# TDC_Práctica3. Diseño de circuitos secuenciales

# 3.1. Flip-Flop D (FFD)

# Testbench: Estímulo reloj

Proyecto13. Creación de un testbench para un Flip-Flop tipo D (circuito secuencial).

Ejemplo: Flip-Flop tipo D **(FFD_Basic)**

```
ARCHITECTURE behavior OF FFD_Basic_tb IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT FFD_Basic
    PORT(
         D : IN  std_logic;
         Q : OUT  std_logic;
         Clk : IN  std_logic
        );
    END COMPONENT;


   --Inputs
   signal D : std_logic := '0';
   signal Clk : std_logic := '0';

   --Outputs
   signal Q : std_logic;

   -- Clock period definitions
   constant Clk_period : time := 10 ns;
```

El reloj de Basys2 es de 50MHz, esto es, de período T=20 ns.

# Testbench: Estímulo reloj

```
BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: FFD_Basic PORT MAP (
          D => D,
          Q => Q,
          Clk => Clk
        );

   -- Clock process definitions
   Clk_process :process
   begin
      Clk <= '0';
      wait for Clk_period/2;
      Clk <= '1';
      wait for Clk_period/2;
   end process;


   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 100 ns.
      wait for 100 ns;

      wait for Clk_period*10;

      -- insert stimulus here
```

ISE construye la señal periódica de reloj, en base al período anteriormente definido.

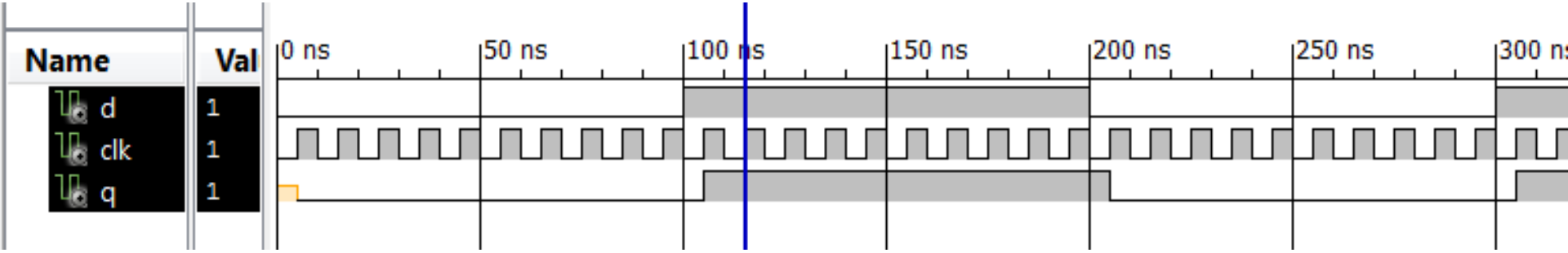
Establece una duración de tiempo basada en el período del reloj. (Eliminarla para este ejemplo)

# Testbench: Estímulo reloj

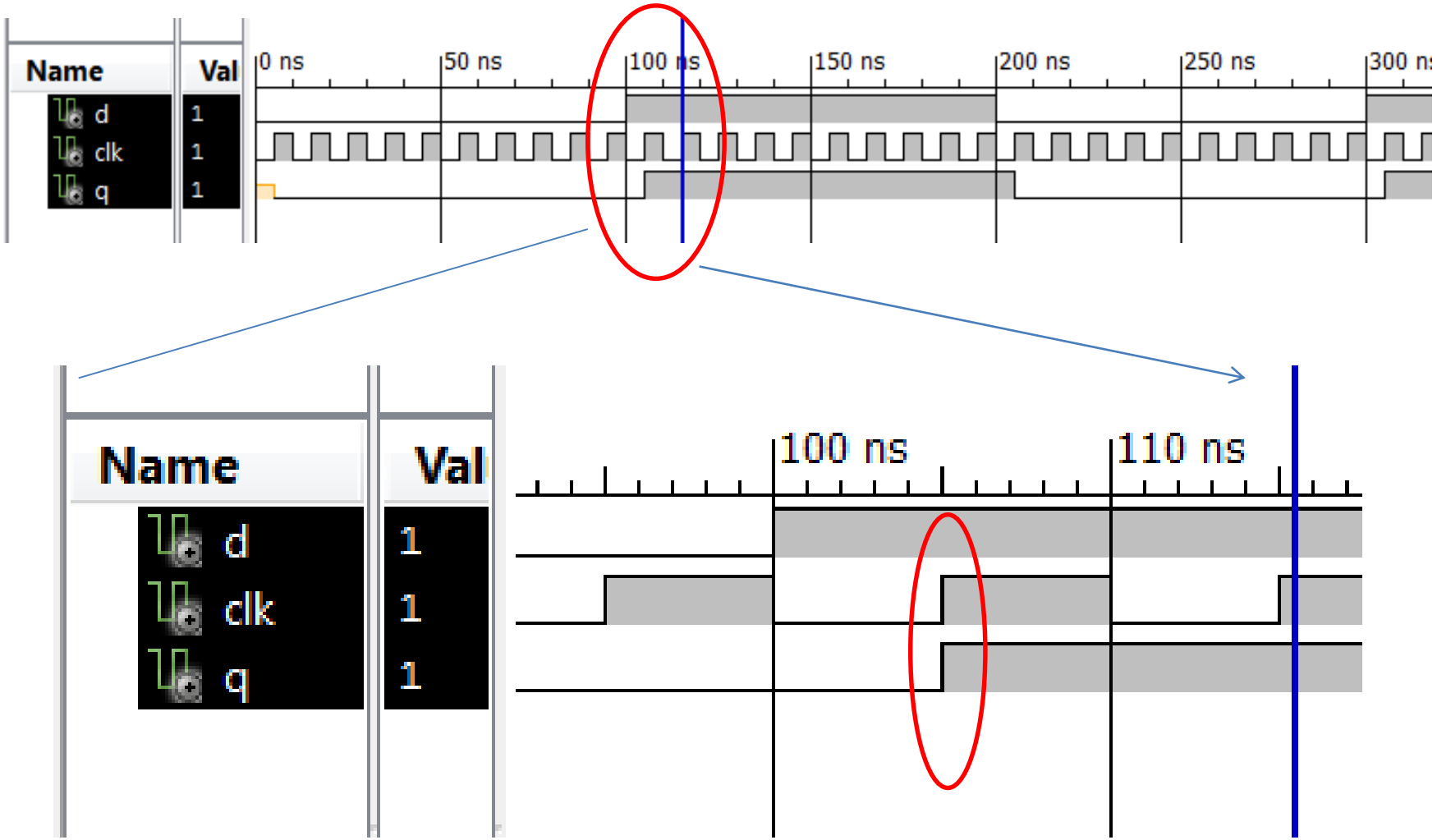
```
   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 100 ns.
      wait for 100 ns;
      -- insert stimulus here
      D<='1';
      wait for 100 ns;
      D<='0';
   end process;

END;
```

