DISEÑO SÍNCRONO DE CIRCUITOS DIGITALES

Miguel Angel Freire Rubio
Septiembre de 2008

Presentación

Este texto ha sido realizado para los alumnos que cursan la asignatura de Electrónica Digital en la especialidad de Sistemas Electrónicos de la titulación de Ingeniería Técnica de Telecomunicación, impartida en la EUIT de Telecomunicación de la Universidad Politécnica de Madrid. Su objetivo es abarcar los contenidos del tema 3 de teoría de la asignatura, Diseño Síncrono, que trata la problemática relacionada con la aplicación de las técnicas de diseño síncrono para la realización de sistemas digitales complejos.

En el capítulo 1 se estudia el régimen transitorio de los circuitos combinacionales para explicar los mecanismos de generación de *glitches* y deducir los problemas de funcionamiento que pueden ocasionar.

En el capítulo 2 se enuncian y justifican las reglas que constituyen las técnicas de diseño síncrono, se presenta el aspecto de la arquitectura genérica de un circuito síncrono y se caracteriza el modo en que se ejecutan las operaciones en estos circuitos. Al final del capítulo se explica cómo puede calcularse la frecuencia máxima de reloj a la que puede funcionar un sistema síncrono.

La primera parte del capítulo 3, con la que se completa la cobertura a los contenidos de teoría del tema 3 de Electrónica Digital, se dedica a las prácticas de diseño que permiten conseguir soluciones funcionales compatibles con las reglas de diseño síncrono; también se tratan aquí los problemas asociados a la sincronización de las entradas. La segunda parte del capítulo 3 presenta las soluciones que pueden adoptarse para aumentar la frecuencia de funcionamiento y la capacidad de procesamiento de un circuito síncrono (paralelismo, segmentación y creación de rutas *multiciclo*). Aunque estos asuntos no forman parte del temario de la asignatura, la lectura de estos apartados resulta aconsejable para aquellos alumnos que quieran tener unas nociones básicas de técnicas de diseño de uso frecuente en la práctica profesional del diseño de sistemas digitales.

Por último, hay que señalar que en el texto se incluye un buen número de ejemplos cuyo propósito es ilustrar los conceptos presentados; han sido realizados utilizando la tecnología de referencia de la asignatura de Electrónica Digital (Lógica Programable) y la herramienta de CAD empleada por los alumnos de la misma (Max+plus II de Altera).

1.- Modelos de funcionamiento de los circuitos combinacionales

En los primeros apartados (1,2 y 3) de este capítulo, y a modo de introducción, se recuerdan las propiedades básicas del funcionamiento de los circuitos combinacionales, así como las características de los modelos de funcionamiento ideal y con retardos que se utilizan en las operaciones de análisis y síntesis de combinacionales. A continuación (apartado 1.4) se analiza el comportamiento en conmutación de los circuitos combinacionales complejos utilizando modelos con retardos, con el objetivo de introducir al lector en la problemática derivada de la aparición de pulsos espurios (glitches) y se presentan los criterios de clasificación de éstos (apartado 1.5). El capítulo finaliza (apartado 1.6) exponiendo los problemas de funcionamiento típicos que pueden ocasionar los glitches y las restricciones que imponen al diseño de sistemas digitales.

1.1 Características del funcionamiento de los Circuitos Combinacionales

La propiedad que caracteriza el funcionamiento de cualquier circuito combinacional es que la combinación de bits que entrega en sus salidas sólo depende de la que simultáneamente esté presente en sus entradas, es decir, el valor que se establece en la salida del circuito no viene determinado en forma alguna por las combinaciones de entrada que hayan podido existir en instantes de tiempo anteriores. Esta propiedad fundamental suele enunciarse diciendo que los circuitos combinacionales *no tienen memoria*. En esta fórmula se resume el comportamiento característico de estos circuitos; vamos a repasarlo utilizando un ejemplo sencillo: un sumador combinacional.

En la figura 1, se muestra un sumador en el que ambas entradas, A y B, toman el valor 2 (por simplicidad se indican los valores en decimal, en lugar de en binario). Puesto que la función del circuito es calcular la suma de A y B, el valor de salida debe ser 4.

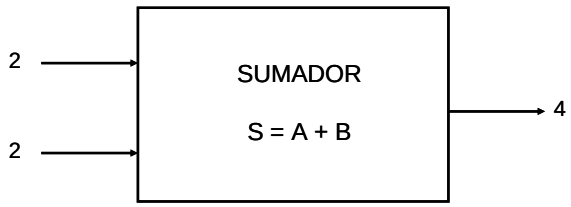

Figura 1. Sumador combinacional

Para determinar el resultado de la suma al circuito le basta con evaluar y procesar el valor presente en las entradas, resultando irrelevante el que hayan tomado en cualquier otro momento (el circuito no necesita tener memoria para realizar la suma). Es evidente que a un determinado valor de entrada, por ejemplo 2 y 2, le corresponde siempre el mismo valor de salida, 4 –esto no se verifica al revés, obviamente la salida también valdrá 4, por ejemplo, cuando las entradas A y B valgan 1 y 3. Observe, por último, que mientras las entradas A y B valgan 2 la salida mantendrá el valor 4, y sólo cuando cambien las combinaciones de entrada de modo que su suma sea distinta de 4 variará la salida del circuito. En definitiva, la salida de un circuito combinacional sólo puede cambiar si también cambia su entrada.

1.2 Modelo ideal de funcionamiento de los Circuitos Combinacionales

El modelo ideal de funcionamiento de los circuitos combinacionales supone que la respuesta a los cambios en sus entradas es instantánea. Su representación más usual son las *tablas de verdad*, que indican el valor que toma cada bit de salida para cada combinación de bits de entrada. La figura 2a muestra la tabla de verdad de un sumador de dos bits.

A_1	A_0	B_1	B_0	S_2	S_1	S_0
0	*0*	*0*	*0*	*0*	*0*	*0*
0	*0*	*0*	*1*	*0*	*0*	*1*
0	*0*	*1*	*0*	*0*	*1*	*0*
0	*0*	*1*	*1*	*0*	*1*	*1*
0	*1*	*0*	*0*	*0*	*0*	*1*
0	*1*	*0*	*1*	*0*	*1*	*0*
0	*1*	*1*	*0*	*0*	*1*	*1*
0	*1*	*1*	*1*	*1*	*0*	*0*
1	*0*	*0*	*0*	*0*	*1*	*0*
1	*0*	*0*	*1*	*0*	*1*	*1*
1	*0*	*1*	*0*	*1*	*0*	*0*
1	*0*	*1*	*1*	*1*	*0*	*1*
1	*1*	*0*	*0*	*0*	*1*	*1*
1	*1*	*0*	*1*	*1*	*0*	*0*
1	*1*	*1*	*0*	*1*	*0*	*1*
1	*1*	*1*	*1*	*1*	*1*	*0*

Figura 2a. Tabla de verdad de un sumador

El funcionamiento ideal también puede expresarse mediante ecuaciones lógicas (booleanas), o gráficamente utilizando cronogramas funcionales. En los cronogramas los bits de entrada y salida se dibujan como señales ideales, sin tiempos de subida o bajada y sin retardos entre entradas y salidas -a este tipo de cronogramas se los denomina *funcionales* precisamente porque representan exclusivamente el funcionamiento lógico del circuito y no consideran tiempos de retardo. En el cronograma de la figura 2b se muestra un ejemplo del funcionamiento ideal del sumador de 2 bits.

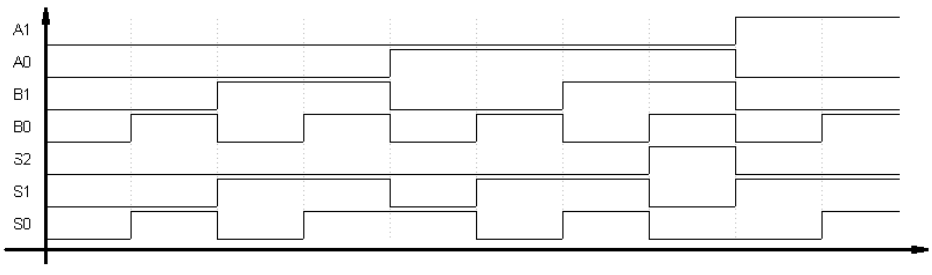

Figura 2b. Cronograma

Las tablas de verdad y los cronogramas funcionales se utilizan en las tareas del ciclo de diseño en que resulta admisible el uso de modelos ideales de funcionamiento (síntesis de ecuaciones booleanas y análisis lógico del funcionamiento del sistema).

1.3 Modelos de funcionamiento de Circuitos Combinacionales simples con retardo

Las metodologías básicas de análisis y diseño (síntesis y verificación) de circuitos combinacionales se basan en modelos ideales de funcionamiento del hardware. El error principal –el único, en realidad, relevante- que se comete al interpretar con estos modelos el funcionamiento de un circuito real se debe a que las primitivas lógicas con que se construyen los circuitos (puertas lógicas o LUTs o PALes o multiplexores), y los elementos de interconexión con que se unen, tienen retardos. El hecho de que tengan retardos significa que sus salidas tardan un cierto tiempo en reaccionar a los cambios que se dan en sus entradas, o sea, que desde que se modifica la combinación de entrada hasta que se establece el valor de salida correcto pasa un cierto tiempo.

Un circuito inversor resulta suficiente para ilustrar la repercusión de los retardos sobre el funcionamiento de un circuito combinacional real. En la figura 3 se muestra un modelo ideal (3a) y con retardo (3b) de una puerta inversora. La única diferencia entre ambos es que en el modelo real la salida no cambia en el mismo instante que la entrada, sino que se retrasa cierto tiempo: el tiempo de retardo (o propagación) de la puerta. Vamos a repasar a continuación algunas cuestiones importantes relacionadas con este modelo de funcionamiento.

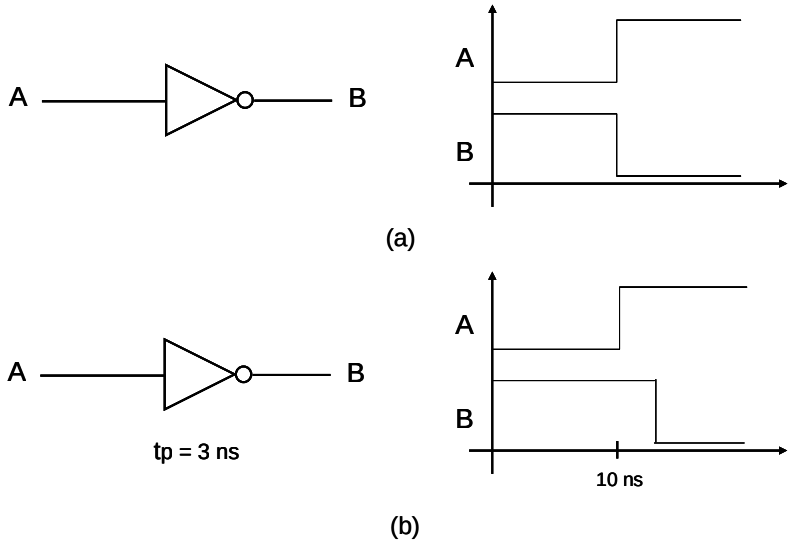

Figura 3. Modelos de funcionamiento de un inversor

En primer lugar hay que recordar que el tiempo de retardo del hardware con el que se construyen los circuitos digitales se caracteriza por medio de valores mínimos y máximos –los fabricantes proporcionan a veces también valores típicos-, de modo que resulta imposible conocer "a priori" su valor exacto; en realidad ni tan siquiera puede hablarse de semejante concepto, un "tiempo de retardo exacto", porque el retardo está sujeto a circunstancias cambiantes (temperatura, presión, humedad, valor de la tensión de alimentación, envejecimiento del componente) e incluso una medida tomada sobre el propio circuito no podría aceptarse como tal –lo único que puede garantizarse es que el retardo medido va a estar dentro de un intervalo de valores si se respetan una serie de condiciones de operación especificadas por el fabricante del *hardware*. Lo que se

representa normalmente en los cronogramas en los que se consideran retardos es el valor máximo o mínimo del tiempo de propagación (figura 4).

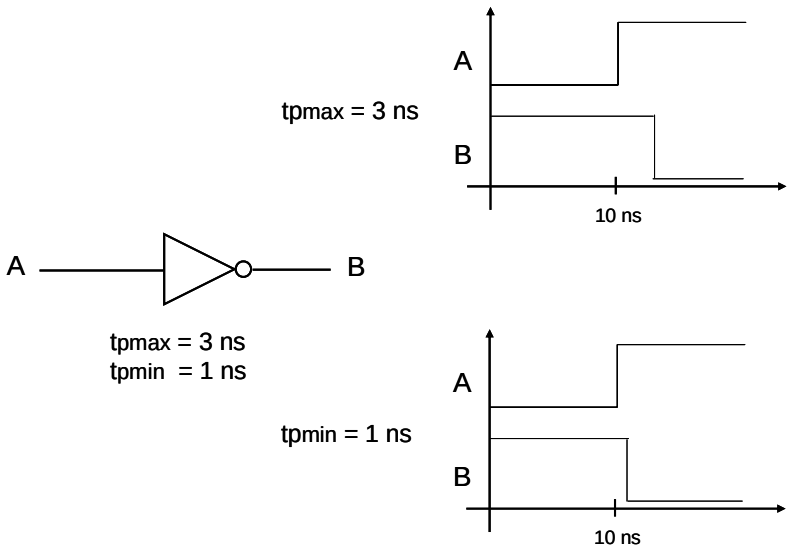

Figura 4. Retardo máximo y mínimo

En segundo lugar resulta necesario resaltar el hecho de que el tiempo de retardo de un circuito combinacional depende tanto de los elementos que realizan operaciones lógicas como de las pistas y recursos de interconexión que los unen y que, dependiendo de la tecnología que se use, uno u otro componente puede ser el dominante (en las FPGAs, por ejemplo, el retardo imputable a la red de interconexión suele ser el factor que más contribuye al retardo total). El modelo de retardo para un circuito inversor podría ser como el de la figura 5.

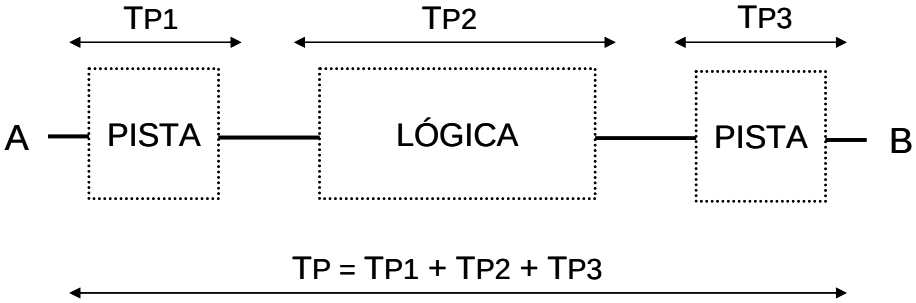

Figura 5. Componentes del retardo

Por último, hay que poner de manifiesto un detalle del funcionamiento del inversor –esencial para la temática que va a tratar este texto- que puede pasar inadvertido: desde que cambia la entrada y hasta que transcurre el tiempo de retardo y, en consecuencia, se establece un nuevo valor en la salida, la operación lógica que realiza la puerta inversora es incorrecta, es decir, la puerta funciona *mal*, pues el nivel lógico de salida no es el complementario del de entrada. Estos intervalos de "mal funcionamiento" se dan en todos los circuitos combinacionales cuando hay cambios en la entrada que llevan aparejados conmutaciones en la salida, y están acotados por el tiempo de propagación. El funcionamiento correcto se restablece cuando, desde la conmutación de la entrada, transcurre un tiempo igual al de propagación; por ello

podemos considerarlo como un *régimen transitorio* de funcionamiento. Teniendo esto en cuenta, diremos a partir de aquí, en este texto, que un circuito combinacional está en *régimen permanente* cuando la entrada que tiene en un determinado momento no ha cambiado durante un tiempo mayor o igual al de propagación del circuito, y diremos que está en *régimen transitorio* en caso contrario. Durante el *régimen permanente* un circuito combinacional funciona como su modelo ideal, mientras que durante el *régimen transitorio* el funcionamiento puede ser o no incorrecto -en el caso del circuito inversor siempre es incorrecto porque no hay dos combinaciones de entrada a las que corresponda la misma combinación de salida, en otros circuitos combinacionales puede haber regímenes transitorios en los que el valor de salida sea correcto, aunque siempre habrá alguno en que no lo sea.

Ejercicio 1.1

En este ejercicio se hace uso de modelos de funcionamiento ideales y con retardos de un circuito inversor, se práctica el cálculo de tiempos de retardos con el analizador de tiempos del entorno MAX+plus II y se muestra cómo el retardo de un circuito puede variar al cambiar la estructura de los recursos de interconexión.

1. Utilice el entorno Max+plus II para crear un nuevo proyecto y, en él, un fichero de esquemas. En este fichero dibuje y chequee el circuito de la figura E1.1.

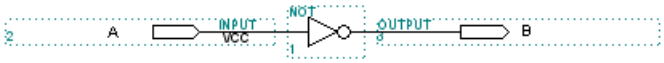

Figura E1.1

2. Genere un modelo de simulación funcional para el circuito inversor y un fichero de formas de onda, para definir los estímulos de una simulación con las siguientes características:

    a. Tiempo Máximo de Simulación (*End Time…*): 200 nanosegundos
    b. Grid: 50 nanosegundos
    c. Forma de onda de la señal A: Periodo de 100 nanosegundos, ciclo de trabajo del 50%.

3. Realice una simulación funcional del circuito. El resultado de la misma debe ser idéntico al de la figura E1.2.

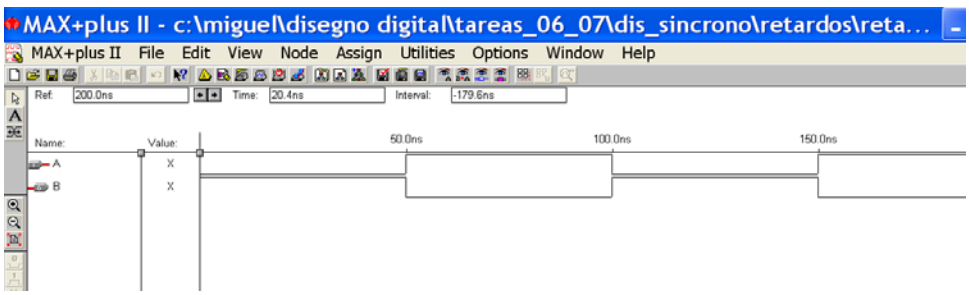

Figura E1.2

Al realizar la simulación funcional de un circuito se está utilizando un modelo de funcionamiento ideal del hardware, por tanto no se consideran retardos de propagación. Este modelo es suficiente para verificar el diseño lógico del circuito.

Cuando este diseño lógico se materializa en una realización hardware es posible realizar análisis que nos informen de los retardos reales del circuito (análisis de tiempos), o simulaciones que representen retardos en cronogramas (simulaciones lógicas de modelos con retardos).

4. Asigne una FPGA 10K20RC208-4 para la materialización del circuito inversor diseñado en el proyecto y ordene la ejecución de una compilación para obtener un modelo se simulación con retardos y los ficheros de configuración de la FPGA.

5. Calcule con el analizador de tiempos el retardo del circuito –para ello seleccione en el menú *Analysis*, la opción *Delay Matrix*. En la figura E1.3 se muestra un posible[1] resultado de esta operación. El análisis indica que hay un tiempo máximo[2] de retardo de 13’7 ns.

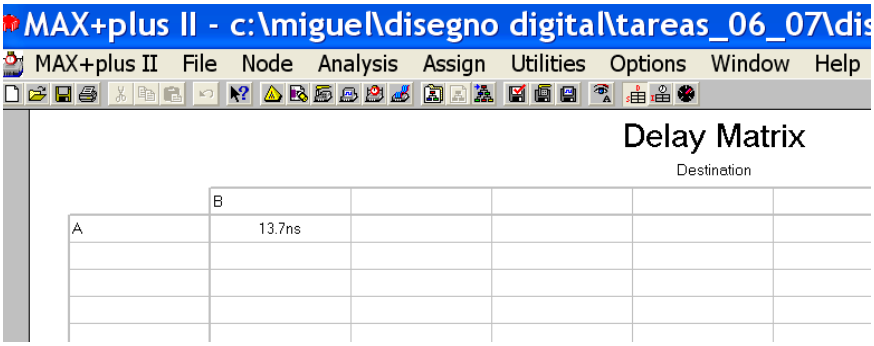

Figura E1.3

Este análisis nos informa de la discrepancia entre el modelo ideal y el modelo con retardos, la realización obtenida para el circuito inversor tendrá un *régimen transitorio* de hasta 13’7 ns en los que la salida del circuito, después de un cambio en la entrada, será incorrecta.

6. Ordene una simulación lógica del circuito –que ahora se realizará utilizando el modelo con retardos- y analice los resultados de la simulación. La salida del simulador debe ser muy parecida a la de la figura E1.4.

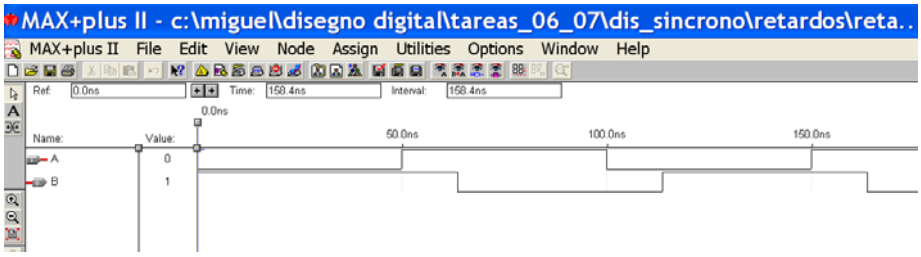

Figura E1.4

[1] Hay una componente aleatoria en el alojamiento (*fitting*) del diseño del inversor en la FPGA que puede dar lugar a diferentes resultados en diferentes compilaciones.
[2] El analizador de tiempos de Max+plus II trabaja exclusivamente con retardos máximos.

7. Utilice el editor de *Floorplan* del entorno Max+plus II para observar cómo se ha colocado la lógica del circuito en la FPGA elegida durante la última compilación –para ello, seleccione la opción *Last Compilation Floorplan* en el menú *Layout*. En la figura E1.5 se muestra un posible[3] aspecto del editor de *Floorplan*. En la vista que muestra dicha figura se ha seleccionado la célula lógica ocupada por el diseño para resaltar[4] la colocación de los pines del inversor en la FPGA.

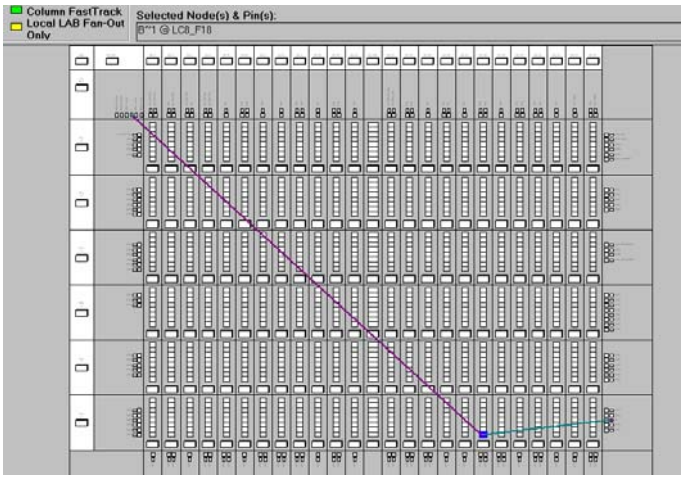

Figura E1.5

Esta figura permite distinguir los componentes que contribuyen al retardo de un circuito combinacional real materializado en hardware (Figura 5), y que son: por un lado el retardo asociado al hardware que materializa las funciones lógicas (*Logic Elements* en el caso de una FPGA de ALTERA) y, por otro, el correspondiente a los elementos que interconectan la lógica del circuito (en nuestro ejemplo, las líneas y recursos de conmutación programables de la red de interconexión, *Fast Track Interconnect*, que unen los pines de entrada y salida al único *Logic Element* que se utiliza). Además por tratarse de un diseño realizado en una FPGA hay una fracción de retardo imputable a las células configurables de entrada-salida (*I/O Elements*, según la denominación de ALTERA).

El retardo de un circuito puede modificarse cambiando la estructura de elementos lógicos con que está construido (en nuestro ejemplo la simplicidad lógica lo impide) o, también, la estructura o longitud de las pistas de interconexión. En nuestro ejemplo podemos cambiar el retardo de las líneas de interconexión modificando la posición de los pines del inversor en la FPGA.

8. Utilice el propio editor de *Floorplan* para asignar los pines A y B a dos de los pines dispuestos a la izquierda de la segunda fila (*Row B*) de la FPGA –para hacerlo, recuerde que debe activar la opción *Current Assigments Floorplan* para abandonar la vista de los resultados de la última compilación. La figura E1.6 ilustra el aspecto del resultado de ambas asignaciones.

[3] Véase la nota 1.
[4] Para que usted también los pueda ver al realizar el ejercicio, debe tener activas las opciones *Show node fan-in* (y *fan-out*) en el menú *Options* y la opción *LAB View* en el menú *Layout*.

9. Vuelva a compilar el diseño, para que la realización hardware incluya la posición de pines indicada anteriormente y, seguidamente, ejecute un análisis de tiempos para verificar que, efectivamente, los retardos del circuito han cambiado (el retardo correspondiente a esta asignación debe ser de 14’6 ns).

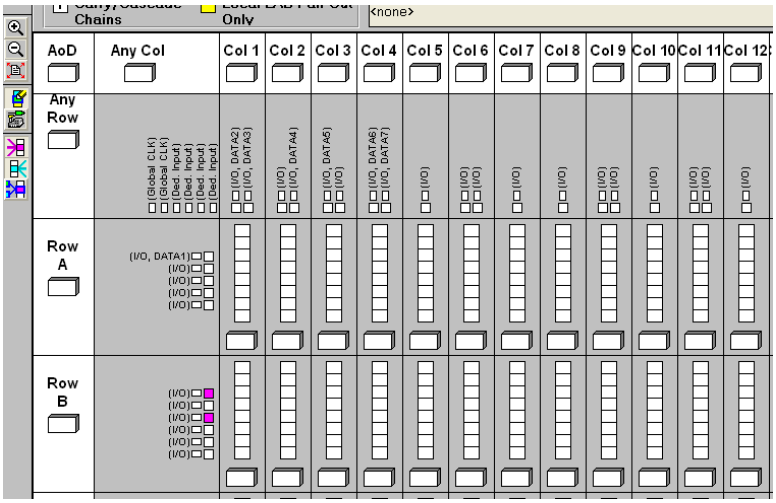

Figura E1.6

Ejercicio propuesto

El objetivo de este ejercicio es la evaluación de los diferentes componentes del retardo de un circuito combinacional realizado en una FPGA.

Diseñe y realice, utilizando un FPGA 10K20RC208-4, el circuito de la figura E2. Observe, con ayuda del editor de *Floorplan*, cómo se encaja en el FPGA y determine con el analizador de tiempos los retardos del circuito. De acuerdo con los resultados obtenidos, deduzca qué parte del retardo total es imputable a los LEs de la FPGA.

Nota: Puede contrastar su estimación con la información que aporta el fabricante, ALTERA, en el catálogo de la familia 10K, donde puede encontrar, también, la contribución de las células de IO al retardo total.

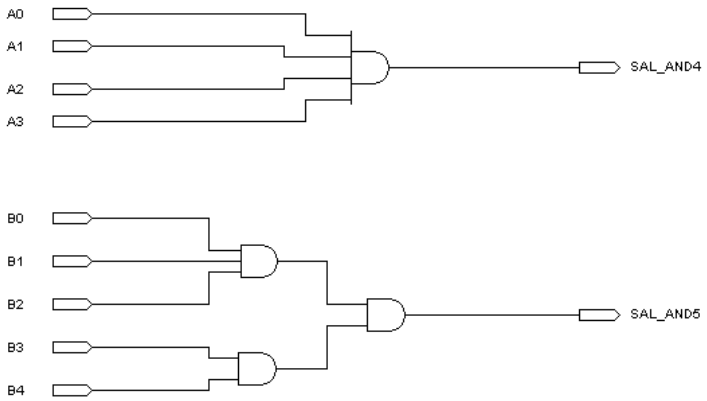

Figura E2

1.4 Circuitos combinacionales. Modelos reales de circuitos complejos

Una vez conocida la casuística asociada a los retardos en los circuitos combinacionales simples, podemos abordar el estudio del comportamiento real de los circuitos combinacionales complejos, entendiendo que esta categoría engloba a cualquier combinacional realizado mediante la interconexión de varias primitivas lógicas de una determinada tecnología (varias puertas, PALes, o LUTs).

El comportamiento real de los circuitos combinacionales complejos durante el régimen transitorio es diferente al de los simples y repercute muy seriamente sobre las metodologías de realización de circuitos digitales, por lo que resulta imprescindible conocerlo y ser capaz de valorar su efecto sobre el funcionamiento de un sistema digital. Lo que tiene de particular el régimen transitorio de un circuito combinacional complejo es que durante el mismo pueden originarse pulsos espurios, denominados *glitches*, debido a los retardos de los operadores lógicos y pistas de interconexión con que están construidos. La figura 6 nos permite estudiar este fenómeno en un circuito que realiza una función lógica sencilla.

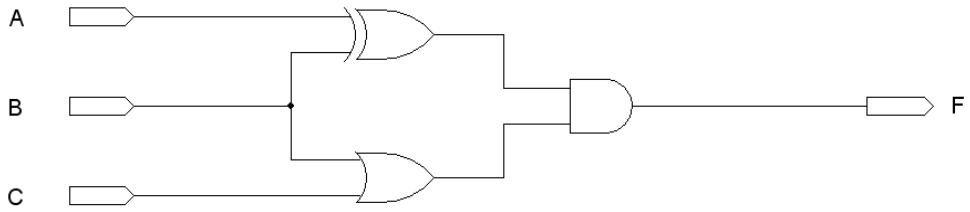

Figura 6. Circuito combinacional

Si el circuito está en régimen permanente –o lo que es lo mismo: considerando un modelo de funcionamiento ideal- y la combinación de entrada es A = 1, B = 0 y C = 0, en la salida de la puerta *xor* habrá un 1 y en la de la puerta *or* un 0, en consecuencia la salida de la puerta *and*, F, valdrá 0 (Figura 7).

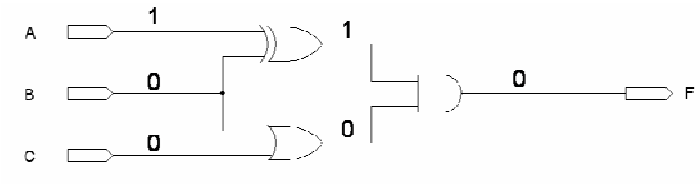

Figura 7. Régimen Permanente para A = 1, B = 0 y C = 0

Si en un determinado momento B pasa a valer 1, la nueva combinación de entrada, A = 1, B = 1 y C = 0, también tiene asociado, en régimen permanente, un 0 en la salida –ya que para esta combinación la salida de la puerta *xor* es 0 (Figura 8).

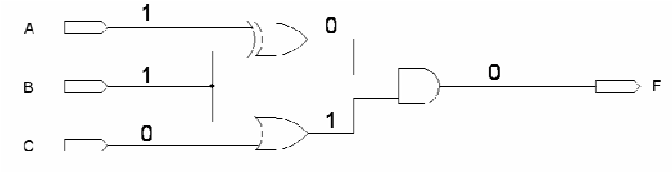

Figura 8. Régimen Permanente para A = 1, B = 1 y C = 0

El régimen transitorio ocasionado por esta conmutación de la entrada B se desarrollaría de la siguiente manera:

- Tras la conmutación de la entrada B del circuito, y al cabo de un tiempo T_1 (que será la suma del tiempo de retardo de la puerta *or* y de los retardos introducidos por las pistas de interconexión), el nivel lógico en la entrada de la puerta *and* conectada a la salida de la puerta *or* pasa a ser 1.

- Simultánea e independientemente, y transcurrido un tiempo T_2 desde la conmutación de B (cuyo valor será, en este caso, la suma de los retardos de otras pistas de interconexión y del tiempo de propagación de la puerta *xor*), el nivel lógico en la otra entrada de la puerta *and* pasará a ser 0.

- Si se da el caso de que T_1 es menor que T_2, habrá un 1 en las dos entradas de la puerta *and* durante un intervalo de tiempo de duración igual a $T_2 - T_1$ –ya que la entrada de la puerta *and* conectada a la salida de la puerta *or* se pone a 1 antes de que deje de haber un 1 en la otra entrada. Esta situación se ilustra en la figura 9.

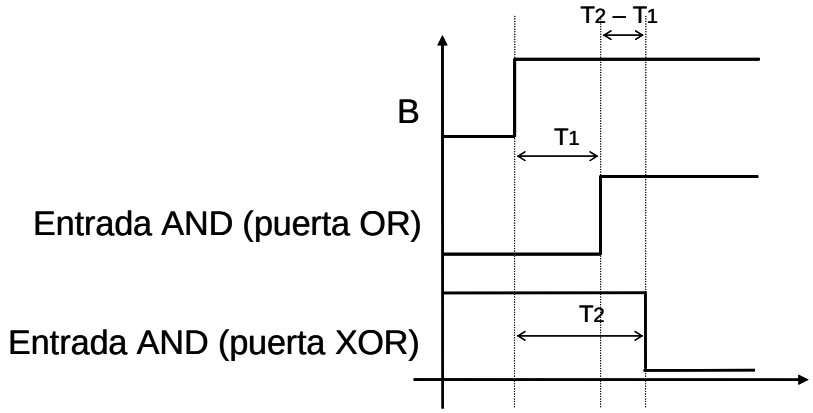

Figura 9. Retardos de Propagación en el R.T.

- Si la puerta *and* y la pista que conecta su salida a la del circuito acumulan un retardo T_3, se genera un *glitch* tal y como se muestra la figura 10.

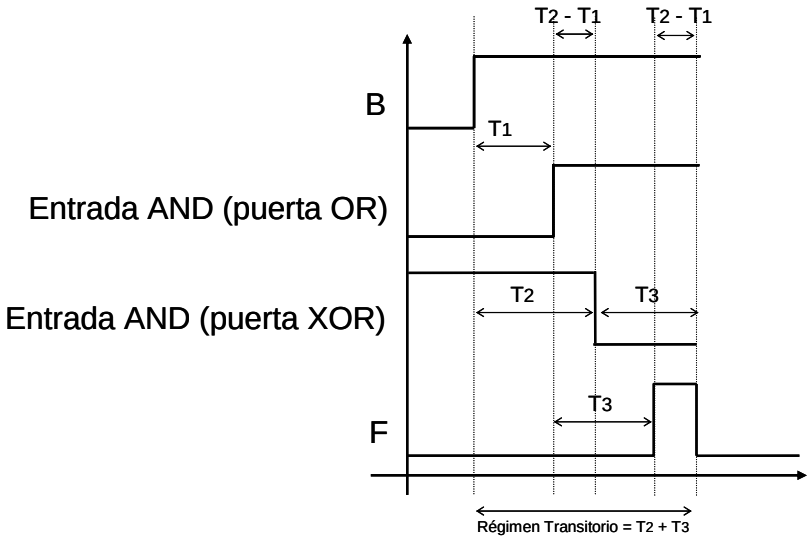

Figura 10. Generación de glitches

El ejemplo anterior muestra cómo una conmutación de los bits de entrada entre dos combinaciones que tienen asociado un mismo valor en la salida, un 0, puede dar lugar a la generación de un pulso espurio, un *glitch*, debido a los retardos asociados a los elementos del circuito; observe que con unos retardos diferentes podría cambiar el comportamiento del circuito: si T_1 fuese mayor que T_2, no se generaría el *glitch* en la transición estudiada porque no llegaría a haber simultáneamente dos unos –sí dos ceros- en la entrada de la puerta *and*.

El número y forma de los *glitches* que se generan en las salidas de un circuito combinacional en conmutación depende de su estructura –es decir, de qué realización concreta, de entre las múltiples posibles, se ha elegido para construirlo-, de los retardos asociados a los componentes del circuito (que son cambiantes y sólo en cierta medida controlables por el diseñador) y de la actividad en las entradas del circuito –que determina, en definitiva, las transiciones que se dan en la práctica y por tanto los regímenes transitorios que van a producirse.

1.5 *Glitches*. Clasificación

Los *glitches* se clasifican tradicionalmente de acuerdo a dos criterios. El primero es morfológico y tiene escaso interés práctico: distingue entre estáticos y dinámicos, siendo los primeros los que aparecen cuando la salida, en régimen permanente, no cambia (como en el ejemplo del apartado anterior) y los segundos los que se generan cuando la salida del circuito cambia de valor (en la figura 11 se muestran ejemplos de ambos tipos).

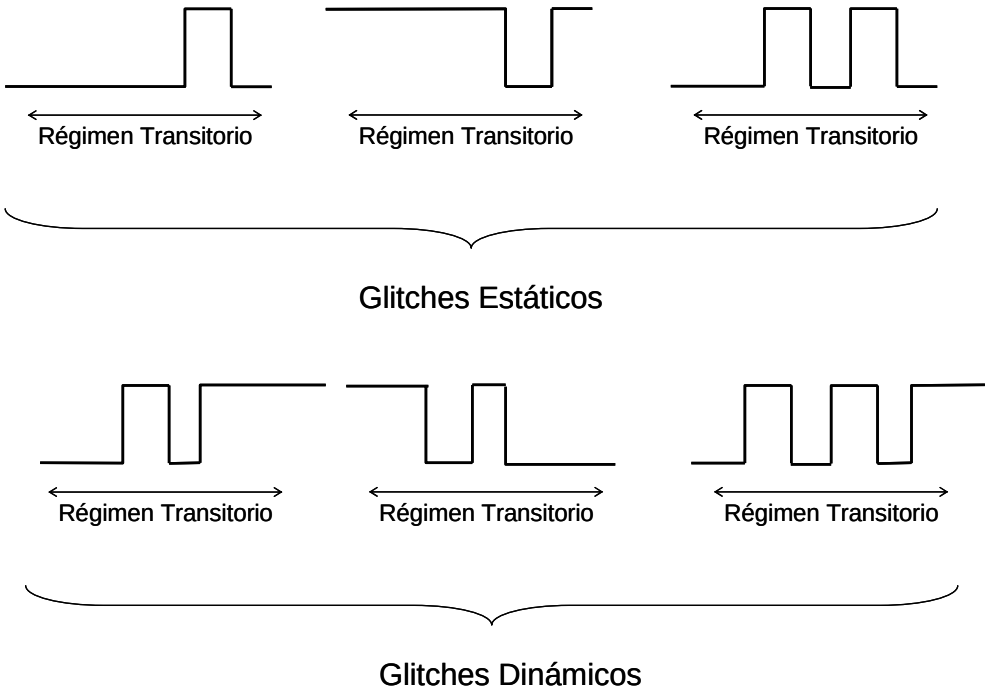

Figura 11. Glitches estáticos y dinámicos

El segundo criterio, más interesante, clasifica los *glitches* de acuerdo con el tipo de conmutación en la entrada del circuito que los origina; distingue entre dos tipos: los

glitches lógicos y los *glitches* funcionales. Los *glitches* lógicos son los que se generan cuando sólo cambia de valor un bit de entrada, y se caracterizan porque es posible suprimirlos modificando la realización del circuito –normalmente añadiendo lógica redundante. Los *glitches* funcionales son los que se producen únicamente cuando cambian simultáneamente dos o más bits en la combinación de entrada; este tipo de *glitches* no puede, en general, ser eliminado.

Ejercicio 1.2

En este ejercicio se constata la aparición de glitches durante el régimen transitorio de los circuitos combinacionales y su dependencia de cada realización concreta del circuito.

1. Utilice el entorno Max+plus II para crear un nuevo proyecto y, en él, un fichero de esquemas. En este fichero dibuje y chequee el circuito de la figura E2.1.

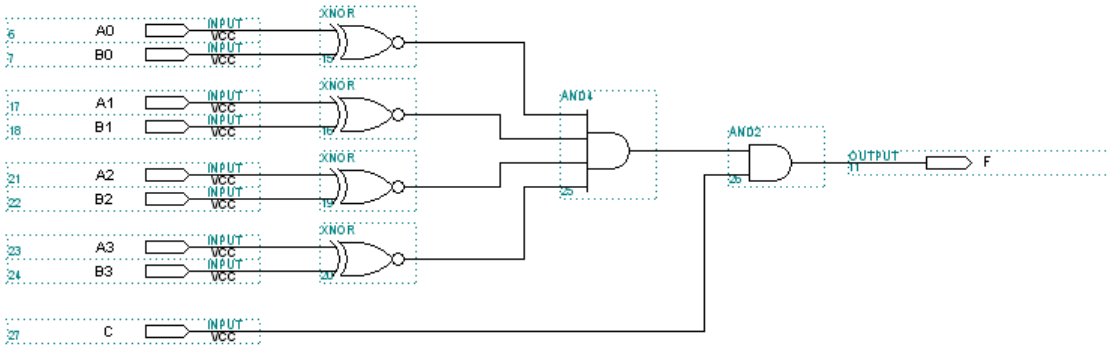

Figura E2.1

2. Genere un modelo de simulación funcional para el circuito y un fichero de formas de onda, para definir los estímulos de una simulación con las siguientes características:

    a. Tiempo Máximo de Simulación (*End Time…*): 350 nanosegundos
    b. Grid: 50 nanosegundos
    c. Forma de onda de la señal C: ‘0’ desde 0 ns hasta 50 ns, ‘1’ desde 50 hasta 350 ns.
    d. Forma de onda de la señal A[3..0]: 0_d (decimal) desde 0 ns hasta 50 ns y desde 100 ns hasta 250 ns, 15_d desde 50 ns hasta 100 ns, 3_d desde 250 ns hasta 300 ns y 4_d desde 300 ns hasta 350 ns.
    e. Forma de onda de la señal B[3..0]: 0_d desde 0 ns hasta 50 ns, 15_d desde 50 ns hasta 100 ns, 8_d desde 100 ns hasta 150 ns, 1_d desde 150 ns hasta 200 ns, 2_d desde 200 ns hasta 250 ns, 3_d desde 250 ns hasta 300 ns y 4_d desde 300 ns hasta 350 ns.

3. Realice una simulación funcional del circuito. El resultado de la misma debe ser similar al de la figura E2.2.

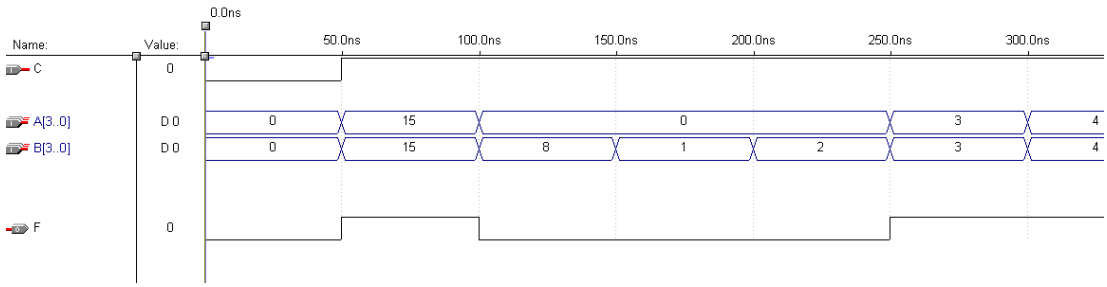

Figura E2.2

4. Asigne el dispositivo EPF10K20RC208-4 y los pines del circuito tal y cómo se indica en la tabla adjunta. Una vez efectuadas las asignaciones, realice una compilación para generar un modelo de simulación con retardos, seleccionando en el menú **Global Project Logic Synthesis**, la opción **WYSIWYG** en la ventana de selección **Global Project Synthesis Style** (Figura E2.3)

NOMBRE	Nº DE PIN
C	182
A0	157
A1	150
A2	144
A3	143
B0	149
B1	148
B2	142
B3	141
F	159

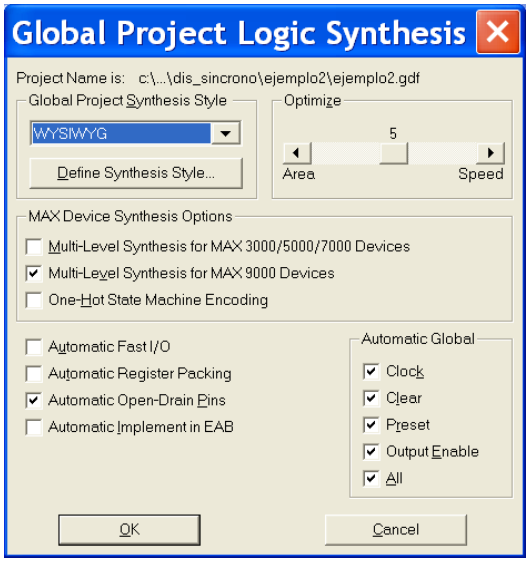

Figura E2.3

5. Una vez realizada la compilación, realice una simulación lógica con retardos del circuito. El fichero de formas de onda obtenido debe ser idéntico al de la figura E2.4.

