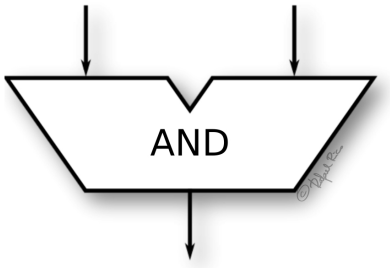
LABORATORIO DE ARQUITECTURA DE COMPUTADORES

SESIÓN 1: operadores lógicos

Objetivo

Una vez escritos los programas de las puertas lógicas, vamos a utilizarlas como instancias básicas en modelos estructurales más complejos. En concreto, diseñaremos operadores lógicos de tamaño 4 bits, es decir, *arrays* de puertas del mismo tipo que realizan en paralelo la misma operación sobre 4 bits.



Modelo VHDL de un operador AND de 4 bits

Asumimos que ya disponemos de nuestra puerta AND de 2 entradas escrita en VHDL (`and2.vhd`). Un posible modelo estructural de un operador AND de 4 bits sería el siguiente (`op_and_4.vhd`):

```
--operador AND de 4 bits

ENTITY op_and_4 IS
      PORT (vector_a, vector_b: IN BIT_VECTOR (3 DOWNTO 0);
      vector_r: OUT BIT_VECTOR (3 DOWNTO 0));
END op_and_4;


ARCHITECTURE estructural OF op_and_4 IS
--declaración de componentes
      COMPONENT and2
            PORT (e1, e2: IN BIT; sal: OUT BIT);
      END COMPONENT;
--indicamos dónde se encuentra la arquitectura de los componentes
      FOR ALL: and2 USE ENTITY WORK.and2(comportamiento);
--realizamos las conexiones
BEGIN
      celda0: and2 PORT MAP (vector_a(0), vector_b(0), vector_r(0));
      celda1: and2 PORT MAP (vector_a(1), vector_b(1), vector_r(1));
      celda2: and2 PORT MAP (vector_a(2), vector_b(2), vector_r(2));
      celda3: and2 PORT MAP (vector_a(3), vector_b(3), vector_r(3));
END estructural;
```

Como vemos, se declara el componente a utilizar y luego se crean 4 instancias de dicho modelo que en el programa compilado funcionan como 4 procesos concurrentes, uno por bit.

Test del operador

El *test bench* del operador es muy similar al propuesto en la sesión anterior para una puerta AND. Puesto que los buses de 4 bits se modelan con el tipo BIT_VECTOR, basta con asignar valor a esas señales. La asignación se puede hacer en binario (`:= X"1010"`) o en hexadecimal (`:= X"A"`) tal y como proponemos a continuación (`tb_op_and_4.vhd`):

```
--este es el test-bench de un operador AND de 4 bits
ENTITY tb_op_and_4 IS
END tb_op_and_4;

ARCHITECTURE estructural OF tb_op_and_4 IS
--declaración del componente a estudiar
  COMPONENT op_and_4
    PORT (vector_a, vector_b: IN BIT_VECTOR (3 DOWNTO 0);
          vector_r: OUT BIT_VECTOR (3 DOWNTO 0));
  END COMPONENT;
--especificación de la configuración del componente
  FOR componente: op_and_4 USE ENTITY WORK.op_and_4(estructural);
--declaración de señales de interconexión
  SIGNAL A, B, C: BIT_VECTOR (3 DOWNTO 0):= X"0";
--comienza la descripción de los procesos
BEGIN
--primer proceso: instanciación del componente a estudiar
  componente: op_and_4 PORT MAP (A, B, C);
--segundo proceso: estimulo de las señales de entrada
  PROCESS
    TYPE tipo_patron IS RECORD
--entradas a la puerta
      a, b, c: BIT_VECTOR (3 DOWNTO 0);
    END RECORD;
--la tabla de verdad
    TYPE array_patron IS ARRAY (NATURAL RANGE <>) of tipo_patron;
    CONSTANT tabla_verdad: array_patron :=
      ((X"0", X"0", X"0"),
       (X"A", X"F", X"A"),
       (X"F", X"7", X"7"),
       (X"3", X"A", X"2"));
  BEGIN
--inyecta las señales desde la tabla de verdad
    FOR i IN tabla_verdad'RANGE LOOP
      A <= tabla_verdad(i).a;
      B <= tabla_verdad(i).b;
--espero al resultado
      WAIT FOR 5 ns;
--compruebo si es correcto
      ASSERT C = tabla_verdad(i).c REPORT "resultado incorrecto" SEVERITY ERROR;
    END LOOP;
--mensaje final
    ASSERT false REPORT "fin del test" SEVERITY NOTE;
    WAIT;
  END PROCESS;
END estructural;
```

Prácticas

A) Escriba los modelos VHDL correspondientes a los operadores lógicos de 4 bits NOT, NAND, NOR, OR y XOR.

B) Escriba el *test bench* de cada operador y compruebe su correcto funcionamiento.

C) Elabore una memoria con los programas correspondientes a los *test benches*.