Mantiene por 100ns los valores iniciales establecidos más arriba

Cambia el valor de la única entrada (D)



# Testbench: Estímulo reloj



# Fichero de restricciones: Pin de reloj

Proyecto13. Crear un fichero de restricciones para comprobar el funcionamiento del Flip-Flop tipo D (**FFD**)

**Oscillators**

The Basys2 board includes a primary, user-settable silicon oscillator that produces 25MHz, 50MHz, or 100MHz based on the position of the clock select jumper at JP4. Initially, this jumper is not loaded and must be soldered in place. A socket for a second oscillator is provided at IC6 (the IC6 socket can accommodate any 3.3V CMOS oscillator in a half-size DIP package). The primary and secondary oscillators are connected to global clock input pins at pin B8 and pin M6 respectively.

Both clock inputs can drive the clock synthesizer DLL on the Spartan 3E, allowing for a wide range if internal frequencies, from 4 times the input frequency to any integer divisor of the input frequency.

The primary silicon oscillator is flexible and inexpensive, but it lacks the frequency stability of a crystal oscillator. Some circuits that drive a VGA monitor may realize a slight improvement in image stability by using a crystal oscillator installed in the IC6 socket. For these applications, a 25MHz (or 50MHz) crystal oscillator, available from any catalog distributor, is recommended (see for example part number SG-8002JF-PCC at www.digikey.com ).

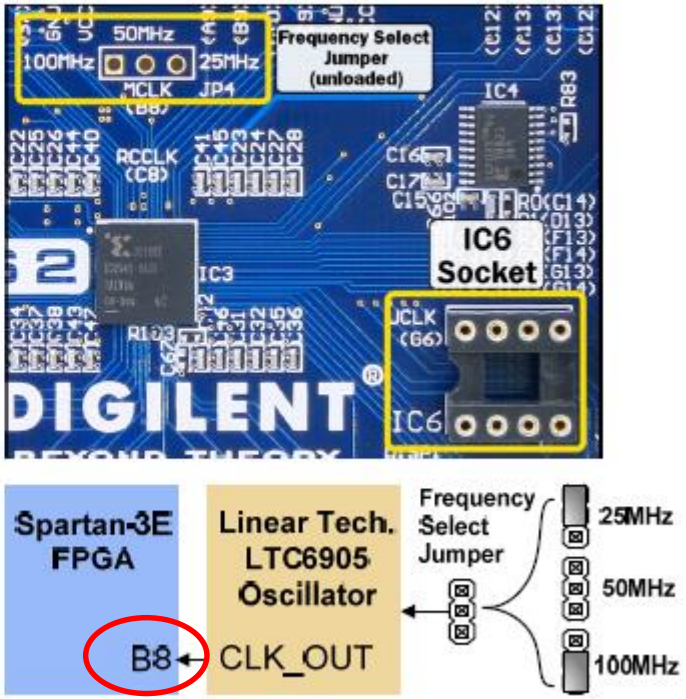



Figure 5. Basys2 oscillator circuits

# Fichero de restricciones: Pin de reloj

Proyecto13. Crear un fichero de restricciones para comprobar el funcionamiento del Flip-Flop tipo D (**FFD_ucf**)

Añadir el recurso “reloj” de la placa al diseño.

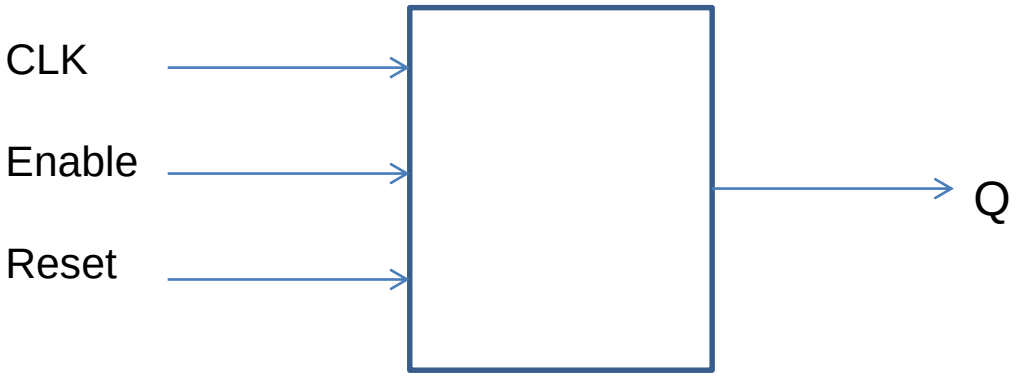
```
# clock pin for Basys2 Board
NET “CLK" LOC = "B8"; # Bank = 0, Signal name = MCLK
```

Añade tambien un pulsador como botón de **RESET** (G12), otro pulsador como entrada de control **ENABLE** (A7), y un interruptor para poder modificar la entrada **D** (P11). Por último añade un LED para visualizar **Q** (M5)

# 3.2. Circuito Conmutador (FFT)

# Circuito conmutador (FFT)

**Proyecto14.** Diseño de un circuito secuencial cuya salida (Q) se invierta en cada flanco positivo del reloj (Nombre entidad: Toggle). Incluir Reset asíncrono y Enable síncrono.




| CLK | RESET | ENABLE | Q(t) | Q(t+1) |
|---|---|---|---|---|
| ↑ | 0 | 1 | 0 | 1 |
| ↑ | 0 | 1 | 1 | 0 |
| ↑ | 1 | X | X | 0 |

# Circuito conmutador (FFT)

**Proyecto14.** Diseño de circuito secuencial cuya salida (Q) se invierta en cada flanco de sincronizacion del reloj .

```
architecture Behavioral of Toggle is
  signal Q_signal: std_logic; -- Dummy signal to read Q
begin
  process (CLK,RESET)
  begin
    if RESET='1' then
      Q_signal <= '0';
    elsif rising_edge(CLK) then
      if ENABLE='1' then
        Q_signal <= not Q_signal; -- Read the output "Q"
      end if; -- If '0' keeps the previous value
    end if;
  end process;

  Q <= Q_signal;

end Behavioral;
```

# Circuito conmutador (FFT)

1. Comprobar en Basys2 el funcionamiento del Proyecto14. Usa un LED como salida Q y sendos pulsadores para RESET y ENABLE

2. ¿Es correcto el funcionamiento? ¿Qué crees que sucede?

