Lenguajes de Descripción de Hardware

Ejemplo HDL 3-1

```
//Descripción del circuito simple de la fig. 3-37
module circuito_smpl(A,B,C,x,y);
    input A,B,C;
    outputx,y;
    wire e;
    and g1(e,A,B);
    not g2(y, C);
    or g3(x,e,y);
endmodule
```

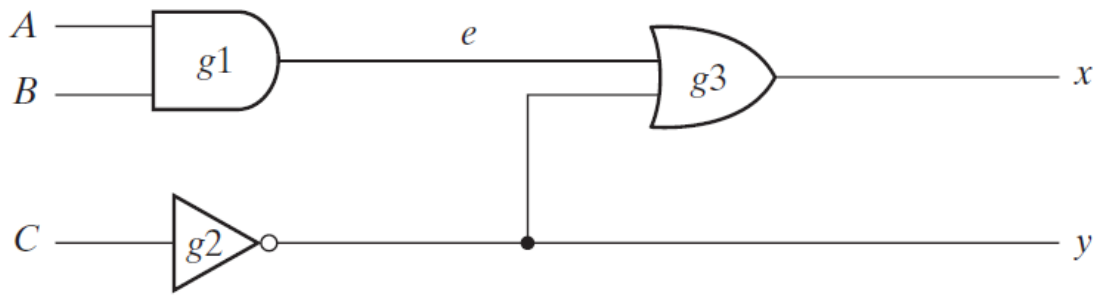


FIGURA 3-37
Circuito para ilustrar HDL

Lenguajes de Descripción de Hardware

Retardos de compuerta

Cuando se usa HDL para hacer simulaciones, tal vez sea necesario especificar la magnitud del retardo que hay entre la entrada y la salida de las compuertas. En Verilog, el retardo se especifica en términos de *unidades de tiempo* y el símbolo #. La asociación de la unidad de tiempo con el tiempo físico se efectúa con la directriz de compilador `**timescale.** (Las directrices al compilador inician con el símbolo ` (acento grave).) Tales directrices se especifican antes de declarar módulos. Un ejemplo de directriz de escala de tiempo es:

`timescale 1ns/100ps

Lenguajes de Descripción de Hardware

Ejemplo HDL 3-2

```
//Descripción de circuito con retardo
module circuito_con_retardo(A,B,C,x,y);
    input A,B,C;
    output x,y;
    wire e;
    and #(30) g1(e,A,B);
    or #(20) g3(x,e,y);
    not #(10) g2(y,C);
endmodule
```

Lenguajes de Descripción de Hardware

Tabla 3-6
Salida de las compuertas después del retardo

	Unidades de tiempo	Entrada	Salida
	(ns)	ABC	*y* e *x*
Inicial	–	000	1 0 1
Cambio	–	111	1 0 1
	10	111	0 0 1
	20	111	0 0 1
	30	111	0 1 0
	40	111	0 1 0
	50	111	0 1 1

Lenguajes de Descripción de Hardware

Ejemplo HDL 3-3

```
//Estímulo para el circuito simple
module stimcrct;
reg A,B,C;
wire x,y;
circuito_con_retardo ccr(A,B,C,x,y);
initial
   begin
        A = 1'b0; B = 1'b0; C = 1'b0;
     #100
        A = 1'b1; B = 1'b1; C = 1'b1;
     #100  $finish;
   end
endmodule

//Descripción de circuito con retardo
module circuito_con_retardo (A,B,C,x,y);
   input A,B,C;
   output x,y;
   wire e;
   and #(30) g1(e,A,B);
   or  #(20) g3(x,e,y);
   not #(10) g2(y,C);
endmodule
```

