SISTEMAS ELECTRÓNICOS. PRÁCTICA Nº 2.

Grado en Ingeniería en Electrónica, Robótica y Mecatrónica.

Objetivo:

Utilizando el software de Xilinx ISE 14.7, se pide implementar un circuito en la placa Nexys3 (FPGA Spartan-6) que realice los siguientes procesos:

- A través de un teclado hexadecimal auxiliar, el usuario introduce un valor entre 0 y 9 (en su versión básica) o entre 0 y 12 (en su versión opcional), validando el dato tecleado mediante la pulsación de la letra "A".
- Las cifras numéricas que se van tecleando a la entrada se mostrarán en los displays 7 segmentos de la placa Nexys3 (**A IMPLEMENTAR**).
- Estas cifras introducidas una a una en formato BCD se convierten a un número binario natural de 4 bits (sin signo) mediante un bloque conversor de BCD-a-binario.
- A continuación, se calculará el factorial de dicho número binario y se dará salida al resultado. Utilizando un bloque multiplicador 18x18 en complemento de 2 de los que incluye la Spartan-6, la salida tendrá como máximo 36 bits de representación, igualmente expresada en complemento a 2 (**A IMPLEMENTAR**).
- Se convertirá dicho número binario de salida a BCD para poder representarlo en los displays de 7 segmentos. Dado que con 36 bits podríamos llegar a utilizar 11 cifras en BCD para representar el resultado, el conversor binario-a-BCD proporcionará todos estos datos.
- Utilizando los 4 displays de 7 segmentos de la placa Nexys3 se mostrará el resultado de forma multiplexada, es decir, utilizando los switches de la placa SW(1:0) se mostrará el resultado por partes (**A IMPLEMENTAR**):

  Resultado de salida (en BCD):

BCD10	BCD9	BCD8	BCD7	BCD6	BCD5	BCD4	BCD3	BCD2	BCD1	BCD0

  Combinaciones de salida en display de 7 segmentos:

SW(1)	SW(0)	Displays Nexys 3
0	0	BCD3 – BCD0
0	1	BCD7 – BCD4
1	0	BCD10 – BCD8
1	1	"0000"

- Se generará una señal de salida "factorial_bcd_nuevo", que será un pulso a "1" lógico durante 1 ciclo de reloj, cuando ya estén los datos presentes en el display de 7 segmentos (**A IMPLEMENTAR**).
- Para volver a comenzar la operación se esperará a que el usuario pulse la tecla "C" mientras se muestra el resultado por los displays (**A IMPLEMENTAR**).

Se proporciona un circuito donde se incluyen la mayoría de módulos indicados salvo los marcados como "A IMPLEMENTAR". Por tanto, el objetivo de la práctica será aprender el funcionamiento de los bloques ya incluidos en la misma y, una vez comprendida, realizar la implementación de los bloques indicados.

Restricciones:

1.- Tened en cuenta que, si se introduce el número "0", el resultado de su cálculo factorial es "1".

2.- Se supone que el usuario no introducirá números mayores de "9" para el cálculo en el diseño básico o "12" para el opcional.

3.- Para realizar el bloque de multiplexación de datos hacia los displays de 7 segmentos se debe tener en cuenta que, en el momento de entrada de datos, se muestran los datos BCD tecleados por el usuario, y en el momento de mostrar el cálculo del factorial, se seleccionarán las cifras BCD del mismo que se indiquen con los switches (1:0), como se ha indicado anteriormente.

Fases de trabajo:

1ª.- **En primer lugar, se modificará el diagrama de bloques funcionales proporcionado para el módulo "bloque_factorial" para que incluya los nuevos módulos.** En dicho diagrama se establecerá todo el camino de los datos ("*datapath*") desde que éstos entran en el ese módulo hasta que salen hacia los displays de 7 segmentos, mostrando a grandes rasgos el intercambio de señales que se produce entre los distintos bloques, e indicando los circuitos de control necesarios para cada una de las partes. El diagrama de bloques funcionales del diseño que se proporciona se encuentra explicado en este mismo enunciado en los siguientes apartados, por lo que el estudiante sólo deberá modificar las partes que sean necesarias del mismo y elaborar el resto del diagrama para completar los objetivos que se piden del diseño. Este diagrama se entregará en una tarea de forma previa a los resultados finales.

2º.- **Se crearán dos diagramas de bloques detallados del interior del bloque que realiza el cálculo del factorial y de la multiplexación de datos hacia los displays 7 segmentos**, por tanto, tendremos dichos bloques como "cajas negras" en el diagrama superior de la 1ª fase ("bloque_factorial"), y en este 2º apartado se detallará el contenido interno de estos bloques. Los diagramas también se entregarán de forma previa a los resultados finales de la práctica.

3ª.- **Se desarrollarán las partes del circuito a implementar utilizando el editor de esquemáticos** incluido en la herramienta ISE, junto con los bloques ya proporcionados en el mismo diseño, los elementos de la librería que sean necesarios (biestables, puertas lógicas, comparadores, multiplexores, etc), y bloques creados con la herramienta **"CORE Generator"**. Asimismo, se utilizará el editor de máquinas de estados (FSM) **Active-HDL** para la elaboración y/o modificación de los circuitos de control (máquinas de estados, FSM) que sean necesarios.

