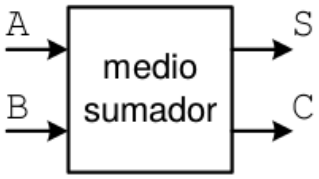
Introducción al Diseño Lógico 2025
Guía de Trabajos Prácticos Nº4: Circuitos Aritméticos

Nota: Considere que los tiempos de propagación de todas las compuertas son de 10 ns.

Ejercicio Nº01

Diseñe el circuito correspondiente a un HA (medio sumador, o half-adder), el cual suma dos bits.

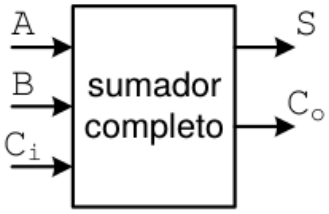


a) Defina y explique con sus palabras las entradas y las salidas del circuito.
b) Haga la tabla de verdad de las salidas del HA.
c) Sintetice las funciones lógicas de las salidas.
d) Dibuje el circuito.
e) ¿Cuántos tiempos de compuerta tendrá el retardo de cada una de las dos salidas?

Ejercicio Nº02

Para sumar números de más de un bit es necesario agregar una entrada adicional para el acarreo. Al circuito resultante se le llama *sumador completo* (SC) o *full-adder*(FA).

Haga un FA a partir de dos HAs, siguiendo estos pasos:



a) Empiece por convencerse de que un FA no es más que un sumador de tres bits. ¿Qué valores puede tomar el resultado? ¿cuántos acarreos pueden producirse?
b) En función de la cantidad de bits en 1 que haya entre las tres entradas ¿Cuándo debe valer 1 la salida S? ¿Cuándo se produce acarreo en la salida Co?
c) Diseñe un FA a partir de dos HAs y cualquier lógica adicional que necesite. Dibuje el circuito.
d) ¿Cuántos tiempos de compuerta tendrá el retardo de cada una de las dos salidas?

Ejercicio Nº03

Aunque la implementación del FA del ejercicio anterior es útil conceptualmente, en la práctica se utilizan otras implementaciones que tienen tiempos de propagación menores.

Implemente un FA directamente a partir de las tablas de verdad de sus salidas.

a) Haga la tabla de verdad de cada una de las salidas del FA.
b) Sintetice las funciones lógicas de las salidas. Escriba la salida S en función de operaciones XOR.
c) Dibuje el circuito resultante.
d) ¿Cuántos tiempos de compuerta tendrá el retardo de cada una de las salidas en esta versión del FA?

Ejercicio Nº04

Implemente los sumadores de los Ejercicios 1 y 3 en Quartus II, y simule para comprobar su funcionamiento.

Una vez que esté seguro de su funcionamiento cree un nuevo componente de librería para cada uno.
Usaremos estos componentes para ahorrarnos trabajo en las simulaciones siguientes.

Ejercicio Nº05

Diseñe utilizando módulos FA un sumador de tipo Ripple-Carry de dos números de cuatro bits. Las entradas son los números X e Y a sumar, y el carry de entrada del bit de menor orden C_0.
Las salidas son el resultado Z y el carry de salida C_4.

a) Dibuje el circuito.
b) Asumiendo que los FAs están construidos como el que usted diseñó en el Ejercicio 3, ¿cuál es el tiempo de propagación de este sumador de 4 bits en tiempos de compuerta? ¿En qué condición ocurre este tiempo máximo?
c) ¿Qué utilidad tienen la entrada C_0 y la salida C_4? Muestre como diseñaría un sumador de 16 bits a partir de cuatro sumadores de 4 bits como el que acaba de crear.
d) Implemente y simule en Quartus II el sumador de cuatro bits. Una vez que haya verificado el funcionamiento del diseño cree un nuevo componente para reutilizar este sumador de 4 bits más adelante.
e) Utilice el componente recién creado para simular un sumador de 16 bits y pruebe su funcionamiento mediante juegos de entradas adecuados.