3. Crea un “testbench” y evalúa el funcionamiento del circuito mediante el simulador Isim. ¿Es correcto su funcionamiento?

# Circuito conmutador (FFT)

**Proyecto14.** Resultado obtenido en la comprobación en placa y en la simulación

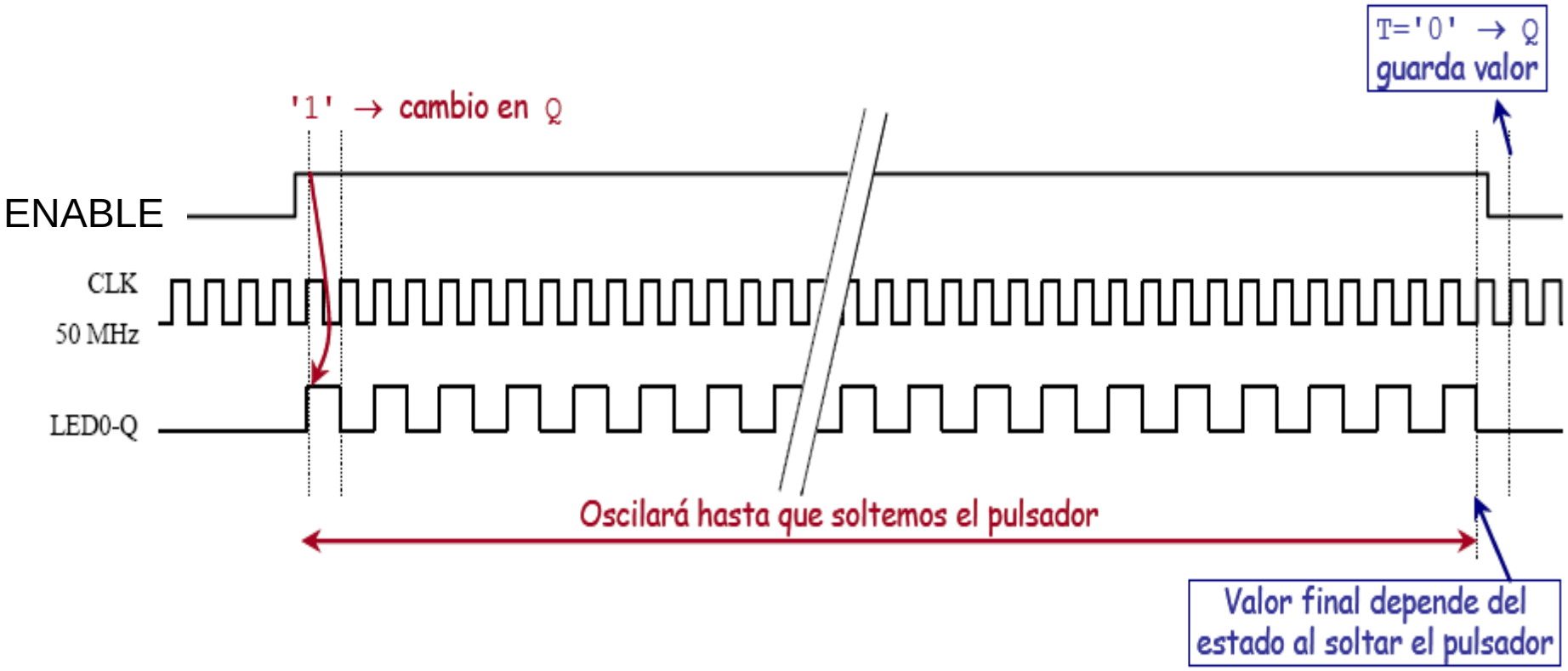


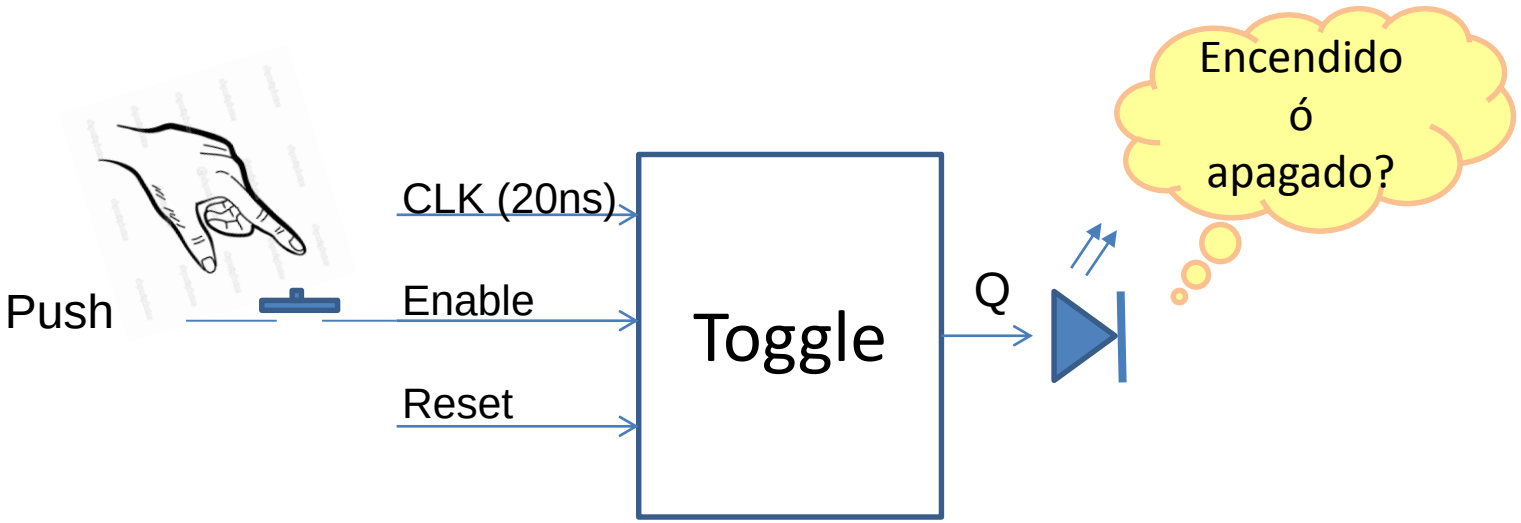

*Figura 5.4: Cronograma resultante de pulsar el botón de encendido/apagado* [1]

# Circuito conmutador (FFT)

**¿Serviría este circuito para Encender/Apagar un LED con un solo pulsador?**
**A tener en cuenta:**

1. ¿Cuánto tiempo se mantiene pulsado Push? Bastante más de 20 ns que es lo que dura un ciclo de CLK.

