SISTEMAS ELECTRÓNICOS. PRÁCTICA Nº 1.

Grado en Ingeniería en Electrónica, Robótica y Mecatrónica.

Objetivo:

Utilizando el software de Xilinx ISE 14.7, se proporciona un circuito ya implementado en una placa Nexys3 (FPGA Spartan-6) que realiza el siguiente proceso: Pulsando los botones derecho e izquierdo de la placa se incrementa o decrementa un contador, respectivamente, cuyo valor se visualiza en el display de 7 segmentos correspondiente a las unidades en dicha placa Nexys3. Este contador está limitado para que sólo pueda tener valores entre 0 (mínimo) y 3 (máximo) de manera que, aunque le demos más pulsaciones a estos botones, nunca sube ni baje de los límites establecidos. Además, dependiendo del valor que tengamos en este contador, se generará una señal PWM (*Pulse-Width Modulation, Modulación del Ancho de Pulso*) que se conectará a un led y hará que el mismo se ilumine con diferentes grados de intensidad dependiendo del valor de dicho contador (contador = 0 → Led apagado; contador > 0 → Cuanto mayor sea el valor, mayor será la intensidad del led).

El objetivo de la práctica será aprender el funcionamiento de la misma y, una vez comprendida, duplicar este mismo mecanismo pero de manera extendida. Por tanto, se modificará el diseño para que el mismo proceso de control de la señal PWM se amplíe de la siguiente manera: Ahora se utilizarán los botones de arriba y abajo de la placa Nexys3 para accionar otro contador, distinto al anterior, limitado entre los valores 0 y 15, y se generará otra señal PWM que la llevaremos a un segundo led de la placa, diferente al anterior, para mostrar hasta estos 16 niveles de luminosidad diferentes. Además, el valor del contador en hexadecimal (de 0 a F) aparecerá en el display de 7 segmentos de la Nexys3 correspondiente a las decenas.

Fases de trabajo:

1ª.- Se realizará un diagrama de bloques funcionales que establezca todo el camino de los datos ("*datapath*") desde que éstos entran en el circuito hasta que salen de la FPGA, indicando 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 en este mismo anunciado 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 que se considere necesario para completar los objetivos que se piden del diseño.

2ª.- Se realizará el circuito 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, etc), y contadores creados con la herramienta "CORE Generator".

3ª.- 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_prac1.vhd") de tal manera que se refleje el funcionamiento completo del circuito.

4ª.- 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.

Trabajo a presentar por el estudiante:

1.- Como se indicó en los objetivos, será necesario modificar la práctica para incluir un nuevo bloque PWM de 4 bits que nos ofrezca 16 niveles de iluminación de otro led de la placa, utilizando un circuito muy similar al proporcionado de 2 bits, controlando dicho incremento y decremento con los botones de arriba y debajo de la placa en este caso, y mostrando el valor en hexadecimal por otro display de 7 segmentos de la placa Nexys3. Se realizará el diagrama de bloques y posteriormente se modificará el diseño de Xilinx ISE para incluir los nuevos bloques.

2.- Modificar el script de simulación para incluir pulsaciones de los botones de arriba y abajo y así verificar que funcionan los diferentes niveles del nuevo PWM creado, además de mostrar la salida hacia el display de 7 segmentos donde se muestra el valor de este nuevo contador, y la señal de salida del PWM de 4 bits.

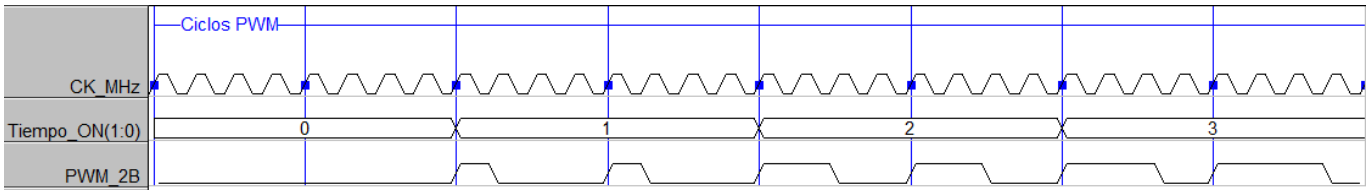
3.- El circuito se deberá presentar implementado correctamente sin errores.

4.- Se subirá a la tarea de la "Práctica 1 – 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) 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.

Trabajo a presentar por el estudiante (OPCIONAL):

5.- Realizar una segunda modificación en la práctica para conseguir algo que no se está logrando en el circuito que se ha proporcionado. Si observamos el funcionamiento del PWM de 2 bits, por ejemplo, vemos que con un "tiempo_ON(1:0)" de "0" el led no se enciende en ningún momento, dado que la señal digital de salida está todo el rato a '0' lógico, pero si ponemos el máximo valor "3" el led no lucirá a la máxima potencia porque siempre hay un ciclo de reloj que bajará a '0' la señal, dado que comparamos el valor del contador con el "tiempo_ON":

Tiempo_ON = 3 y Cuenta = 0 ➔ Tiempo_ON > Cuenta ➔ Señal PWM = 1
Tiempo_ON = 3 y Cuenta = 1 ➔ Tiempo_ON > Cuenta ➔ Señal PWM = 1
Tiempo_ON = 3 y Cuenta = 2 ➔ Tiempo_ON > Cuenta ➔ Señal PWM = 1
Tiempo_ON = 3 y Cuenta = 3 ➔ Tiempo_ON = Cuenta ➔ Señal PWM = 0



Se pide, tanto para el PWM de 2 bits proporcionado como para el de 4 bits que habéis realizado en vuestra práctica, modificar lo que sea necesario del diseño para conseguir que haya un estado más de "tiempo_ON" que permita encender completamente el led, dejando la salida del PWM a '1' lógico fijo.