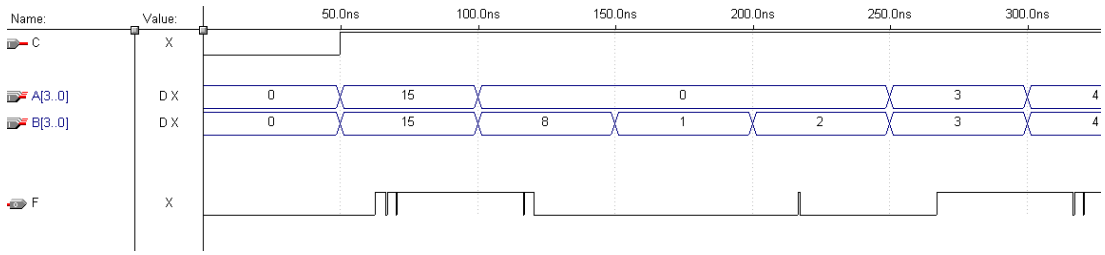

Figura E.2.4

Observe que en la simulación aparecen *glitches* dinámicos (en los transitorios correspondientes a las conmutaciones de entrada en 50 y 100 ns) y *glitches* estáticos (el resto). Puede comprobar, haciendo un *zoom* cercano sobre uno cualquiera, que tienen una anchura reducida -en el ejemplo, de 0'6 ns; en general suelen durar una porción pequeña del tiempo de retardo del circuito combinacional: lo que duren los desajustes lógicos provocados por las diferencias de retardo en los caminos internos del circuito- y que se producen en el régimen transitorio (puede comprobarlo calculando los retardos del circuito con el analizador de tiempos). Observe también que todos los *glitches* que aparecen en la simulación son funcionales.

6. La realización física del circuito, resultado de la compilación realizada, puede revisarse con ayuda del editor de *floorplan*. En la figura E2.5 se muestra su aspecto.

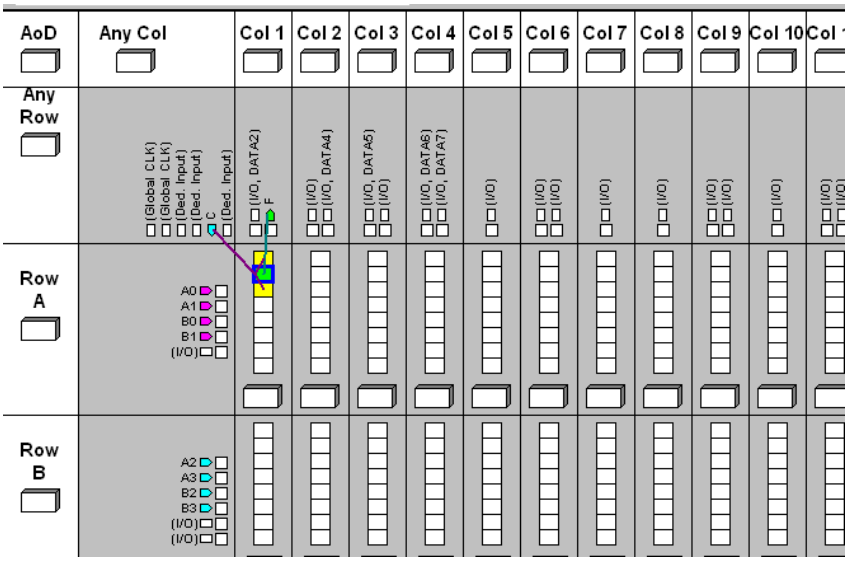

Figura E2.5

La realización se ha materializado en tres LUTs, una determina si A[1..0] es igual ó no a B[1..0], otra realiza la misma operación para A[3..2] y B[3..2], y la tercera evalúa la salida de las anteriores y C para determinar la salida del circuito –que será '1' si, y sólo si, A y B son iguales y C vale 1. Puede comprobar esto en el editor de *floorplan* con ayuda del visor de ecuaciones.

Si cambiamos la realización del circuito, probablemente –casi con toda seguridad- el patrón de *glitches* generados será diferente.

7. En la opción **Ignore Project Assigments** del menú **Assign**, seleccione la casilla **Pin & I/O Cell Assigments** (figura E2.6) y pulse **Aceptar**.

8. Recompile el diseño para generar una nueva realización del circuito y un modelo de simulación de la misma con retardos. Una vez finalizada la compilación, realice una simulación lógica y revise los resultados (figura E2.6).

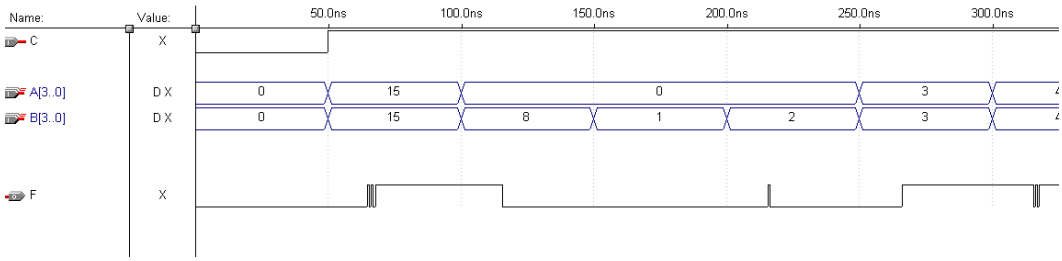

Figura E2.6

Efectivamente los resultados son diferentes a los de la simulación con retardos correspondiente a la realización anterior, porque el compilador ha cambiado la ubicación de los pines y LUTs en la FPGA y, como consecuencia, cambian los retardos del circuito y también los *glitches* que se generan –en el ejemplo desaparecen los que antes se producían en el transitorio asociado a la conmutación en 100 ns.

1.6 Efecto de los *glitches* sobre el funcionamiento de los sistemas digitales.

Como hemos visto en los apartados anteriores, el funcionamiento de los circuitos combinacionales reales se diferencia del funcionamiento ideal en que existe un retardo en el establecimiento de la combinación de salida correcta cuando cambiamos la combinación de entrada y en que, además, durante ese periodo de tiempo –que consideramos como un régimen transitorio de funcionamiento- puede haber pulsos espurios, *glitches*, en las salidas. Las salidas de un circuito combinacional que genere *glitches* pueden causar o no errores de funcionamiento dependiendo de "a qué" y "cómo" estén conectadas. Hay un conjunto de casos donde los *glitches* ocasionan problemas o funcionamientos incorrectos:

1. Cuando las salidas de un combinacional se conectan a la entrada de reloj de un *flip-flop* o la entrada de habilitación de un *latch*; en ambos casos un *glitch* constituye un evento que puede determinar la memorización incontrolada de un dato.

2. Cuando las salidas de un combinacional se conectan a entradas asíncronas de *latches* o *flip-flops*; porque los *glitches* pueden producir una inicialización indeseada.

3. Cuando salidas con *glitches* se conectan a entrada síncronas de *flip-flops* o *latches* pueden provocar violaciones de tiempo de *set-up* o *hold*, o el registro de un valor incorrecto; esto puede ocurrir, por ejemplo, si un flanco activo de reloj llega a un *flip-flop* mientras un circuito combinacional, cuya salida esté

conectada a la entrada síncrona de éste, pasa por un régimen transitorio en el que se produzcan *glitches* (ocurre lo mismo en el caso de los *latch*).

4. Cuando la salida del circuito combinacional es también una salida de un sistema digital cuyas especificaciones morfológicas son incompatibles con la presencia de *glitches*; un ejemplo de esto son las señales de control de buses asíncronos: señales de validación de datos o direcciones, de control de lectura/escritura, etcétera.

También hay casos, frecuentes en el diseño de sistemas digitales, donde la existencia de *glitches* en las salidas de un combinacional resulta irrelevante:

1. Cuando se encadenan varios circuitos combinacionales; el conjunto equivale a un único combinacional, cuyas salidas potencialmente podrán producir *glitches* durante un régimen transitorio de duración acotada por el tiempo de propagación del conjunto de bloques combinacionales.

2. Cuando las salidas del combinacional actúan sobre dispositivos de visualización o electromecánicos.

3. Cuando las salidas del combinacional se registran en *flip-flops* una vez que el circuito ha alcanzado el régimen permanente de funcionamiento.

El siguiente ejemplo ilustra algunos de los casos mencionados anteriormente. La figura EP.1 muestra un circuito que sirve para controlar la validez de las monedas insertadas en una máquina de dispensación automática de productos de consumo. En esta máquina se admite el pago en euros mediante cualquier moneda de curso legal.

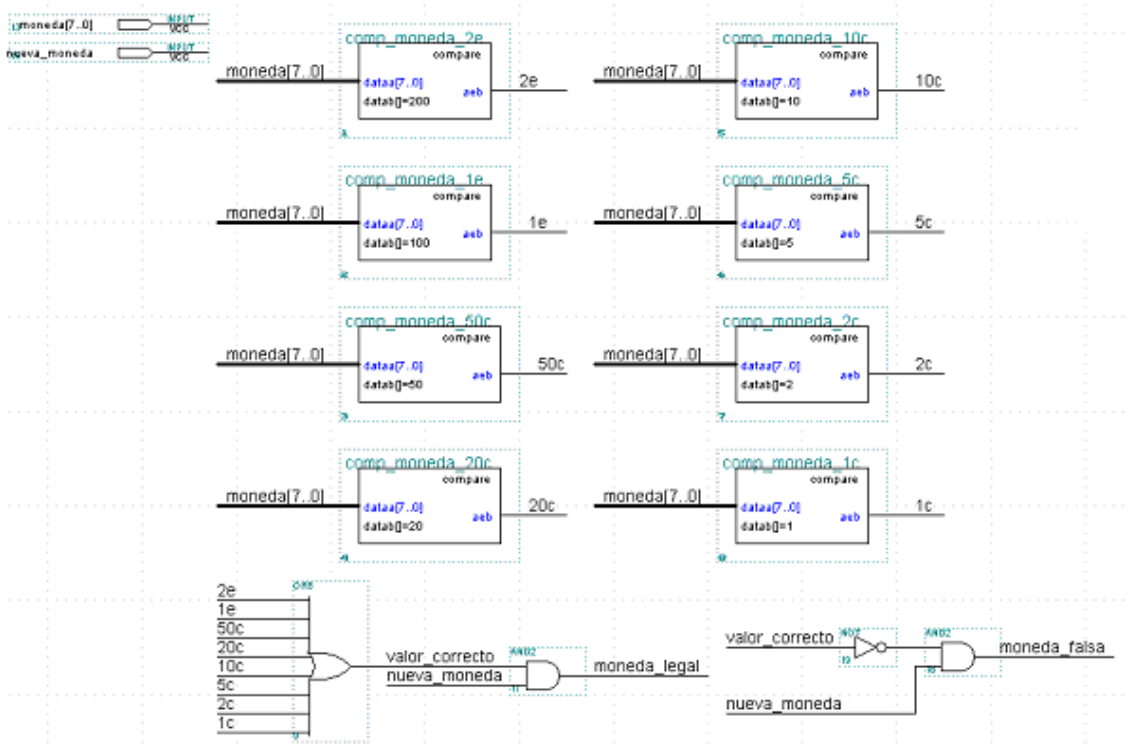

Figura EP.1

La máquina dispone de un mecanismo que analiza las monedas introducidas para determinar su valor y validez: cada vez que el mecanismo detecta una nueva moneda genera un pulso y, durante el mismo, se indica el valor, en céntimos de euro, de las monedas válidas, o un valor distinto –que no se especifica y puede variar en cada caso, pero nunca coincidir con el valor de una moneda de curso legal- si no es reconocida como tal. El valor nominal de la moneda se codifica en binario natural. Las entradas del circuito **moneda[7..0]** y **nueva_moneda** están conectadas a las salidas del mecanismo de análisis de monedas.

El circuito compara el valor de la moneda con los valores legales para generar dos señales que indican si la moneda introducida es válida o no. Se trata evidentemente de un circuito combinacional y es posible, por tanto, que genere *glitches* en sus salidas.

En la figura EP.2 se muestra parte del resultado de una simulación con retardos del circuito para una realización del mismo sobre una FPGA EPF10K30RC208-4 de ALTERA. Los *glitches* que se observan en la simulación resultan irrelevantes, por su corta duración, si las salidas del circuito se conectan a LEDs –para indicar al comprador, por ejemplo, la aceptación o rechazo de la última moneda introducida- pero pueden causar problemas si actúan sobre otros módulos afectando a la operación de *latches* o *flip-flops*.

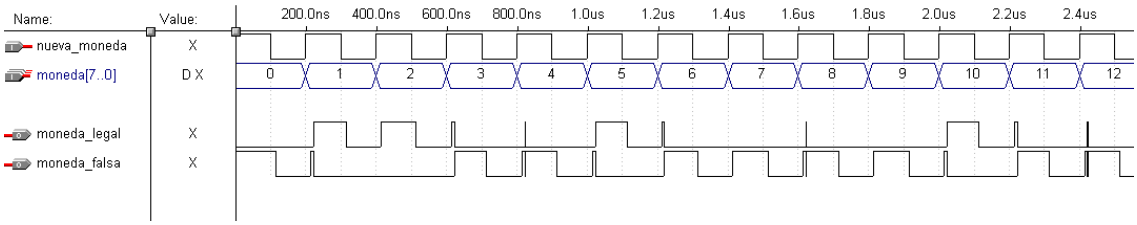

Figura EP.2

El circuito de la figura EP.3 utiliza el módulo anterior, denominado aquí **ctrl_monedas**, para, por un lado calcular el valor acumulado de las monedas introducidas para adquirir un producto y, por otro, detectar intentos de fraude, o mal uso de la máquina, debido a la inserción reiterada de monedas inválidas: cuando se dan cuatro casos de detección de monedas incorrectas se genera una señal que puede servir para bloquear la máquina o avisar de un uso “sospechoso” de la misma.

Este circuito es un buen ejemplo de cómo los *glitches* pueden provocar discrepancias entre el funcionamiento real e ideal (sin retardos o considerando retardos sin *glitches*) de un sistema digital –y también de prácticas de diseño que jamás deben aplicarse. La salida **moneda_legal** del módulo **ctrl_monedas** se utiliza como reloj del acumulador que lleva la cuenta del importe acumulado (salida **valor_acumulado[8..0]**), bajo el supuesto –evidentemente erróneo- de que sólo va a haber un flanco de subida cada vez que se detecte una moneda correcta y que, en consecuencia, se acumulará su valor correctamente. La otra salida del módulo, **moneda_falsa**, es también el reloj (por una suposición análoga a la anterior e igualmente equivocada) de un contador que sirve para detectar el número máximo (cuatro) de monedas incorrectas que provocan el disparo de la salida **alarma_bloqueo**; esta señal, que debe abortar el proceso de admisión de monedas, actúa también sobre el reset asíncrono del registro del acumulador (en combinación con un reset externo que también puede realizar la misma función, **resetn_acum**). Esta señal puede tener también *glitches*, pues se genera a partir

de la señal **moneda_falsa** y las salidas del contador y es posible, por tanto, que dé lugar a inicializaciones no deseadas. Si la lógica combinacional del circuito no produjera *glitches* el funcionamiento del sistema sería correcto: ante la detección de una moneda, o bien se acumula el valor de la misma, en el flanco de subida de la señal **moneda_legal**, o bien se incrementa la cuenta de monedas incorrectas detectadas (flanco de subida de **moneda_falsa**); el reset del acumulador se produciría cuando se detectan cuatro monedas inválidas por la activación, a nivel bajo, de la señal **ini_acum_n**.

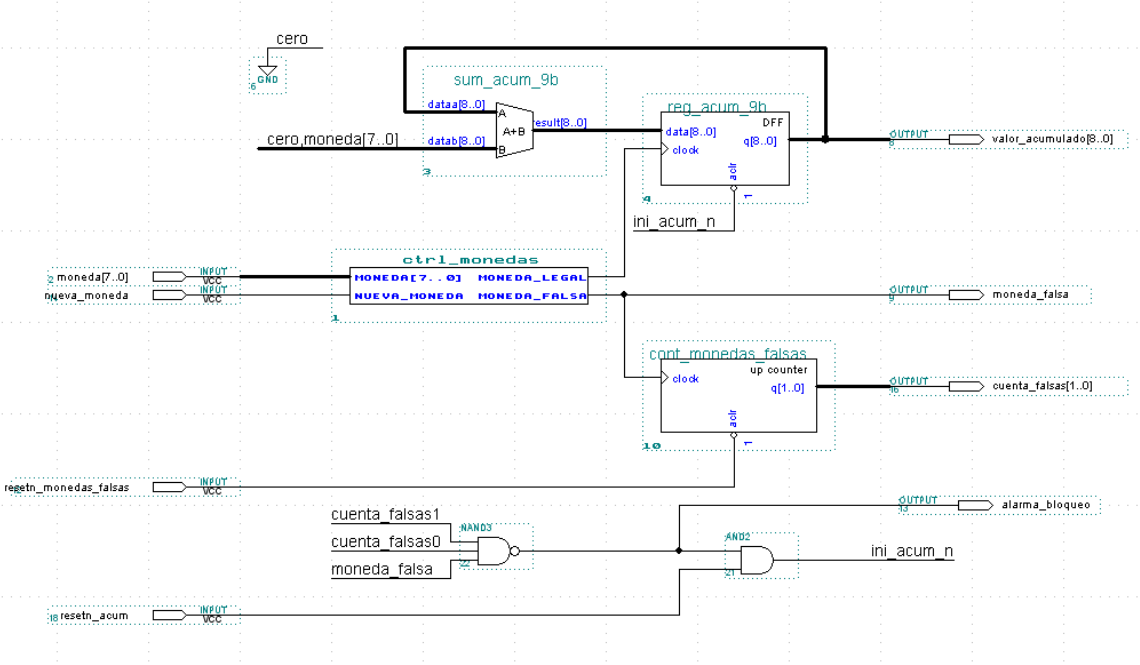

Figura EP.3

La figura EP.4 muestra el resultado de una simulación ideal (sin retardos) del funcionamiento del circuito en respuesta a la inserción de ocho monedas, siendo rechazadas como inválidas dos de ellas (en 1.6 y 2.2 microsegundos).

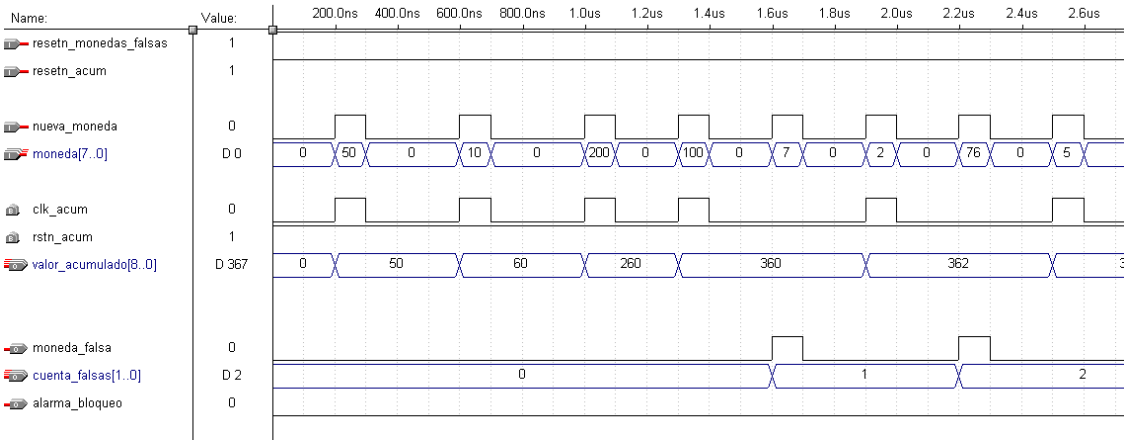

Figura EP.4

La figura EP.5 muestra, mediante la realización de una simulación con retardos, como podría ser el funcionamiento real del circuito en una FPGA de ALTERA.

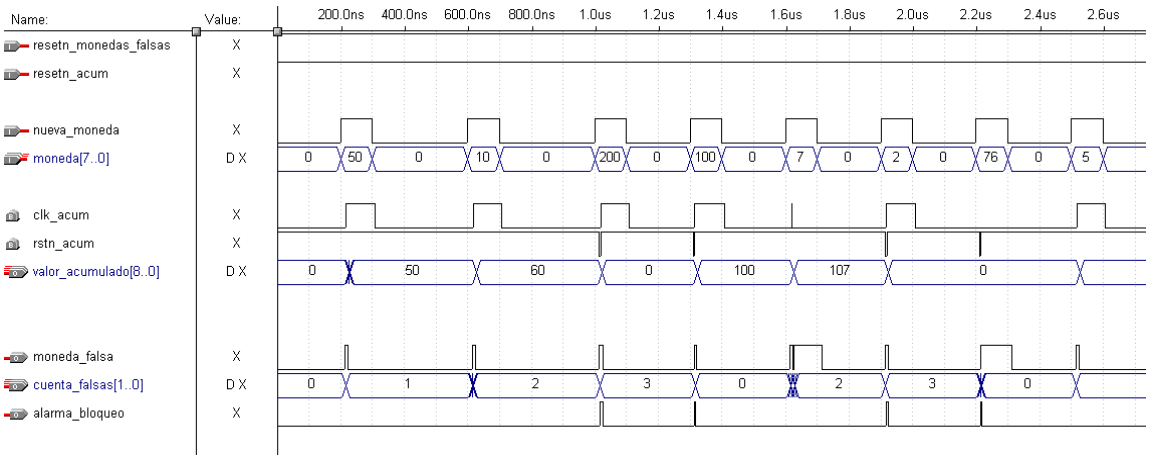

Figura EP.5

Se observan claramente tres anomalías:

1. Los *glitches* en la señal **moneda_falsa** provocan detecciones incorrectas de monedas falsas y, en consecuencia, incrementos indeseados del contador.

2. Los *glitches* en **clk_acum** (**moneda_legal**) dan lugar a que se acumulen los valores asociados a monedas no válidas (en 1.6 microsegundos, por ejemplo, se acumula el valor 7).

3. Los *glitches* en la señal que inicializa asíncronamente el registro del acumulador, **rstn_acum** (**ini_acum_n** en el esquema) provocan *resets* indeseados (en 1 microsegundo, por ejemplo).

El efecto de los *glitches* en este circuito es un funcionamiento potencialmente caótico e impredecible. El funcionamiento calculado por el simulador no representa exactamente el del circuito real, ya que el simulador utiliza unos valores de retardos (en el caso del Max+plus II, los máximos) para sus cálculos que no van a coincidir exactamente con los de la FPGA en funcionamiento –ya se ha comentado que los retardos dependen de múltiples factores y cambian con las condiciones de funcionamiento- y, además, los *glitches* pueden provocar violaciones de tiempos en los *flip-flops* (por ser demasiado estrechos o porque no respeten tiempos de *set-up* o *hold*) que plantean una cierta indeterminación sobre el comportamiento de estos. A pesar de esto la simulación con retardos pone en cuestión que el circuito funcione como el diseñador desea.

En el siguiente ejemplo se muestran otros casos interesantes. El circuito de la figura EP.6 realiza la suma de un dato de cuatro bits almacenado en un registro y la salida de un contador binario, también de cuatro bits. El contador es síncrono y tiene el mismo reloj que el registro, por tanto las entradas del sumador (**Q[3..0]** y **D[3..0]**) sólo pueden cambiar en los flancos de subida del reloj del circuito.

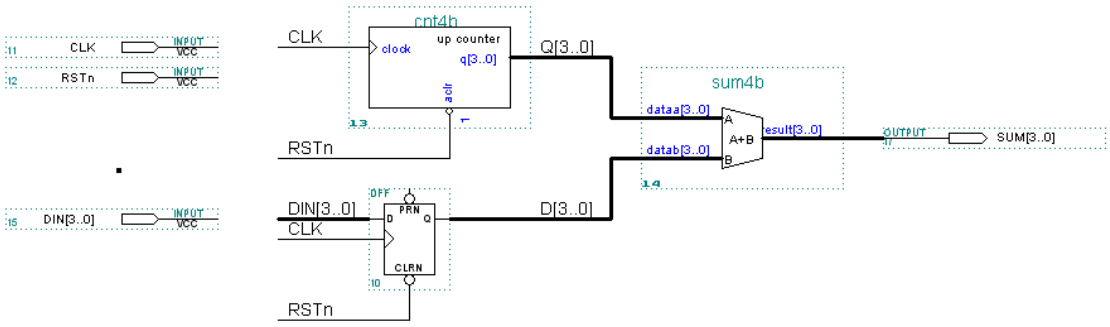

Figura EP.6

El módulo sumador es un circuito combinacional y, por tanto, en sus salidas pueden producirse *glitches*. La figura EP.7 muestra una simulación con retardos para una realización del circuito con FPGAs de ALTERA en la que, efectivamente, aparecen pulsos espurios durante el régimen transitorio del sumador.

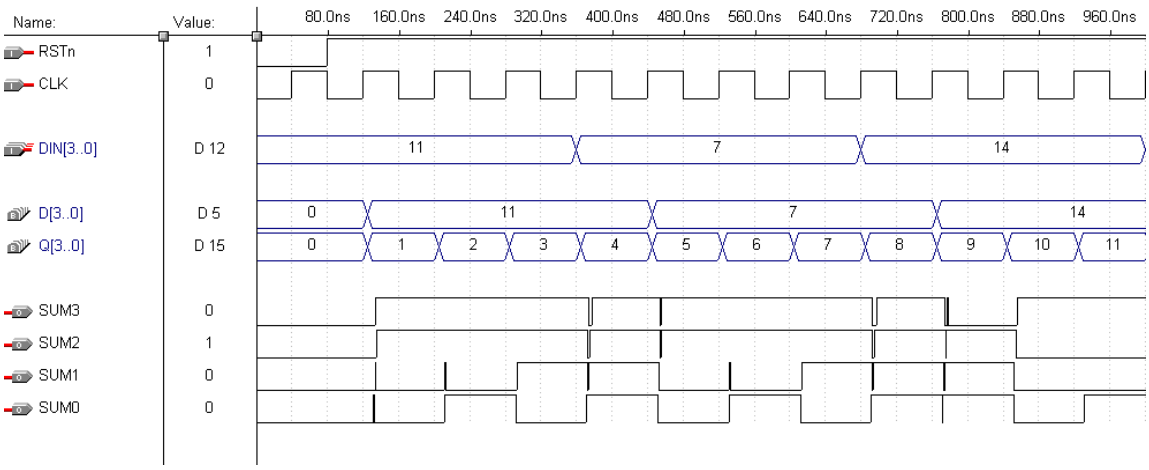

Figura EP.7

Un detalle muy interesante de este circuito es que, puesto que las entradas del sumador sólo pueden cambiar en los flancos de subida del reloj, el sumador alcanza el régimen permanente, y sus salidas están, por tanto, libres de *glitches* cuando después de un flanco ha transcurrido un tiempo mayor o igual a su tiempo de propagación. Si transcurrido ese tiempo se registran las salidas del sumador, se obtendrá una "copia" sin *glitches* de la salida de éste. La figura EP.8 muestra una versión del circuito en la que se registran las salidas del sumador con el mismo reloj con el que se ponen los datos en sus entradas, de este modo se dispone de un periodo de reloj para que el sumador alcance el régimen permanente y se registre el resultado correcto.

El registro de salida elimina los *glitches*, pero también retarda la operación: el resultado de la suma correspondiente a dos sumandos establecidos en un determinado flanco de reloj no es accesible en la salida registrada hasta el siguiente flanco, en consecuencia el "retardo efectivo" del sumador pasa a ser un periodo de reloj.

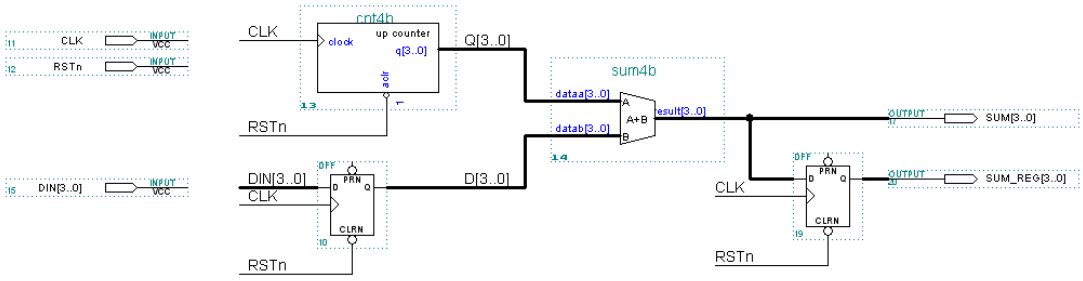

Figura EP.8

En la figura EP.9 se muestra el resultado de una simulación con retardos de la versión modificada del circuito. Puede observarse como, efectivamente, la salida **SUM_REG** es una versión libre de *glitches* y retardada de la salida del sumador (**SUM**). El reloj tiene un periodo de 80 ns que podría reducirse si resultara necesario que el circuito fuese más rápido, el límite de la reducción vendría impuesto por el retardo del sumador.

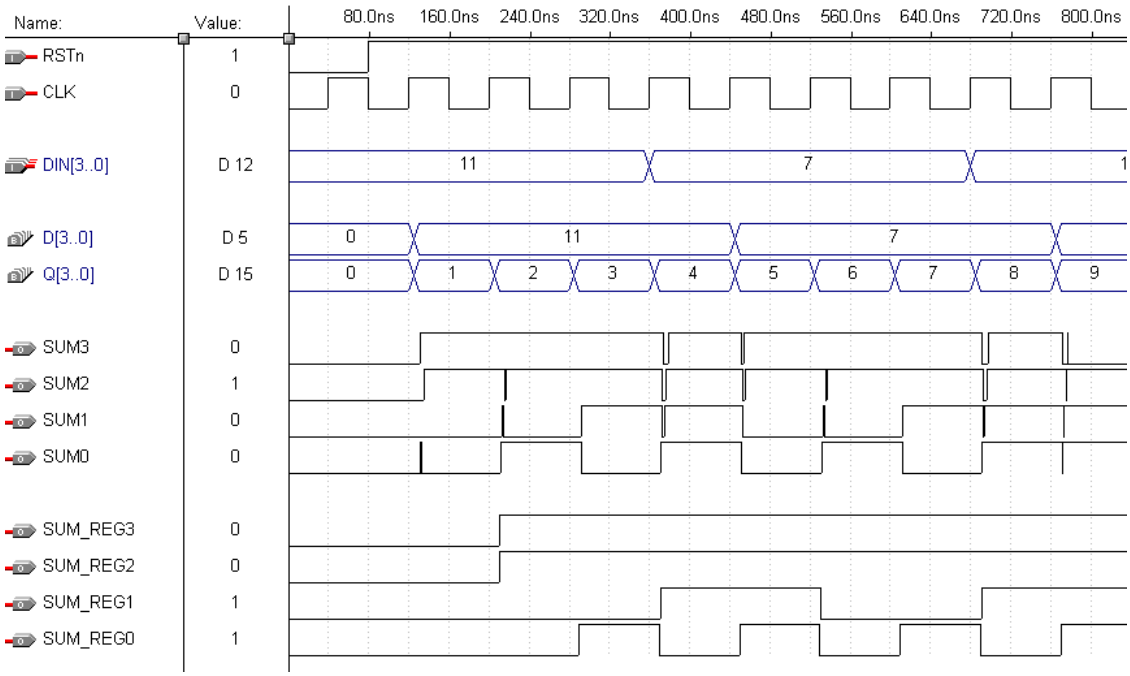

Figura EP.9

El circuito de la figura EP.8 muestra un procedimiento para generar señales libres de *glitches*: para aplicarlo resulta imprescindible tener la seguridad de que cuando se registran las salidas del circuito combinacional éste entrega valores válidos y estables –es decir, ha transcurrido su tiempo de propagación y se encuentra en régimen permanente.

2.- Diseño Síncrono

En este capítulo se presentan y justifican las reglas de diseño síncrono (apartado 2.2), se analiza la arquitectura genérica de los circuitos síncronos (apartado 2.3), se describe el modo de operación de las estructuras que pueden distinguirse en dicha arquitectura (apartado 2.4) y, después, se deduce la expresión de la frecuencia máxima de reloj con que puede operar un circuito síncrono ideal -sin skew de reloj- (apartado 2.5). En el apartado 2.6 se analiza el efecto del skew de reloj sobre la operación de un circuito síncrono y su impacto sobre la frecuencia máxima de funcionamiento. Finalmente, en el apartado 2.7, se realiza un breve análisis de las arquitecturas síncronas que funcionan con dos fases de reloj.

2.1 Introducción

Como hemos visto en el capítulo anterior, la existencia de *glitches* en las salidas de los circuitos combinacionales puede ocasionar errores en el funcionamiento de los sistemas digitales. La realización de circuitos combinacionales libres de *glitches* no es una opción razonable cuando se está abordando el diseño de un sistema digital complejo: implica establecer restricciones sobre el modo en que conmutan las entradas de los combinacionales, para que sólo pueda modificarse un bit en cada fase de conmutación, y realizar, para que dicha restricción tenga efecto, un diseño libre de *glitches* lógicos. Cumplir estas condiciones supone complicar enormemente el diseño del circuito, de modo que es mejor elegir otro tipo de alternativas.

La mejor solución a los problemas de funcionamiento ocasionados por los *glitches* consiste en asumir su existencia y adoptar normas de diseño que, estableciendo una serie de restricciones al modo en que pueden construirse los circuitos, garanticen que los *glitches* no puedan ser causa de errores de funcionamiento. Las condiciones que deben cumplirse en el diseño de un circuito para que esto ocurra son tres:

1. La salida de un circuito combinacional no debe conectarse nunca a la entrada de reloj o las entradas asíncronas de un *flip-flop* o un *latch*.

2. Cuando la salida de un circuito combinacional está conectada a la entrada síncrona de un *flip-flop* debe tenerse la certeza de que el combinacional se encuentra en su régimen permanente de funcionamiento siempre que llegue un flanco activo de reloj al *flip-flop*.

3. Cuando es necesario “eliminar” los *glitches* que aparecen en una salida combinacional del circuito, la salida deberá registrarse.

Respetando estas tres normas pueden realizarse circuitos en los que la presencia de *glitches* resulte inocua. Pero las metodologías para la realización de circuitos digitales complejos tienen que ocuparse también de otras cuestiones que afectan a la funcionalidad y prestaciones de los circuitos. Las técnicas (o metodología) de diseño síncrono son un conjunto de reglas de diseño que facilitan la solución de diversos problemas de diseño que surgen al abordar la realización de circuitos complejos. Su uso conlleva ventajas relevantes: por un lado los circuitos resultantes de su aplicación

resultan simples (en términos de análisis y diseño funcional) y fiables (en cuanto a su insensibilidad a los *glitches*), por otro lado, no menos importante, la metodología de diseño síncrono facilita el cálculo de las prestaciones del circuito (la velocidad de procesamiento y la frecuencia de funcionamiento) y la detección de los componentes del sistema que las determinan. A los circuitos realizados utilizando estas técnicas se los denomina –como cabría esperar- circuitos síncronos

2.2 Reglas de diseño síncrono

La metodología de diseño síncrono se materializa en una serie de reglas para el diseño de circuitos -a los que denominaremos, en adelante, síncronos. Cada regla es una recomendación o prohibición que afecta a la estructura del circuito y repercute sobre su modo de funcionamiento. La aplicación de las reglas determina un conjunto de características comunes a todos los circuitos síncronos.

En este apartado vamos a enumerar y comentar brevemente las reglas de diseño; más adelante se tratará la problemática que plantea su cumplimiento. Resulta necesario advertir al lector que alguna de las reglas que se presentan es de aplicación genérica en el área del diseño de sistemas digitales –es decir, debe cumplirse también en circuitos que no se realizan siguiendo la metodología de diseño síncrono- y se trae aquí con la finalidad de exponer de manera completa las condiciones que debe cumplir un buen diseño. En fin, el conjunto de normas que deben cumplirse para realizar el diseño de un sistema síncrono es el siguiente:

1. **Todos los circuitos secuenciales deben realizarse utilizando *flip-flops*, preferentemente de tipo D, sensibles al mismo flanco de reloj.**

   El uso de *flip-flops* en lugar de *latches* es una regla general de diseño, por los problemas que acarrea el uso de *latches* en la realización de circuitos secuenciales con lógica de cálculo del estado futuro –con realimentación combinacional desde la salida hacia la entrada. La elección del tipo D se debe a que facilita el análisis y la síntesis de los circuitos; además, hoy en día es el único tipo de *flip-flop* disponible en muchas tecnologías para la realización *hardware* de circuitos.

   La obligación de que todos los *flip-flops* sean sensibles al mismo flanco de reloj es una recomendación –algo menos que una regla- de las técnicas de diseño síncrono, ya que aunque es posible plantear el diseño de un sistema síncrono en que unos *flip-flops* sean activos por flanco de subida y otros por el de bajada, esta situación no aporta en la mayoría de los casos ninguna ventaja, sino que, más bien al contrario, complica el diseño y entorpece el análisis del funcionamiento y rendimiento del circuito. La realización de circuitos síncronos mezclando *flip-flops* activos por flanco de subida y de bajada, y los problemas que conlleva, se analizará de todos modos en el último apartado de este capítulo.

2. **Las entradas asíncronas de los *flip-flops* del circuito sólo pueden ser manejadas por una señal de inicialización (*Reset*) global y nunca durante la operación normal del sistema.**

Nota: En el caso de que la señal de *Reset* global opere sobre las entradas asíncronas del circuito debe garantizarse que su tiempo mínimo de activación sea mayor que un periodo del reloj del circuito.

El objetivo de esta regla es impedir que durante el funcionamiento normal del circuito ningún *flip-flop* pueda conmutar su salida en un instante distinto al de ocurrencia de un flanco activo de reloj (véase la regla número 3). En el arranque o en una reinicialización se permite su uso por simplicidad y economía de diseño y porque no compromete el buen funcionamiento del circuito, ya que una activación de duración adecuada de la señal de *Reset* global permite a los módulos secuenciales del circuito alcanzar sus estados iniciales y a los combinacionales estabilizar los niveles lógicos en sus salidas, de forma que todo el sistema quede preparado para el paso al modo de funcionamiento síncrono cuando la señal de *Reset* se desactive.

Aunque es la solución más sencilla y frecuente, no es obligatorio que la señal de *Reset* global actúe sobre las entradas asíncronas de los *flip-flops*; en módulos secuenciales donde se disponga de una entrada síncrona de *reset*, que lleve al circuito al estado de arranque, puede aprovecharse esta entrada para que actúe sobre ella el *Reset* global, con esta medida se puede conseguir un cierto ahorro *hardware* en algunas tecnologías (en concreto, en el diseño de ASICs con *standard cells*, porque un *flip-flop* con reset asíncrono ocupa más espacio que uno sin él; en el caso de la lógica programable no se consigue ninguna ventaja, al contrario, es posible que esta opción suponga una penalización en el número de recursos utilizados).

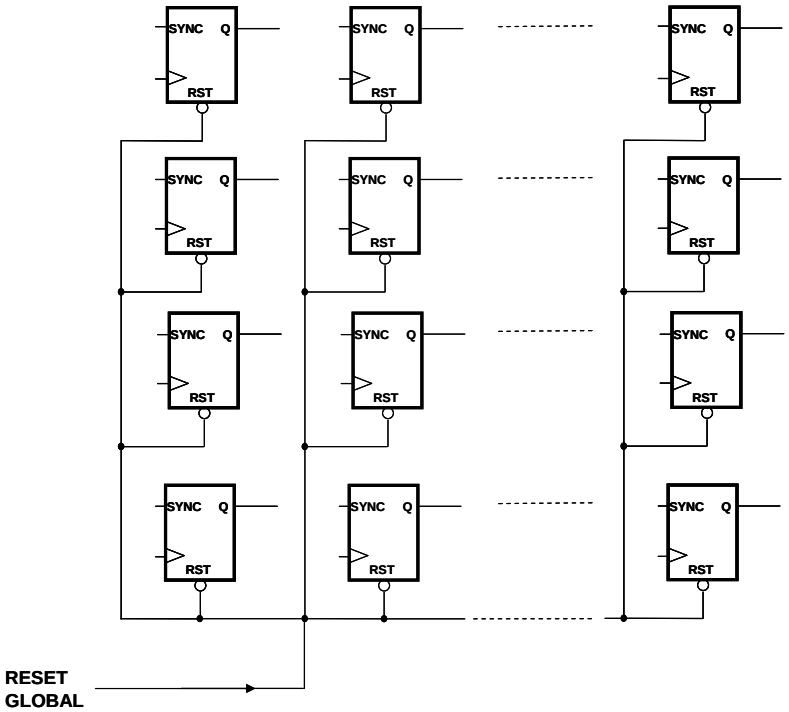

Figura 12. Reset Global

La entrada global de *Reset* del circuito suele tener un *fan-out* muy alto (Figura 12), por lo que se precisa del uso de *drivers* para poder actuar sobre

todos los *flip-flops* del circuito que sea necesario inicializar. En el caso de las FPGAs, y otros chips VLSI configurables, existen entradas dedicadas para la señal global de *Reset* que disponen de una red de interconexión separada de la del resto del circuito. Esta red es capaz de actuar sobre las entradas asíncronas de *Preset* o *Reset* de todos los *flip-flops*, manteniendo equilibrados los retardos de propagación para que la señal llegue simultáneamente a todos ellos –éste es el motivo por el que en los dispositivos lógicos programables no resulta ventajosa la posibilidad de que el *Reset* global se realice síncronamente.

3. **El circuito debe disponer de una señal de reloj única y común para todos los *flip-flops* del circuito.**

Nota: Es un objetivo de diseño que esta señal de reloj se distribuya de modo que el retardo con el que llega a las entradas de todos los *flip-flops* del circuito sea el mismo –en caso contrario el instante en que actúan los flancos sobre éstos sería distinto y el sistema dejaría de ser propiamente síncrono.

Nota: El único uso de la señal de reloj del circuito es el control del funcionamiento síncrono de los *flip-flops*.

Esta es la regla fundamental del diseño síncrono, y la que condiciona el modo de funcionamiento característico de los circuitos síncronos. Cumplirla implica que la ocurrencia de un flanco activo de reloj es percibida simultáneamente por todos los *flip-flops* del circuito y, puesto que sus entradas asíncronas no se usan (salvo en el arranque o tras un *reset* de reinicialización), los instantes de captura de datos y conmutación de las salidas son exactamente los mismos para todos los *flip-flops* y se suceden periódica e indefinidamente, al ritmo marcado por la frecuencia de reloj del circuito, mientras el sistema opera. De este modo de funcionamiento se deriva la simplicidad del funcionamiento de los circuitos síncronos y, en última instancia, su inmunidad a los *glitches* generados por los circuitos combinacionales.

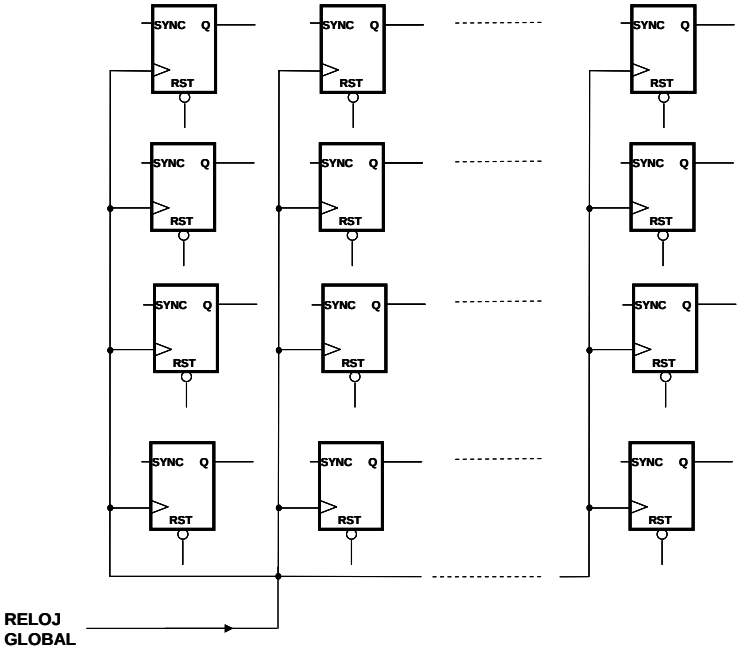

Figura 14. Red de distribución del reloj

El alto *fan-out* de la señal de reloj (Figura 14), y la necesidad de que el retardo con que llegue a todos los *flip-flops* sea el mismo, hace que el diseño de su red de interconexión resulte delicado. Al igual que en el caso de la señal global de *reset*, las FPGAs y otros dispositivos lógicos programables disponen de entradas especiales –entradas dedicadas para relojes globales- con capacidad para atacar a todos los *flip-flops* del chip con una dispersión de retardos mínima. En cualquier caso hay que entender que es imposible ajustar exactamente los tiempos de retardo para todos los *flip-flops* y el objetivo real debe ser que el desfase sea mínimo: la diferencia entre los retardos de distribución de reloj a dos *flip-flops* de un circuito se conoce con el nombre de *skew*; como se vera más adelante su valor puede afectar a la frecuencia máxima de funcionamiento del circuito e incluso, si es demasiado grande, puede ser causa de errores de funcionamiento.