2. ¿Cuántos flancos positivos de CLK han tenido lugar durante ese tiempo? En cada uno de los flancos ocurridos habrá tenido lugar una conmutación de la salida.



# 3.3. Circuito detector de flancos positivos (RisingEdge)

# Circuito detector de flancos

**Proyecto15.** Diseño de circuito secuencial que detecte una transición de “0” a “1” en la entrada (Push) . Cuando se detecte el flanco positivo la salida(Pulse) tomará el valor “1”.

Nombre del Proyecto: **Proyecto15**

**Nombre del fichero y de la entidad VHDL:** RisingEdge

- Entradas: Reset, Push, Clk
- Salidas: Pulse

Flanco positivo

| Push en t-1 | Push en t | Salida Pulse |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

# Circuito detector de flancos

```
architecture Behavioral of RisingEdge is
 signal RegisteredPush: std_logic; -- To avoid async. input
 signal PreviousPush: std_logic; -- To store "Push"
Begin
------------------------------------------------- -- ------
-- SEQUENTIAL CIRCUIT
------------------------------------------------- --
-- Syncronizes the input “Push”
SincPush:process (CLK,Reset)
 begin
       if Reset='1' then
             RegisteredPush<='0';
       elsif rising_edge(CLK) then
             RegisteredPush<= Push;
       end if;
end process;
```

# Circuito detector de flancos

```
-- Stores "PreviousPush" (FFD)
StorePrevPush:process (CLK,Reset)
begin
	if Reset='1' then
		PreviousPush<='0';
	elsif rising_edge(CLK) then
		PreviousPush<=RegisteredPush;
	end if;
end process;

-------------------------------------------------
-- COMBINATIONAL CIRCUIT
-------------------------------------------------
-- Compares Push and PreviousPush

Pulse<='1' when PreviousPush='0' and RegisteredPush='1'
	else '0';
end Behavioral;
```

# Encendido/Apagado de un LED con un solo pulsador

4. Realiza la simulación del Proyecto15 y comprueba en la onda obtenida cuantos ciclos de retardo tienen lugar antes de que cambie el estado de la salida “Pulse”. ¿A qué se debe este retardo?

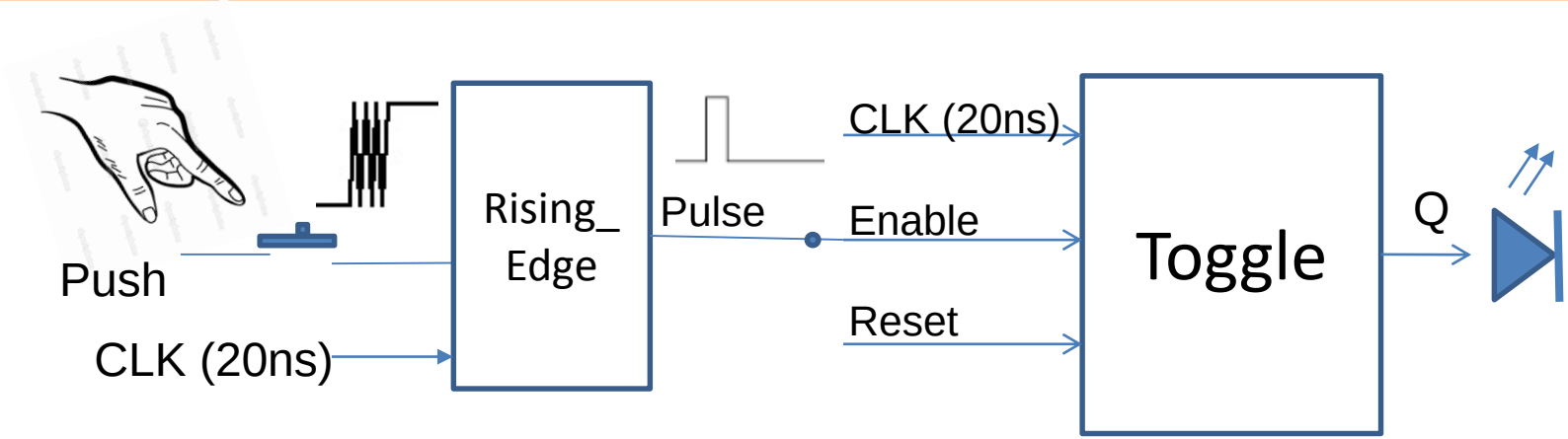
5. Comprueba ahora con Basys2 el funcionamiento del circuito.

# Encendido/Apagado de un LED con un solo pulsador

6. Combina el módulo "*Detector de Flanco Positivo*" del Proyecto15 con el "*Conmutador*" del Proyecto14 para conseguir encender y apagar un LED con un solo pulsador.