Lenguajes de Descripción de Hardware

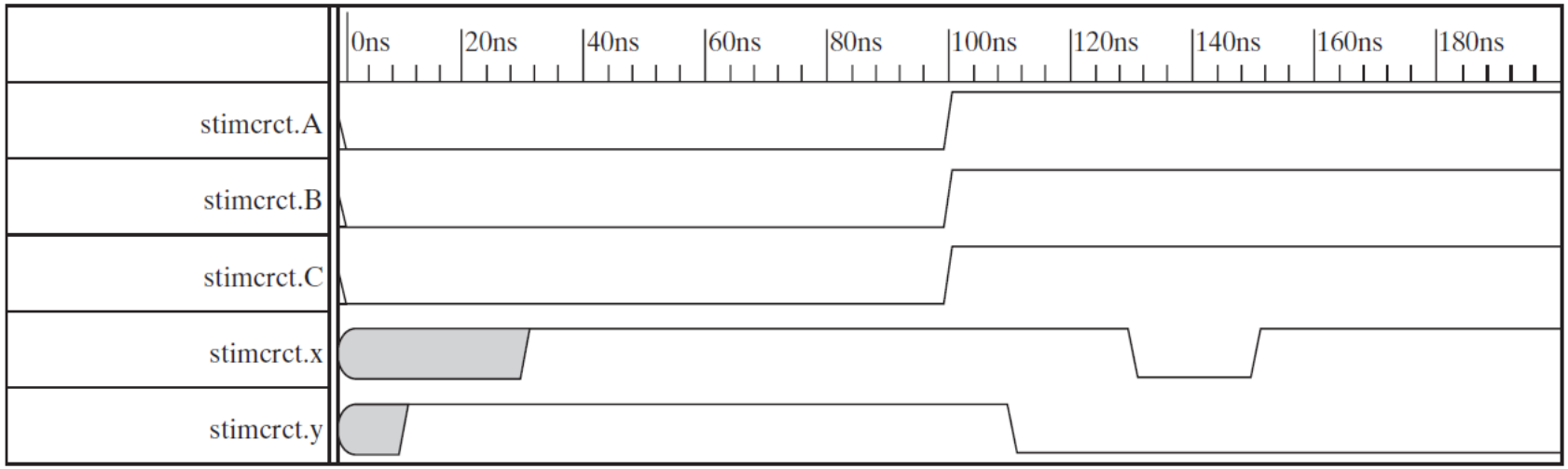


FIGURA 3-38
Salida de la simulación del ejemplo HDL 3-3

Lenguajes de Descripción de Hardware

Ejemplo HDL 3-4

```
//Circuito especificado con expresiones booleanas
module circuit_bln (x,y,A,B,C,D);
    input A,B,C,D;
    output x,y;
    assign x = A | (B & C) | (~B & D);
    assign y = (~B & C) | (B & ~C & ~D);
endmodule
```

Lenguajes de Descripción de Hardware

Ejemplo HDL 3-5

```
//Primitiva definida por el usuario (UDP)
primitive crctp (x,A,B,C);
   output x;
   input A,B,C;
//Tabla de verdad para x(A,B,C) = Σ(0,2,4,6,7)
   table
//     A   B   C  :  x   (Esto es sólo un comentario)
       0   0   0  :  1;
       0   0   1  :  0;
       0   1   0  :  1;
       0   1   1  :  0;
       1   0   0  :  1;
       1   0   1  :  0;
       1   1   0  :  1;
       1   1   1  :  1;
   endtable
endprimitive

//Crear una copia de la primitiva
module declare_crctp;
  reg x,y,z;
  wire w;
  crctp (w,x,y,z);
endmodule
```

Lenguajes de Descripción de Hardware

- Modelado a nivel de compuertas y primitivas
- Modelado de flujo de datos (assign)
- Modelado de comportamiento (always)

