



**Instituto Politécnico Nacional  
Escuela Superior de Cómputo  
Academia de Sistemas Digitales**



## **Manejo de Archivos en VHDL**

**Autor: Victor Hugo García Ortega**



# ARCHIVOS

---

Un archivo es un objeto que permite la comunicación del diseño con su entorno exterior.

Por medio de ellos se pueden leer y escribir datos cuando se hacen evaluaciones del circuito.

Se pueden leer y escribir datos a o desde memorias.

Un archivo consiste de un flujo secuencial de un tipo de dato determinado.



# ARCHIVOS

---

Una variable puede ser asignada mediante una sentencia, pero un archivo no.

A un archivo se le pueden realizar siguientes operaciones:

- Lectura
- Escritura
- Verificación de fin de archivo.

Estas operaciones se realizan con dos procedimientos y una función especiales, los cuales son:



# ARCHIVOS

---

Procedimiento: `READ( file, data )`. Lee un objeto del archivo especificado por *file* y retorna el objeto en el argumento *data*.

Procedimiento: `WRITE( file, data )`. Escribe un objeto especificado por *data* al archivo *file*.

Función: `ENDFILE( file )`. Retorna *true* cuando se alcanzó la marca de fin de archivo.

El uso de estos procedimientos y funciones requiere una declaración de tipo de archivo y de objeto.



# DECLARACIÓN

---

Los archivos se tienen que especificar con TYPE:

```
TYPE type_name IS FILE OF data_type;
```

Después podemos declarar objetos de tipo FILE usando ese nuevo tipo de dato:

```
FILE file_type_name : type_name IS file_mode path_file;
```



## DECLARACIÓN

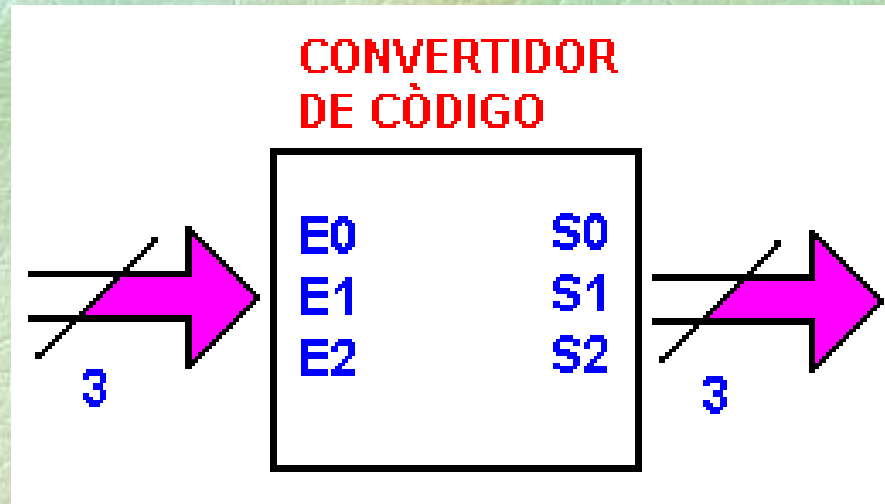
---

El atributo `file_mode` puede ser IN o OUT.

Si el atributo `file_mode` es IN, entonces el archivo se puede leer con el procedimiento READ.

Si el atributo `file_mode` es OUT, entonces el archivo se puede escribir con el procedimiento WRITE.

# EJEMPLO: CONVERTIDOR DE CÓDIGO



Código binario secuencial			Código Gray		
E2	E1	E0	S2	S1	S0
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0



