Simulación de Arquitecturas de Computadores
Fe de erratas

La ecuación de la página 14:

$$\begin{aligned} c_{i+1} &= g_i + p_i \cdot c_i \\ g_i &= a_i \cdot b_i \\ p_i &= a_i + b_i \end{aligned} \qquad (1.2)$$

debe sustituirse por:

$$\begin{aligned} c_{i+1} &= g_i + p_i \cdot c_i \\ g_i &= a_i \cdot b_i \\ p_i &= a_i \oplus b_i \end{aligned} \qquad (1.2)$$

donde la función propagación de la etapa *i-ésima* es la OR exclusiva de los bits *i-ésimos* de los operandos de entrada.

El modelo de la página 74 del libro no es correcto. La versión correcta es la siguiente:

```
--modelo de memoria ROM 64x8
--FUNCIONA con el standard 1987
--NO FUNCIONA con el standard 1993


USE STD.TEXTIO.ALL;
--se incluye el paquete de manejo de ficheros ASCII
USE WORK.arq_pck1.ALL;
--se incluye el paquete arq_pck1 (procedimiento de conversión de lógico a entero, de entero a lógico,
--declaración de tipos y señal resuelta)
--el tipo vector_bus es de 8 lineas
USE WORK.file_pck.ALL;
--se incluye el paquete file_pck (declaración del tipo memoria_nx8 y del procedimiento
--que carga la ROM)


ENTITY rom_64x8 IS
        GENERIC(retardo:TIME:=50 ns);
        PORT(ADDRESS:IN BIT_VECTOR(5 DOWNTO 0):="000000";
        ME:IN BIT:='0';
        DATA_OUT:OUT vector_bus:=('Z','Z','Z','Z','Z','Z','Z','Z'));
END rom_64x8;

ARCHITECTURE comportamiento OF rom_64x8 IS
BEGIN
        PROCESS
        ----------------------------------------------------------------
        --declaracion de variables

        VARIABLE palabra:memoria_nx8(0 TO 63);      --matriz con todas las posiciones de memoria
        VARIABLE comienzo:BOOLEAN:=true
        VARIABLE dir_logica:BIT_VECTOR (5 DOWNTO 0);
        VARIABLE dir_entero:INTEGER;

        --ruta donde se encuentra el fichero con los datos de la ROM

        FILE fichero_rom:text IS IN "c:/arquitec/practicas/rom64.txt";
        ----------------------------------------------------------------

        --comienza el proceso
        --se despierta con cambios en ADDRESS y ME

        BEGIN
        WAIT ON ADDRESS, ME;
        ----------------------------------------------------------------
        --la primera vez que leo el fichero guardo un puntero a STRING por cada
        --línea del fichero de texto

        IF comienzo THEN
                cargar_rom(palabra, fichero_rom);
                comienzo:=false;
        END IF;

        ----------------------------------------------------------------
        --chip de memoria activo
```

```
        IF (ME='1') THEN        --la memoria está activada
        ------------------------------------------------------------
        --se lee la dirección
                dir_logica:=ADDRESS;
                logico_entero(dir_logica, dir_entero);
                --se convierte el valor lógico de la dirección a entero
                --se usa el procedimiento declarado en el paquete arq_pck1
        ------------------------------------------------------------
                --comienza el ciclo de lectura

                FOR i IN 0 TO 7 LOOP
                        IF palabra(dir_entero)(i)='0'THEN
                        DATA_OUT(i)<=TRANSPORT'0'AFTER retardo;
                        ELSE DATA_OUT(i)<=TRANSPORT'1'AFTER retardo;
                        END IF;


                END LOOP;
        ------------------------------------------------------------
        --chip de memoria inactivo

        ELSE                    --la memoria no está activada
                FOR i IN 0 TO 7 LOOP
                        DATA_OUT(i)<=TRANSPORT 'Z' AFTER retardo;
                END LOOP;
        END IF;
        ------------------------------------------------------------
        END PROCESS;
END comportamiento;
```

