

4.6 – 4.12

Síntesis y circuitos multinivel; herramientas CAD

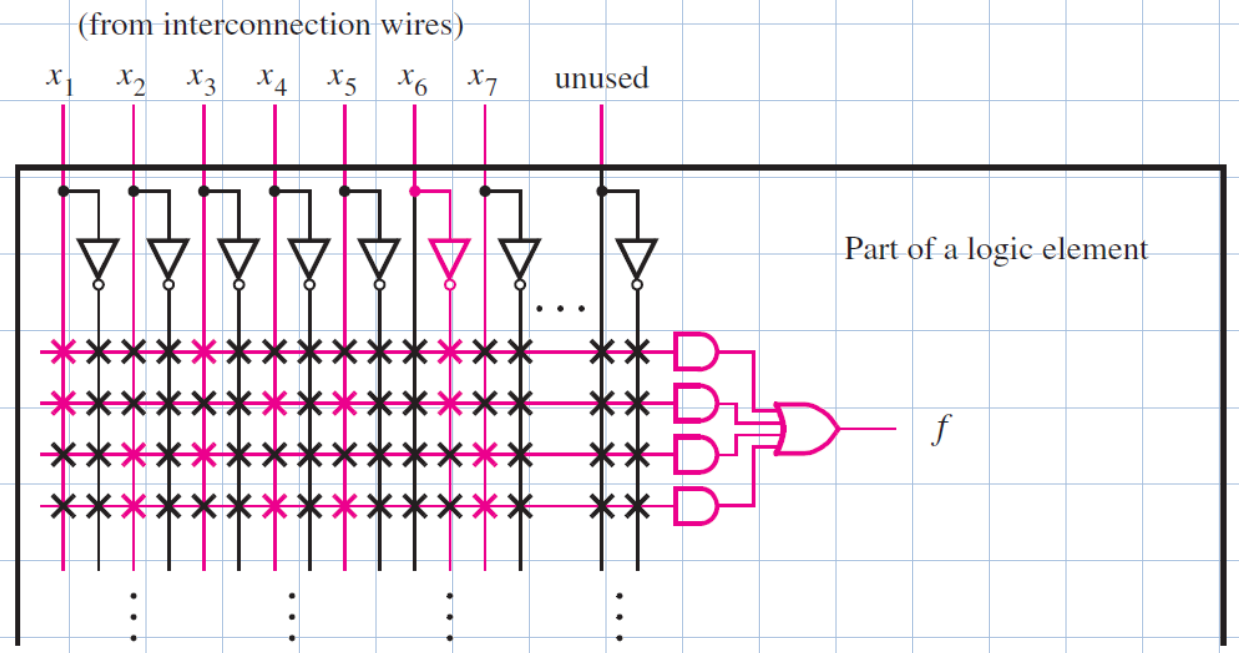
angel@uc.cl

IEE2713 Sistemas Digitales

4.6 Síntesis multinivel

Hablamos de síntesis multinivel cuando consideramos más de dos niveles de compuertas lógicas. Por ejemplo, en SOP, las compuertas AND de los productos sería el primer nivel conectado a una compuerta OR, que sería el segundo nivel.

$$f(x1 \ldots x7) = x_1 x_3 \bar{x_6} + x_1 x_4 x_5 \bar{x_6} + x_2 x_3 x_7 + x_2 x_4 x_5 x_7$$



Aquí hay sólo dos niveles

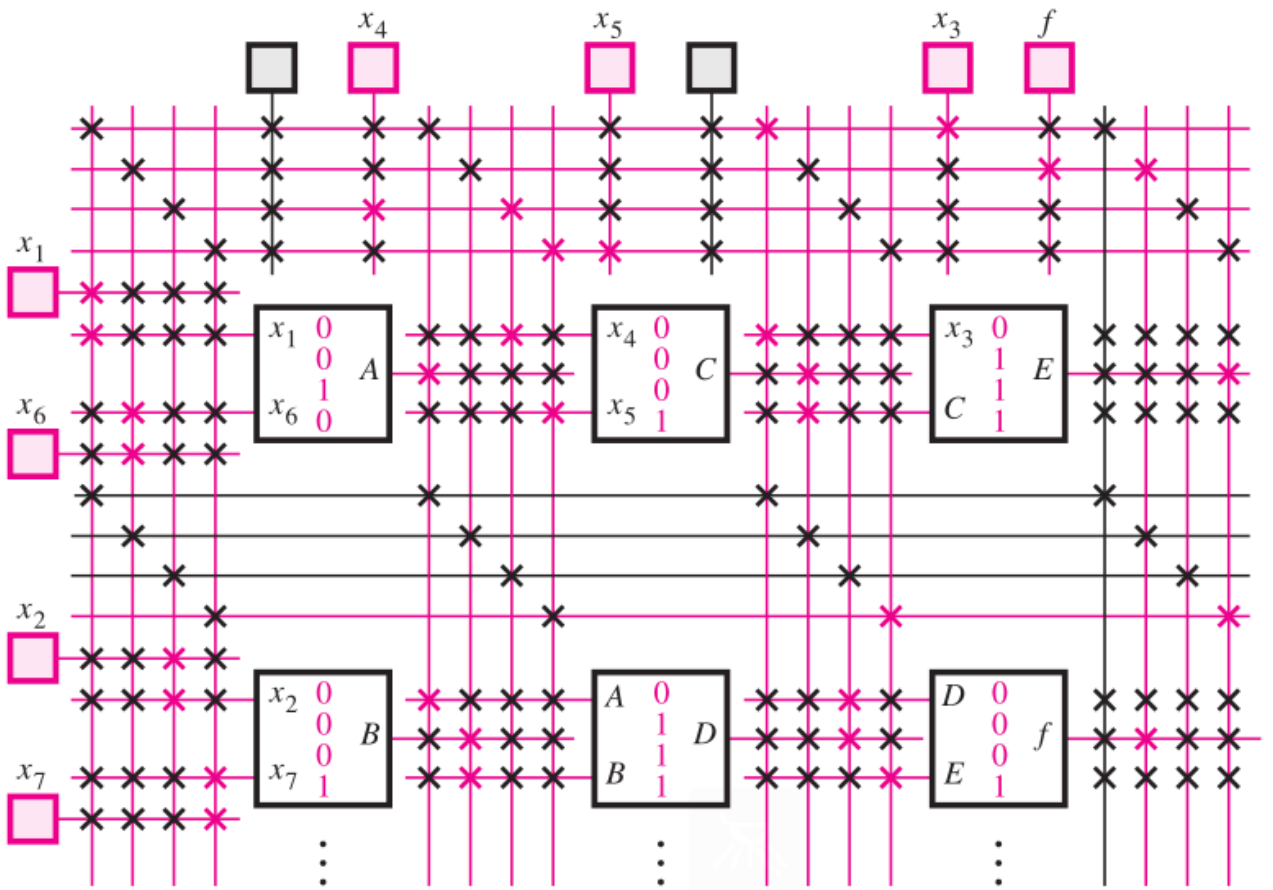
Figure 8.1. Implementation in a CPLD.