4ª.- **Se simulará el diseño** de manera funcional hasta que trabaje correctamente. Para la simulación será imprescindible la modificación del "Test Bench" en VHDL proporcionado en la práctica ("sim_bloque_factorial.vhd") **de tal manera que se refleje el funcionamiento completo del circuito**. En la hoja de resultados final se adjuntarán diversas capturas de las señales más importantes del circuito que muestren que se está realizando correctamente el cálculo del factorial, junto con los comentarios que el estudiante estime oportunos.

5ª.- **Se procederá a la implementación del diseño**, de manera que se asegure que funciona a la máxima velocidad posible, **y se probará sobre la placa Nexys3 junto con la placa auxiliar del teclado hexadecimal**). Se podrá utilizar el Remote Lab para probar el funcionamiento del diseño igualmente.

Diseño opcional.

La versión básica del diseño aceptará datos a la entrada entre 0 y 9 para calcular el factorial. Para la parte opcional se ampliará el dato de entrada hasta 12, dado que presenta una dificultad añadida al no poder calcularse el factorial de manera totalmente iterativa utilizando un multiplicador de 18x18 bits. Se pide por tanto **diseñar una solución opcional para poder ampliar el rango de datos de entrada hasta 12 manteniendo el tamaño de dicho multiplicador en 18x18 bits en complemento a 2**.

Trabajo a presentar por el estudiante (OBLIGATORIO):

1.- Como se indicó en los objetivos, será necesario modificar la práctica para incluir toda la funcionalidad requerida, es decir, cálculo del factorial y multiplexación de los resultados hacia los displays de 7 segmentos de la Nexys3. Se realizará y se mostrará en clase al profesor, en primer lugar, los diagramas de bloques, y se entregarán los mismos en la tarea indicada. A continuación, se modificará el diseño de Xilinx ISE para incluir los nuevos bloques, y se mostrará en clase al profesor el diseño realizado. Se podrá utilizar el programa Active-HDL para realizar nuevas máquinas de estado si son necesarias, o para modificar las existentes.

2.- Modificar el script de simulación para incluir las nuevas situaciones a simular, es decir, introducción de diversos datos para el cálculo factorial, incluyendo los casos más "extremos" (números muy pequeños y números muy grandes), observando correctamente los valores de salida. Se mostrará en clase al profesor la simulación del circuito realizada.

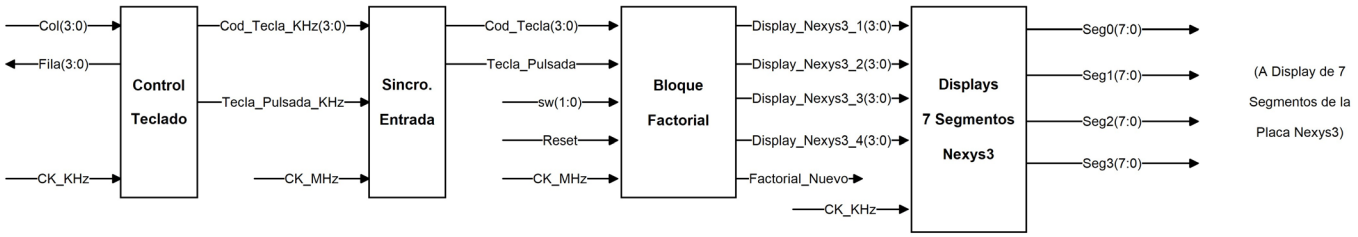
3.- Se implementará el circuito correctamente sin errores, y se probará en la placa Nexys3, mostrando igualmente al profesor el resultado final. Este diseño funcionando de manera correcta será el que se deba presentar como resultado de la práctica, junto con la hoja de resultados que se incluye aparte.

4.- Se subirá a la tarea de la "Práctica 2 – Entrega de Diseño y Hoja de Resultados" del Campus Virtual un archivo comprimido que contenga la carpeta con la práctica completa (será el mismo diseño por cada pareja que trabaje junta en el laboratorio), incluyendo diseño de Xilinx ISE y el de Active-HDL para las máquinas de estado, y, aparte, en un segundo archivo separado, la "Hoja de Resultados" que se incluye en el archivo comprimido de la práctica, una vez rellena (en formato PDF o Word) y completada de manera INDIVIDUAL.

