INGENIERÍA DE COMPUTADORES 3

Solución al Trabajo Práctico - Septiembre de 2014

EJERCICIO 1

Se desea diseñar un circuito digital que implemente las funciones `X` e `Y` mostradas a continuación, que dependen de las tres variables `A`, `B` y `C`:

```
X = A'B'C' + A'BC' + A'BC + AB'C + ABC'
Y = A'BC' + AB'C' + AB'C
```

1.a) (0.5 puntos) Escriba en VHDL la **entity** del circuito que implemente las dos funciones lógicas. Es decir, que tenga tres entradas `A`, `B` y `C`, y dos salidas `X` e `Y`.

1.b) (1 punto) Escriba en VHDL la **architecture** que describa el *comportamiento* del circuito.

1.c) (0.5 puntos) Dibuje el diagrama de un circuito que implemente estas dos funciones lógicas al nivel de puertas lógicas. No es necesario que el circuito esté simplificado. A continuación, escriba en VHDL la **entity** y la **architecture** de cada una de las puertas lógicas que componen el circuito que acaba de dibujar.

1.d) (1 punto) Escriba en VHDL una **architecture** que describa la *estructura* del circuito que ha dibujado, instanciando y conectando las puertas lógicas que ha diseñado anteriormente.

1.e) (1 punto) Escriba en VHDL un banco de pruebas que permita visualizar, para todos los posibles valores de las entradas, las salidas de los circuitos diseñados en los Apartados 1.b y 1.d. Compruebe mediante inspección visual que los dos diseños funcionan correctamente.

Solución al Ejercicio 1

La **entity** del circuito que implementa las dos funciones lógicas se muestra en Código VHDL 1.1.

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

entity funcLog is
   port ( X, Y : out std_logic;
            A, B, C : in std_logic );
end entity funcLog;
-------------------------------------
```

Código VHDL 1.1: Solución al Apartado 1.a: **entity** del circuito.

El Código VHDL 1.2 muestra la **architecture** del circuito describiendo su comportamiento.

```
-------------------------------------
architecture Comp of funcLog is
begin
 X <= (not A and not B and not C ) or
       (not A and B and not C) or (not A and B and C) or
      (A and not B and C) or (A and B and not C);
 Y <= (not A and B and not C) or (A and not B and not C) or
       (A and not B and C);
end architecture Comp;
-------------------------------------
```

Código VHDL 1.2: Solución al Apartado 1.b: **architecture** del circuito describiendo su comportamiento.

La Figura 1.1 muestra el diagrama del circuito empleando seis puertas AND de tres entradas, tres inversores y tres puertas OR de tres entradas.

El código VHDL de la **entity** y la **architecture** de estas tres puertas lógicas se muestra en Código VHDL 1.3, 1.4 y 1.5, respectivamene.

El Código VHDL 1.6 muestra la **architecture** del circuito describiendo estructura.

El código VHDL del banco de pruebas del circuito se muestra en Código VHDL 1.7.

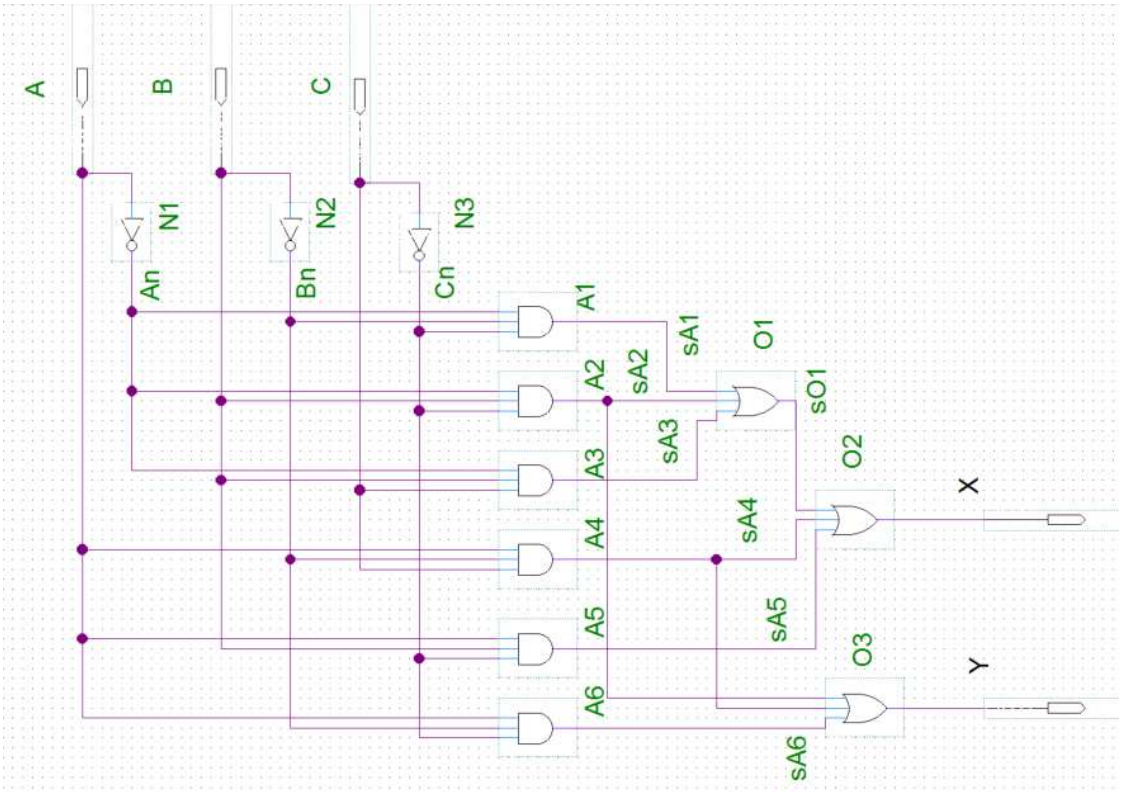

Figura 1.1: Solución al Apartado 1.c: diagrama al nivel de puertas lógicas.

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

entity and3 is
   port ( y0 : out std_logic;
          x0, x1, x2 : in std_logic );
end entity and3;

architecture and3 of and3 is
begin
   y0 <= x0 and x1 and x2;
end architecture and3;
----------------------------------
```

