INGENIERÍA DE COMPUTADORES 3

Solución al examen de Junio 2017, Segunda Semana

PREGUNTA 1 (2 puntos)

Tomando como base el siguiente código VHDL, dibuje el cronograma de evolución de las señales `y1`, `y2`, `y3` e `y4` entre los instantes 0 y 400 ns.

```
library IEEE;
use IEEE.std_logic_1164.all;

entity crono2 is
end entity crono2;

architecture crono2 of crono2 is
 signal y1, y2, y3, y4: std_logic;
begin
    y1 <= '0',
          '1' after 100 ns;
    y2 <= '1',
          '0' after 100 ns,
          '1' after 200 ns,
          '0' after 250 ns;
    y3 <= y2 after 100 ns;
    y4 <= transport y2 after 100 ns;
  Proc1: process
  begin
    y1 <= '0';
    wait for 50 ns;
    y1 <= '0';
  end process;
end architecture crono2;
```

Solución a la Pregunta 1

En la Figura 1.1 se muestra el cronograma de evolución de las señales.

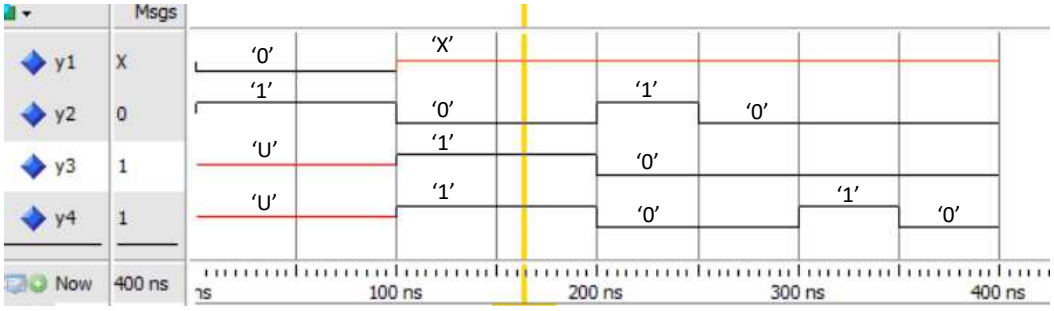
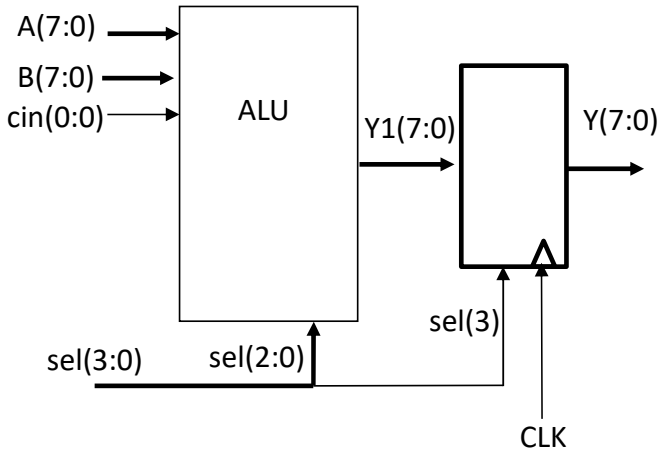


Figura 1.1: Cronograma de evolución de las señales.

PREGUNTA 2 (3 puntos)

A continuación, se muestra el circuito, la tabla de operaciones y la **entity** de una ALU seguida por un registro que opera en el flanco de subida de la señal de reloj.



Unidad	Sel	Operación
ALU	-000	A+B
	-001	A+B+cin
	-010	B-1
	-011	A+1
	-100	A and B
	-101	A or B
	-110	A xor B
	-111	not B
Registro	0---	Carga
	1---	Desplaza

```
entity ALUReg is
   port( Y    : out std_logic_vector ( 7 downto 0);
         A, B : in std_logic_vector (7 downto 0);
         sel  : in std_logic_vector ( 3 downto 0);
         cin  : in std_logic_vector(0 downto 0);
```

```
         CLK  : in std_logic);
end entity ALUReg;
```

La ALU realiza operaciones sobre dos operandos de 8 bits, denominados `A` y `B`. La operación que realiza la ALU se especifica con los tres bits menos significativos de la señal `sel`. La salida de la ALU (señal `Y1`) es la entrada de un registro. El registro realiza las dos siguientes operaciones que se especifican con `sel(3)`:

- Cuando la señal `sel(3)` tiene el valor '0' se carga en el registro la señal `Y1` en el flanco de subida de la señal de reloj.
- Cuando la señal `sel(3)` tiene el valor '1' se desplaza el contenido del registro 1 bit a la derecha y se introduce un cero por la izquierda.

