75.562 · Fundamentos de Computadores · 2023-24

PEC2 - Segunda prueba de evaluación continua

Apellidos: *López Henestrosa*
Nombre: José Carlos

Formato y fecha de entrega

- Para dudas y aclaraciones sobre el enunciado debéis dirigiros al consultor responsable de vuestra aula.
- Hay que entregar la solución en un fichero PDF utilizando una de las plantillas entregadas conjuntamente con este enunciado.
- Se debe entregar a través de la aplicación de **Entrega de la Actividad** correspondiente del apartado **Contenidos** de vuestra aula.
- La fecha límite de entrega es el **9 de abril** (a las 24 horas).
- **Razonad la respuesta en todos los ejercicios. Las respuestas sin justificación no recibirán puntuación.**

Respuestas

Ejercicio 1 [25 %]

Dada la siguiente expresión lógica de la función $f(a, b, c, d)$:

$$f(a,b,c,d) = \overline{(\overline{a} + b + c)} + (a \cdot b \cdot c) + \overline{c} + (a \oplus b) \cdot d + \overline{d}$$

a) [10 %] Simplificad la función f usando las propiedades y las leyes del álgebra de Boole.
(Secciones 1.2. Álgebra de Boole y 1.3.1. Expresiones algebraicas)

A continuación, se muestra una posible minimización a partir de las leyes del álgebra de Boole. Hay que tener en cuenta que pueden existir varias soluciones según el orden de aplicación de las leyes:

```
f = (a'+b+c)' + abc + c' + d(ab' + a'b) + d' =
  = a''b'c' + abc + c' + d(ab' + a'b) + d' =  → Ley de De Morgan
  = ab'c' + abc + c' + d(ab' + a'b) + d' =    → Ley de involución
  = abc + c' + d(ab' + a'b) + d'              → Ley de absorción
```

b) [5 %] Obtened la tabla de verdad de la función.
(Sección 1.2. Tablas de verdad)

NOTA: Tenéis disponible el ejercicio en VerilChart para verificarlo (`20242_PAC2_1b`). No hay limitación en el número de intentos.

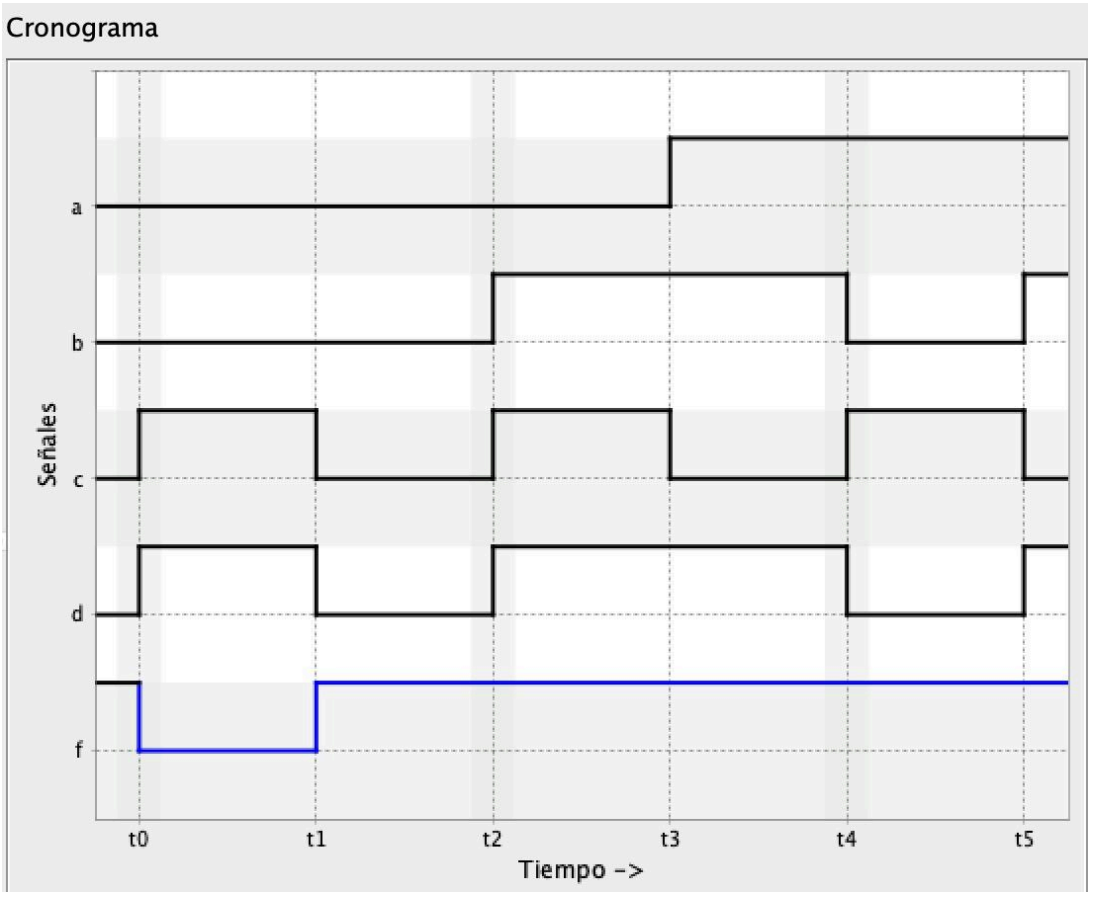
La expresión algebraica obtenida en el anterior apartado consta de 4 variables (`a`, `b`, `c` y `d`). Por lo tanto, la tabla de verdad tendrá $2^4 = 16$ combinaciones posibles. A partir de este principio y de las subexpresiones intermedias de la expresión algebraica (indicadas en la tabla de verdad), podemos determinar los valores de cada columna de la siguiente forma:

a	b	c	d	abc	c'	ab'	a'b	ab'+a'b	d(ab'+a'b)	d'	f
0	0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	1	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	1	1
0	0	1	1	0	0	0	0	0	0	0	0
0	1	0	0	0	1	0	1	1	0	1	1
0	1	0	1	0	1	0	1	1	1	0	1
0	1	1	0	0	0	0	1	1	0	1	1
0	1	1	1	0	0	0	1	1	1	0	1
1	0	0	0	0	1	1	0	1	0	1	1
1	0	0	1	0	1	1	0	1	1	0	1
1	0	1	0	0	0	1	0	1	0	1	1
1	0	1	1	0	0	1	0	1	1	0	1
1	1	0	0	0	1	0	0	0	0	1	1
1	1	0	1	0	1	0	0	0	0	0	1
1	1	1	0	1	0	0	0	0	0	1	1
1	1	1	1	1	0	0	0	0	0	0	1

c) [10 %] Completad el siguiente cronograma:
(Sección 2.2.1. Retardos. Cronogramas. Niveles de puertas)

NOTA: Tenéis disponible el ejercicio en VerilChart para verificarlo (`20242_PAC2_1c`). No hay limitación en el número de intentos.

Vamos buscando en el cronograma las diferentes combinaciones `abcd` y miramos cuál es la salida `f` en cada caso. Por ejemplo, en el primer intervalo, vemos que `abcd = 0011` y, para esta combinación, la salida `f` vale 0, por lo que añadimos el valor 0 al cronograma. Y así sucesivamente, obtenemos el siguiente cronograma:



Captura del cronograma hecho en VerilChart

Ejercicio 2 [20 %]

Dada la siguiente tabla de verdad:

a	*b*	*c*	*d*	*f*	*g*
0	0	0	0	X	1
0	0	0	1	0	0
0	0	1	0	1	0
0	0	1	1	0	1
0	1	0	0	X	0
0	1	0	1	0	0
0	1	1	0	1	1
0	1	1	1	0	0
1	0	0	0	0	1
1	0	0	1	X	0
1	0	1	0	0	1
1	0	1	1	1	1
1	1	0	0	0	1
1	1	0	1	X	0
1	1	1	0	1	1
1	1	1	1	1	0

NOTA: Tenéis disponible el ejercicio en VerilCirc (`20242_PAC2_2a_circ` y `20242_PAC2_2c`) y en KeMAP (`20242_PAC2_2a_KM`) para verificarlo. Os recomendamos hacer pruebas con KeMAP con otros ejercicios antes de hacer el de la PEC.