Código VHDL 1.3: Puerta AND lógica.

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

entity or3 is
   port ( y0 : out std_logic;
      x0, x1, x2 : in std_logic );
end entity or3;

architecture or3 of or3 is
begin
   y0 <= x0 or x1 or x2;
end architecture or3;
-------------------------------------
```

Código VHDL 1.4: Puerta OR lógica.

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

entity not1 is
   port ( y0 : out std_logic;
            x0 : in std_logic );
end entity not1;

architecture not1 of not1 is
begin
   y0 <= not x0;
end architecture not1;
-------------------------------------
```

Código VHDL 1.5: Puerta NOT lógica.

```
-----------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
architecture circuito_Estruc of funcLog is
   signal An, Bn,Cn, sA1: std_logic;
   signal sA2, sA3, sA4, sA5, sA6, sO1: std_logic;
-- Declaración de las clases de los componentes
   component and3 is
      port (  y0 : out std_logic ;
      x0, x1, x2 : in std_logic);
   end component and3;
   component not1 is
      port ( y0 : out std_logic;
              x0 : in std_logic );
   end component not1;
   component or3 is
      port ( y0 : out std_logic;
          x0, x1, x2 : in std_logic );
   end component or3;
begin
-- Instanciación y conexión de los componentes
   N1 : component not1 port map (An, A);
   N2 : component not1 port map (Bn, B);
   N3 : component not1 port map (Cn, C);
   A1 : component and3 port map (sA1, An, Bn, Cn);
   A2 : component and3 port map (sA2, An, B, Cn);
   A3 : component and3 port map (sA3, An, B, C);
   A4 : component and3 port map (sA4, A, Bn, C);
   A5 : component and3 port map (sA5, A, B, Cn);
   A6 : component and3 port map (sA6, A, Bn, Cn);
   O1 : component or3 port map (sO1, sA1, sA2, sA3);
   O2 : component or3 port map (X, sO1, sA4, sA5);
   O3 : component or3 port map (Y, sA2, sA4, sA6);
end architecture circuito_Estruc;
---------------------------------
```

