Capítulo 7

Diseño jerárquico en VHDL

Introducción

El diseño jerárquico es una herramienta de apoyo que permite la programación de extensos diseños mediante la unión de pequeños bloques; es decir, un diseño jerárquico agrupa varias entidades electrónicas, las cuales se pueden analizar y simular de manera individual con facilidad, para luego relacionarlas a través de un algoritmo de integración llamado Top Level (Fig. 7.1a).

La conceptualización del diseño Top Level difiere de la programación de los sistemas digitales, integrados en una entidad (Cap. 5). En este caso no se trata de integrar dos o más módulos electrónicos individuales (Fig. 7.1b), si no diseñar cada módulo por separado y luego coordinar su funcionamiento a través del algoritmo de programación Top Level.

Una ventaja importante del diseño jerárquico en la programación de grandes diseños es la facilidad para trabajar al mismo tiempo con otros diseñadores (paralelismo), ya que mientras uno puede diseñar una parte del sistema, otro puede desarrollar un bloque distinto para unirlos en un solo proyecto más tarde.

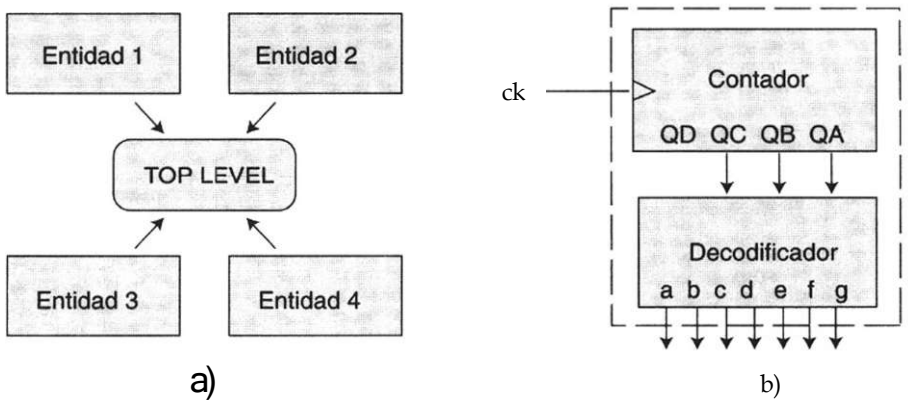


Figura 7.1 a) Estructuras jerárquicas, b) Integración de entidades.

7.1 Metodología de diseño de estructuras jerárquicas

Una metodología que se recomienda al programar extensos diseños es la siguiente:

1) Analizar con detalle el problema y descomponer en bloques individuales la estructura global.
2) Diseñar y programar módulos individuales (componentes).
3) Crear un paquete de componentes.
4) Diseñar el programa de alto nivel (Top Level).

Con el fin de describir y detallar la metodología empleada en el diseño de estructuras jerárquicas, consideremos el circuito AMD2909. Este dispositivo es un secuenciador de 4 bits desarrollado por la compañía Advanced Micro Devices, cuya función es transferir a su bus de salida (Y) una de entre cuatro fuentes internas y externas de datos. En la figura 7.2 se muestra la estructura externa del circuito y en la tabla 7.1 se indica la función de cada terminal.

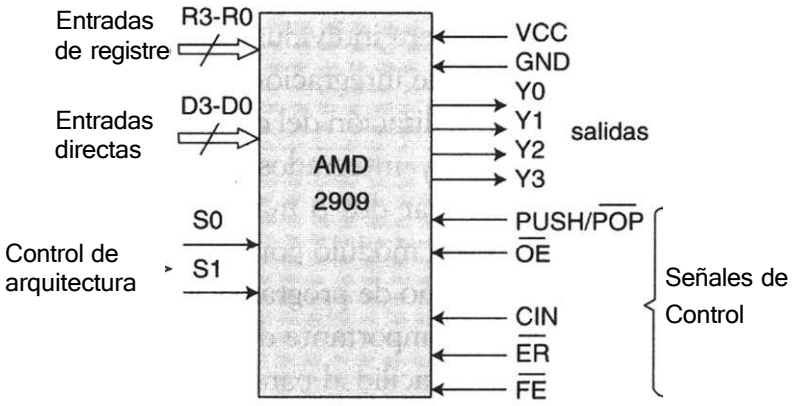


Figura 7.2 Secuenciador de cuatro bits AMD2909.