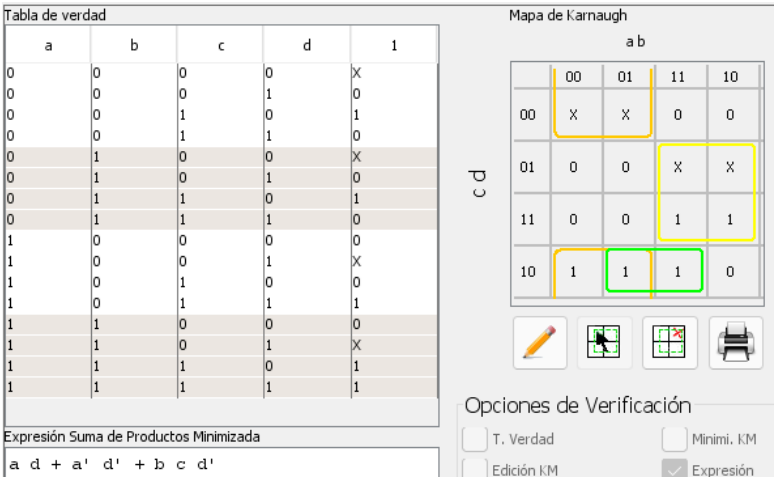
a) [7,5 %] Utilizando el método de Karnaugh, indicad las dos funciones mínimas posibles de la función f. ¿Cuál de las dos expresiones mínimas utiliza una cantidad menor de puertas lógicas? Implementad esta última opción con puertas lógicas en dos niveles.
(Secciones 2.3.2. Síntesis mínima a dos niveles. Método de Karnaugh)

Uno de los mapas de Karnaugh posibles para la función `f` es el siguiente:

Del cual obtenemos esta expresión mínima:

$$f_1 = ad + a'd' + abc$$

El otro mapa de Karnaugh posible es este:



Del que hallamos esta expresión mínima:

$$f_2 = ad + a'd' + bcd'$$

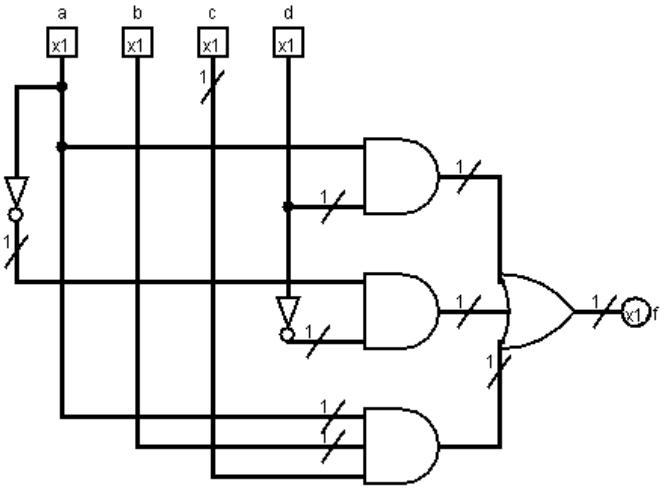
Este es el desglose de las puertas lógicas que requiere cada expresión:

- f_1
  - 2 puertas lógicas NOT (`a'` y `d'`).
  - 3 puertas lógicas AND (`ad`, `a'd'` y `abc`).
  - 1 puerta lógica OR (los 3 términos se suman).
  - **Total**: 6

- f_2
  - 2 puertas lógicas NOT (`a'` y para `d'` se usa dos veces la misma puerta).
  - 3 puertas lógicas AND (`ad`, `a'd'` y `bcd'`).
  - 1 puerta lógica OR (los 3 términos se suman).
  - **Total**: 6

Como podemos observar, ambas expresiones utilizan el mismo número de puertas lógicas. No obstante, es preferible decantarse por la primera (f_1), ya que solo requiere invertir una conexión de `d'`, lo que simplifica el diseño físico del circuito y optimiza su velocidad al evitar el retardo que se añade cuando la señal es procesada por una puerta lógica.