¿Entonces para qué hacer síntesis multinivel?

- Una realización de 2 niveles (simple POS o SOP) **sólo es eficiente en funciones de pocas variables**
- Cuando aparecen muchas variables, **crece el fan-in** y resulta en circuitos más **lentos**
- La implementación ideal, por supuesto, depende de la tecnología objetivo
  - Ej: si mi CPLD tiene PALs con fan-in grande, no hay mucho problema en hacer la implementación en 2 niveles
  - Si mi FPGA tiene LUTs con sólo 2 entradas (siguiente lámina), la única forma de implementar funciones con gran fan-in es con implementaciones multinivel
- Hay dos métodos: **factorización** y **descomposición funcional**

Implementación en FPGA

$$f(x1 \ldots x7) = x_1 x_3 \bar{x_6} + x_1 x_4 x_5 \bar{x_6} + x_2 x_3 x_7 + x_2 x_4 x_5 x_7$$



¿Cómo implementamos esta función lógica en un FPGA que tiene LUTs con sólo 2 entradas?

Una posibilidad…

Podemos implementar compuertas lógicas de mayor fan-in en forma más eficiente

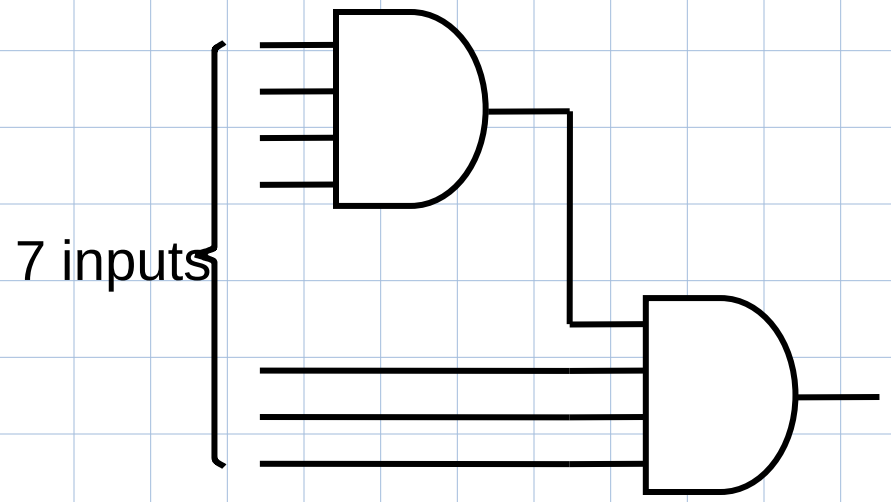


Figure 8.3. Using four-input AND gates to realize a seven-input product term.

Pero es más eficiente factorizar…

4.6.1 Factorización

Sinteticemos esta función

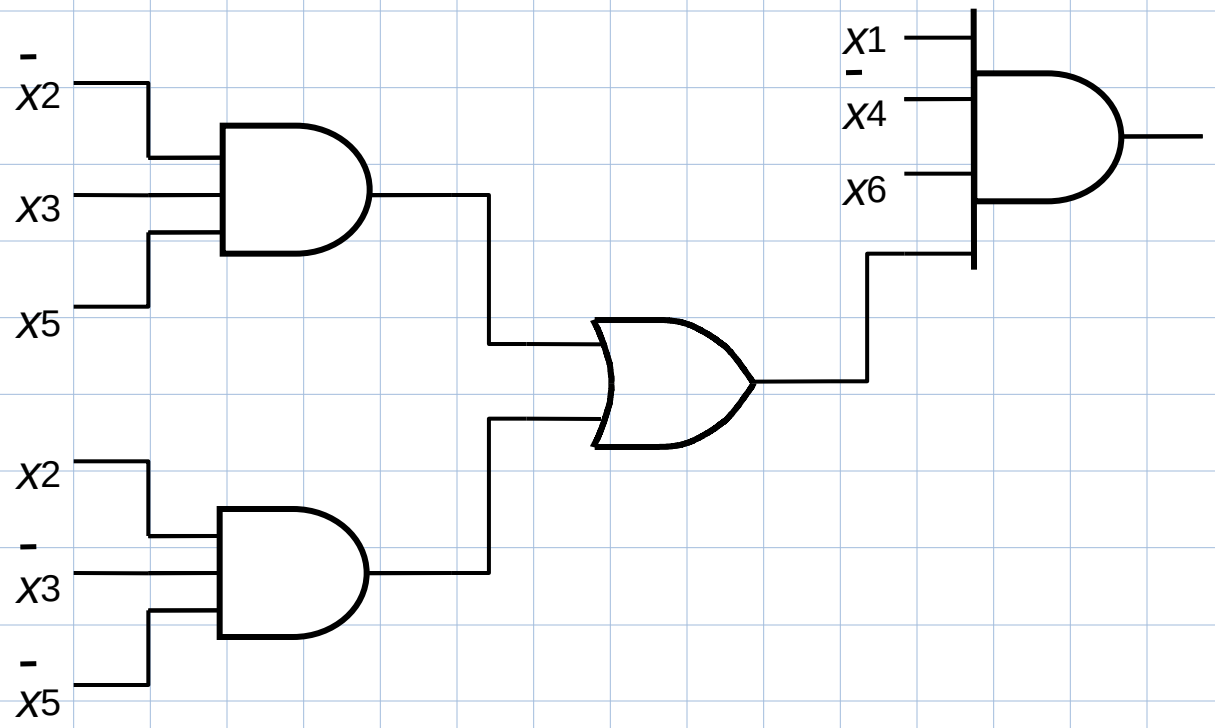
$$f(x1 \ldots x7) = x_1 x_3 \bar{x_6} + x_1 x_4 x_5 \bar{x_6} + x_2 x_3 x_7 + x_2 x_4 x_5 x_7$$

El fan-in es muy grande. Si queremos dejarlo limitado, por ejemplo, a 3, entonces tenemos que factorizar

$$\begin{aligned} f &= x_1 \overline{x_6}(x_3 + x_4 x_5) + x_2 x_7 (x_3 + x_4 x_5) \\ &= (x_1 \overline{x_6} + x_2 x_7)(x_3 + x_4 x_5) \end{aligned}$$