# PROGRAMA EJEMPLO

---

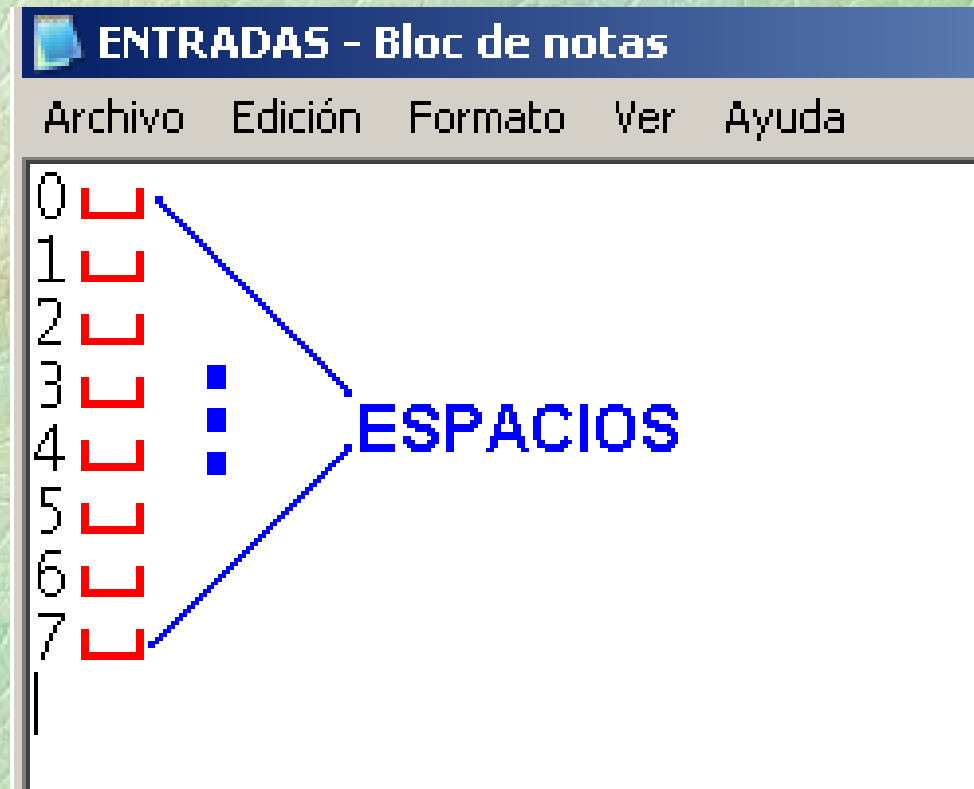
```
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22
23 entity CONV_COD is
24     Port ( E : in  STD_LOGIC_VECTOR (2 downto 0);
25           S : out  STD_LOGIC_VECTOR (2 downto 0));
26 end CONV_COD;
27
28 architecture PROGRAMA of CONV_COD is
29 begin
30     S <= "000" when E="000" else
31         "001" when E="001" else
32         "011" when E="010" else
33         "010" when E="011" else
34         "110" when E="100" else
35         "111" when E="101" else
36         "101" when E="110" else
37         "100";
38 end PROGRAMA;
```



## ARCHIVO ENTERO DE PRUEBA

---

Se debe dejar un espacio después de cada valor entero para que se realice correctamente la lectura del número.



# LECTURA DEL ARCHIVO DE PRUEBA

---

```
81  stim_proc: process
82
83  TYPE ARCHIVO_ENTERO IS FILE OF INTEGER;
84  FILE MI_ARCHIVO : ARCHIVO_ENTERO IS IN  "ENTRADAS.TXT";
85
86  VARIABLE DATO : INTEGER RANGE 0 TO 7;
87  begin
88
89  WHILE NOT ENDFILE(MI_ARCHIVO) LOOP
90      READ( MI_ARCHIVO, DATO );
91      E <= CONV_STD_LOGIC_VECTOR( DATO, 3 );
92      WAIT FOR 100 NS;
93  END LOOP;
94  wait;
95  end process;
```



## PAQUETE TEXTIO

---

El paquete (Textual Input and Output - TextIO) contiene procedimientos y funciones que dan al diseñador la habilidad para leer y escribir hacia archivos de texto formateados.

Estos archivos de texto son archivos ASCII con cualquier formato que el diseñador desee.

TextIO trata a los archivos como archivos de líneas, donde una línea es una cadena terminada con el retorno de carro.



## PAQUETE TEXTIO

---

TextIO contiene el tipo de dato *line* donde se encuentra la línea que se escribe o se lee del archivo.

Cuando se hace una lectura del archivo, primero se lee una línea del archivo usando el tipo de dato *line*. Después la información contenida en *line* es procesada campo por campo.

Para escritura, primero se forma la línea campo por campo y después se escribe al archivo.



