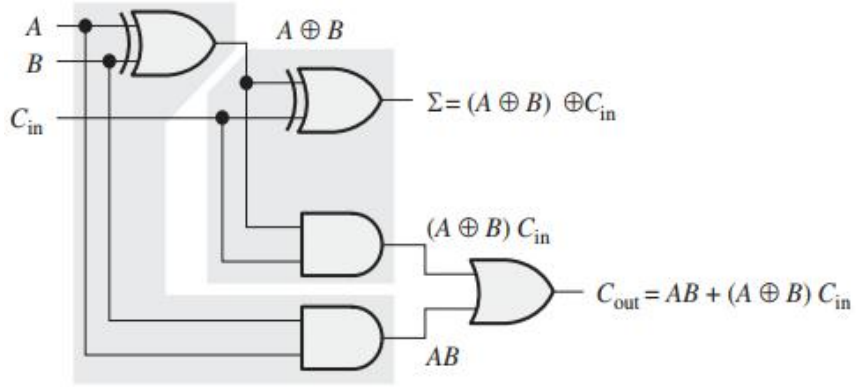
Aclaraciones y “pistas” para la resolución del Boletín 2.

Juan J Pombo

Mayo 2022

Circuito restador para 2 números A y B (6 bits)

1) Antes de nada debemos entender para qué usamos el complemento a 2 en aritmética binaria (revisar parte final del tema 4): usando el formato de representación Ca2 no necesitamos circuitos “restadores” porque todo se hace sumando (entender esto a fondo, MUY IMPORTANTE!)
2) Por otro lado sabemos lo que es el complemento a 2: hacer el complemento a 1 (invertir todos los bits) y sumar 1: ¡OTRA VEZ un SUMADOR!
3) ¿Qué es un sumador? ¿Cómo se implementa? (volvemos al tema 4)



Circuito restador para 2 números A y B (6 bits)

4) La forma de presentar el circuito en el apartado 3) para compactar usa puertas EXOR. ¿Pero una puerta exor se puede representar con puertas AND/OR/NOT?

A exor B = (A && ~B) || (~A && B)

5) ¿Cómo se conectan n-full adders para hacer un sumador de n-bits? Os remito a los apuntes del tema 4 de nuevo (ojo a los Cin y Cout)

Nota 1: El problema también se puede abordar como un circuito combinacional de 6 variables de entrada desde el principio pero así resultará muy largo y engorroso.

Circuito restador para 2 números A y B (6 bits)

Nota 2: Para la parte de Logisim: Obviamente, no es una solución en Logisim pillar el elemento sumador multibit que ya viene hecho. Eso no tiene “mérito”.

Nota 3: Algunos protestais porque el circuito es muy grande en Logisim. Me parece que habéis estudiado muy poco las posibilidades de Logisim.

Si no queréis pintar tantas veces el full-adder que hayáis diseñado (insisto, diseñado vosotros, no insertado el de la librería de Logisim), os remito, por ejemplo, a este tutorial para la parte de subcircuitos (hay infinidad de tutoriales en internet):

https://www.youtube.com/watch?v=OjM5gbOoP_c

Exercicio 4: Circuito secuencial para una secuencia de estados

`La secuencia es la siguiente : 000 → 010 → 111 → 100 → 011 → 101 → 000`

1) Se han resuelto varios diseños de circuitos secuenciales en clase. El circuito secuencial tiene dos “partes” comunes casi siempre:
   a) una serie de elementos de memoria (biestables, flip-flop’s) que mantienen los estados mientras no se cumplan las condiciones de cambio de estado.
   b) Un lógica combinacional que define las entradas a los elementos de memoria para que “cambien” de estado adecuadamente cuando los condiciones a las que se refería el apartado a) anterior se cumplen.
2) La lógica combinacional se puede implementar de múltiples formas: puertas lógicas de diversos tipos (el que más hemos usado), plas, ROMs y también aplicaciones realizadas con circuitos de Media Scala de Integración (MSI), como son los multiplexores (uno de los que nos da mas juego). De nuevo nos remitimos al tema 4, diapositiva 14. **La clave de este problema es que esta lógica combinacional se haga exclusivamente con MUX 4:1** (si hace falta un inversor en un momento dado se pone y listo).

Exercicio 4: Circuito secuencial para una secuencia de estados

Un MUX 4:1 junto con la función que implementa se muestra abajo. Y esto se parece mucho a transformar tablas de verdad en funciones lógicas. Ahí está el asunto. Con estos dispositivos se puede hacer CUALQUIER función, de la misma manera que se podía hacer cualquier función con puertas NAND.

$$Y = D_0\bar{S}_1\bar{S}_0 + D_1\bar{S}_1S_0 + D_2S_1\bar{S}_0 + D_3S_1S_0$$