Fundamentos de Sistemas Digitales

Primera actividad evaluable: Diseño, Implementación, Simulación y Validación de un Circuito en Lógica Combinacional

xx de xxxx de 2016

Nombre y Apellidos: xxxx
DNI: xxxx
Centro asociado: xxxx

Nota del autor: este documento ha sido escrito por un alumno, no por un experto, por lo que el documento se ha subido para que pueda servir de guía a furturos estudiantes. No obstante, la nota obtenida fue 10/10.

Índice

1. Enunciado

A-E-1-007

Disponemos de una palabra de 4 bits $(P3, P2, P1, P0)$ y de una señal adicional x, con las que queremos controlar el funcionamiento de una Unidad Aritmética Lógica, pero la programación de la ALU no depende directamente de estas se ñales, sino de la paridad o no de la palabra junto con el valor "0"ó "1"de la variable x. Así, el criterio para controlar las operaciones que realiza la ALU sobre las 2 palabras de 4 bits $A(A3, A2, A1, A0)$ y $B(B3, B2, B1, B0)$ es la siguiente:

a) Si la palabra es par y $x = 1$, la ALU hace la operación aritmética sin acarreo $A\ PLUS\ A\overline{B}$ b) Si la palabra es par y $x = 0$, la ALU hace la operación lógica $\overline{A \oplus B}$

c) Si la palabra es impar y $x = 1$, la ALU hace la operación aritmética con arrastre $A\ PLUS\ B\ PLUS1$

d) Si la palabra es par y $x = 0$, entonces la ALU debe ponerse a 0.

Diseñe el circuito del codificador y úselo para controlar las operaciones de la ALU que se han especificado.

2. Solución

2.1. Planteamiento del problema

El funcionamiento de la ALU depende de la paridad de la palabra P y del valor de la variable x.

Decimos que la palabra $P(P3, P2, P1, P0)$ es par si tiene un número par de bits que son 1. Sea F la *función paridad* de esta palabra, que vale 0 cuando la palabra es par y 1 cuando no lo es, el funcionamiento de la ALU es el siguiente:

F	X	ALU
0	0	$\overline{A \oplus B}$
0	1	$A\ PLUS\ A\overline{B}$
1	0	Puesta a 0
1	1	$A\ PLUS\ B\ PLUS\ 1$

Cuadro 1: Operaciones a realizar por la ALU

Por tanto, nuestro proyecto consiste en el diseño de un circuito (circuito codificador) que, dados los datos iniciales (la palabra P y x), proporcionará a la ALU la sentencia de la operación a realizar según estas entadas.
Obviamente, este circuito codificador estará conectado a la ALU ($ALU\ SN74181$) en lógica positiva, que realizará la operación.
Las palabras A y B, de 4 bits, que son aquellas sobre las que la ALU opera, son datos de entrada. Puesto que las palabras son desconocidas, haremos una simulación (usando relojes digitales) con todos los posibles casos para, finalmente, obtener la tabla de verdad del circuito final.
Sin embargo, como veremos, el gran número de estos casos nos impide la obtención de la tabla de verdad por el considerable tiempo de trabajo que esto supondría.
Tenemos entonces la necesidad de realizar diferentes simulaciones parciales para poder afirmar, con mayor o menor seguridad, que el circuito funciona correctamente.

2.1.1. Consideraciones respecto al formato del texto

En el texto se muestran los circuitos y los cronogramas de los mismos, realizados los primeros y obtenidos los segundos, con el programa *PSpice* proporcionado por el equipo docente de la universidad.
En el título de las imágenes de los circuitos se indica, entre paréntesis, el nombre del archivo del circuito (con extensión *.sch*). Estos archivos se encuentran adjuntos al documento de texto.
Las imágenes de los cronogramas de sendos circuitos se incluyen en el texto pero no se adjuntan, pues pueden obtenerse directamente a partir de los circuitos.
Se intenta indicar en el texto todos los tiempos empleados en los relojes de los circuitos.
En cuanto a los elementos de los circuitos, se usan aquellos que aparecen en el libro de texto de la asignatura.

2.2. Función paridad, F

En la subsección anterior hemos explicado la función paridad, F, de la palabra P. Esta función se comporta del modo indicado en su tabla de verdad

P3	P2	P1	P0	P
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Cuadro 2: Función de paridad

Como comentario podemos decir que la función vale 1 ó 0 el mismo número de veces, 8, lo cual es evidente al haber cuatro bits y dos posibilidades para cada uno.
A partir de la tabla podemos obtener la expresión analítica de F en función de los bits de la palabra P. Para ello nos quedamos con las configuraciones cuya salida (F) es 1 (representación por minterms).

$$F = \overline{P3}P2\overline{P1}P0 + \overline{P3}P2P1\overline{P0} + \overline{P3}P2\overline{P1}\overline{P0} + \overline{P3}P2P1P0$$

$$+P3\overline{P2}\overline{P1}\overline{P0} + P3\overline{P2}P1P0 + P3P2\overline{P1}P0 + P3P2P1\overline{P0}$$

Si tenemos en cuenta la operación XOR (OR exclusivo), podemos escribir F como

$$F = \overline{P3}(\overline{P2}(\overline{P1}P0 + P1\overline{P0}) + P2(\overline{P1P0} + P1P0))+$$

$$+P3(\overline{P2}(\overline{P1P0} + P1P0) + P2(\overline{P1}P0) + P1\overline{P0})$$

$$= P3 \oplus P2 \oplus P1 \oplus P0$$

Donde hemos usado las propiedades asociativa, conmutativa y autoinversa del operador XOR.