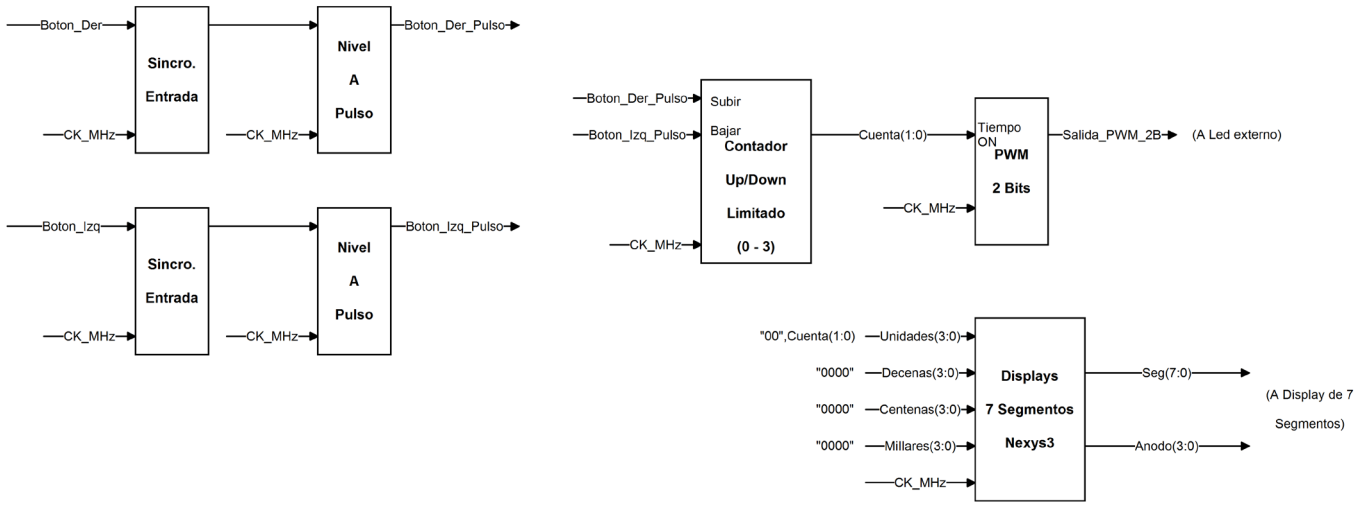
Recomendación: Será necesario añadir un bit más a "tiempo_ON" para que pudiéramos ajustar un valor superior con las pulsaciones de los botones, de manera que si, por ejemplo, el PWM es de 2 bits (cuenta de 0 a 3) podamos tener un valor de "tiempo_ON" de "4", el cual permitiría que estuviera encendido permanentemente el led ya que el contador de 2 bits nunca podría alcanzar dicho valor:

Tiempo_ON = 4 y Cuenta = 0 ➔ Tiempo_ON > Cuenta ➔ Señal PWM = 1
Tiempo_ON = 4 y Cuenta = 1 ➔ Tiempo_ON > Cuenta ➔ Señal PWM = 1
Tiempo_ON = 4 y Cuenta = 2 ➔ Tiempo_ON > Cuenta ➔ Señal PWM = 1
Tiempo_ON = 4 y Cuenta = 3 ➔ Tiempo_ON > Cuenta ➔ Señal PWM = 1

Si se realiza esta parte opcional, se subirá al campus virtual la misma información indicada para la parte obligatoria, pero **sólo se subirá el proyecto que incluya todas las mejoras**, no uno para cada parte (obligatoria y opcional).

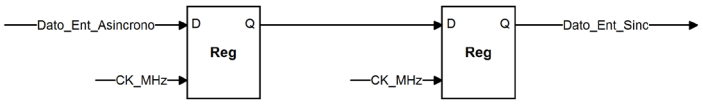
Diagrama de Bloques General:

El siguiente diagrama muestra el recorrido de los datos de entrada (pulsaciones de los botones derecho e izquierdo) desde que entran al circuito, llegan hasta el contador, y sale el valor del mismo por un display de 7 segmentos y se genera la señal PWM correspondiente. Posteriormente, detallaremos cada uno de los bloques por separado.



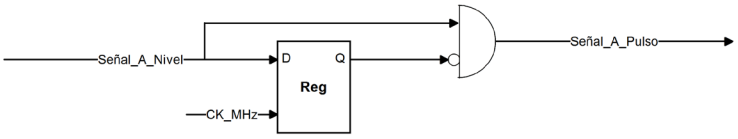
Sincronización de Entrada:

En este bloque se capturan las pulsaciones externas de los botones de la placa. Como son señales asíncronas, dado que las pulsa el usuario cuando lo desea, 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.



