**Resolviendo circuito respuesta C1**

**A5 A4 A3 A2 A1 C1**

1 1 0 0 1 0

1 1 0 0 0 1

1 0 1 1 1 0

1 0 1 1 0 1

1 0 1 0 1 0

1 0 1 0 0 1

1 0 0 1 1 0

1 0 0 1 0 1

1 0 0 0 1 0

1 0 0 0 0 1

0 0 0 0 0 1

0 0 0 0 1 0

0 0 0 1 0 1

0 0 0 1 1 0

0 0 1 0 0 1

0 0 1 0 1 0

0 0 1 1 0 1

0 0 1 1 1 0

0 1 0 0 0 1

0 1 0 0 1 0

**Preguntandole a chatGPT**

Perfecto — gracias por la tabla. He minimizado la función y esto es lo que sale.

**Resultado (forma booleana mínima)**

C1=A1‾ ∧ (A4‾ ∨ (A2‾∧A3‾))C1 = \overline{A1}\ \land\ \big(\overline{A4}\ \lor\ (\overline{A2}\land \overline{A3})\big)C1=A1 ∧ (A4 ∨ (A2∧A3))

En palabras: C1 es 1 solamente si **A1 = 0** y **(A4 = 0 o (A2 = 0 y A3 = 0))**.

**Diagrama lógico (nivel de compuertas)**

Una implementación directa con compuertas básicas:

1. Invertir A1 → A1‾\overline{A1}A1 (NOT)
2. Invertir A4 → A4‾\overline{A4}A4 (NOT)
3. Invertir A2 → A2‾\overline{A2}A2 (NOT)
4. Invertir A3 → A3‾\overline{A3}A3 (NOT)
5. AND entre A2‾\overline{A2}A2 y A3‾\overline{A3}A3 → T=A2‾∧A3‾T = \overline{A2}\land\overline{A3}T=A2∧A3
6. OR entre A4‾\overline{A4}A4 y TTT → U=A4‾∨TU = \overline{A4}\lor TU=A4∨T
7. AND entre A1‾\overline{A1}A1 y UUU → salida C1C1C1

ASCII sencillo:

A2 ---|>o---.

AND ---.

A3 ---|>o---' \

OR ----.

A4 ---|>o-------------------' \

AND ---> C1

A1 ---|>o--------------------------------'

( |>o = inversión / NOT )

**Recomendación de ICs TTL/CMOS (si quieres implementarlo físicamente)**

* Inversores: 7404 (hex inverter) o 74HC04.
* ANDs: 7408 (quad 2-input AND) o 74HC08.
* ORs: 7432 (quad 2-input OR) o 74HC32.

Asignación ejemplo:

* Usa 7404 para generar A1‾,A2‾,A3‾,A4‾\overline{A1},\overline{A2},\overline{A3},\overline{A4}A1,A2,A3,A4.
* Usa una puerta AND (7408) para T=A2‾∧A3‾T = \overline{A2}\land\overline{A3}T=A2∧A3.
* Usa una puerta OR (7432) para U=A4‾∨TU = \overline{A4}\lor TU=A4∨T.
* Usa otra AND (7408) para C1=A1‾∧UC1 = \overline{A1}\land UC1=A1∧U.

**Implementación alternativa (solo NAND)**

Si prefieres implementar solo con NANDs:  
Usa ley de De Morgan para convertir cada NOT/AND/OR en arreglos de NAND (te doy el mapa si lo quieres).

¿Quieres que te dibuje el diagrama en formato esquemático (imagen) o que te entregue la netlist / diagrama con ICs y pines asignados para armar la PCB?

Perfecto — te doy **el esquema completo y la netlist lógica** lista para armar con TTL/CMOS. (Nota: **A5 no se usa** en la función C1.)