Terminal	Función
Entradas de registro R3-R0	Conservan el estado presente en una función de *hold o* retén.
Entradas directas D3-D0	Se usan como entradas del secuenciador para indicar un cambio de dirección en la lógica del programa.
Entrada /ER	Habilitación del registro R.
Entrada /FE	Habilitación del puntero de pila (stack pointer: ST).
Entrada CIN	Acarreo de entrada.
Entrada /OE	Habilitación de salidas.
Entrada PUSH/POP	Señales para brincos y retornos de subrutina.
Entradas SO y SI	Líneas de selección que determinan una de cuatro fuentes diferentes de entrada.
Salidas Y3-Y0	Salidas del secuenciador.
Salida COUT	Acarreo de salida.
Vcc y Gnd	Alimentación del circuito.

Tabla 7.1 Descripción de las señales de entrada y salida del secuenciador.

Descripción del circuito AMD2909

El secuenciador de 4 bits tiene entrada para un bus de datos externo D (D3 - *DO),* un registro *R (R3 - RO),* un apuntador de pila de una palabra ST y un sumador de 4 bits, que funciona como contador de programa (Fig. 7.3).

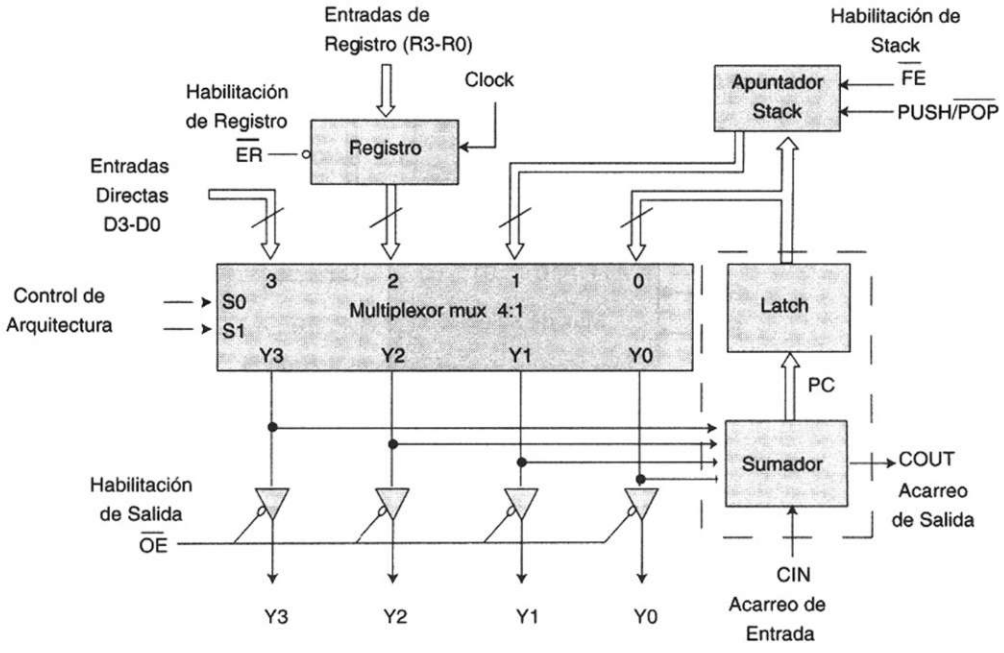


Figura 7.3 Descripción interna del circuito secuenciador AMD2909 [1].