Lenguajes de Descripción de Hardware
Modelado a nivel de Compuertas

- Modelado a nivel de Compuertas

Tabla 4-9
Tablas de verdad para las compuertas primitivas predefinidas

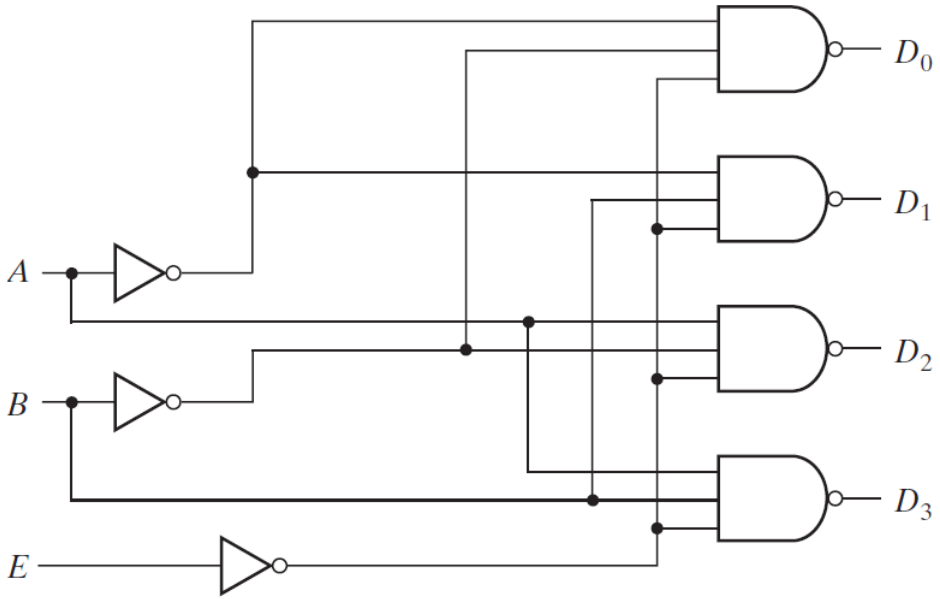
and	0	1	x	z
0	0	0	0	0
1	0	1	x	x
x	0	x	x	x
z	0	x	x	x

or	0	1	x	z
0	0	1	x	x
1	1	1	1	1
x	x	1	x	x
z	x	1	x	x

xor	0	1	x	z
0	0	1	x	x
1	1	0	x	x
x	x	x	x	x
z	x	x	x	x

not	Entrada	Salida
	0	1
	1	0
	x	x
	z	x

Lenguajes de Descripción de Hardware
Modelado a nivel de Compuertas



a) Diagrama lógico

E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

b) Tabla de verdad

FIGURA 4-19
Decodificador de 2 a 4 líneas con entrada habilitadora

Lenguajes de Descripción de Hardware
Modelado a nivel de Compuertas

Ejemplo HDL 4-1

```
//Descripción a nivel de compuertas del decodificador 2 a 4
//de la figura 4-19
module decoder_gl (A,B,E,D);
    input A,B,E;
    output [0:3]D;
    wire Anot,Bnot,Enot;
    not
        n1 (Anot,A),
        n2 (Bnot,B),
        n3 (Enot,E);
    nand
        n4 (D[0],Anot,Bnot,Enot),
        n5 (D[1],Anot,B,Enot),
        n6 (D[2],A,Bnot,Enot),
        n7 (D[3],A,B,Enot);
endmodule
```

Lenguajes de Descripción de Hardware
Modelado a nivel de Compuertas

Subíndice i:	**3**	**2**	**1**	**0**	
Acarreo de entrada	0	1	1	0	C_i
Sumando	1	0	1	1	A_i
Sumando	0	0	1	1	B_i
Suma	1	1	1	0	S_i
Acarreo de salida	0	0	1	1	C_{i+1}