Puesto que ya tenemos F escrita mediante XOR, no hemos buscado su forma mínima mediante el diagrama de *Veitch - Karnaugh*. De hecho, si lo hacemos, nos encontraremos con que no podemos asociar los minterms y, por tanto, la expresión ya es mínima.
Además, la expresión en términos de XOR nos permite representar a F con un circuito con tan sólo tres puertas XOR como se muestra en la *Figura 1*.

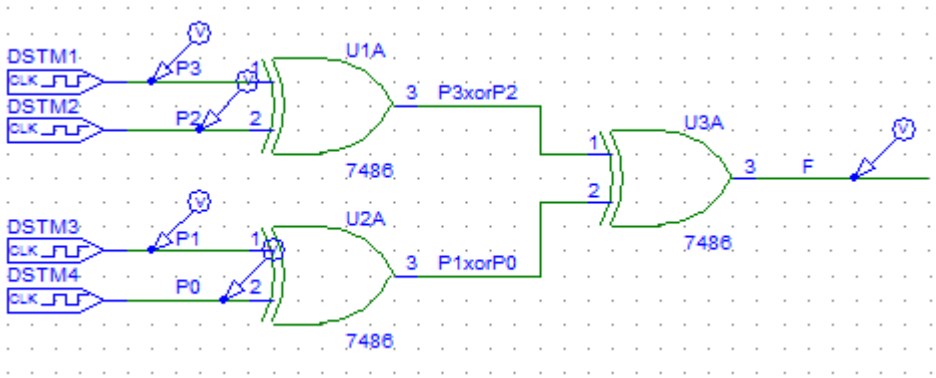

Figura 1: Circuito correspondiente a F (circuitoF.sch)

Notemos que una de las dos puertas iniciales tiene las entradas $P3$ y $P2$ y la otra $P1$ y $P0$. Pero por la asociatividad y conmutatividad podemos escoger cualquier combinación de entradas.
Los relojes digitales tienen los siguientes tiempos para DELAY:

	P3	P2	P1	P0
ONTIME	1us	2us	4us	8us
OFFTIME	1us	2us	4us	8us

Cuadro 3: Tiempos para el circuito que genera F

La simulación produce el siguiente cronograma que coincide con la tabla de verdad:

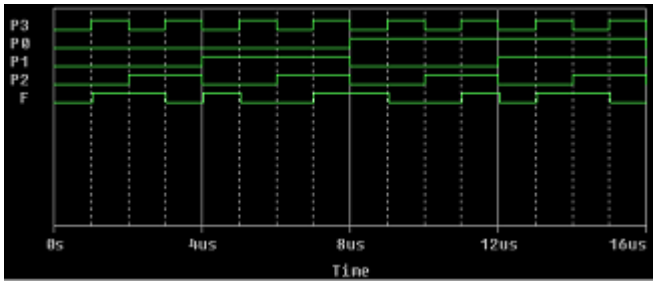

Figura 2: Cronograma del circuito

2.3. Las operaciones de la ALU

Recordemos que la ALU (SN74181) contiene las funciones lógicas y aritméticas más usadas. Acepta como operandos dos palabras de 4 bits y un acarreo, $\overline{C_n}$, para facilitar la conexión en cascada, en caso de necesitar operar sobre palabras de más de 4 bits.
La elección de la función depende de los valores (en alta, H, o en baja, L) de las entradas M, $\overline{C_n}$, $S3$, $S2$, $S1$ y $S0$. La entrada $\overline{C_n}$ sólo afecta en la elección cuando M está en baja, $M = L$ (usaremos lógica positiva).

SELECCIÓN	DATO ACTIVO EN ALTA		
	M=H	M=L Operaciones Aritméticas	
S_3 S_2 S_1 S_0	Funciones Lógicas	$\overline{C_n}$=H (sin acarreo)	$\overline{C_n}$=L (con acarreo)
L L L L	$F = \overline{A}$	$F = A$	$F = A$ PLUS 1
L L L H	$F = \overline{A+B}$	$F = A + B$	$F = (A + B)$ PLUS 1
L L H L	$F = \overline{A}B$	$F = A + \overline{B}$	$F = (A + \overline{B})$ PLUS 1
L L H H	$F = 0$	$F =$ MINUS 1 (comp. a 2)	$F = Cero$
L H L L	$F = \overline{AB}$	$F = A$ PLUS $A\overline{B}$	$F = A$ PLUS $A\overline{B}$ PLUS 1
L H L H	$F = \overline{B}$	$F = (A + B)$ PLUS $A\overline{B}$	$F = (A + B)$ PLUS $A\overline{B}$ PLUS 1
L H H L	$F = A \oplus B$	$F = A$ MINUS B MINUS 1	$F = A$ MINUS B
L H H H	$F = A\overline{B}$	$F = A\overline{B}$ MINUS 1	$F = A\overline{B}$
H L L L	$F = \overline{A} + B$	$F = A$ PLUS AB	$F = A$ PLUS AB PLUS 1
H L L H	$F = \overline{A \oplus B}$	$F = A$ PLUS B	$F = A$ PLUS B PLUS 1
H L H L	$F = B$	$F = (A + \overline{B})$ PLUS AB	$F = (A + \overline{B})$ PLUS AB PLUS 1
H L H H	$F = AB$	$F = AB$ MINUS 1	$F = AB$
H H L L	$F = 1$	$F = A$ PLUS A	$F = A$ PLUS A PLUS 1
H H L H	$F = A + \overline{B}$	$F = (A + B)$ PLUS A	$F = (A + B)$ PLUS A PLUS 1
H H H L	$F = A + B$	$F = (A + \overline{B})$ PLUS A	$F = (A + \overline{B})$ PLUS A PLUS 1
H H H H	$F = A$	$F = A$ MINUS 1	$F = A$

Figura 3: Tabla de verdad en lógica positiva de ALU SN74181

De la tabla de la *Figura 2* obtenemos las combinaciones de entrada que nos atañen en nuestro problema. Escribmos la siguiente tabla a partir de la del *Cuadro 1* y la de la *Figura 3*.