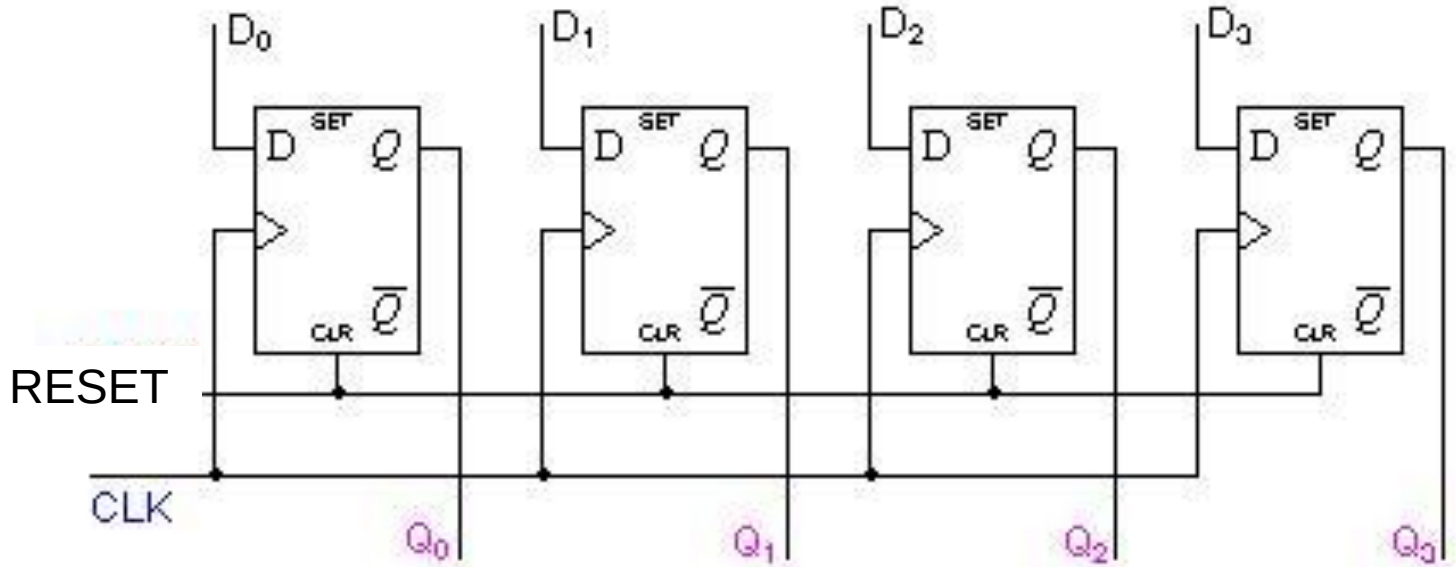
- Nombre del Proyecto: Proyecto16
- Nombre del Módulo: ON_OFF
- Entradas: Buttom, CLK, Reset
- Salidas: Toggle_Out

7. Comprueba que el circuito funciona ahora correctamente. Obtén el esquema RTL para ver el diagrama de bloques del diseño.



# 3.4. Registros

# Registros




8. Basándote en la descripción VHDL del FFD, diseña un registro de 4 bits.
   - Nombre del Proyecto: **Proyecto17**
   - **Nombre del fichero y de la entidad VHDL:** Reg_4bits
   - Entradas: RESET, DataIn, CLK, ENABLE
   - Salidas: DataOut

9. Añade un segundo módulo al **Proyecto17** que describa un registro de 3 bits (Reg_3bits).

10. Añade un tercer módulo al **Proyecto17** que describa un registro de 10 bits (Reg_10bits).

# 3.5. Contadores

# Contadores

11. Crear el **Proyecto18** y añadir un módulo VHDL que describa un contador síncrono de 2 bits (Counter_2bits).
12. Diseñar un módulo de *testbench* para el contador binario de 2 bits y simula para ver la secuencia completa de los valores de salida (Counter_2bits_tb)
13. Añadir un fichero de restricciones para comprobar el funcionamiento del contador en la placa de desarrollo. (Counter_2bits_ucf). Usa un pulsador para Enable y dos LEDs para ver la salida binaria.
14. ¿Funciona adecuadamente el circuito contador en la placa? ¿Por qué?

# Contadores

15. Crear el **Proyecto19** y añadir un módulo VHDL que describa un circuito que incremente una cuenta de 0 a 3 cada vez que se actúa sobre un pulsador.

    Nombre del Proyecto: Proyecto19

    Nombre del Módulo: YourTurn

    Entradas: CLK, Reset, Push_Inc

    Salidas: Count

Nota: Reutiliza el circuito detector de flancos para conseguir un correcto funcionamiento del pulsador. Tambien puedes reutilizar el contador del Proyecto 18.

# 3.6. Divisor de frecuencia

# Divisor de frecuencia

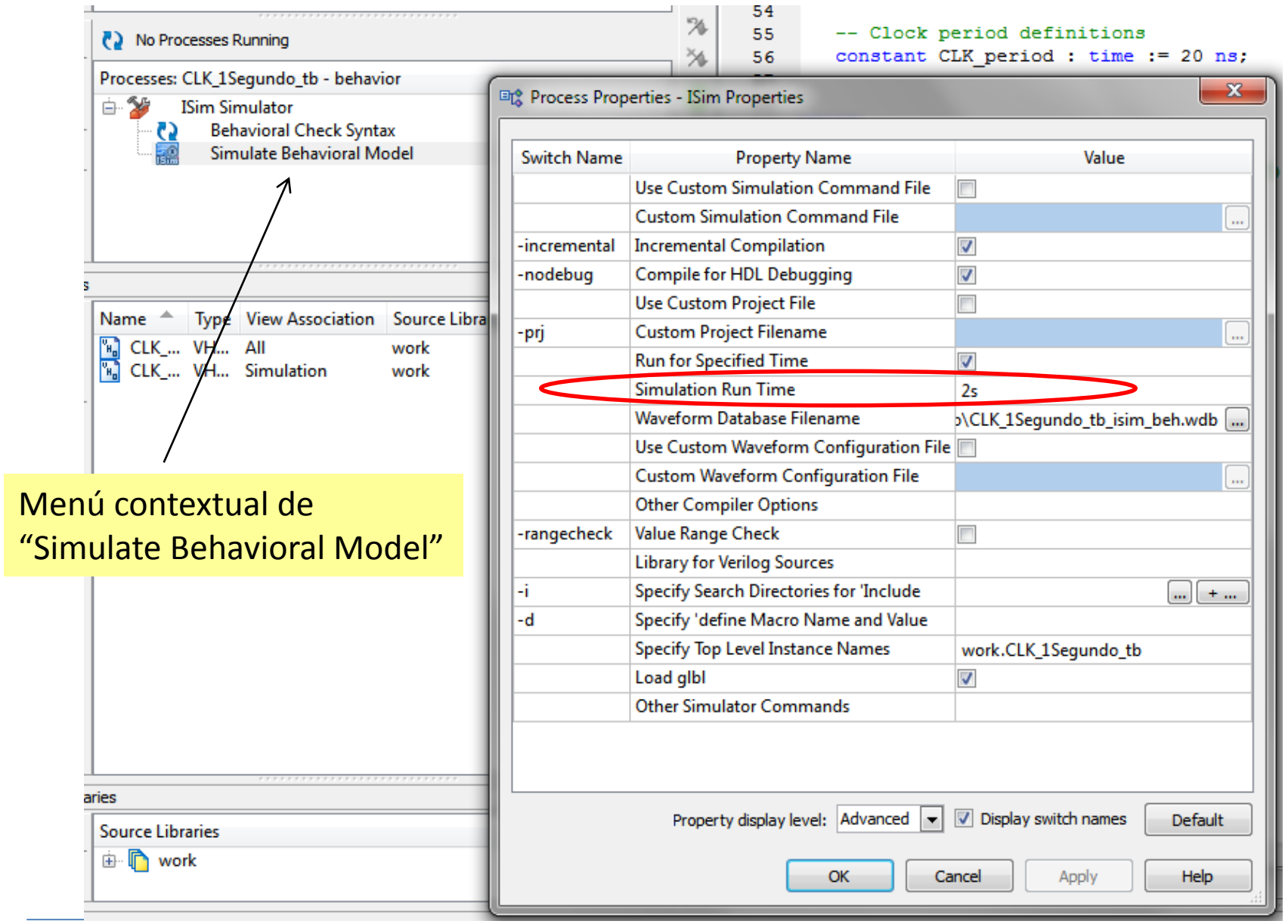
16. Crea un nuevo proyecto que incluya un módulo VHDL de un divisor de frecuencia, de modo que a partir de los 50MHz del reloj de la placa, devuelva un pulso de 20ns cada 1 segundo (1 Hz). (Hecho en clase)