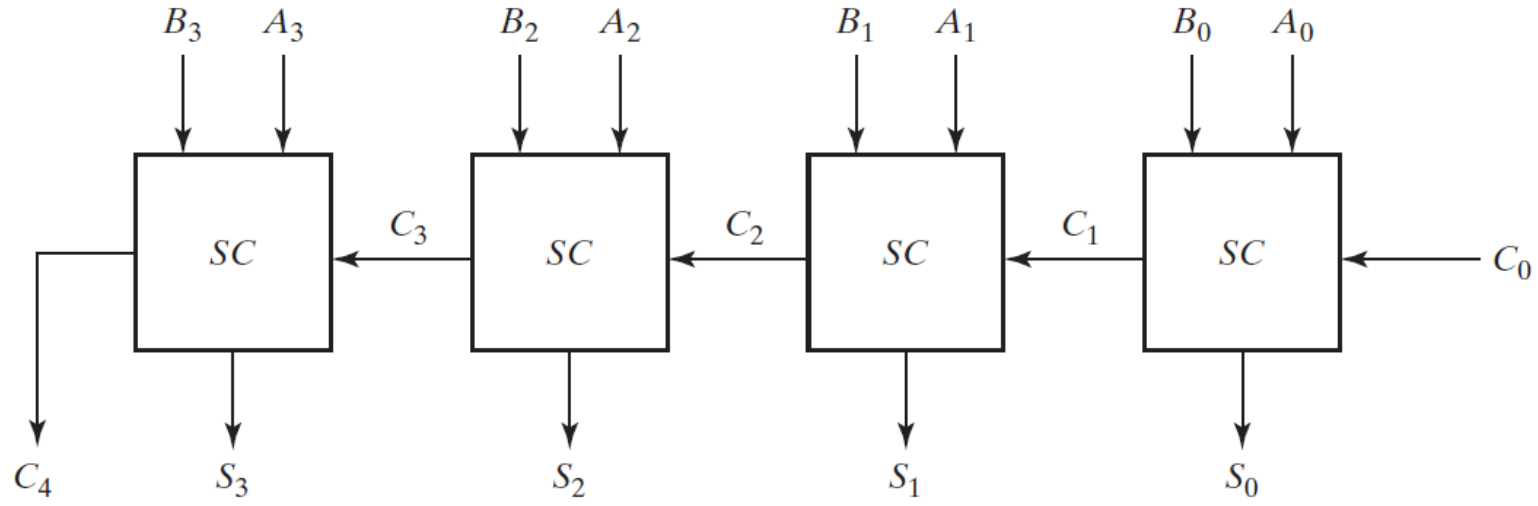
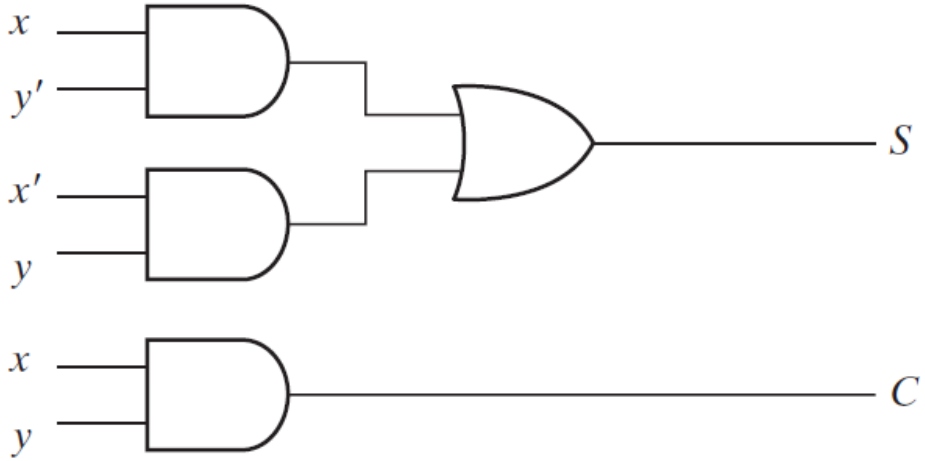
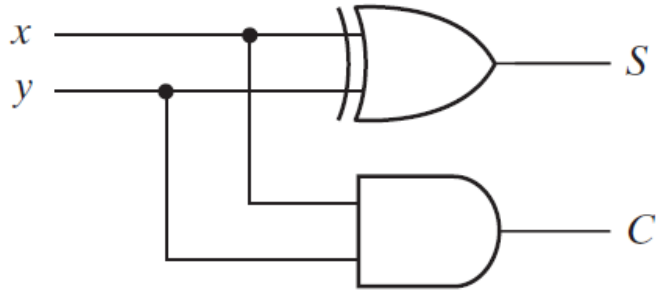