# FUNCIONES DEL PAQUETE TEXTIO

---

Procedimiento: `file_open([file_status], file_handle, file_name, file_mode);`

Los modos de archivos son:

- `READ_MODE`
- `WRITE_MODE`
- `APPEND_MODE`

Las condiciones de estado son:

- `OPEN_OK`
- `STATUS_ERROR`
- `NAME_ERROR`
- `MODE_ERROR`



# FUNCIONES DEL PAQUETE TEXTIO

---

Procedimiento:

`readline(file_handle, line_variable);`

Lee una línea de texto completa del archivo.

Procedimiento:

`writeline(file_handle, line_variable);`

Escribe una línea de texto completa al archivo.



## FUNCIONES DEL PAQUETE TEXTIO

---

Procedimiento: `write(file_handle, object);`  
Escribe un objeto al archivo o a una línea.

Procedimiento: `read(file_handle, object);`  
Lee un objeto del archivo o de una línea.

Función: `endfile(file_handle);`  
Verifica el fin de un archivo.

Función: `file_close(file_handle);`  
Cierra el archivo.



## FUNCIONES DEL PAQUETE TEXTIO

---

El procedimiento `write` puede llevar dos argumentos opcionales: justificado y posición.

```
write(MI_LINEA, VAR_OPER, right, 5);    --ESCRIBE EL CAMPO OPER
write(MI_LINEA, VAR_Q, right, 10);      --ESCRIBE EL CAMPO Q
```

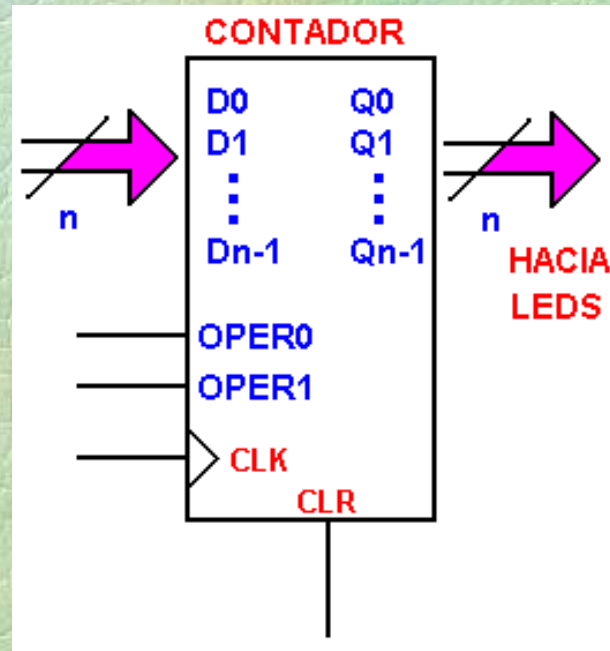
Además de los procedimientos `write` y `read`, se tienen los procedimientos `hwrite` y `hread`. Estos procedimientos realizan la lectura y escritura de un `STD_LODIC_VECTOR` en hexadecimal.

```
Hwrite(LINEA_RES, WDATA, right, 6);
```

```
Hread(MI_LINEA, WDATA);
```



# EJEMPLO: CONTADOR GENÉRICO



OPER1	OPER0	OPERACIÓN
0	0	CARGA
0	1	RETENCIÓN
1	0	CONTEO ASCENDENTE
1	1	CONTEO DESCENDENTE

# PROGRAMA EJEMPLO

```
1  LIBRARY IEEE;
2
3  USE IEEE.STD_LOGIC_1164.ALL;
4  USE IEEE.STD_LOGIC_ARITH.ALL;
5  USE IEEE.STD_LOGIC_UNSIGNED.ALL;
6
7  ENTITY CONT_GEN IS
8      PORT(
9          CLK, CLR : IN STD_LOGIC;
10         D : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
11         OPER : IN STD_LOGIC_VECTOR( 1 DOWNTO 0 );
12         Q : INOUT STD_LOGIC_VECTOR( 7 DOWNTO 0 )
13     );
14  END CONT_GEN;
15
16  ARCHITECTURE ACONT OF CONT_GEN IS
17  BEGIN
18      PCONT : PROCESS( CLK, CLR )
19      BEGIN
20          IF( CLR = '1' ) THEN
21              Q <= (OTHERS => '0');
22          ELSIF( CLK'EVENT AND CLK = '1' ) THEN
23              CASE OPER IS
24                  WHEN "00" =>
25                      Q <= D;
26                  WHEN "01" =>
27                      Q <= Q;
28                  WHEN "10" =>
29                      Q <= Q + 1;
30                  WHEN OTHERS =>
31                      Q <= Q - 1;
32              END CASE;
33          END IF;
34      END PROCESS PCONT;
35  END ACONT;
```