    - Nombre del Proyecto: Proyecto20
    - Nombre del Módulo: CLK_1Hz
    - Entradas: CLK, Reset
    - Salidas: Out_1Hz

17. Comprueba mediante simulación su correcto funcionamiento.
18. Tenga en cuenta que el tiempo de simulación establecido por defecto por ISIM es de 1000ns, insuficiente para comprobar el funcionamiento del reloj de 1 segundo. Para modificar el tiempo de simulación deberá modificar sus propiedades.

    (Ver siguiente transparencia)

# Divisor de frecuencia




Menú contextual de "Simulate Behavioral Model"

# Divisor de frecuencia

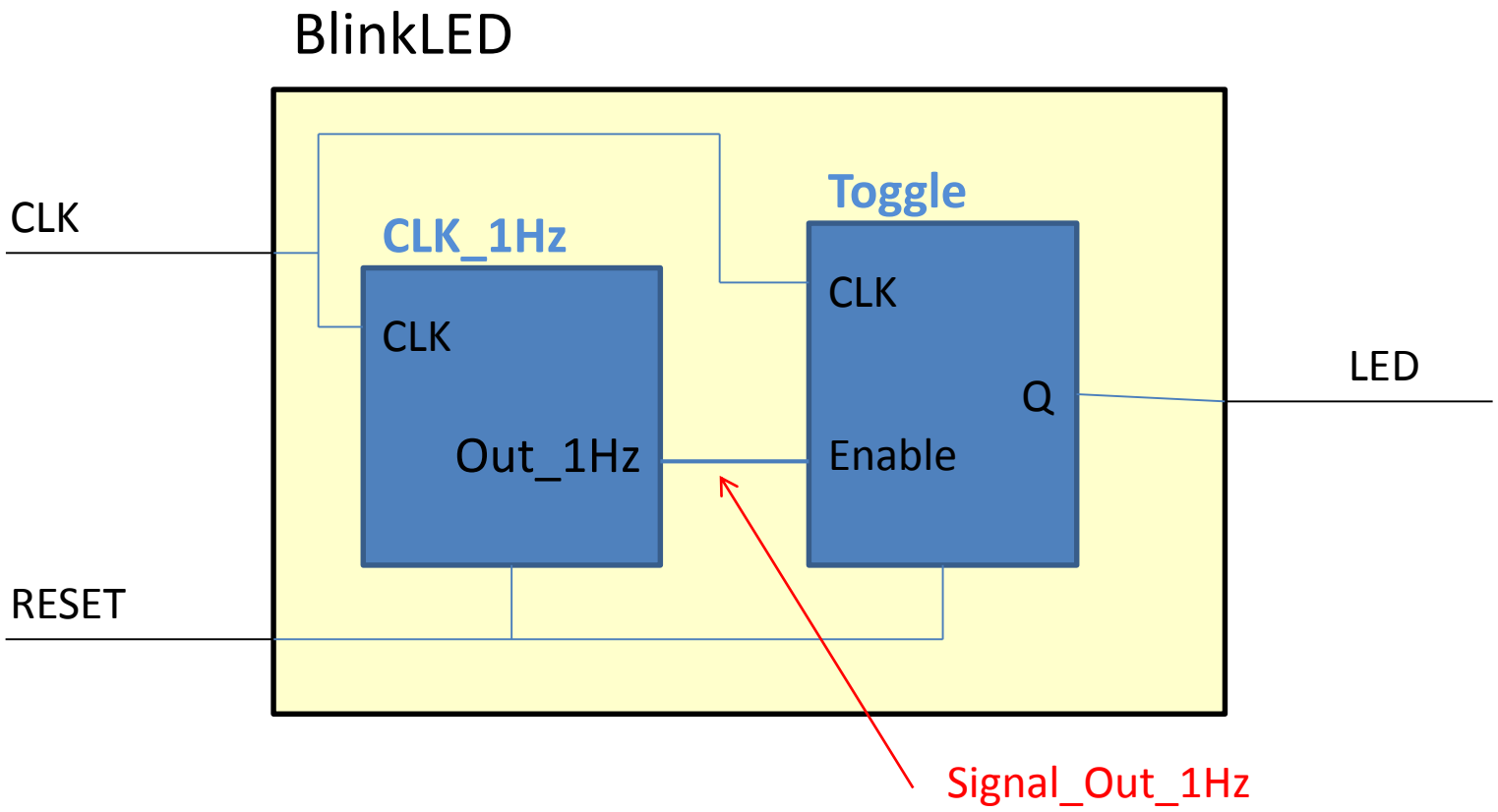
19. Añade un nuevo módulo VHDL al proyecto 20 que describa un nuevo divisor de frecuencia. El nuevo módulo debe, a partir de los 50MHz del reloj de la placa, devolver un pulso de 20ns cada 1 milisegundo (1 KHz).

    - Nombre del Proyecto: Proyecto20
    - Nombre del Módulo: CLK_1KHz
    - Entradas: CLK, Reset
    - Salidas: Out_1KHz

20. Comprueba mediante simulación su correcto funcionamiento.

# Divisor de frecuencia: Aplicación 1

**22. Proyecto21.** Usando el modulo de reloj de frecuencia 1 Hz, crea un nuevo diseño (BlinkLED) que haga parpadear un LED una vez por segundo.



# Divisor de frecuencia: Aplicación 2

**Proyecto21.** Añade un tercer módulo al proyecto denominado BlinkLED_1Process.

Modifica el módulo BlinkLED_2Process explicado en clase para que solo tenga un proceso. El objetivo será conseguir una señal de reloj de 1Hz simétrica que pudiera aplicarse a cualquier sistema. Para este caso, aplique dicha señal directamente a un LED. El resultado deberá ser que un LED de la placa parpadea cambiando de estado cada 1 segundo.