El circuito que implementa la expresión mínima con la menor cantidad de puertas lógicas a dos niveles (f_1) es el siguiente:



Captura del circuito hecho en VerilCirc

b) [5 %] Hemos preguntado a una herramienta de IA generativa la minimización mediante el método de Karnaugh de la función g, y nos ha dado el siguiente resultado:
(Sección 2.3.2. Síntesis mínima a dos niveles. Método de Karnaugh)

Paso 1: *Organización en un mapa de Karnaugh 4x4*

ab\cd	00	01	11	10
00	1	0	1	0
01	0	0	0	1
11	1	0	1	1
10	1	0	1	1

Paso 2: *Agrupaciones y obtención de la función mínima*

- ***Grupo 1***: $g = a$ *(seleccionando la columna* $a = 1$ *donde* $g = 1$*)*
- ***Grupo 2***: $g = c \cdot d$ *(seleccionando la columna donde* c *y* d*)*

Resultado final: *Función mínima simplificada:*

$g(a, b, c, d) = a + (c \cdot d)$

Razonad la corrección de la solución y, en caso de ser incorrecta, indicad en cada paso los errores detectados, pero no hace falta obtener la solución correcta.

La solución propuesta por la IA tiene errores en todos sus pasos, los cuales identificamos en el siguiente desglose:

- **Paso 1**: El mapa de Karnaugh generado es **incorrecto**, ya que el valor para `abcd = 1111` es 0, no 1.
- **Paso 2**: Los grupos identificados en base al mapa incorrectamente generado son **incorrectos**. Además, son **insuficientes**, ya que existen tres grupos más.

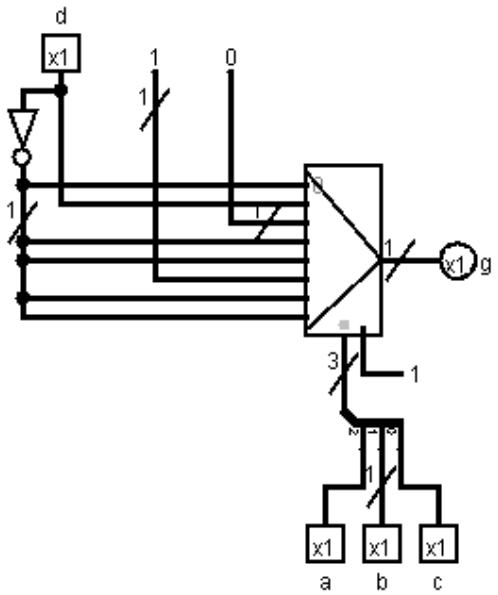
En consecuencia, la función mínima simplificada propuesta en el resultado final es **incorrecta**.

c) [7,5%] Sintetizad la función *g* con un multiplexor de 3 variables de control y el menor número de puertas lógicas.
(Sección 3.1. Multiplexor. Multiplexor de buses. Demultiplexor)

En este caso, estamos sintetizando una función de 4 variables con un multiplexor que tiene 3 variables de control (`abc`). Para cada combinación de `abc`, el valor de `g` depende de `d` de forma específica. A partir de la tabla de verdad del ejercicio, determinamos cada valor de `g` para cada caso, el cual puede ser 0, 1 o `d` (o su negado), tal y como se aprecia en esta tabla:

a	b	c	Salida g (en función de d)	Línea MUX
0	0	0	g = d'	0
0	0	1	g = d	1
0	1	0	g = 0	2
0	1	1	g = d'	3
1	0	0	g = d'	4
1	0	1	g = 1	5
1	1	0	g = d'	6
1	1	1	g = d'	7