Una última cuestión que debe quedar clara sobre la señal de reloj es que en un circuito síncrono está conectada directamente a todos los *flip-flops* del circuito y sólo a ellos, lo que supone una prohibición expresa de hacer cualquier otro uso de ella.

4. Las entradas de los circuitos combinacionales de un sistema síncrono deben estar conectadas a salidas de circuitos secuenciales gobernados por la señal de reloj (o a salidas de circuitos combinacionales que cumplan esta condición).

Esta regla determina que el régimen transitorio de los circuitos combinacionales sólo puede iniciarse en los flancos activos de reloj; gracias a ello puede garantizarse que una vez transcurrido un cierto tiempo desde la ocurrencia del flanco de reloj todos los circuitos combinacionales del sistema síncrono tendrán salidas estables y libres de *glitches* (figura 15). Una cota para ese tiempo es el máximo de entre los tiempos de propagación del conjunto de circuitos combinacionales que haya en el sistema síncrono, es decir, el combinacional con mayor retardo del circuito es el último que alcanza el régimen permanente de funcionamiento y el que determina, por tanto, que las salidas de todos son válidas.

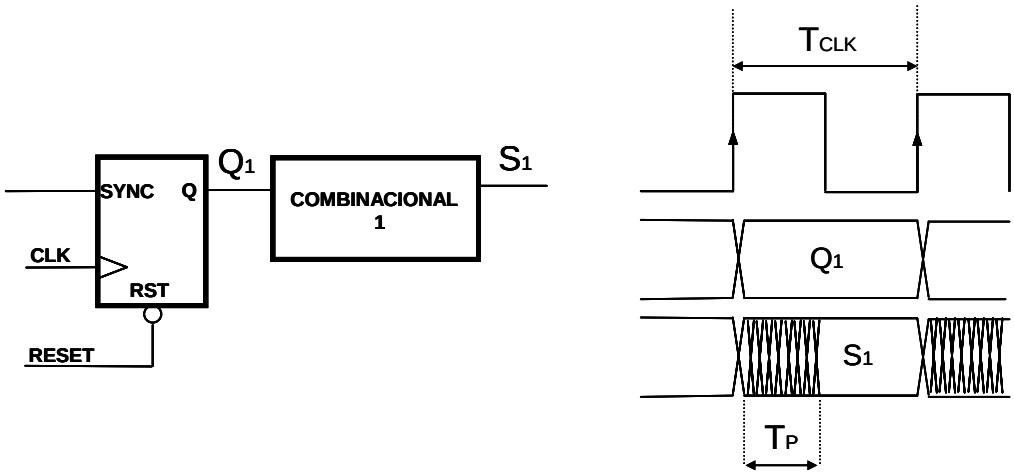

Figura 15. Entradas registradas en todos los circuitos combinacionales

Esta norma no impide el encadenamiento de módulos combinacionales, pero sí establece que las entradas del primer (o primeros) módulos de un grupo de circuitos combinacionales interconectados tienen que ser salidas de *flip-flops*. En relación con esto, hay que tener en cuenta que el encadenamiento de combinacionales puede dar lugar a estructuras con tiempos de retardo grandes que penalicen la velocidad de estabilización del sistema (Figura 16).

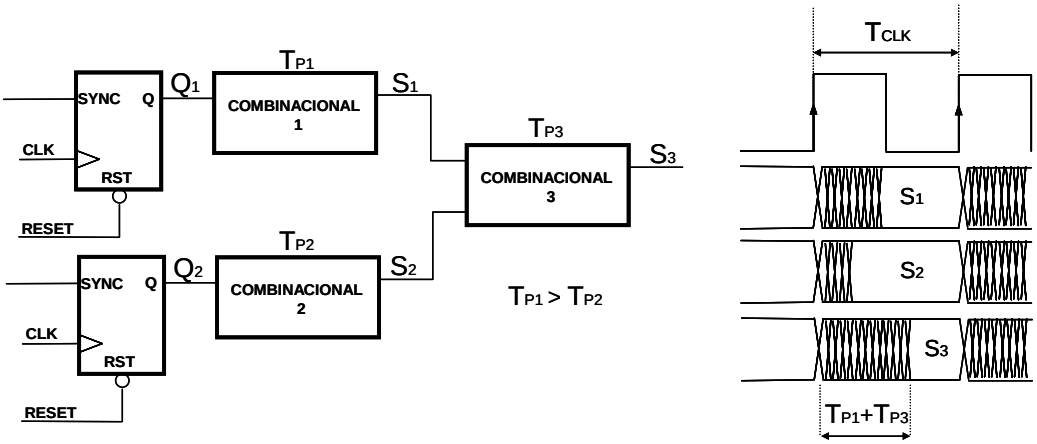

Figura 16. Módulos combinacionales encadenados

Hay que resaltar, por último, que una consecuencia importante de esta regla es que cualquier entrada de un sistema síncrono tiene que ser registrada antes de poder actuar sobre la entrada de un combinacional (Figura 17).

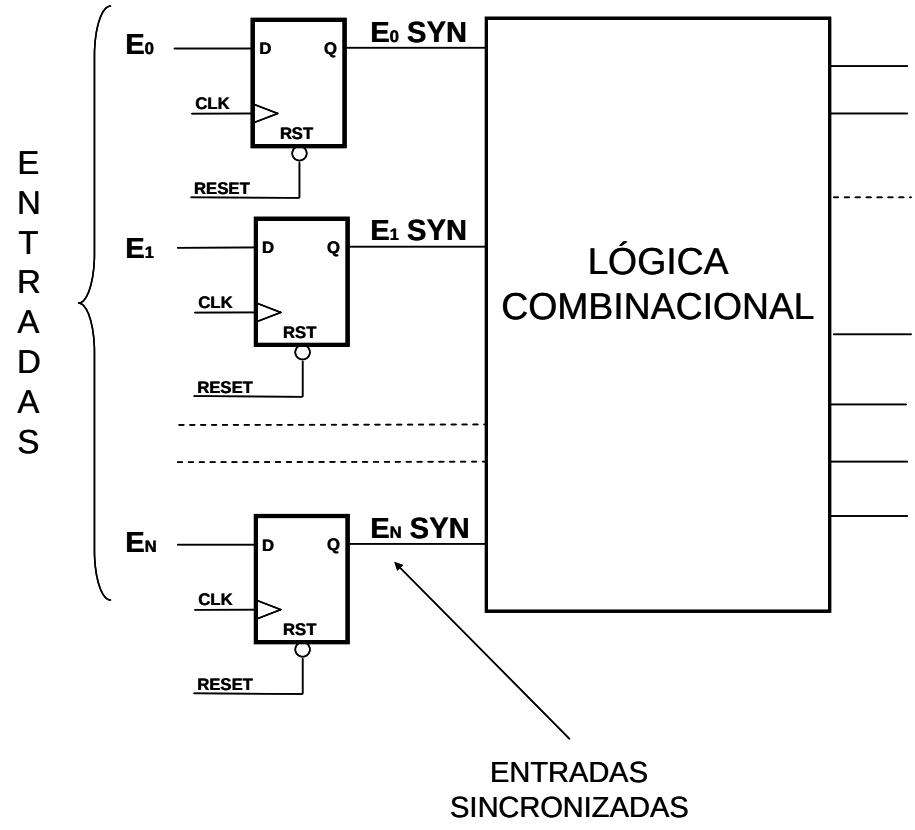

Figura 17. Entradas registradas

5. **Las entradas síncronas de un circuito secuencial deben ser manejadas por salidas de circuitos secuenciales gobernados por la señal de reloj o por salidas de combinacionales que cumplan la regla número 4.**

Esta regla puede parecer redundante con la anterior, pero no lo es: obliga a que las entradas al sistema también tengan que registrarse para poder actuar sobre las entradas síncronas de circuitos secuenciales -la combinación de ambas reglas (4 y 5) obliga a que cualquier entrada a un sistema síncrono deba registrarse.

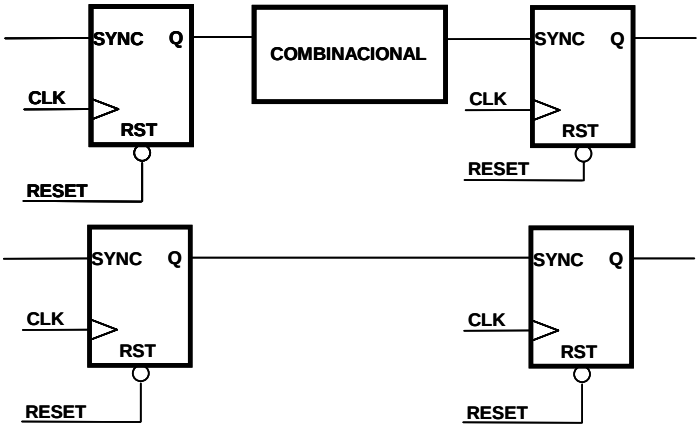

Figura 18. Entradas síncronas de circuitos secuenciales

La necesidad de que las entradas síncronas de los secuenciales sean, en definitiva, señales síncronas del sistema (Figura 18), se deriva de la necesidad de garantizar su estabilidad en los flancos activos de reloj, con el fin de asegurar que se respeten los tiempos de *hold* y *set-up* de los *flip-flops* de los módulos secuenciales.

6. **Las salidas de un módulo combinacional sólo pueden conectarse a entradas de otros combinacionales y a entradas síncronas de secuenciales; alternativamente, pueden ser salidas del circuito síncrono.**

Esta regla impide implícitamente la realización de tres tipos de conexiones en las salidas de los módulos combinacionales: con la entrada de reloj o las entradas asíncronas de cualquier *flip-flop* (también prohibidas por las reglas 2 y 3) y, por último, con cualquier entrada del mismo circuito combinacional (Figura 19), es decir, prohibe la utilización de *latches* asíncronos (lógica combinacional realimentada).

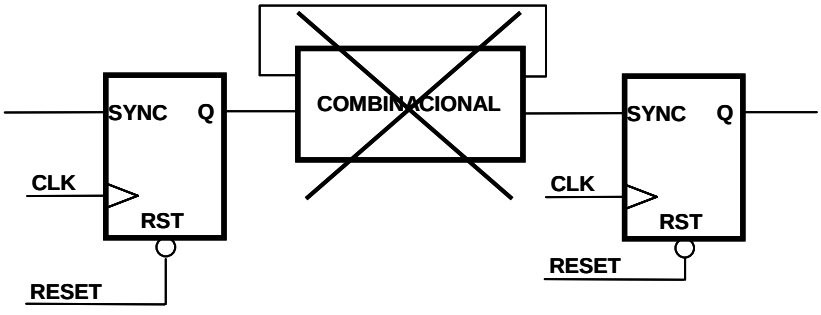

Figura 19. Realimentaciones combinacionales prohibidas

La exclusión de los *latches* asíncronos como recurso de diseño se debe a que su régimen transitorio no puede, en general, acotarse –y, por tanto, no se puede garantizar que sus salidas vayan a ser estables cuando actúen sobre entradas síncronas de *flip-flops*- y porque es posible sustituirlos por circuitos secuenciales síncronos.

Entre las reglas de diseño síncrono suelen incluirse también dos restricciones que se derivan de las reglas enunciadas anteriormente y que ya hemos mencionado, las enunciamos seguidamente como reglas 7 y 8.

7. **Las entradas de un sistema síncrono deben registrarse con *flip-flops* manejados por el reloj del sistema.**

8. **Esta prohibido utilizar *latches*, síncronos o asíncronos.**

2.3 Arquitectura de los circuitos síncronos

La arquitectura de cualquier circuito digital complejo consiste en un conjunto de módulos combinacionales y secuenciales interconectados inteligentemente para construir un sistema de procesamiento capaz de realizar una determinada función. La metodología de diseño síncrono resulta idónea para abordar la ingeniería de diseño de este tipo de sistemas y debe su nombre a que una de las condiciones que impone –y que caracteriza de manera mas significativa el resultado del diseño- es que todos los bloques secuenciales del sistema deben ser síncronos, es decir, deben tener una señal de reloj única y común –también por este motivo a los circuitos realizados siguiendo esta metodología se denominan circuitos o sistemas síncronos.

Como hemos visto, una de las reglas básicas de las técnicas de diseño síncrono es que las entradas de cualquier bloque combinacional -que podrá consistir, bien en un módulo funcional, bien en un grupo de módulos encadenados- deben estar conectadas a salidas de bloques secuenciales manejados por la señal de reloj del sistema. Esto es debido a que el diseño síncrono aborda la eliminación de los problemas asociados a la presencia de *glitches* obligando a que las entradas de los circuitos combinacionales cambien en sincronía con el reloj del sistema, para que su régimen transitorio quede acotado y sus salidas puedan operar con garantías de seguridad sobre las entradas síncronas de otros bloques secuenciales.

La combinación de estas normas de diseño permite esbozar un modelo general para la arquitectura de los circuitos síncronos: son una red interconectada de bloques combinacionales y secuenciales síncronos, realizados con *flip-flops* y con un reloj común, en la que las entradas de los circuitos combinacionales son salidas de bloques secuenciales, mientras que sus salidas están conectadas a entradas síncronas de bloques secuenciales o, si no, a salidas del sistema. Añadamos, para completar la descripción, que habitualmente el circuito contará con una señal de inicialización única cuyo propósito es la realización de un *reset* para el arranque del circuito. En la figura 20 se muestra un esquema de esta estructura genérica de un circuito síncrono.

Los bloques secuenciales que aparecen en la figura 20 se representan como cajas conectadas a dos entradas globales: la señal de reloj (CLK) y la señal de inicialización

del circuito (RESET). A estas entradas se las denomina globales precisamente porque están conectadas a todos los bloques secuenciales del circuito (en un circuito complejo, a cientos o miles de *flip-flops*). El *fan-out* de estas señales es, evidentemente, muy grande y da lugar a que la realización de la red de pistas y *drivers* que las conecta al circuito tenga que ser objeto de un diseño especial.

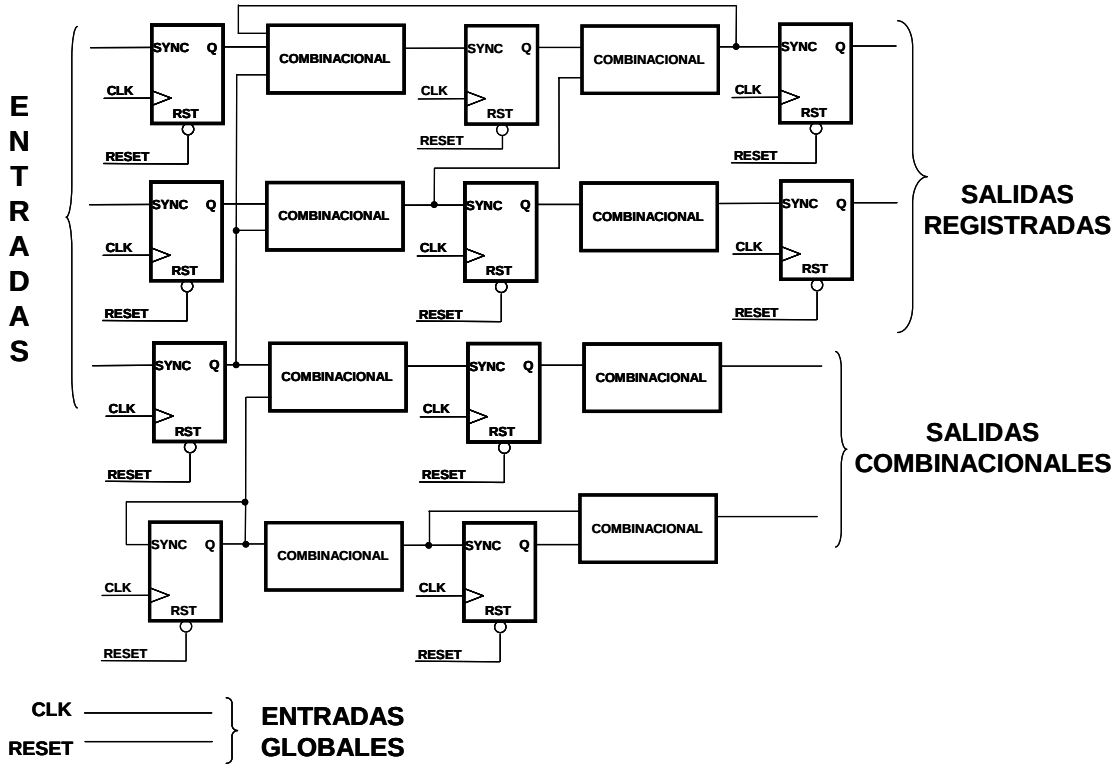

Figura 20. Arquitectura de un circuito síncrono

Otra característica estructural de los circuitos síncronos representada en la figura 20 es que las entradas externas nunca están conectadas directamente a entradas de módulos combinacionales. Antes de poder actuar sobre un circuito combinacional, una entrada manejada desde el exterior del sistema debe pasar, al menos, por un *flip-flop* gobernado por el reloj del circuito, para obtener una versión sincronizada –que sólo puede conmutar en sincronía con el reloj del circuito- de la entrada. Las entradas de un sistema síncrono se consideran asíncronas con él –y se denominan por ello comúnmente, en el contexto del diseño síncrono, entradas asíncronas- porque pueden cambiar de valor en instantes distintos a los flancos activos del reloj del circuito –no debe confundirse este término con el homónimo que se utiliza para distinguir en los bloques secuenciales las entradas síncronas y asíncronas: son dos conceptos que no guardan ninguna relación entre sí.

En la figura 20 se ilustra también el modo en que pueden interconectarse los circuitos combinacionales y secuenciales dentro del sistema síncrono. Observe que las entradas de un circuito combinacional pueden ser tanto las salidas de circuitos secuenciales, como las de otros circuitos combinacionales cuyas entradas estén manejadas, a su vez, por salidas de módulos secuenciales. Este tipo de interconexiones da lugar a la creación de una cadena de dos o más circuitos combinacionales que, en

definitiva, equivalen a un solo circuito combinacional, con una funcionalidad que resulta ser una composición de la de los circuitos asociados y con un retardo total mayor.

Para terminar con la revisión de los aspectos estructurales de los circuitos síncronos ilustrados en la figura 20, hay que apuntar un detalle sobre las salidas del sistema: puede haber salidas que están conectadas directamente a las de circuitos combinacionales internos (salidas combinacionales) y otras que están conectadas a salidas de *flip-flops* (salidas registradas). La elección de uno u otro tipo depende, normalmente, de la necesidad o no de eliminar *glitches*; en general una salida combinacional sólo debe registrarse –y convertirse, en consecuencia, en salida registrada- cuando su uso externo o la temporización del circuito lo exija, en otro caso es un desperdicio innecesario *flip-flops*.

En el siguiente ejemplo puede observarse materializada la arquitectura genérica de los sistemas síncronos presentada anteriormente. El esquema de la figura EP.10 muestra la arquitectura de un circuito síncrono que genera señales cuadradas de periodo y ciclo de trabajo programable. El circuito cuenta con dos entradas globales de reloj y reset (CLK y nRST) que están conectadas a todos los flip-flops (42, en total) del circuito.

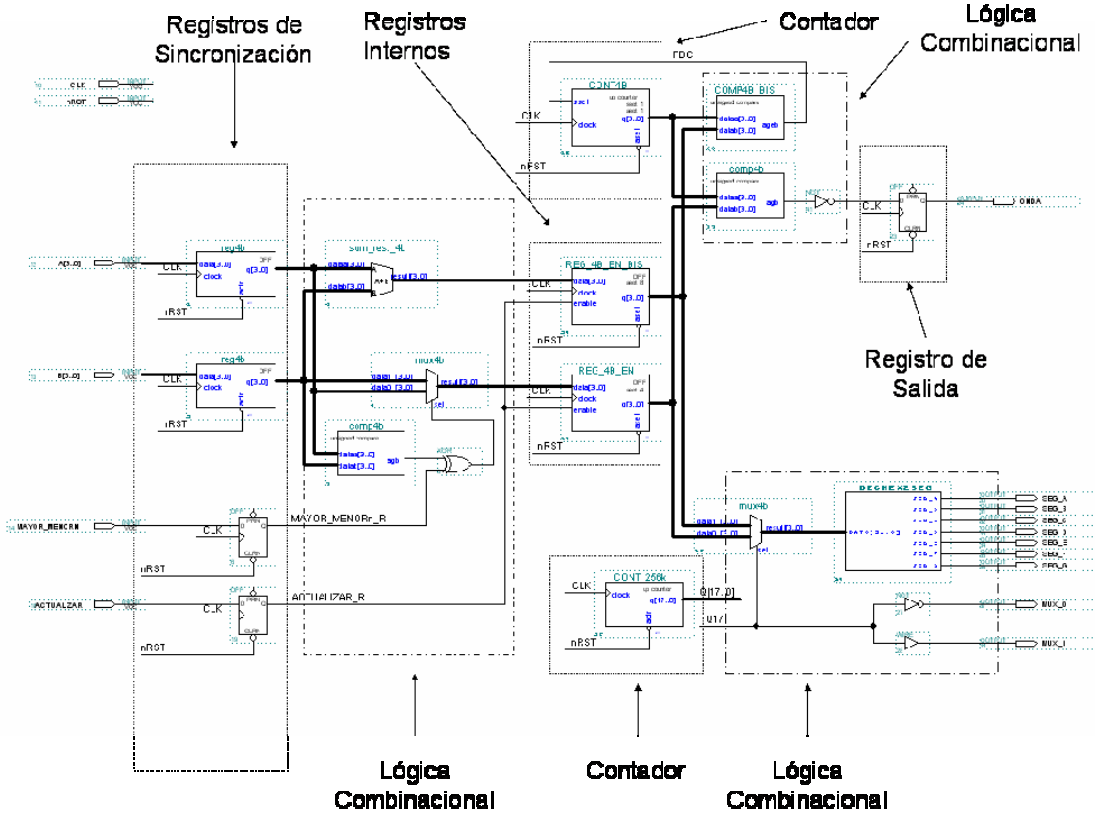

Figura EP. 10

Las entradas del circuito (dos entradas de datos de 4 bits, A y B, y dos entradas de control, ACTUALIZAR y MAYOR_MENORn) se registran mediante flip-flops. En la figura EP. 10 se puede observar esta lógica enmarcada en un bloque denominado REGISTROS DE SINCRONIZACIÓN –puede observarlos en detalle en la figura EP. 11.

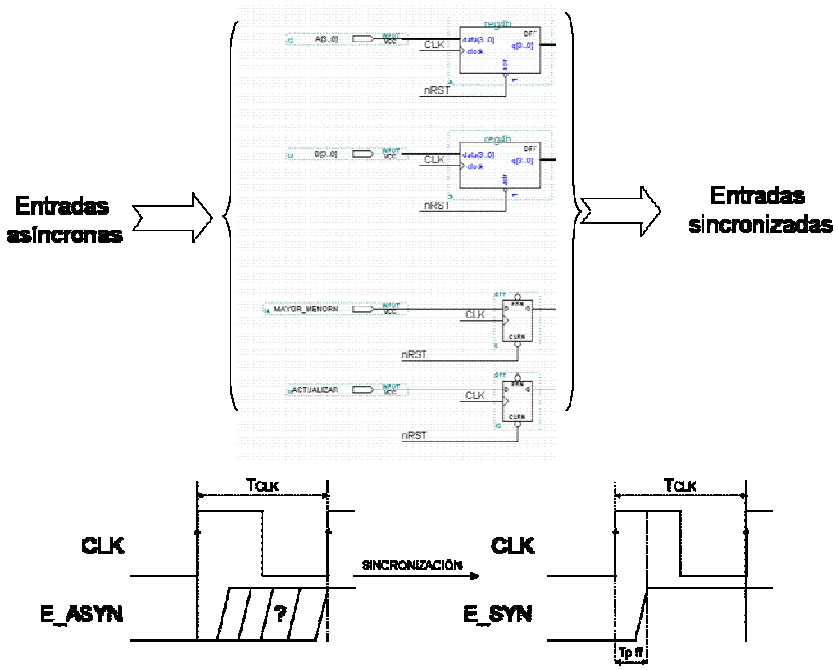

Figura EP.11

La lógica combinacional del circuito es manejada directamente por salidas de *flip-flops* o forma parte de cadenas de bloques manejados por ellas.

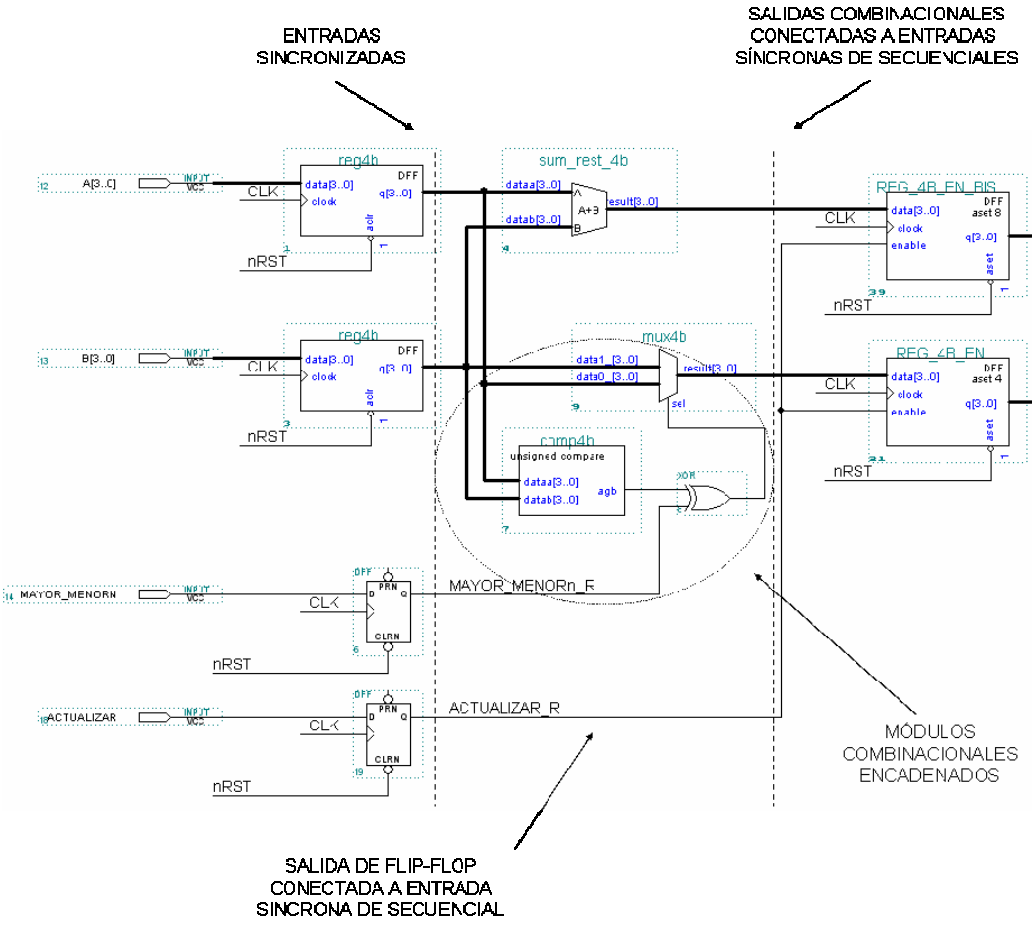

Figura EP.12

En la figura EP. 12 se muestran, en detalle, bloques que calculan la suma de los datos de entrada A y B y seleccionan el mayor o menor de ambos en función del nivel lógico de la entrada MAYOR_MENORn (si vale 1, el mayor y si no el menor) –observe que el circuito sumador está intercalado entre dos bloques secuenciales, mientras que el comparador, la puerta xor y el multiplexor se combinan para realizar la operación de selección; ambas estructuras son compatibles con las reglas de diseño síncrono. Las entradas que manejan toda esta lógica (entradas sincronizadas) sólo pueden cambiar en los flancos de subida del reloj, cuando esto ocurre, y una vez transcurrido un tiempo (el de propagación de los bloques combinacionales), las salidas serán estables y podrán actuar, en el siguiente flanco de reloj, sobre los flip-flops de los registros –en los que se almacena el resultado de la suma y el dato (mayor o menor) seleccionado.

En la figura EP. 12 puede observarse también una conexión directa entre dos bloques secuenciales: la salida del flip-flop que sincroniza la entrada asíncrona ACTUALIZAR y las entradas síncronas de habilitación de carga de los registros que almacenan el resultado del procesamiento de los bloques combinacionales –la entrada ACTUALIZAR determina, en definitiva, cuando debe guardarse la suma de A y B y el dato seleccionado entre ambos.

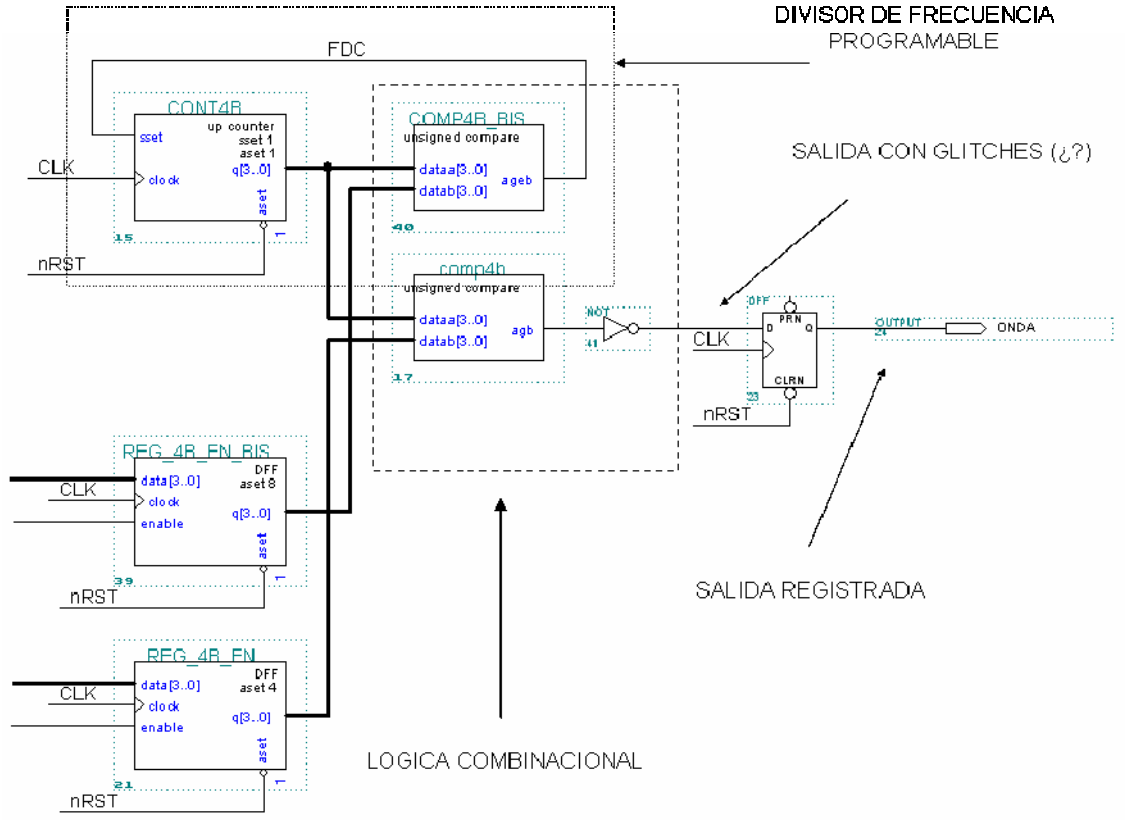

Figura EP.14

Hay lógica combinacional en el circuito de la figura EP. 10 que forma parte de otro tipo de estructuras. En la figura EP.14 se muestran los módulos del circuito que generan la señal de salida. El periodo de la señal generada es controlado por el registro que guarda la suma de los datos de A y B; el valor de dicho registro se compara con la salida de un contador; cuando ambos valores son iguales, se reinicia la cuenta desde 1 (para que el módulo de cuenta sea igual al valor almacenado en el registro). Observe que las entradas del comparador son las salidas del contador y su

salida reinicia síncronamente los flip-flops del propio contador: es un caso particular del ilustrado en la figura EP.12.

En la figura EP.14 hay un segundo comparador que genera la forma de onda comparando el estado de cuenta con el valor almacenado en el registro que guarda el valor mayor, o menor, de A y B –este valor determina el ciclo de trabajo de la señal de salida. Su salida se registra porque puede tener glitches y se desea generar una onda cuadrada perfecta.

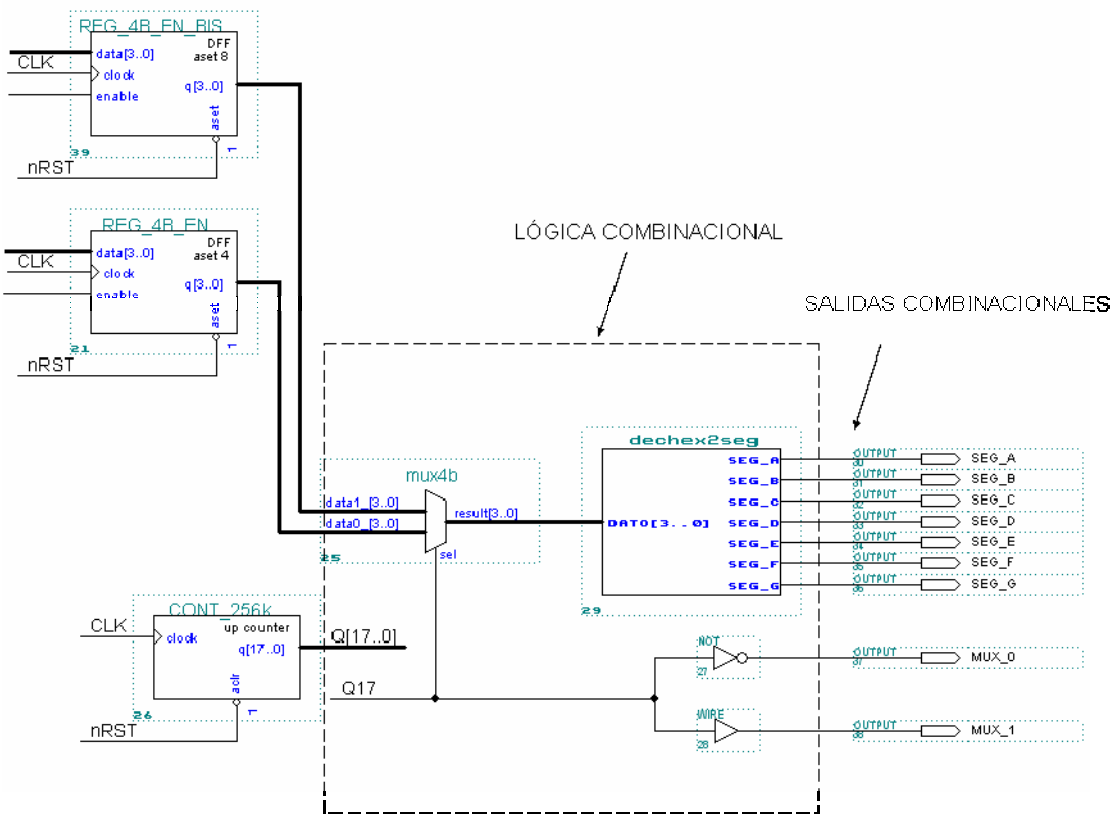

Figura EP.15

La figura EP.15 muestra el módulo de visualización del circuito: permite mostrar en dos displays de 7 segmentos el contenido de los registros de suma y dato seleccionado. Observe que en este caso las salidas síncronas del sistema son combinacionales, debido a que la existencia de glitches (por su pequeña duración) en estas señales resulta imperceptible en los LEDs de los displays.

2.4 Ejecución de operaciones en los circuitos síncronos

Los sistemas realizados siguiendo técnicas de diseño síncrono tienen un modo de operación caracterizado por el gobierno ejercido por la señal de reloj. En los siguientes apartados se va a analizar la ejecución de operaciones en las estructuras internas y de entrada y salida de los sistemas síncronos.

2.4.1 Operaciones en las estructuras internas

En la estructura genérica ilustrada en la figura 20 pueden darse dos esquemas básicos para la interconexión interna de circuitos combinacionales (figura 21). En la figura 21a se muestran las condiciones típicas de operación de un bloque combinacional interno: sus entradas son salidas de un circuito secuencial síncrono y, por tanto, podrán cambiar únicamente cuando haya flancos activos de reloj y sus salidas están conectadas a entradas síncronas de otros módulos secuenciales, actuando sobre ellos, también, sólo cuando haya flancos activos del reloj del circuito.

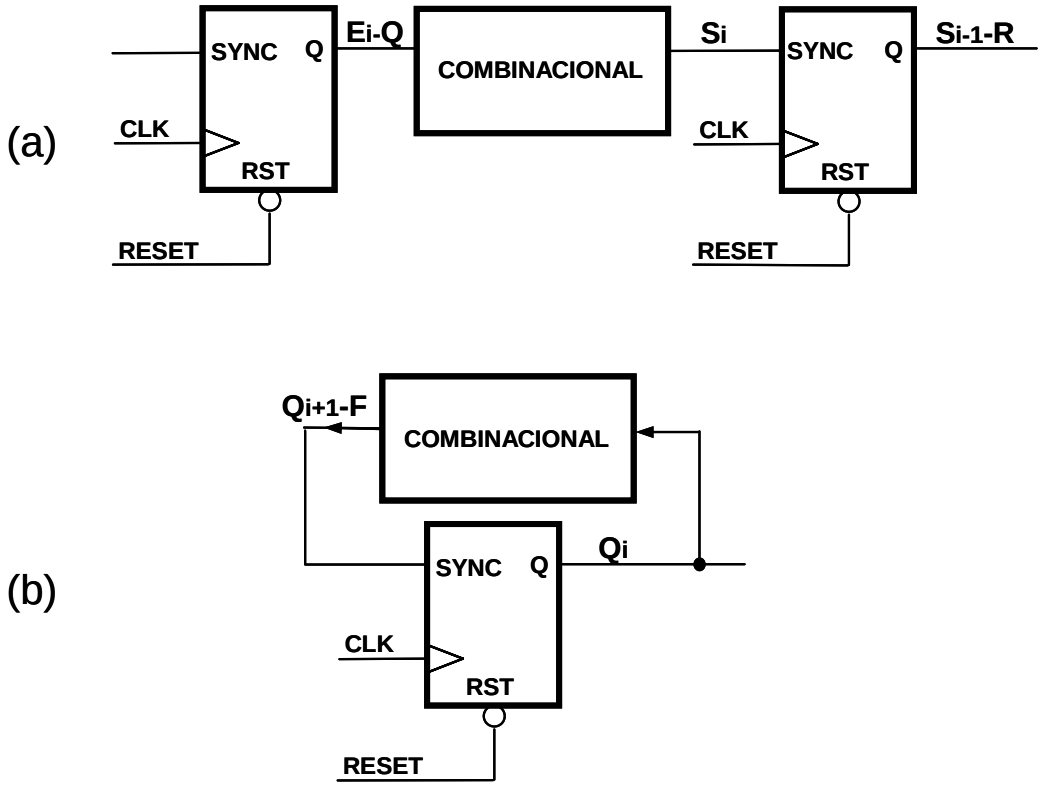

Figura 21 (a y b). Estructuras internas

Para que este esquema funcione correctamente es necesario que en el tiempo que transcurre entre dos flancos de reloj –entre el que maneja el cambio de las entradas y el que determina que las salidas actúen sobre un módulo secuencial- el circuito combinacional haya estabilizado sus salidas o, dicho de otro modo, que el régimen transitorio del combinacional (su tiempo de propagación) sea menor que el periodo de reloj del circuito.

En la figura 22 se ilustra este modo de funcionamiento para una operación de registro de las salidas de un combinacional embutido en una estructura como la de la figura 21a. En un determinado flanco de subida de reloj (flanco *i*), la entrada del combinacional (E_i-Q) cambia -puede cambiar- y las salidas (S_{i-1}) correspondientes a las entradas del ciclo anterior (las establecidas por el flanco *i-1*) se registran (S_{i-1}-R). El circuito combinacional tarda un cierto tiempo (el de retardo del circuito, representado en la figura mediante un rayado vertical) en establecer valores de salida (S_i) correctos y sin *glitches*, si este tiempo –el régimen transitorio- es menor que el periodo del reloj (T_{CLK}),

las salidas se registran (S_i-R) y están disponibles tras el siguiente flanco, en el que las entradas del circuito pueden volver a cambiar para que se realice una nueva operación combinacional.

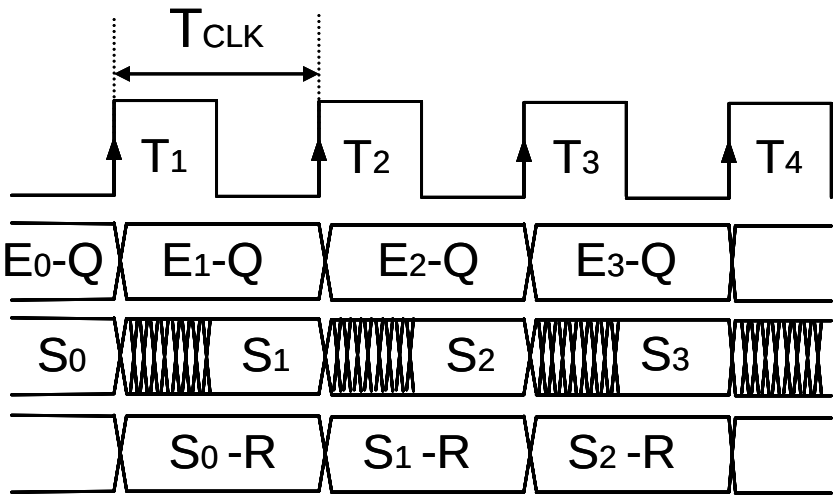

Figura 22. Operaciones combinacionales entre registros

En términos de comportamiento dinámico, el retardo "equivalente" de cualquier operación realizada por un circuito combinacional empotrado entre dos bloques secuenciales es un periodo del reloj del circuito.

El esquema representado en la figura 21b corresponde a circuitos combinacionales que realizan el cálculo de transiciones de estado en circuitos secuenciales síncronos. En algunos subsistemas (en generadores de patrones de tiempo, (*timers*), y, en general, en secuenciadores de formas de señal periódicas) esta lógica constituye la única fuente de excitación síncrona de los *flip-flops*.

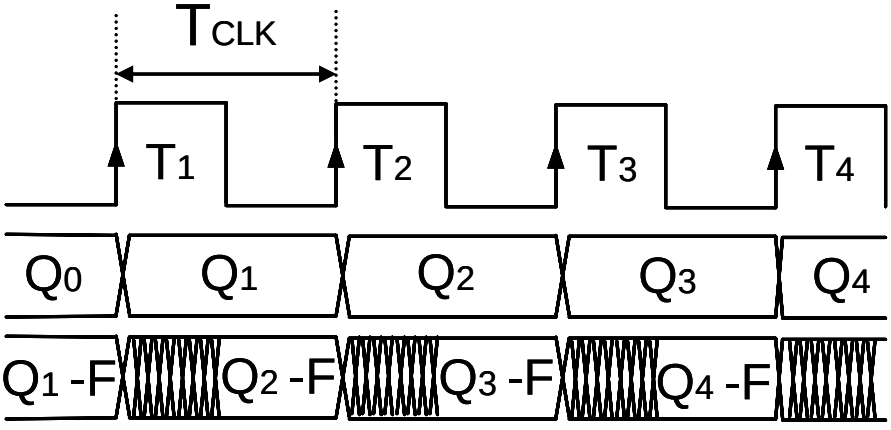

Figura 23. Operación de un secuenciador

Este esquema es reducible a un caso particular del mostrado en la figura 15. La particularidad que tiene es que los mismos *flip-flops* que manejan las entradas del combinacional también registran sus salidas, pero que esta operación se lleve a cabo correctamente requiere la misma condición que el caso general (que el tiempo de

propagación del combinacional sea menor que un periodo de reloj). En la figura 23 se ilustra este modo de funcionamiento: Q_i es el estado actual (entrada del combinacional) y Q_i-F el estado futuro (salida tras el transcurso del régimen transitorio).

A modo de ejemplo, los módulos de la figura EP.12 (que se trae aquí para comodidad del lector) forman una estructura que se ajusta al modelo de la figura 21a: un conjunto de módulos combinacionales encastrados entre flip-flops.