Las entradas directas D se utilizan para canalizar las direcciones de carga desde una memoria de programa y/o control hacia el secuenciador de microprograma. En alguna de las arquitecturas, las entradas R sirven para almacenar el estado presente en una función de *hold* o retén. El contador de microprograma, *PC,* incrementa la salida en PC+ 1, siempre y cuando el acarreo de entrada *CIN* sea igual a 1. Esto permite realizar una instrucción de cuenta o almacenar en la pila la siguiente dirección cuando se hace un llamado a subrutina, por lo que al salir de ésta la dirección se obtiene de la pila y se canaliza hacia el bus de salida Y. Por último, el circuito contiene cuatro multiplexores de 4:1 que seleccionan una de sus cuatro entradas *PC, ST, R* y D, a través de sus líneas de selección *SO y SI.*

7.2 Análisis del problema y descomposición en bloques individuales de la estructura global

Como se mencionó, el diseño jerárquico basa su fortaleza en la descomposición o división de un diseño. Lo anterior permite analizar los diferentes subsistemas que lo forman y unirlos más tarde a través de un programa denominado Top Level.

En la figura 7.4 se muestra la interconexión interna de cada uno de los bloques y las señales de control que interactúan en cada uno de los subsistemas (multiplexor de selección, contador de microprograma, registro y apuntador de pila) del circuito AMD2909.

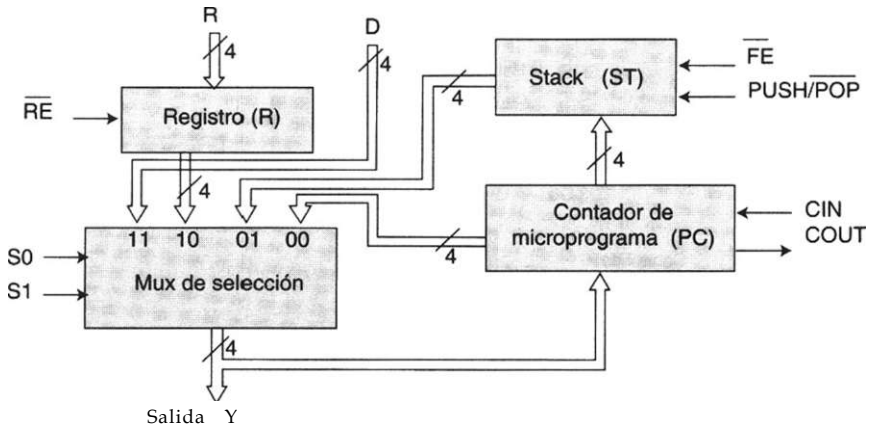


Figura 7.4 Diagrama general de diseño.

7.3. Diseño y programación de componentes o unidades del circuito

El primer paso de diseño consiste en programar de manera individual cada uno de los componentes y/o unidades del circuito. Un componente es la parte de un programa que define un elemento físico, el cual se puede usar en otros diseños o entidades. Con base en la figura anterior, iniciaremos la programación de cada uno de los bloques que forman el circuito AMD2909.

Diseño del registro (R)

En la figura 7.5 se muestra el registro R y las señales de control asociadas a él, mientras que en el listado 7.1 se puede observar el código VHDL correspondiente a esta entidad.

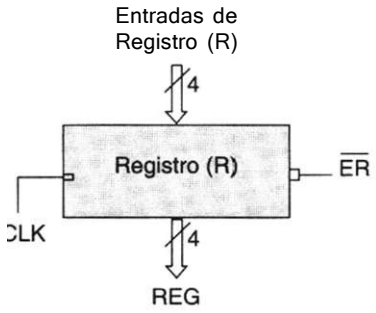


Figura 7.5 Registro de 4 bits.

```
-- Diseño del registro R
library ieee;
use ieee.std_logic_1164.all;
entity registro is port(
        R:    in std_logic_vector(3 downto 0);
 ER, CLK:     in std_logic;
      REG:    inout std_logic_vector(3 downto 0));
end registro;

architecture arq_reg of registro is
begin
 process (CLK,RE,REG,R) begin
   if  (CLK'event and CLK ='1')then
        if ER = '0' then
           REG <= R;
   else
           REG <= REG;
        end if;
   end if;
 end process;
end arq__reg ;
```

Listado 7.1 Programación de un registro de 4 bits.