Diagrama de Bloques General:

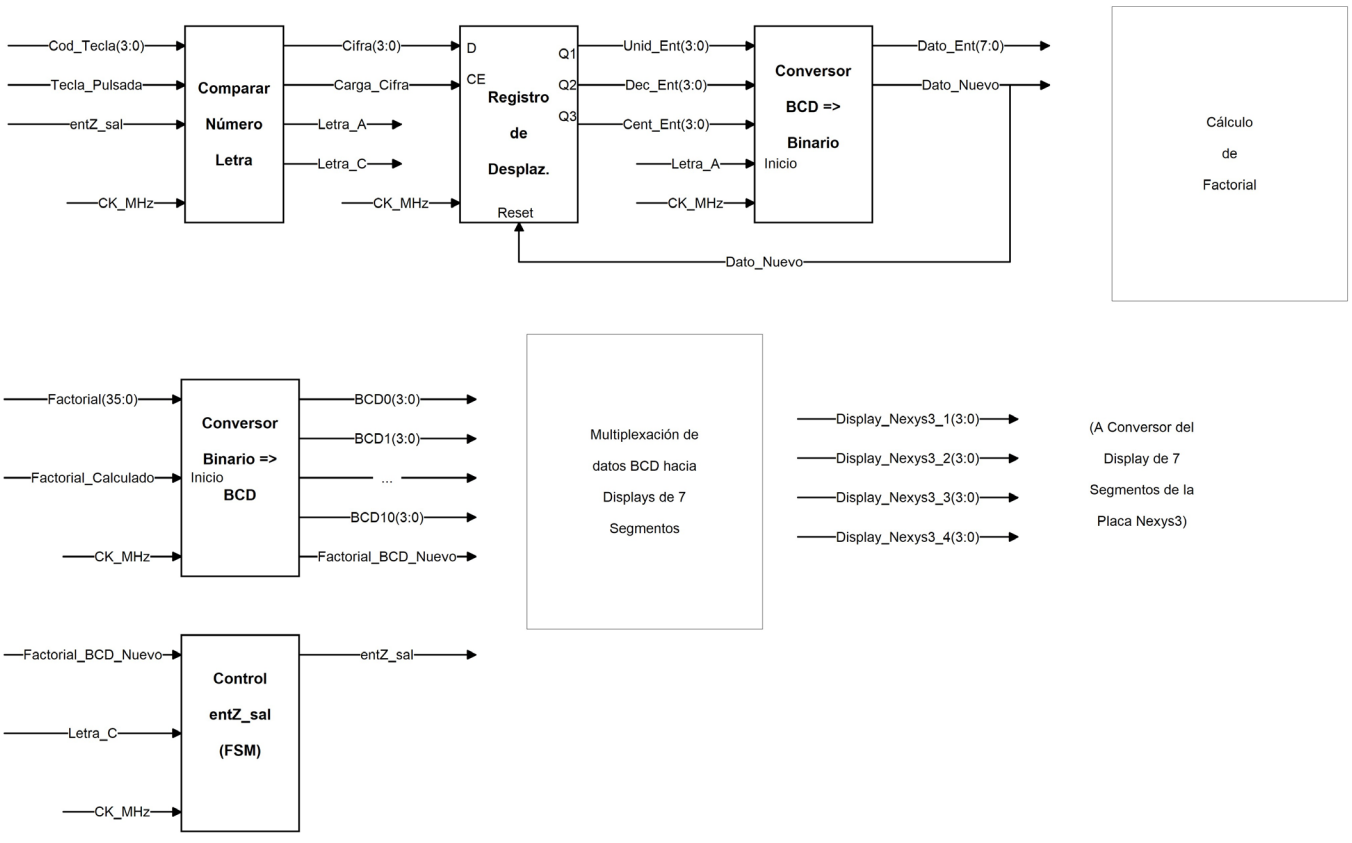
El siguiente diagrama muestra el recorrido de los datos de entrada (introducidos desde el teclado hexadecimal, a través del bloque "Control Teclado") desde que entran al circuito, se sincronizan adecuadamente, se introducen en el bloque del factorial, y salen del mismo los datos en BCD a mostrar por los 4 displays de 7 segmentos de la placa Nexys3:

Diagrama de Bloques General (Top)
Práctica 2

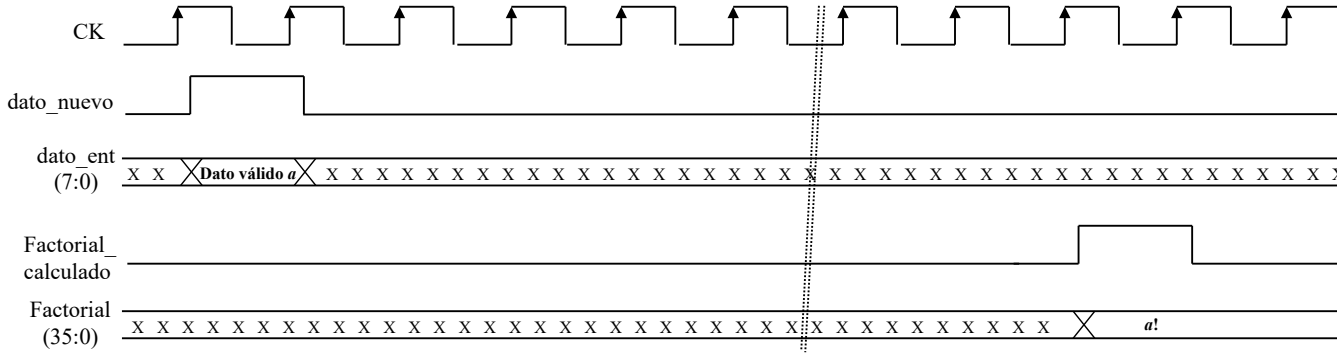


Mirando dentro del interior del "Bloque Factorial", vemos que dentro del mismo se detecta si la pulsación de la tecla corresponde a un número o a una letra (para realizar la acción correspondiente), se almacenan en un registro de desplazamiento (sólo las cifras numéricas) para su posterior conversión de BCD a binario al pulsar la tecla "A", y se envía al bloque de cálculo del factorial iniciándose el cálculo, y al acabar se ofrece el dato a su salida ("Factorial(35:0)") así como una señal de sincronismo "Factorial_Calculado" que indica que se ha acabado el proceso, por lo que el dato en binario se puede convertir a BCD, multiplexar después los datos BCD según proceda, y darle salida a través de los displays de 7 segmentos. También se incluye una pequeña máquina de estados "Control entZ_sal" encargada de gestionar si se está en momento de introducción de datos (entZ_sal = '0') o en momento de salida de datos (entZ_sal = '1'):

Diagrama de Bloques "Factorial"
Práctica 2



Es importante remarcar que casi todos estos bloques generan uno o varios datos de salida y, a la vez, una señal de sincronismo consistente en un "pulso" a 1 lógico, de manera que avisa al siguiente bloque que debe recoger el dato o datos que le está suministrando, tal y como se indica en el cronograma del cálculo que se muestra a continuación:



Control de Teclado:

Este bloque se encarga de realizar un barrido continuo sobre el teclado hexadecimal para recoger las pulsaciones que se vayan produciendo y generar 2 señales de salida:

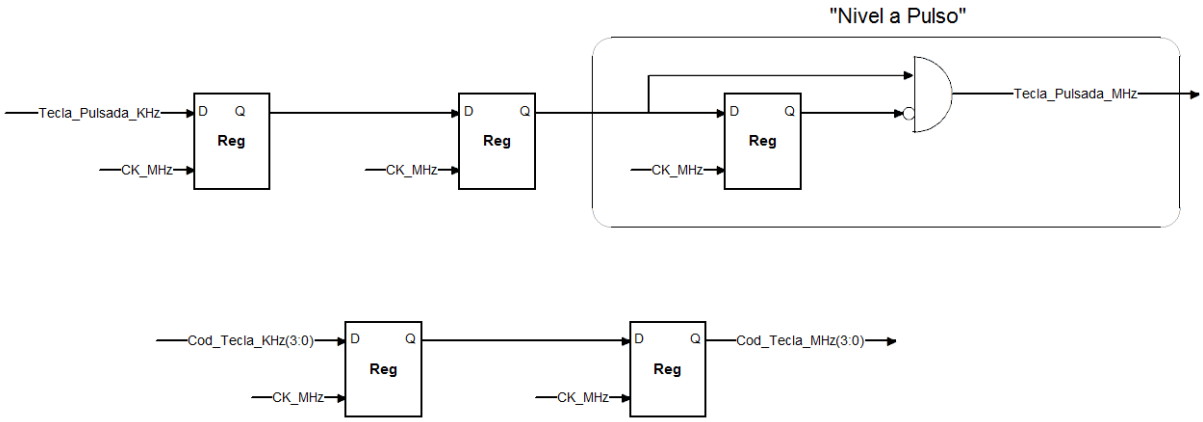
1) Cod_Tecla_Pulsada_KHz(3:0): Es el código en hexadecimal de la tecla que se ha pulsado (de 0 a F).
2) Tecla_Pulsada_KHz: Es 1 solo pulso del reloj de este bloque ("CK_KHz") a "1" lógico para indicarle al siguiente módulo que hay una nueva pulsación disponible.

Hay que indicar que para recoger los valores de las pulsaciones se utiliza un reloj lento, de unos 20 KHz, con el objetivo de eliminar los posibles rebotes en la señal que proviene del teclado hexadecimal. Posteriormente, será necesario sincronizar estas señales con el reloj usado en el sistema, normalmente del orden de los MHz (mucho más rápido, lógicamente).

Sincronización de Entrada:

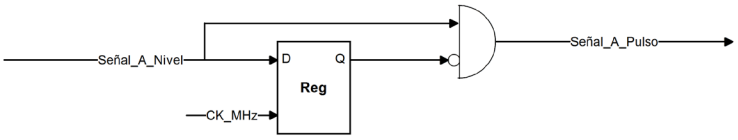
En este bloque se capturan las pulsaciones externas del teclado hexadecimal. Como son señales que vienen sincronizadas con un reloj más lento ("CK_KHz"), es necesario sincronizarlas con el reloj interno de la FPGA, llamado aquí "CK_MHz", haciendo que las mismas pasen por 2 biestables tipo D seguidos.

Además, como la señal "Tecla_Pulsada_KHz" debe indicar con un pulso de 1 ciclo de reloj del sistema que hay una nueva tecla hexadecimal detectada y que el código de dicha tecla pulsada viene en el bus "Cod_Tecla_MHz(3:0)", haremos que la señal de sincronismo pase por un bloque de "Nivel a Pulso", para transformar el pulso de 1 ciclo de reloj "CK_KHz" en un pulso del reloj "CK_MHz", mucho más corto que el anterior, y así el siguiente bloque lo recogerá adecuadamente.