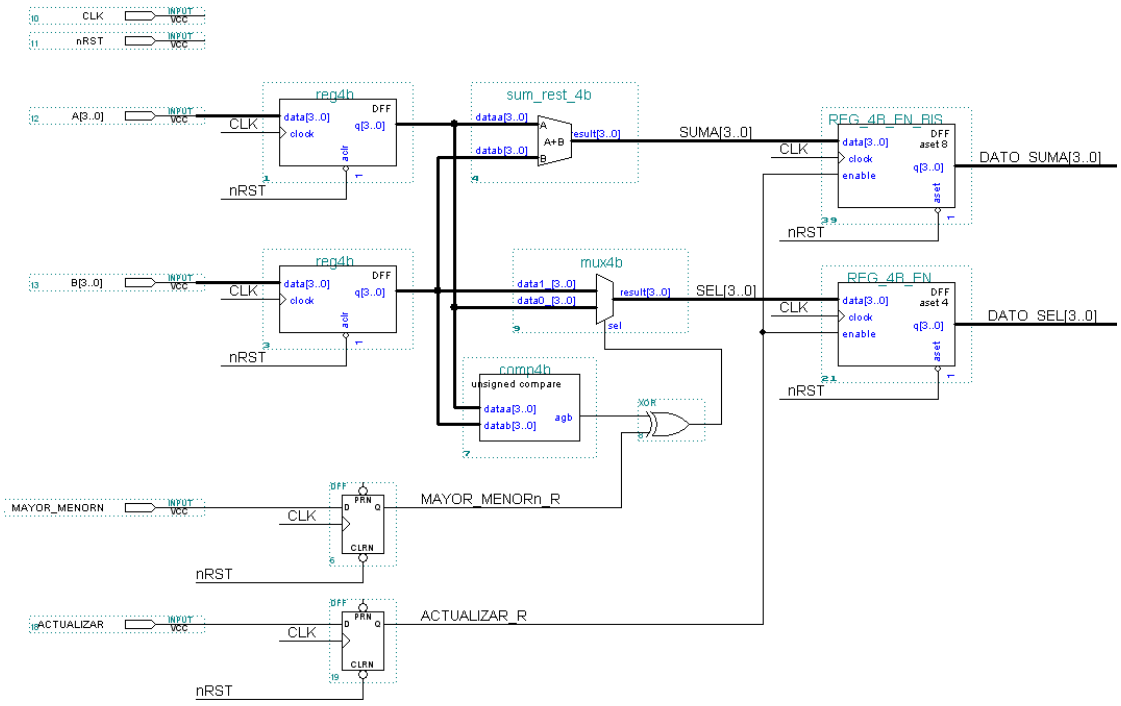

Figura EP.12 (repetida)

Las entradas de los circuitos combinacionales son manejadas por los registros de sincronización mientras que sus salidas actúan sobre las entradas de los registros de almacenamiento de datos. Observe que hay una conexión directa entre un registro de sincronización (el correspondiente a la entrada ACTUALIZAR) y un registro de almacenamiento (el que guarda el valor de DATO_SUMA); es un caso particular –y muy frecuente en la práctica- del modelo de la figura 21a: al no haber lógica combinacional entre los flip-flops, el retardo de la conexión y el de propagación del flip-flop son los que determinan el tiempo que debe transcurrir, desde un flanco activo de reloj, para que la entrada síncrona del registro de almacenamiento sea estable y válida y pueda actuar con garantías en el siguiente flanco.

En el cronograma de la figura EP.16 se muestra el funcionamiento de esta parte del circuito, para una la señal de reloj con un periodo de 40 ns (25 MHz) que permite que la lógica combinacional alcance con holgura el régimen permanente –en una realización del circuito sobre una FPGA de la familia 10K de ALTERA.

Cada vez que en un flanco se verifica un cambio en las entradas asíncronas A, B ó MAYOR_MENORn, la lógica combinacional recalcula el valor del dato seleccionado y el resultado de la suma de A y B; si es capaz de realizar la operación en un tiempo menor que el periodo de reloj del circuito, los resultados se registran en el siguiente

flanco de reloj. En el ejemplo del cronograma EP.16, A y B pasan a valer 5 y 4, respectivamente y la señal MAYOR_MENORn vale 1 (seleccionar el mayor). El tiempo requerido para realizar la operación (que SUMA valga 9 y DATO_SEL, 5) es de un periodo de reloj (en el ejemplo 40 ns).

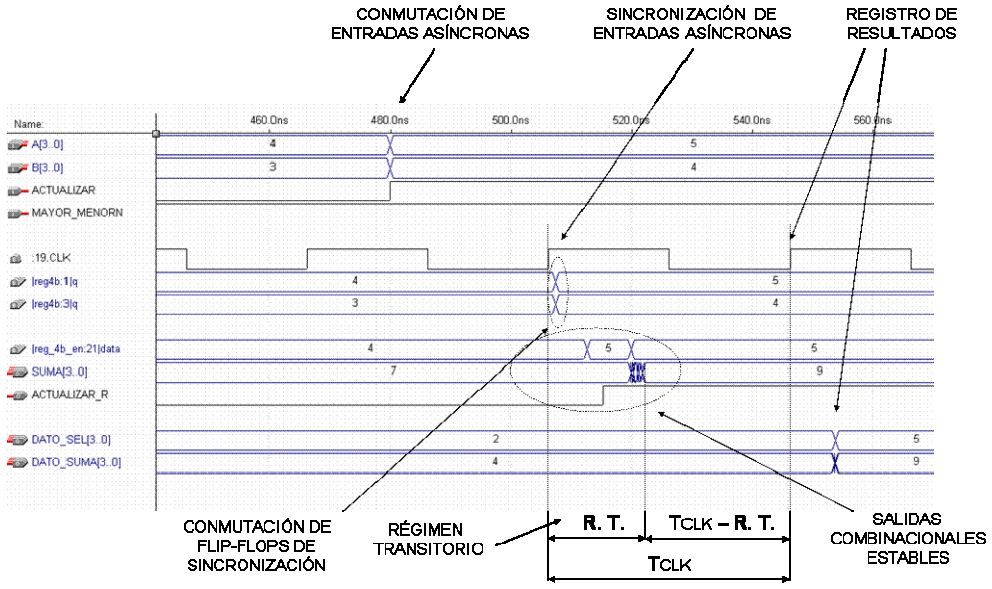

Figura EP.16

Si se desea acelerar la velocidad de ejecución de operaciones en un sistema síncrono es evidente que hay que disminuir la duración del periodo de reloj (aumentar la frecuencia de reloj), pero teniendo en cuenta que siempre habrá un límite para el valor mínimo del periodo de la señal de reloj, fijado por la duración del régimen transitorio de la lógica combinacional del circuito. En el cronograma de la figura EP.17 se muestra el comportamiento del circuito de la figura EP.12 para un reloj de 83'3 MHz (un periodo de 12 ns). En este caso el circuito funcionaría incorrectamente porque el régimen transitorio de la lógica combinacional es mayor que el periodo de reloj. Más adelante se explicará cómo puede calcularse la frecuencia máxima de operación de un circuito síncrono.

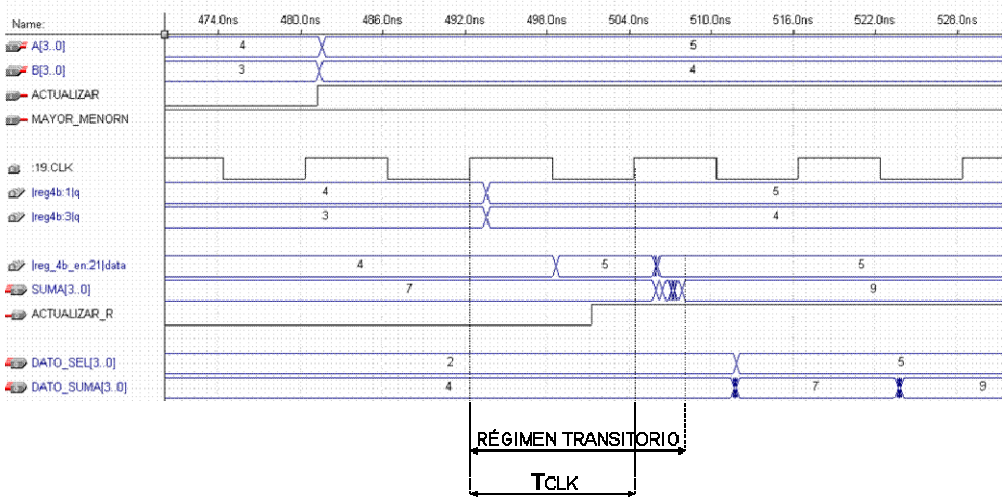

Figura EP. 17

El análisis del modo de operación de los circuitos internos de un sistema síncrono nos permite deducir una característica genérica muy importante: el periodo del reloj es la unidad mínima de tiempo (la resolución de tiempo) de procesamiento en el circuito: cualquier operación de un circuito síncrono tarda en realizarse un número entero de ciclos de reloj. Medida en tiempo real, la duración de una operación es el valor del periodo de reloj multiplicado por el número de ciclos que tarda en completarse. Esta propiedad permite determinar el número de ciclos de reloj de que se dispone para realizar operaciones en tiempo real o para calcular la frecuencia a la que debe funcionar un circuito para poder cumplir las especificaciones de tiempo de una aplicación.

Ejemplo

Si un circuito síncrono debe entregar las salidas correspondientes al procesamiento de los datos de entrada en 400 ns y la frecuencia de reloj del circuito es de 10 MHz, determine el número máximo de ciclos de reloj que pueden utilizarse para completar dicho procesamiento.

Solución: El periodo de la señal de reloj es de 100 ns; se dispone, por tanto de de 4 ciclos de reloj.

Ejemplo

Un circuito síncrono que tarda 10 ciclos de reloj en calcular la salida correspondiente al procesamiento de los datos de entrada, tiene que tardar, como máximo, 200 ns en realizar sus operaciones. Calcule la frecuencia a la que debe funcionar.

Solución: El periodo máximo de la señal de reloj es de 200 ns/10, 20 ns; por tanto, la frecuencia mínima a la que debe funcionar es 50 MHz.

2.4.2 Circuitos de entrada y salida

Una vez analizado el modo operación de las estructuras lógicas internas de los circuitos síncronos, vamos a revisar ahora los aspectos característicos del funcionamiento de los circuitos que realizan funciones de interfaz con las entradas y salidas.

Salidas

Las salidas de un circuito síncrono son, siempre, síncronas, independientemente de que estén conectadas a salidas de *flip-flops* o a salidas de lógica combinacional, porque es posible caracterizar su actividad (los instantes en que pueden conmutar) respecto a los flancos activos del reloj del circuito. Lo que puede diferenciar ambos casos es la presencia o ausencia de *glitches* –las salidas de *flip-flops* nunca van a tener *glitches*, mientras que las de circuitos combinacionales pueden, o no, tenerlos. La figura 24 muestra los dos tipos posibles de conexión de las salidas de un circuito síncrono.

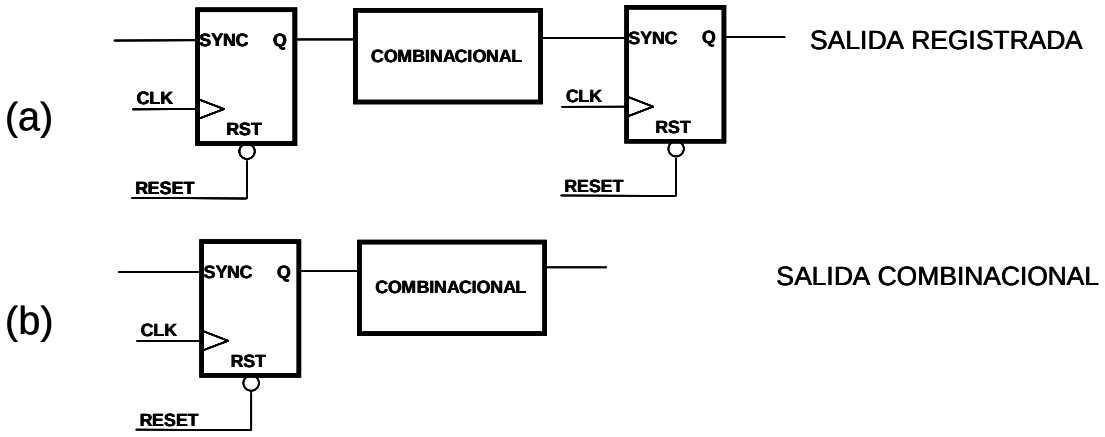

Figura 24. Estructuras de salida

La conexión de la figura 24a, garantiza una salida sin *glitches* (figura 25a) que puede conmutar únicamente en los flancos activos de reloj, con un retardo respecto a estos ($tp_{SALIDA\ REGISTRADA}$) cuyo valor es la suma del tiempo de propagación del *flip-flop* (tp_{ff}) y el de la ruta de interconexión que une la salida de éste con la del sistema (tp_{ruta}).

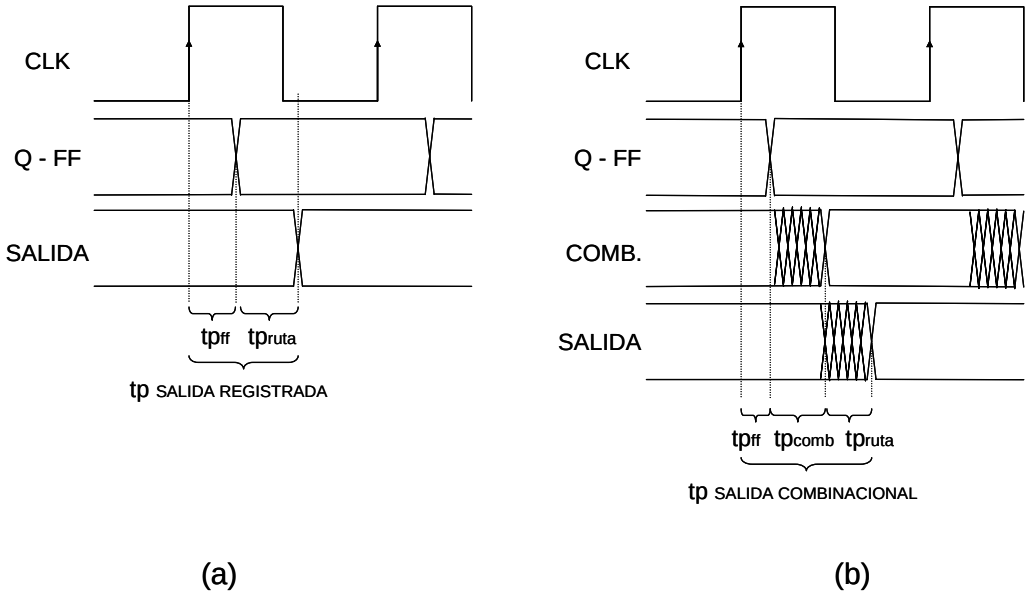

Figura 25. Operación de las salidas combinacionales y registradas

En las salidas combinacionales la existencia o no de *glitches* dependerá de la actividad en las entradas del circuito y de la propia realización del circuito combinacional. Al igual que en el caso anterior, la conmutación de estas salidas (figura 25b) está condicionada por la ocurrencia de un flanco activo de reloj, aunque ahora el establecimiento de valores correctos se demorará, respecto al flanco, un tiempo igual a la suma del de propagación de un *flip-flop* (tp_{ff}) más el de retardo del combinacional (tp_{comb}) y el de los recursos de interconexión de las salidas de éste con la propia salida del sistema (tp_{ruta}).

En el circuito de la figura EP.10, por ejemplo, hay salidas combinacionales y registradas; están señaladas en la figura EP.18.

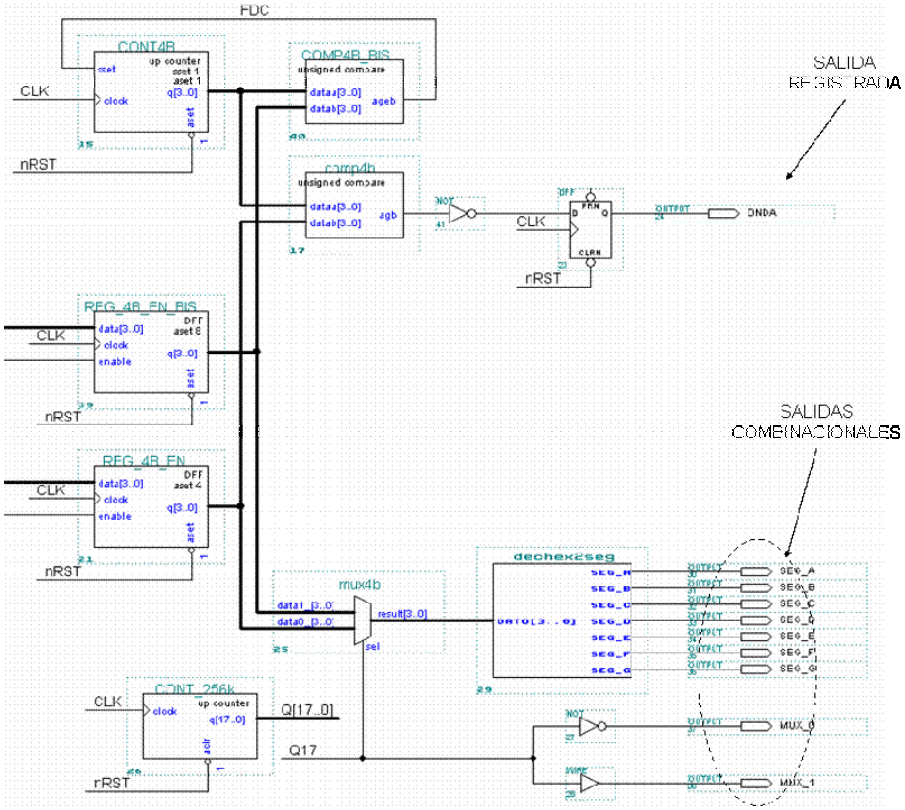

Figura EP.18

En el cronograma de la figura EP.19 se muestra un detalle del comportamiento de estas salidas en una simulación –en la que se ha escalado la frecuencia de las señales de multiplexación para que pueda observarse la conmutación de las salidas combinacionales.

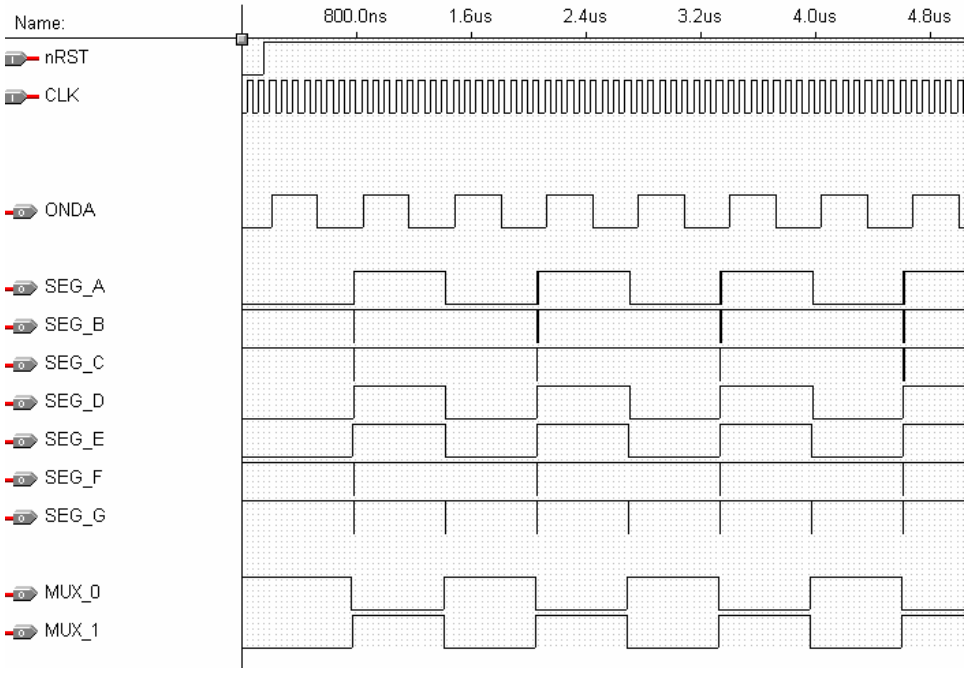

Figura EP. 19

La salida registrada (ONDA) y las salidas combinacionales MUX_0 y MUX_1 no tienen glitches –MUX_1 es, en realidad, una salida registrada (Q17) y MUX_0 es MUX_1 negada- mientras que las salidas que manejan los LEDs de los displays de 7 segmentos sí (no importa, porque son imperceptibles debido a su corta duración). Independientemente de su naturaleza, puede observarse que las conmutaciones de las salidas se producen en sincronía con el flanco activo de reloj.

Entradas

Las entradas de un sistema síncrono son siempre asíncronas con el reloj del circuito –si no fuera así, la entrada provendría de un sistema con el mismo reloj y se podría considerar que ambos son susbsistemas de un sistema síncrono y la supuesta entrada se convertiría, en definitiva, en una señal "interna" de éste-, entendiendo que esta asincronía significa que no se puede establecer ninguna relación entre los instantes de tiempo en que dicha entrada puede conmutar y los flancos activos de reloj.

En un sistema síncrono no se puede permitir que una entrada asíncrona actúe directamente sobre la entrada de un bloque combinacional interno -puesto que en tal caso no puede garantizarse que sus salidas sean correctas y estables en los instantes de ocurrencia de flancos activos de reloj; para compatibilizar la naturaleza asíncrona de las entradas y esta norma de diseño es necesario registrar las entradas para obtener una "versión" sincronizada. En la figura 26 se muestra el circuito de sincronización y la forma de la entrada.

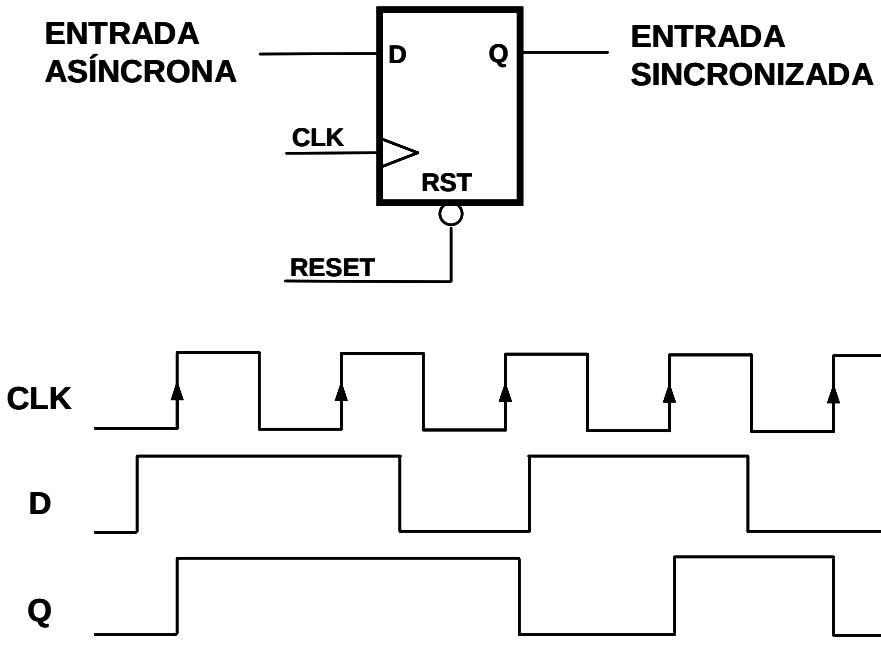

Figura 26. Sincronización de entradas

La sincronización de las entradas asíncronas, siendo necesaria para cumplir las reglas de diseño síncrono, tiene efectos –observables en el cronograma de la figura 26- que el diseñador debe evaluar, pues pueden afectar al diseño del circuito. El primero de

ellos es que el circuito de sincronización retarda las conmutaciones de la señal asíncrona tanto como tarde en llegar un flanco activo de reloj: en el caso peor, cuando la entrada asíncrona conmuta justo después de un flanco activo, la demora que se produce es de un periodo de reloj. Un segundo efecto notable de la sincronización es que modifica la forma de la señal: los niveles lógicos de la entrada asíncrona pueden tener, en general, una duración cualquiera, pero la de los niveles de la señal sincronizada siempre va a ser un múltiplo del periodo de reloj.

En el circuito de la figura EP.10 (página 31), por ejemplo, se sincronizan las entradas de datos, A y B, y las de control, ACTUALIZAR y MAYOR_MENORn. En el cronograma de la figura EP. 20 se muestra una simulación del circuito que pone de manifiesto los efectos de la sincronización: la demora de hasta un ciclo de reloj en la detección de las conmutaciones y el ajuste en la duración de las señales al periodo de la señal de reloj.

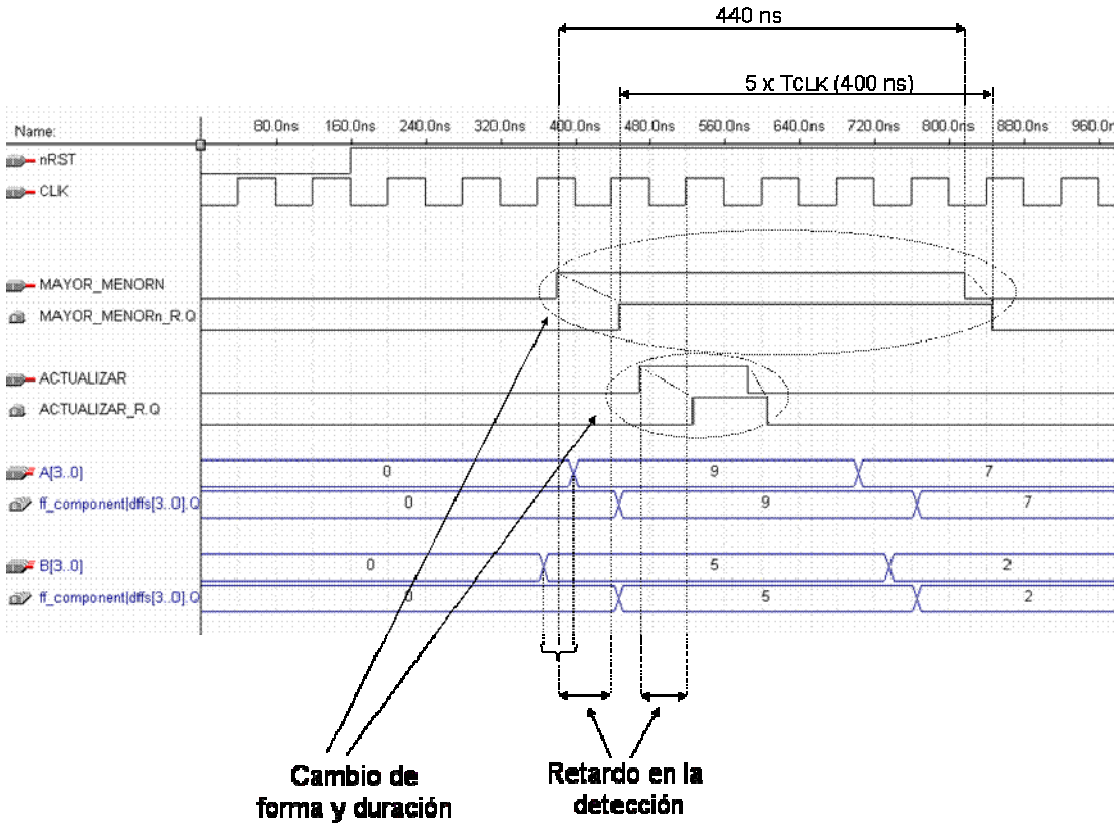

Figura EP.20

La duración de los pulsos de la señal sincronizada depende del número de flancos de reloj en que la entrada asíncrona esté a nivel alto (o bajo). Esta última propiedad es válida también para cualquier señal interna de un circuito síncrono; en términos de tiempo, la anchura de una señal es relativa a su nivel en los flancos de reloj: un pulso a nivel alto con una duración de N ciclos de reloj se corresponde con un nivel alto que se detecta en N flancos consecutivos de reloj –entiéndase que lo que se quiere enfatizar aquí es que independientemente del tiempo real que se mantenga un nivel lógico, o de que haya conmutaciones en el nivel de una señal entre flancos de reloj, para un circuito síncrono sólo es relevante el nivel de las señales en los flancos activos de reloj; de acuerdo con esto, las señales de la figura 27 son equivalentes en forma y duración.

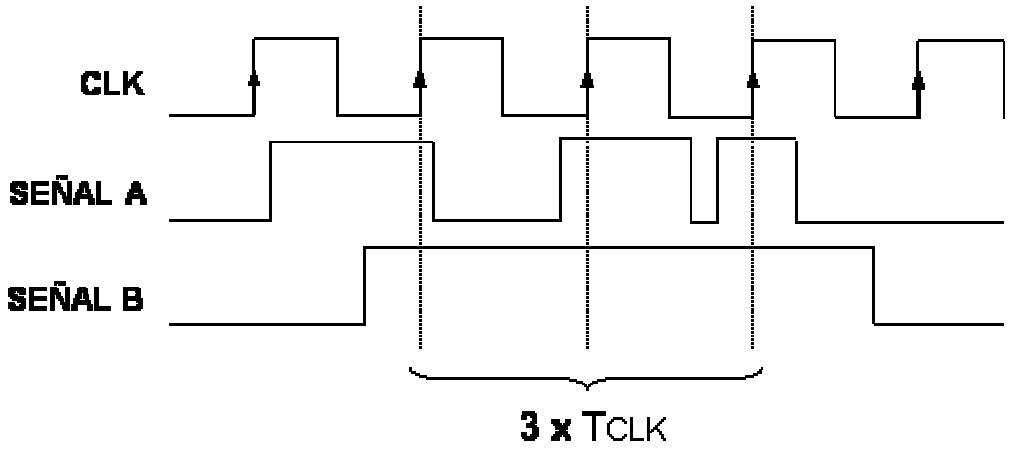

Figura 27. Duración y forma de las señales síncronas

2.5 Frecuencia máxima de funcionamiento de un circuito síncrono

Del análisis realizado sobre el modo de funcionamiento de las estructuras internas y de entrada y salida de un circuito síncrono se puede deducir la característica propia que distingue su funcionamiento: las operaciones de todo el sistema están controladas por los flancos activos del reloj, en ellos, y sólo en ellos, se verifican las conmutaciones que afectan a la lógica, combinacional o secuencial, disponiéndose del tiempo equivalente a un periodo de reloj para que todo el sistema se estabilice, antes del siguiente flanco, en el que podrán ejecutarse una nueva serie de operaciones. Con este modo de operación se garantiza un funcionamiento real del sistema idéntico al ideal concebido por el diseñador, pues el tiempo que transcurre entre dos flancos activos (un periodo de reloj) permite que transcurra el régimen transitorio de los circuitos combinacionales, impidiendo que los *glitches* pueda alterar la operación ideal del sistema.

La capacidad de proceso de un circuito síncrono depende de dos parámetros: el primero es el número y tipo de operaciones que se ejecutan en cada ciclo de reloj, y el segundo la frecuencia del reloj del circuito. Por ejemplo, un circuito que realiza dos sumas y dos multiplicaciones por ciclo de reloj, funcionando con un reloj de 10 MHz ejecuta veinte millones de sumas y de multiplicaciones por segundo; si se duplica la frecuencia de reloj o el número de operaciones que pueden ejecutarse en un ciclo de reloj, también se duplica el número de operaciones que es capaz de ejecutar en una unidad de tiempo.

En general, cuanto mayor sea la frecuencia de reloj de un circuito, menor será el número de operadores necesarios para alcanzar la potencia de procesamiento requerida por la aplicación y, por tanto, el circuito y su desarrollo resultarán más simples y baratos. Es evidente que la frecuencia del reloj no se puede elegir arbitrariamente: en un circuito síncrono su valor está limitado por los retardos de los módulos combinacionales encastrados entre *flip-flops*, puesto que en el tiempo comprendido entre dos flancos activos (un periodo de reloj) deben alcanzar el régimen permanente de funcionamiento.

El cálculo de la frecuencia máxima que puede tener el reloj de un sistema síncrono es conceptualmente sencillo. Se determina evaluando el tiempo mínimo que debe separar dos flancos de reloj. Supongamos, inicialmente, que todos los *flip-flops* del circuito son activos por flanco de subida y que sus tiempos característicos (de propagación (t_P), *set-up* (t_{SU}) y *hold* (t_H)) son los mismos; se va a considerar también que el reloj llega en el mismo instante de tiempo a todos ellos –que el *skew* de reloj es cero. Cumpliéndose estas premisas, las entradas de todos los circuitos combinacionales (E_{COMB}) pueden cambiar únicamente en los flancos activos de reloj, con un retardo respecto a estos (Figura 28) igual al tiempo de propagación de un *flip-flop* ($t_{P\text{-}ff}$).

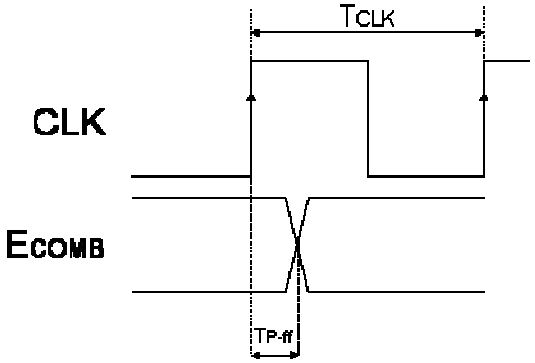

Figura 28. Periodo mínimo de reloj (1)

A partir del momento en que las entradas de los combinacionales conmutan, comienza su régimen transitorio; el combinacional que tenga el tiempo de retardo (t_{Pcomb}) mayor ($t_{Pcomb(max)}$) será el último en estabilizar niveles lógicos válidos en sus salidas (S_{COMB}), cuando esto ocurra (Figura 29) todos los circuitos combinacionales del sistema síncrono estarán preparados para actuar sobre las entradas síncronas de los *flip-flops* -en el tiempo de propagación de la lógica combinacional se incluye el retardo imputable a los elementos de interconexión.

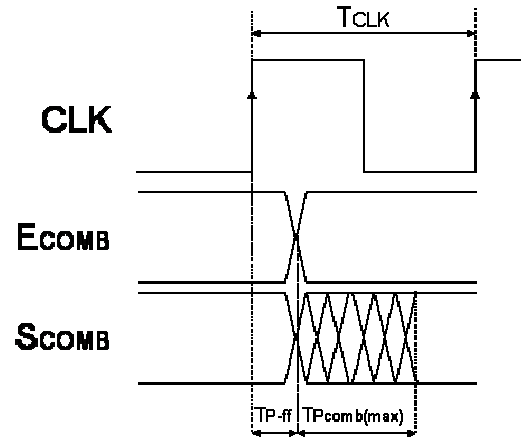

Figura 29. Periodo mínimo de reloj (2)

Como las entradas síncronas de los *flip-flops* tienen que permanecer estables antes del flanco activo de reloj un tiempo mayor o igual al tiempo de *set-up* (t_{SU}), debemos sumar éste a los dos anteriores (Figura 30) para determinar la separación mínima entre dos flancos (el periodo mínimo del reloj, $T_{CLK(min)}$) que garantiza el correcto funcionamiento del circuito.

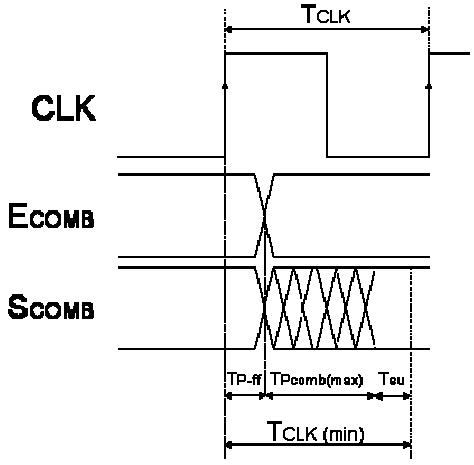

Figura 30. Periodo mínimo de reloj (y 3)

El valor del periodo mínimo de la señal de reloj se calcularía, por tanto, con la expresión:

$T_{CLK(min)} = t_{Pff} + t_{PCOMB(max)} + t_{SU}$

La frecuencia máxima del reloj del circuito sería:

$F_{CLK(max)} = 1/ T_{CLK(min)}$

Desde el punto de vista metodológico, los pasos que hay que seguir para calcular la frecuencia máxima del reloj son:

1. Calcular el retardo combinacional máximo existente en el circuito síncrono (considerando únicamente la lógica combinacional encajada entre *flip-flops*).

2. Para calcular el periodo mínimo del reloj hay que sumar, al retardo máximo obtenido, el tiempo de propagación y de *set-up* de un *flip-flop*; su inverso será la frecuencia máxima de funcionamiento.

Conocida la frecuencia máxima de reloj, podrá garantizarse que el circuito síncrono va a funcionar correctamente si su reloj tiene una frecuencia menor o igual a ésta. Debe tenerse en cuenta que el cálculo de los retardos debe realizarse para las condiciones peores –las más lentas- de funcionamiento del circuito, es decir, utilizando tiempos de propagación máximos.

Ejemplo

Conociendo los siguientes datos de un circuito síncrono:

Tiempos característicos de los flip-flops: $t_{P\,FF\,max} = 3$ *ns;* $t_{SU\,FF} = 2$ *ns;* $t_{H\,FF} = 1$ *ns*

Tiempos de propagación de la lógica combinacional:

$T_{P\,COMB\,max} = 35$ *ns;*

Se puede calcular el periodo mínimo de la señal de reloj:

$$T_{CLK\ min} = T_{P\ COMB\ max} + t_{P\ FF\ max} + t_{SU\ FF} = 35 + 3 + 2 = 40\ ns,\ y$$

Y la frecuencia máxima:

$$F_{CLK\ max} = 1/40\ ns = 25\ MHz$$

La complejidad del cálculo de la frecuencia máxima de reloj depende del número de *flip-flops* y bloques combinacionales del circuito. Existen herramientas de CAD electrónico, los analizadores (o simuladores) de tiempos, que realizan esta tarea rápida y eficazmente. Este tipo de herramientas identifica, además, la lógica y las interconexiones que limitan la frecuencia máxima -los caminos más lentos-, por lo que, cuando es necesario aumentar la frecuencia de funcionamiento para conseguir la velocidad de procesamiento requerida, se puede determinar fácilmente la parte del circuito que hay que modificar. La simplicidad de este análisis y lo mucho que facilita las operaciones de rediseño para obtener la frecuencia objetivo de funcionamiento son una de las ventajas más importantes derivada del uso de las técnicas de diseño síncrono.

La frecuencia máxima de operación del circuito de la figura EP.10 sobre una FPGA de ALTERA, la EPF10K20RC208-4, puede ser calculada con ayuda del analizador de tiempos del MAX+plus II. El resultado del análisis se muestra en la figura EP. 21. Cuando se configura en el modo ***Registered Performance****, el analizador de tiempos realiza un recuento de todos los retardos internos del circuito, selecciona el mayor de ellos y determina la frecuencia máxima de funcionamiento.*

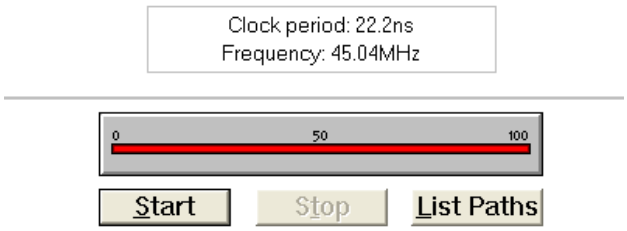

Figura EP.21

El analizador indica que el periodo mínimo del reloj es de 22'2 ns y la frecuencia máxima, por tanto, de 45'04 MHz. El botón ***List Paths*** *permite obtener un listado de las rutas combinacionales internas en las que se ha calculado un retardo mayor (figura EP.22) y a las que, en consecuencia, cabe imputar la limitación de frecuencia calculada.*

En el listado aparece, para cada camino, el valor del retardo de la lógica combinacional y las rutas de interconexión (el valor máximo para nuestro ejemplo es de 18'6 ns) y el consecuente valor de periodo de reloj, que se obtiene añadiendo al retardo combinacional el tiempo de propagación y de set-up de un flip-flop (el máximo

es de 22'2 ns y coincide, lógicamente, con el periodo mínimo del reloj calculado por el analizador).

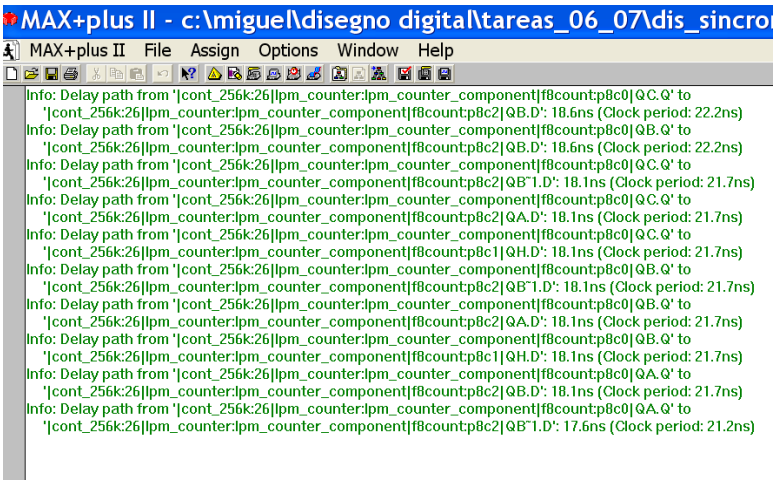

Figura EP.22

Cada entrada del listado enumera los dos flip-flops (fuente y destino) entre los que se encuentra la lógica combinacional (una, o varias, LUTs, en el caso de esta FPGA), permitiendo identificar –y, si se desea, localizar en el esquema o en el plano físico de la realización del circuito sobre la FPGA- los elementos que limitan la velocidad de funcionamiento.

El analizador de tiempos de MAX+plus II puede utilizarse también para calcular el retardo de las salidas del circuito en relación con los flancos activos de la señal de reloj (figura EP.23). En los casos en que la salida está manejada por varios flip-flops (esto, evidentemente, sólo puede ocurrir en salidas combinacionales), se presenta el retardo menor y mayor -en todos los casos con tiempos máximos- de entre todos los caminos existentes.

Delay Matrix

Destination

	ONDA	SEG_A	SEG_B	SEG_C	SEG_D	SEG_E	SEG_F	S
CLK	13.1ns	24.0ns/31.0ns	24.4ns/34.4ns	23.1ns/27.1ns	23.6ns/31.2ns	20.6ns/24.1ns	23.2ns/29.0ns	

Figura EP. 23

La salida registrada ONDA, por ejemplo, tiene un retardo, de pin a pin, de 13'1 ns –es la suma del tiempo de propagación de la red global de distribución de reloj (6,4 ns), el retardo de propagación del flip-flop, el retardo de interconexión con el bloque de salida y el retardo imputable al PAD del pin de la FPGA-, mucho menor, por ejemplo que el retardo máximo de cualquier salida combinacional, entre 24'1 y 34'4 ns de pin a pin –esta diferencia se debe al retardo imputable a la lógica combinacional y su red de interconexión, un factor que no afecta al retardo de las salidas registradas.

2.6 Skew de Reloj.

En un circuito síncrono real, la señal de reloj no llega exactamente en el mismo instante de tiempo a todos los *flip-flops* del circuito. Como se señaló en el apartado 2.3, el desfase con el que llega el reloj a dos *flip-flops* se conoce como *skew* de reloj. Cuando en un circuito síncrono se habla, genéricamente, del *skew* de la señal de reloj, se hace referencia al valor máximo de *skew* existente en todo el circuito.

El *skew* del reloj entre dos *flip-flops* de un circuito síncrono que interaccionan por medio de un circuito combinacional "encajado" entre ambos puede dar lugar a los dos casos que se ilustran en la figura 31.

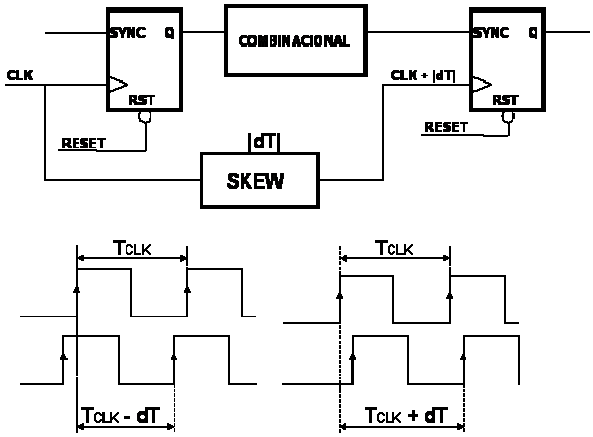

Figura 31. *Skew* de reloj

El *skew* de reloj se representa en el circuito de la figura mediante un bloque que tiene asociado un retardo que puede ser negativo (cronograma de la izquierda), lo que quiere decir que el reloj llega antes al *flip-flop* de salida que al de entrada, o positivo (cronograma de la derecha), provocando que se adelante el reloj del *flip-flop* de entrada al de salida. En ambos casos el *skew* puede ser una fuente de problemas.