Diseño del multiplexor

Este componente es simplemente un multiplexor cuádruple de 4:1, con dos líneas de selección (SO y *SI),* cuatro entradas *(R, ST, D, PC)* y una salida (Y) de 4 bits (Fig. 7.6).

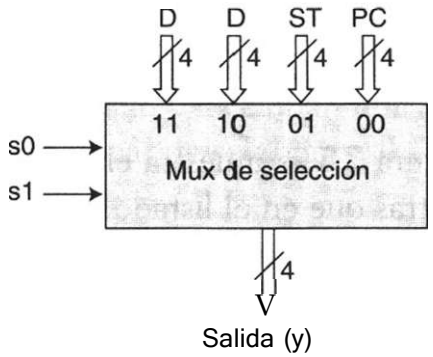


Figura 7.6 Multiplexor de selección.

El código de este componente se muestra en el listado 7.2.

```
-- Diseño del mux que selecciona una operación
library ieee;
use ieee.std_logic_1164.all;
   entity mux_4 is port(
      D,R,ST,PC: in std_logic_vector(3 downto 0);
              S: in std_logic_vector(1 downto 0);
              Y: out std_logic_vector(3 downto 0));
   end mux_4;

architecture arq_mux of mux_4 is
begin
    with S select
         Y <=   R when "00",
               ST when "01",
               PC when "10",
                D when others;
end arq_mux;
```

Listado 7.2 Código del multiplexor de selección.

Contador de microprograma (PC)

En esencia, la función de este bloque es incrementar la dirección de entrada cuando *CIN* es igual a uno. El diagrama correspondiente se encuentra en la figura 7.7.

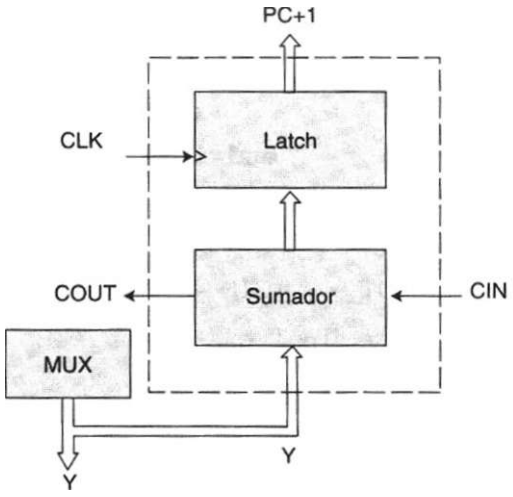


Figura 7.7 Contador de microprograma.

En el listado 7.3 se aprecia el código que describe el funcionamiento del contador de microprograma.

Como puede observarse, la manera más sencilla de programarlo es a través de la arquitectura funcional, considerando el circuito sumador y el contador como una entidad de diseño, con entradas y salidas generales. El funcionamiento es muy simple: cuando el acarreo de entrada *(CIN)* tiene el valor de '1' y el reloj del sistema está en la transición de '0' a '1', la dirección Y se incrementa en uno y su valor pasa al bus de salida PC. En caso contrario, cuando *CIN* = '0', se asigna a PC el valor de Y sin cambios.

```
-- Diseño del bloque de cuenta (contador de microprograma)
library ieee;
use ieee.std_logic_1164.all;
use work.std_arith.all;
     entity mpc is port (
    CIN, CLK:    in std_logic;
            Y:   in std_logic_vector(3 downto 0);
         COUT:   inout std_logic;
           PC:   inout std_logic_vector(3 downto 0));
     end mpc;
```

Continúa

```
architecture arq_mpc of mpc is
begin
  process (CLK, Y, CIN) begin
            if (CLK'event and CLK = '1') then
                 if (CIN = '1') then
                     PC <= Y + 1;
                 else
                     PC<= Y;
                 end if;
            end if;
  end process;

   COUT <= (CIN and Y(0)and Y(1) and Y(2) and Y(3));

end arq_mpc;
```

Listado 7.3 Diseño del contador de microprograma.

Apuntador de pila (Stack Pointer)

La pila (stack) de la figura 7.8 está diseñada para almacenar un dato de 4 bits, de modo que cuando en un programa se hace un llamado a subrutina, la siguiente dirección (PC + 1) se almacena en la pila, por lo que al salir de la subrutina la dirección se toma de la pila y se envía al multiplexor de selección, el cual la canaliza al bus de salida (Y).