Nivel A Pulso:

Este circuito se encarga de convertir una señal que dura varios ciclos de reloj en una señal que dura 1 solo ciclo de reloj, es decir, forma un "pulso" al cambiar dicha señal de '0' a '1' lógico.



Comparar Número Letra:

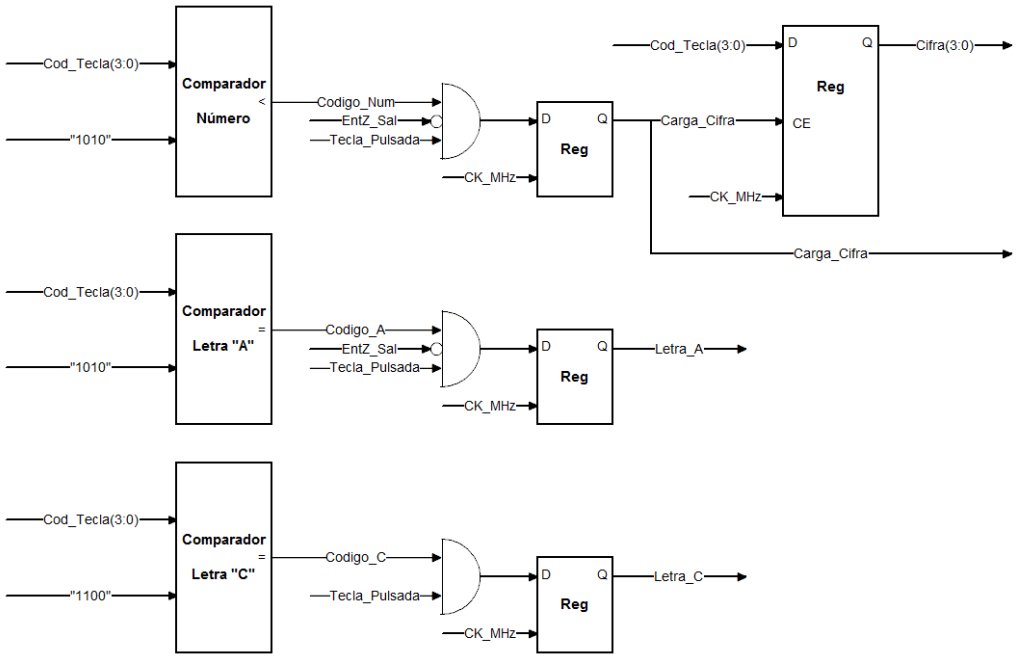
Este bloque realiza diversas comparaciones entre el código de la tecla que se acaba de pulsar y unos valores conocidos:

1) Cod_Tecla(3:0) < 10 ➔ Indica que se ha pulsado un número, y por tanto éste se deberá almacenar en el registro de desplazamiento que hay a continuación, para su posterior conversión de BCD a binario.
2) Cod_Tecla(3:0) = 10 ("A" en hexadecimal) ➔ Indica que se ha pulsado la letra "A" del teclado, por lo que se deberá indicar al bloque de conversión de BCD a binario que realice su tarea.
3) Cod_Tecla(3:0) = 12 ("C" en hexadecimal) ➔ Indica que se ha pulsado la letra "C" del teclado, por lo que se deberá indicar al bloque de cálculo que vuelva a su estado inicial para aceptar un nuevo dato.

Tras cada comparador existe una puerta AND que deja pasar el resultado de la comparación si existe un pulso en "Tecla_Pulsada", ya que el código de la tecla está disponible en todo momento, pero el resultado de la comparación sólo debe ser importante en el momento de la pulsación, no después. El resultado se recoge finalmente en un biestable dependiendo del objetivo final del mismo:

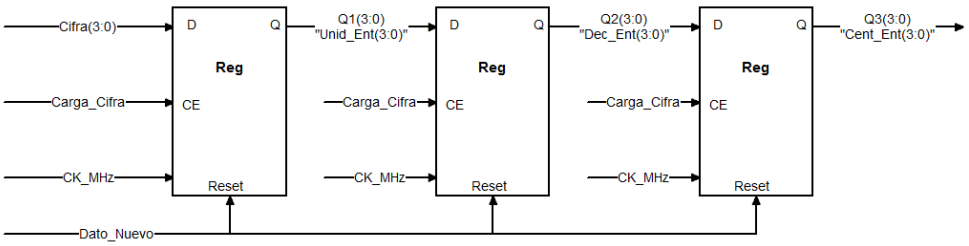
1) Si la tecla correspondía a un número, se genera "Carga_Cifra", que servirá para el registro de desplazamiento posterior, y además con ella se cargará el código de la tecla en un biestable para que esté disponible y sin alteraciones para el siguiente bloque.
2) Si la tecla correspondía a la letra "A" o la "C", se generan sendas señales que irán a distintos bloques, como se aprecia en el diagrama general.