Ejercicio Nº06

Un diseño alternativo del sumador que mejora los tiempos de propagación es el *Carry-Look-Ahead*. Para cada par de bits X_n y Y_n se definen dos funciones:

G_n: *(generate)* Vale 1 si X_n e Y_n generan un acarreo independientemente de si reciben un acarreo de los bits inferiores.
P_n: *(propagate)* Vale 1 si X_n e Y_n son tales que propagan un acarreo recibido recibido de los bits inferiores (esto es, generan acarreo de salida cuando hay acarreo de entrada).

Entonces el carry que entra a la etapa *i+1* es

$$C_{i+1} = G_i + P_i\, C_i$$

Desarrollando esta expresión hasta llegar a C_0 (el carry de entrada al sumador) se obtiene la función lógica de cada carry de entrada de los FA del sumador *Carry-Look-Ahead*. Por ejemplo:

$$C_2 = G_1 + P_1C_1 = G_1 + P_1(G_0 + P_0\, C_0) = G_1 + P_1G_0 + P_1P_0C_0$$

a) Explique con sus palabras cómo funcionan los circuitos de *Carry-Look-Ahead* y relaciónelo con la regla de generación anterior.
b) Determine a partir de las definiciones anteriores las funciones lógicas de G_n y P_n en función de X_n y Y_n.
c) Diseñe un sumador de tipo Carry-Look-Ahead de 4 bits, a partir de cuatro FAs y la lógica de generación de acarreos adelantados. Genere las funciones lógicas de cada una de los

acarreos de entrada a los FAs usando la regla anterior. Escríbalas en forma SOP.

d) ¿Cuántos tiempos de compuerta es el tiempo de propagación de este sumador de 4 bits?
e) ¿Qué desventaja tiene este diseño frente al sumador Ripple-Carry?
f) Implemente y simule en Quartus II. Una vez verificado el diseño cree un nuevo componente para el sumador de 4 bits.

Ejercicio Nº07

Diseñe un sumador/restador de 4 bits. El circuito tiene dos entradas $X_3X_2X_1X_0$ y $Y_3Y_2Y_1Y_0$, y una entrada de control $\overline{SUMA}$ que indica la operación ($\overline{SUMA}$ = 0 para una suma X+Y, $\overline{SUMA}$ = *1* para una resta X-Y). Haga el circuito a partir de un bloque sumador de 4 bits y la lógica auxiliar que considere conveniente.
Implemente y simule en Quartus II el circuito utilizando alguno de los sumadores de 4 bits que agregó a la librería en los ejercicios anteriores. Una vez verificado el diseño cree un nuevo componente para el sumador/restador de 4 bits.
Nota: Recuerde que puede hacer una resta sumando el primer número con el inverso en complemento-2 del segundo.

Ejercicio Nº08

En un circuito sumador de números con codificación complemento-2 ocurre un desborde (*overflow*) cuando el resultado de la operación cae fuera del rango de representación en el sistema.

a) Diseñe un circuito que permita obtener la bandera (*flag*) de desborde OVERFLOW a partir de los valores de los bits de signo de los operandos y del resultado.
b) Otro mecanismo alternativo de detección que no requiere mirar el resultado es mirar los acarreos del último y del anteúltimo bit de la suma. Diseñe un circuito que permita obtener la bandera de desborde a partir de los bits de acarreo (*carry*).