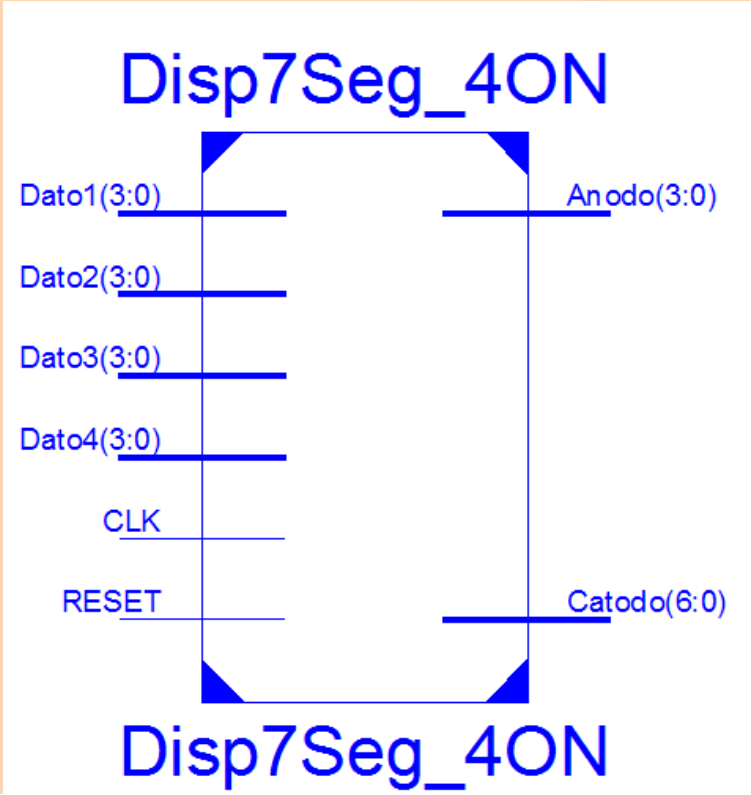
Comprueba su correcto funcionamiento en una nueva placa denominada Nexys3.(Manual en Campus Virtual)

# Divisor de frecuencia: Aplicación 2

**21. Proyecto22.** Diseño de un circuito que permita visualizar 4 datos diferentes en los cuatro displays de 7 segmentos disponibles en la Basys2.

Nombre del Proyecto: Proyecto22

Nombre del Módulo: Display7Seg_4ON



# Divisor de frecuencia: Aplicación 2

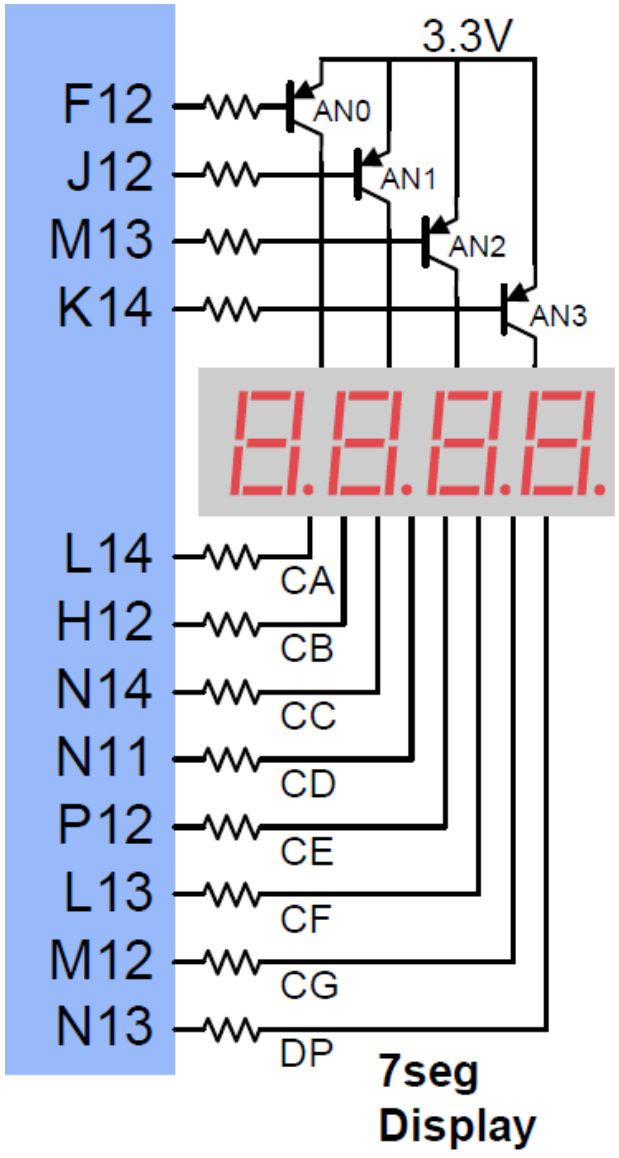
## Proyecto22

Consideraciones:

- Será un diseño estructural
- Deberá reutilizar varios módulos ya diseñados
- El divisor de frecuencia deberá proporcionar una f=1KHz para controlar la línea Enable de un contador. (Utilizar el CLK_1KHz del Proyecto20)
- Empiece por dibujar un diagrama de bloques que incluya los componentes y las señales de interconexión.

Nota: Este circuito "se aprovecha" de la dificultad del ojo humano para ver sucesos extremadamente rápidos. Debemos tomar este camino porque los 4 displays se gestionan con las mismas líneas de cátodos, y por tanto el circuito lo que hará es encender cada display durante 1ms, el ojo percibirá una imagen estática del valor que se envíe a ese display.