En el modelo anterior se ha utilizado el siguiente paquete:

```
--este paquete contiene los tipos usados en las prácticas del LABORATORIO DE ARQUITECTURA,
--un procedimiento para transformar un número en binario a su valor entero, otro para realizar la
--operación contraria y la señal resuelta con su función de resolución para el bus de datos de salida
--la única diferencia con el paquete ARQ_PACK es que el tipo vector_bus es de 8 bits

PACKAGE arq_pck1 IS
        TYPE tri_estado IS ('0', '1', 'Z');
        --este tipo cambia respecto al paquete ARQ_PACK.VHD
        TYPE vector_bus IS ARRAY (7 DOWNTO 0) OF tri_estado;

        --una vez declarado el tipo del bus de datos se da una
        --función que asigne una 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_pck1;
```

```
PACKAGE BODY arq_pck1 IS
        FUNCTION resolucion (entrada: array_vector_bus) RETURN vector_bus IS
        VARIABLE dato: vector_bus:=('Z', 'Z', 'Z', 'Z','Z', 'Z', 'Z', 'Z');
        BEGIN
                FOR i IN entrada'RANGE LOOP
                        FOR k IN 0 TO 7 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_pck1;
```

En el modelo de memoria ROM 64x8 también utiliza el siguiente paquete:

```
--este paquete contiene procedimientos usados en el manejo de
--ficheros ASCII de VHDL que cargan memorias ROM
--(se basa a su vez en el paquete TEXTIO)

--basicamente un fichero ASCII es una colección de estructuras STRING
--donde un STRING es una cadena de caracteres seguida de un RETURN
--se define LINE como un puntero a STRING

--así leer un fichero es encontrar el puntero LINE al STRING
--y luego leer los elementos del STRING


USE STD.textio.all;
PACKAGE file_pck IS
        --en primer lugar declaro un tipo llamado memoria_nx8 que es un
        --array ilimitado (n posiciones) de BIT_VECTOR de 8 elementos
        --(bits). Cada dirección será una palabra(i).
        TYPE memoria_nx8 IS ARRAY(INTEGER RANGE<>) OF BIT_VECTOR(7 DOWNTO 0);
        --ahora declaro un procedimiento que lee los datos de un
```

```
        --fichero ASCII y lo carga en la memoria.
        PROCEDURE cargar_rom (VARIABLE palabra:INOUT memoria_nx8;VARIABLE fichero_rom:IN text);
END file_pck;


PACKAGE BODY file_pck IS
        --definicion de un procedimiento que inicializa la memoria
        --ROM con un fichero de texto (ASCII).
        PROCEDURE cargar_rom (VARIABLE palabra:INOUT memoria_nx8;VARIABLE fichero_rom:IN text) IS
        VARIABLE L:line;
        VARIABLE i:integer;
        BEGIN
                FOR i IN palabra'RANGE
                LOOP
                        readline (fichero_rom, L);          --devuelve un puntero (LINE) a STRING
                        read (L, palabra(i));               --escribo el STRING sobre palabra(i)
                END LOOP;
        END cargar_rom;
END file_pck;
```

En la página 86 se hace referencia al modelo de la página 74 que se ha corregido más arriba. La fracción de código que se muestra debería ser como sigue:

```
USE STD.TEXTIO.ALL;
USE WORK.arq_pck1.ALL;
USE WORK.file_pck.ALL;

ENTITY rom_64x8 IS
        GENERIC(retardo:TIME:=50 ns; nombre:STRING:="rom64.txt");
        PORT(ADDRESS:IN BIT_VECTOR(5 DOWNTO 0):="000000";
        ME:IN BIT:='0';
        DATA_OUT:OUT vector_bus:=('Z','Z','Z','Z','Z','Z','Z','Z'));
END rom_64x8;

ARCHITECTURE comportamiento OF rom_64x8 IS
BEGIN
        PROCESS
        ------------------------------------------------------------
        --declaracion de variables

        VARIABLE palabra:memoria_nx8(0 TO 63);      --matriz con todas las posiciones de memoria
        VARIABLE comienzo:BOOLEAN:=true
        VARIABLE dir_logica:BIT_VECTOR (5 DOWNTO 0);
        VARIABLE dir_entero:INTEGER;

        --ruta donde se encuentra el fichero con los datos de la ROM

        FILE fichero_rom:text IS IN nombre;
        ------------------------------------------------------------
```