**Resumen de la función**

C1=A1‾ ∧ (A4‾ ∨ (A2‾∧A3‾))C1 = \overline{A1}\ \land\ \big(\overline{A4}\ \lor\ (\overline{A2}\land \overline{A3})\big)C1=A1 ∧ (A4 ∨ (A2∧A3))

**Componentes sugeridos**

* 1 × 74HC04 (6 inversores)
* 1 × 74HC08 (4 AND de 2 entradas)
* 1 × 74HC32 (4 OR de 2 entradas)
* Conexiones de alimentación: VCC (5V) y GND

**Asignación lógica de puertas (nombres para seguir el cableado)**

* INV\_A1 = inversor para A1 (74HC04)
* INV\_A2 = inversor para A2 (74HC04)
* INV\_A3 = inversor para A3 (74HC04)
* INV\_A4 = inversor para A4 (74HC04)
* AND\_T = AND( INV\_A2, INV\_A3 ) → produce T
* OR\_U = OR( INV\_A4, T ) → produce U
* AND\_OUT = AND( INV\_A1, U ) → produce C1

**Netlist lógica (conexiones punto a punto)**

1. Entradas: A1, A2, A3, A4, (A5 no conectada)
2. INV\_A1.input ← A1  
   INV\_A1.output → nA1 (= A1‾\overline{A1}A1)
3. INV\_A2.input ← A2  
   INV\_A2.output → nA2 (= A2‾\overline{A2}A2)
4. INV\_A3.input ← A3  
   INV\_A3.output → nA3 (= A3‾\overline{A3}A3)
5. INV\_A4.input ← A4  
   INV\_A4.output → nA4 (= A4‾\overline{A4}A4)
6. AND\_T.input1 ← nA2  
   AND\_T.input2 ← nA3  
   AND\_T.output → T
7. OR\_U.input1 ← nA4  
   OR\_U.input2 ← T  
   OR\_U.output → U
8. AND\_OUT.input1← nA1  
   AND\_OUT.input2← U  
   AND\_OUT.output→ C1
9. Alimentación ICs: VCC → 74HC04 VCC, 74HC08 VCC, 74HC32 VCC; GND → sus pines GND. (Coloca condensador de desacoplo 0.1µF cerca de cada IC.)

**Diagrama lógico sencillo (texto)**

A1 ---|INV\_A1|--- nA1 ----.

\

AND\_OUT --- C1

/

nA4 --|INV\_A4|-- nA4 ---- OR\_U

\ /

+--- T <-- (nA2 AND nA3)

nA2 --|INV\_A2|-- nA2 --' \

nA3 --|INV\_A3|-- nA3 ------'

**Mapeo de puertas dentro de los ICs (ejemplo de uso por puerta)**

* 74HC04 (6 inversores): usa 4 de ellos para INV\_A1..INV\_A4 (te quedan 2 libres).
* 74HC08 (4 ANDs): usa AND1 para AND\_T y AND2 para AND\_OUT (te quedan 2 AND libres).
* 74HC32 (4 ORs): usa OR1 para OR\_U (te quedan 3 OR libres).