F	X	ALU	S3	S2	S1	S0	M	$\overline{C_n}$
0	0	$\overline{A \oplus B}$	H	L	L	H	H	X
0	1	$A\ PLUS\ A\overline{B}$	L	H	L	L	L	H
1	0	Puesta a 0	L	L	H	H	H	X
1	1	$A\ PLUS\ B\ PLUS\ 1$	H	L	L	H	L	L

Cuadro 4: Operaciones a realizar por la ALU y sus entradas

En las filas en las que $M = H$ hemos escrito $\overline{C_n} = X$ ya que su valor no se tiene en cuenta.

2.4. Preconexión del circuito codificador con la ALU

El circuito codificador nos proporciona la salida F a partir de la palabra P. Esta salida, ha de combinarse con la entrada X formando la combinación (F, X) que indica la operación a realizar por la ALU.
Pero para indicar a la ALU que realice la dicha operación se necesitan las entradas $S3$, $S2$, $S1$, $S0$, M y $\overline{C_n}$ que han de obtenerse, evidentemente, a partir de la combinación (F, X). Por tanto, tenemos que relacionar estas entradas de algún modo con la combinación obtenida.
Si nos fijamos en el *Cuadro 4* podemos deducir que

$$M = \overline{X}$$

$$S0 = \overline{S2}$$

$$S3 = \overline{F \oplus X}$$

Necesitamos relacionar todavía $S0$ y $S1$. Para ello, obtenemos sus expresiones en minterms a partir del *Cuadro 4* considerando $H = 1$ y $L = 0$.
Obtenemos

$$S0 = \overline{F}\,\overline{X} + F\overline{X} + FX$$

$$S1 = F\overline{X}$$

Finalmente, como el valor de $\overline{C_n}$ no importa cuando $M = H$, podemos escribir

$$\overline{C_n} = S2$$

El circuito resultante queda

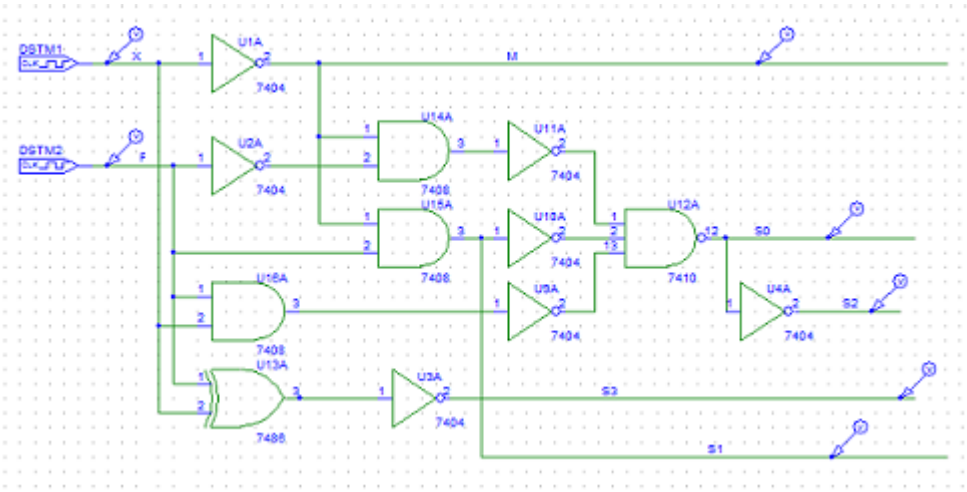

Figura 4: Circuito de preconexión (*preALU.sch*)

Los relojes digitales tienen tienen los tiempos para DELAY:

	X	F
ONTIME	1us	2us
OFFTIME	1us	2us

Cuadro 5: Tiempos para el circuito *preALU.sch*

El diagrama correspondiente es el siguiente, que coincide con su tabla de verdad

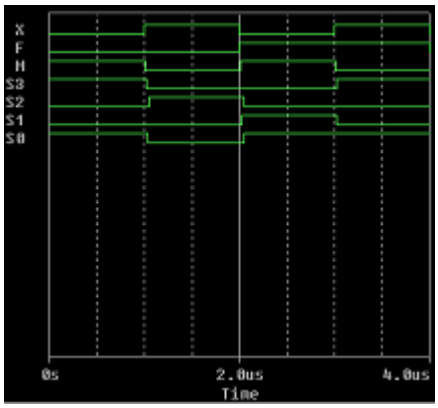

Figura 5: Simulación del circuito de preconexión

2.5. Conexión del circuito codificador con la ALU: Circuito Final

Recordemos que hasta ahora, hemos diseñado dos circuitos: el primero proporciona la paridad de la palabra P (bloque A) y el segundo, proporciona las entradas de la ALU (las que determinan la operación a realizar) mediante la combinación de las configuraciones de la palabra P y la variable x (bloque B). Estos dos circuitos los unimos dando lugar al que llamamos circuito codificador (bloque C). Y éste, a su vez, lo conectamos a la ALU.

