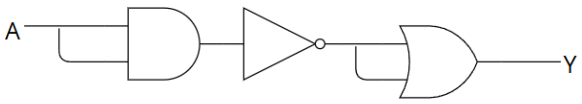


Figura 1

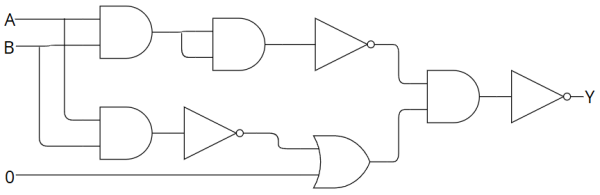


Figura 2

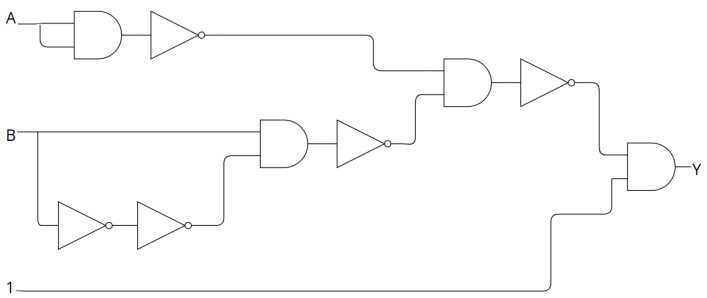
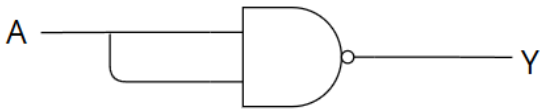


Figura 3

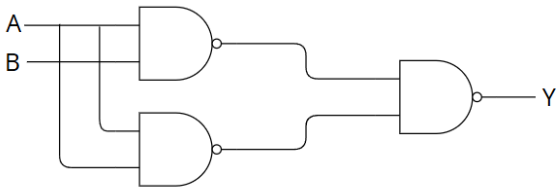
Solución: Para poder resolver este ejercicio, es preciso recordar algunas reglas de álgebra booleana:

- **IDENTIDAD:**
  A AND 1 = A A OR 0 = A
- **IDEMPOTENCIA:**
  A AND A = A A OR A = A
- **INVOLUCIÓN:**
  NOT(NOT A) = A

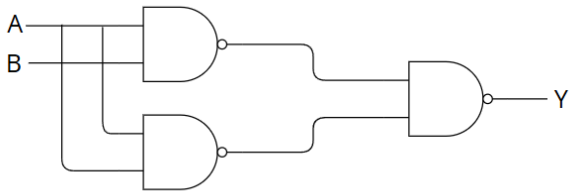
En primer lugar, lo que hacemos es reducir los circuitos a circuitos más simples.
Para el caso de la Figura 1, podemos transformar la compuerta AND y NOT en la compuerta NAND.
Así mismo, notamos que cuando ambas entradas de la compuerta OR son iguales su salida es igual a la entrada, por lo tanto podemos eliminar dicha compuerta.
De esta manera la versión reducida del circuito es la siguiente:



Para el caso de la Figura 2, notamos que al igual que la compuerta OR, cuando ambas entradas de la compuerta AND son iguales su salida es igual a la entrada, por ende la podemos eliminar.
De igual forma, notamos que si una de las entradas de una compuerta OR es 0, la salida será igual al valor de la otra entrada, razón por la cual podemos eliminarla también.
Finalmente, transformamos las compuertas AND y NOT en compuertas NAND obteniendo el siguiente circuito:



Por último, para el caso de la Figura 3, notamos que por la propiedad de involución podemos eliminar las dos compuertas NOT seguidas.
Luego, notamos que si una de las entradas de la compuerta AND es 1, su salida será igual al valor de la otra entrada, por ende podemos eliminarla.
Y para concluir, nuevamente transformamos las compuertas AND y NOT en compuertas NAND.



Para construir la tabla de verdad de cada circuito deberemos evaluar con todas las combinaciones posibles de entradas. Para lograr esto, nos será de gran ayuda la tabla de verdad de la compuerta NAND.

A	B	$NAND$
0	0	1
0	1	1
1	0	1
1	1	0

De esta manera las tablas de verdad son:

Tabla 1: Figura 1

A	Y
0	1
1	0

Tabla 2: Figura 2

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Tabla 3: Figura 3

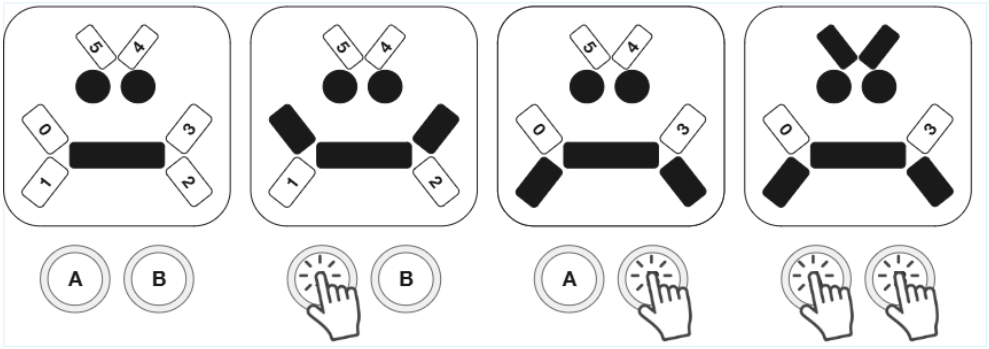
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Podemos observar que las tablas de verdad de los circuitos A, B y C corresponden a las tablas de las compuertas lógicas básicas NOT, AND y OR, respectivamente. A partir de esto, se

puede concluir que la compuerta NAND es universal, lo que significa que cualquier circuito puede ser implementado utilizando solo este tipo de compuerta, dado que cualquier circuito puede expresarse en términos de las compuertas NOT, AND y OR.

Pregunta 3: Circuitos Digitales (I1-2025-2)

Su docente, con el fin de comunicar sus emociones sin interrumpir la clase, decide usar un *display* de 6 segmentos para representar su estado de ánimo en torno a la participación del curso:



Como se observa en la figura, este funciona a partir de dos botones A y B: Si presiona solo el botón A, el *display* mostrará que el docente **está contento**; si presiona solo el botón B, el *display* mostrará que el docente **está triste**; y si presiona A y B de forma simultánea, esto señalará que el docente **está molesto**.

A partir de este contexto diseñe, para cada segmento S_i del *display*, un circuito con dos señales de entrada A y B de 1 bit, y una señal de salida de 1 bit. Las señales de entrada indican si los botones A y B están presionados (1) o no (0), mientras que la salida indica si el segmento S_i se prende (1) o no (0) según los botones presionados a partir de lo ilustrado en la figura. Puede, si lo desea, diseñar un único circuito para computar la salida de todos los segmentos simultáneamente.

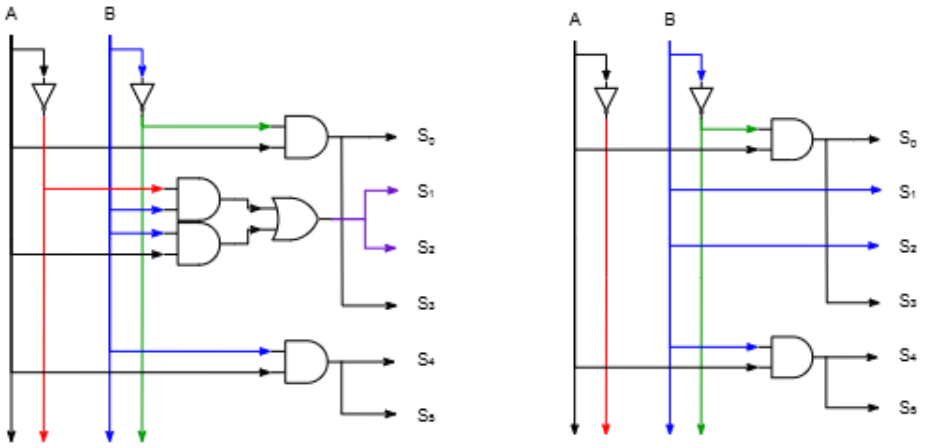
Solución: A continuación, la tabla de verdad para los segmentos S_i a partir de A y B:

A	B	S_0	S_1	S_2	S_3	S_4	S_5
0	0	0	0	0	0	0	0
0	1	0	1	1	0	0	0
1	0	1	0	0	1	0	0
1	1	0	1	1	0	1	1

A partir de ella, se puede hacer uso de *minterms* y *maxterms* para obtener la expresión que representa cada salida. A continuación, se listan expresiones válidas para cada segmento, junto con su reducción a través de equivalencias lógicas.

- $S_0 = S_3 = A \wedge \overline{B}$
- $S_1 = S_2 = (\overline{A} \wedge B) \vee (A \wedge B) = (\overline{A} \vee A) \wedge B = B$
- $S_4 = S_5 = A \wedge B$

Finalmente, se puede diseñar un circuito a partir de estas expresiones. Se incluye un circuito para las primeras expresiones obtenidas con *minterms* y *maxterms* (izquierda), y otro reducido a partir de equivalencias lógicas (derecha).



Es importante notar que se evaluará la correctitud del circuito diseñado, no que sea igual al de esta pauta. Se otorgan **0.5 ptos.** por el circuito correctamente diseñado de cada segmento. En cada caso se otorga la mitad del puntaje si existe, como máximo, una combinación de señales de entrada que generen una salida incorrecta.

Feedback ayudantía

Escanee el QR para entregar feedback sobre la ayudantía.