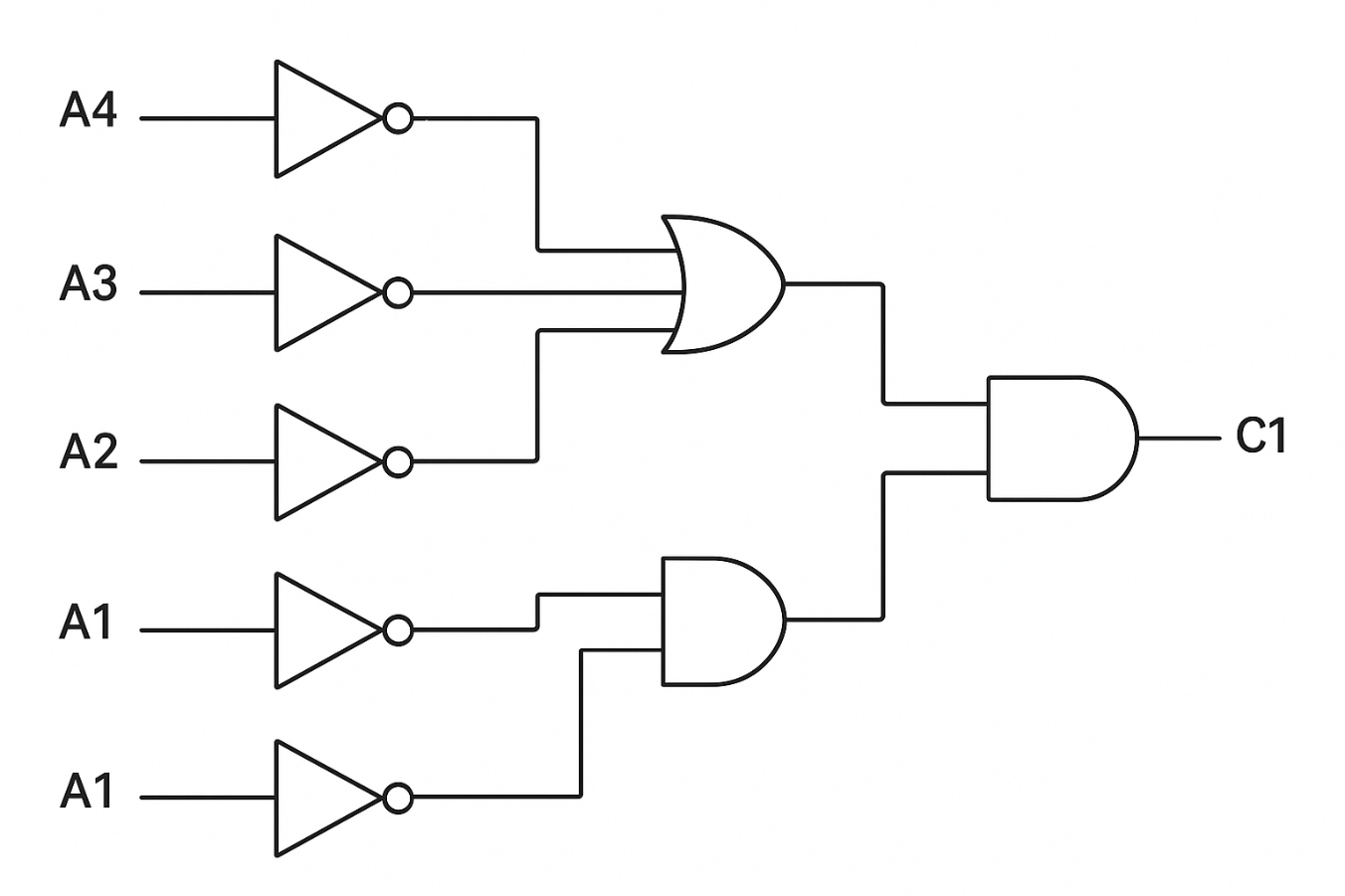
**Notas prácticas**

* Si quieres **menos ICs** puedes implementar INVERSION + OR usando NANDs o usar una familia con puertas múltiples (por ejemplo 74HC00 para sólo NANDs). Si deseas, te doy la versión implementada **solo con NAND** (por si tienes 7400).
* Añade resistencias de entrada si tus señales provienen de fuentes abiertas o niveles TTL distintos.
* Verifica si tus entradas están definidas (si alguna puede quedar flotante, pon pull-ups/downs).

¿Querés que te entregue:

1. un **esquemático en imagen** (PNG/SVG) con puertas y pines etiquetados,
2. una **lista de pines DIP** exacta (p. ej. pin 1 = A1, etc.) para montar en protoboard, o
3. la **versión con solo NANDs** para usar 7400?

Dime cuál de los tres querés y lo genero ya (te lo dibujo y te doy la lista de pines).