Además, la ALU necesita otras entradas: las palabras A y B sobre las que realizará la operación seleccionada. Como ya hemos comentado en *2.1. Planteamiento del problema*, puesto que no conocemos A y B, de 4 bits, usaremos cuatro relojes digitales que nos generen los pulsos, uno el doble del anterior, para considerar todas las configuraciones posibles (bloque D).
Así pues, el circuito final queda como

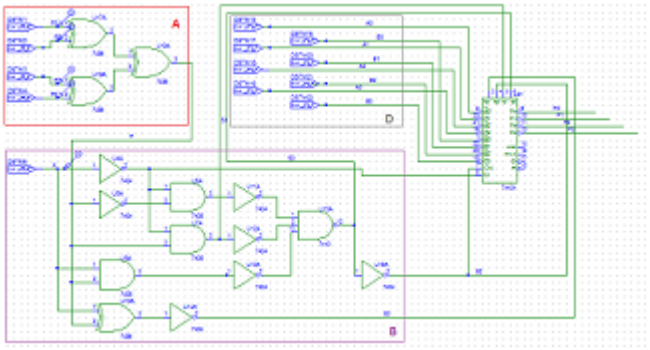

Figura 6: Circuito Final (*CIRCUITO.sch*)

No simulamos el circuito porque debido al gran número de configuraciones posibles, PSpice no nos proporciona el cronograma entero. Más adelante comprobaremos el funcionamiento del circuito por partes.

2.6. Consideraciones sobre el Circuito Final

Para ajustar los tiempos en los relojes del circuito final tenemos que tener en cuenta el gran número de posibles configuraciones de entradas. Recordemos que las palabras son de 4 bits y x es un bit.
Por un lado, la palabra P presenta $2^4 = 16$ configuraciones posibles que junto que las dos de x suman 32 combinaciones.
Por otro lado, las posibles combinaciones con las palabras A y B suman un total de $2^4 \cdot 2^4 = 256$.

Como cada combinación de palabras A y B se combina a su vez con una P y una x, se obtinen un total de $32 \cdot 256 = 8192$ casos.

Por esta razón, no vamos a simular el circuito con tantas señales (el cronograma se corta). Lo que haremos es fijar algunas de ellas para acotar el número de casos.

2.7. Circuitos de pruebas

Vamos a realizar diferentes simulaciones cambiando, en cada caso, diferentes variables de entrada. El circuito es siempre el mismo, pero con pequeñas modificaciones que no alteran la esencia del mismo. Intercambiamos algunos relojes digitales por *digital HI / LO ports* para generar las señales especifícas que se desean en cada caso. También es necesario cambiar algunos tiempos de los relojes para su generación. Una vez realizada la simulación, usamos la opción *Toggle Coursor* del *PSpice* que nos muestra los valores de cada marcador (*Level Marker*) en cualquier punto que se seleccione en el cronograma correspondiente. Debido la gran cantidad de posibles configuraciones de entrada, analizamos en cada caso un pequeño número de éstas de forma aleatoria (en la medida que sea posible), seleccionando de entre éstas al menos una configuración para cada operación que realiza la ALU.

2.7.1. Circuito de prueba 1

Ya que las palabras A y B no las hemos usado propiamente en nuestro diseño (sólo las hemos conectado a la ALU), fijaremos éstas.
Establecemos los dos bits más significativos de A y B en alta, esto es, $A3 = A2 = B3 = B2 = 1$. Esto lo hacemos cambiando un reloj digital por un *digital HI port*.

Usuamos los mismos tiempos que en los circuitos anteriores para P y X y añadimos los siguientes:

	A1	A0	B1	B0
ONTIME	125ns	250ns	500ns	1us
OFFTIME	125ns	250ns	500ns	1us

Cuadro 6: Tiempos para el circuito de prueba 1

El circuito correspondiente es

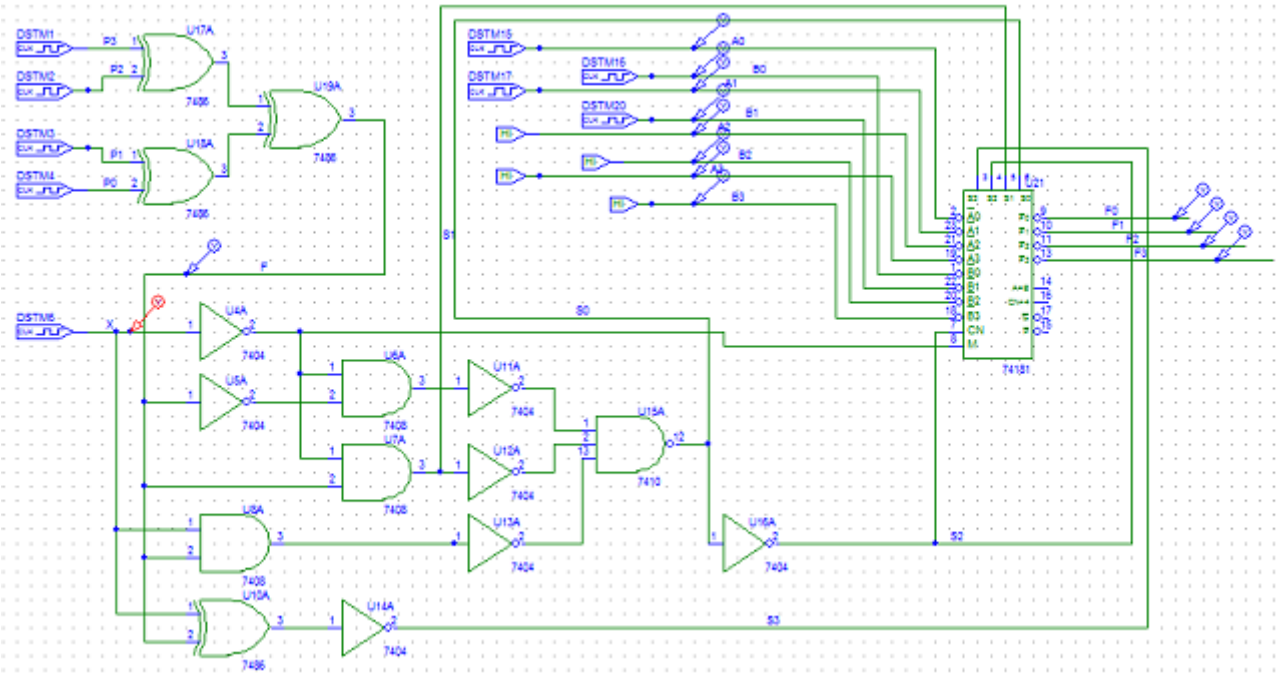

Figura 7: Circuito de prueba 1 (Prueba1.sch)

Realizamos la simulación y obtenemos el cronograma siguiente:

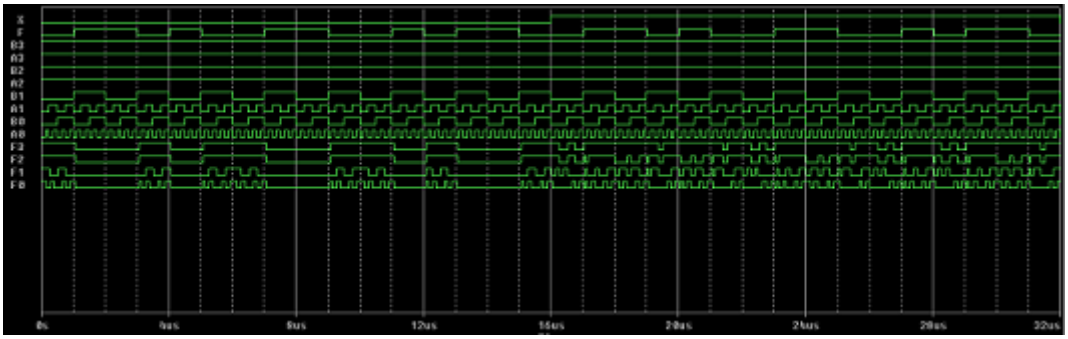

Figura 8: Cronograma del circuito de prueba 1

En la tabla siguiente se muestran los valores de 5 puntos escogidos al azar, las palabras que representan y la operación que realiza la ALU

X	F	B3	A3	B2	A2	B1	A1	B0	A0	F3	F2	F1	F0	A	B	Operación
0	0	1	1	1	1	0	0	0	1	1	1	1	0	1101	1100	$\overline{A \oplus B}$
1	0	1	1	1	1	0	1	0	0	0	0	0	0	1110	1100	$A\ PLUS\ A\overline{B}$
0	1	1	1	1	1	1	0	1	0	0	0	0	0	1100	1111	Puesta a 0
1	0	1	1	1	1	0	0	1	0	0	0	0	0	1100	1101	$A\ PLUS\ A\overline{B}$
1	1	1	1	1	1	1	0	1	0	1	1	0	0	1100	1111	$A\ PLUS\ B\ PLUS 1$

Cuadro 7: Comprobaciones para el circuito de prueba 1

Los valores de $F3$, $F2$, $F1$ y $F0$ son la salida de la ALU. Son los bits del resultado de la operación y son correctos.

2.7.2. Circuito de prueba 2

En esta simulación usaremos el circuito anterior pero ahora fijamos en baja las entradas $B3$,$B2$,$A3$ y $A2$.
No cambiamos ningún tiempo en los relojes, sólo los cuatro *digital HI port* por cuatro *digital LO port*.
El circuito es

