LABORATORIO DE ARQUITECTURA DE COMPUTADORES.

I. T. I. SISTEMAS / GESTIÓN

GUÍA DEL ALUMNO

Práctica 1: Operaciones de transferencia

Objetivos

- Comprender qué son los registros y los buffer
- Comprender cómo se realiza una operación de transferencia
- Comprender cómo se calcula la frecuencia máxima a la que puede funcionar un circuito

Método de trabajo

Lee cuidadosamente las indicaciones de la guía, realizando los pasos que se indican. La guía tiene como propósito que **TÚ** te asegures de que comprendes lo que va sucediendo mientras realizas lo que te indica que hagas: no tiene propósitos evaluativos.

Para todas las sesiones, necesitaremos textos de referencia del lenguaje para realizar programas o para comprender lo que ocurre durante las simulaciones. Por lo tanto se recomienda traer libros, apuntes y cualquier bibliografía de VHDL que sirva para hacer consultas. En el apartado "bibliografía" se proporcionan algunas referencias.

También traer un **disquete** o una **memoria USB** para llevar copia de los ficheros que se generen y se deseen conservar ya que, al tratarse de una máquina compartida, no hay garantías de que más tarde encontremos los trabajos que dejamos hoy. Pero dado que V-System no trabaja bien con los disquetes siempre habrá **que copiar de/al disco duro y trabajar en él.**

Bibliografía

- "VHDL: Lenguaje para síntesis y modelado de circuitos". F.Pardo y J.A.Boluda. Ed. Ra-Ma,1999.
- Lluis Terés y otros "VHDL: Lenguaje estándar de diseño electrónico" Mc Graw Hill, 1998
- Buscar en la WEB: Google encuentra referencias y pueden encontrarse libros en pdf

Introducción

Un **registro** es un circuito secuencial síncrono capaz de almacenar varios bits de información. El formato de esta información puede ser de dos tipos:

- Serie: los bits se transfieren uno a continuación del otro por una misma línea.
- Paralelo: se intercambian todos los bits al mismo tiempo, utilizando un número de líneas de transferencia igual al número de bits

El siguiente código en VHDL modela un registro de 4 bits que carga el valor de la señal de entrada al recibir un flanco de bajada. El dato almacenado siempre está disponible en la salida.

Fuente: registro.vhd

```
---------------------- Programa registro.vhd -------------------------------------
--registro
USE WORK.arq_pack.ALL;

ENTITY registro IS
        GENERIC (retardo_carga: TIME:= 10 ns);
        PORT (entrada:IN bus_resuelto; control: IN BIT:='0';
                salida: OUT bus_resuelto:=(others=>'0'));
END registro;

ARCHITECTURE comportamiento OF registro IS
BEGIN
        PROCESS (control)
        VARIABLE contenido_registro:vector_bus:=(others=>'0');
        BEGIN
                IF (control'EVENT AND control='0') THEN
                        FOR i IN 0 TO 3 LOOP
                                IF entrada(i)='1' OR entrada(i)='Z' THEN
                                        contenido_registro(i):='1';
                                  ELSE contenido_registro(i):='0';
                                END IF;
                        END LOOP;
                salida <= contenido_registro AFTER retardo_carga;
                END IF;
        END PROCESS;
END comportamiento;
-----------------------------------------------------------------------------------------------------------
```

Un **buffer** de datos es una ubicación de la memoria en una computadora o en un instrumento digital reservada para el almacenamiento temporal de información digital, mientras que está esperando ser procesada evitando el efecto de carga en un circuito.

El siguiente código en VHDL modela un buffer triestado que cuanta con una señal de control, activa por nivel bajo, que mantiene la salida del circuito en alta impedancia impidiendo la carga en el bus de datos. Si la señal de control está a nivel alto, entonces, la entrada se transfiere a la salida.