FIGURA 4-9
Sumador de cuatro bits

Lenguajes de Descripción de Hardware
Modelado a nivel de Compuertas



a) $S = xy' + x'y$
$C = xy$



b) $S = x \oplus y$
$C = xy$

FIGURA 4-5
Implementación de semisumador

Lenguajes de Descripción de Hardware
Modelado a nivel de Compuertas

Ejemplo HDL 4-2

```
//Descripción jerárquica a nivel de compuertas de un sumador de 4 bits
//Descripción del semisumador (véase la figura 4-5b)
module halfadder (S,C,x,y);
    input x,y;
    output S,C;
//Crear ejemplares de compuertas primitivas
    xor (S,x,y);
    and (C,x,y);
endmodule
```

Lenguajes de Descripción de Hardware
Modelado a nivel de Compuertas

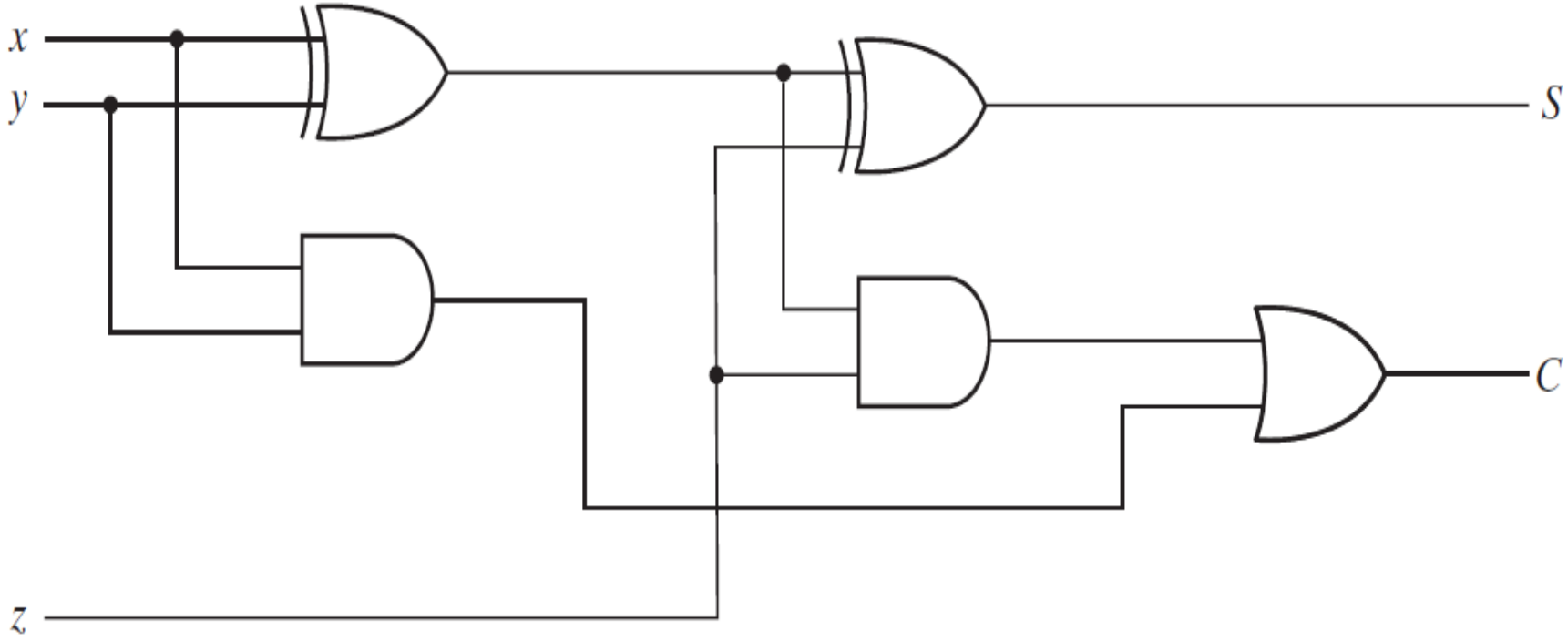


FIGURA 4-8

Implementación de un sumador completo con dos semisumadores y una compuerta OR

Lenguajes de Descripción de Hardware
Modelado a nivel de Compuertas

```
//Descripción de sumador completo (véase la figura 4-8)
module fulladder (S,C,x,y,z);
   input x,y,z;
   output S,C;
   wire S1,D1,D2; //Salidas de primera XOR y dos compuertas AND
//Crear un ejemplar del semisumador
    halfadder HA1 (S1,D1,x,y),
              HA2 (S,D2,S1,z);
    or g1(C,D2,D1);
endmodule
```

Lenguajes de Descripción de Hardware
Modelado a nivel de Compuertas

Subíndice i:	**3**	**2**	**1**	**0**	
Acarreo de entrada	0	1	1	0	C_i
Sumando	1	0	1	1	A_i
Sumando	0	0	1	1	B_i
Suma	1	1	1	0	S_i
Acarreo de salida	0	0	1	1	C_{i+1}