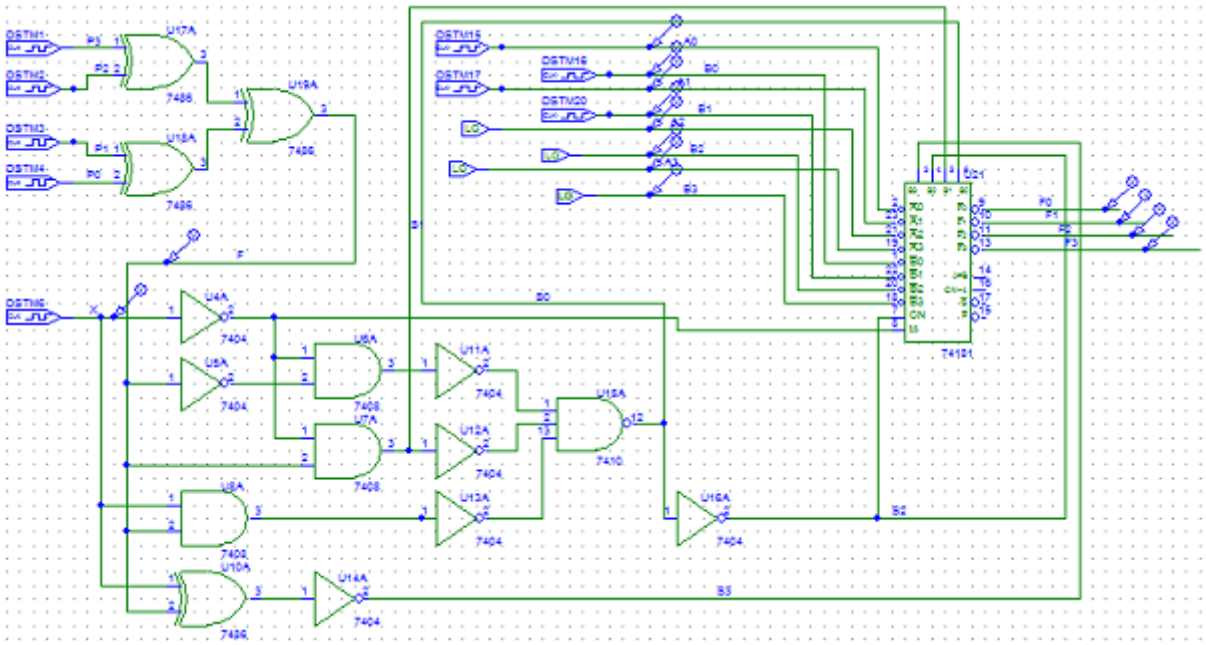

Figura 9: Circuito de prueba 2 (Prueba2.sch)

Realizamos la simulación y obtenemos el cronograma siguiente:

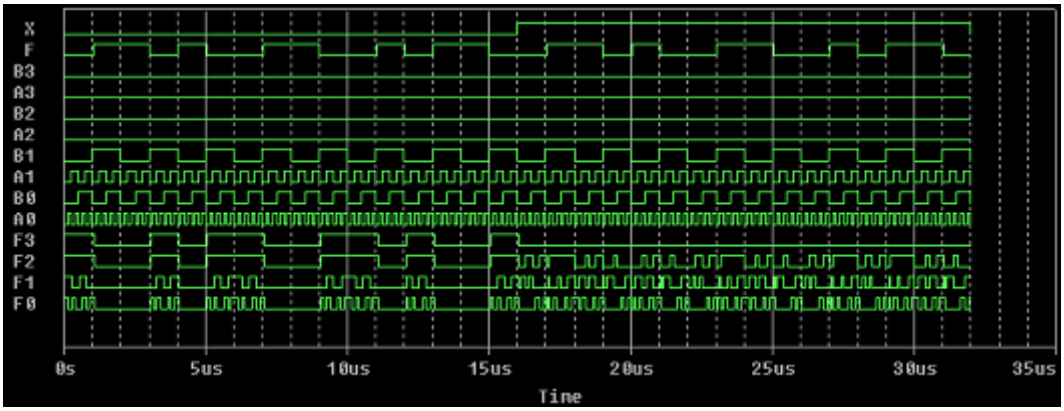

Figura 10: Cronograma del circuito de prueba 2

En la tabla siguiente se muestran los valores de 7 puntos escogidos al azar, las palabras que representan y la operación que realiza la ALU

F	X	B3	A3	B2	A2	B1	A1	B0	A0	F3	F2	F1	F0	A	B	Operación
0	0	0	0	0	0	0	0	1	1	1	1	1	1	0001	0001	$\overline{A \oplus B}$
1	0	0	0	0	0	1	1	1	0	0	0	0	0	0010	0011	Puesta a 0
0	0	0	0	0	0	0	1	0	1	1	1	0	0	0011	0000	$\overline{A \oplus B}$
1	0	0	0	0	0	1	1	0	1	0	0	0	0	0011	0010	Puesta a 0
0	1	0	0	0	0	0	0	1	0	0	0	0	0	0000	0001	$A\ PLUS\ A\overline{B}$
1	1	0	0	0	0	1	1	0	0	0	1	0	1	0010	0010	$A\ PLUS\ B\ PLUS1$
1	1	0	0	0	0	1	0	1	1	0	1	0	1	0001	0011	$A\ PLUS\ B\ PLUS1$

Cuadro 8: Comprobaciones para el circuito de prueba 2

Los valores de $F3$, $F2$, $F1$ y $F0$ son la salida de la ALU. Son los bits del resultado de la operación y son correctos.

2.7.3. Circuito de prueba 3

Las operaciones que realiza la ALU sobre las palabras A y B dependen de la variable x y de la paridad de la palabra P, que sólo pueden tomar los valores 0 y 1.
En esta simulación generamos señales directamente sobre F (la paridad) y sobre X. Es decir, anulamos el bloque A, que calcula la paridad, cambiándolo por un reloj digital que nos proporciona directamente el valor en alta o en baja.
En cuanto a las palabras A y B, las consideramos como $A(010W)$ y $B(W011)$. Es decir, los bits de cada palabra son fijos excepto los que corresponden a W.
Así pues, tendremos un total de 16 señales distintas.
Hemos escogido esta elección porque ya se ha comprobado anteriormente que el bloque A (generador de paridad, F) es correcto. De este modo, se reducen las configuraciones de entradas posibles, aunque no tanto como se quisiera, razón por la cuál hemos fijado también bits en A y B.
Con estas modificaciones el circuito queda como