Nivel A Pulso:

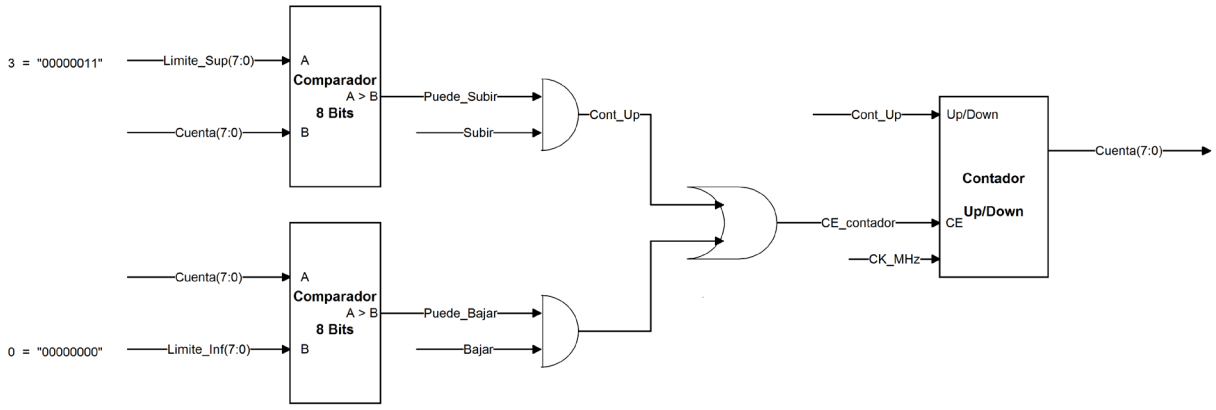
Este circuito se encarga de convertir una señal que dura varios ciclos de reloj en una señal que dura 1 solo ciclo de reloj, es decir, forma un "pulso" al cambiar dicha señal de '0' a '1' lógico.



Contador Up/Down Limitado (8 bits):

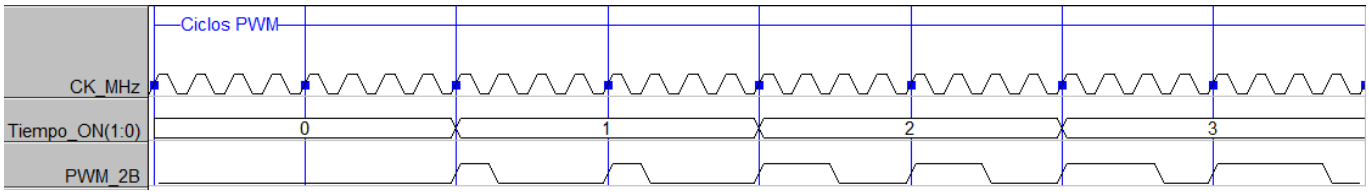
Este bloque será un contador que poseerá un par de comparadores para verificar si hemos llegado a los límites superior o inferior en todo momento; si no hemos llegado al límite superior, se podrá seguir aumentando la cuenta, y si no hemos llegado al límite inferior, se podrá seguir disminuyendo la misma. El ejemplo que hemos puesto es un contador de 8 bits generado con el CORE Generator, que posee previamente 2 comparadores que toman los límites que les hayamos puesto y los compara con la cuenta del dicho contador, realizando la operación ">"; Si la cuenta no ha llegado al límite superior permitirá seguir incrementándose, cosa que se hará cuando se active la señal "Subir" (conectada externamente al botón derecho de la placa), y si esta situación se da, se activará el Clock Enable del contador para que realice un incremento (señal Up/Down = '1'); Si la cuenta no ha llegado al límite inferior permitirá seguir decrementándose, cosa que se hará cuando se active la señal "Bajar" (conectada

externamente al botón izquierdo de la placa), y si esta situación se da, se activará el Clock Enable del contador para que realice un decremento (señal Up/Down = '0').



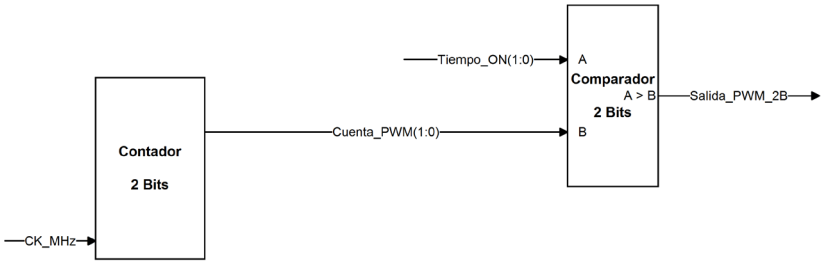
PWM 2 Bits:

Este bloque se encarga de generar la señal PWM de salida hacia el led externo. La idea sería la que se muestra en el siguiente cronograma:



La señal "Tiempo_ON(1:0)" indica el número de ciclos del PWM que permanecerá la señal a '1' lógico; al ser de 2 bits, las opciones son que la señal esté a '1' lógico de 0 a 3 ciclos como máximo de los 4 ciclos posibles. Como dicha señal se conectará al led de salida, sería como si estuviéramos apagando y encendiendo varias veces por segundo el led, y cuanto más tiempo esté a '1' lógico la señal, más se iluminará el mismo.