Ejercicio Nº09

Analice el funcionamiento del circuito que se muestra en la Figura 1. Considere que A_H y B_H corresponden al *nibble* -un *nibble* es un conjunto de 4 bits- alto (x_7 x_6 x_5 x_4) y que A_L y B_L corresponden al nibble bajo (x_3 x_2 x_1 x_0) de sendos números de 8 bits A y B. Suponga que el tiempo de propagación de los multiplexores es de 10ns, mientras que los sumadores de 4 bits -por su mayor complejidad- tardan 50ns en propagar los cambios de sus entradas a sus salidas. Considere que la señal C_{LO} controla ambos multiplexores.

a) Determine a qué corresponden las salidas X, Y y Z producidas por este circuito. ¿Cómo implementaría la misma funcionalidad con sólo dos sumadores de cuatro bits?
b) ¿Qué implementación será más rápida, la de la figura o la alternativa que usted propuso? ¿Cuál se imagina que ocupará más área en silicio una vez llevada a un chip?

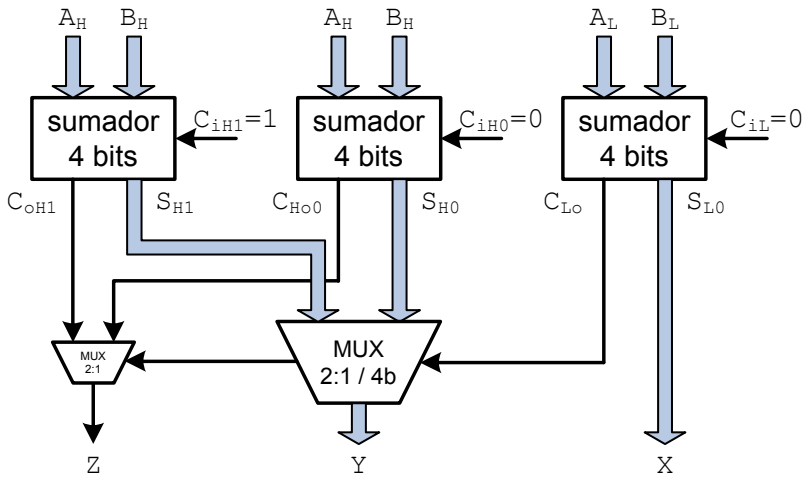


Figura 1

Ejercicio Nº10

Dibuje el diseño de un multiplicador convencional de números de 4 bits sin signo realizado utilizando FAs.

¿Cuál es el tiempo de propagación del multiplicador, medido en tiempos de compuerta, si en cada nivel de la multiplicación las sumas se hacen con un sumador *Ripple-Carry*? ¿Mejora el tiempo si se utilizan sumadores *Carry-Look-Ahead?* ¿A qué costo?

Implemente y simule en Quartus II. Utilice los módulos sumadores de 4 bits que ya creó en ejercicios anteriores.

Ejercicio Nº11

Se puede lograr una mejora en el tiempo de propagación del multiplicador utilizando un diseño de tipo *carry-save*.

Dibuje el diseño de un multiplicador de números de 4 bits sin signo de este tipo utilizando FAs. ¿Cuál es el tiempo de propagación del multiplicador, medido en tiempos de compuerta? Explique con sus palabras de dónde proviene la mejora respecto del multiplicador del ejercicio anterior.

Implemente y simule en Quartus II.

Ejercicio Nº12

Diseñe un sumador/restador de 8 bits. El circuito tiene dos entradas $X_7X_6X_5X_4X_3X_2X_1X_0$ e $Y_7Y_6Y_5Y_4Y_3Y_2Y_1Y_0$, y una entrada de control $\overline{SUMA}$ que indica la operación ($\overline{SUMA}$ = 0 para una suma X+Y, $\overline{SUMA}$ = *1* para una resta X-Y).
Las salidas serán el resultado de la suma y también una salida de Carry (C), una de overflow (V) y una salida Zero (Z), que se activa cuando todos los bits del resultado son ceros.
Implemente y simule en Quartus II, puede utilizar alguno de los sumadores de 4 bits que agregó a la librería en los ejercicios

anteriores. Una vez verificado el diseño cree un nuevo componente para el sumador/restador de 8 bits.

Nota: Recuerde que puede hacer una resta sumando el primer número con el complemento-2 del segundo.