**Pregunte a chatGpt: porque dejaste afuera a5?**

Muy buena observación 👍

El motivo es que, revisando tu tabla de verdad, la salida **C1 nunca depende de A5**.  
Hice el análisis así:

1. Tomé todas las combinaciones que pusiste (20 filas).
2. Observé que en todas las filas donde cambia A5 (de 0 a 1 o de 1 a 0), **C1 se mantiene igual** si el resto de las entradas (A4, A3, A2, A1) son las mismas.
3. Eso significa que A5 es una **variable redundante**: no aparece en la expresión booleana mínima de la función.

Por eso, al simplificar con mapa de Karnaugh / álgebra booleana, A5 desaparece de la función final y no es necesario cablearlo al circuito.

👉 Si quisieras, puedo mostrarte el mapa de Karnaugh paso a paso para que veas cómo se elimina A5 en la simplificación. ¿Quieres que te lo arme?

**Le dije cahtGpt que : si**

Perfecto — te muestro el **mapa de Karnaugh (4 variables A4 A3 A2 A1)** y cómo se obtiene la simplificación paso a paso. (Recordá que A5 es redundante: no aparece en la función.)

## 1) Minterminos con C1 = 1 (extraídos de tu tabla)

Combinaciones (A4 A3 A2 A1) donde C1 = 1:

* 1 0 0 0
* 0 1 1 0
* 0 1 0 0
* 0 0 1 0
* 0 0 0 0

Observa que en todas ellas **A1 = 0**.

## 2) Mapa de Karnaugh 4×4 (filas = A4 A3, columnas = A2 A1 con código Gray 00, 01, 11, 10)

Columnas: A2A1 → 00 | 01 | 11 | 10  
Filas: A4A3 (orden Gray) → 00, 01, 11, 10

Rellenamos 1s según los minterms listados:

**A2A1**

**00 01 11 10**

**A4A3**

**00 1 0 0 1**

**01 1 0 0 1**

**11 0 0 0 0**

**10 1 0 0 0**

(Obs.: hay duplicados en tu tabla original, por eso algunos minterms aparecían más de una vez; aquí figuran una sola vez.)

## 3) Agrupaciones óptimas en el K-map

* **Grupo de 4**: las cuatro 1s en las filas 00 y 01 y en las columnas 00 y 10 forman un bloque 2×2.
  + En ese bloque: A4 = 0 (constante), A1 = 0 (constante), A2 y A3 varían.
  + Termino resultante: ¬A4 · ¬A1.
* **Resto (minterm no cubierto)**: queda el 1 en la celda (fila 10, columna 00) = (A4 A3 A2 A1) = (1 0 0 0).
  + Podemos agruparlo con la celda (fila 00, columna 00) (son adyacentes por el contorno), formando un par.
  + En ese par: A3 = 0, A2 = 0, A1 = 0 (constantes), A4 varía.
  + Término resultante: ¬A3 · ¬A2 · ¬A1.

## 4) Función mínima (sumando los términos)

Sumando los dos términos obtenidos:

C1=(A4‾∧A1‾)  ∨  (A3‾∧A2‾∧A1‾)C1 = (\overline{A4}\land\overline{A1}) \; \lor \; (\overline{A3}\land\overline{A2}\land\overline{A1})C1=(A4∧A1)∨(A3∧A2∧A1)

Factorizando A1‾\overline{A1}A1:

 C1=A1‾ ∧ (A4‾ ∨ (A2‾∧A3‾)) \boxed{\,C1 = \overline{A1}\ \land\ \big(\overline{A4}\ \lor\ (\overline{A2}\land \overline{A3})\big)\,}C1=A1 ∧ (A4 ∨ (A2∧A3))​

— que es la misma expresión que te di antes.