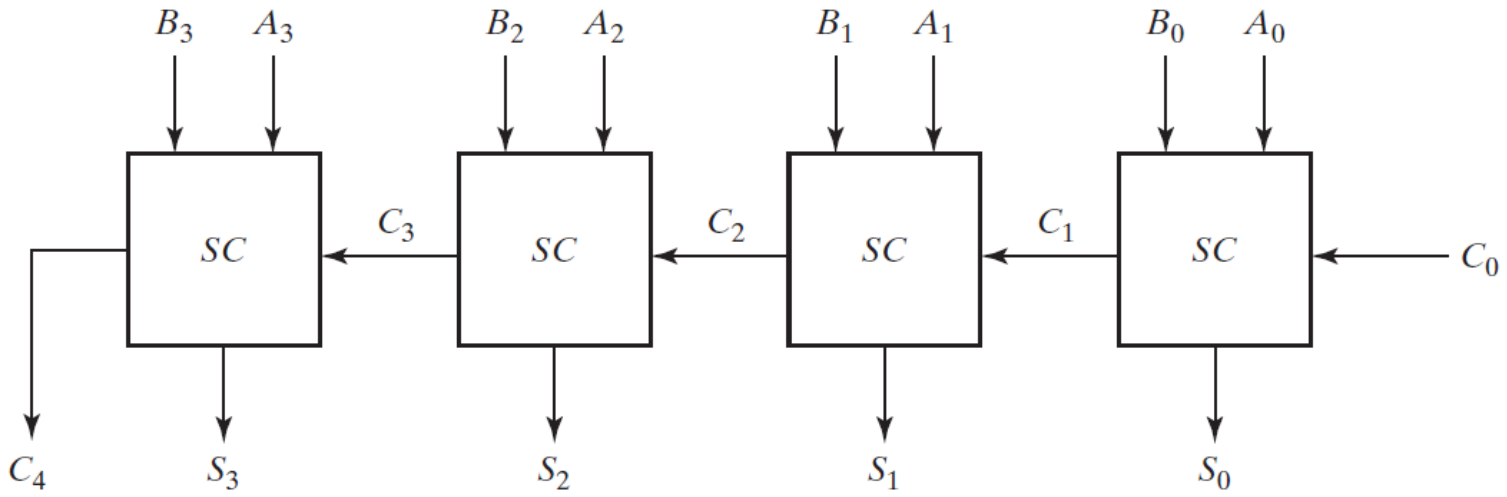


FIGURA 4-9
Sumador de cuatro bits

Lenguajes de Descripción de Hardware
Modelado a nivel de Compuertas

```
//Descripción del sumador de 4 bits (véase la figura 4-9)
module _4bit_adder (S,C4,A,B,C0);
    input [3:0] A,B;
    input C0;
    output [3:0] S;
    output C4;
    wire C1,C2,C3;  //Acarreos intermedios
//Crear un ejemplar del sumador completo
    fulladder  FA0 (S[0],C1,A[0],B[0],C0),