Hay que indicar que la señal "EntZ_Sal" nos indica con un "0" lógico cuando estamos en un proceso de aceptar datos a la entrada, y con un "1" cuando estamos mostrando los resultados de salida; es por ello que para detectar pulsaciones de números y la de la letra "A" tenemos que asegurarnos que estemos en un proceso de entrada de datos, por lo que esta señal entrará negada en la puerta AND tras los comparadores correspondientes a dichos procesos. Para la letra "C" no será necesario, ya que sólo tiene la función de reiniciar el proceso de cálculo cuando estamos visualizando el resultado final.



<u>Registro de Desplazamiento</u>:

Es una simple "cola" de datos formada con 3 biestables tipo D de 4 bits cada uno, con señal de CE, que cargará hasta 3 cifras BCD del dato de entrada. Posee una señal de "Reset" síncrono para que, cada vez que se cargue un dato completo y se convierta de BCD a binario, se ponga a 0 tras dicha conversión.

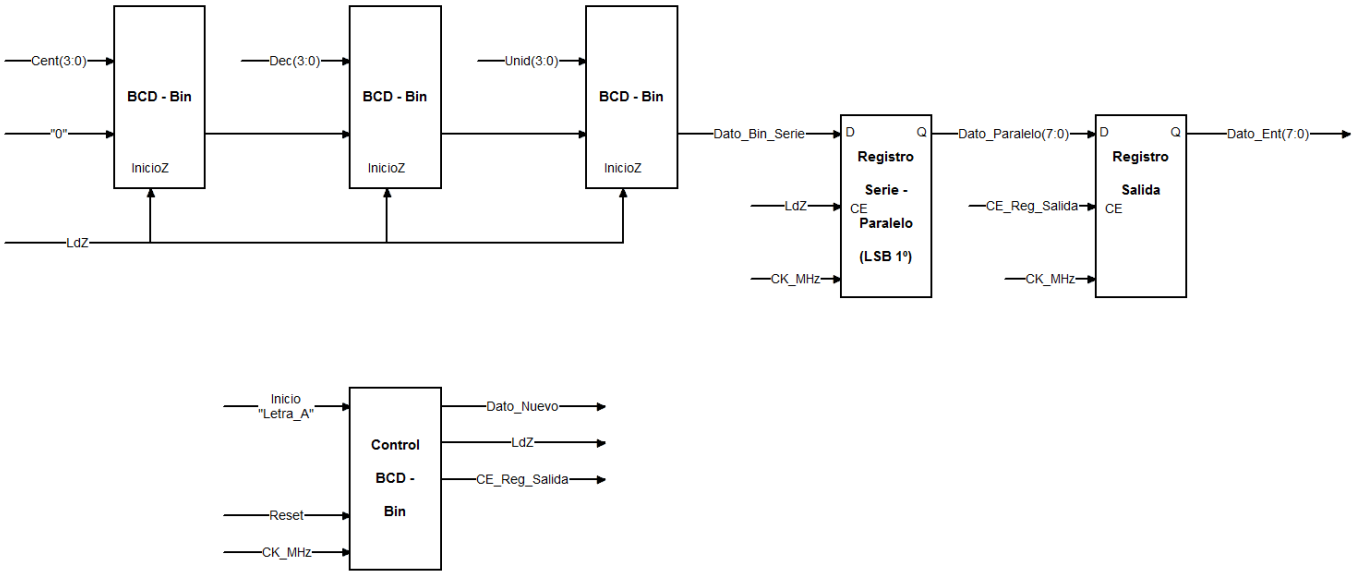


<u>Conversor BCD => Binario</u>:

Este conversor está realizado siguiendo el esquema de la nota de aplicación de Xilinx "XAPP 029", disponible en el Campus Virtual de la asignatura. Se trata de un conversor BCD-Binario serie, de manera que se ponen tantos bloques "BCD - Bin" como cifras BCD se tengan, y se obtiene el dato convertido a binario, bit a bit, en tantos ciclos como sea su longitud máxima en binario (en nuestro caso, 8 bits, 8 ciclos). Durante la conversión se van cargando esos bits en un registro con entrada serie y salida paralelo, de manera que se ofrezca el dato completo al exterior una vez finalizado el proceso.

Toda la conversión está controlada por una máquina de estados, "Control BCD-Bin", que genera las señales necesarias para que todos los bloques funcionen y se sincronicen adecuadamente. El inicio de todo el proceso se desencadena con la pulsación de la letra "A" del teclado, como se ha comentado anteriormente.

Se ha utilizado este bloque conversor de 8 bits, que son más de los necesarios para nosotros que sólo utilizaremos los 4 menos significativos, para reutilizar este bloque ya diseñado previamente en otro circuito. Se podría optimizar en nuestro caso para que generara solamente 4 bits a partir de 2 cifras BCD, obviamente.