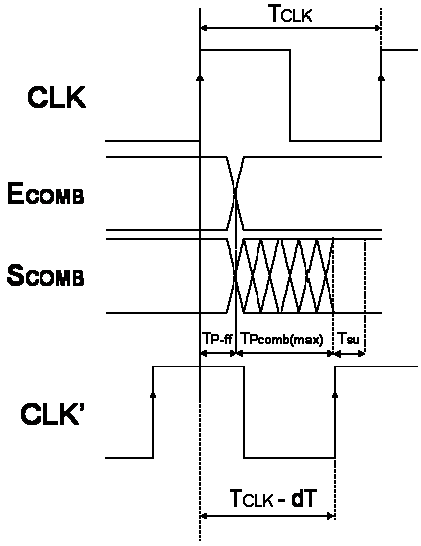

Figura 32. Reducción del periodo de reloj

En el primer caso, el flanco de reloj del segundo *flip-flop* se adelanta al instante previsto, de modo que si las salidas del circuito combinacional aun no son estables, o lo

son pero por un tiempo menor al de *set-up* del *flip-flop*, el circuito podría funcionar mal (Figura 32).

Este efecto puede corregirse modificando la frecuencia de la señal de reloj. Para ello hay que incrementar el periodo mínimo de la señal de reloj en una cantidad igual al *skew* del circuito:

$$T_{CLK(min)} = t_{Pff} + t_{PCOMB\ (max)} + t_{SU} + skew$$

De la ecuación se deduce que el *skew* penaliza la frecuencia máxima de operación y por tanto debe intentarse que su valor sea lo más pequeño posible.

Ejemplo

Conociendo los siguientes datos de un circuito síncrono:

Tiempos característicos de los flip-flops: $t_{P\ FF\ max} = 3$ *ns;* $t_{SU\ FF} = 2$ *ns;* $t_{H\ FF} = 1$ *ns*

Tiempos de propagación de la lógica combinacional: $T_{P\ COMBmax} = 35$ *ns*

Skew máximo de reloj: 3 ns

Se puede calcular el periodo mínimo de la señal de reloj:

$$T_{CLKmin} = T_{P\ COMBmax} + t_{P\ FF\ max} + t_{SU\ FF} + skew = 35 + 3 + 2 + 3 = 43\ ns$$

La frecuencia máxima sería:

$$F_{CLK\ max} = 1/43\ ns = 23'2\ MHz$$

Cuando el *skew* provoca un retraso en la llegada del reloj al *flip-flop* de salida, pueden surgir problemas si la entrada síncrona del *flip-flop* de salida cambia demasiado pronto, es decir, cuando el retardo de la lógica combinacional es mínimo –normalmente cuando los dos *flip-flops* están conectados directamente (Figura 33).

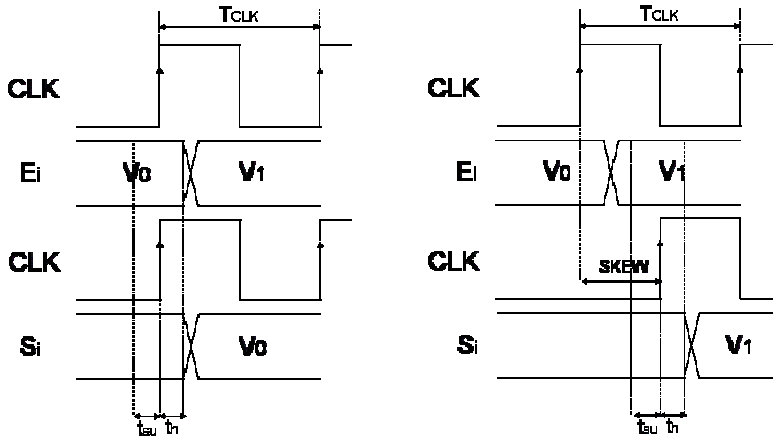

Figura 33. *Skew* positivo

En el cronograma de la izquierda se muestra el funcionamiento ideal (sin *skew*) y en el de la derecha lo que puede ocurrir si existe un desfase importante entre los relojes de los *flip-flops*. Como puede observarse, si las entradas síncronas (Ei) del *flip-flop* de salida cambian con un retardo menor que el *skew* –menos el tiempo de *set-up*-, no se registran los valores correctos (V_0), sino los correspondientes al siguiente periodo de reloj.

El ejemplo de fallo de la figura 33 es muy difícil que se dé en un circuito real, pues requiere, para que se respete el tiempo de *set-up* del segundo *flip-flop*, un *skew* de reloj muy grande y un retardo de propagación muy pequeño entre los *flip-flops* –en una conexión sin lógica combinacional este retardo es el tiempo de propagación de un *flip-flop* (el de entrada) más el tiempo de propagación de los recursos de interconexión que unan ambos *flip-flops*. Es mucho más frecuente que se dé uno de los dos casos que se muestran en la figura 34.

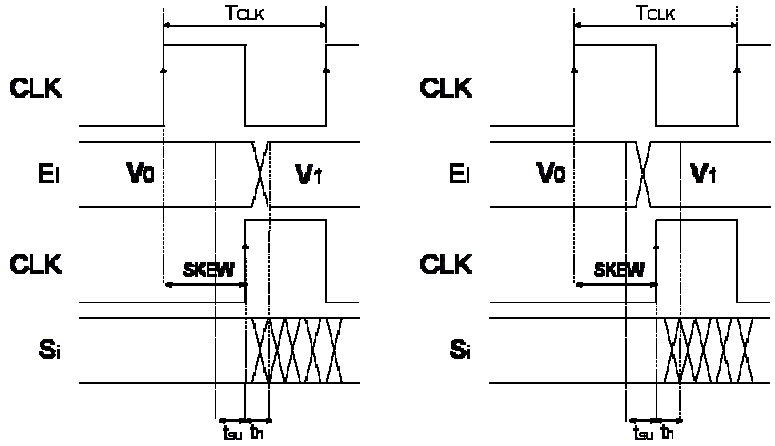

Figura 34. Fallos de funcionamiento por *skew*

En el cronograma de la izquierda no se respeta el tiempo de *hold*, en el de la derecha se viola el tiempo de *set-up*. El fallo más común es el primero, ya que es el que requiere un menor *skew* para llegar a producirse, y es el que permite establecer una cota para el máximo valor que puede tener. El valor de dicha cota es fácilmente deducible (Figura 35).

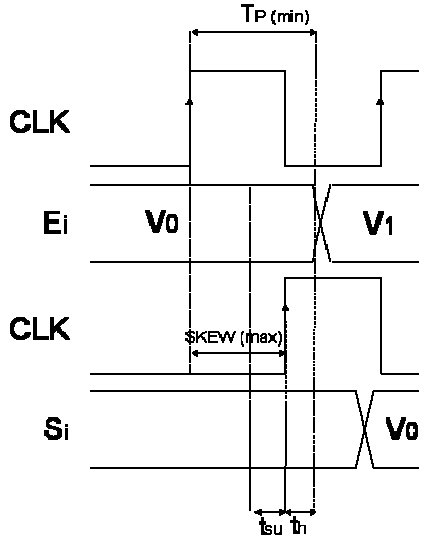

Figura 35. *Skew* máximo

Suponiendo que el tiempo de propagación mínimo –pues cuanto más pequeño sea, más le afectará el skew- entre dos *flip-flops* es T_{Pmin} (incluyendo todos los componentes de retardo entre la salida de uno y la entrada de otro), el reloj del segundo podrá desplazarse respecto al primero, garantizando la captura del dato correcto y que se respeta el tiempo de *hold*, una cantidad de tiempo ($skew_{max}$) igual a (Figura 29):

$$Skew_{max} = T_{Pmin} - t_H$$

Nótese que en este caso el problema es tanto mayor cuanto menor sea el retardo de la lógica y, por tanto, la estimación del valor máximo de *skew* requiere el uso de los tiempos mínimos de propagación y debe evaluarse en la interconexión del circuito que, en esas condiciones, tenga un tiempo de propagación menor. Es importante señalar, por último, que este problema, si se da, sólo puede resolverse reduciendo el valor del *skew* del circuito.

Ejemplo

Conociendo los siguientes datos de un circuito síncrono:

Tiempos característicos de los flip-flops: $t_{P\ FF\ min} = 1$ *ns;* $t_{SU\ FF} = 2$ *ns;* $t_{H\ FF} = 1$ *ns*

Tiempos de propagación de la lógica combinacional: $T_{P\ INTERNO\ min} = 3$ *ns*

Se puede garantizar que el skew de circuito no va a afectar a su correcto funcionamiento si su valor es menor que:

$$Skew_{max} = T_{Pmin} - t_H, \text{ donde } T_{Pmin} = t_{P\ FF\ min} + T_{P\ COMB\text{-}INT\ min}$$

$$Skew_{max} = (1 + 3) - 1 = 3 \text{ ns}$$

Cuando la tecnología de realización es una FPGA o un PLD, el problema de *skew* viene resuelto por la disponibilidad de entradas dedicadas que tienen un retardo equilibrado, y pequeño, hasta la entrada de reloj de todos los *flip-flops* del circuito; en los ASICs se dispone de *buffers* con gran *fan-out* que facilitan el equilibrado de la red de distribución del reloj –más adelante se describirá cómo debe abordarse el diseño de esta red.

Para terminar con la exposición de la casuística asociada a la interconexión del reloj, hay que decir que aunque el *skew* sea pequeño, también puede resultar problemático que el retardo de propagación desde la entrada de reloj hasta los *flip-flops* sea grande. Este retardo da lugar a que aumente también el de las salidas del sistema en relación con el reloj de entrada, pudiendo dificultar el logro de los requisitos de retardo en las salidas. En algunos dispositivos lógicos se dispone de recursos especiales (PLLs o DLLs) que permiten solventar este tipo de problemas.

2.7 Circuitos síncronos con dos fases de reloj

Es posible plantearse la realización de circuitos síncronos en el que se empleen *flip-flops* activos por flanco de subida y *flip-flops* activos por flanco de bajada. En estos circuitos aparecen conexiones como las de la figura 36.

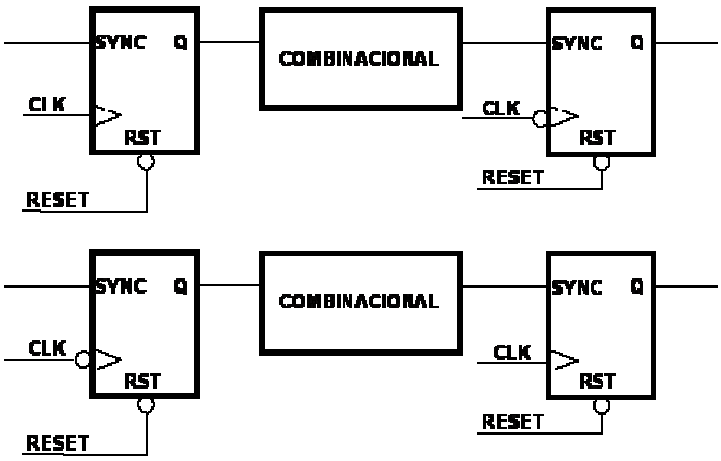

Figura 36. Dos fases de reloj

La ventaja aparente del modo de funcionamiento de estos circuitos sobre los que usan una sola fase de reloj es que realizan dos ciclos de operaciones en cada periodo de reloj: el primero en el tiempo que va del flanco de subida al de bajada y el segundo entre éste y el siguiente flanco de subida –para lo que resulta imprescindible, en cualquier caso, ir alternando sistemáticamente *flip-flops* manejados por flanco de subida y de bajada. El problema asociado a este planteamiento es que si, por ejemplo, las entradas de un circuito combinacional cambian en los flancos de subida, sus salidas deben ser estables antes del flanco de bajada y cumplir el tiempo de *set-up* para poder actuar sobre las entradas síncronas de los *flip-flops* activos en los flancos negativos; para ello, el tiempo entre un flanco de subida y de bajada –en un reloj con un ciclo de trabajo del 50%- debería ser igual al periodo mínimo de reloj en un circuito semejante que funcionara con una sola fase de reloj. Extrapolando esta situación a los elementos del circuito que funcionan al revés –es decir, aquellas en las que las entradas de los combinacionales se mueven en flancos de bajada y actúan en flancos de subida-, la conclusión que se obtiene es que, siendo cierto que se realizan dos operaciones en cada periodo de reloj, éste tiene que ser ahora mayor, la frecuencia menor y la mejora de rendimiento discutible: dependerá de cada circuito concreto y de las características específicas del reloj del circuito –de su ciclo de trabajo y de la estabilidad del mismo .

El uso de dos fases de reloj en el desarrollo de un circuito síncrono es, en principio, desaconsejable: acarrea un funcionamiento más complejo, la necesidad de utilizar dos tipos de *flip-flops* (o de dos relojes desfasados medio periodo), complicaciones en la distribución del reloj, no garantiza siempre una mejora de rendimiento y obliga a tener que considerar como un problema de diseño el valor y la estabilidad del ciclo de trabajo de la señal de reloj. Sin embargo, hay casos en que resulta necesario recurrir a esta práctica: cuando se dan es recomendable utilizar una de las fases (flancos) para manejar la mayor parte del sistema y hacer un análisis de tiempo separado para los bloques en que se conectan *flip-flops* que funcionan con flancos distintos.

El circuito de la figura EP.24 está realizado con flip-flops activos por flanco de subida (los que sincronizan las entradas y registran la salida) y de bajada[5] *(los registros internos). El circuito realiza un cálculo aritmético en dos pasos: en el primero suma, por un lado, dos datos de entrada, A y B, y, por otro, multiplica B por 3; en el segundo obtiene el valor de salida sumando el resultado de las operaciones realizadas en el primer paso.*

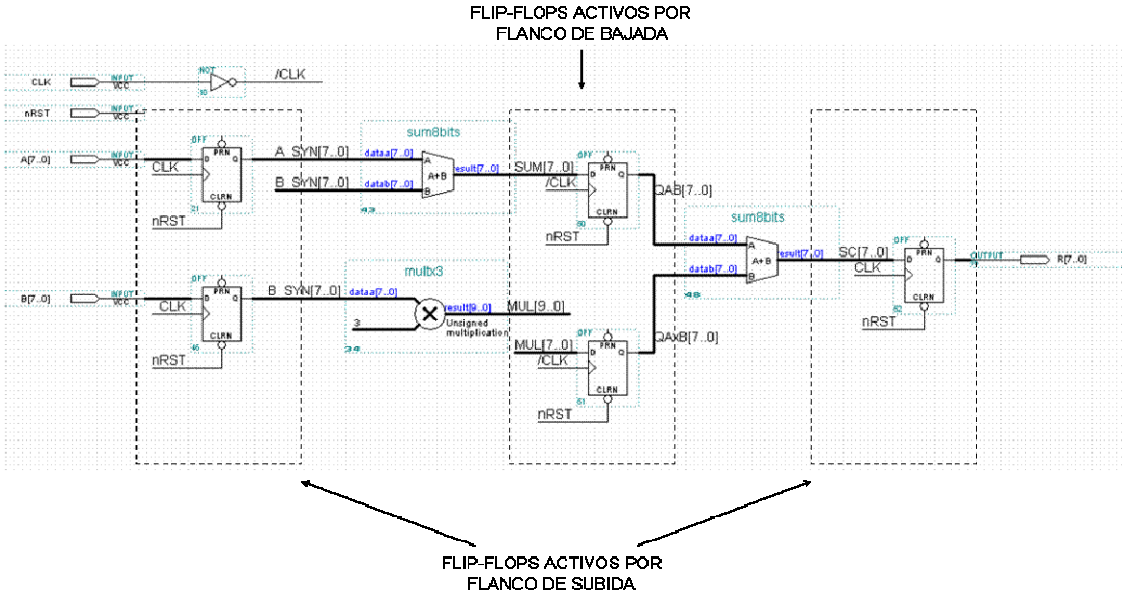

Figura EP. 24

El cronograma de la figura EP.25 muestra el resultado de una simulación del funcionamiento descrito, para una realización sobre una FPGA de ALTERA.

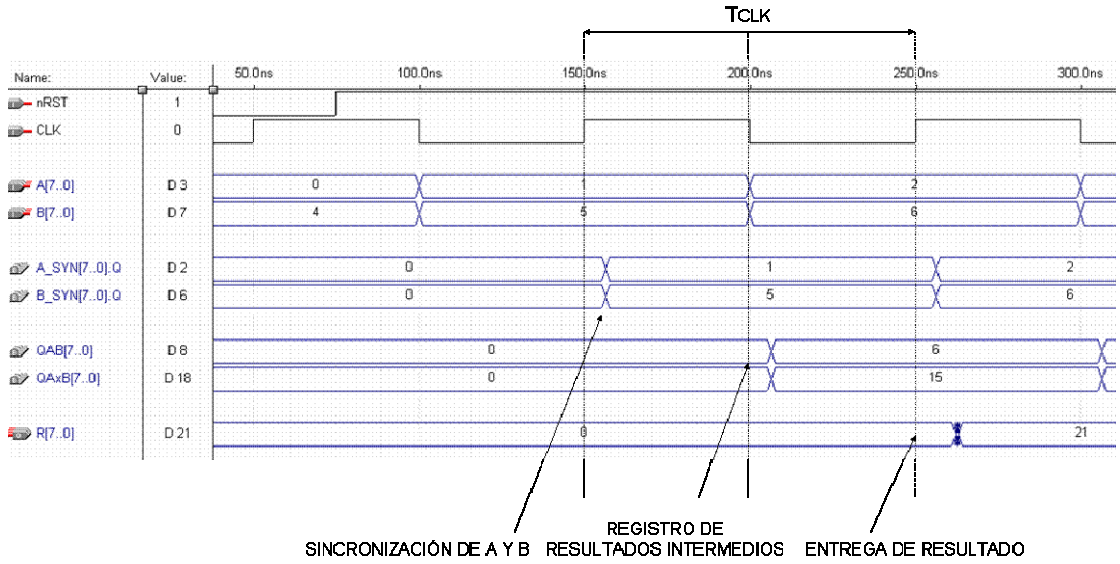

Figura EP.25

[5] El esquema de la figura EP.26 está realizado con el editor del entorno MAX+plus II, en el que no se dispone de *flip-flops* disparados por flanco de bajada; los que se comportan como tales son los que tienen conectada la entrada de reloj invertida (/CLK).

Como puede verse, la operación se completa en un tiempo igual a un ciclo de reloj, por lo que resulta aparentemente más rápida que si se realizara en un circuito análogo en el que todos los flip-flops fueran activos por flanco de subida –en ese caso, la operación tardaría en completarse dos ciclos de reloj (figura EP.28).

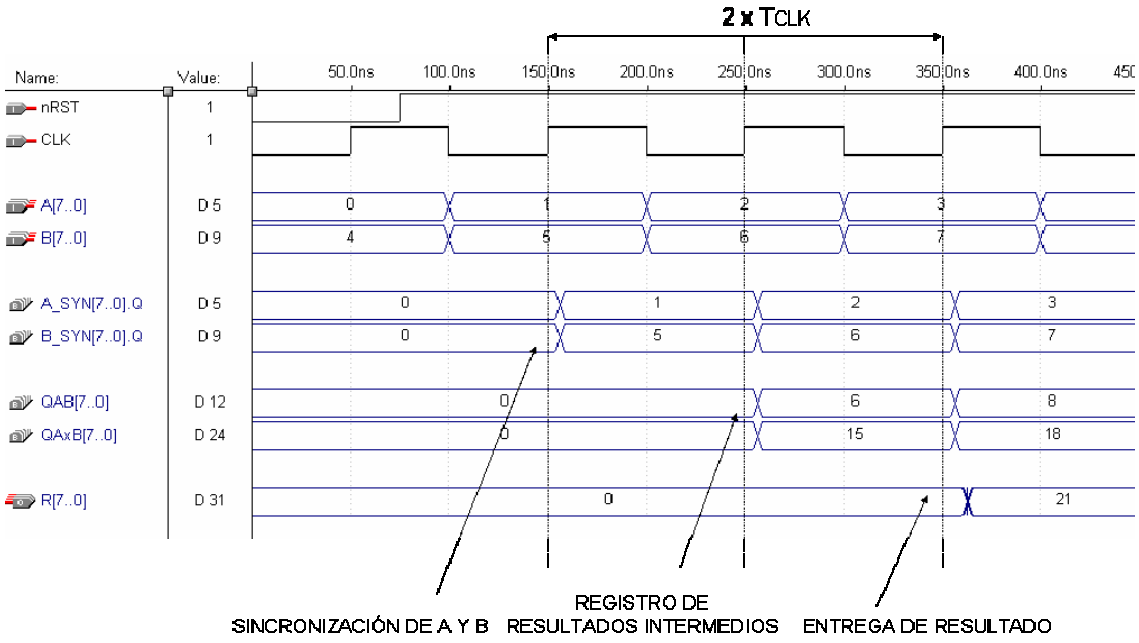

Figura EP.28

La realización de un análisis de tiempo para calcular el periodo mínimo del reloj en las dos versiones del circuito pone de manifiesto que no se consigue ninguna ventaja por utilizar dos fases de reloj: en este caso se obtiene un valor[6] para el periodo mínimo de reloj de 47'6 ns (la frecuencia máxima sería de 21 MHz). Si, en cambio, todos los flip-flops son activos por flanco de subida, el periodo mínimo sería de 23'8 ns (42 MHz de frecuencia máxima), es decir, este circuito es el doble de rápido que el anterior. En conclusión: la velocidad de ambos circuitos es la misma.

[6] Para un reloj con un ciclo de trabajo del 50%

3. Diseño de circuitos Síncronos

En este capítulo nos vamos a ocupar de las soluciones que se utilizan habitualmente para sortear las limitaciones impuestas por las reglas de diseño síncrono. Las vamos a dividir en dos partes: las soluciones que están relacionadas con los requisitos de funcionamiento (apartado 3.1) y las que afectan a restricciones de tiempos y a la velocidad de operación (apartado 3.2). En cada caso de estudio se explicará el problema planteado por las reglas de diseño, se propondrán soluciones y se analizarán las características de las mismas.

3.1 Soluciones funcionales para el diseño de circuitos síncronos

La aplicación de las reglas para el diseño de circuitos síncronos da lugar a restricciones que condicionan el tipo de circuitos que pueden idearse para conseguir una determinada funcionalidad. En muchas ocasiones pueden impedir la realización de la solución funcional encontrada por el diseñador; cuando esto ocurre es necesario encontrar un diseño alternativo que satisfaga las reglas de diseño, aunque en algunas ocasiones –afortunadamente raras- no queda más remedio que infringir, siempre como último recurso y con mucho cuidado, las reglas de diseño síncrono.

3.1.1 Habilitación de reloj

En un circuito síncrono está prohibido que la salida de un circuito combinacional o de un *flip-flop* pueda conectarse a la entrada de reloj de un *flip-flop*, la cual, por otra parte, ha de estar forzosamente conectada a la señal global de reloj. Esto impide controlar la operación de un circuito secuencial generando un flanco de reloj cuando se da una combinación de valores en un dato o se pasa por determinado estado en un módulo secuencial (figura 37).

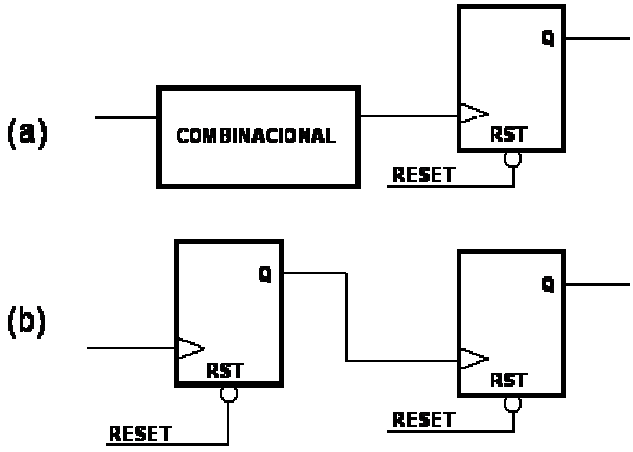

Figura 37. Relojes asíncronos

Como ya se sabe, en el circuito de la figura 37a hay riesgo de que el combinacional genere *glitches* y, en consecuencia, de que el secuencial ejecute operaciones indeseadas. Esto no puede ocurrir, en cambio, en el circuito de la figura 37b, ya que la salida que controla el reloj está registrada; el problema de esta idea es que genera un *skew* de reloj importante que afecta al secuencial controlado (figura 38), desplazando el instante de conmutación de la lógica a la que pueda estar conectada su

salida –todo esto se agravaría y podría alegarse también en contra del circuito de la figura 37a, por lo que resultaría inaceptable aún en el caso de que no hubiera *glitches* en la salida del combinacional.

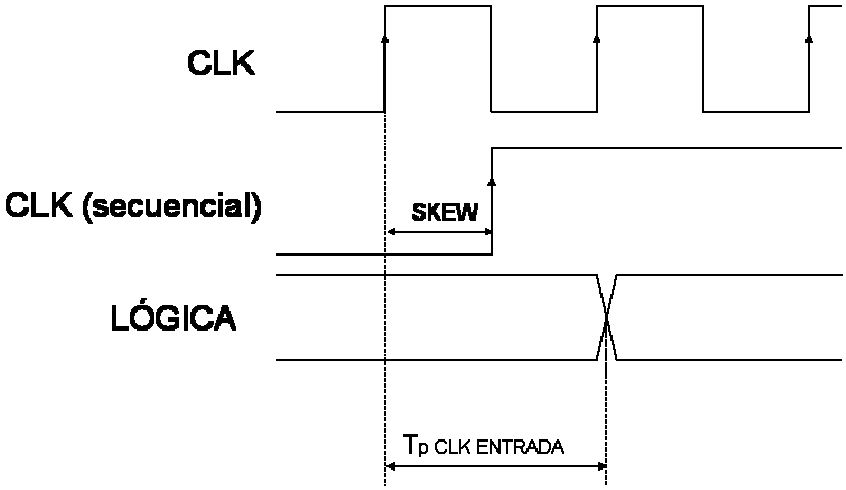

Figura 38. *Skew*

Las alternativas idóneas para estas estructuras de conexión en un circuito síncrono se basan en utilizar circuitos secuenciales con habilitación de reloj (*clock enable*). Una entrada de habilitación de reloj es una entrada síncrona que funciona del siguiente modo: si está activa, el secuencial responde, de acuerdo con su función, a los niveles lógicos de sus entradas –síncronas- en los flancos activos de reloj, si no lo está, el secuencial mantiene su estado. En el caso de un *flip-flop* tipo D con *clock enable*, por ejemplo, cuando la entrada de habilitación está activada se captura el dato de entrada, en caso contrario se mantiene el dato almacenado independientemente del nivel lógico presente en la entrada D del *flip-flop*. En la figura 39 se ilustra el modo de operación de este tipo de *flip-flops*.

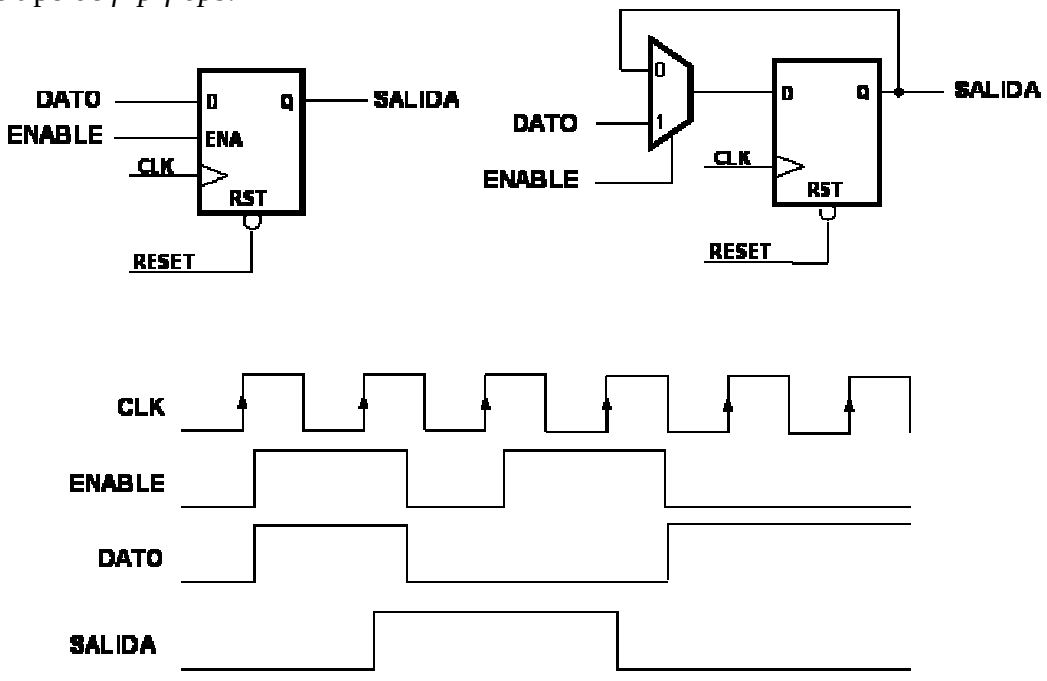

Figura 39. *Flip-flop* con habilitación de reloj

En algunas tecnologías se dispone de *flip-flops* con entrada de habilitación en la colección de módulos básicos de diseño, en otras es necesario construirlos utilizando un *flip-flop* tipo D y un multiplexor de dos canales, tal y como se muestra, también, en la figura 39.

La versión síncrona del circuito de la figura 37a se muestra en la figura 40, donde se compara además el modo de funcionamiento en ambos casos.

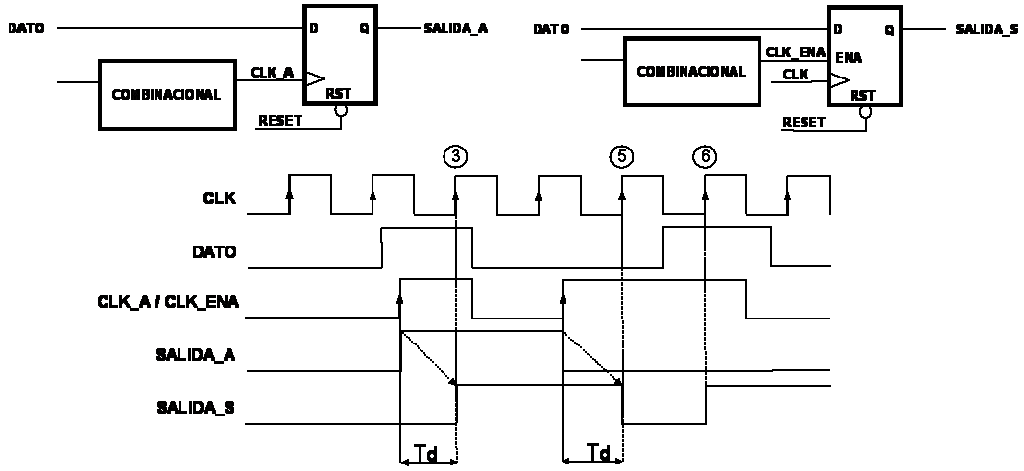

Figura 40

En el cronograma de la figura 40 puede observarse que si la salida del circuito combinacional está activa en un solo flanco (como en el número 3 del cronograma) ambos circuitos son funcionalmente equivalentes, con la diferencia de que en la versión síncrona la salida no cambia hasta que llega el flanco de reloj –esto es lógico, es consecuencia del hecho de que en un sistema síncrono el retardo "efectivo" de cualquier módulo combinacional es de un ciclo de reloj. Esta equivalencia desaparece si la salida del combinacional está activa durante varios flancos (en el cronograma del ejemplo, en los flancos 5 y 6); cuando esto ocurre, para obtener un comportamiento análogo al de la versión asíncrona, es necesario modificar la salida del circuito combinacional para que sólo esté activa en un flanco de reloj. El circuito que suele utilizarse para realizar este acondicionamiento se muestra en la figura 41.

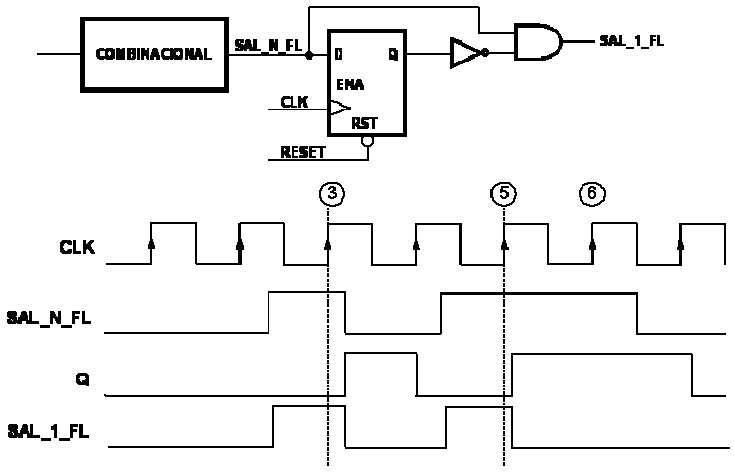

Figura 41. Conformación de pulsos de habilitación de reloj

Este circuito genera, sin retardar la actividad del circuito combinacional, un pulso que está activo un solo flanco de reloj, independientemente de la duración del pulso de salida del circuito combinacional. Intercalándolo en la versión síncrona del circuito de la figura 37a se obtiene ahora un funcionamiento completamente equivalente -que se muestra en el cronograma de la figura 42.

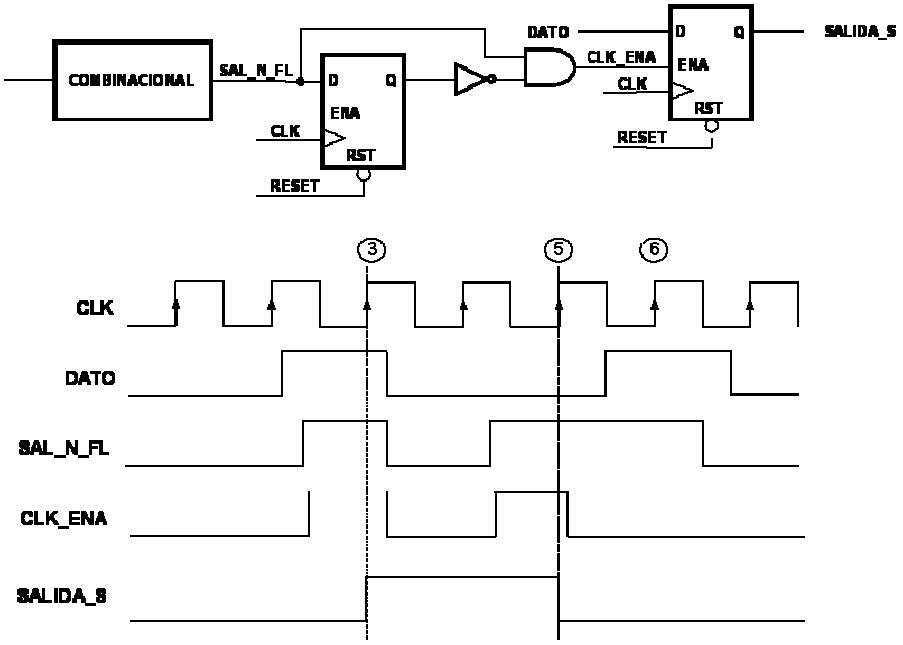

Figura 42

La versión síncrona del circuito de la figura 37b sería idéntica a la que acabamos de proponer para el de la figura 37a; presenta la misma problemática y sólo cabe destacar una diferencia: en la versión síncrona el control del secuencial se demora un ciclo entero de reloj respecto a la versión asíncrona. Este detalle debe tenerse muy en cuenta a la hora de ajustar la temporización del circuito.

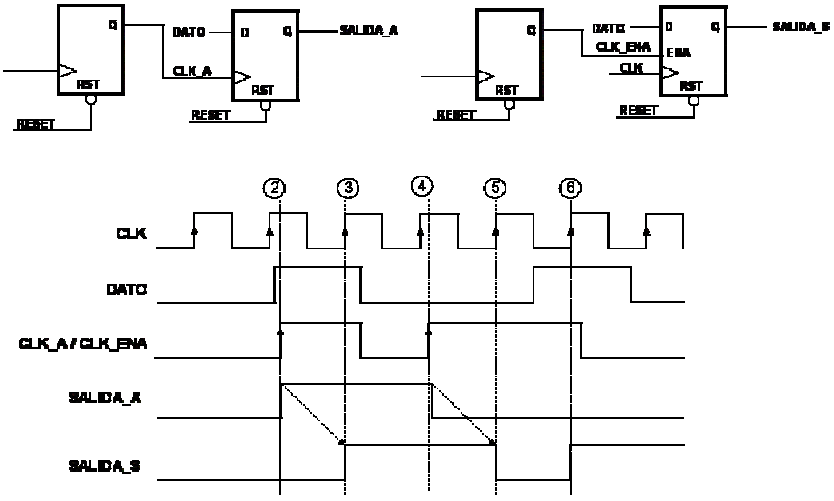

Figura 43

En la figura 43 se ilustra este modo de funcionamiento. Observe que, al igual que en el caso anterior, si la salida del secuencial que controla el *clock enable* dura más

de un ciclo de reloj, el funcionamiento de los dos circuitos difiere. Se puede utilizar, también aquí, el circuito conformador del ancho de pulso para que la señal de habilitación actúe sólo en un flanco, aunque en este caso es frecuente que pueda modificarse ligeramente el secuencial que genera la habilitación para que esta señal dure siempre un ciclo de reloj –con lo ahorramos recursos lógicos en la realización del sistema.

Por ejemplo, el contador del circuito de la figura EP.29 debe incrementar su valor cada vez que un dato de 8 bits toma un valor mayor que 100. Para realizar esta función se dispone de una lógica que detecta cuando hay un cambio en el valor del dato de entrada (señal CAMBIO) y de un comparador que activa su salida cuando el dato de entrada es mayor que 100. Cuando se cumplen ambas condiciones se genera un pulso que actúa sobre la entrada de reloj del contador.

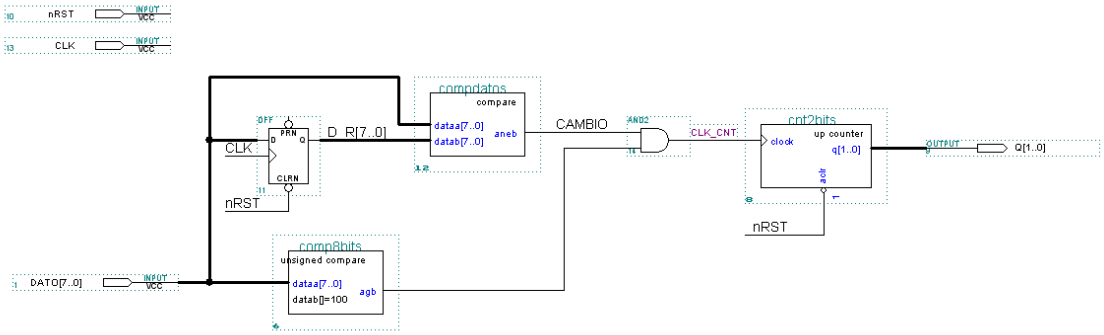

Figura EP.29

El cronograma de la figura EP.30, correspondiente a una simulación con retardos del circuito, muestra los problemas de funcionamiento que ocasionan los glitches de la señal que maneja el reloj del contador: en las conmutaciones del dato de entrada puede incrementarse la cuenta sin que se cumpla la condición de que el dato de entrada sea mayor que 100.

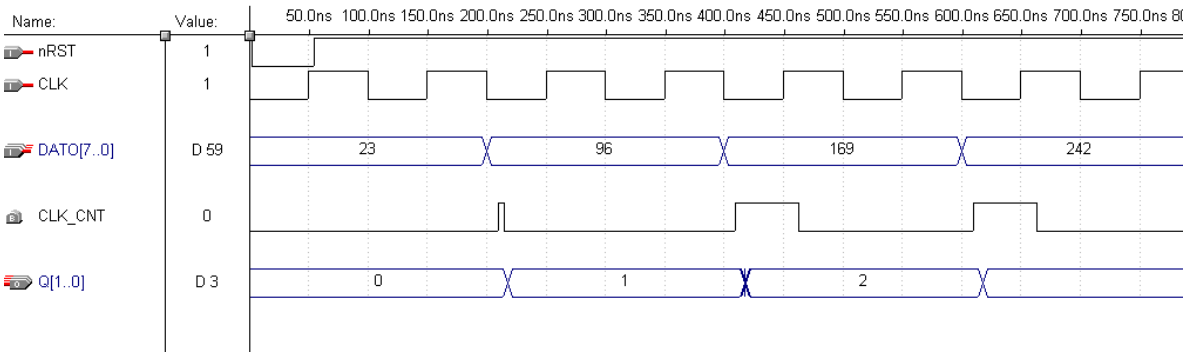

Figura EP.30

Los glitches podrían eliminarse registrando la señal que actúa como reloj del contador, tal y como se muestra en la figura EP.31.

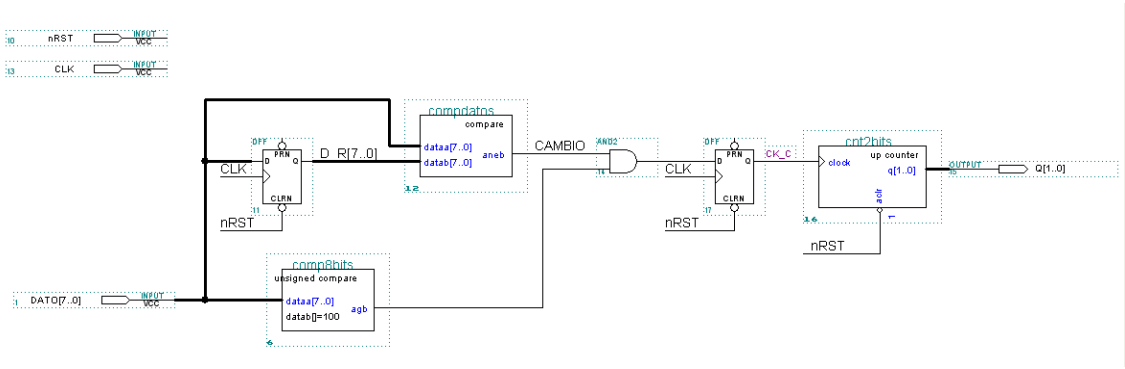

Figura EP.31

El flip-flop filtra los glitches y restablece el correcto funcionamiento del circuito, pero el reloj del contador (CLK_C en el cronograma de la figura EP.32) está retrasado, tiene skew, respecto al global del circuito. En un sistema complejo este tipo de desajustes no resultan admisibles.

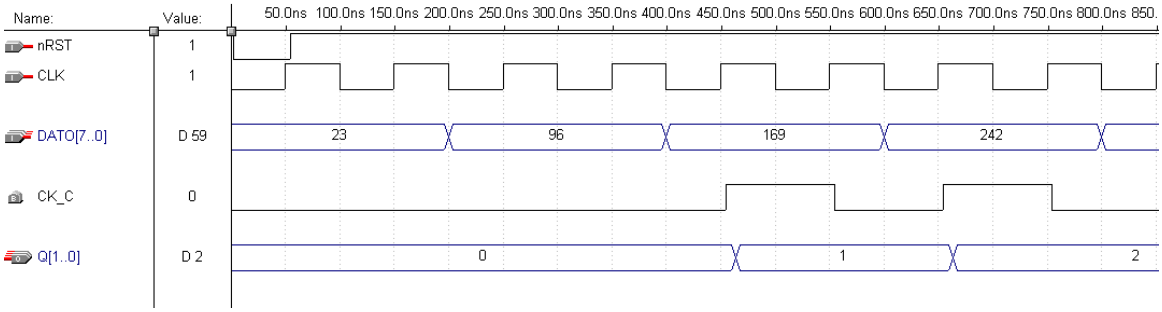

Figura EP.32

La solución al problema se reduce a utilizar flip-flops con habilitación de reloj en el contador y conectar a la entrada de habilitación la señal que controla el incremento de cuenta, tal y como se muestra en la figura EP.33.