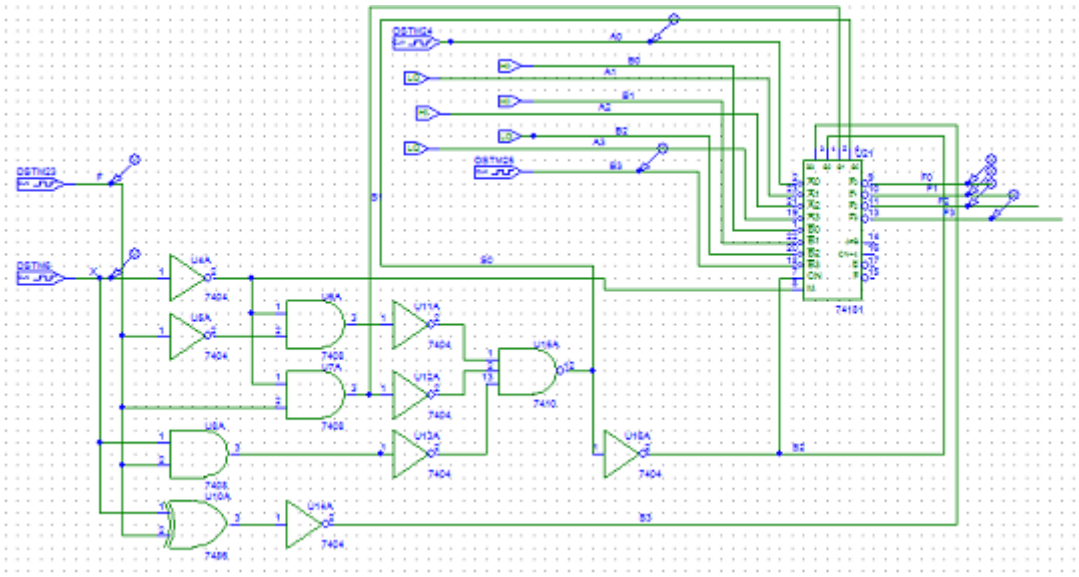

Figura 11: Circuito de prueba 3 (Prueba3.sch)

En el que los relojes digitales tienen tienen los tiempos para DELAY:

	F	X	B3	A0
ONTIME	8us	4us	2us	1us
OFFTIME	8us	4us	2us	1us

Cuadro 9: Tiempos para el circuito de prueba 3

Realizamos la simulación y obtenemos el cronograma siguiente:

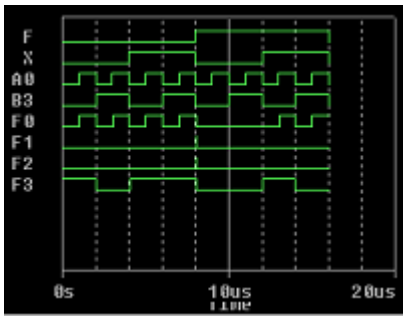

Figura 12: Cronograma del circuito de prueba 3

Que nos proporciona la siguiente tabla de verdad, en la que incluimos la operación que realiza la ALU, las palabras A y B y el resultado teórico (el resultado esperado).

F	X	A0	B3	F0	F1	F2	F3	A	B	Operación	Resultado
0	0	0	0	0	0	0	1	0100	0011	$\overline{A \oplus B}$	1000
0	0	1	0	1	0	0	1	0101	0011	$\overline{A \oplus B}$	1001
0	0	0	1	0	0	0	0	0100	1011	$\overline{A \oplus B}$	0000
0	0	1	1	1	0	0	0	0101	1011	$\overline{A \oplus B}$	0001
0	1	0	0	0	0	0	1	0100	0011	$A\ PLUS\ A\overline{B}$	1000
0	1	1	0	1	0	0	1	0101	0011	$A\ PLUS\ A\overline{B}$	1001
0	1	0	1	0	0	0	1	0100	1011	$A\ PLUS\ A\overline{B}$	1000
0	1	1	1	1	0	0	1	0101	1011	$A\ PLUS\ A\overline{B}$	1001
1	0	0	0	0	0	0	0	0100	0011	Puesta a 0	0000
1	0	1	0	0	0	0	0	0101	0011	Puesta a 0	0000
1	0	0	1	0	0	0	0	0100	1011	Puesta a 0	0000
1	0	1	1	0	0	0	0	0101	1011	Puesta a 0	0000
1	1	0	0	0	0	0	1	0100	0011	$A\ PLUS\ B\ PLUS\ 1$	1000
1	1	1	0	1	0	0	1	0101	0011	$A\ PLUS\ B\ PLUS\ 1$	1001
1	1	0	1	0	0	0	0	0100	1011	$A\ PLUS\ B\ PLUS\ 1$	(1)0000
1	1	1	1	1	0	0	0	0101	1011	$A\ PLUS\ B\ PLUS\ 1$	(1)0001

Cuadro 10: Tabla de verdad del circuito de prueba 3

En la columna de *Resultado* algunos resultados comienzan por *(1)*, que es el último acarreo de la suma (en caso de haberlo) y que la ALU no tiene en cuenta debido a que no hay suficientes bits para representarlo.
Notemos que el orden de las columnas de salida de la ALU está escrito al revés, es decir, se han escrito primero los bits menos significativos, por lo que deben leerse de derecha a izquierda.
Las salidas de la ALU coinciden con los resultados esperados (despreciando el último acarreo).