Código VHDL 1.6: Solución al Apartado 1.d: **architecture** del circuito describiendo su estructura.

```
--------------------------------------------------
-- Banco de pruebas del circuito Ejercicio 1
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity bp_funcLog is
    constant DELAY     : time      := 20 ns; -- Retardo usado en el test
end entity bp_funcLog;

architecture bp_funcLog of bp_funcLog is
    signal X, Y : std_logic;
    signal A, B, C  : std_logic;

    component funcLog is
     port (  X, Y : out std_logic;
           A, B, C : in std_logic );
     end component funcLog;

begin
    UUT : component funcLog port map
         (X, Y, A, B, C);

vec_test : process is
      variable valor  : unsigned (2 downto 0);
    begin
       -- Generar todos los posibles valores de entrada
       for i in 0 to 7 loop
         valor := to_unsigned(i,3);
         A <= std_logic(valor(2));
         B <= std_logic(valor(1));
         C <= std_logic(valor(0));
         wait for DELAY;
       end loop;
       wait;      -- Final de la simulación
   end process vec_test;
end architecture bp_funcLog;
--------------------------------------------------
```

Código VHDL 1.7: Solución al Apartado 1.e: banco de pruebas del circuito.

EJERCICIO 2

A continuación se describe un algoritmo para realizar la división entre dos enteros ($y = a/b$), donde dividendo (a), divisor (b) y cociente (y) tienen el mismo número de bits. El tamaño de dividendo, divisor y cociente es $n + 1$ bits.

Paso 1 Se declaran dos vectores a_inp y b_inp de tamaño $2 * n + 1$ bits.

Paso 2 Se realizan las dos asignaciones siguientes:

$a\_inp = 0...0a_n a_{n-1}...a_1 a_0$

$b\_inp = 0...0b_n b_{n-1}...b_1 b_0$

Paso 3 Se realiza la asignación $i = n$.

Paso 4 Se rota i bits a la izquierda a b_inp y se almacena en b_inpr.

Paso 5 Se compara a_inp con b_inpr. Si $a\_inp >= b\_inpr$, entonces $y_i = \text{`}1\text{'}$ y se almacena en $a\_inp$ el resultado de la operación $a\_inp - b\_inpr$. En caso contrario, se realiza la asignación $y_i = \text{`}0\text{'}$ y no se modifica el valor de a_inp.

Paso 7 Se actualiza el índice $i = i - 1$.

Paso 6 Si $i >= 0$, entonces se vuelve al Paso 4. En caso contrario, se finaliza el algoritmo. El valor del cociente está almacenado en y y el valor del resto son los $n + 1$ bits menos significativos de a_inp.

En la tabla siguiente se muestran los resultados obtenidos al aplicar el algoritmo siendo $a = \text{“}1011\text{”}$ (valor 11 en decimal) y $b = \text{“}0011\text{”}$ (valor 3 en decimal), para cada valor de i. Como puede observarse, se obtiene $y = \text{“}0011\text{”}$ (valor 3 en decimal) y el resto “0010” (valor 2 en decimal).

i	a_inp	b_inpr	y_i
3	“0001011”	“0011000”	0
2	“0001011”	“0001100”	0
1	“0001011”	“0000110”	1
0	“0000101” “0000010” (resto)	“0000011”	1

2.a) (3 puntos) Escriba en VHDL la **architecture** que describe el comportamiento del circuito digital combinacional de un divisor de números de $n + 1$ bits siguiendo el algoritmo descrito anteriormente. El circuito ha de tener una señal llamada `err` que se pone a '1' sólo en el caso de que el divisor sea 0.

La **entity** del circuito se muestra a continuación.

```
entity divider is
   generic(n: integer := 3);
   port ( y: out std_logic_vector(n downto 0);
          rest  : out  integer range 0 to 2**(n+1)-1;
          err: out std_logic;
          a, b: in integer range 0 to 2**(n+1)-1);
end entity divider;
```

2.b) (3 puntos) Programe en VHDL un banco de pruebas que testee el circuito digital divisor. El banco de pruebas debe testear todas las posibles entradas al circuito combinacional. El banco de pruebas debe comparar las salidas de la UUT con las salidas esperadas, mostrando el correspondiente mensaje de error en caso de que las salidas obtenidas de la UUT no correspondan con las esperadas. Emplee este banco de pruebas para comprobar el diseño realizado al contestar el Apartado 2.a.