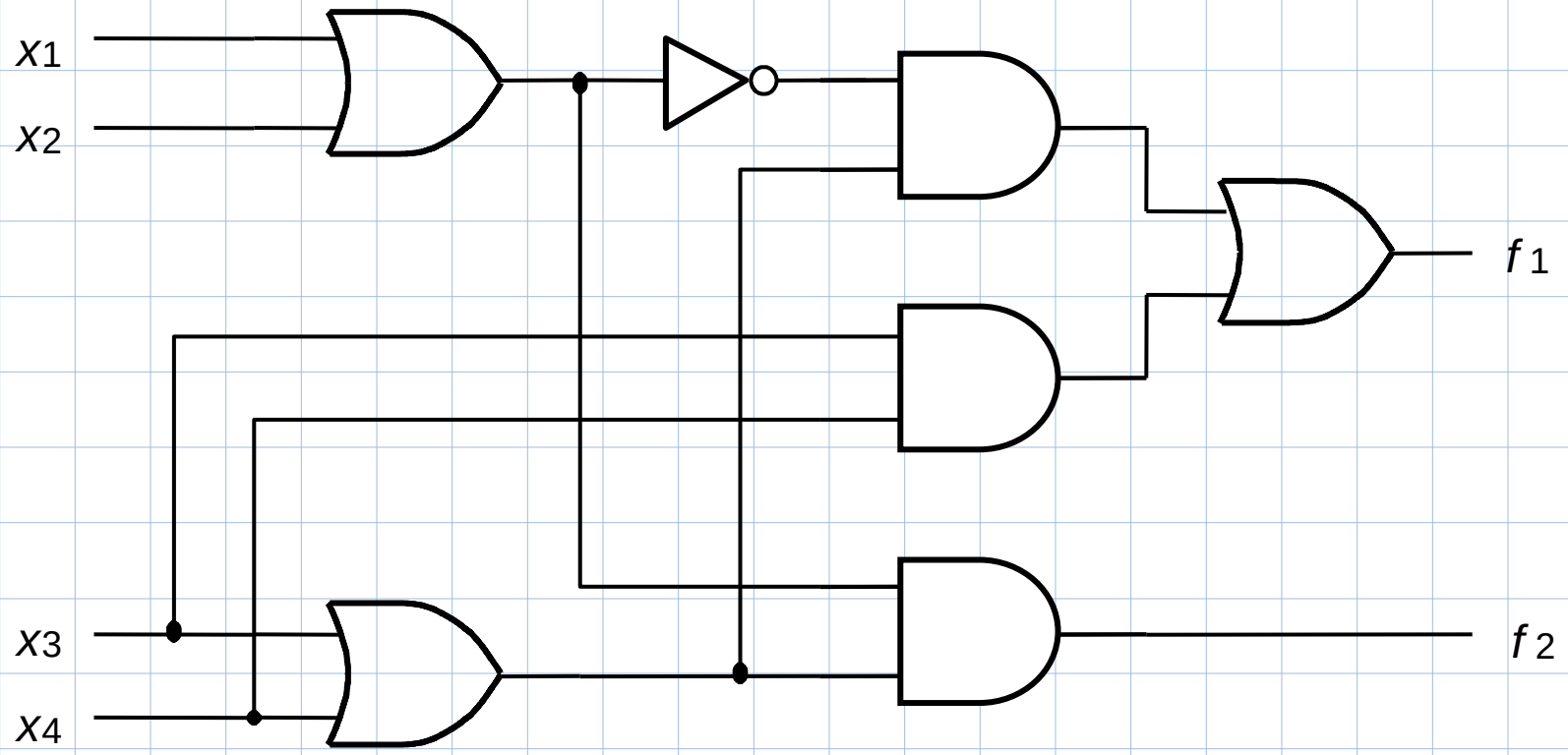
Redujimos el máximo fan-in a 2 – ahora sí cabe en el FPGA

Ejemplo – un circuito factorizado



Ejemplo – Circuito factorizado

No siempre es fácil encontrar factores comunes entre dos funciones que comparten las mismas variables

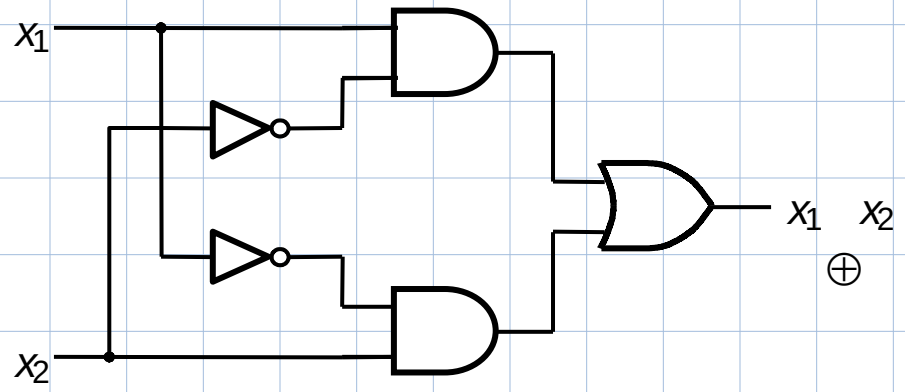


En circuitos complicados, una parte de la superficie o “bien raíz” del silicio está ocupada por wires o pistas entre compuertas lógicas. A continuación veremos una forma de reducir la complejidad de circuitos, incluyendo el ruteo de pistas.

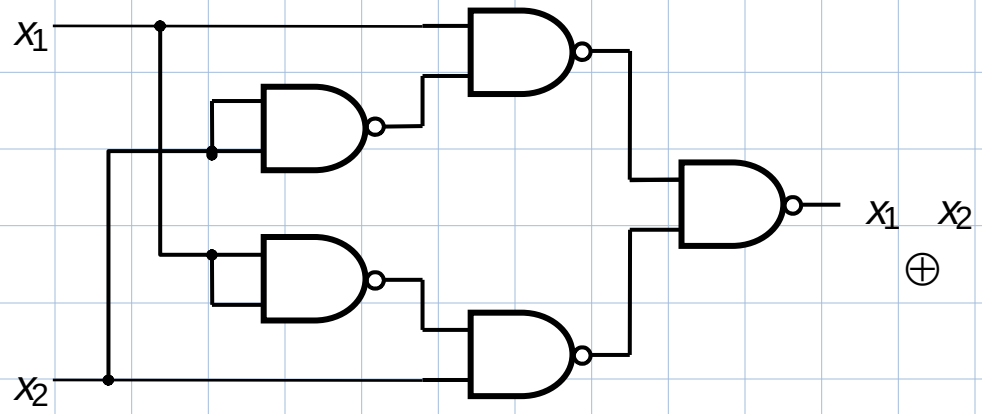
4.6.2 Descomposición funcional

- A veces es posible reducir la complejidad de un circuito descomponiéndolo en subcircuitos que aportan a varias funciones y que pueden ser empleados en varios lugares
- Algunos sintetizadores toman en cuenta detalles de la tecnología, como por ejemplo, ruteo de pistas, y optimiza de manera de reducir la complejidad global
- El CAD utiliza extensivamente esta técnica
- No entraremos en mucho detalle de cómo encontrar descomposiciones óptimas