2.8. Observaciones específicas sobre los cronogramas

En todos los cronogramas estudiados se aprecian una especie de pequeños retardos en las señales. Observermos, por comodidad, el cronograma del circuito de prueba 3 ampliado

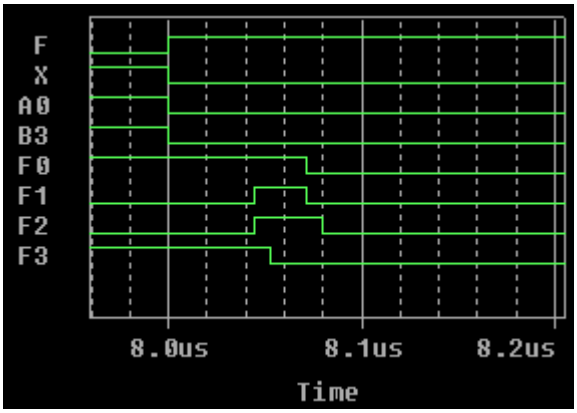

Figura 13: Cronograma ampliado del circuito de prueba 3

Se aprecian perfectamente cuatro retardos entre los tiempos 8.0us y 8.1us en las señales $F0$, $F1$, $F2$ y $F3$ que son, precisamente, las salidas de la ALU.
La señal más corta en este circuito es la de $A0$ con un intervalo de 1us, mientras que estos retardos duran, aproximadamente, 0.065us para $F0$ y $F1$, 0.08us para $F2$ y 0.045us para $F3$.
Deducimos que se deben a que el programa mide la tensión en de todos los marcadores (*Voltage/Level Marker)* en cada instante. De este modo, cuando la señal cambia en las entradas hay un determinado instante que las salidas de la ALU tienen todavía las señales correspondientes al resultado de la operación realizada para la entrada anterior.
Cuando pasa este tiempo, que consideramos que se trata del que necesita la ALU para trabajar, se obtienen los valores esperados (el resultado correspondiente a la actual entrada).
La operación a realizar por la ALU es la puesta a 0.
En el intervalo del retardo, los valores de cada salida se ponen en alta, excepto el de $F0$ que ya estaba en alta y continua en alta. Instantáneamente se ponen en baja, primero $F0$, seguido de $F1$ y $F2$ (aparentamente al mismo tiempo) y finalmente $F3$, es decir, del bit menos significativo al más significativo.
Sin embargo, en las tres siguientes operaciones, que también realizan la puesta a 0, no ocurren estos errores. Consideramos que puede deberse a que las salidas ya están en baja para la operación anterior.
Es decir, los retardos se producen cuando hay una conmutación en las salidas de la ALU.

Ahora, si pensamos que en efecto los retardos se deben al tiempo que necesita cada elemento del circuito, intuimos que deben ser acumulativos. Es decir, que los retardos en los marcadores de los elementos que actúan después debe ser mayor que el de los elementos anteriores.
Para ver si esto ocurre, situamos marcadores antes y después de un elemento del circuito de la *Figura 4* (circuito *preALU.sch*) de este modo

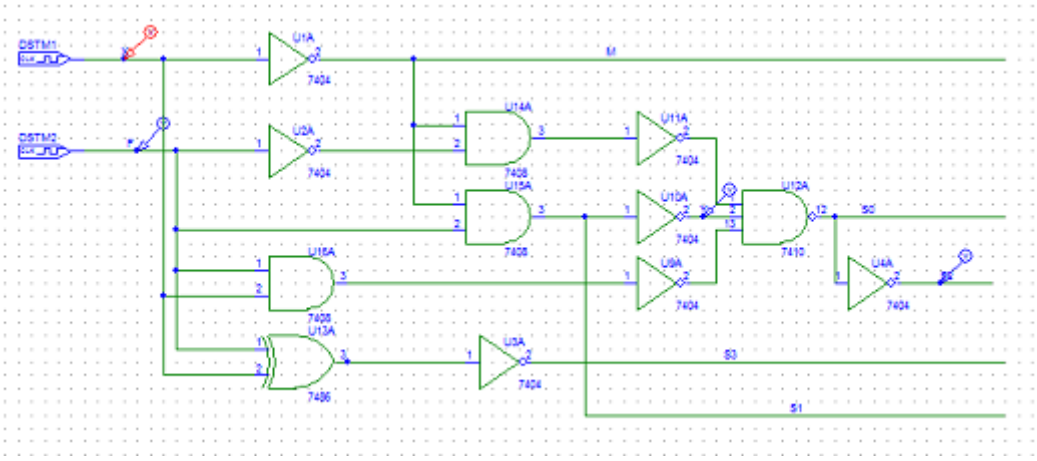

Figura 14: Circuito *preALUretardo.sch*

Y este es el cronograma ampliado en el momento de la primera conmutación de F

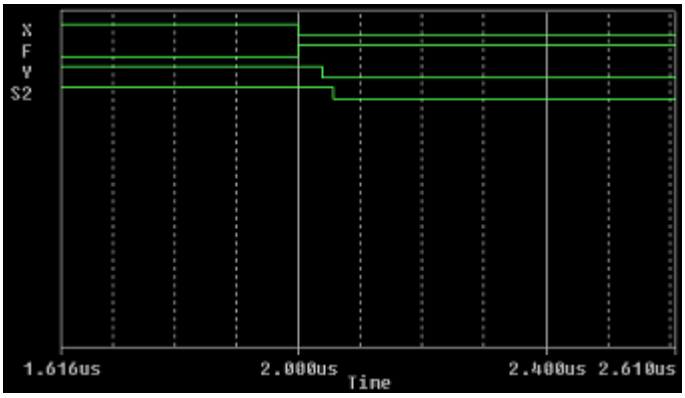

Figura 15: Cronograma ampliado de *preALUretardo.sch*

Como esperamos, el retardo ocurrido en $S2$ es mayor que en Y ya que la señal en $S2$ depende de la de Y.