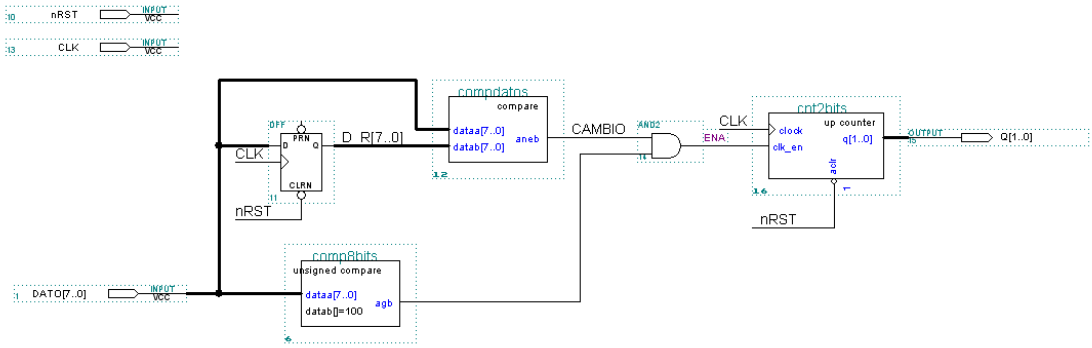

Figura EP.33

El funcionamiento de este circuito es equivalente desde el punto de vista funcional al de los dos anteriores (como puede observarse en el cronograma de la figura EP.34), pero no tiene ninguno de los inconvenientes de aquellos.

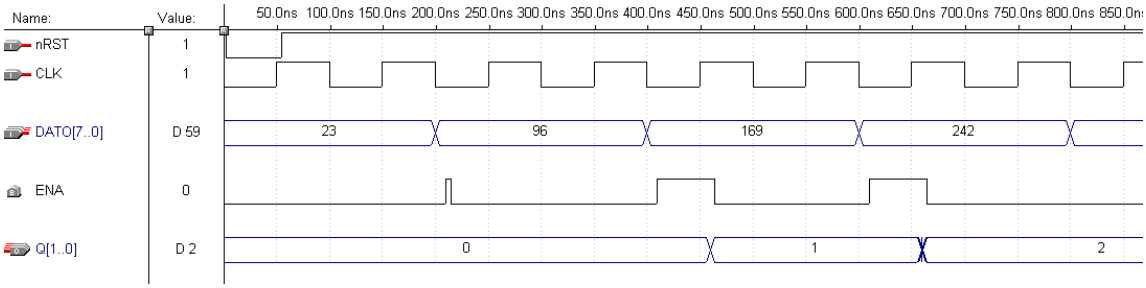

Figura EP.34

Otro ejemplo: la realización del circuito de la figura EP.35 incumple las reglas de diseño síncrono. La entrada INC, una vez registrada, funciona como reloj del primer contador, la condición de fin de cuenta de éste es, a su vez, el reloj del segundo contador.

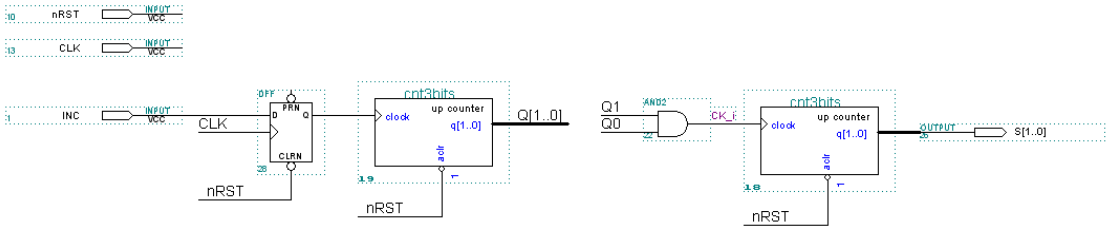

Figura EP.35

El cronograma adjunto (figura EP.36) muestra una simulación del funcionamiento del circuito.

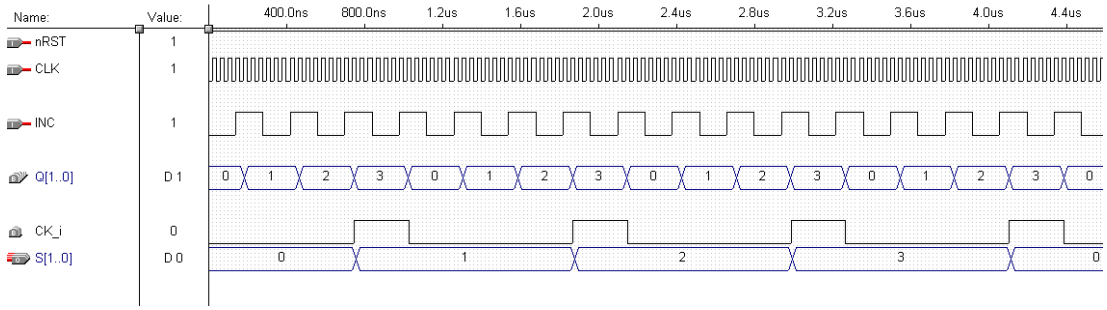

Figura EP.36

Para realizar una versión síncrona del circuito hay que disponer de entradas de habilitación de reloj en los contadores y conectarlas a las señales que en el circuito original funcionan como reloj. En la figura EP.37 se muestra el circuito descrito.

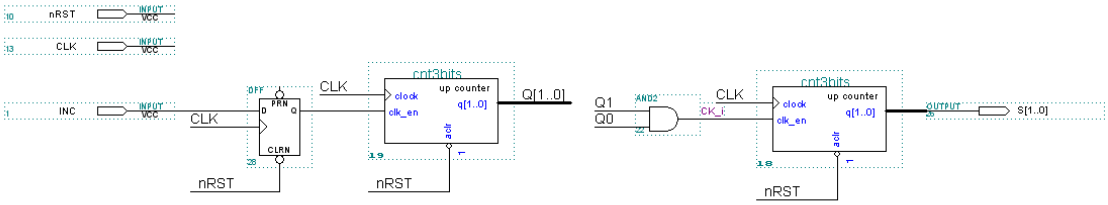

Figura EP. 37

Este circuito no funciona igual que el original porque las entradas de habilitación de reloj pueden estar activas durante varios flancos de reloj. En el cronograma de la figura EP.38 puede observarse como los contadores incrementan su cuenta mientras las entradas de habilitación están activas.

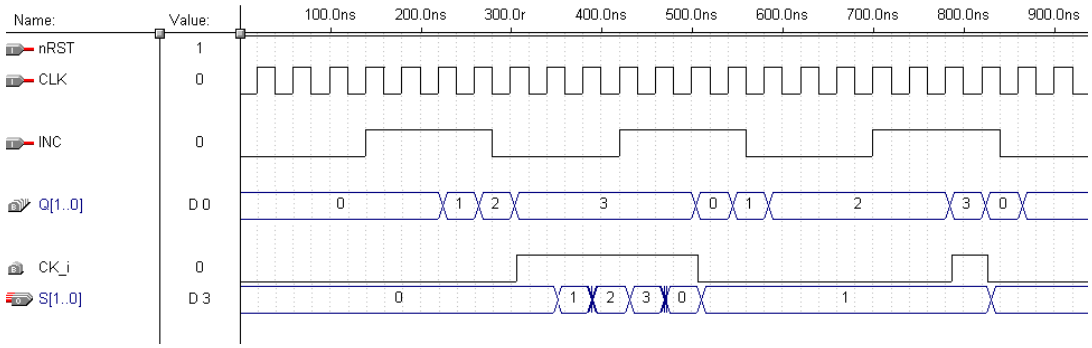

Figura EP.38

Para obtener un funcionamiento equivalente al del circuito asíncrono hay que acondicionar las entradas de habilitación para que sus pulsos duren un periodo de reloj. Esto puede hacerse con ayuda del circuito descrito en el apartado 3.1.1. En la figura EP.39 se muestra la versión síncrona con un funcionamiento idéntico al original.

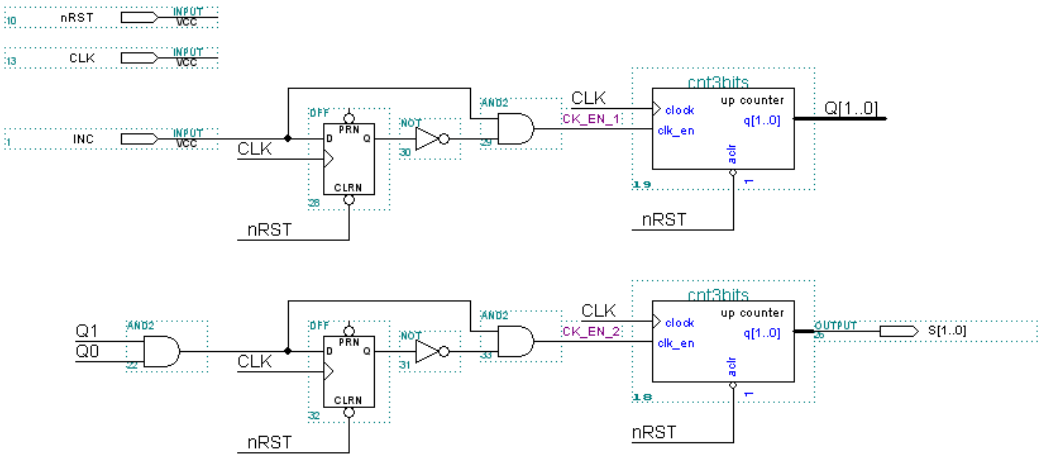

Figura EP.39

La lógica que acondiciona la señal de habilitación del segundo contador puede simplificarse. Si se desea que el segundo contador se incremente cuando el primero alcance el último estado de cuenta, podría modificarse el circuito tal y como se muestra en la figura EP.40.

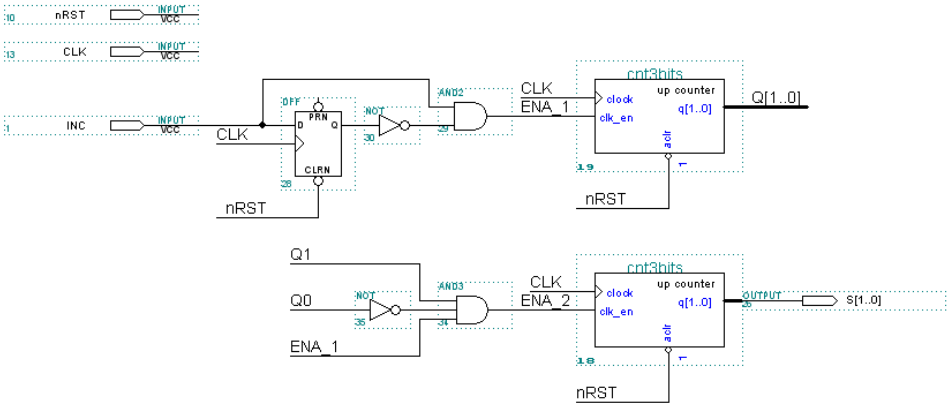

Figura EP.40

Este tipo de soluciones, que ahorran el empleo del flip-flop del circuito de sincronización, suelen ser factibles cuando la señal de habilitación es la salida de un flip-flop o, en general, la salida de un circuito secuencial con arquitectura de Moore. En el cronograma de la figura EP.41 se muestra una simulación del funcionamiento de esta última versión del circuito.

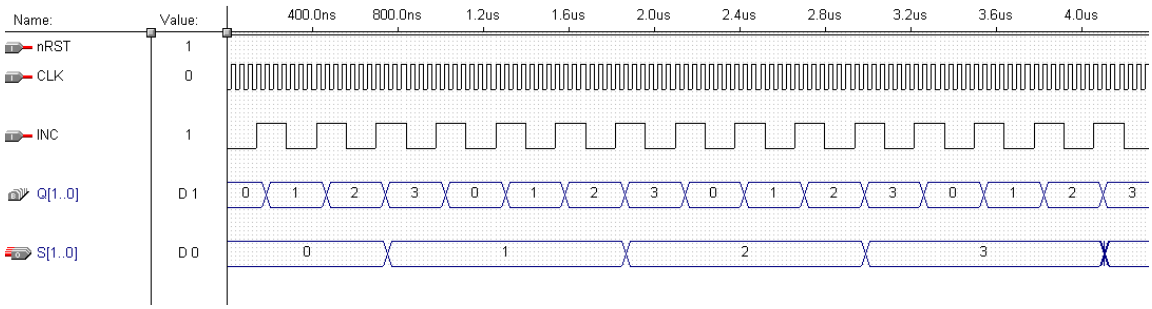

Figura EP.41

3.1.2 Reset y Preset síncronos

Mientras un circuito digital está funcionando resulta, ocasionalmente, necesario inicializar los *flip-flops* de algunos módulos secuenciales -por ejemplo: para establecer un determinado estado de cuenta (en un contador), asignar un valor por defecto a un dato (en un registro) o llevar al estado inicial cierta lógica de control. La forma más simple de fijar un estado lógico en un *flip-flop* consiste en actuar sobre sus entradas de inicialización asíncronas de *reset* o *preset*. Este tipo de acción está prohibida por las reglas de diseño síncrono -por motivos análogos a los señalados en el apartado anterior para el manejo de la señal de reloj mediante salidas de módulos combinacionales o secuenciales: se produciría un *skew* derivado de que las salidas de los *flip-flops* podrían cambiar en instantes distintos a los flancos de reloj o, en el caso de que una entrada

asíncrona de un *flip-flop* estuviera conectada a la salida de un combinacional con *glitches*, podría producirse un cambio de estado indeseado en los *flip-flops*.

Para cubrir la necesidad de inicialización de los *flip-flops* del circuito durante su modo de operación normal –entiéndase: cuando no está siendo inicializado por la señal de *reset* global, que sí puede (y, en general, debe) actuar sobre las entradas de inicialización asíncronas- deben utilizarse *flip-flops* con entradas síncronas de *reset* y/o *preset*. Este tipo de *flip-flops* no están disponibles como recursos primitivos de diseño en ninguna tecnología, por lo que deben construirse combinando *flip-flops* "simples" con puertas lógicas. En la figura 44 se muestran algunos ejemplos de circuitos que incorporan entradas síncronas de inicialización.

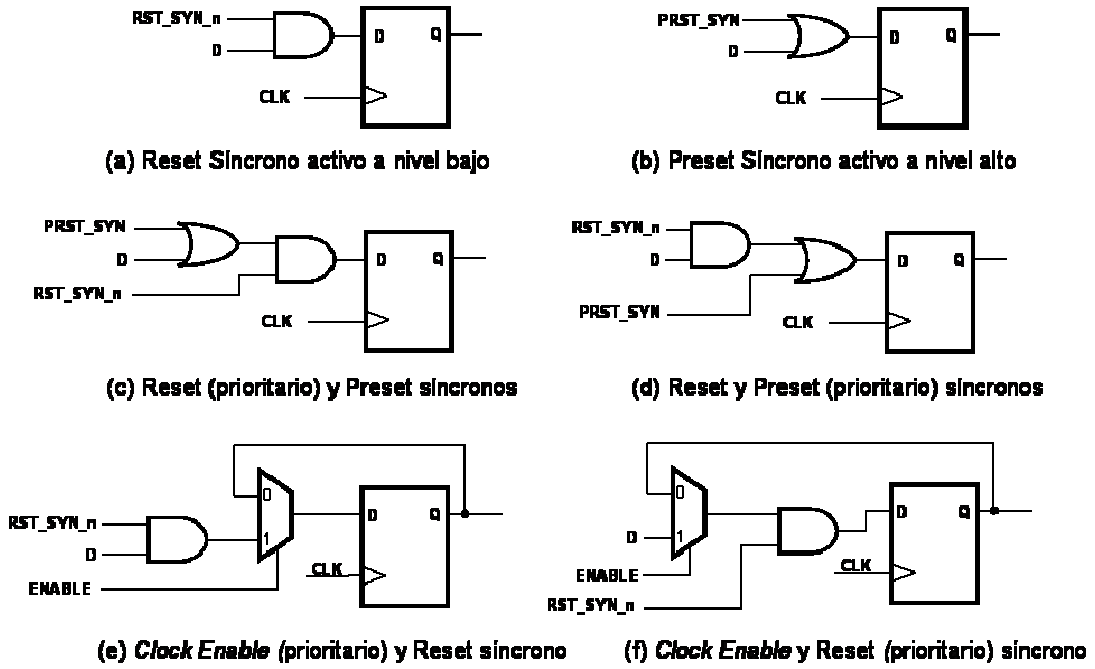

Figura 44. *Flip-flops* con entradas de inicialización síncrona

Como puede observarse en la figura, pueden construirse *flip-flops* que incorporen sólo una entrada síncrona de *reset* o *preset* (circuitos a y b), ambos tipos de entrada (circuitos c y d -pudiendo establecerse la prioridad que se desee entre ambas: en el circuito c la entrada de *reset* es prioritaria sobre la de *preset*, mientras que en el circuito d ocurre lo contrario), o combinarse con una entrada de habilitación de reloj –en este caso puede condicionarse la actuación de las entradas de inicialización a que el *flip-flop* esté habilitado (circuito e) o no (circuito f).

Desde el punto de vista funcional, la diferencia entre inicializar asíncrona o síncronamente un *flip-flop* consiste en que en la acción asíncrona el efecto de la actividad de la entrada es prácticamente instantáneo –la inicialización se verifica cuando transcurre el tiempo de propagación del *flip-flop*-, mientras que, en la síncrona, la inicialización se verifica, tras la activación de la entrada síncrona de inicialización, cuando ocurre un flanco activo de reloj. En la figura 45 se ilustra la diferencia entre ambos modos de funcionamiento.

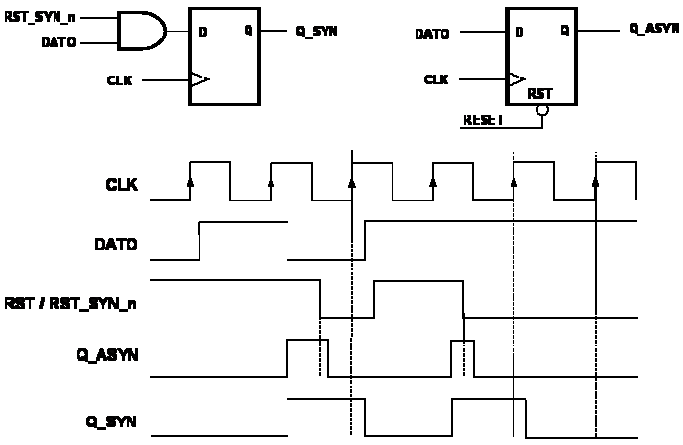

Figura 45. Inicialización síncrona y asíncrona

En el cronograma puede constatarse que la versión síncrona es equivalente a la asíncrona, independientemente de la duración de la señal de *reset*. El retardo derivado de la actuación del *reset* síncrono en los flancos de reloj puede ocasionar problemas que hay que considerar a la hora de calcular la temporización de un sistema síncrono.

Por ejemplo, el flip-flop del circuito de la figura EP.42 se pone a cero cuando el dato de entrada es menor que 100, a uno cuando es mayor que 200 y, en cualquier otro caso, mantiene su valor. El circuito infringe, evidentemente, las reglas de diseño síncrono.

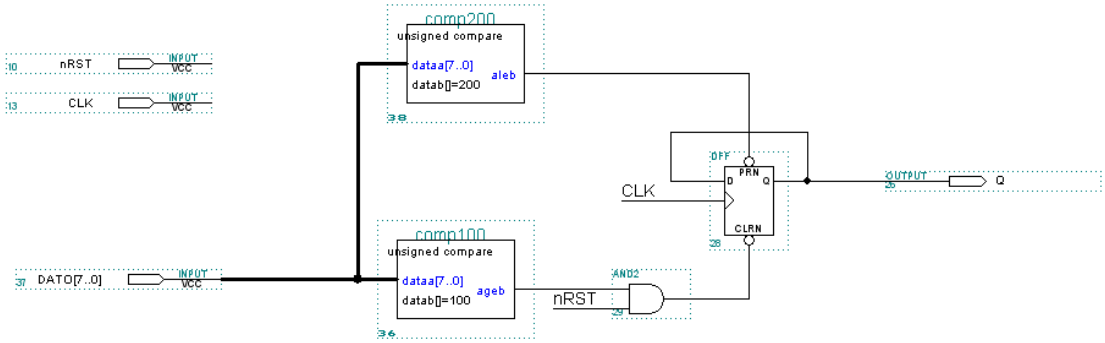

Figura EP.42

En el cronograma de la figura EP.43 puede apreciarse como los glitches en las entradas asíncronas (28:PRN y 28:CLRN) del flip-flop perturban su correcto funcionamiento.

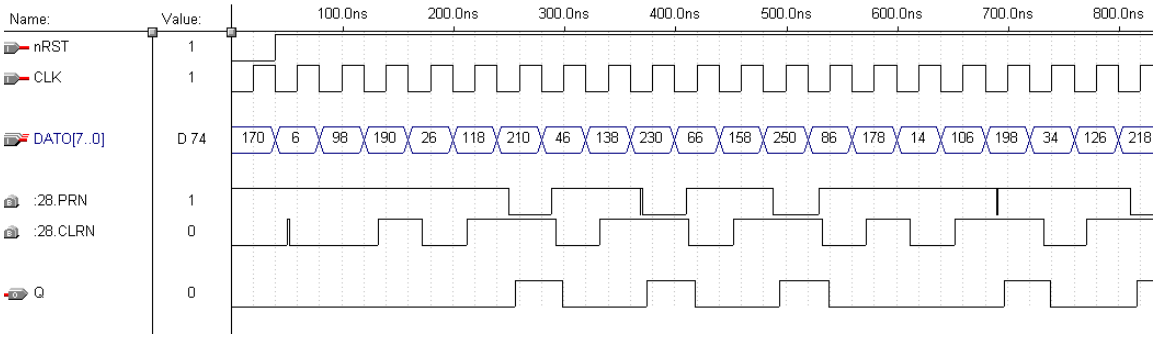

Figura EP.43

Basta con disponer de un flip-flop con reset y preset síncronos para realizar la versión síncrona del circuito. Es la que se muestra figura EP.44.

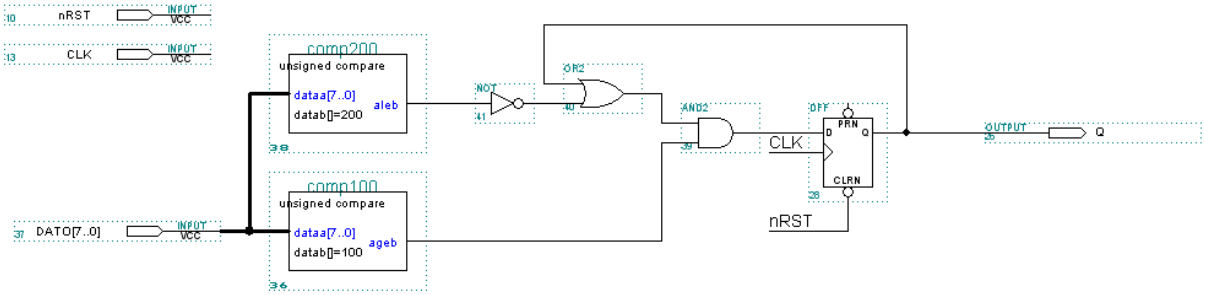

Figura EP.44

En este circuito el reset es prioritario sobre el preset –la prioridad resulta en este caso indiferente porque las dos señales no pueden estar activas al mismo tiempo- y ambas entradas son activas a nivel bajo. En el siguiente cronograma se muestra el funcionamiento –ahora correcto- del circuito.

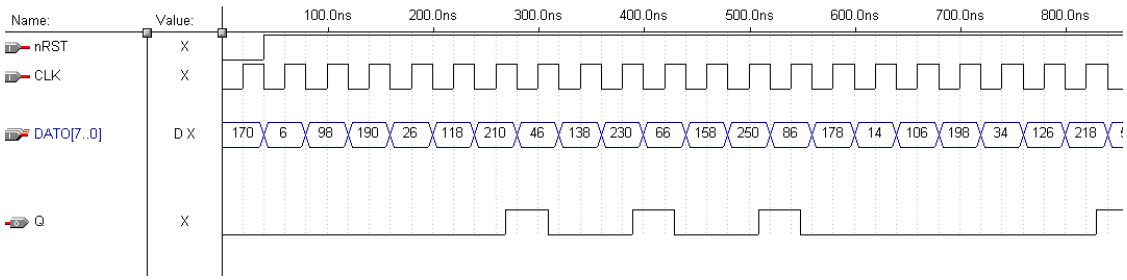

Figura EP.45

El circuito de la figura EP.46 es un contador de módulo programable. Cuando el estado de cuenta iguala el valor del módulo, el contador es inicializado asíncronamente, por lo que el circuito es sensible a los glitches que potencialmente pueda generar el comparador. Este modo de operación viola las reglas de diseño síncrono.

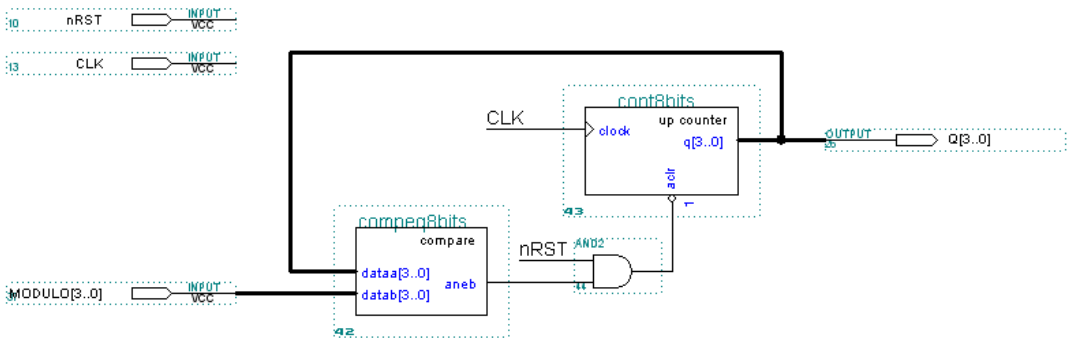

Figura EP.46

En el cronograma de la figura EP.47 se muestra una simulación del funcionamiento del circuito. Observe que en cada secuencia completa de cuenta hay un

ciclo de reloj, el último, en que el contador muestra dos estados de cuenta: el valor del módulo programado, mientras no actúa el reset asíncrono, y, después, el estado inicial, cero.

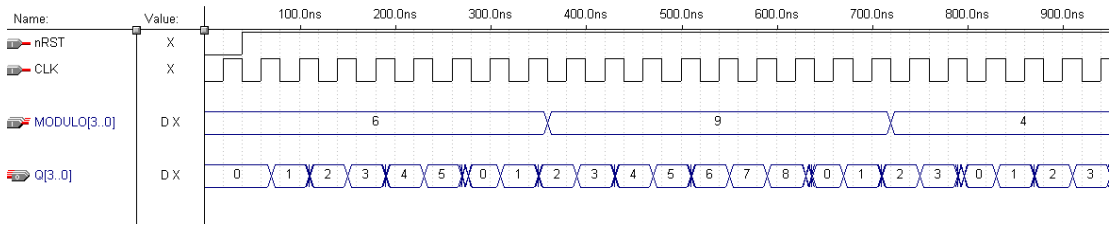

Figura EP.47

La versión síncrona del circuito se realiza dotando al contador de una entrada de inicialización síncrona. Pero ahora, para que el contador tenga el mismo módulo que en la versión asíncrona, debemos inicializarlo, en lugar de a cero, a uno. Esto es debido a que el último estado de cuenta (el valor del módulo programado) se mantiene durante un ciclo completo de reloj –ya que la señal que indica que se ha detectado este valor actúa en el siguiente flanco- por lo que si iniciáramos la cuenta desde cero, el módulo del contador sería el programado más uno. En las figuras EP.48 y EP.49 se muestra la versión síncrona del circuito y una simulación de su funcionamiento.

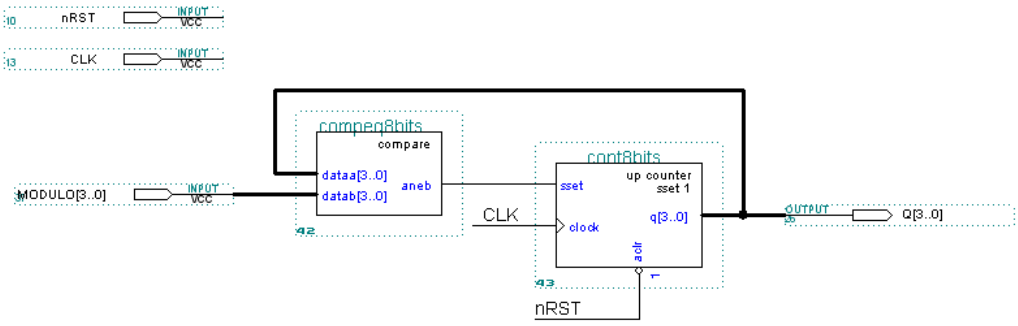

Figura EP.48

Observe que se ha modificado el reset asíncrono del contador para que también la actuación del reset global del circuito lleve el contador a uno.

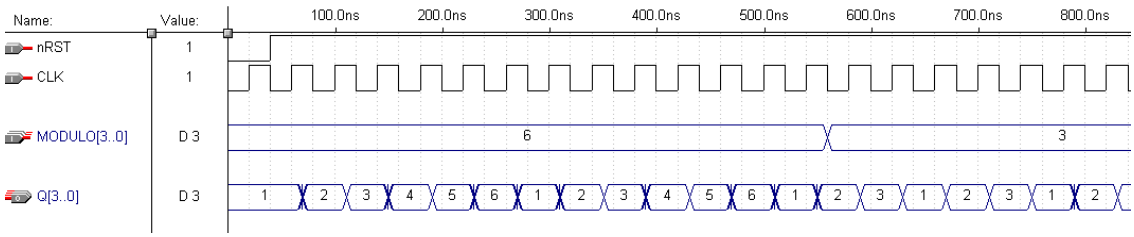

Figura EP.49

3.1.3 Sincronización de Entradas

Las reglas de diseño síncrono exigen que deba registrarse cualquier entrada asíncrona del circuito –cualquier señal manejada desde fuera del dominio del reloj del sistema síncrono- con el fin de que sus conmutaciones queden sincronizadas con el reloj del circuito. El cumplimiento de esta regla conlleva una serie de problemas que vamos a analizar en detalle en este apartado.

En los comentarios de la figura 32 -que traigo aquí para comodidad del lector- se describían dos de los efectos derivados del registro de las entradas asíncronas: los cambios de estado en las entradas se detectan, en el circuito síncrono, con cierto retardo –de valor aleatorio y acotado por el valor del periodo de reloj- y la forma de la señal (la duración de sus niveles lógicos) se modifica, ajustándose a un número entero de periodos de reloj. En el caso de que la velocidad de respuesta del sistema a los cambios en las entradas sea un requisito de funcionamiento, o cuando sea necesario evaluar la duración de los estados de la entrada con gran precisión, los efectos de la sincronización pueden repercutir seriamente sobre el diseño del circuito; pueden requerir, en concreto, que la frecuencia mínima de funcionamiento del circuito sea muy grande, un requisito que, quizás, pueda ser difícil de alcanzar si la lógica de procesamiento tiene asociados retardos de propagación altos.

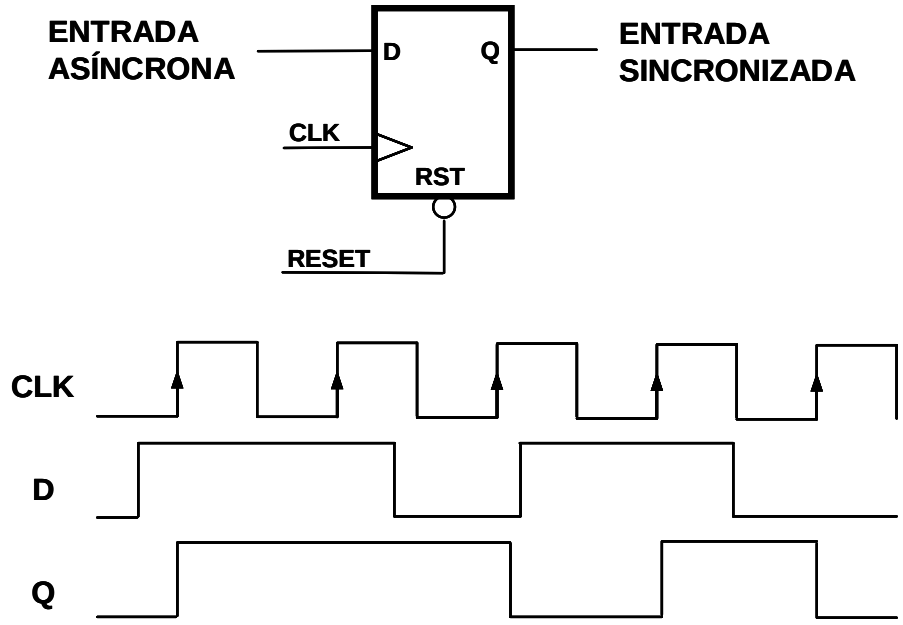

Figura 32 (repetida)

Otro problema derivado de la necesidad de sincronizar las entradas es el que nos encontramos cuando los niveles lógicos de una entrada asíncrona tienen –o puedan a veces tener- una duración menor que el periodo del reloj del circuito síncrono –bien porque sean salidas de otro sistema síncrono que funciona con una frecuencia de reloj mayor, o bien porque su duración se derive del mecanismo particular de generación de la entrada en cuestión. Este caso se ilustra en la figura 46. Como puede verse, la detección de pulsos es aleatoria –depende de si coinciden con los flancos de reloj- y no es posible, en general, discriminar la ocurrencia de trenes de pulsos.

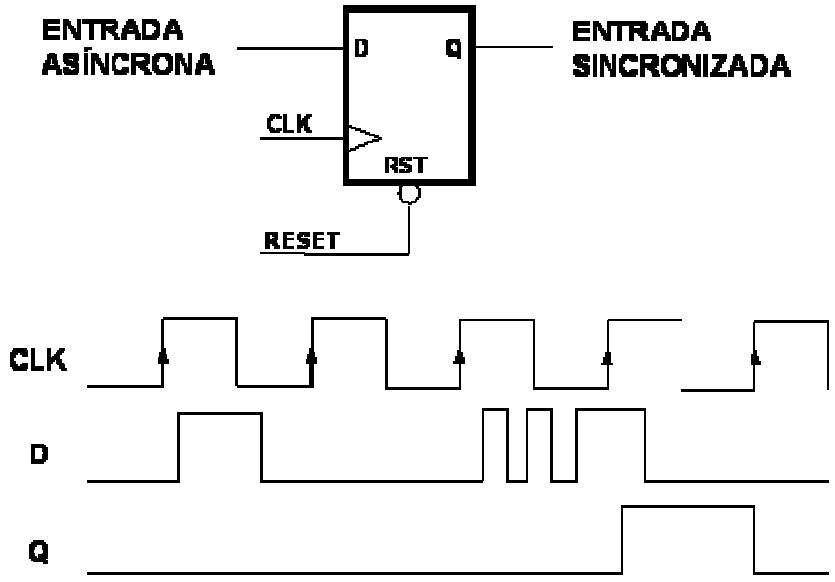

Figura 46. Pulsos de duración menor que un periodo de reloj

Una solución parcial –aunque suficiente en la mayoría de los casos- para este problema se muestra en la figura 47.

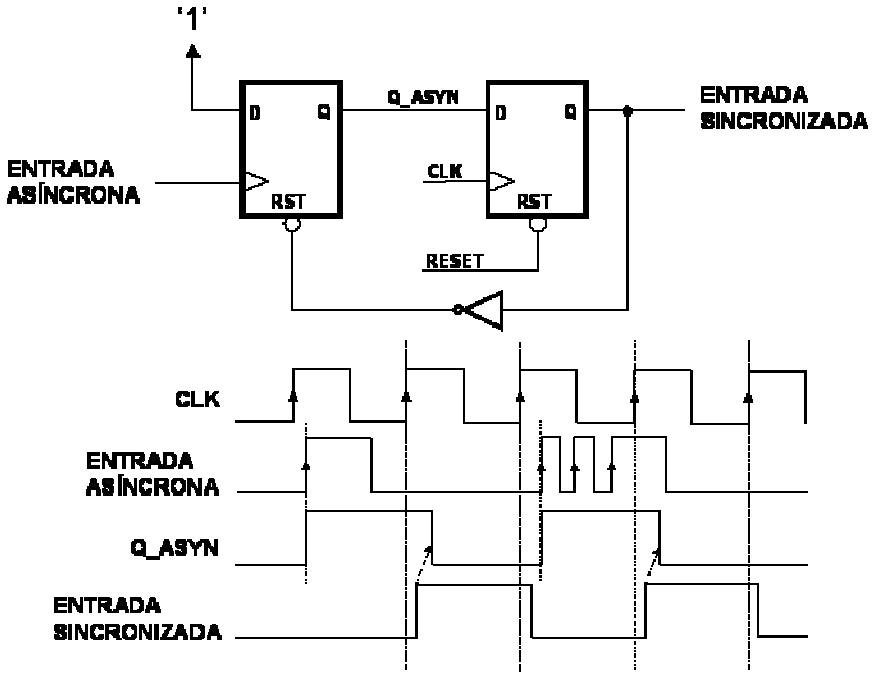

Figura 47. Sincronización de señale de duración menor que T_{CLK}

Es evidente que este circuito viola flagrantemente las reglas de diseño síncrono: el reloj del primer *flip-flop* no es el reloj global del sistema y, además, su reset asíncrono es manejado por la salida del segundo *flip-flop*. Esta infracción puede justificarse con dos argumentos: el primero, de carácter formal, es que podemos considerar que el primer *flip-flop* no es parte del circuito síncrono, sino que es una especie de adaptador externo que acondiciona la señal de entrada asíncrona e interacciona con el sistema por medio de una salida síncrona y registrada de éste que

actúa sobre su *reset* asíncrono; el segundo argumento, más prosaico –pero más didáctico e interesante- es que este circuito, aunque no se atiene a las reglas de diseño, funciona y nos permite realizar el resto del sistema sin perder las ventajas que acarrea el uso de las técnicas de diseño síncrono, porque es una solución aislada (su uso está limitado a solucionar el problema de sincronización) con una interfaz hacia el circuito (la salida del *flip-flop* manejado por el reloj global) que satisface los requisitos de operación síncrona (disponemos de una entrada sincronizada). Esta segunda justificación pone de manifiesto un hecho del que se debe ser consciente: en ocasiones resulta inevitable infringir las reglas de diseño síncrono -para realizar interfaces con elementos externos al sistema (memorias asíncronas, buses asíncronos, etc.); esto debe ser considerado por el diseñador como un "último recurso", pero un recurso al fin y al cabo. Cuando tenga que recurrir a este tipo de soluciones deberá asegurarse de que el bloque asíncrono quede aislado del resto del sistema, permitiendo que el análisis y diseño de la mayor parte del circuito pueda realizarse de acuerdo con la metodología de diseño síncrono. Debe evitarse también el uso de soluciones asíncronas donde existan versiones síncronas que respeten las reglas de diseño, aunque le parezca que la solución asíncrona es más simple o ahorra recursos lógicos; el circuito de la figura 47, por ejemplo, sólo debe ser usado si es necesario sincronizar una entrada asíncrona con pulsos cuya duración sea menor que un periodo de reloj, y nunca para resolver ninguna otra funcionalidad que pueda ser necesario incorporar al sistema síncrono -este circuito conforma la duración del pulso de entrada, de modo que, independientemente de cuál sea ésta, la duración del puso de salida es siempre un periodo de reloj, pero no se puede utilizar como un conformador de señales síncronas.

Centrándonos en cómo funciona el circuito de la figura 47, debemos notar que garantiza que la salida del primer *flip-flop* –que es aún una señal asíncrona, pues puede ponerse a nivel alto en cualquier instante- va a estar activa, tras un pulso en la señal asíncrona, cuando llegue el flanco de reloj al segundo *flip-flop* (que es el que propiamente sincroniza la entrada), pero no permite distinguir la ocurrencia de más de un pulso en la entrada durante un ciclo de reloj.

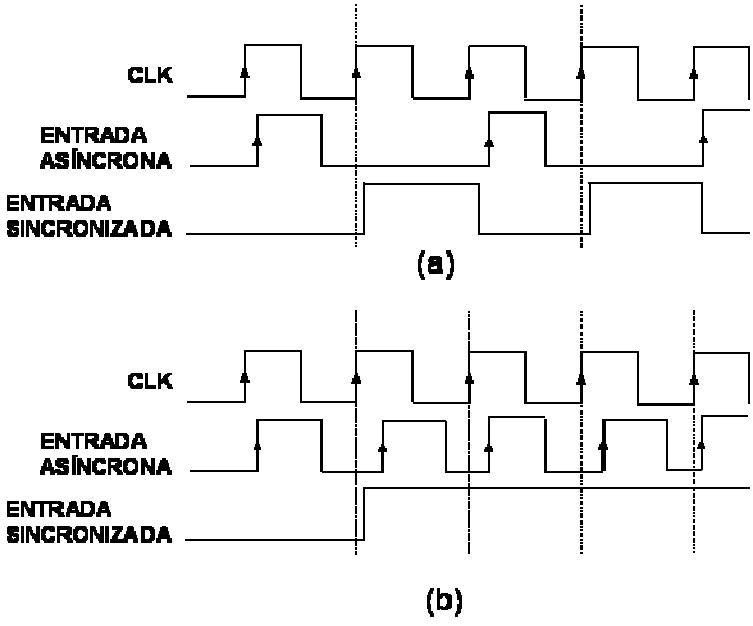

Figura 48. Funcionamiento de la interfaz asíncrona

Si se desea que cada pulso de entrada se corresponda con otro, síncrono y de duración igual a un ciclo de reloj, la frecuencia de la entrada asíncrona ha de ser menor o igual a la mitad de la frecuencia del reloj del sistema síncrono (Figura 48a); si basta con que se manifieste con un nivel alto en el ciclo de reloj inmediato a la ocurrencia del pulso (figura 48b), la frecuencia de la señal de entrada puede llegar hasta la del reloj –aunque en este caso el aspecto de la señal sincronizada puede ser un pulso a nivel alto que dure varios ciclos de reloj (tantos como pulsos de la entrada asíncrona haya en una serie continua de ciclos de reloj).

El último problema derivado de la sincronización es que existe el riesgo de que los cambios de estado de las entradas violen los tiempos de *set-up* o *hold* del *flip-flop* de sincronización. Este *flip-flop* está controlado por el reloj del circuito síncrono, y las entradas asíncronas pueden conmutar en cualquier instante de tiempo, luego existe cierta probabilidad de que el cambio de nivel ocurra en la ventana de tiempo delimitada por el tiempo de *set-up* y *hold* en torno al flanco activo de reloj (figura 49).

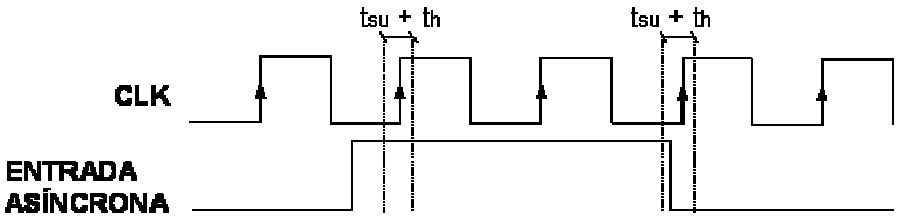

Figura 49. Sincronización (1)

La probabilidad de que no se respeten los tiempos del *flip-flop* es directamente proporcional a la frecuencia de reloj del circuito –la ventana de tiempos a respetar es más grande respecto al periodo de reloj cuanto menor sea éste y, en consecuencia, es más probable que el instante de conmutación de la entrada se produzca dentro de ella- y a la frecuencia de conmutación de la entrada asíncrona –lógicamente, cuantas más veces conmute la entrada, más posibilidades habrá de que alguna vez lo haga en la ventana de tiempos. Cuando no se respetan los tiempos del *flip-flop* podemos encontrarnos con dos respuestas diferentes. La primera y más frecuente es que el *flip-flop* establezca en su salida, dentro de su tiempo de propagación, un nivel lógico estable que se corresponda con el de conmutación de la entrada o el anterior a ésta (figura 50).

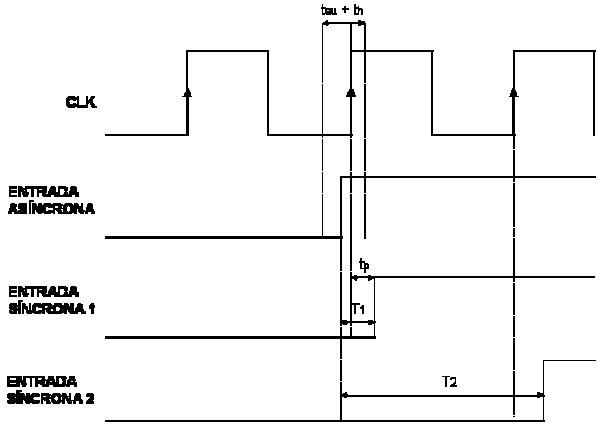

Figura 50. Sincronización (2)

Ninguno de los casos ilustrados en la figura 50 resulta demasiado problemático, ya que en ambos se detecta, aunque con diferente retardo, el cambio de estado. La situación alternativa, más infrecuente y grave, es que el *flip-flop* entre en un estado de metaestabilidad. En metaestabilidad la tensión de la salida del *flip-flop* se establece en una zona intermedia a la correspondiente a los niveles lógicos válidos (cero y uno), dando lugar a una indeterminación lógica (no es ni un '0' ni un '1') y ocasionando un aumento sustancial en el consumo de potencia del *flip-flop*.