Escriba en VHDL la **architecture** que describe el comportamiento del circuito, empleando para ello una sentencia de asignación concurrente de selección (**with–select**) para describir el comportamiento de la ALU y un bloque **process** que describa el comportamiento del registro. Asimismo, en el diseño únicamente pueden emplearse los dos siguientes paquetes de la librería IEEE:

```
IEEE.std_logic_1164
IEEE.numeric_std
```

Solución a la Pregunta 2

El diseño de la ALU seguida del registro se muestra en Código VHDL 1.1.

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
entity ALUReg is
    port( Y      : out std_logic_vector ( 7 downto 0);
          A, B   : in std_logic_vector (7 downto 0);
          sel    : in std_logic_vector ( 3 downto 0);
          cin    : in std_logic_vector(0 downto 0);
          CLK    : in std_logic);
end entity ALUReg;
architecture ALUReg of ALUReg is
 constant WIDTH      :   integer := 8;
                              -- Núm. bits de los operandos
 constant SEL_BITS   :   integer := 4;
                              -- Núm. bits selección de operación
 signal Y1, Reg: std_logic_vector(WIDTH-1 downto 0);
begin
  --ALU
 with sel(SEL_BITS-2 downto 0) select
  Y1 <=   std_logic_vector(signed(A)+signed(B)) when "000",
          std_logic_vector(signed(A)+signed(B)+signed("000"&cin)) when "001",
          std_logic_vector(signed(B)-1) when "010",
          std_logic_vector(signed(A)+1) when "011",
          (A and B) when "100",
          (A or B) when "101",
          (A xor B)   when "110",
          (not B) when others;
 Registro:process (CLK)
 begin
   if rising_edge(CLK ) then
      if (sel(SEL_BITS-1)= '0') then
           Reg <= Y1;
      elsif (sel(SEL_BITS-1) = '1') then
           Reg<='0'&Reg(WIDTH-1 downto 1);
      end if;
   end if;
 end process Registro;
 Y <= Reg;
end architecture ALUReg;
-----------------------------------
```

Código VHDL 1.1: Diseño que describe el comportamiento de la ALU seguida del registro.

PREGUNTA 3 (2 puntos)

Programe en VHDL un banco de pruebas para el circuito que ha diseñado al contestar a la Pregunta 2. La señal de reloj (`CLK`) debe tener un periodo de 100 ns e inicialmente valer '0'. El programa de test debe realizar las acciones siguientes:

1. *Dar valores a las señales* `A`, `B` *y* `cin` .

   `A` ha de valer "00000000". `B` ha de valer "10000000". `cin` ha de valer "0".

2. *Realizar la operación A or B y cargar el resultado en el registro*.

3. *Realizar tres operaciones de desplazamiento del registro*.

4. *Realizar la operación A xor B y cargar el resultado en el registro*.

El correcto funcionamiento del circuito debe comprobarse mediante inspección visual. No es necesario que el banco de prueba compruebe que las salidas de la UUT son las esperadas. Dibuje el cronograma de evolución que han de seguir las señales de entrada y salida de la UUT.

Solución a la Pregunta 3

El código VHDL del banco de pruebas se muestra en Código VHDL 1.2.

En la Figura 1.2 se muestra el cronograma de evolución de las señales.

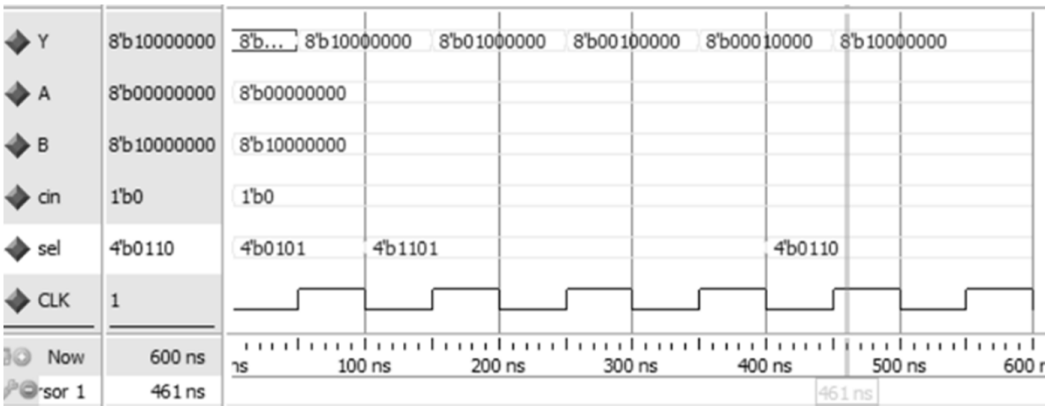


Figura 1.2: Cronograma de evolución de las señales correspondientes al banco de prueba.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity bp_ALUReg is
end entity bp_ALUReg;
architecture bp_ALUReg of bp_ALUReg is
   constant PERIODO    : time          := 100 ns; -- Reloj
   signal     Y          : std_logic_vector(7 downto 0);-- Salidas UUT
   signal     A, B       : std_logic_vector(7 downto 0);-- Entradas UUT
   signal cin: std_logic_vector(0 downto 0);
   signal sel: std_logic_vector(3 downto 0);
   signal     CLK        : std_logic   := '0';      -- Entradas UUT

   component ALUReg is
     port( Y       : out std_logic_vector ( 7 downto 0);
           A, B  : in std_logic_vector (7 downto 0);
           sel    : in std_logic_vector ( 3 downto 0);
           cin    : in std_logic_vector(0 downto 0);
           CLK    : in std_logic);
   end component ALUReg;
begin
   -- Instanciar y conectar UUT
   uut : component ALUReg port map
                      (Y, A, B, sel, cin, CLK);
   clk <= not clk after (PERIODO/2);
   A <= "00000000";
   B <= "10000000";
   cin <= "0";
   gen_vec_test : process is
      variable error_count : integer := 0; -- Núm. errores
   begin
      report "Comienza la simulación";
       -- Vectores de test y comprobación del resultado
         sel <= "0101";   --Operación A or B y carga del resultado en el registro
         wait for PERIODO;     -- 1
         sel<="1101";     -- Operacion desplazamiento
         wait for 3*PERIODO;     -- 2
         sel <="0110";--Operación Ax or B y carga del resultado en el registro
         wait for PERIODO;
         wait;        -- Final del bloque process
   end process gen_vec_test;
end architecture bp_ALUReg;
```

