Funcionamiento de la ALU

La ALU puede programarse para que opere tanto en *activa en alta* como *activa en baja*.

En esta asignatura todos los circuitos los diseñamos y usamos considerando *activa en alta* porque nos parece la forma de trabajo más intuitiva.

La tabla de verdad de la ALU que aparece en el texto base *es correcta* y se corresponde con la que proporciona Texas Instruments y cuya hoja de característica se adjunta con este documento.

El problema está en que las tablas de verdad del funcionamiento que proporcionan los distintos fabricantes dependen de la *interpretación* de las señales, pero las señales de entrada y salida a la ALU son las que le entran y salen al circuito, sólo que se interpretan de una forma u otra dependiendo de si se considera activa en alta o en baja.

La tabla siguiente (proporcionada por el fabricante) muestra la forma en la que se pueden interpretar dichas señales en función de si se considera activa en alta o en baja.

PIN NUMBER	2	1	23	22	21	20	19	18	9	10	11	13	7	16	15	17
Active-low data (Table 1)	$\overline{A}_0$	$\overline{B}_0$	$\overline{A}_1$	$\overline{B}_1$	$\overline{A}_2$	$\overline{B}_2$	$\overline{A}_3$	$\overline{B}_3$	$\overline{F}_0$	$\overline{F}_1$	$\overline{F}_2$	$\overline{F}_3$	C_n	C_{n+4}	$\overline{P}$	$\overline{G}$
Active-high data (Table 2)	A_0	B_0	A_1	B_1	A_2	B_2	A_3	B_3	F_0	F_1	F_2	F_3	$\overline{C}_n$	$\overline{C}_{n+4}$	X	Y

Observen que las señales de activa en baja son negadas respecto de las de activa en alta.

Como hemos dicho anteriormente, nosotros vamos a trabajar en *activa en alta* tanto en los diseños como en las simulaciones. Por tanto, para activa en alta, tanto las palabras de entrada, A3 A2 A1 A0 y B3 B2 B1 B0, como las señales de control, S3 S2 S1 S0, M, $\overline{C_n}$, las interpretamos tal cual, o sea, son directamente (sin negar) los datos que salen de los generadores de pulsos y que entran en la ALU y que se corresponden con la cabecera de la tabla de verdad de la ALU para activa en alta. Estos serán los valores sobre los que deberemos operar a la hora de comprobar el funcionamiento de dicha ALU.

Observen que las únicas señales que aparecen negadas son los acarreos o arrastres de entrada y salida, $\overline{C_n}$ y $\overline{C_{n+4}}$.

El significado de $\overline{C_n}$ en la tabla de verdad (de activa en alta, la del texto) es:

$\overline{C_n}$ =H (sin Acarreo). Por tanto, su significado es $C_n = 0$. O sea, Acarreo = 0

$\overline{C_n}$ =L (con Acarreo). Por tanto, su significado es $C_n = 1$. O sea, Acarreo = 1

Si miramos la página 3-712 de las hojas de características adjuntas verán que esta señal está negada, esta es la razón por la que a este terminal lo *nombran* con $\overline{C_n}$. De nuevo, nosotros en el terminal marcado con $\overline{C_n}$ ponemos directamente la señal, sin negarla. Es decir, a la ALU no hay que añadirle ningún inversor. Se programa directamente con las señales tal cual y se interpretan también tal cual (salvo C_{n+4}, como veremos más adelante). Así, si queremos programar la ALU para que realice la función aritmética sin acarreo "*A PLUS B*" deberemos poner los siguientes valores a las señales de control:

S3 S2 S1 S0 = H L L H, M = L, terminal C_n (olvidaros del negado) = H.

Si por el contrario, queremos que realice la operación "*A PLUS B PLUS 1*" deberemos poner:

S3 S2 S1 S0 = H L L H, M = L, terminal C_n (olvidaros del negado) = L.

La señal de acarreo de salida, C_{n+4}, sale negada. Es decir, cuando no hay acarreo de salida en C_{n+4} se obtiene un "1" y cuando hay acarreo se obtiene "0". Aquí si que al comprobar el funcionamiento en el simulador hay que tener en cuenta que está negada.