El diagrama de bloques de este circuito sería el siguiente:

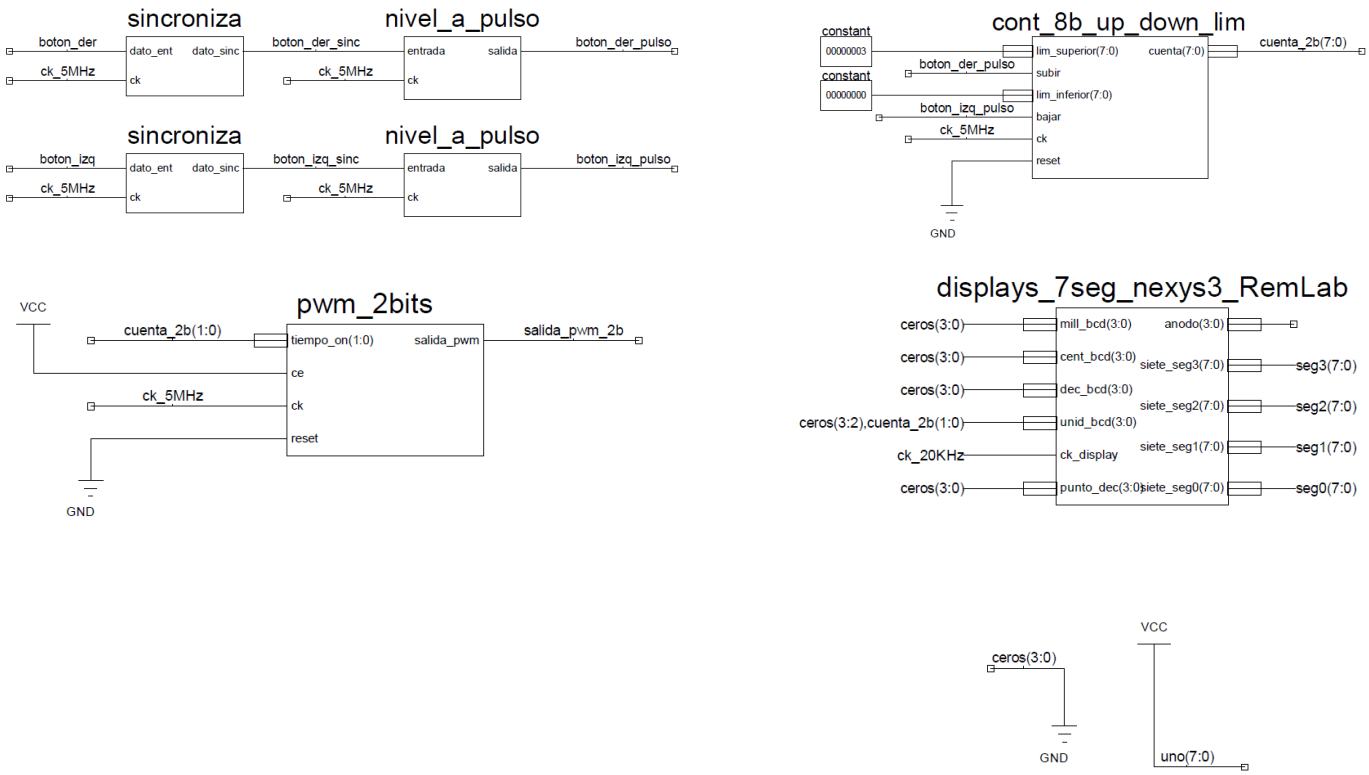


Displays 7 Segmentos Nexys3:

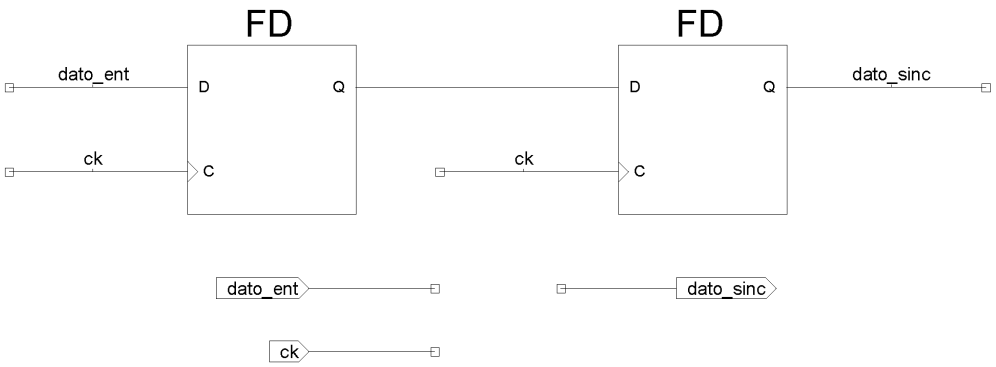
Este bloque se proporciona ya implementado, y es capaz de enviar los datos de los 4 displays de 7 segmentos de manera multiplexada a través de un único bus de salida "Seg(7:0)" y los ánodos de los correspondientes segmentos "Anodo(3:0)". Se proporciona el dato en 4 bits en hexadecimal para cada uno de los displays (4 cifras máximo).