Código VHDL 1.2: Diseño del banco de pruebas de la ALU.

PREGUNTA 4 (3 puntos)

Diseñe un circuito secuencial síncrono capaz de detectar cuando le llega la secuencia de bits “0101” por su entrada serie de un bit. El circuito no detecta secuencias solapadas. Por ejemplo, en la secuencia “01010111” se detectaría una única vez la secuencia “0101”. La **entity** del circuito se muestra a continuación.

```
entity detector is
   port( Y     : out std_logic;
         state : out std_logic_vector(2 downto 0);
         X     : in std_logic;
         reset : in std_logic;
         clk   : in std_logic);
end entity detector;
```

El circuito tiene una señal de reloj (`clk`), un entrada serie de un bit (`X`), una señal de reset asíncrona activa en ‘1’ (`reset`), una señal que indica el estado en que se encuentra el circuito (`state`) y una señal de salida de un bit (`Y`).

La señal `Y` se pone a ‘1’ cuando se detecta la secuencia “0101” sin solapamientos.

La señal `reset` pone el circuito en su estado inicial. Todos los cambios tienen lugar en el flanco de subida de la señal de reloj.

Escriba en VHDL la **architecture** que describe el comportamiento del circuito en términos de una máquina de Moore. Dibuje el diagrama de estados correspondiente al circuito que ha diseñado indicando la valor de la señal de salida `Y` en cada estado.

Solución a la Pregunta 4

En la Figura 1.3 se muestra el diagrama de estados correspondiente al circuito detector. El código VHDL correspondiente al circuito detector se muestra en Código VHDL 1.3–1.4.