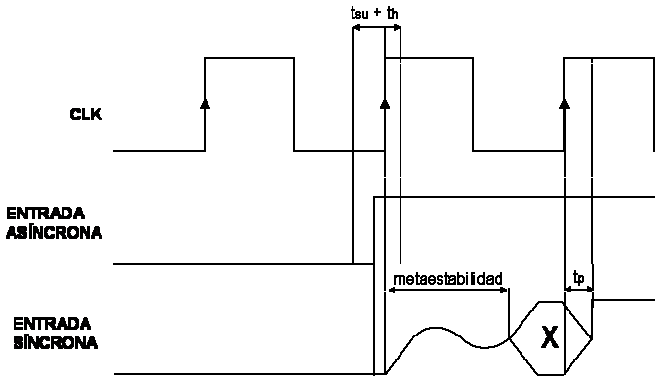

Figura 51

Cuando la salida de un *flip-flop* alcanza un estado metaestable termina, al cabo de un tiempo –denominado tiempo de recuperación, t_R, indeterminado pero susceptible de ser caracterizado por una función de distribución de probabilidad-, por bascular aleatoriamente (figura 51) a un estado estable ('0' ó '1') –en caso de que esto no ocurra, pasará a '0' ó '1' cuando llegue un flanco activo de reloj con las entradas síncronas estables.

La metaestabilidad puede propagarse a elementos del circuito conectados a la salida del *flip-flop* de sincronización. Si se trata de lógica combinacional, el "contagio" es inmediato, en el caso de que se trate de entradas síncronas de *flip-flops*, se produce si llega un flanco activo de reloj. En ambos casos los efectos perniciosos de la metaestabilidad se extienden por el circuito, pudiendo producir errores funcionales o, incluso, dar lugar a un deterioro del *hardware* por el calentamiento provocado por el exceso de consumo.

Desde un punto de vista práctico, la entrada de un *flip-flop* en estado metaestable se manifiesta como un incremento del tiempo de propagación síncrono del *flip-flop* –es decir, del tiempo de propagación desde la entrada de reloj a la salida. Este incremento, denominado tiempo de recuperación o establecimiento (*recovery* o *settling time*), tiene un valor aleatorio que en la mayoría de los casos es muy pequeño, aunque, eventualmente, puede tomar valores que produzcan fallos en el circuito –para que se produzca un fallo es necesario que el estado metaestable se prolongue durante un ciclo completo de reloj. La probabilidad de que el *flip-flop* entre en metaestabilidad, y una vez en ella cause un fallo, debe ser tan baja como exijan los requisitos del circuito que se está desarrollando –evidentemente no tiene la misma importancia que fallen, por ejemplo, circuitos digitales en un avión o que lo hagan los del mando a distancia de un televisor. En relación con este problema, lo primero que hay que tener claro es que aunque resulta imposible garantizar absolutamente la inmunidad de un circuito frente al

problema de la metaestabilidad, sí es posible reducir la probabilidad de fallo a niveles despreciables. Los factores de los que depende la probabilidad de fallo son:

1. La frecuencia del reloj del circuito y la frecuencia media de conmutación de la entrada asíncrona –cómo ya se justificó anteriormente.

2. La tecnología empleada en la realización del sistema y las condiciones ambientales de funcionamiento.

Los fabricantes de chips para la realización de sistemas digitales cableados ofrecen gráficas, tablas y fórmulas que permiten calcular el tiempo medio entre fallos (MTBF) de un circuito en función del tiempo de asentamiento que tolere. La figura 52, por ejemplo, es una gráfica de ALTERA que caracteriza el tiempo medio entre fallos para dos de sus familias de PLDs y para diferentes frecuencias de reloj. De acuerdo con la gráfica, en un circuito síncrono con una frecuencia de reloj de 40 MHz, en el que se pueda tolerar, cuando surja un problema de metaestabilidad, un incremento en el tiempo de propagación del *flip-flop* de sincronización de 4 ns, el tiempo medio entre fallos (MTBF) es de 100 años.

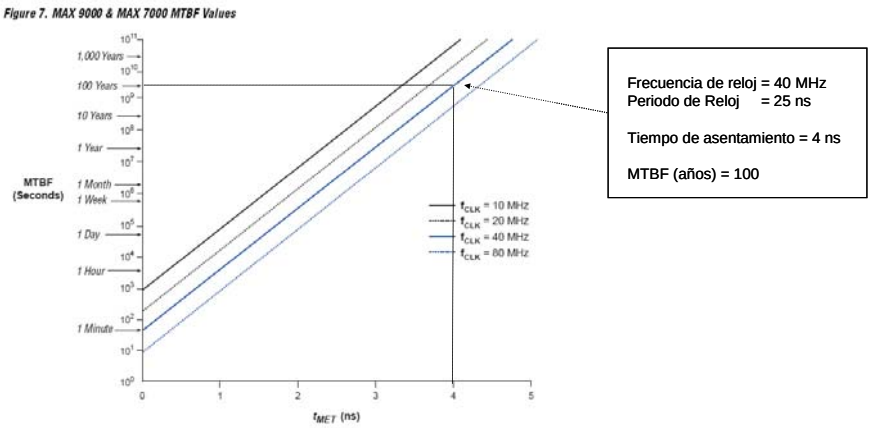

Figura 52. Grafica del MTBF de ALTERA

La fórmula y la tabla de la figura 53 nos permiten realizar un cálculo del MTBF para diferentes familias de dispositivos lógicos programables de ALTERA –la mayor parte de los fabricantes dan una información análoga a la que se muestra aquí.

$$\text{MTBF} = \frac{e^{(C_2 \times t_{MET})}}{C_1 \times f_{CLOCK} \times f_{DATA}}$$

Table 1. Metastability Equation Constants

Device	C_1	C_2
FLEX 10K	1.01×10^{-13}	1.268×10^{10}
FLEX 8000	1.01×10^{-13}	1.268×10^{10}
FLEX 6000	1.01×10^{-13}	1.268×10^{10}
MAX 9000	2.98×10^{-17}	5.023×10^{9}
MAX 7000	2.98×10^{-17}	5.023×10^{9}

Figura 53. Cálculo del MTBF

En la fórmula, C_1 y C_2 son dos parámetros que se obtienen a partir de medidas del tiempo de recuperación realizadas sobre muestras de los circuitos fabricados por ALTERA. Haciendo uso de ellos se puede estimar el MTBF para unas condiciones de funcionamiento dadas o las condiciones de funcionamiento que deben garantizarse para obtener un determinado valor del MTBF.

Ejemplo

Vamos a calcular el tiempo de establecimiento que debe tener el flip-flop de sincronización de un circuito síncrono que se desea realizar con una FPGA de ALTERA de la familia 10K, y que debe funcionar con un reloj de 20 MHz, si la frecuencia media de la entrada asíncrona es de 10 MHz y se desea conseguir un MTBF de 500 años (10^{10} segundos).

Despejando de la ecuación de la figura 47, obtenemos:

$$t_{MET} = \frac{\ln(MTBF \times C_1 \times f_{CLK} \times f_{DATA})}{C_2} = \frac{\ln(10^{10} \times 1.01 \times 10^{-13} \times 2 \times 10^7 \times 10^7)}{1.268 \times 10^{10}} \approx 2\ ns$$

Normalmente no es necesario realizar este tipo de cálculos, porque si un MTBF de decenas de años no resulta aceptable, basta con añadir al circuito de sincronización un segundo *flip-flop* (figura 54). Este *flip-flop* extra actúa como barrera de seguridad ante un fallo: cuando el primer *flip-flop* de sincronización permanece en estado metaestable durante un ciclo de reloj y lo propaga al segundo –supongamos, por ejemplo, que una vez cada año de funcionamiento del circuito-, la probabilidad de que el segundo *flip-flop* propague a su vez este suceso es extraordinariamente remota –la misma que la anterior, del orden de 1 entre millones (o decenas o cientos de millones) de casos-, consiguiéndose valores enormes de MTBF.

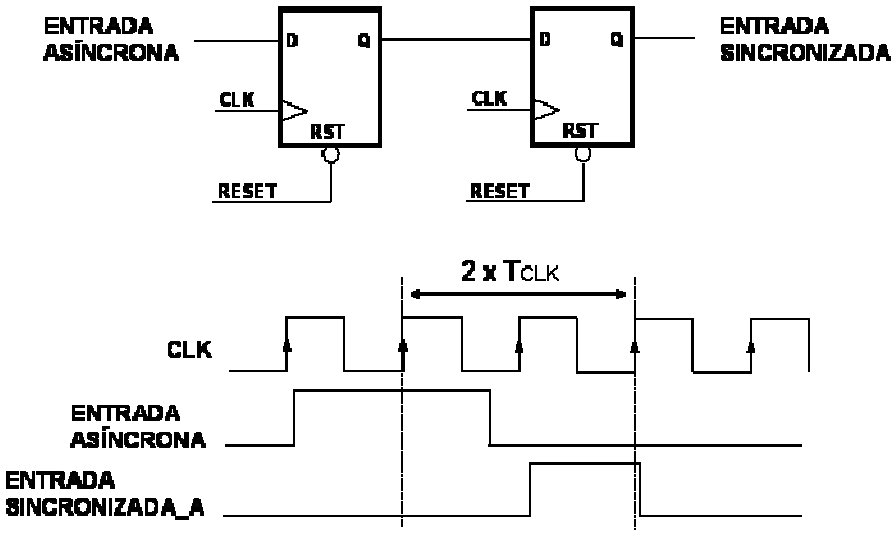

Figura 54. Sincronización con 2 *flip-flops*

El problema que presenta este circuito es que introduce un retardo adicional, de un ciclo de reloj, en la detección del cambio de estado de la entrada asíncrona. En las

aplicaciones en que este retardo no resulta crítico, y en cambio la seguridad es esencial, puede llegar a incluirse un tercer *flip-flop*. El problema se complica si es necesario detectar muy rápidamente los cambios de nivel de la entrada y, al mismo tiempo, se desea tener valores de MTBF altos. Una solución de compromiso para estas situaciones es utilizar en el circuito de sincronización un reloj con una frecuencia múltiplo de la del sistema síncrono –se puede generar fácilmente si se dispone de un *clock manager* (un PLL o un DLL)- y en fase con él; alternativamente se pueden utilizar flancos alternos de reloj en el *flip-flop* de sincronización y el de barrera. En la figura 55 se ilustra el funcionamiento de estas soluciones. El circuito que funciona con un reloj cuya frecuencia es el doble que la del sistema síncrono, al estar en fase con éste –los DLLs y PLLs son capaces de generar relojes con la misma fase que el de referencia-, sincroniza la señal en el flanco activo del reloj del sistema. El segundo circuito captura la señal asíncrona en el flanco de bajada y la sincroniza en el flanco de subida del reloj. Ambos circuitos consiguen un MTBF mejor que el circuito de sincronización con un solo *flip-flop*, y peor que el que utiliza un *flip-flop* adicional como barrera para la metaestabilidad –porque en estos circuitos los *flip-flops* tienen un tiempo de recuperación menor (aproximadamente la mitad).

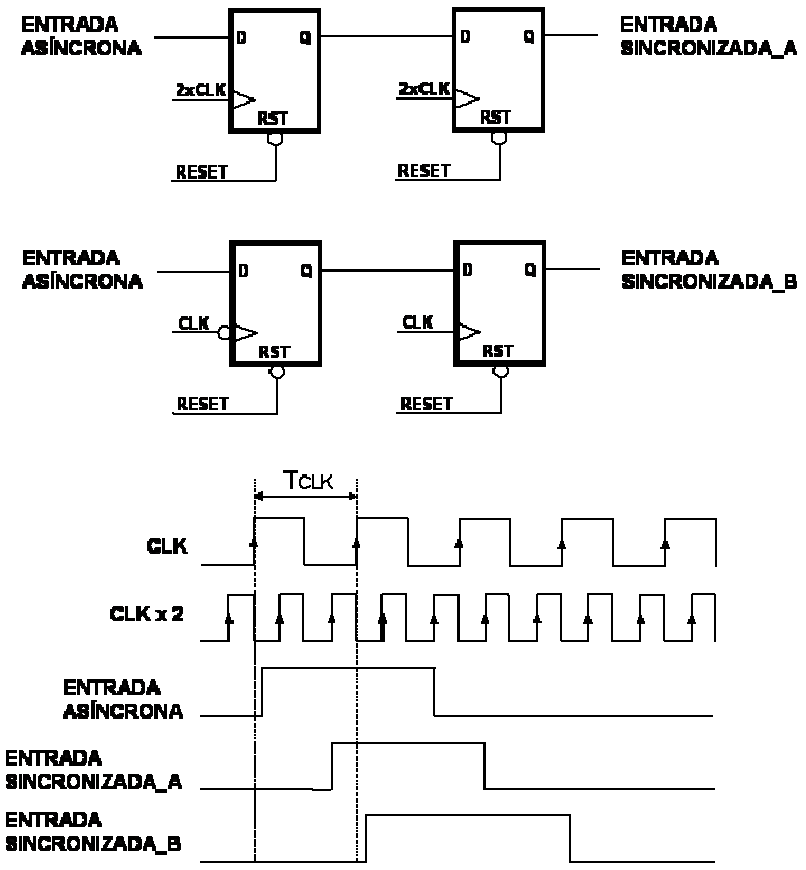

Figura 55. Circuitos de sincronización

Una última opción que permite aumentar el MTBF es la de usar en el circuito de sincronización los *flip-flops* más rápidos de entre todos los disponibles. Hay tecnologías donde no es posible elegir entre diferentes tipos de *flip-flops* (en el diseño con PLDs, por ejemplo), pero hay otras en las que sí (en el diseño de ASICs con *standard cells*) y se puede optar por esta solución –que refuerza, no sustituye, las descritas anteriormente.

El uso de *flip-flops* rápidos se basa en que cuanto menores son los tiempos característicos de un *flip-flop*, menor es también el tiempo de recuperación y la probabilidad de entrar en un estado metaestable.

En ciertas ocasiones resulta necesario acondicionar los pulsos de una entrada asíncrona para que dentro del sistema síncrono su duración sea de un periodo de reloj. Para ello suele acoplarse al flip-flop de sincronización un circuito conformador, tal y como se muestra en la figura EP.50.

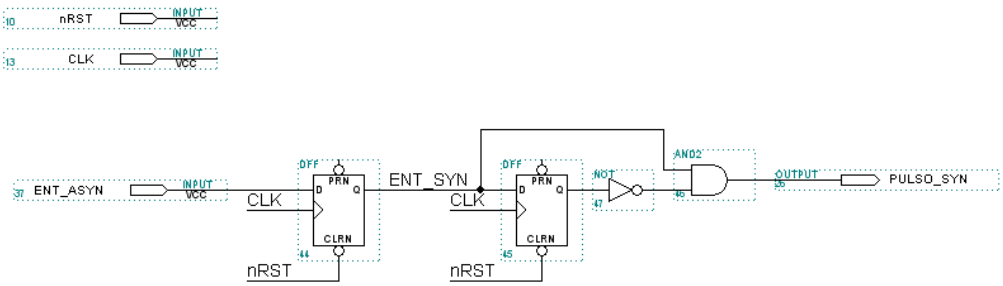

Figura EP.50

Aparentemente, el primer flip-flop sincroniza la entrada y el segundo, junto con las dos puertas, transforma los pulsos positivos de la señal sincronizada para que su duración sea de un solo ciclo de reloj.

En realidad, se trata de un diseño erróneo que puede causar problemas si se emplea en un circuito en el que exista la necesidad de obtener un tiempo medio entre fallos (MTBF) por metaestabilidad elevado. El error radica en que se conecta lógica combinacional a la salida del flip-flop de sincronización. Esta lógica propagaría, con toda seguridad, el estado metaestable en el caso de que el flip-flop de sincronización entre en él, reduciendo el tiempo de asentamiento del circuito –y por tanto el MTBF- en una cantidad igual a su tiempo de propagación.

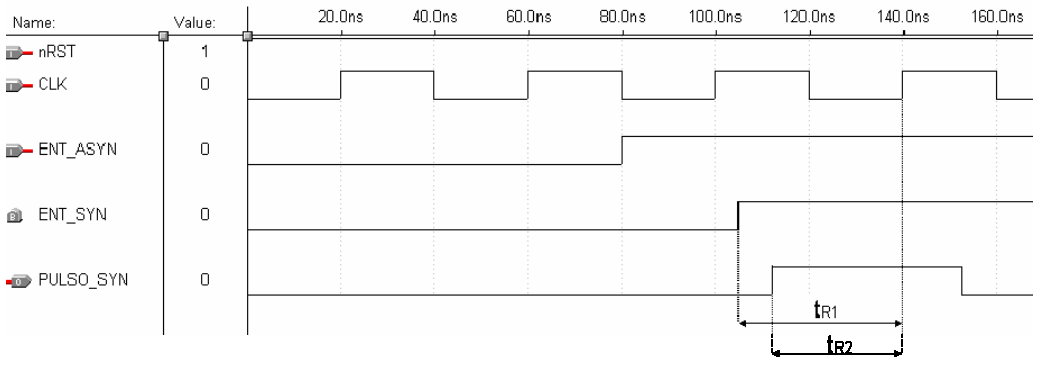

Figura EP.51

En el cronograma de la figura EP.51 puede observarse como se reduce el tiempo de recuperación del pulso conformado (t_{R2}) en relación con el de la salida del flip-flop de sincronización (t_{R1}).

En general, salvo que el valor del MTBF resulte de poca importancia o el periodo de la señal de reloj sea muy grande, no debe conectarse nunca lógica combinacional a la salida de un flip-flop de sincronización. El circuito del ejemplo podría modificarse añadiendo un flip-flop al circuito de sincronización (figura EP.52).

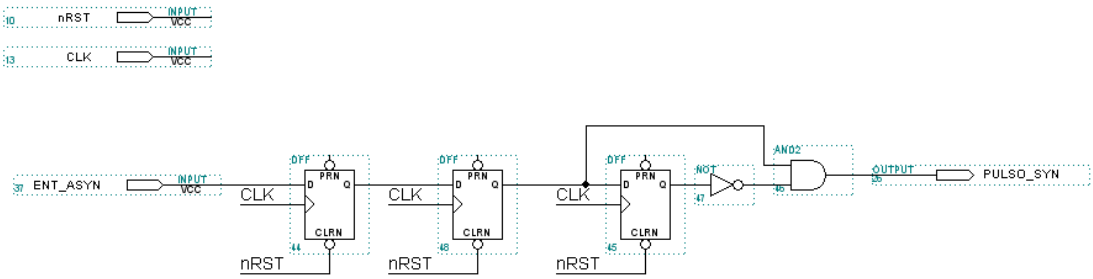

Figura EP.52

EL funcionamiento sería análogo al del circuito anterior, aunque con un MTBF mucho mayor y un periodo de reloj de retardo para la generación del pulso conformado.

3.1.4 Señales globales

Las señales globales de un circuito síncrono son la señal de reloj y la señal de inicialización (*reset*) del circuito. Desde el punto de vista del diseñador estas señales plantean dos problemas: tienen un *fan-out* muy alto y es necesario distribuirlas minimizando el *skew*.

Hay tecnologías, como la lógica programable, donde ambos problemas vienen resueltos en los propios dispositivos donde se realiza el circuito, porque los *chips* disponen de entradas especiales (entradas dedicadas) para ambas señales –en las FPGAs con una alta densidad de integración se suele disponer de varias entradas globales de reloj y *reset*, y de un buen número de redes de distribución de señales globales previendo la posibilidad de que en el mismo *chip* se integren varios circuitos síncronos independientes-, de modo que el problema se reduce a utilizar estos *pines* como puntos de conexión para las señales globales.

En otras tecnologías el diseño de la red de distribución del reloj (en jerga, el árbol de reloj) y el *reset* es responsabilidad del diseñador. Para resolver el problema derivado del alto *fan-out* de estas señales, se pueden utilizar *super-buffers –buffers* capaces de atacar miles de entradas-, si se está diseñando un ASIC (figura 56a), o, en éste y otros casos, árboles de *buffers* simétricos con retardo equilibrado; este tipo de redes, cuya topología se muestra en las figuras 56b y 56c, tratan de conseguir retardos idénticos desde el punto de conexión de la entrada global hasta cualquier entrada de reloj (o *reset*) de los *flip-flops* del circuito -con el fin de minimizar el *skew*- intercalando el mismo tipo y número de *buffers* en cada rama de la red y distribuyendo equitativamente la carga (las entradas de los *flip-flops)* entre ellas. En la figura 56d se muestra un ejemplo de cómo no debe diseñarse una red de interconexión para señales globales.

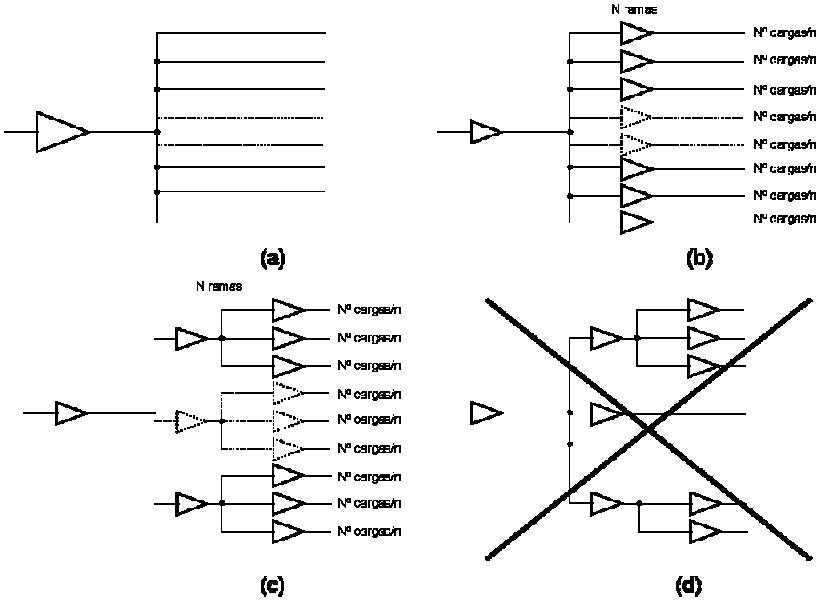

Figura 56. Arboles de reloj

La minimización de *skew* en la interconexión de señales globales puede afinarse durante el diseño físico del circuito (partiendo de un árbol de reloj equilibrado), tratando de definir una estructura que dé lugar a pistas de interconexión con una dispersión de longitudes pequeña –el retardo de propagación asociado a una pista depende en gran medida de su anchura y longitud (también de otros efectos que el diseñador no puede controlar). Normalmente las señales globales se conectan mediante pistas más gruesas que el resto de las señales (esto reduce el retardo de interconexión por unidad de longitud) y se rutan antes que el resto de conexiones del circuito para equilibrar las longitudes (con el fin de que el retardo de todas las pistas sea parejo).

Siendo todo lo dicho un objetivo de diseño, es evidente que resulta imposible igualar la longitud de todas las pistas de la red, por lo que siempre habrá un componente de *skew* –muy pequeño y compatible con el perfecto funcionamiento síncrono del circuito, si se siguen los criterios de diseño indicados- asociado al diseño de la red de interconexión.

Reset

Para terminar con la revisión de cuestiones relativas a las señales globales que son de interés para el diseño de un circuito síncrono, es necesario comentar un par de características de la señal de *reset*: su duración y su sincronización con el reloj del sistema.

La señal de *reset* global debe activarse durante un tiempo mayor o igual al periodo mínimo de la señal de reloj. Sólo si se cumple esta condición se podrá garantizar que en el primer flanco activo de reloj, tras su desactivación, las señales del circuito tengan niveles correctos y estables: si el *reset* global actúa asíncronamente, cuando se mantiene activo durante un tiempo que permite la estabilización de las salidas de los circuitos combinacionales (el periodo mínimo del reloj del circuito), se puede

pasar al modo de funcionamiento normal (síncrono, tras su desactivación) del circuito con seguridad; si actúa síncronamente deberá estar activo, al menos, en un flanco de reloj y podrá desactivarse a partir del siguiente flanco.

La actuación del *reset* global sobre los *flip-flops* del circuito plantea un problema relacionado con los tiempos característicos de éstos: si el *reset* opera asíncronamente, las entradas de inicialización asíncronas de un *flip-flop* tienen que desactivarse un tiempo antes de la ocurrencia de un flanco activo de reloj –este tiempo característico de los *flip-flops* con entradas de inicialización asíncrona se conoce como *tiempo de recuperación*- para garantizar que el *flip-flop* va a funcionar correctamente (en caso contrario, puede pasar a un estado metaestable); si actúa síncronamente, tiene que cumplir los tiempos de *set-up* y *hold* del *flip-flop* o, sino, puede dar lugar también a problemas de metaestabilidad.

Hay dos formas de abordar este problema: la primera es correr el riesgo de que no se respeten los tiempos, ya que al ser el *reset* global una señal con una frecuencia media de conmutación extremadamente baja, si opera sobre una red de interconexión de bajo *skew*, tendrá un tiempo medio entre fallos (MTBF) muy alto; la segunda alternativa es registrar el *reset* global con un *flip-flop* manejado por el reloj del circuito antes de conectarlo a la red de interconexión global para que sus conmutaciones respeten los tiempos característicos de los *flip-flops* del sistema síncrono. La primera opción es aceptable en sistemas sin fuertes requisitos de seguridad, la segunda es la indicada cuando no se puede correr el riesgo de que ocurra un fallo en la actuación del *reset*.

3.2 Técnicas para alcanzar los requisitos de velocidad de funcionamiento en un circuito síncrono.

Como ya sabemos, la frecuencia máxima de reloj con que puede funcionar un circuito síncrono está determinada por la ruta de procesamiento combinacional más lenta y su valor es, aproximadamente, el valor inverso de su retardo. De nada vale, por tanto, que la mayor parte de la lógica de un sistema síncrono sea muy rápida si existe algún módulo, o cadena de módulos combinacionales, con un retardo muy grande: será este elemento del circuito el que limitará la velocidad de todo el conjunto al tener que funcionar todo el sistema al ritmo marcado por el reloj global.

Cuando una determinada unidad de procesamiento combinacional impide conseguir la frecuencia de funcionamiento necesaria para satisfacer los requerimientos de velocidad de un circuito, se pueden tomar distintos tipos de medidas:

1. En primer lugar, se puede intentar disminuir los retardos de la lógica combinacional rehaciendo la síntesis, colocación e interconexión de los elementos del circuito. Esta solución es la que requiere menos esfuerzo por parte del diseñador y en muchos casos es suficiente para conseguir que el sistema pueda alcanzar la frecuencia de funcionamiento deseada.

2. Otra de las opciones posibles consiste en sustituir la tecnología de realización del circuito por otra más rápida. Esta solución puede acarrear un notable

incremento en el coste del *hardware* –a veces no, hay que evaluar cada caso en concreto- pero puede evitar esfuerzos adicionales al diseñador.

3. Si se descartan, o resultan insuficientes, las medidas anteriores, hay que rediseñar el circuito. El rediseño puede orientarse a la paralelización o segmentación de operaciones o, alternativamente, a la realización de operaciones multiciclo. La elección de una o varias de entre estas opciones de rediseño –que se analizarán en detalle posteriormente- dependerá de las características específicas de cada diseño.

En los siguientes apartados se tratará, brevemente, el procesamiento paralelo y, más en detalle, por ser técnicas de diseño muy utilizadas en el desarrollo de sistemas síncronos, la segmentación y la realización de operaciones multiciclo.

Ejemplo

El circuito de la figura EP.53 realiza operaciones de multiplicación y acumulación sobre datos de 32 bits y debe ser capaz de realizar 15 millones de operaciones por segundo.

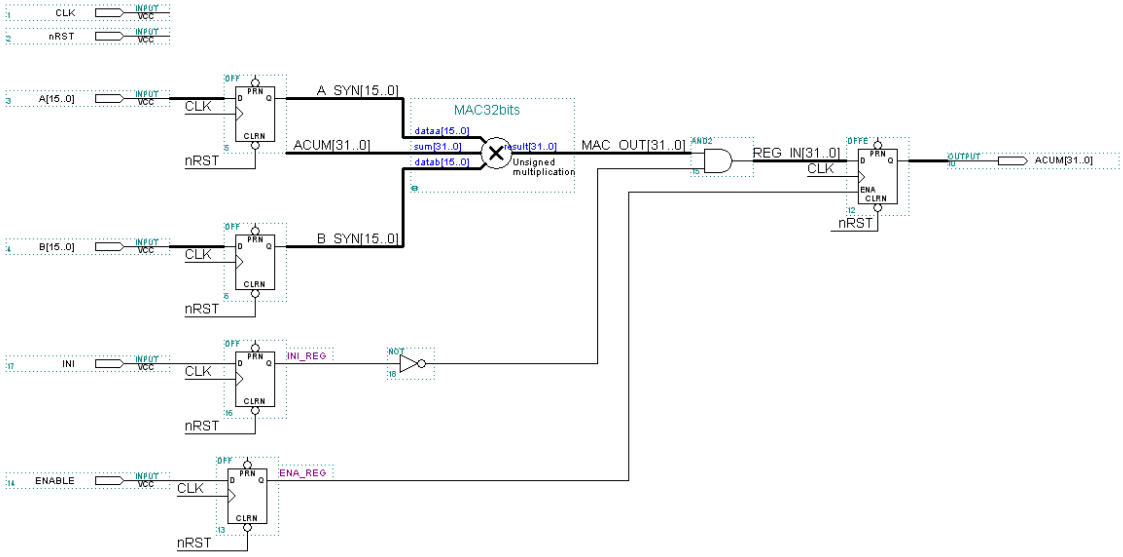

Figura EP.53

En el cronograma de la figura EP.54 se muestra una simulación del funcionamiento del circuito.

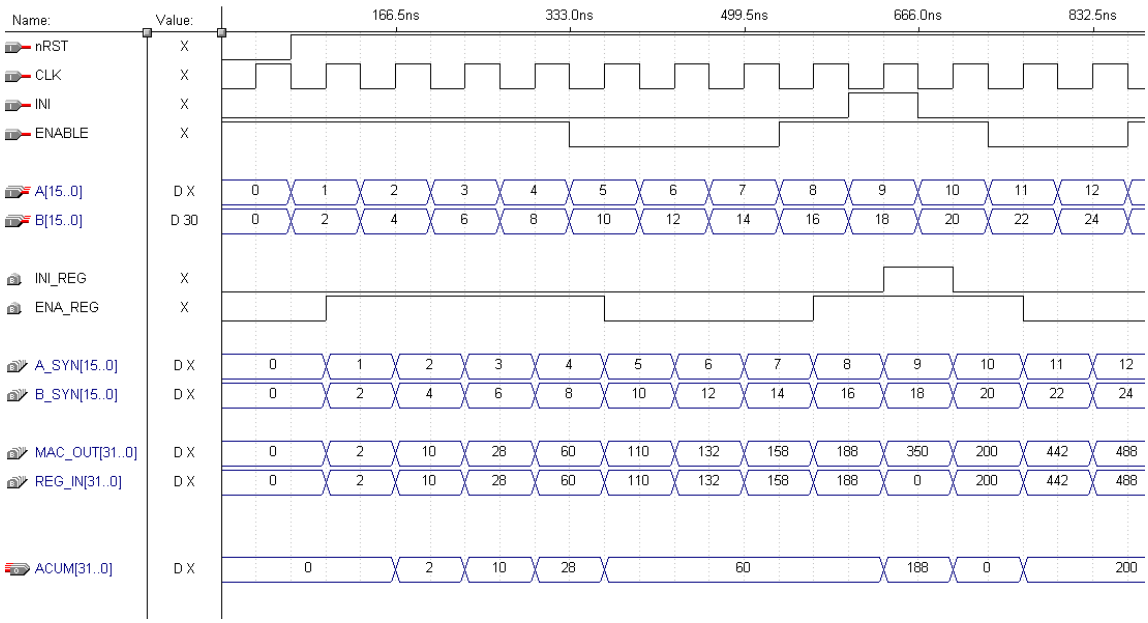

Figura EP.54

El circuito se desea realizar sobre una FPGA EPF10K30RC208-4 de ALTERA. Tras una primera compilación, se realiza un análisis de tiempos en el que se obtiene una frecuencia máxima de 7'43 MHz, aproximadamente la mitad de la frecuencia máxima necesaria para cumplir los requisitos del circuito.

El compilador del entorno MAX+plus II dispone de opciones de configuración que permiten guiar el proceso de síntesis y colocación de la lógica en las FPGAS para conseguir diseños más rápidos. La opción ***Global Project Logic Synthesis****, del menú* ***Assign****, nos permite configurar la síntesis lógica para realizar una versión "rápida" (Figura EP.55) del circuito (estilo de síntesis* ***Fast****).*

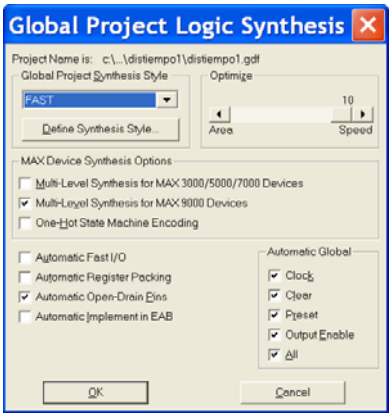

Figura EP.55

Tras elegir esta opción y recompilar, se obtiene, en un análisis de tiempos, que la frecuencia máxima de operación de la nueva realización del circuito es de 13'88 MHz, insuficiente aún para cumplir los requisitos de velocidad del circuito. Se puede continuar, ahora, intentado manipular la configuración de la herramienta de CAD para intentar aumentar la velocidad del circuito –en este caso, en el entorno MAX+plus II, no es de esperar que pueda mejorarse el último resultado obtenido- o, alternativamente, puede optarse por cambiar de FPGA; la alternativa más simple y

prometedora es utilizar una versión más rápida de la pieza elegida: la EPF10K30RC208-3.

Una vez realizado el cambio y compilado el diseño, el análisis de tiempos indica que la frecuencia máxima de operación es de 17'73 MHz. Esta velocidad resulta suficiente, pero se obtiene a cambio de incrementar el coste del chip en el que se realiza el circuito (el aumento de precio por utilizar una pieza más rápida puede ser muy grande cuando se trata de PLDs o FPGAs). Una alternativa más interesante, una vez tomada la decisión de cambiar de FPGA, para obtener la frecuencia de funcionamiento deseada consiste en, si es posible, utilizar un dispositivo más pequeño, es decir, con menor número de recursos lógicos –es un buen "truco" cuando el circuito se realiza con FPGAs, pero sólo puede recurrirse a él cuando el tamaño del chip elegido inicialmente está sobredimensionado. Si se migra el circuito a una EPF10K20RC208-4 (una FPGA con un pinout compatible con las anteriores), se consigue una frecuencia de operación de 15'4 MHz, con un chip cuyo precio es menor que el elegido originalmente.

3.2.1 Paralelismo

El diseño de arquitecturas con ejecución de operaciones en paralelo persigue el aumento de la potencia de procesamiento de un circuito mediante la replicación de la lógica operativa. El fundamento de esta estrategia de diseño es simple: si a una determinada frecuencia de reloj la lógica de un circuito es capaz de realizar N operaciones por segundo, construyendo un sistema en el que existan M replicas de dicha lógica, y utilizando la misma frecuencia de reloj, se podrán realizar M x N operaciones por segundo. Evidentemente el aumento de las prestaciones del circuito se consigue a cambio de un aumento en los recursos lógicos necesarios para realizarlo: los correspondientes a la lógica duplicada y a los elementos que tienen que controlar la arquitectura paralela y el flujo de datos en el sistema.

Por ejemplo, la potencia computacional del circuito acumulador de multiplicaciones de la figura EP.53 puede aumentarse replicando la lógica aritmética del circuito, tal y como se muestra en la figura EP.56.

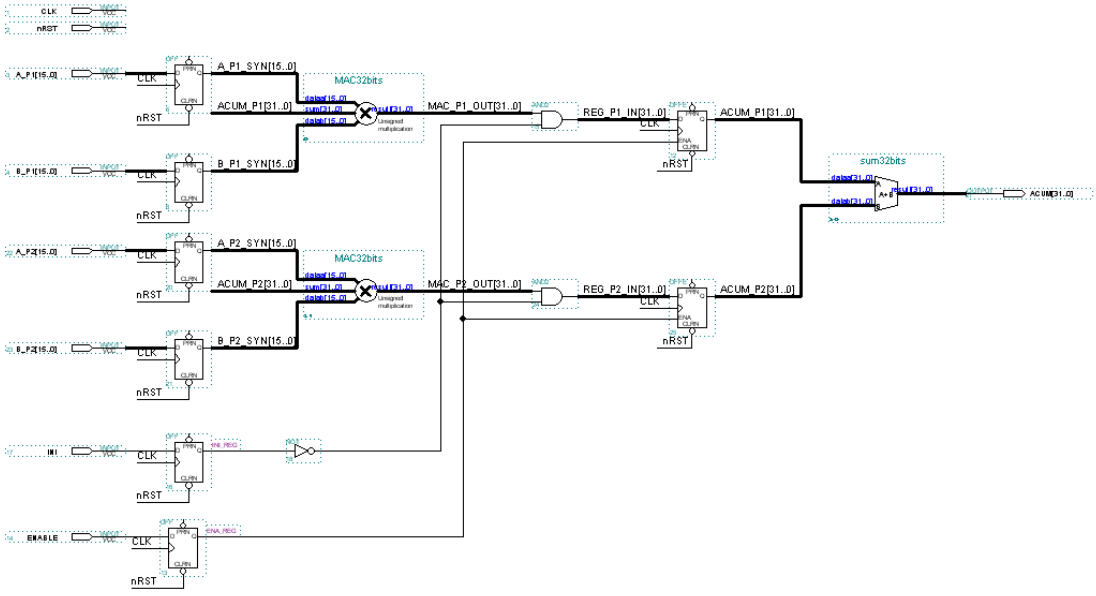

Figura EP.56

Al circuito original se ha añadido una nueva unidad de acumulación de multiplicaciones y un sumador que totaliza el valor acumulado en los dos acumuladores. Observe que esta solución implica añadir dos nuevos puertos de entrada de datos.

En el siguiente cronograma (figura EP.57) se muestra una simulación del funcionamiento del circuito, donde puede observarse que en cada ciclo de reloj se realizan dos productos y acumulaciones.

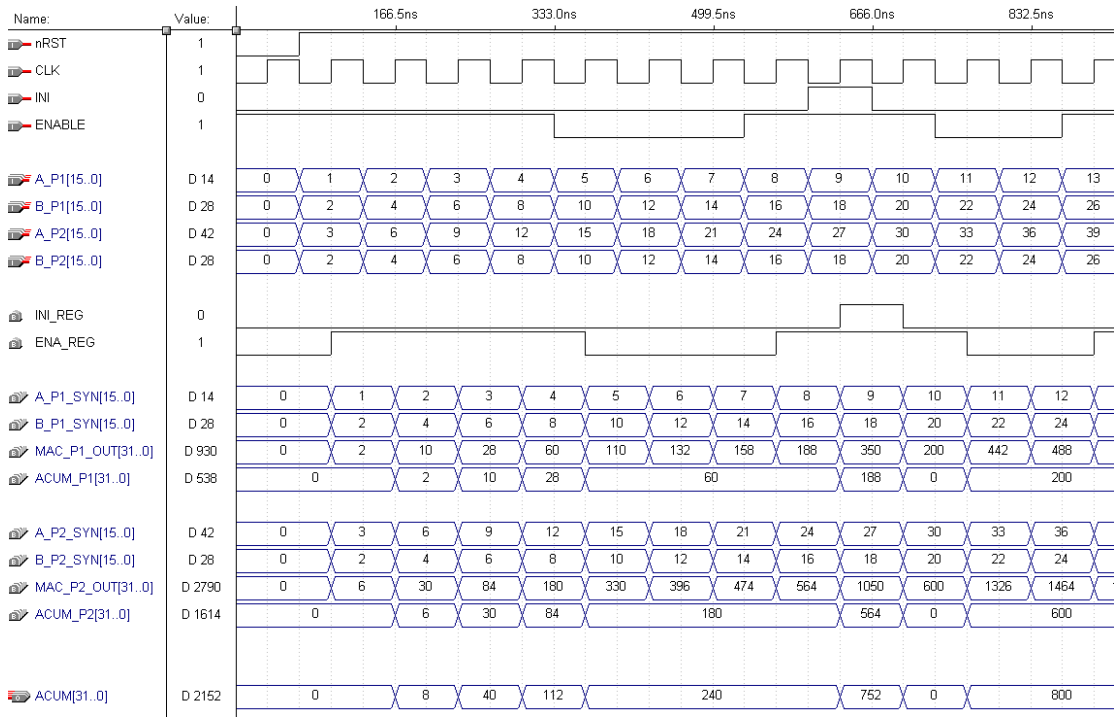

Figura EP.57

Este circuito ocupa más del doble de recursos lógicos que el original y es, también, más lento –su frecuencia máxima de funcionamiento, utilizando la misma FPGA para realizarlo, es menor-, por lo que su potencia de cálculo no alcanza a duplicar la de aquel. Su interfaz de entrada debería, además, simplificarse; lo normal sería utilizar un único puerto para la entrada de datos (estos deberían, por tanto, introducirse a una frecuencia que doble la del circuito para mantener ocupados los operadores aritméticos), lo que implica añadir lógica de control de la interfaz, haciendo más complejo y costoso el circuito: la gestión de los flujos de datos es uno de los mayores problemas (de diseño) que acarrea la realización de arquitecturas paralelas, aún cuando el grado de paralelismo, como en el ejemplo anterior, no sea muy alto.

En resumen, la realización de arquitecturas paralelas se basa en la replica de operadores que ejecutan concurrentemente la misma operación; en condiciones óptimas de operación puede obtenerse un aumento de prestaciones proporcional al número de operadores utilizados. El inconveniente básico que presenta esta técnica es su alto coste *hardware*. Hay que dejar constancia, por último, de que la realización de arquitecturas paralelas no es propiamente un recurso para resolver problemas de velocidad, sino, más bien, una técnica para la realización de arquitecturas con gran potencia de

procesamiento y un óptimo aprovechamiento del ancho de banda de las interfaces de entrada.

3.2.2 Segmentación

Cuando resulta preciso acelerar la velocidad de procesamiento de un sistema síncrono es frecuente recurrir a técnicas de segmentación. El término segmentación (*pipelining*) hace referencia a la división de las rutas de procesamiento combinacional con tiempos de retardo elevados para aumentar la frecuencia de operación del sistema. De la forma en que se mueven los datos en este tipo de estructuras, como un fluido por una tubería, se deriva su denominación original inglesa: *pipelining*. No se pretende aquí tratar en profundidad la problemática y metodología del diseño segmentado, pues escapa al alcance y propósito de este texto, pero sí se van a describir los fundamentos de esta técnica de diseño así como sus ventajas e inconvenientes, con la finalidad de que se disponga de los suficientes conocimientos como para aplicarla para resolver problemas sencillos.

En la figura 57 se muestra la estructura típica de un circuito interno de un sistema síncrono; como puede observarse, consta de lógica combinacional cuyas entradas y salidas se encuentran registradas.

Figura 57

Como ya se sabe, en un circuito síncrono la frecuencia máxima de funcionamiento viene dada por la expresión*:*

$$f_{clk\,\max} = \frac{1}{t_{pff\ \max} + t_{pLC\ \max} + t_{su}}$$

Donde t_{pff} y t_{su} son los tiempos de propagación y de *set-up* de los *flip-flops*, y dependen de las características de la tecnología con que se realice el circuito, y $\mathbf{t_{pLC}}$ es el mayor tiempo de propagación combinacional existente en el circuito; la función combinacional más lenta es, por tanto, la que define la máxima frecuencia de operación.

En el diseño de circuitos lógicos, el empleo de técnicas de segmentación para alcanzar una determinada frecuencia de funcionamiento se basa en dividir los caminos combinacionales con un retardo superior al admisible intercalando registros (figura 58). Como contrapartida al aumento de velocidad hay que *pagar* un mayor coste en recursos lógicos y consumo –por los *flip-flops* añadidos- y, normalmente, una mayor complejidad en el control de la lógica del circuito.