Solución al Ejercicio 2

La **architecture** describiendo el comportamiento del circuito divisor se muestra en Código VHDL 1.8.

```
-------------------------------------
architecture divider of divider is
begin
process (a, b)
   variable temp1: integer range 0 to 2**(n+1)-1;
   variable temp2: integer range 0 to 2**(n+1)-1;
begin
  -- Error e inicializacion
   temp1 := a;
   temp2 := b;
   if (b = 0) then err <= '1';
   else err <= '0';
   end if;
  -- y
   for i in n downto 0 loop
     if (temp1 >= temp2 * 2**i) then
        y(i) <= '1';
        temp1 := temp1 - temp2 * 2**i;
     else y(i) <= '0';
     end if;
   end loop;
  -- resto
   rest <= temp1;
end process;
end architecture divider;
-------------------------------------
```

Código VHDL 1.8: Solución al Apartado 2.a: **architecture** del circuito divisor describiendo su comportamiento.

El banco de pruebas del circuito divisor se muestra en Código VHDL 1.9–1.10.

```
-------------------------------------------------
-- Banco de pruebas del divisor
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity bp_divider is
    constant DELAY : time := 10 ns; -- Retardo usado en test
    constant n   : integer := 3;
end entity bp_divider;

architecture bp_divider of bp_divider is
  signal y     : std_logic_vector (n downto 0);
  signal rest  : integer range 0 to 2**(n+1)-1;
  signal err   : std_logic;
  signal a, b  : integer range 0 to 2**(n+1)-1;
  -- Declaracion UUT
  component divider is
    generic (n : integer := n);
    port (y     : out std_logic_vector(n downto 0);
          rest  : out integer range 0 to 2**(n+1)-1;
          err   : out std_logic;
          a, b  : in integer range 0 to 2**(n+1)-1);
  end component divider;
begin
  -- Instancia y conecta la UUT
  uut : component divider
    generic map (n)
    port map (y, rest, err, a, b);
  -- Generacion vectores test
  gen_vec_test : process
    variable test_a, test_b : integer range 0 to 2**(n+1)-1;
    variable esperado_coc: integer range 0 to 2**(n+1)-1;
    variable esperado_rest : integer range 0 to 2**(n+1)-1;
    variable esperado_err: std_logic;
    variable n_err : integer := 0;
  begin
```

Código VHDL 1.9: Solución al Apartado 2.b: banco de pruebas del circuito divisor.

```
    report("COMIENZA LA SIMULACION...");
    for test_b in 0 to 2**(n+1)-1 loop
      b <= test_b;
        for test_a in 0 to 2**(n+1)-1 loop
          a <= test_a;
          wait for DELAY;
          if(test_b /= 0) then
            -- Valores esperados para cociente y resto
            esperado_coc := test_a / test_b;
            esperado_rest := test_a REM test_b;
            esperado_err := '0';
            -- Compara los valores esperados con los actuales
            if (esperado_coc /= to_integer(unsigned(y))
              or   esperado_err /= err
              or esperado_rest /= rest) then
             report("Error. Cociente esperado (" &
              integer'image(esperado_coc) &
              ") no es igual que el actual (" &
              integer'image(to_integer(unsigned(y)))
              & "), o resto esperado (" &
              integer'image(esperado_rest) &
              ") no es igual que el actual (" &
              integer'image(rest)
              & ") o el error esperado (" &
              std_logic'image(esperado_err)
              & ") no es igual que el actual.");
             n_err := n_err + 1;
            end if;
          else esperado_err := '1';
            if (esperado_err /= err) then
            report("Error. Error esperado (" &
                 std_logic'image(esperado_err) &
                 ") no es igual que el actual.");
            n_err := n_err + 1;
            end if;
          end if;
        end loop;
    end loop;
    report("Test completo. Hay "& integer'image(n_err) & "
errores.");
    -- Mensaje de finalización
    report("FIN DE LA SIMULACION.");
    wait;
  end process gen_vec_test;
end architecture bp_divider;
-------------------------------------------------
```

Código VHDL 1.10: Solución al Apartado 2.b: continuación del banco de pruebas del circuito divisor.