# Divisor de frecuencia: Aplicación 2

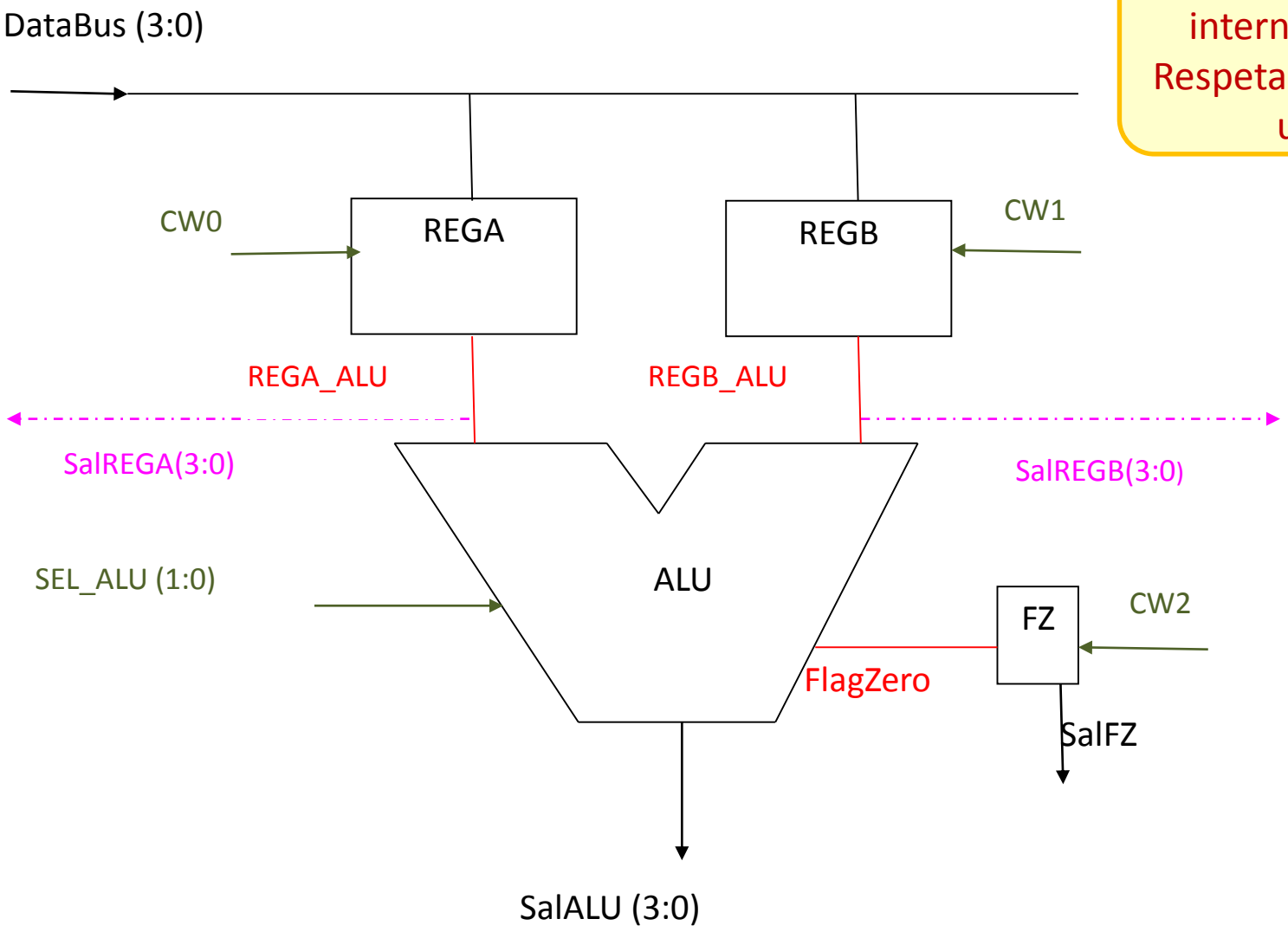


# 3.7. Camino de datos

# Camino de datos

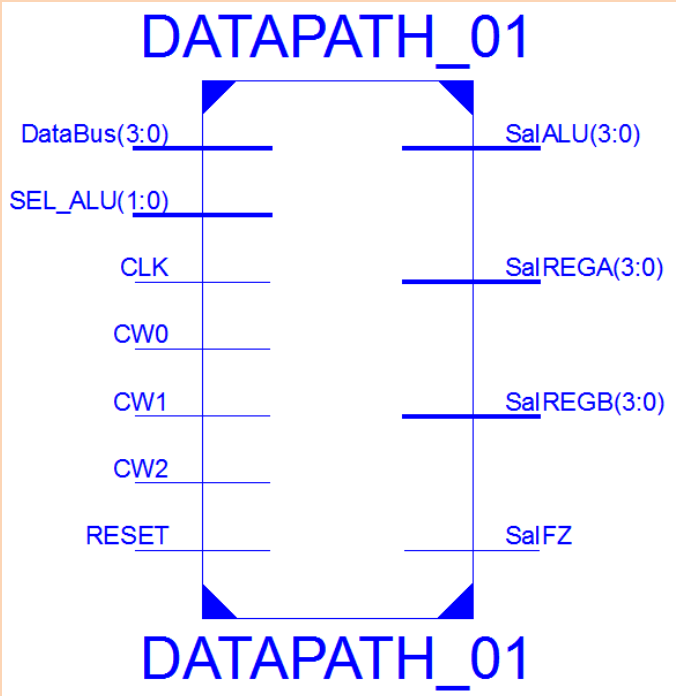
**Proyecto23.** Camino de datos de 4-bits (DataPath_01)

En rojo las señales internas a definir. Respetar los nombres usados



# Camino de datos

22. Realiza el diseño del camino de datos del esquema anterior. Usa descripción estructural reutilizando los módulos ya diseñados
    - Nombre del Proyecto: **Proyecto23**
    - Nombre del fichero y de la entidad VHDL**:** DataPath_01
    - Módulos a reutilizar: REG_4, ALU_4bits,FFD
    - Entradas y Salidas: Ver diagrama de bloques de la entidad.



# Camino de datos

23. Comprueba el funcionamiento de DataPath_01 en Basys2. Para ello añade al Proyecto23 otro fichero denominado "Top01". Este nuevo módulo debe incluir al módulo "DataPath_01" y el módulo "Display7Seg_4ON" del Proyecto22.

Añade un fichero de restricciones con la siguiente correspondencia de recursos de Basys2:

- Pulsador BTN0 para el Reset
- Pulsadores BTN3-BTN1 para las lineas CW2-CW0.
- Interruptores Sw7-Sw4 para DataBus
- Interruptores Sw1-Sw0 para SEL_ALU
- Displays 2-0 para SalREGA, SalREGB y SalALU
- Led LD0 con SalFZ

# Bibliografía del tema

[1] Introducing the Spartan3E FPGA and VHDL (Ch.11)
http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Dev/FPGA/IntroToSpartanFPGABook.pdf

[2] Free Range VHDL (Ch.5)
http://www.freerangefactory.org/dl/free_range_vhdl.pdf

[3] Diseño de circuitos digitales con VHDL (Ch.5)
http://eciencia.urjc.es/handle/10115/4045