Fuente: buffer4.vhd

```
---------- Programa buffer4.vhd ---------------------------------
--simulación de un buffer de 16 bits
USE WORK.arq_pack.ALL;

ENTITY buffer4 IS
        GENERIC (retardo_transferencia: TIME:=15 ns; retardo_Z:TIME:=10 ns);
       PORT (entrada: IN bus_resuelto; control: IN BIT:='0'; salida: OUT bus_resuelto);
END buffer4;

ARCHITECTURE comportamiento OF buffer4 IS
BEGIN
        PROCESS (control, entrada)
        VARIABLE estado:bus_resuelto:=(others=>'0');
        BEGIN

          -- Se almacena el valor de la entrada
          FOR i IN 0 TO 3 LOOP
               estado(i) := entrada(i);
          END LOOP;

          -- Si está a uno se transfiere el dato al bus
          -- Si está a 0 se vuelca alta impedancia
                IF control='1' THEN
                  salida<=  TRANSPORT estado AFTER retardo_transferencia;
                ELSE salida <=  TRANSPORT (others=>'Z') AFTER retardo_Z;
                END IF;
        END PROCESS;
END comportamiento;
-----------------------------------------------------------------------------------------------------------
```

El reloj del sistema genera una señal cuadrada de una frecuencia determinada. En el caso del reloj modelado será de 50 MHz

Fuente: reloj.vhd

```
---------- Programa reloj.vhd
ENTITY reloj IS
        GENERIC (periodo: TIME:= 20 ns);
        PORT (reloj:OUT BIT:='0');
END reloj;

ARCHITECTURE comportamiento OF reloj IS
BEGIN
        PROCESS
        BEGIN
                WAIT FOR periodo/2;
                reloj<='1';

                WAIT FOR periodo/2;
                reloj<='0';
        END PROCESS;
END comportamiento;
```

Para modelar el circuito en VHDL se debe emplear el paquete arq_pack que contiene la declaración de tipos especiales, de procedimientos que pasan de valor lógico a entero y viceversa, así como la función de resolución necesaria para conectar los elementos a través de un mismo bus.

El paquete arq_pack queda como sigue:

Fuente: arq_pack.vhd

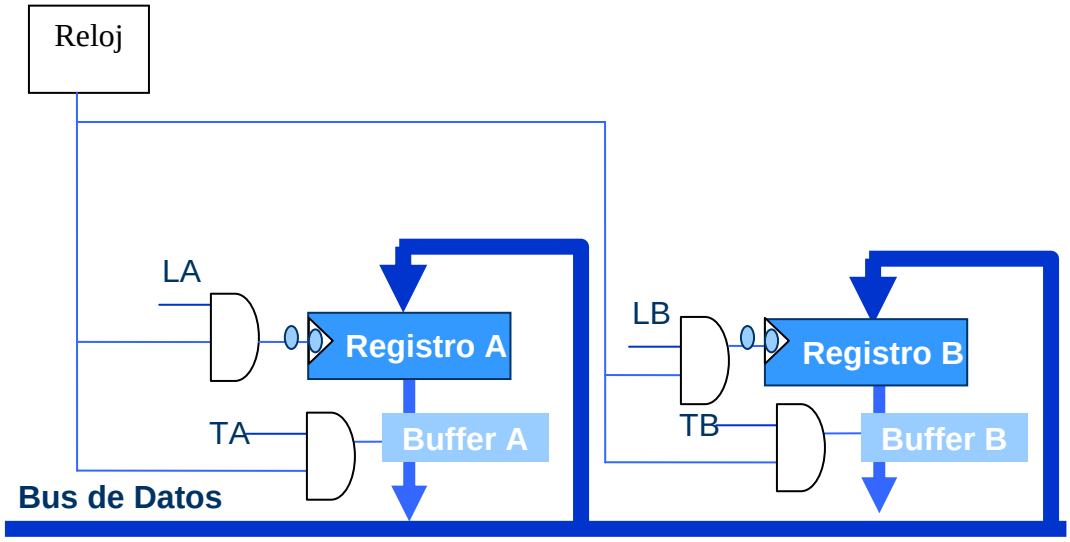
```
---------- Paquete arq_pack.vhd
PACKAGE arq_pack IS
        TYPE tri_estado IS ('0','1','Z');
        TYPE vector_bus IS ARRAY (3 DOWNTO 0) OF tri_estado;

        --una vez declarado el tipo del bus de datos se da una
        --función que asigne un valor de entre todos los fuentes
        --(todos los fuentes estan en el array_vector_bus)

        TYPE array_vector_bus IS ARRAY (INTEGER RANGE <>) OF vector_bus;

        FUNCTION resolucion (entrada: array_vector_bus) RETURN vector_bus;

        --la señal del bus será la resuelta
        SUBTYPE bus_resuelto IS resolucion vector_bus;

        PROCEDURE logico_entero (VARIABLE vector: IN BIT_VECTOR;
                                 VARIABLE entero: OUT INTEGER);

        PROCEDURE entero_logico (VARIABLE entero: IN INTEGER;
                                 VARIABLE vector: OUT BIT_VECTOR);
```

```
END arq_pack;

PACKAGE BODY arq_pack IS
        FUNCTION resolucion (entrada: array_vector_bus) RETURN vector_bus IS
                VARIABLE dato:vector_bus:=('Z','Z','Z','Z');
        BEGIN
                FOR i IN entrada'RANGE LOOP
                        FOR k IN 0 TO 3 LOOP
                                IF entrada(i)(k)='1' THEN dato(k):='1';
                                  ELSIF (entrada(i)(k)='0' AND dato(k)/='1') THEN
                                        dato(k):='0';
                                END IF;
                        END LOOP;
                END LOOP;
                RETURN dato;
        END resolucion;

        PROCEDURE logico_entero (VARIABLE vector:IN BIT_VECTOR;
                                 VARIABLE entero: OUT INTEGER) IS
                VARIABLE valor:INTEGER;
        BEGIN
                valor:=0;
                FOR i IN vector'RANGE LOOP
                        IF vector (i)='1' THEN valor:=valor+2**i; END IF;
                END LOOP;
                entero:=valor;
        END logico_entero;

        PROCEDURE entero_logico (VARIABLE entero:IN INTEGER;
                                 VARIABLE vector:OUT BIT_VECTOR) IS
                VARIABLE valor:INTEGER;
        BEGIN
                valor:=entero;
                FOR i IN vector'REVERSE_RANGE LOOP
                        vector (i):=BIT'VAL(valor REM 2);
                        valor:=valor/2;
                END LOOP;
        END entero_logico;
END arq_pack;
-------------------------------------------------------------------------------------------------------------------
```

Práctica

El objetivo de la práctica es comprender cómo son las operaciones elementales de transferencia entre distintos elementos de almacenamiento.

Para ello, y a partir de los modelos anteriores, se pide modelar de manera estructural el esquema de la siguiente figura, en la que los buffer triestado se han representado como círculos, y ser capaces de realizar una operación de transferencia.



Circuito a modelar en VHDL

En la práctica se pide:

1. Modelar la pareja registro-buffer del circuito de la figura anterior de manera estructural (tanto el registro como el buffer deben ser de 16 bits)
2. Averiguar la máxima frecuencia a la que puede trabajar el circuito.
3. Realizar el test bench para comprobar el correcto funcionamiento del circuito anterior.
4. Modelar el datapath del circuito de la figura anterior de manera estructural (también con búferes y registros de 16 bits)
5. Realizar el test bench correspondiente a una operación de transferencia en la que un dato presente en el bus de datos, se almacene en el registro A, luego se cambie el valor en el bus de datos y se almacene en el registro B, para, finalmente, que el dato almacenado en el registro A se transfiera al registro B.
6. Poner una frecuencia mayor, y observar que pasa.