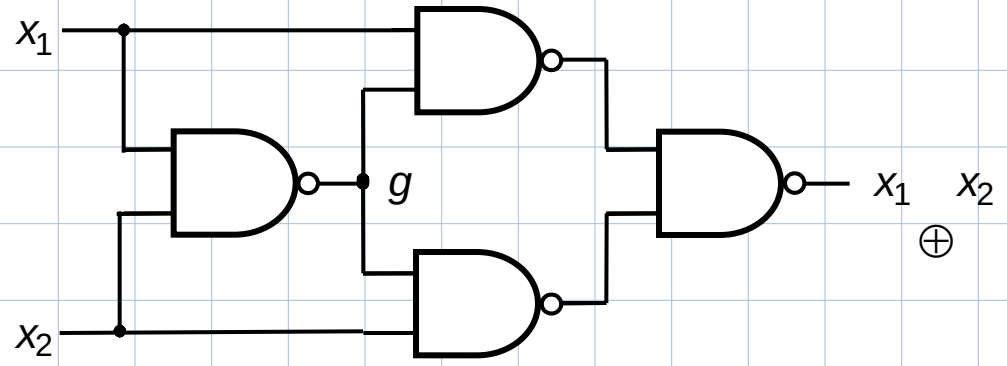
Descomposición funcional de un XOR



(a) Sum-of-products implementation



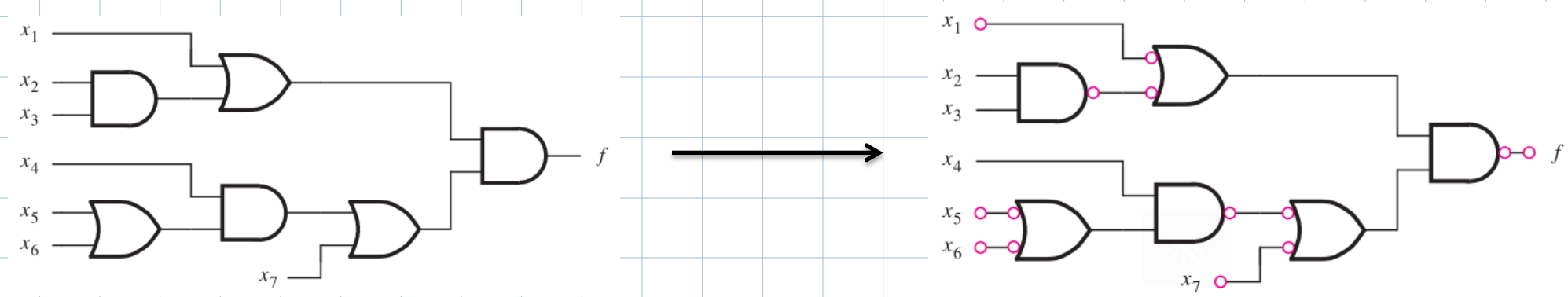
(b) NAND gate implementation



(c) Optimal NAND gate implementation

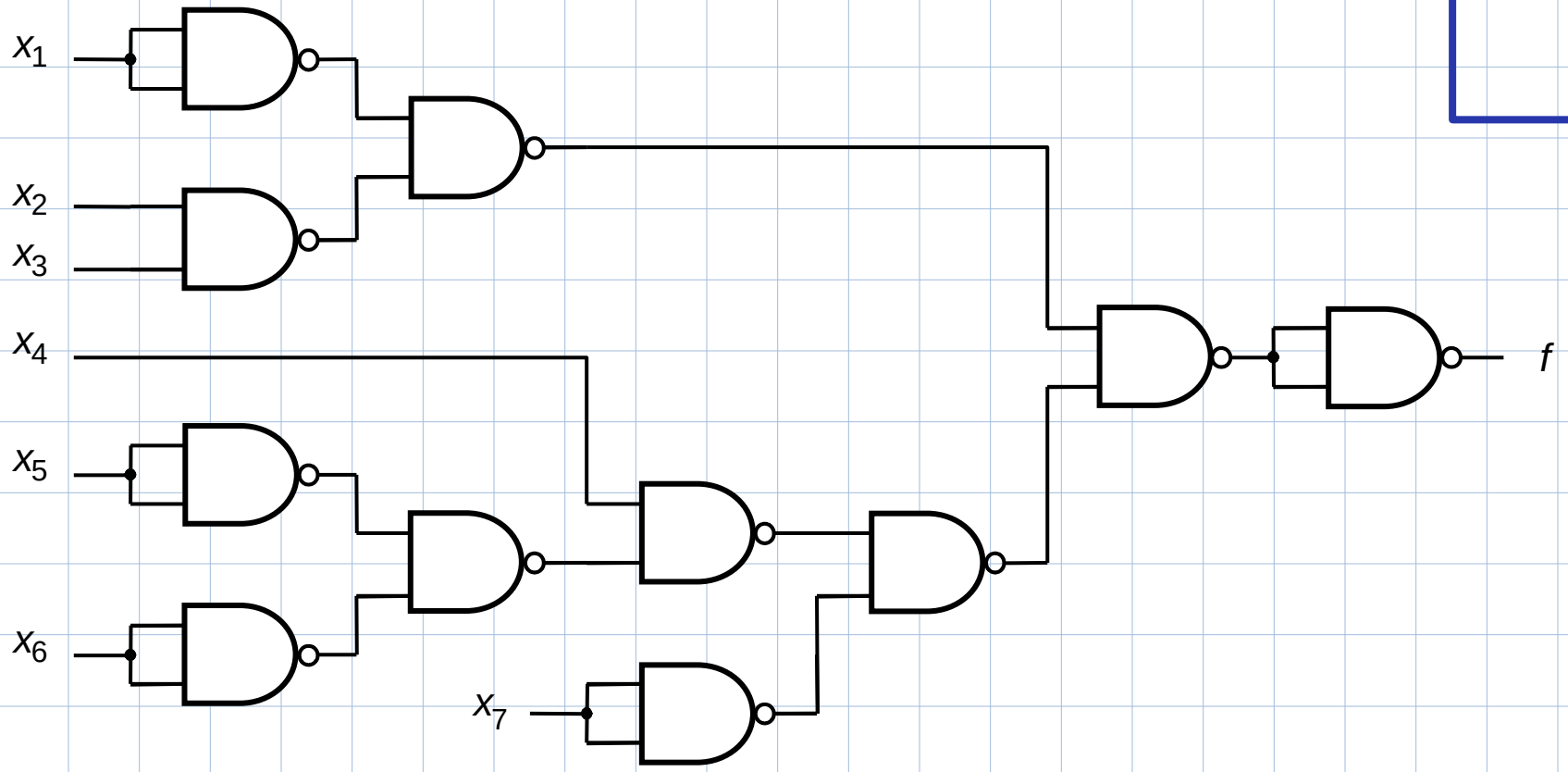
4.6.3 Circuitos con NAND y NOR multinivel

- En 2.7 aprendimos que los circuitos con AND-OR pueden ser implementados con NAND únicamente
- También aprendimos que los circuitos con OR-AND pueden ser implementados con puras compuertas NOR
- Este mismo concepto puede ser aplicado a circuitos multinivel