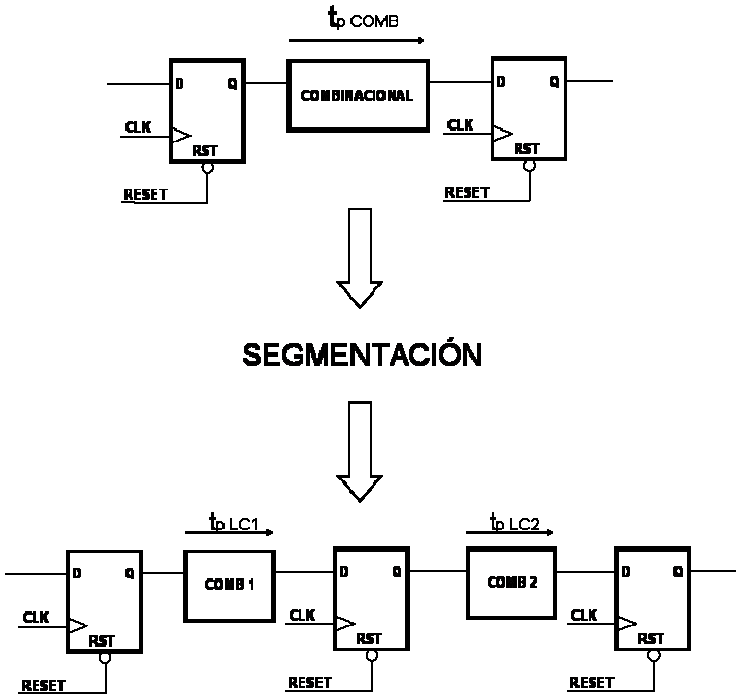

Figura 58. Segmentación

En la figura 58 se ilustra el fundamento de la segmentación: consiste en sustituir un bloque de procesamiento combinacional por dos o más bloques, que realizan en conjunto la misma función que el original, intercalando *flip-flops* entre los segmentos combinacionales; los retardos entre registros en el circuito segmentado son menores que en el original y, en consecuencia, su frecuencia de funcionamiento mayor.

El circuito segmentado es más complejo que el original y puesto que su realización requiere un mayor número de *flip-flops* (en general, la partición del bloque combinacional en trozos no tiene porque suponer un mayor coste), sólo debe aplicarse a los elementos del circuito que impiden alcanzar la frecuencia de funcionamiento deseada.

Por ejemplo, el circuito de la figura EP.58 es un sumador de 16 bits, realizado en una FPGA de ALTERA, la EPF10K30RC208-4.

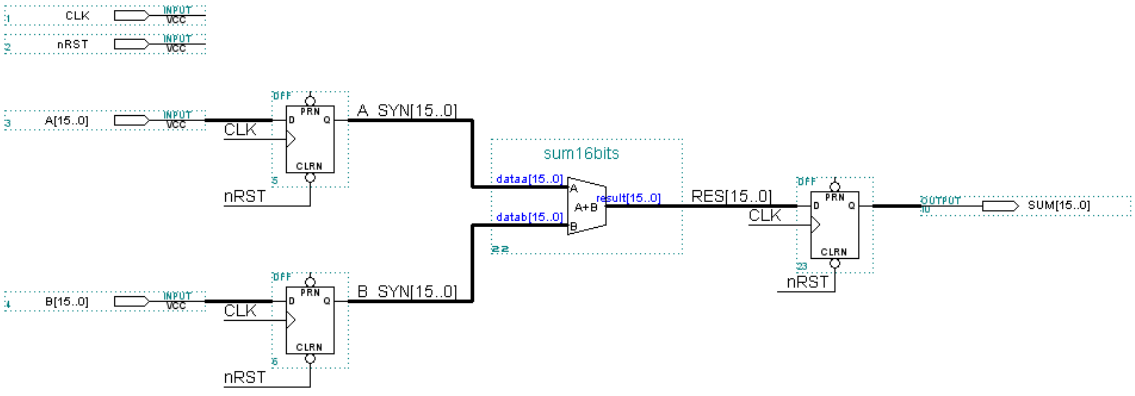

Figura EP.58

Tras realizar una compilación para obtener una realización rápida del circuito, se obtiene, con el analizador de tiempos de MAX+plus II, una frecuencia máxima de funcionamiento de 72'99 MHz (un periodo mínimo de la señal de reloj de 13'7 ns). Supongamos que se desea que el circuito pueda funcionar a 100 MHz. Para conseguir esta frecuencia se puede segmentar el sumador como se muestra en la figura EP.59.

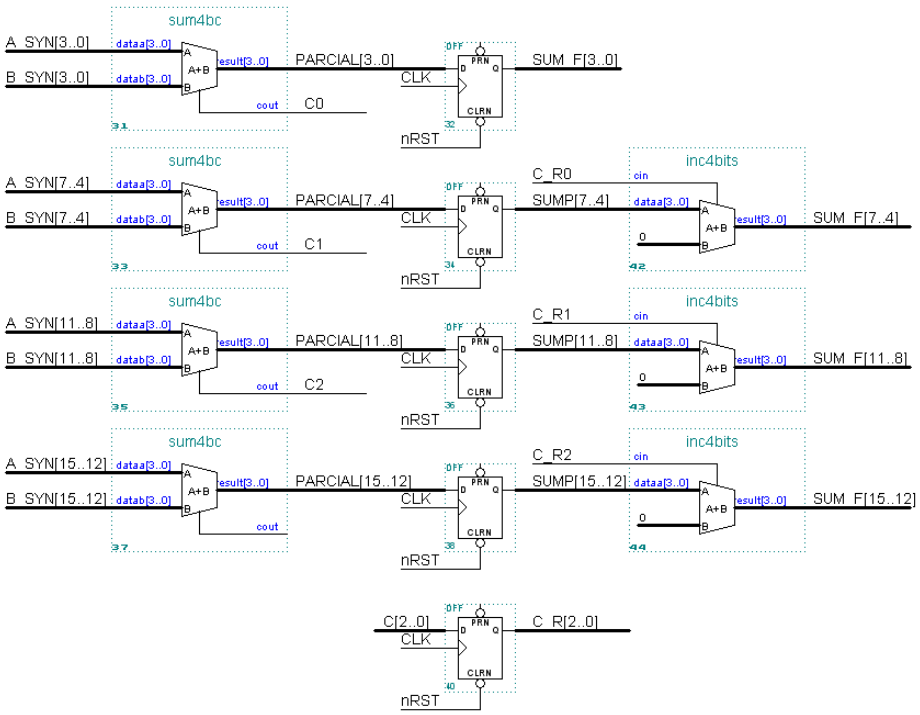

Figura EP.59

La suma se realiza ahora en dos fases: en la primera se suman los datos en grupos de 4 bits, con un retardo mucho menor que el del sumador de 16 bits, y se obtiene el acarreo de cada grupo; en la segunda se corrige el valor de la suma teniendo en cuenta el acarreo registrado. Con esta versión del circuito y una configuración de compilación idéntica a la anterior se obtiene una frecuencia máxima de funcionamiento de 102'4 MHz (un periodo mínimo de reloj de 9'8 ns). En este ejemplo el uso de recursos de la FPGA aumenta en un porcentaje del 48% (en el circuito sin segmentar se usan 48 LEs (Logic Elements), mientras que en el segmentado se usan 67) –cuando los circuitos se realizan en FPGAs el uso de técnicas de segmentación puede resultar "económico" si los bloques que implementan la lógica combinacional incorporan flip-flops que pueden aprovecharse en la segmentación.

El sumador segmentado tarda dos ciclos de reloj en realizar una suma, de modo que, aún pudiendo funcionar a una frecuencia de reloj sustancialmente mayor que el sumador combinacional resulta, aparentemente, más lento. En efecto, si se contempla una operación individual de suma, en el caso de utilizar el sumador combinacional de 16 bits el resultado de una operación aparece en la salida al cabo de 1 ciclo de reloj (descontando el retardo introducido por los registros de sincronización). En el cronograma adjunto (figura EP.60) se muestra una simulación del funcionamiento de este circuito. Utilizando la frecuencia máxima de reloj, el tiempo que se tarda en realizar la operación es de 13'7 ns.

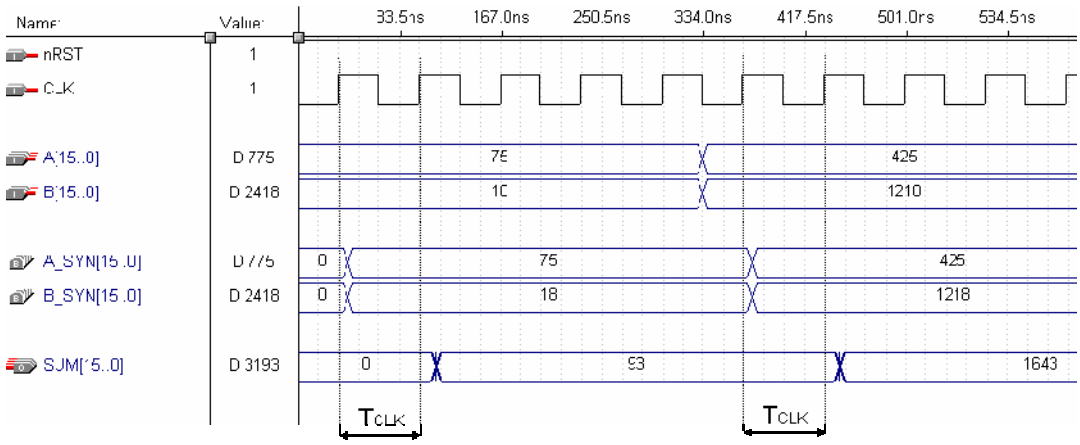

Figura EP.60

En el sumador segmentado, el retardo es de dos ciclos de reloj (figura EP.61) y, por tanto, cada suma tarda en realizarse 2 x 9'8 ns, en total 19'2 ns.

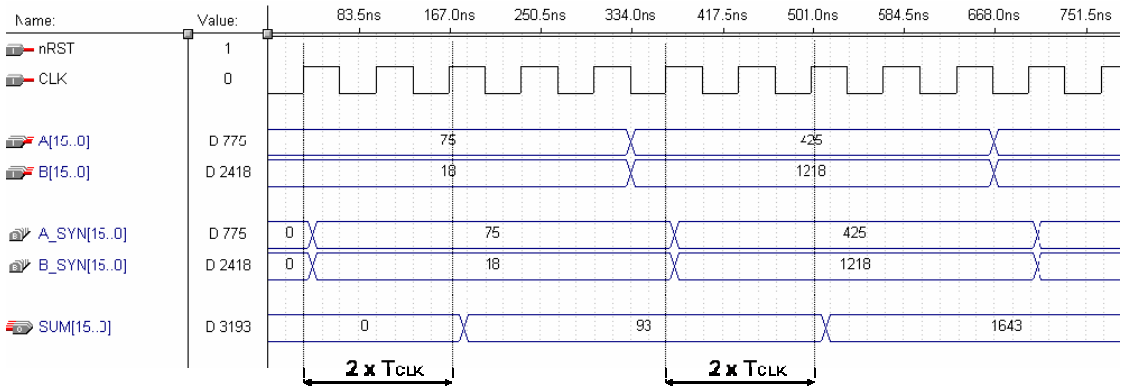

Figura EP.61

En las condiciones mostradas en los cronogramas, para que el circuito segmentado sea igual de rápido que el combinacional es necesario que las partes en que se "trocea" éste tengan el mismo retardo, un objetivo que es prácticamente imposible alcanzar, y que, por tanto, su frecuencia de funcionamiento sea el doble de grande. Pero para valorar correctamente los efectos de la segmentación hay que tener en cuenta dos consideraciones que no hemos incluido en nuestro análisis.

En primer lugar hay que ser conscientes de que la lógica segmentada puede ser sólo una parte del sistema que se está realizando y que, aunque dicha lógica se vea penalizada por la segmentación, el aumento de la frecuencia de reloj del sistema puede beneficiar otras operaciones cuya velocidad sea crítica para cumplir los objetivos de diseño.

En segundo lugar, ocurre que no es cierto que un circuito segmentado tenga que ser más lento, en cuanto a su capacidad de procesamiento en tiempo real, que un circuito sin segmentar, al contrario, si se dan las condiciones de operación apropiadas puede ser mucho más rápido; para demostrarlo vamos a utilizar nuevamente el ejemplo –del que dedujimos lo contrario- del circuito sumador.

Supongamos que a nuestro circuito llegan dos sumandos en cada ciclo de reloj. El sumador combinacional entrega en su salida, con un retardo igual a un periodo de reloj, la suma de los datos de entrada tal y como se muestra en el cronograma de la figura EP.62. Es capaz de realizar por tanto, mientras el flujo de entrada de datos sea continuo, un número de sumas igual a la frecuencia de reloj del circuito; operando con la frecuencia máxima de reloj, 72'99 millones de sumas por segundo.

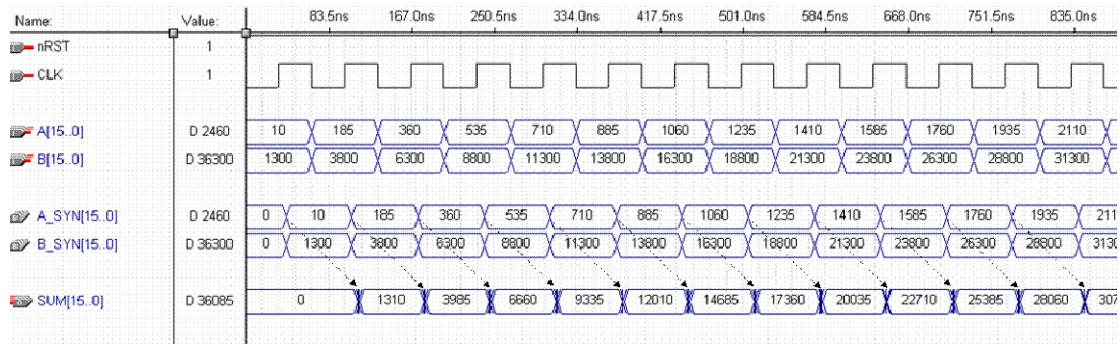

Figura EP.62

En las mismas condiciones, el sumador segmentado entrega también en su salida un resultado en cada ciclo de reloj, aunque con un retardo de dos periodos de reloj respecto a la entrada de los datos. Al igual que en el caso anterior, el número de sumas que realiza por segundo con una entrada continua de datos es igual a la frecuencia de reloj del circuito, como máximo 102'04 millones de sumas por segundo, es decir, 30 millones más que el sumador sin segmentar. En el siguiente cronograma (figura EP.63) se muestra una simulación del funcionamiento de este circuito.

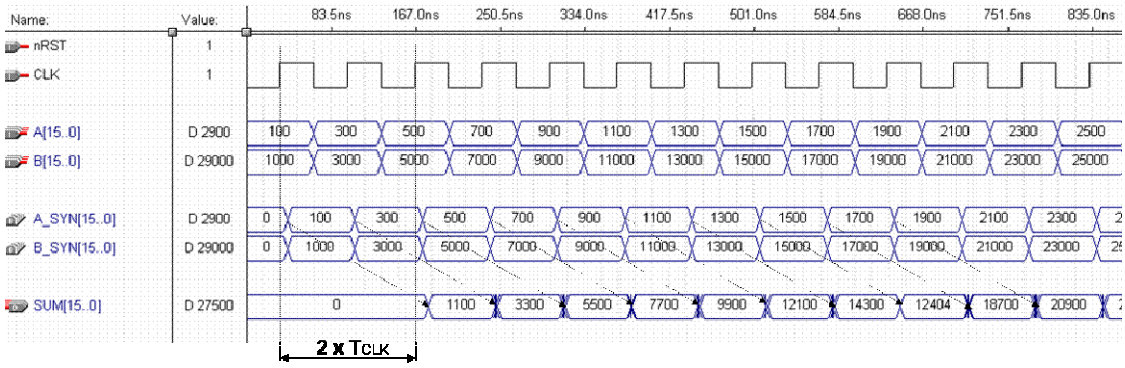

Figura EP.63

El sumador segmentado es más rápido porque, aunque tarda dos ciclos de reloj en entregar el resultado de una operación, está procesando en cada una de sus etapas, simultáneamente y en cada ciclo de reloj, una fase distinta de dos operaciones consecutivas: está, por una parte, realizando el cálculo provisional de la suma de los datos de entrada con los sumadores de 4 bits y, por otra, corrigiendo con el acarreo registrado el resultado de la suma correspondiente a los datos de entrada anteriores.

El ejemplo anterior pone de manifiesto una característica propia de la lógica segmentada: el resultado del procesamiento tarda en ser calculado, y entregado a la salida del circuito, un cierto número de ciclos de reloj, a este retardo se lo conoce como

latencia[7] del *pipeline*. La latencia indica el tiempo que tarda en *llenarse* el *pipeline* para ser capaz de empezar a entregar resultados por su salida, se mide en ciclos de reloj y depende del grado de segmentación del circuito; cuanto mayor sea el número de pedazos en que se trocea el circuito original, y por tanto, también, el número de bloques de *flip-flops* intercalados, mayor será la latencia del circuito –y, normalmente, también la frecuencia a la que pueda funcionar. La latencia hace que el circuito segmentado sea más lento que su equivalente combinacional si el flujo de entrada de datos es irregular, pero cuando el flujo de datos de entrada es continuo su efecto resulta despreciable y permite que se puedan alcanzar velocidades de procesamiento muy altas.

Por ejemplo, si un circuito combinacional realiza una función que puede ser ejecutada a una frecuencia de reloj de 10 MHz, es posible, evidentemente, realizar 10 millones de operaciones por segundo. Segmentándolo en cuatro bloques se obtiene un circuito con una latencia de cuatro ciclos de reloj que, supongamos, puede funcionar a una frecuencia de 30 MHz. En este circuito, si los datos de entrada no se renuevan hasta que no se haya terminado el procesamiento de los anteriores, se pueden realizar como máximo 7'5 (30/4) millones de operaciones por segundo; si hay datos de entrada nuevos en cada ciclo de reloj, el circuito puede realizar 30 millones de operaciones por segundo –siendo estrictos, en el primer segundo de funcionamiento se realizarían 29 millones novecientas noventa y nueve mil novecientas noventa y seis operaciones, porque el primer resultado tardaría en calcularse cuatro ciclos de reloj (la latencia del circuito).

La mejora de velocidad que se obtiene con la segmentación es, esencialmente, proporcional al número de etapas en que se divida la lógica segmentada, pero a la hora de decidir el grado de segmentación que resulta razonable deben tenerse en cuenta varios factores. En primer lugar, hay un límite en el número máximo de etapas en que se puede segmentar un circuito, que viene dado por el hecho, extremo, de que como mínimo habrá un nivel de lógica en cada segmento del *pipeline*. Por otra parte, hay que ser consciente de que el retardo asociado a los *flip*-flops y las pistas de interconexión contrarrestan en parte los beneficios de la segmentación. También hay que valorar que puesto que hay que añadir *flip-flops* en cada segmento, que no aportan de por sí una mejora en la capacidad de procesamiento, resulta conveniente estimar la relación entre la mejora en velocidad que se obtiene y el coste en área que conlleva. Un último factor a tener en cuenta es la relación directa entre el número de etapas y la latencia del *pipeline*, que será, como mínimo[8], igual al número de etapas y, por tanto, puede penalizar seriamente la velocidad de arquitecturas muy segmentadas con flujos irregulares de entrada de datos.

En resumen, la segmentación de la lógica combinacional, permite resolver problemas de velocidad cuando en un sistema hay lógica combinacional compleja, con un retardo de propagación alto pero con una potencia de procesamiento que no resulta crítica, que impide alcanzar la frecuencia de reloj requerida. En estos casos la segmentación debe utilizarse para reducir los retardos de esta lógica aunque pueda

[7] Hay otra acepción del término latencia en el ámbito de las metodologías de diseño de circuitos segmentados que hace referencia al ritmo con el que pueden introducirse los datos de entrada a un *pipeline*.

[8] En este texto, por simplicidad, nos estamos refiriendo exclusivamente a *pipelines* lineales (sin realimentaciones entre etapas); en otro tipo de circuitos segmentados la latencia puede ser mayor que el número de etapas del *pipeline*.

ralentizar su operación. En otros casos la segmentación se utiliza como un recurso de diseño para realizar circuitos muy rápidos, con una alta capacidad de procesamiento, subordinada, siempre, al mantenimiento de flujos constantes y continuos de entrada de datos.

En el entorno MAX+plus II es posible generar módulos funcionales segmentados, especificando, mediante un valor de latencia, el número de etapas de segmentación (figura EP.64). Haciendo uso de esta utilidad se puede, de un modo muy simple, realizar una versión segmentada del circuito sumador con una latencia de, por ejemplo, 1 ciclo de reloj.

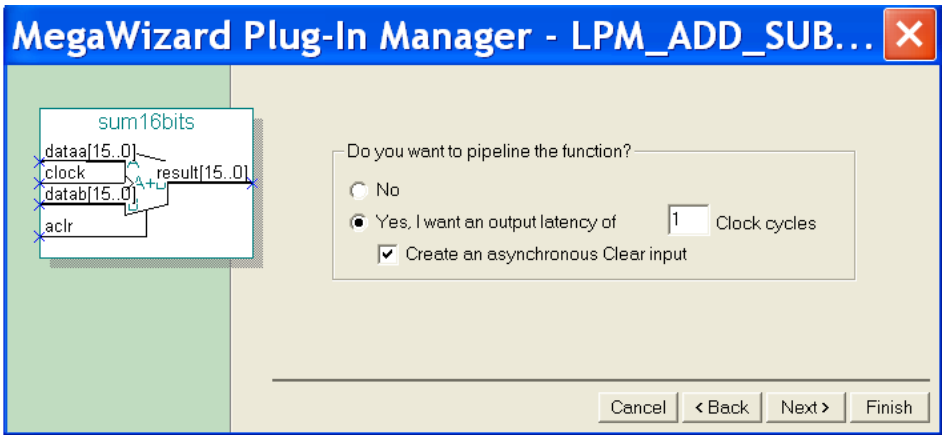

Figura EP.64

Con esta versión del sumador se puede construir el circuito de la figura EP.65, en el que puede observarse que el símbolo del sumador tiene una entrada de reloj y otra de reset asíncrono, puesto que está construido con dos bloques de lógica combinacional separados por una barrera de flip-flops.

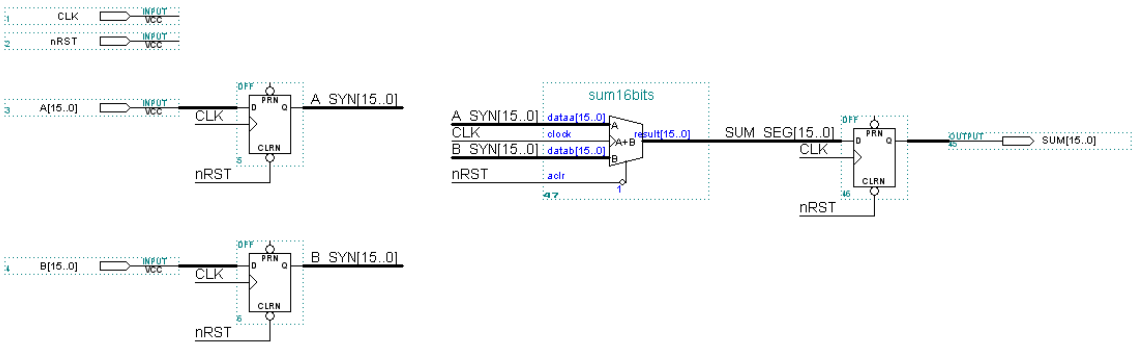

Figura EP.65

El funcionamiento del circuito es idéntico al de la versión segmentada realizada en el ejemplo de la figura EP.59. Para una realización sobre la misma FPGA que aquel se obtiene una frecuencia máxima de funcionamiento de 84'74 MHz y una ocupación de 65 LEs.

Si la velocidad alcanzada resulta insuficiente, se puede aumentar el grado de segmentación. Con una latencia de 3 ciclos de reloj, el circuito puede funcionar a

106’38, con una ocupación de 102 LEs. En el cronograma de la figura EP.66 se muestra una simulación del funcionamiento de esta versión del circuito.

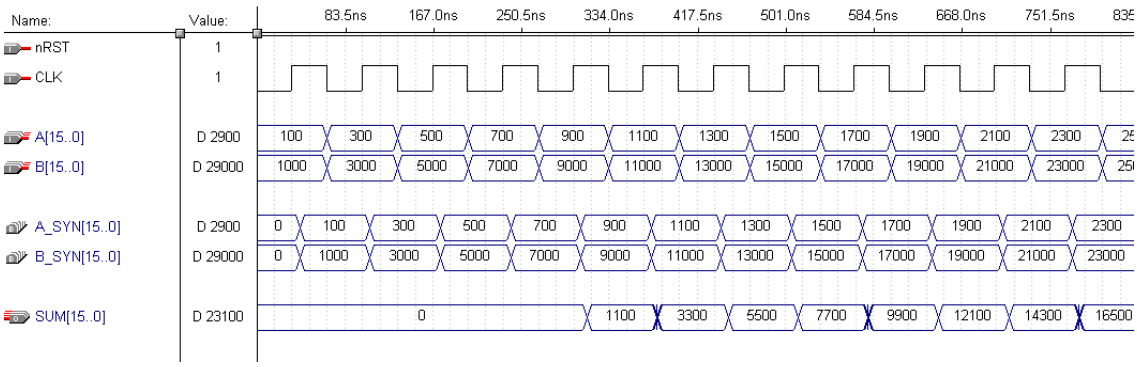

Figura EP.66

Con latencias mayores, la frecuencia máxima de operación apenas aumenta –o incluso disminuye- y, en cambio, se dispara el consumo de recursos lógicos. Esto indica que el factor predominante en los retardos es ya el asociado a los recursos de interconexión y a los flip-flops y que, por tanto, no se puede sacar provecho de mayores grados de segmentación en este circuito.

3.2.2 Rutas multiciclo

Una forma de evitar que la lógica combinacional lenta de un circuito impida alcanzar la frecuencia de funcionamiento deseada consiste en permitir que su régimen transitorio se desarrolle a largo de dos o más ciclos de reloj. A las rutas de procesamiento combinacional que operan de esta forma se las denomina rutas multiciclo –en jerga de diseño digital, *paths* multiciclo.

El esquema de construcción de los *paths* multiciclo se muestra en la figura 59. Como puede verse, las salidas de la lógica combinacional están conectadas a *flip-flops* con *clock enable*; realizando un control adecuado de la habilitación de reloj puede permitirse que las salidas de la lógica combinacional dispongan de más de un periodo de reloj para estabilizarse antes de actuar sobre el *flip-flop* de salida.

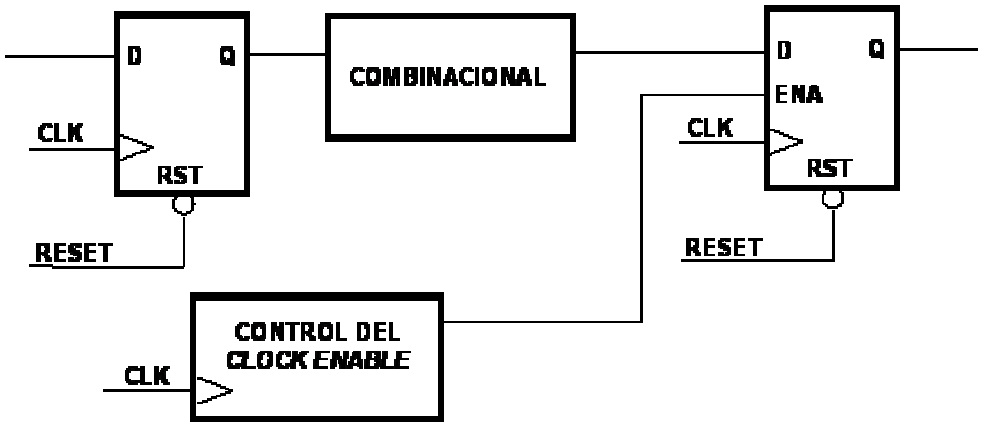

Figura 59. Ruta multiciclo

En el cronograma de la figura 60 se muestra la dinámica del funcionamiento de una ruta multiciclo con un retardo de dos periodos de reloj. El circuito combinacional tiene un retardo mayor que un periodo de reloj, pero menor que dos. La habilitación de reloj de los *flip-flops* se activa dos flancos después de que se produzca un cambio en las entradas de la lógica combinacional, evitando que las salidas puedan actuar en el flanco de reloj que ocurre mientras el combinacional está aún en régimen transitorio.

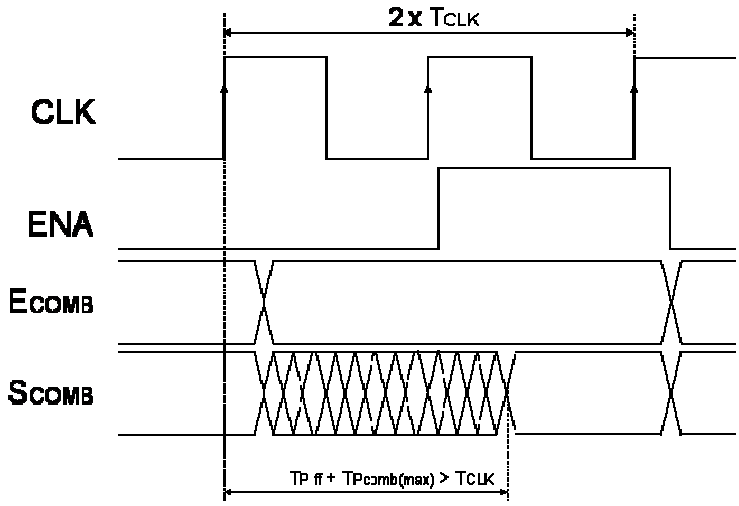

Figura 60. Retardo de dos ciclos

El número de periodos de reloj, N, que debe durar una ruta multiciclo es el menor número que cumpla la siguiente relación.

$$N \times T_{CLK} > T_{P\ COMBINACIONAL\ max} + t_{P\ FF\ max} + t_{SU\ FF}$$

Por ejemplo, en un circuito síncrono que debe funcionar con una frecuencia de reloj de 100 MHz hay un bloque combinacional, encajado entre *flip-flops*, con un tiempo de retardo de 32 ns. Si se desea recurrir a la construcción de una ruta multiciclo para poder alcanzar la frecuencia de reloj requerida, y suponiendo que los tiempos de propagación y *set-up* de los *flip-flops* utilizados suman 5 ns, es necesario que el *path* multiciclo dure:

$$N \times 10ns > (32\ ns + 5\ ns) = 37\ ns;\ N = 4$$

Hay que hacer notar que para aplicar esta técnica resulta imprescindible que las entradas de la ruta conmuten con una cadencia menor o igual al número de periodos de reloj que tarde en ejecutarse la operación; en caso contrario hay que optar por la segmentación del combinacional. En general, esta es la alternativa preferible, por su simplicidad y el escaso coste *hardware* y de diseño que conlleva –comparado con las técnicas de segmentación-, siempre y cuando se cumpla la condición anterior.

Por ejemplo, el circuito de la figura EP.67 es una versión del sumador empleado en ejemplos anteriores, en la que se han realizado modificaciones para construir una ruta multiciclo con un retardo de dos periodos de reloj.

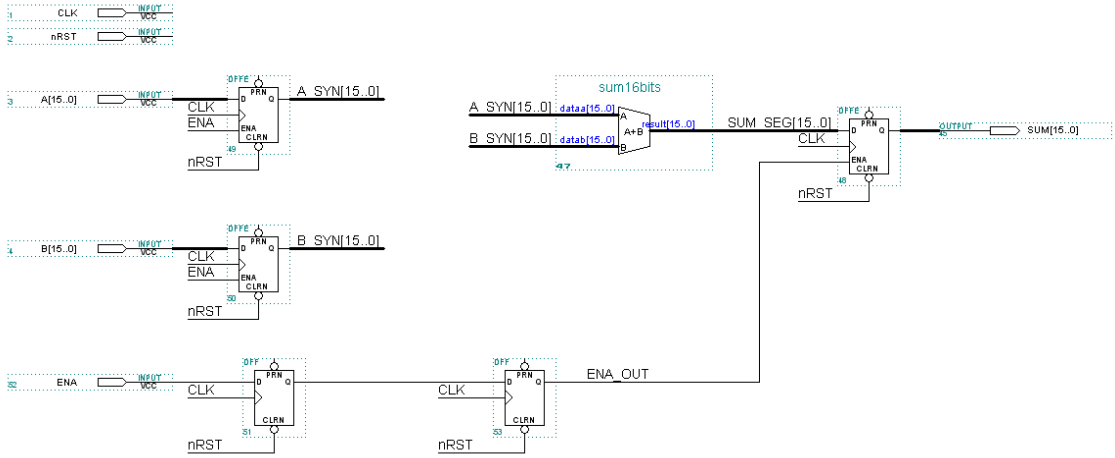

Figura EP. 67

Como puede observarse, el circuito cuenta con una entrada (ENA) que controla la carga de los sumandos en los registros de entrada. Esta misma entrada, retardada, garantiza que se dispone de dos ciclos de reloj para calcular la suma, antes de que se habiliten los flip-flops de salida.

La realización de este circuito en una FPGA de ALTERA nos permite analizar las características de esta solución. El analizador de tiempos del entorno MAX+plus II indica que el circuito puede funcionar a 33'22 MHz; los retardos que limitan la frecuencia de reloj son debidos, obviamente, al sumador; el retardo combinacional máximo vale 30'1 ns. Como cada suma se realiza en dos ciclos de reloj, podríamos hasta duplicar la frecuencia de reloj calculada por el analizador (suponiendo que el resto de la lógica puede funcionar a dicha frecuencia). El cronograma de la figura EP.68 muestra el funcionamiento del circuito con una señal de reloj de 50 MHz (un periodo de 20 ns).

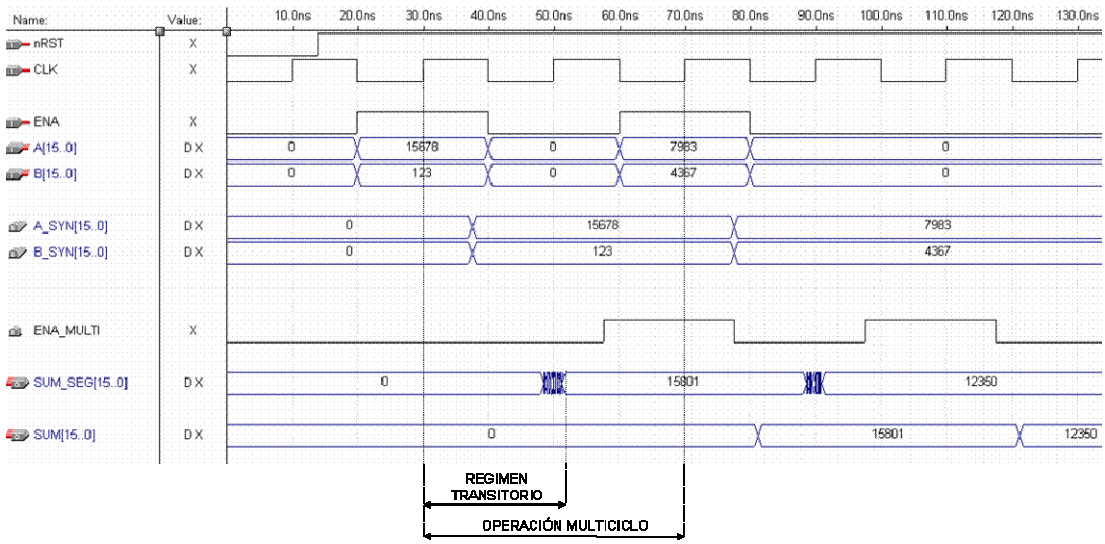

Figura EP.68

Como puede observarse, aunque el régimen transitorio del sumador dura más de un ciclo de reloj, el resultado de la suma se registra correctamente al cabo de dos ciclos, que es, en definitiva, el retardo equivalente de una suma. En este ejemplo se pone de manifiesto que en una ruta multiciclo los datos de entrada pueden renovarse, como máximo, con una cadencia igual a la duración de cada operación; en el caso del sumador, cada dos ciclos de reloj.

Un problema importante que acarrea el uso de *paths* multiciclo es que complica el cálculo automático de la frecuencia máxima de funcionamiento debido a que hay que especificar, al analizador de tiempos que se vaya a utilizar, que algunos elementos del circuito disponen de varios ciclos de reloj para completar sus operaciones. Los analizadores de tiempos disponen de opciones que permiten distinguir la lógica del circuito que dispone de varios ciclos de reloj para completar la estabilización de sus salidas.

3.2.3 Retardos de Salida. *Clock Managers*

En los apartados anteriores hemos presentado técnicas para alcanzar una determinada frecuencia de funcionamiento cuando surgen problemas derivados de los retardos de la lógica interna de un circuito síncrono. Pero la frecuencia de reloj no es la única especificación de tiempos que podemos encontrarnos entre los requisitos dinámicos establecidos para un sistema, es frecuente, también, que existan restricciones que afectan a los retardos máximos de las salidas respecto a los flancos de reloj.

Los retardos de salida de cualquier chip están penalizados, respecto a los internos, por el retardo de los amplificadores de corriente asociados a los pines. Por ello solemos encontrarnos con que los retardos de salida son mayores que los de las rutas combinacionales internas que limitan la frecuencia de operación. Por ello, cuando el circuito funciona a una frecuencia próxima a la máxima, los retardos de las salidas pueden tener una duración mayor que el periodo de reloj. Esta situación puede resultar, o no, tolerable. En otras ocasiones ocurre, simplemente, que las salidas tienen que cumplir con un determinado retardo máximo porque actúan sobre lógica que utiliza su mismo reloj –u otro con el que mantiene un desfase conocido- y tienen que respetar un tiempo de *set-up* dado. En cualquier caso, independientemente de la causa que las determine, cuando las especificaciones para los retardos de salida son exigentes en comparación con las características dinámicas de la tecnología de realización, es frecuente que resulten insuficientes los esfuerzos que pueden realizarse variando el diseño lógico del circuito –si una salida registrada no cumple con los tiempos máximos de retardo, no hay ninguna alternativa de diseño que mejore la situación; si es una salida combinacional, se puede intentar reducir el retardo hasta un determinado límite que puede, o no, permitirnos alcanzar el retardo deseado-; suele resultar más eficaz, en cambio, intentar optimizar la colocación de la lógica que genera las salidas dentro del *chip* que se esté utilizando o utilizar versiones modificadas del reloj del circuito generadas mediante PLLs o DLLs (*clock managers*).

En las FPGAs la medida más simple y económica para mejorar los retardos de las salidas registradas, considerando el esfuerzo a emplear por el diseñador, consiste en ubicar los *flip-flops* que las manejan en las estructuras lógicas asociadas a los pines del *chip*. En la figura 61 se muestran los recursos disponibles en un elemento de entrada-

salida (IOE, *I/O Element*) de una FPGA de la familia Cyclone II de ALTERA (en otras familias de ALTERA y en las FPGAs de otros fabricantes existen estructuras parecidas a esta). Como puede observarse se dispone de recursos que permiten ubicar *flip-flops* para manejar las entradas o salidas en función de la direccionalidad del *pin*.

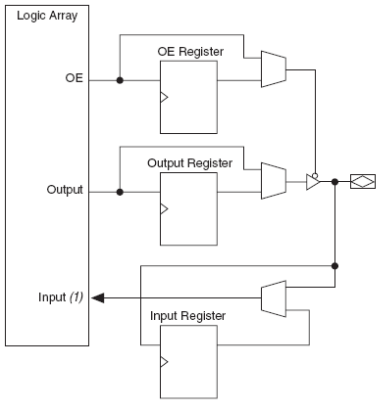

Figura 61. IOE de la familia Cyclone II

Si se coloca el *flip-flop* que maneja la salida registrada en el bloque asociado a un *pin* se eliminan, del retardo total, los retardos de interconexión que existirían si el *flip-flop* estuviera colocado en una célula interna. Esta solución tiene algunas limitaciones: en primer lugar, no sirve para reducir los retardos de las salidas combinacionales, porque la salida del *flip-flop* está conectada directamente al *pin*; por otra parte el margen de reducción de los retardos no es, en general, muy grande y, en consecuencia, sólo permite ajustar retardos de salida que exceden levemente los niveles deseados; por último, puede llevar aparejada una disminución de la frecuencia máxima de funcionamiento del circuito porque la colocación de los *flip-flops* de salida en los *pines* aumenta los retardos de algunas rutas combinacionales internas.

Por ejemplo, el sumador de 8 bits de la figura EP.69, realizado sobre una FPGA EPF10K30RC208-4 de ALTERA puede funcionar a una frecuencia de 101.01 MHz (T_{CLK} = 9'9 ns).

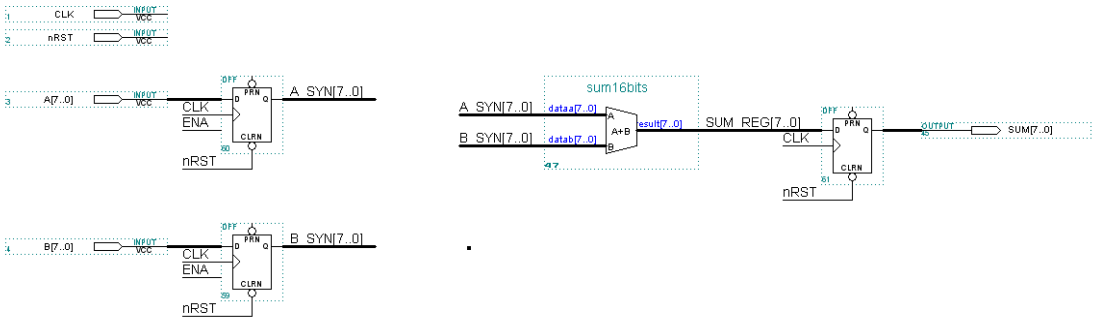

Figura EP.69

El retardo máximo de las salidas (SUM7 a SUM0) respecto al flanco de subida del reloj varía entre 13'3 ns (la más rápida) y 14'9 ns (la más lenta). Si se colocan los flip-flops de salida en los IOEs de la FPGA, se obtiene, para todas ellas, un retardo máximo de 11'2 ns, pero la frecuencia máxima de operación pasa a ser de 48 MHz (T_{CLK} = 20'5 ns).

La solución más eficaz para reducir los retardos de salida consiste en utilizar una versión desfasada del reloj del circuito para controlar las salidas. El fundamento de esta idea es muy sencillo (figura 62) : si a los *flip-flops* de salida les llega un reloj adelantado en fase (CLK'), los retardos de las salidas controladas por este reloj se reducen respecto al reloj del resto del circuito (CLK) en un tiempo igual al desfase existente entre ambas señales de reloj.

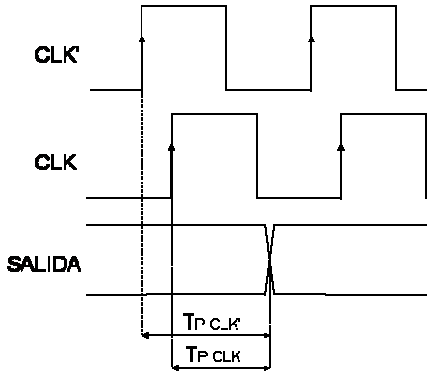

Figura 62. Reloj de salida adelantado en fase

Esto constituye, evidentemente, una infracción de las reglas de diseño síncrono, pero, como en otros casos ya mencionados en este texto, se puede defender por su eficacia, la ausencia de alternativas síncronas para resolver el problema y porque no afecta apenas a la parte síncrona del circuito: simplemente hay que tener en cuenta, en el cálculo de la frecuencia máxima de funcionamiento, el desfase del reloj como una componente de skew que afecta a los *flip-flops* de salida que utilizan la versión desfasada del reloj.

Para generar una versión desfasada del reloj del circuito la mejor solución consiste en utilizar un *clock manager*, es decir un módulo de "manejo de relojes". Estos bloques funcionales son un recurso disponible en la mayor parte de las familias de FPGAs modernas –es *hardware* que forma parte de la estructura del chip, que se puede o no utilizar y que se añade como un componente de diseño más en los ficheros de especificación del circuito-; se construyen utilizando PLLs o un DLLs (los PLLs son analógicos, mientras que los DLLs son digitales) y tienen diversas aplicaciones prácticas, entre las que cabe destacar la eliminación del *jitter* de reloj (de la inestabilidad de la frecuencia), la generación de relojes de una frecuencia múltiplo o divisor de la de un reloj patrón y, la que resulta de interés en nuestro caso, la generación de relojes desfasados respecto a un reloj patrón.

Cuando no se dispone de *clock managers*, pueden utilizarse *flip-flops* activos por flanco de bajada para conseguir un adelanto de 180 grados en la fase del reloj de los *flip-flops* de salida. Esta solución -que da lugar, por cierto, a un sistema síncrono con dos fases de reloj como los descritos en el apartado 2.7 de este texto-, tiene "efectos colaterales" serios (para que funcione, los retardos mínimos suelen tener que ser relativamente grandes, mientras que los máximos no pueden ser excesivamente grandes), por lo que debe recurrirse a ella sólo cuando no queda ninguna otra alternativa.