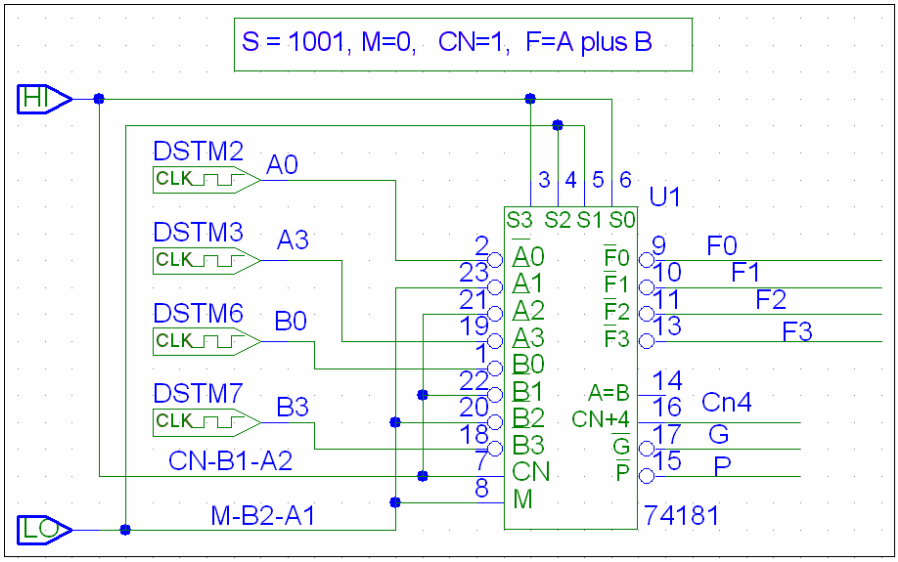
La ALU está diseñada para que se pueda ampliar su funcionamiento a palabras de más bits, permitiendo su conexión en cascada, uniendo directamente la salida $\overline{C_{n+4}}$ de la primera ALU (4 bits menos significativos) a la entrada $\overline{C_n}$ de la segunda (4 bits más significativos) y así sucesivamente cada vez que ampliemos, en 4 bits, las palabras sobre las que opera. Esto justifica el hecho de que sea $\overline{C_n}$ = H (sin acarreo) y $\overline{C_n}$ = L (con acarreo) ya que se va a unir directamente al $\overline{C_{n+4}}$ y cuyo significado es: $\overline{C_{n+4}}$ = H cuando no hay acarreo y $\overline{C_{n+4}}$ = L cuando hay acarreo. El significado se mantiene y no hay que introducir ningún inversor para su ampliación, sino que se une una ALU a la siguiente directamente.

Esta forma de conexión en cascada implica que tiene lugar un acarreo encadenado o enlazado de la misma forma que hemos visto en los sumadores. El problema que tiene este tipo de conexión es el retardo en la presentación del resultado final debido a que hasta que no se consiga el acarreo de la primera ALU, no se puede obtener el resultado de la segunda ALU y así sucesivamente.

Para subsanar este problema en la rapidez del cálculo, el circuito tiene la posibilidad de poderse usar con acarreo adelantado. Para ello, se usa la ALU junto con el circuito SN74182 que es un generador de acarreo adelantado del tipo del circuito de la figura 5.12 del texto. Para su conexión utiliza las salidas $\overline{P}$ y $\overline{G}$, siendo $\overline{P}$ = señal de Propagación del Acarreo y $\overline{G}$ = señal de Generación de Acarreo, al igual que consideramos en la página 277 del texto. Así, para ampliar la ALU a palabras de más bits, por ejemplo a 16 bits y con acarreo adelantado, se conectan las salidas $\overline{P}$ y $\overline{G}$ de cada una de las ALU (SN74181) a una de las parejas de entradas $\overline{P_i}$ y $\overline{G_i}$ del generador de acarreo adelantado (SN74182).