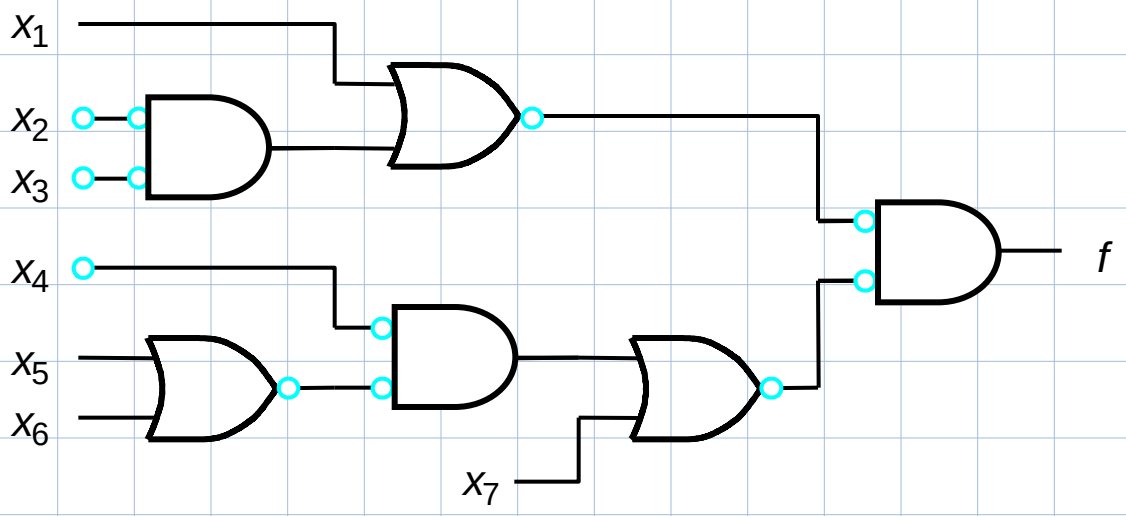
¿Cómo nos queda con puras NANDs?

4.6.3 Circuitos con NAND y NOR multinivel

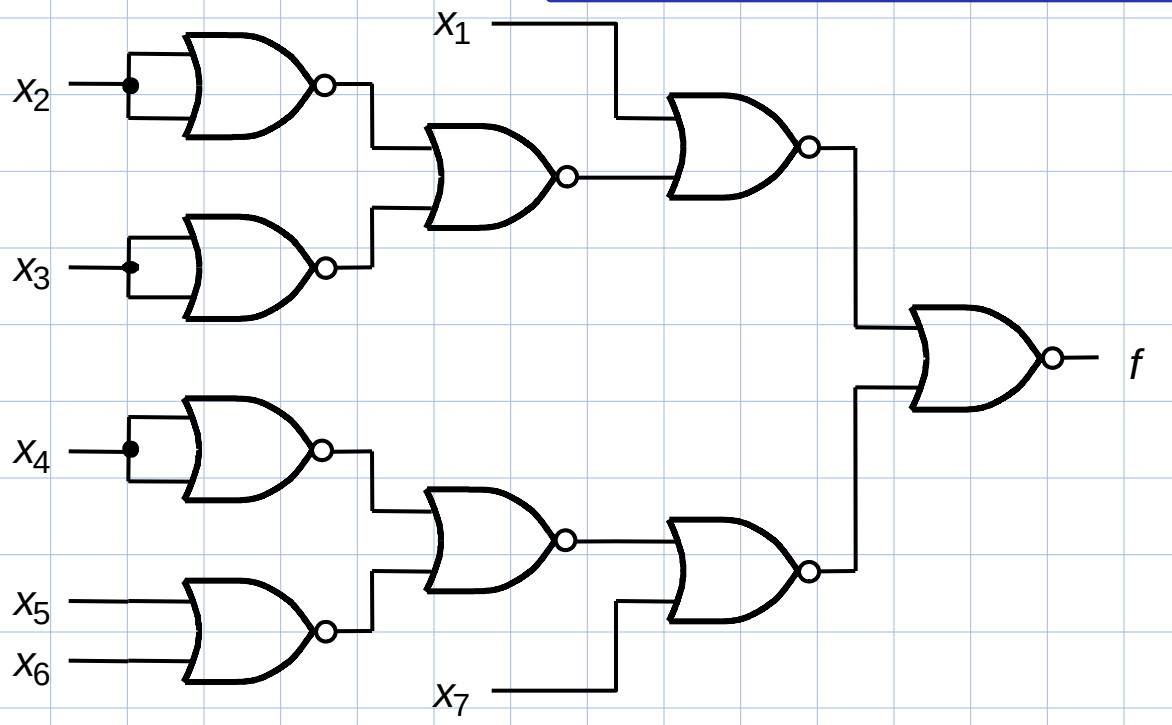


(c) NAND-gate circuit

4.6.3 Circuitos con NAND y NOR multinivel



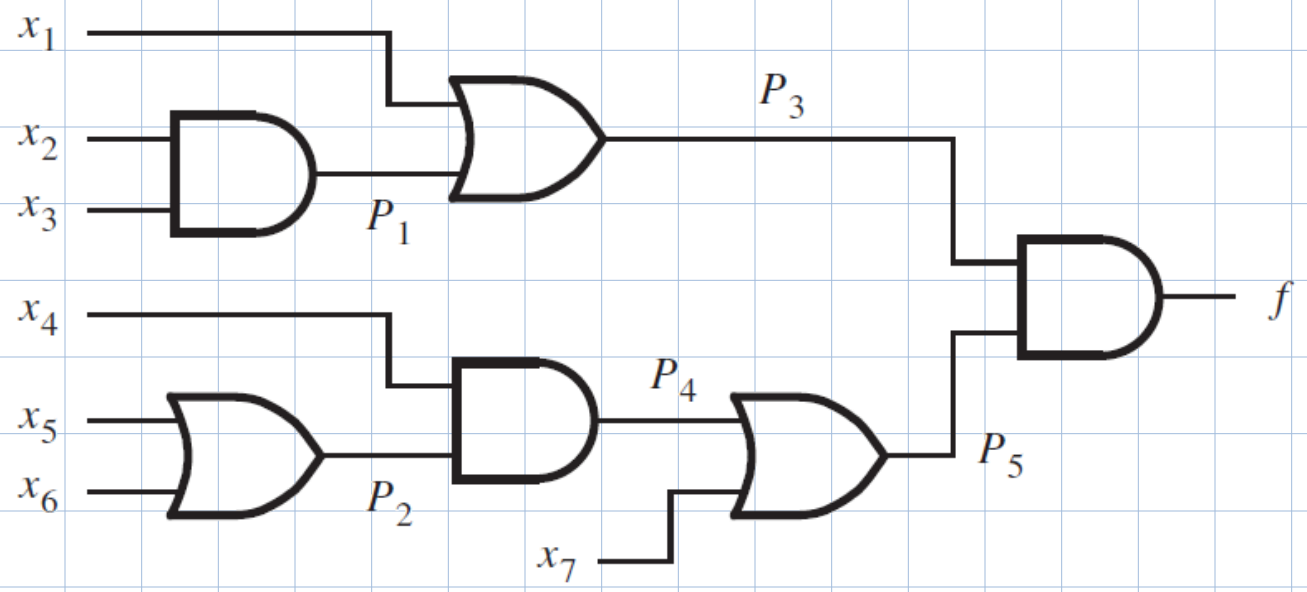
(a) Inversions needed to convert to NORs