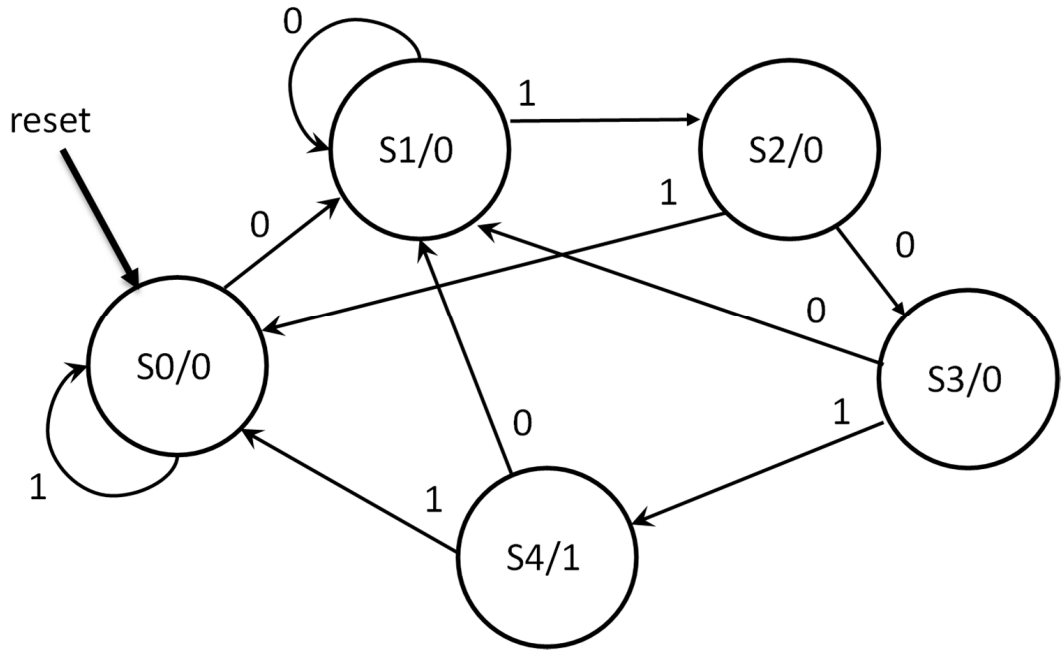


Figura 1.3: Diagrama de estados correspondiente al circuito detector.

```
-------------------------------------------------------
---Detector de la secuencia "0101" no solapada
-------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;

entity detector is
    port( Y       : out std_logic;
          state   : out std_logic_vector(2 downto 0);
          X       : in std_logic;
          reset   : in std_logic;
          clk     : in std_logic);
end entity detector;

architecture detector of detector is
 signal internal_state: std_logic_vector(2 downto 0);
begin
 state <= internal_state;

--Cálculo salida
 salida: process (internal_state) is
 begin
    if (internal_state = "100") then
        Y<= '1';
    else
        Y<= '0';
    end if;
 end process salida;
```

Código VHDL 1.3: Diseño del circuito detector.

```
--Cálculo del próximo estado
 proximo_estado: process (clk, reset)
 begin
      if (reset = '1') then --reset asíncrono
           internal_state <= "000";
      elsif (rising_edge(clk)) then
         case internal_state is
           when "000" =>     -- Estado actual: S0
               if X = '0' then
                  internal_state <= "001";
               end if;
           when "001" =>   --Estado actual: S1
                if X = '1' then
                   internal_state <= "010";
                end if;
            when "010" =>   --Estado actual: S2
                if X = '1' then
                   internal_state <= "000";
                else
                   internal_state <= "011";
                end if;
             when "011" =>   -- Estado actual: S3
                if X = '1' then
                    internal_state <= "100";
                 else
                   internal_state <= "001";
                 end if;
             when "100" =>   -- Estado actual: S4
                if X = '1' then
                    internal_state <= "000";
                 else
                   internal_state <= "001";
                 end if;
             when others=>   -- Por completitud
                internal_state <= "000";
              end case;
           end if;
end process proximo_estado;

end architecture detector;
-----------------------------------------
```

Código VHDL 1.4: Continuación del diseño del circuito detector.