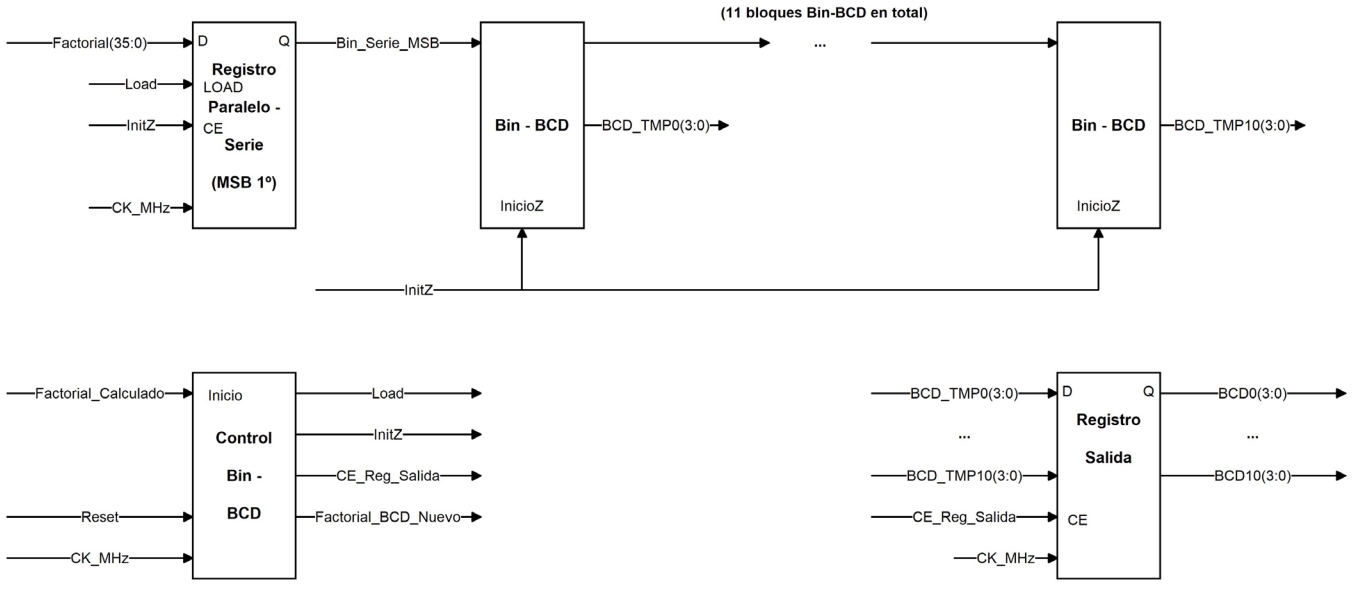


Conversor Binario => BCD:

Este conversor también está realizado siguiendo el esquema de la nota de aplicación de Xilinx "XAPP 029", disponible en el Campus Virtual de la asignatura. Se trata de un conversor Binario-BCD serie, de manera que se ponen tantos bloques "Bin - BCD" como cifras BCD se deseen obtener, y se proporciona el dato binario a convertir bit a bit, en tantos ciclos como sea su longitud máxima (en nuestro caso, 36 bits de parte entera del resultado. Para realizar esta conversión es necesario cargar previamente el dato en binario en un registro de entrada paralelo y salida serie, enviando en primer lugar el bit más significativo (MSB). Al finalizar la conversión se cargará el resultado en un registro de salida, de manera que se ofrezca el dato completo al exterior.

Todo el proceso está gobernado por la máquina de estados "Control Bin - BCD", que se iniciará con un pulso de la señal "Factorial_Calculado", generará las señales necesarias para controlar la conversión bit a bit del dato binario en BCD, cargará los resultados en un registro de salida, y al final del proceso generará la señal "Factorial_BCD_Nuevo" que la recogerá el siguiente bloque para proceder a mostrar los resultados por los displays de 7 segmentos.

Diagrama de Bloques "Conversor Binario => BCD"
Práctica 2

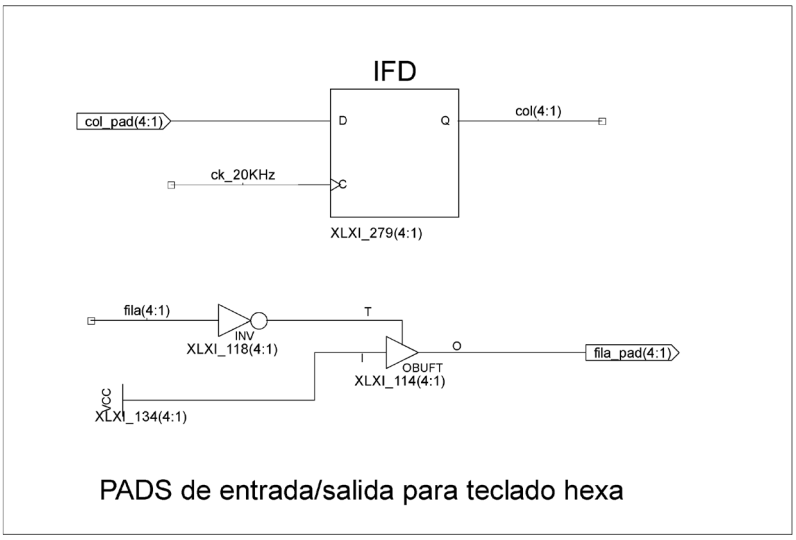


Circuitos Adicionales:

Los siguientes circuitos corresponden a lógica adicional necesaria para que funcione la FPGA. Algunos ya se describieron en la práctica 1, por lo que aquí sólo detallamos los nuevos.