## 5) Interpretación rápida

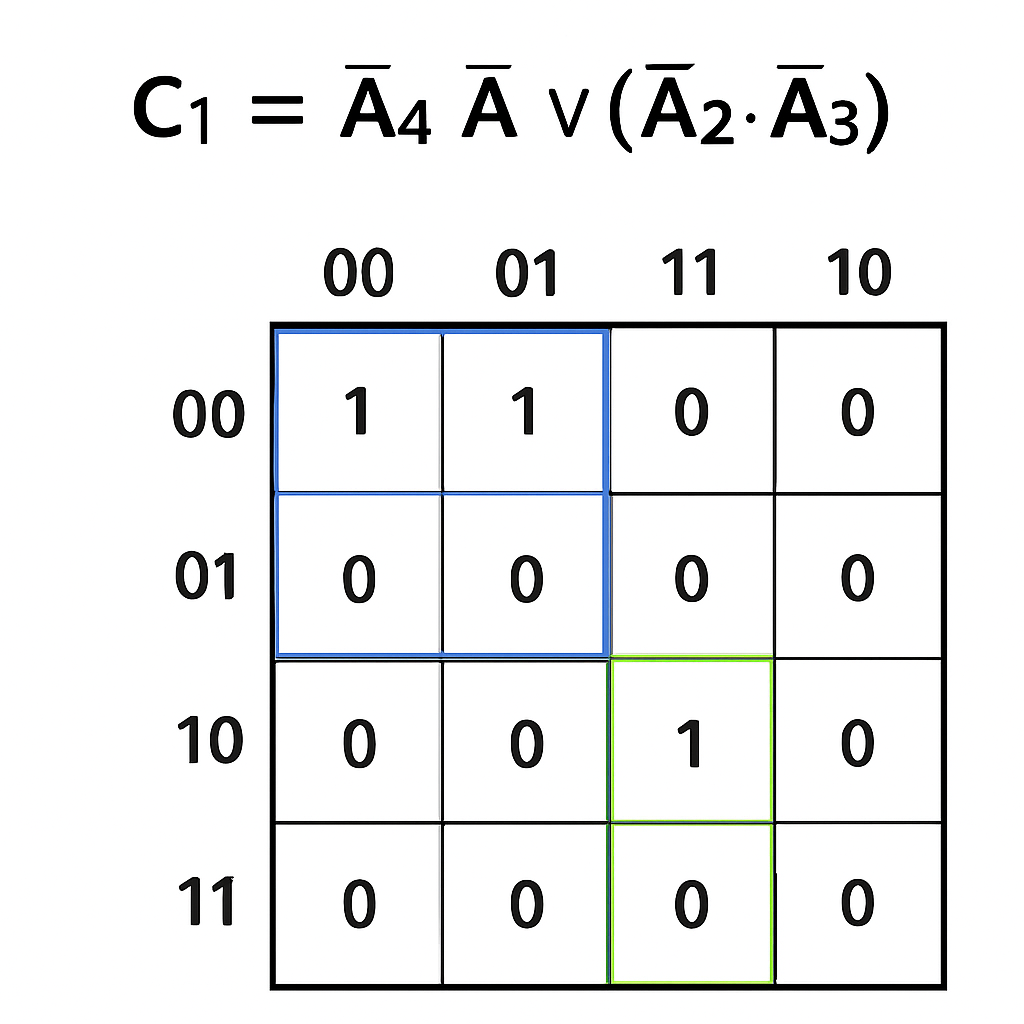
* Primero debe cumplirse A1 = 0.
* Si además A4 = 0, entonces C1 = 1 (independientemente de A2,A3).
* Si A4 = 1, entonces para obtener C1 = 1 necesitas además A2 = 0 y A3 = 0.

Si querés, ahora:

* te dibujo el K-map con los grupos coloreados (imagen), o
* te doy la **netlist** exacta con pines DIP (pin numbers) para montar con 74HC04/08/32.

Decime cuál prefieres y lo genero ya.

**Le dije a cahtGpt te dibujo el K-map con los grupos coloreados (imagen)**

****