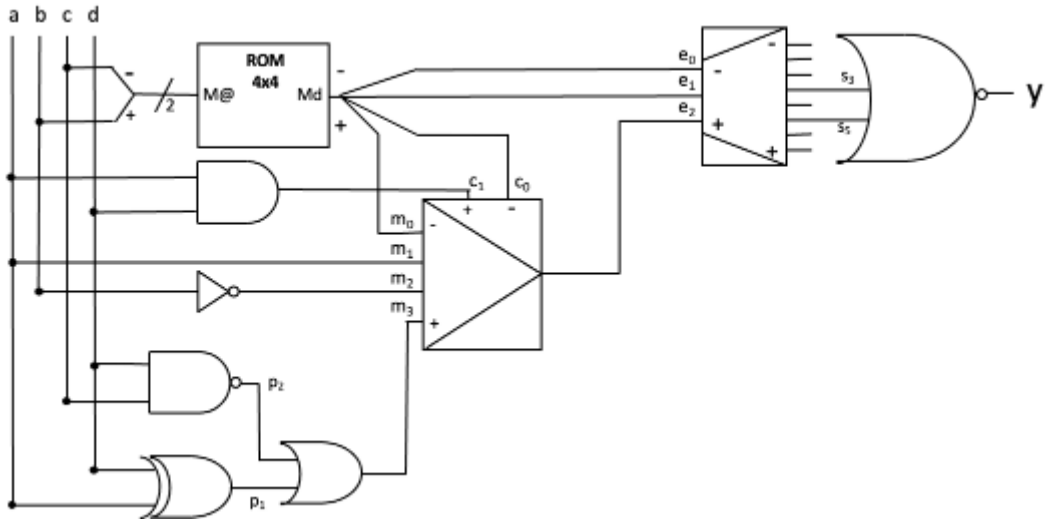
En base a la salida `g` para cada línea del multiplexor, elaboramos el siguiente circuito:



Captura del circuito hecho en VerilCirc

Ejercicio 3 [25 %]

Dado el circuito lógico combinacional siguiente:



Donde el contenido de la memoria ROM es el siguiente:

M@	Md
0	0000
1	0111
2	1100
3	1001

Completad la tabla de verdad que especifica la salida y en función de las entradas a, b, c y d. Calculad previamente los valores intermedios (p_2, p_1, m_3, m_2, m_1, m_0, c_1, c_0, e_2, e_1, e_0, s_5, s_3) indicados en el circuito y añadidos en la tabla de verdad siguiente: (Sección 3. Bloques combinacionales)

NOTA: Tenéis disponible el ejercicio en VerilChart para verificarlo (`20242_PAC2_3`). No hay limitación en el número de intentos.

El circuito recibe como entrada las señales a, b, c y d, las cuales se conectan a otros bloques combinacionales hasta llegar a la salida y. Podemos obtener su tabla de verdad a partir del circuito mostrado en la imagen superior.

Para entender la procedencia de cada valor, realizamos el siguiente desglose:

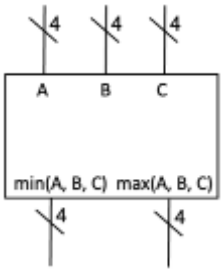
- p_2: (cd)'
- p_1: a⊕d
- m_3: p1+p2
- m_2: b'
- m_1: a
- m_0: Md_3
- c_1: ad
- c_0: Md_2
- e_2: Salida del multiplexor
- e_1: Md_1
- e_0: Md_0
- s_5: Salida s_5 del decodificador
- s_3: Salida s_3 del decodificador
- y: (s_5+s_3)

Con esta información, podemos completar las columnas de la tabla:

a	b	c	d	p_2	p_1	m_3	m_2	m_1	m_0	c_1	c_0	e_2	e_1	e_0	s_5	s_3	y
0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	1
0	0	1	0	1	0	1	1	0	0	0	1	0	1	1	0	1	0
0	0	1	1	0	1	1	1	0	0	0	1	0	1	1	0	1	0
0	1	0	0	1	0	1	0	0	1	0	1	0	0	0	0	0	1
0	1	0	1	1	1	1	0	0	1	0	1	0	0	0	0	0	1
0	1	1	0	1	0	1	0	0	1	0	0	1	0	1	1	0	0
0	1	1	1	0	1	1	0	0	1	0	0	1	0	1	1	0	0
1	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	1
1	0	0	1	1	0	1	1	1	0	1	0	1	0	0	0	0	1
1	0	1	0	1	1	1	1	1	0	0	1	1	1	1	0	0	1
1	0	1	1	0	0	0	1	1	0	1	1	0	1	1	0	1	0
1	1	0	0	1	1	1	0	1	1	0	1	1	0	0	0	0	1
1	1	0	1	1	0	1	0	1	1	1	1	1	0	0	0	0	1
1	1	1	0	1	1	1	0	1	1	0	0	1	0	1	1	0	0
1	1	1	1	0	0	0	0	1	1	1	0	0	0	1	0	0	1