(b) NOR-gate circuit

4.7 Análisis de circuitos multinivel

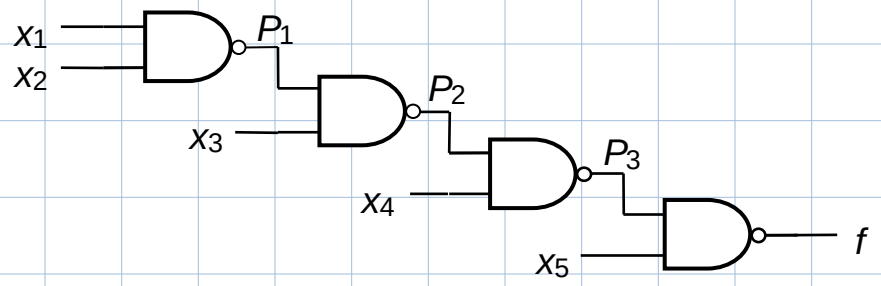
- No siempre es fácil escribir la función por inspección
- Ideal usar funciones de pasos intermedios



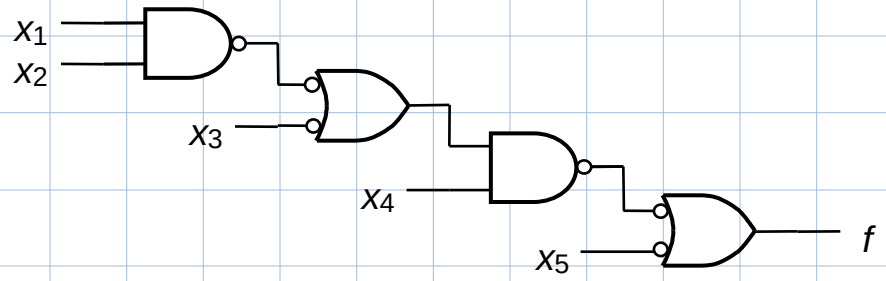
$$P_1 = x_2x_3$$
$$P_2 = x_5 + x_6$$
$$P_3 = x_1 + P_1 = x_1 + x_2x_3$$
$$P_4 = x_4P_2 = x_4(x_5 + x_6)$$
$$P_5 = P_4 + x_7 = x_4(x_5 + x_6) + x_7$$