# LIBRERÍA Y PAQUETE TEXTIO

---

AGREGAR LA LIBRERÍA ESTANDAR STD Y EL PAQUETE TEXTIO AL TEST BENCH

```
27 -----
28 LIBRARY ieee;
29 LIBRARY STD;
30 USE STD.TEXTIO.ALL;
31 USE ieee.std_logic_TEXTIO.ALL;    --PERMITE USAR STD_LOGIC
32
33 USE ieee.std_logic_1164.ALL;
34 USE ieee.std_logic_UNSIGNED.ALL;
35 USE ieee.std_logic_ARITH.ALL;
36
```



# EMPLEO DE ARCHIVOS

---

Declaración de variables y manejadores de archivo. La variable CADENA se usa para poner en la primer línea del archivo de resultados una cabecera con los nombres de las señales.

```
92  -- Stimulus process
93  stim_proc: process
94  file ARCH_RES : TEXT;
95  variable LINEA_RES : line;
96  VARIABLE VAR_Q : STD_LOGIC_VECTOR(7 DOWNT0 0);
97
98  file ARCH_VEC : TEXT;
99  variable LINEA_VEC : line;
100 VARIABLE VAR_OPER : STD_LOGIC_VECTOR(1 DOWNT0 0);
101 VARIABLE VAR_D : STD_LOGIC_VECTOR(7 DOWNT0 0);
102 VARIABLE VAR_CLR : STD_LOGIC;
103 VARIABLE CADENA : STRING(1 TO 4);
104 begin
```

# EMPLEO DE ARCHIVOS

---

Se abren los manejadores de archivos y se escribe una cabecera en el archivo de resultados con los nombres de las señales.

El atributo LENGTH nos da el tamaño de la cadena.

```
104  begin
105      file_open(ARCH_VEC, "VECTORES.TXT", READ_MODE);
106      file_open(ARCH_RES, "RESULTADO.TXT", WRITE_MODE);
107
108      CADENA := "OPER";
109      write(LINEA_RES, CADENA, right, CADENA'LENGTH+1);  --ESCRIBE LA CADENA "OPER"
110      CADENA := "  D";
111      write(LINEA_RES, CADENA, right, CADENA'LENGTH+1);  --ESCRIBE LA CADENA "  D"
112      CADENA := " CLR";
113      write(LINEA_RES, CADENA, right, CADENA'LENGTH+1);  --ESCRIBE LA CADENA " CLR"
114      CADENA := "  Q";
115      write(LINEA_RES, CADENA, right, CADENA'LENGTH+1);  --ESCRIBE LA CADENA "  Q"
116      writeline(ARCH_RES, LINEA_RES); -- escribe la linea en el archivo
117
118      WAIT FOR 100 NS;
119      FOR I IN 0 TO 14 LOOP
```



# EMPLEO DE ARCHIVOS

---

Se leen los vectores de prueba y después del flanco de reloj se escriben los resultados en el archivo de resultados.

```
119     FOR I IN 0 TO 14 LOOP
120         readline(ARCH_VEC, LINEA_VEC); -- lee una linea completa
121
122         read(LINEA_VEC, VAR_OPER);
123         OPER <= VAR_OPER;
124         Hread(LINEA_VEC, VAR_D);
125         D <= VAR_D;
126         read(LINEA_VEC, VAR_CLR);
127         CLR <= VAR_CLR;
128
129         WAIT UNTIL RISING_EDGE(CLK); --ESPERO AL FLANCO DE SUBIDA
130
131         VAR_Q := Q;
132         write(LINEA_RES, VAR_OPER, right, 5); --ESCRIBE EL CAMPO OPER
133         Hwrite(LINEA_RES, VAR_D, right, 5); --ESCRIBE EL CAMPO D
134         write(LINEA_RES, VAR_CLR, right, 5); --ESCRIBE EL CAMPO CLR
135         Hwrite(LINEA_RES, VAR_Q, right, 5); --ESCRIBE EL CAMPO Q
136
137         writeline(ARCH_RES, LINEA_RES); -- escribe la linea en el archivo
138
139     end loop;
```