Ejercicio 4 [30 %]

Se desea diseñar un circuito que, dados 3 números de 4 bits en complemento a 2, muestre el menor de los tres valores de salida $min\,(A, B, C)$ y el mayor de los tres valores en la salida $max(A, B, C)$ siguiendo la siguiente estructura:
(Sección 3. Bloques combinacionales)



NOTA: Tenéis disponible el ejercicio en VerilCirc para verificarlo (`20242_PAC2_4a` y `20242_PAC2_4b`). No hay límite en el número de intentos que uséis para comprobar vuestra solución.

NOTA2: En el caso de diseñar el circuito en VerilCirc, la herramienta no admite utilizar bloques diseñados por el usuario. Por lo tanto, en caso de reutilizar un circuito, como por ejemplo el circuito del apartado a), tendréis que copiar el circuito tantas veces como necesitéis utilizarlo.

a) [20%] Diseñad un circuito que, dados dos números A y B de 4 bits en complemento a 2, obtenga el menor y el mayor de los dos operandos a partir de bloques combinacionales (excepto ROMs) y las puertas que consideréis necesarias. Llamad a las salidas del circuito como $min2$ para $min(A, B)$ y $max2$ para $max(A, B)$.

Sabemos que el MSB de los números binarios en complemento a 2 determinan su signo (0 para positivo y 1 para negativo). Si los signos son diferentes, el número negativo es siempre menor, por lo que podemos determinar fácilmente qué cuál es el menor y el mayor a través de un comparador.

En cambio, si los signos son iguales, tenemos que realizar una resta en complemento a 2 para, a partir del MSB del resultado, determinar cuál de los dos números es el menor y el mayor. Para ello, usamos un sumador, cuya entrada `A` será el número `A` y cuya entrada `B` será el número `B` invertido. Además, también tenemos que introducir un acarreo inicial de 1 bit al sumador para realizar la operación correctamente.

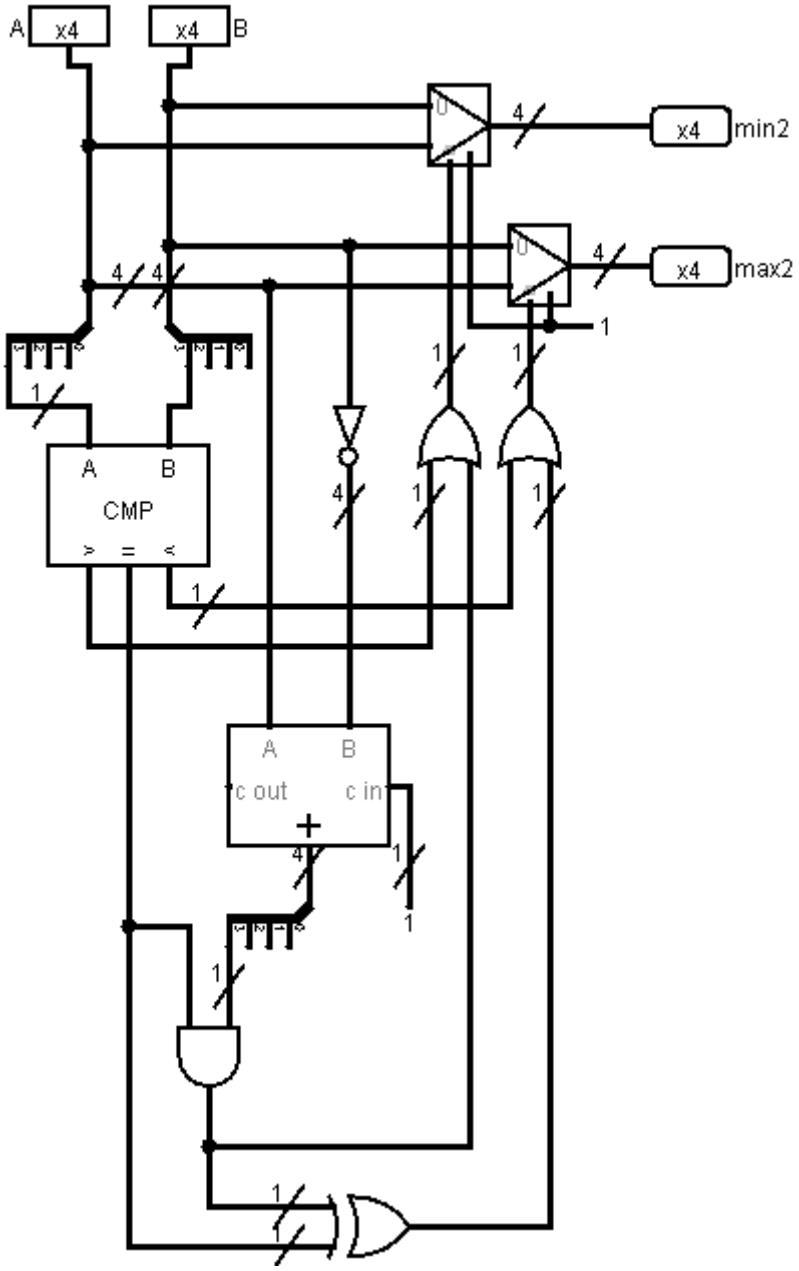
La interpretación del resultado de la resta es la siguiente:

- Si `A - B = -X` (MSB es 1) y los signos de `A` y `B` son iguales → `A < B`.
  Si no se cumple ninguna de las dos condiciones → `A > B`.
- Si `A - B = +X` (MSB es 0) o los signos de `A` y `B` son iguales, pero no ambos al mismo tiempo → `A > B`.
  Si ambas condiciones se dan al mismo tiempo → `A < B`.

Una vez tenemos la lógica planteada para determinar las condiciones en las que `A` es menor o mayor que `B`, conectamos cada señal a la entrada de selección del multiplexor que le corresponda:

- Lógica del multiplexor que determina qué número es el **menor** (`min2`):
  - Si la entrada de selección es 0, `B` es `min2`.
  - Si la entrada de selección es 1, `A` es `min2`.
- Lógica del multiplexor que determina qué número es el **mayor** (`max2`):
  - Si la entrada de selección es 0, `B` es `max2`.
  - Si la entrada de selección es 1, `A` es `max2`.

A partir de este razonamiento, podemos elaborar el circuito, el cual queda tal que así:



Captura del circuito hecho en VerilCirc

b) [10%] Utilizando el circuito del apartado a), bloques combinacionales (excepto ROMs) y las puertas que consideréis necesarias, diseñad el circuito $min(A, B, C)$ y $max(A, B, C)$. Llamad a las salidas del circuito como $min3$ y $max3$ respectivamente.