Moviendo burbujas

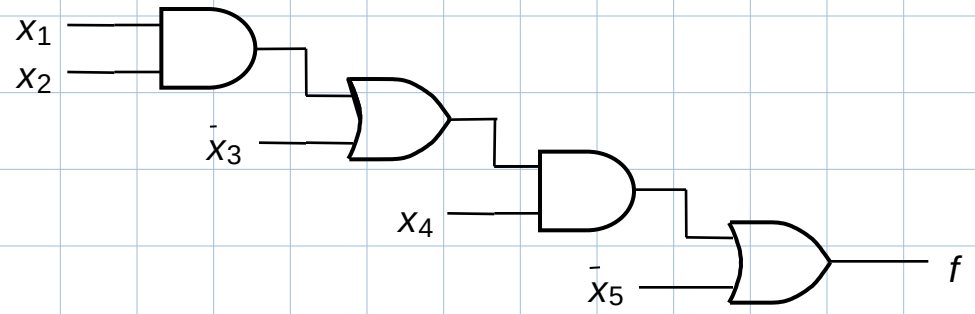
- Aplicando teorema de DeMorgan



(a) NAND-gate circuit



(b) Moving bubbles to convert to ANDs and ORs



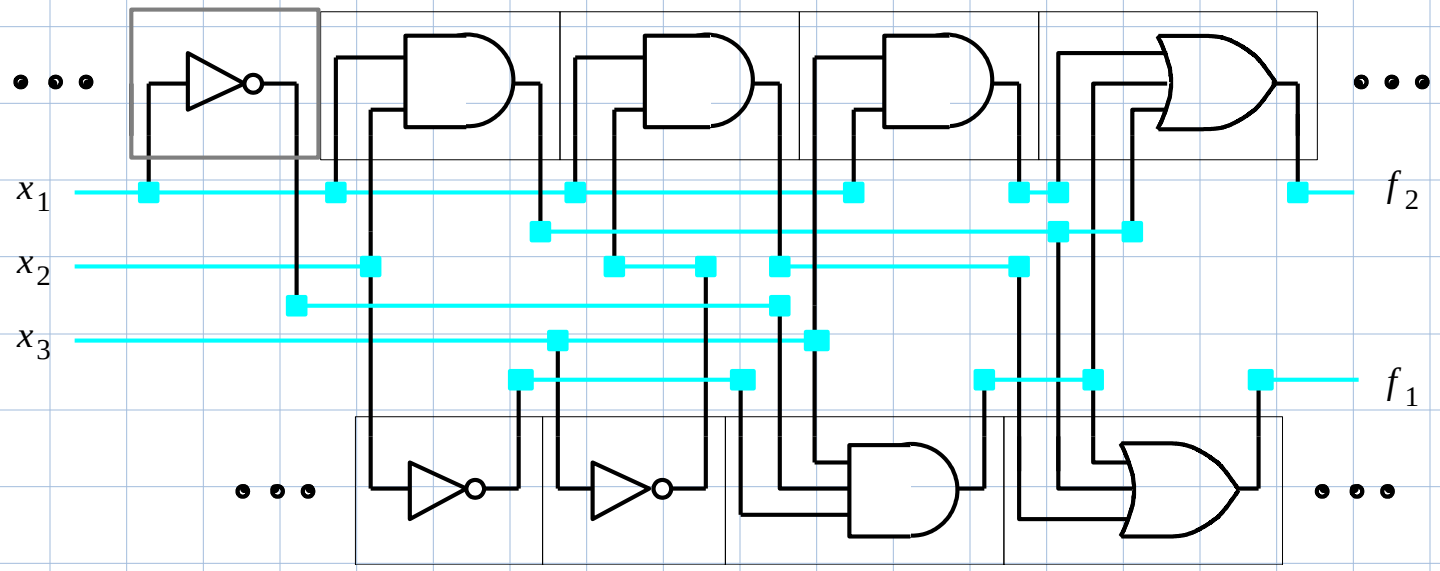
(c) Circuit with AND and OR gates

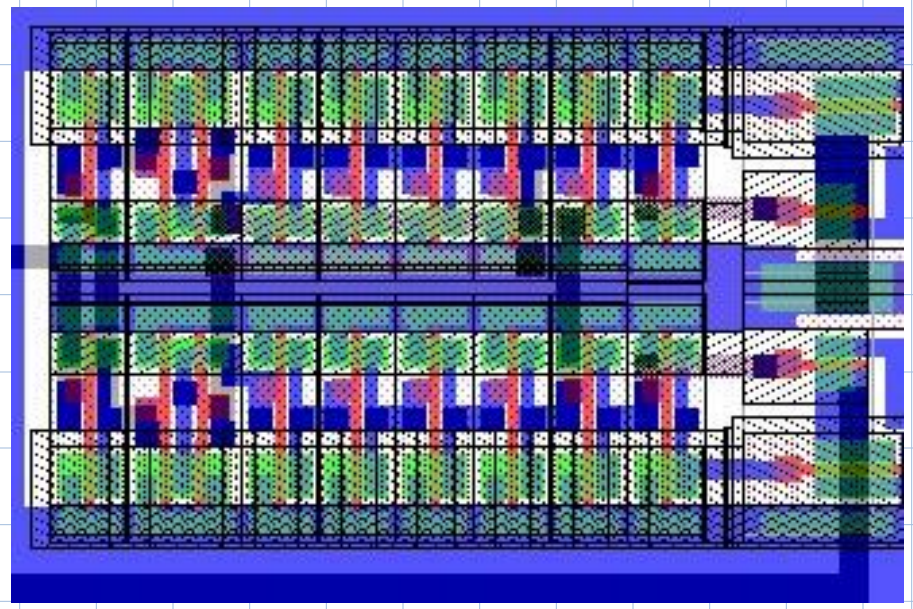
4.12 Herramientas de CAD

- Sirven para implementar circuitos aplicando automáticamente técnicas de síntesis
- Circuitos más pequeños se pueden sintetizar manualmente, pero los más complejos (como los microprocesadores) necesitan de estas herramientas

4.12.1 Síntesis y optimización lógica

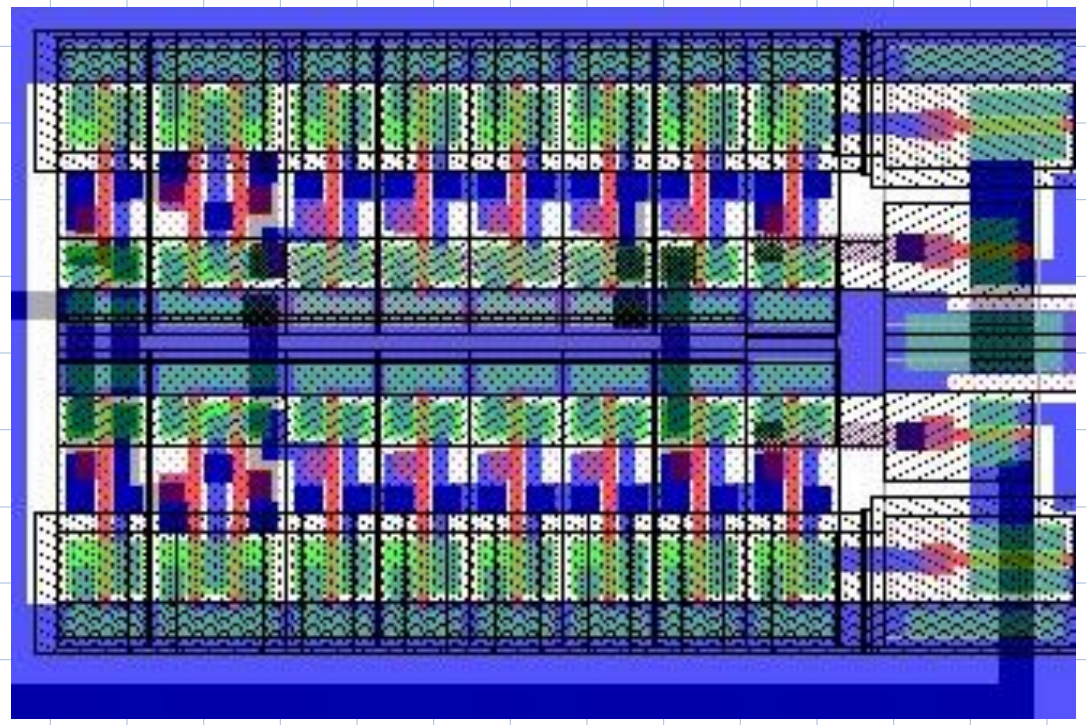
- La síntesis es el proceso por el cual se produce un circuito lógico en base a una especificación inicial (un diagrama esquemático o un código descriptivo)
- Dependiendo del tipo de tecnología utilizada, el ‘mejor’ circuito no necesariamente es el de menor costo





4.12.2 Diseño físico

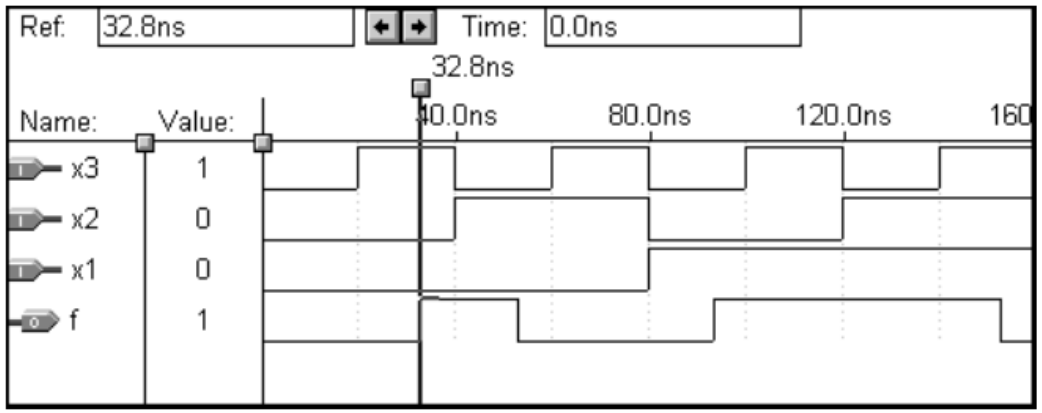
- Determina cómo implementar el circuito en un chip
- Declara la posición específica de elementos lógicos (floorplanning) y las conexiones (routing) entre ellos