# EMPLEO DE ARCHIVOS

---

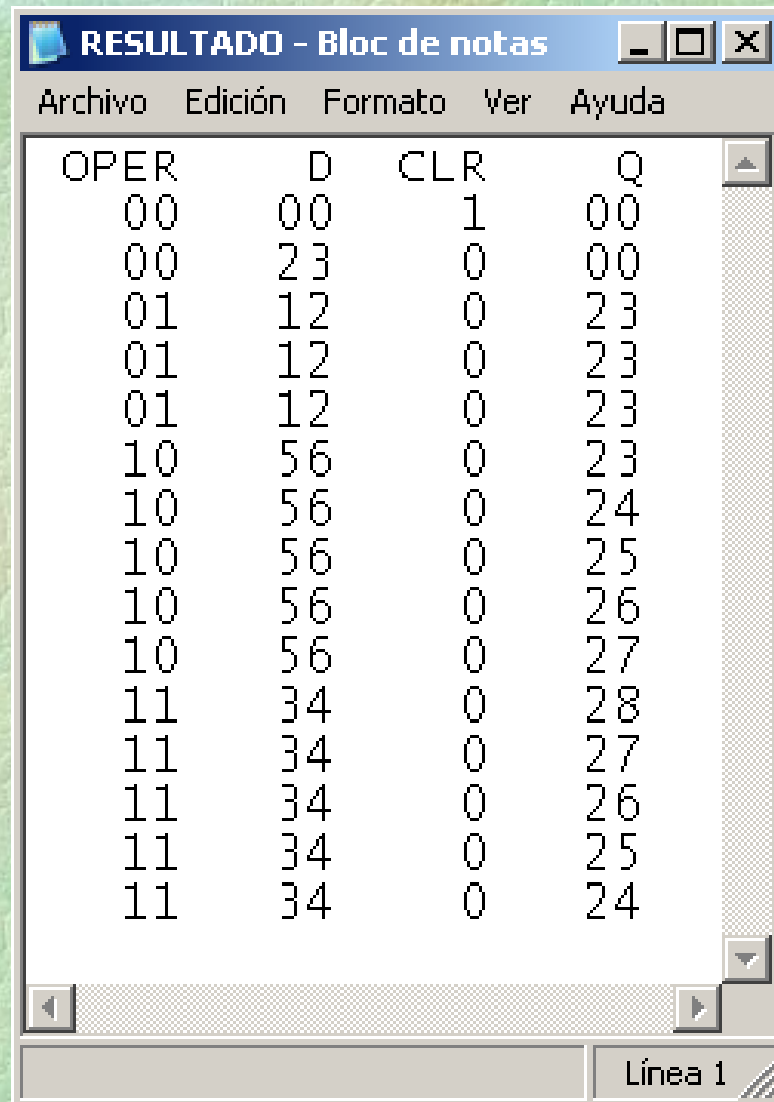
Después del ciclo FOR LOOP se cierran los manejadores de archivo y se detiene la simulación con wait.

```
139     end loop;
140     file_close(ARCH_VEC);  -- cierra el archivo
141     file_close(ARCH_RES);  -- cierra el archivo
142
143     wait;
144 end process;
```



# ARCHIVO DE RESUTADOS

---



A screenshot of a Notepad window titled "RESULTADO - Bloc de notas". The window contains a table with four columns: OPER, D, CLR, and Q. The table lists 16 rows of data. The first row has OPER=00, D=00, CLR=1, and Q=00. The subsequent rows show various combinations of OPER and D values, with CLR values being 0 or 1, and Q values ranging from 00 to 28. The window has a standard menu bar with Archivo, Edición, Formato, Ver, and Ayuda. The status bar at the bottom indicates "Línea 1".

OPER	D	CLR	Q
00	00	1	00
00	23	0	00
01	12	0	23
01	12	0	23
01	12	0	23
10	56	0	23
10	56	0	24
10	56	0	25
10	56	0	26
10	56	0	27
11	34	0	28
11	34	0	27
11	34	0	26
11	34	0	25
11	34	0	24

---

**GRACIAS POR SU ATENCIÓN**

**[vgarciao@yahoo.com.mx](mailto:vgarciao@yahoo.com.mx)**