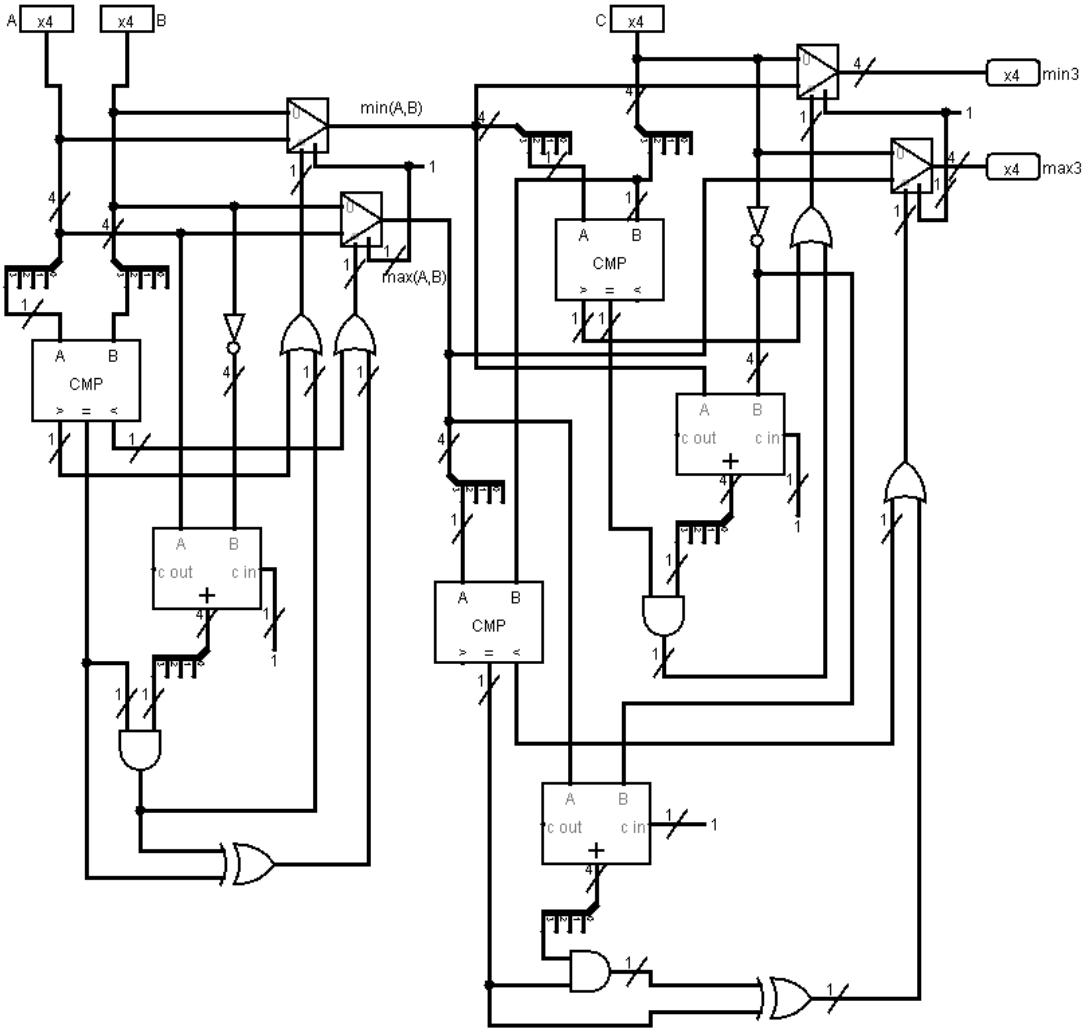
A partir del circuito planteado en el apartado anterior, añadimos la segunda parte propuesta en este apartado, la cual consiste en añadir un tercer número (`C`) y determinar cuál de los tres es menor y cuál es mayor. Para ello, repetimos el proceso explicado en el apartado a) para las salidas `min2` y `max2`. No obstante, la lógica para ambos casos se simplifica, ya que en la comparación entre `C` y `max2` solo nos interesa saber si `max2` es menor que `C`, y de forma similar, para `min2` solo nos interesa saber si `min2` es mayor que `C`.

- Si `min2 - C = -X` (MSB es 1) y los signos de `min2` y `C` son iguales → `min2 < C`.
  Si no se cumple ninguna de las dos condiciones → `min2 > C`.
- Si `max2 - C = +X` (MSB es 0) o los signos de `max2` y `C` son iguales, pero no ambos al mismo tiempo → `max2 > C`.
  Si ambas condiciones se dan al mismo tiempo → `max2 < C`.

Al igual que antes, cada señal se conecta a la entrada de selección del multiplexor que le corresponda, los cuales siguen la siguiente lógica:

- Lógica del multiplexor que determina qué número es el **menor** (`min3`):
  - Si la entrada de selección es 0, `C` es `min3`.
  - Si la entrada de selección es 1, `min2` es `min3`.
- Lógica del multiplexor que determina qué número es el **mayor** (`max2`):
  - Si la entrada de selección es 0, `C` es `max3`.
  - Si la entrada de selección es 1, `min2` es `max3`.

A partir de este razonamiento y del circuito y la lógica planteados en el apartado a), obtenemos lo siguiente:



Captura del circuito hecho en VerilCirc