PADS de E/S para el teclado hexadecimal:

El teclado hexadecimal posee unas conexiones a las filas y columnas del mismo, mediante las cuales exploramos continuamente si se está pulsando alguna tecla. Para ello, enviamos un "1" lógico a través de una fila y comprobamos si se recibe dicho "1" por alguna de las columnas, lo cual indicaría que se ha pulsado la tecla que cruza esas dos líneas. A continuación, se sigue testeando la siguiente línea, y así consecutivamente.



PADS de entrada/salida para teclado hexa

Remote Lab:

Como ya se indicó en la práctica 1, este bloque se encarga de comunicar al usuario con el laboratorio remoto para las pruebas vía web sobre la Nexys3. Dicho bloque controla los leds, switches, botones, y displays de 7 segmentos de la placa Nexys3. También se puede utilizar el teclado hexadecimal con el Remote Lab, si bien no necesita pasar los datos a través de este bloque del Xilinx, sino que funciona como un accesorio que se introduce en la web a la hora de programar la FPGA con el diseño.

Simulación:

Se ha creado un script de simulación ("sim_bloque_factorial.vhd") que define el comportamiento de las señales de entrada del módulo "bloque_factorial". En este caso necesitamos definir el reloj del sistema "ck", el reset del sistema "reset", el valor del switch "sw(1:0)" y las señales que provienen del controlado del teclado hexadecimal, que son "cod_tecla(3:0)" y "tecla_pulsada".

Definición del reloj:

Se crea en VHDL un proceso aparte del resto de señales para estimular el reloj, así estará ejecutándose en paralelo al resto de forma indefinida. Se ha definido un reloj de 10 ns (100 MHz) con cierto desfase respecto al resto de señales para que no coincidan sus flancos de subida con los cambios en estas otras señales que definiremos después.

```
-- *** Test Bench - User Defined Section ***
CK_process :process
      begin
            CLOCK_LOOP : LOOP
            ck <= transport '0';
            WAIT FOR 4 ns;
            ck <= transport '1';
            WAIT FOR 5 ns;
            ck <= transport '0';
            WAIT FOR 1 ns;
      END LOOP CLOCK_LOOP;
end process;
```

Definición de una constante con el periodo del reloj:

Se crea en VHDL una constante que tendrá como valor la longitud del periodo de reloj, en nuestro caso 10 ns, y se utilizará para estimular las señales durante un tiempo que sea múltiplo del periodo de reloj:

```
      constant CLK_period : time := 10 ns;
```

Estimulación de señales:

Se crea en VHDL otro proceso para definir el valor del resto de señales en todo momento. Es muy importante inicializar todas ellas a un valor definido desde el instante "0", para que no haya valores indefinidos durante toda la simulación.

En el script se inicializa el reset a '1' lógico y se mantiene durante 200 ns (20 periodos de reloj, definido en la constante "CLK_period"), lo suficiente para que transcurran varios ciclos de reloj de 100 MHz, tras lo cual se baja a '0' lógico dicho reset y se espera un tiempo prudencial antes de que se comience la operación normal del circuito; en este caso hemos usado 30 periodos del reloj:

```
-- Reset e inicialización de señales
reset <= transport '1';
cod_tecla <= transport "0000";
tecla_pulsada <= transport '0';
sw <= transport "00";
WAIT FOR CLK_period*20;

reset <= transport '0';
WAIT FOR CLK_period*30;
```

A continuación, simulamos una pulsación de un número, seguido de la letra “A” del teclado hexadecimal para confirmar el valor del mismo que hemos introducido y comenzar la operación de cálculo del factorial. Esperamos un tiempo suficiente para que se realice la misma y se vean los resultados (1000 ciclos de reloj) y pulsamos la letra “C” para devolver el circuito a su estado original y poder empezar de nuevo:

```
-- Introducimos el número 0 y pulsamos "A"
cod_tecla <= transport "0000"; -- "0"
tecla_pulsada <= '1';
WAIT FOR CLK_period;
tecla_pulsada <= transport '0';
WAIT FOR CLK_period*10;

cod_tecla <= transport "1010"; -- "A"
tecla_pulsada <= '1';
WAIT FOR CLK_period;
tecla_pulsada <= transport '0';
WAIT FOR CLK_period*1000; -- Dejamos tiempo suficiente para ver la salida del
factorial


-- Pulsamos "C" para inicializar los cálculos
cod_tecla <= transport "1100"; -- "C"
tecla_pulsada <= '1';
WAIT FOR CLK_period;
tecla_pulsada <= transport '0';
WAIT FOR CLK_period*10;
```

Este mismo código se puede repetir para simular las pulsaciones de otras cifras y comprobar el funcionamiento completo del circuito. También sería necesario incluir en el script, una vez realizada la operación del factorial, modificaciones del valor del switch “sw(1:0)” para comprobar que se seleccionan correctamente a la salida los datos BCD adecuados para observarlos en los displays de 7 segmentos.

En caso de realizar la parte opcional, se introducirá más de una cifra numérica antes de pulsar la letra “A” para confirmar el dato de entrada.