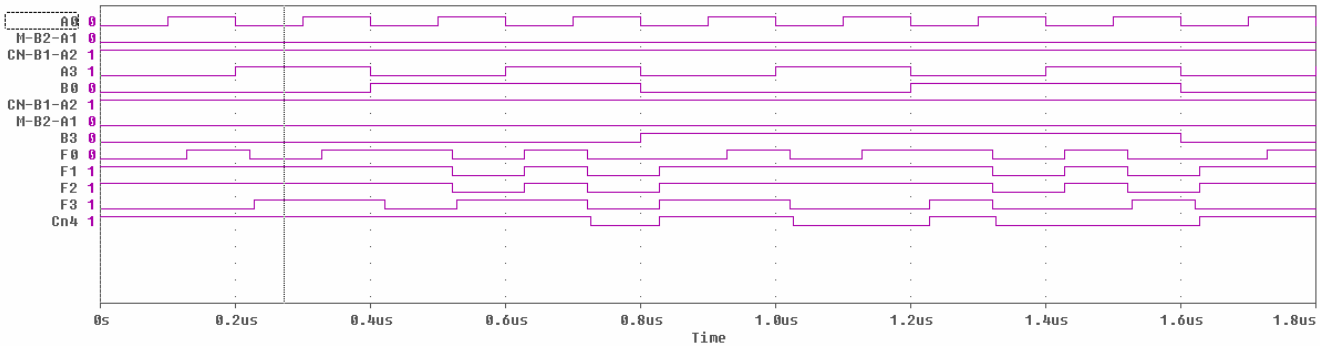
Para concretar y por si después de esta explicación queda alguna duda acerca de la interpretación de las señales de entrada y salida de la ALU, vamos a realizar el siguiente ejemplo:

Vamos a programar la ALU para que, según la Tabla de Verdad del texto, realice la función aritmética sin acarreo, A PLUS B. Los valores de las señales de entrada son las que se muestran en el siguiente circuito:



Con el fin de simplificar la tabla de verdad y en respuestas a la pregunta que habéis realizado respecto a este tema, os recomendamos que uséis relojes en dos de los bits de cada una de las palabras de entrada y que el resto de los bits los pongáis a un valor fijo de "0" ó "1". Así, para ver los valores que va tomando C_{n+4} y comprobar que realmente sale negado, una buena solución es usar los generadores de pulsos para los bits menos significativos y para los bits más significativos de ambas palabras, y los bits intermedios ponerlos fijos, uno a "0" y el otro a "1", por ejemplo. Con estos valores los resultados que se obtienen en el cronograma y, por tanto, en la tabla de verdad (sólo contiene 16 términos mínimos) son bastantes significativos como para comprobar que el circuito funciona correctamente.

El cronogrma que se obtiene es:



A continuación mostramos la tabla de verdad práctica obtenida a partir del cronograma y la teórica obtenida realizando la operación A PLUS B bits a bit (S0, S1, S2, S3) y teniendo en cuenta el acarreo de cada bit (C1, C2, C3, C4) en la suma del bit de orden superior:

Tablas de Verdad Práctica				**Tabla de Verdad Teórica**							
A3 A2 A1 A0	**B3 B2 B1 B0**	**F3 F2 F1 F0**	**Cn+4**	**C4**	**S3**	**C3**	**S2**	**C2**	**S1**	**C1**	**S0**
0100	0010	0110	1	0	**0**	0	**1**	0	**1**	0	**0**
0101	0010	0111	1	0	**0**	0	**1**	0	**1**	0	**1**
1100	0010	1110	1	0	**1**	0	**1**	0	**1**	0	**0**
1101	0010	1111	1	0	**1**	0	**1**	0	**1**	0	**1**
0100	0011	0111	1	0	**0**	0	**1**	0	**1**	0	**1**
0101	0011	1000	1	0	**1**	1	**0**	1	**0**	1	**0**
1100	0011	1111	1	0	**1**	0	**1**	0	**1**	0	**1**
1101	0011	0000	0	1	**0**	1	**0**	1	**0**	1	**0**
0100	1010	1110	1	0	**1**	0	**1**	0	**1**	0	**0**
0101	1010	1111	1	0	**1**	0	**1**	0	**1**	0	**1**
1100	1010	0110	0	1	**0**	0	**1**	0	**1**	0	**0**
1101	1010	0111	0	1	**0**	0	**1**	0	**1**	0	**1**
0100	1011	1111	1	0	**1**	0	**1**	0	**1**	0	**1**
0101	1011	0000	0	1	**0**	1	**0**	1	**0**	1	**0**
1100	1011	0111	0	1	**0**	0	**1**	0	**1**	0	**1**
1101	1011	1000	0	1	**1**	1	**0**	1	**0**	1	**0**
0100	0010	0110	1	0	**0**	0	**1**	0	**1**	0	**0**

Como podemos comprobar las columnas encabezadas por S3, S2, S1, S0, obtenidas sumando las palabras A y B, coinciden con la columna encabezada por F3 F2 F1 F0 obtenida del cronograma. El acarreo total, C4, coincide con la columna Cn+4 negada. Por tanto, como hemos podido comprobar el circuito funciona correctamente.