Circuito Implementado:

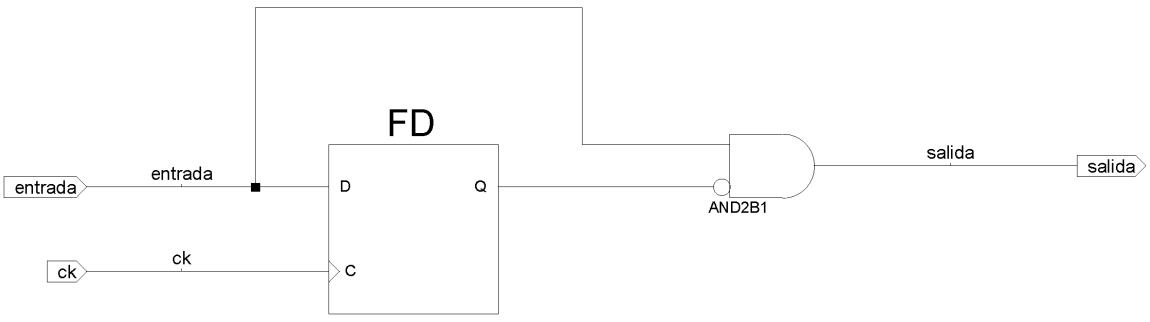
Prac1_top.sch



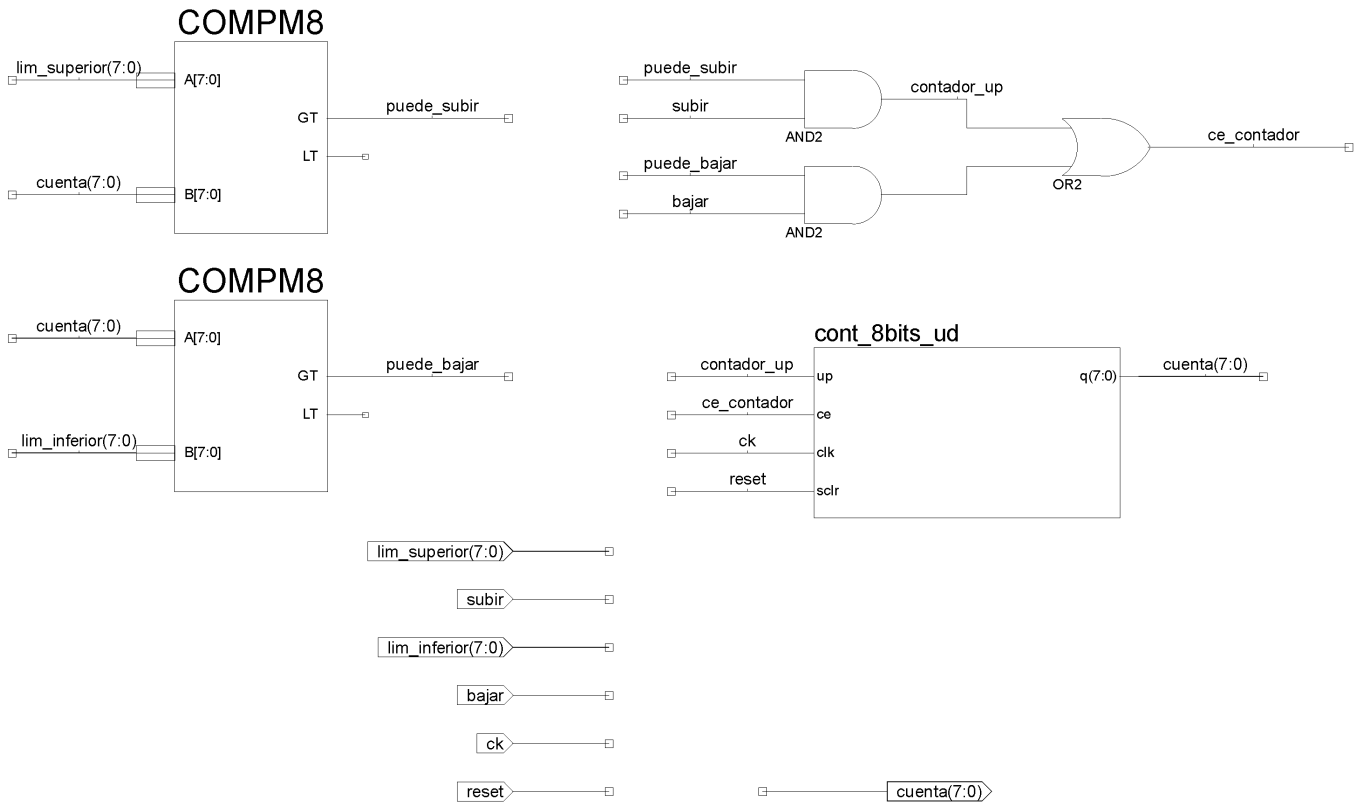
Sincroniza.sch



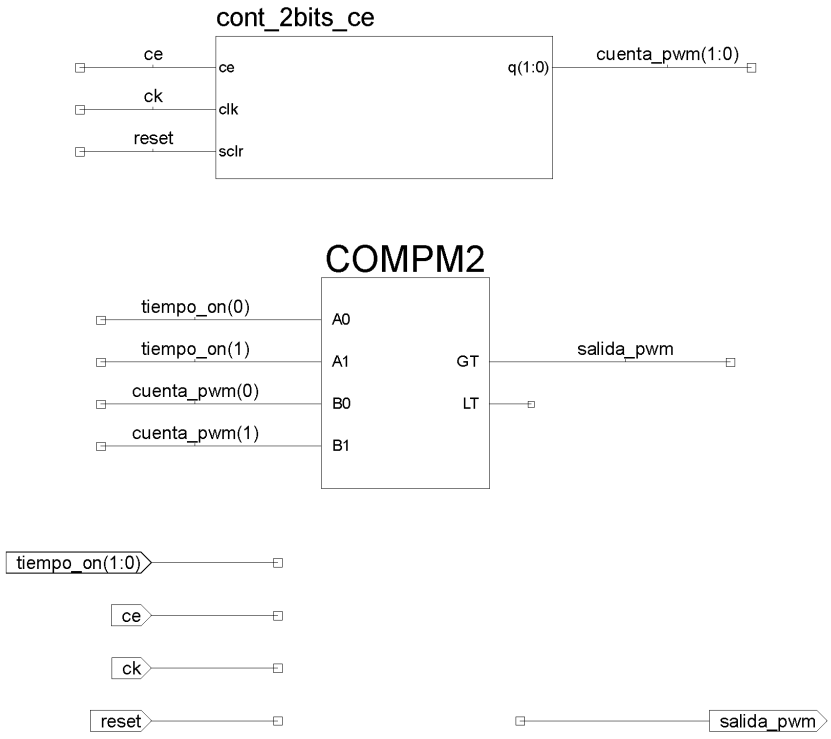
Nivel_a_pulso.sch



Cont_8b_up_down_lim.sch



PWM_2bits.sch

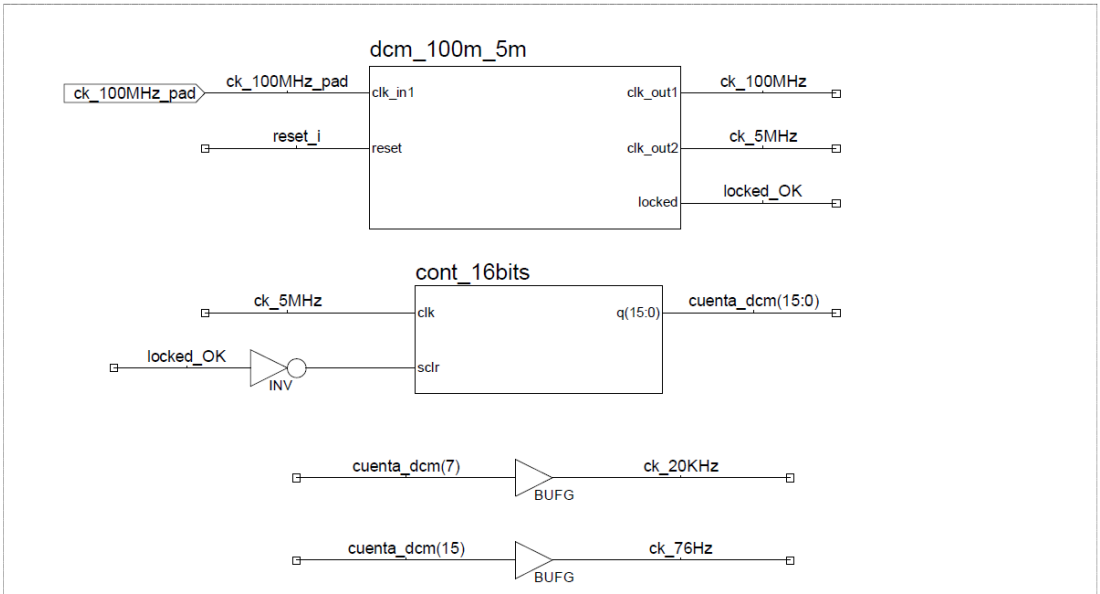


Circuitos Adicionales:

Los siguientes circuitos corresponden a lógica adicional necesaria para que funcione la FPGA. Los que no se muestran a continuación, pero están presentes en el diseño de la práctica 1, es porque no se utilizan ahora mismo y sólo aparecen para ser usados en futuras prácticas.

DCM:

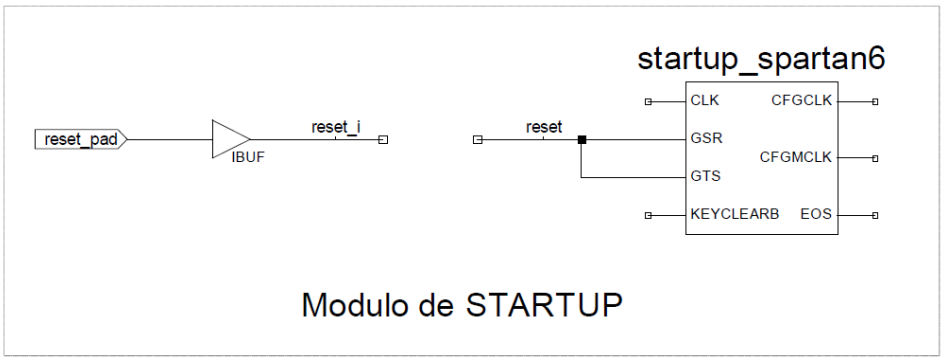
Se utiliza para sincronizar correctamente el reloj de entrada de 100 MHz, además de para generar otros relojes de menor frecuencia, como el de 5 MHz para el PWM. Dicho reloj se usa también en un contador de 16 bits ("cont_16bits") para bajar aún más la frecuencia, ya que el DCM no puede trabajar con frecuencias tan bajas, y obtener así relojes del orden de los KHz o incluso de pocos Hz, utilizados para la sincronización con los dispositivos de entrada y salida de datos de la placa (los usaremos también en futuras prácticas).



Generacion de reloj. Se usa el DCM para sincronizar el reloj de 100 MHz de la entrada, y para obtener un reloj de 5 MHz.
Se toma el bit 7 del contador de 16 bits para dividir el reloj hasta 20 KHz.
Se toma el bit 15 del contador de 16 bits para dividir el reloj hasta 76 Hz.

STARTUP:

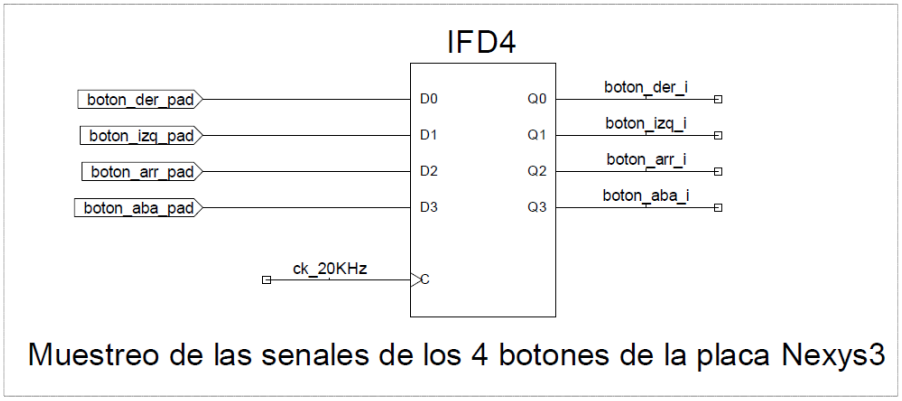
Se utiliza para inicializar a '0' ó '1' todos los biestables de la FPGA (señal GSR, Global Set Reset). La señal GTS (Global Tri-State) acciona todos los triestados de los IOB mientras se resetea el circuito. El reset del circuito se encuentra situado en el botón central de los 5 pulsadores que posee la Nexys3.



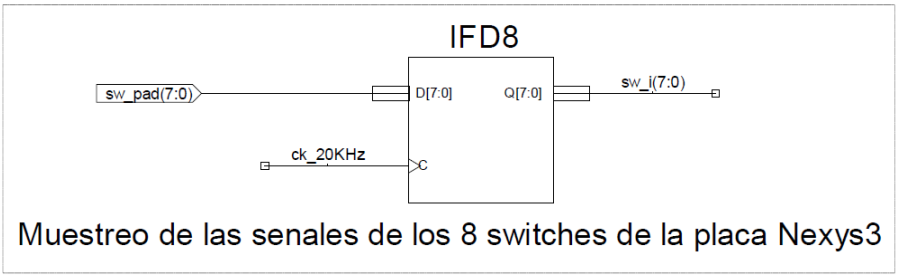
Modulo de STARTUP

Botones y switches de la placa:

Muestreamos los botones de la placa con los biestables de entrada del IOB (IFD) utilizando un reloj de muy baja frecuencia (del orden de los KHz) para evitar los rebotes de los botones, obteniendo así una señal "limpia". Realizamos el mismo proceso para los 8 switches de la placa Nexys3:



Muestreo de las senales de los 4 botones de la placa Nexys3

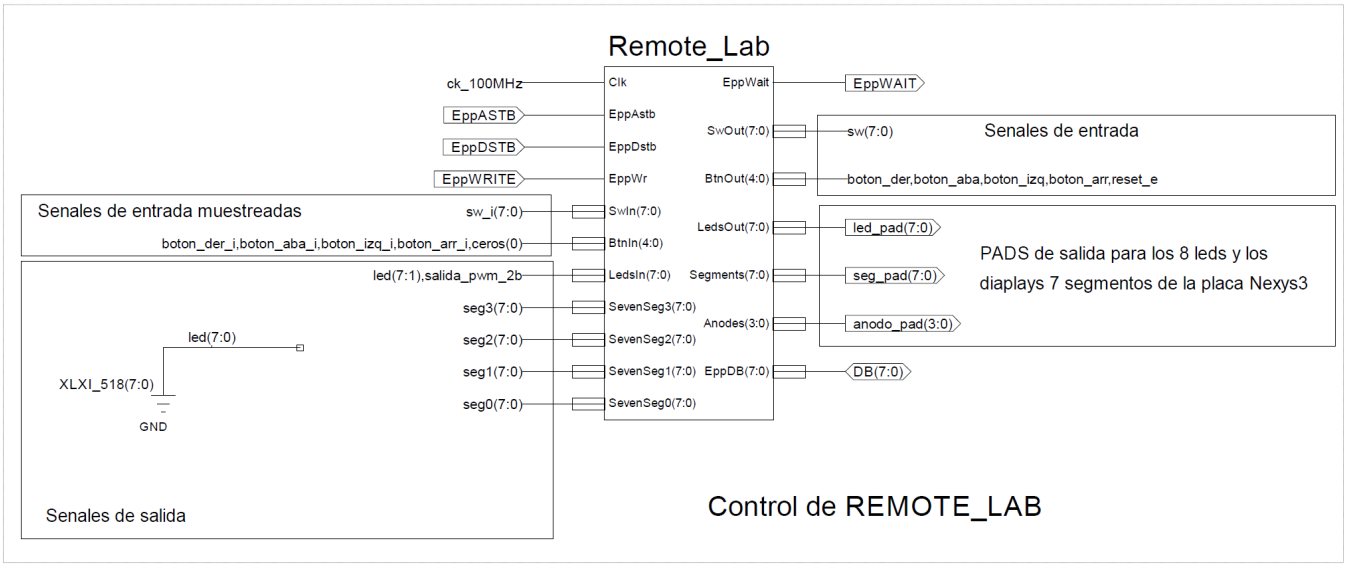


Muestreo de las senales de los 8 switches de la placa Nexys3

RemoteLab:

Al estar esta práctica preparada para utilizar el "RemoteLab", se incluye dicho módulo que es el encargado de gestionar la comunicación de salida con los leds y displays de 7 segmentos, así como las entradas correspondientes a pulsaciones de los botones y las activaciones de los switches de la placa. Los Leds de la placa los ponemos todos a '0' lógico ayudándonos del símbolo de tierra (GND), salvo el que utilizamos para enviar la salida del módulo PWM.

El funcionamiento del RemoteLab es completamente transparente de manera que, si no se utiliza, se puede probar directamente el circuito en la placa Nexys3 funcionando de manera correcta.



Simulación:

Se ha creado un script de simulación ("sim_prac1_rl.vhd") que define el comportamiento de las señales de entrada del circuito. En este caso necesitamos definir el reloj del sistema "ck_100MHz_pad", el reset del sistema "reset_pad", y los botones de la placa, fundamentalmente. Se incluyen en el script inicializaciones de otras señales que no se utilizan pero que están en el "top" del diseño para poder ser utilizadas en futuros diseños.

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_100MHz_pad <= transport '0';
            WAIT FOR 4 ns;
            ck_100MHz_pad <= transport '1';
            WAIT FOR 5 ns;
            ck_100MHz_pad <= transport '0';
            WAIT FOR 1 ns;
      END LOOP CLOCK_LOOP;
end process;
```

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, 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 200 µs, pero puedes variar según la práctica (usar un poco más de tiempo). A continuación, simulamos 4 pulsaciones del botón derecho de la placa, que en la vida real durarían varios segundos cada una pero, para acelerar la simulación, le hemos asignado 300 µs a cada una de dichas pulsaciones, lo suficiente para que se capturen con el reloj de 20 KHz y hagan el efecto oportuno. En el resultado de la simulación que se obtiene al ejecutar el script, se puede ver cómo se modifica el valor de "tiempo_ON(1:0)" conforme se dan las pulsaciones del botón derecho, cambiando la señal PWM de salida que se dirige hacia el led, y actualizándose el valor del display de 7 segmentos de las unidades de la placa Nexys3.

```
tb : PROCESS
    BEGIN
      reset_pad <= transport '1';
      EppASTB <= transport '0';     -- Entrada al módulo Remote Lab (no tocar)
      EppDSTB <= transport '0';     -- Entrada al módulo Remote Lab (no tocar)
      EppWRITE <= transport '1';    -- Entrada al módulo Remote Lab (no tocar)
      DB <= transport "00000000";   -- Entrada/Salida al módulo Remote Lab (no tocar)
      col_pad <= transport "0000";
      boton_der_pad <= transport '0';
      boton_izq_pad <= transport '0';
      boton_arr_pad <= transport '0';
      boton_aba_pad <= transport '0';
      sw_pad <= transport "00000000";
      uart_rx_pad <= transport '1';
```

```
      WAIT FOR 200 ns;

      reset_pad <= transport '0';
      WAIT FOR 200 us;

      -- Damos las pulsaciones de tecla derecha (4 veces)
      boton_der_pad <= transport '1';
      WAIT FOR 300 us;

      boton_der_pad <= transport '0';
      WAIT FOR 300 us;

      boton_der_pad <= transport '1';
      WAIT FOR 300 us;

      boton_der_pad <= transport '0';
      WAIT FOR 300 us;

      boton_der_pad <= transport '1';
      WAIT FOR 300 us;

      boton_der_pad <= transport '0';
      WAIT FOR 300 us;

      boton_der_pad <= transport '1';
      WAIT FOR 300 us;

      boton_der_pad <= transport '0';
      WAIT FOR 300 us;

      WAIT; -- will wait forever
END PROCESS;
-- *** End Test Bench - User Defined Section ***
```