Para poder introducir y almacenar un dato en la pila, se debe habilitar primero la señal *FE* (habilitación de pila) y la señal PUSH (FE = '0' y PUSH=T). De esta forma, cuando la transición del reloj *(CLK)* sea de '0' a '1', el dato que se encuentra en PC se introduce y almacena en la pila hasta que la señal PUSH se deshabilite *(PUSH = '0').* Para sacar el dato se habilita la señal POP (POP = '0'), con lo que el dato se canaliza al bus de salida *ST.*

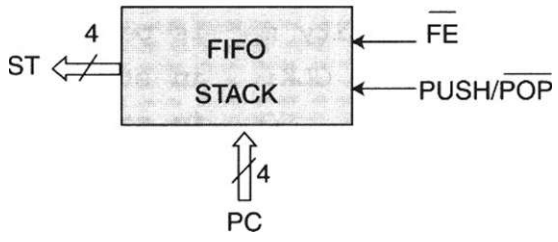


Figura 7.8 Pila de una palabra de 4 bits.

En el listado 7.4 se muestra el programa que indica el funcionamiento de la entidad correspondiente a la pila (stack).

```
— Diseño de una pila de una palabra de 4 bits
library ieee;
use  ieee.std_logic_1164.all;
use work.std_arith. all;

  entity stack is port(
   CLK,FE,PUSH,  POP:  in stdJLogic;
                PC:  in std_logic_vector (3 downto 0) ;
                ST:  inout std_logic_vector (3 downto 0));
    end stack;

architecture arq_stack of stack is
  signal var: std_logic_vector  (3 downto 0);
 begin
   process (FE, CLK, PUSH, POP, PC)
     variable x: std_logic_vector (3 downto 0);
       begin
           if (CLK'event and CLK = '1') then
                if (FE = '0') then
           if (PUSH = '1') then
                        x := PC; -- almacena dato
                        var <= x;
                   elsif (POP = '0') then
                        ST <= VAR; — saca dato
                    else
                        ST <= ST;
                   end if;
                end if;
            end if;
   end process;
end arq_stack;
```

Listado 7.4 Diseño de una pila de una palabra de 4 bits.

7.4 Creación de un paquete de componentes

Una vez que se ha diseñado cada módulo que forma la arquitectura general, se crea un programa que contenga los componentes de cada una de las entidades de diseño descritas con anterioridad. Para esto es necesario identificar primero el paquete en que se almacenarán los diseños (línea 4). En nuestro ejemplo el paquete recibe el nombre de *comps_sec* (el usuario escoge dicho nombre).

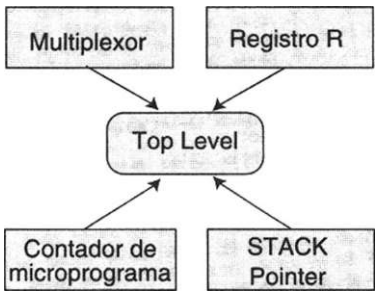
Como puede observar, en el listado 7.5 se ilustra la manera de declarar cada componente de un paquete llamado *comps_sec.* Note que cada componente se declara como una entidad de diseño, con la omisión de la palabra reservada is y agregando la cláusula component (líneas 5,11,18 y 24).

```
1 --Creación del paquete de componentes del secuenciador 2909
2 library ieee;
3 use ieee.std_logic_1164.all ;
4 package comps_sec is — declaración del paquete
5 component registro port(
6         R:       in std_logic_vector(3 downto 0);
7         ER,CLK:  in std_logic;
8         REG:     inout std_logic_vector(3 downto 0));
9   end component;
10
11 component mpc port( — declaración del contador
12  CIN,CLK:       in std_logic;
13        Y:       in std_logic_vector (3 downto 0) ;
14     COUT:       inout std_logic;
15       PC:       inout std_logic_vector(3 downto 0));
16   end component;

18   component stack port( -- declaración del stack
19      CLK,  FE,PUSH,POP:  in std_logic;
20       PC:   inout std_logic_vector(3 downto 0);
21       ST:   inout std_logic_vector(3 downto 0));
22   end component;

24    component mux_4 port( — declaración del multiplexor
25      D,R,ST,PC:  in std_logic_vector(3 downto 0);
26              S:  in std_logic_vector(1 downto 0);
27              Y:  buffer std_logic_vector (3 downto 0));
28   end component;
29 end comps_sec;
```

Listado 7.5 Creación del paquete que contiene los componentes del secuenciador.