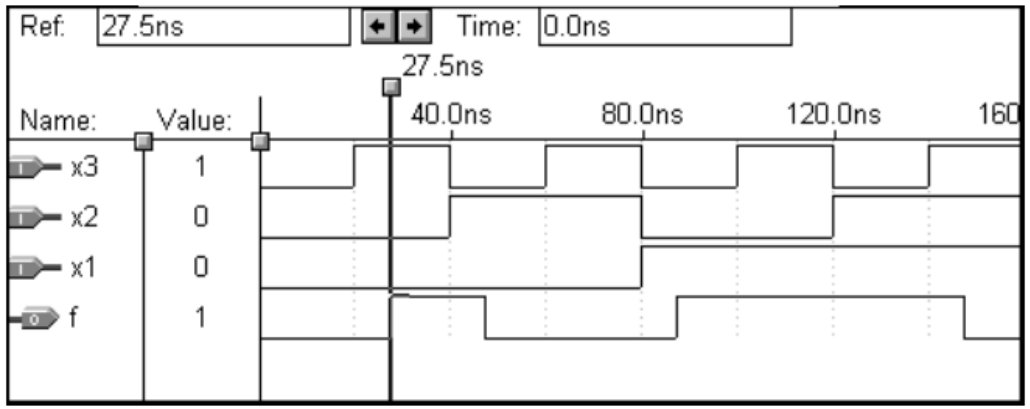
Ejemplo absurdamente simple (lo hice a mano)

4.12.3 Simulación de tiempos

- Todo circuito tiene un delay de propagación
- Hay delays de dos tipos: el que cada elemento tiene para generar un cambio entre su señal de entrada y la de salida y el delay que se produce en comunicar una señal de un elemento a otro
- Una simulación de tiempos permite ver si el circuito cumple con los requerimientos del tiempo esperado para el diseño
- CPLDs pueden ser más rápidos que FPGAs, pero en general sus velocidades son comparables

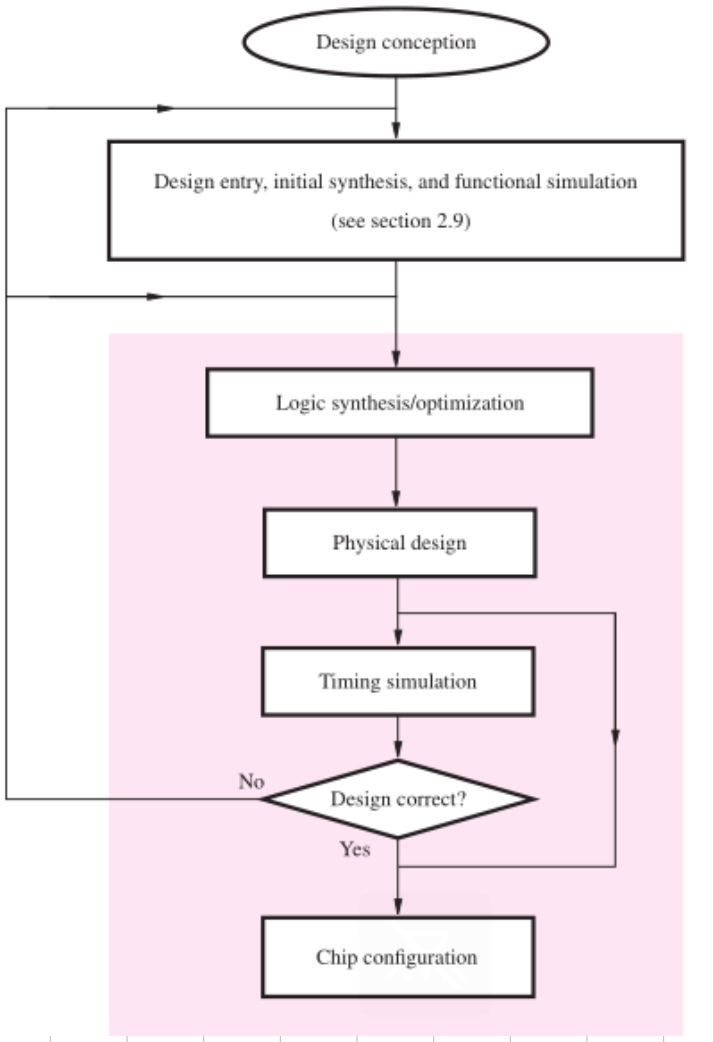


(a) Timing in an FPGA



(b) Timing in a CPLD

4.12.4 Resumen del flujo de diseño

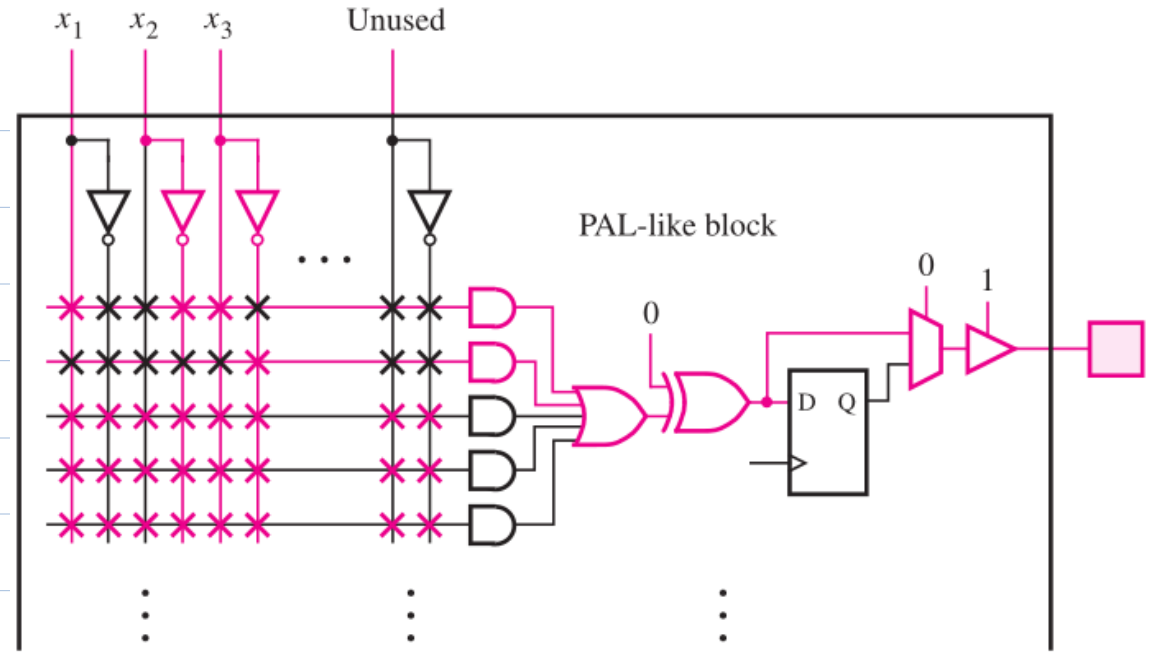


4.12.5 Ejemplos de circuitos sintetizados desde código Verilog

```
module example4_21 (x1, x2, x3, f);
    input x1, x2, x3;
    output f;

    assign f = (~x1 & ~x2 & ~x3) | (~x1 & x2 & ~x3) |
               (x1 & ~x2 & ~x3) | (x1 & ~x2 & x3) | (x1 & x2 & ~x3);

endmodule
```



Mismo circuito con síntesis XOR

- Algunas implementaciones pueden resultar mucho más eficientes cuando usamos XOR