7.5 Diseño del programa de alto nivel (Top Level)

Por último y como señalamos desde el principio, el algoritmo que une cada bloque o componente de diseño interconectado a través de señales o buses internos se denomina Top Level (Fig. 7.9).



a)

Figura 7.9 Programa de alto nivel (Top Level).

El programa de alto nivel que realiza esta función se muestra en el listado 7.6.

En la parte inicial del programa se llama a la librería *amd* (línea 1). Esta contiene el paquete *comps_sec,* que cuenta con los componentes que se utilizarán en el diseño (creación de librerías).

```
-- Diseño de los componentes del secuenciador 2909

1  library ieee, amd;
2  use ieee.std_logic_1164.all;
3  use work.std_arith.all;
4  use amd. comps_sec. all"; --paquete creado en la librería amd
5           entity amd2909 is port (
6          R:   in std_logic_vector (3 downto 0) ;
7          D:   in std_logic_vector (3 downto 0) ;
8         ER:   in std_logic;
9        CLK:   in std_logic;
10         S:   in std_logic_vector (1 downto 0) ;
11        FE:   in std_logic;
12      PUSH:   in std_logic;
13       POP:   in std_logic;
14       CIN:   in std_logic;
15      COUT:   inout std_logic;
16         Y:   buffer std_logic_vector (3 downto 0));
17     end amd2909;
```

Continúa

```
18
19 architecture arq_amd of amd2909 is
20  signal REG: std_logic_vector (3 downto 0);
21  signal ST:  std_logic_vector (3 downto 0);
22  signal PC: std_logic_vector (3 downto 0);

23  begin
24
25      -- inicia interconexión de los componentes
26
27  ul: registro port map (CLK => CLK, ER => ER, REG => REG, R => R);
28  u2: mpc    port map (CIN=>CIN, COUT=>COUT, CLK=>CLK, Y=>Y,
29                                PC=>PC);
30  u3: stack  port map (CLK =>CLK, SE => SE, PUSH=>PUSH, POP => 31
31                                POP, MPC => MPC, ST => ST) ;
32  u4: mux_4  port map (D=>D, R=>R, ST=>ST, PC=>PC, S=>S, Y=>Y);
33
34 end arq_sec;
```

Listado 7.6 Creación del programa principal.

La entidad se declara asignando las terminales de entrada y salida del secuenciador, las cuales se nombran tal como están referidas en su módulo (líneas 6 a 16). A continuación las señales que interconectan cada uno de los módulos se declaran dentro de la arquitectura (líneas 20 a 22). Note que estas señales no tienen asignada alguna terminal del dispositivo.

La segunda parte del código hace referencia a la conexión de los distintos componentes utilizando la cláusula port map (líneas 27 a 32). Además, u1, u2, u3 y u4 se denominan etiquetas de asignación inmediata. En cada asignación, el símbolo = > se usa para asociar (map) las señales actuales (es decir las que conforman la entidad amd2909) con las locales (los puertos que componen cada módulo de diseño). Una vez que se conecta cada módulo, el diseño se compila, generando un archivo .jed, que contiene todos los componentes creados.

Es importante resaltar en este programa la función de las terminales de entrada/salida (inout). Estas se utilizan para acoplar los diferentes bloques, debido a que son salidas de un bloque y entradas a un bloque diferente.

7.6 Creación de una librería en Warp

Como se puede apreciar, en el listado anterior (listado 7.6) se utilizó una librería llamada *amd,* la cual se creó para almacenar el paquete *comps_sec* que