               FA1 (S[1],C2,A[1],B[1],C1),
               FA2 (S[2],C3,A[2],B[2],C2),
               FA3 (S[3],C4,A[3],B[3],C3);
endmodule
```

Lenguajes de Descripción de Hardware
Modelado a nivel de Compuertas

- Lógica con tercer estado.

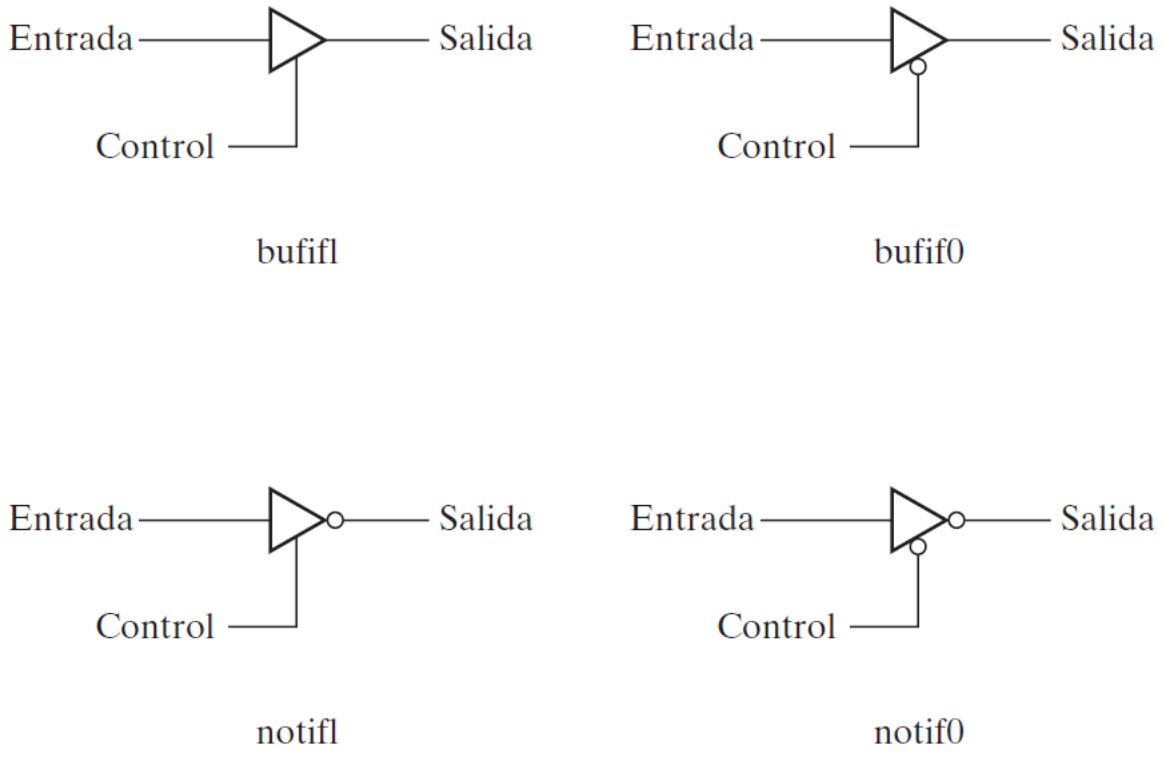


FIGURA 4-31
Compuertas de tres estados

Lenguajes de Descripción de Hardware
Modelado a nivel de Compuertas

- Lógica con tercer estado.

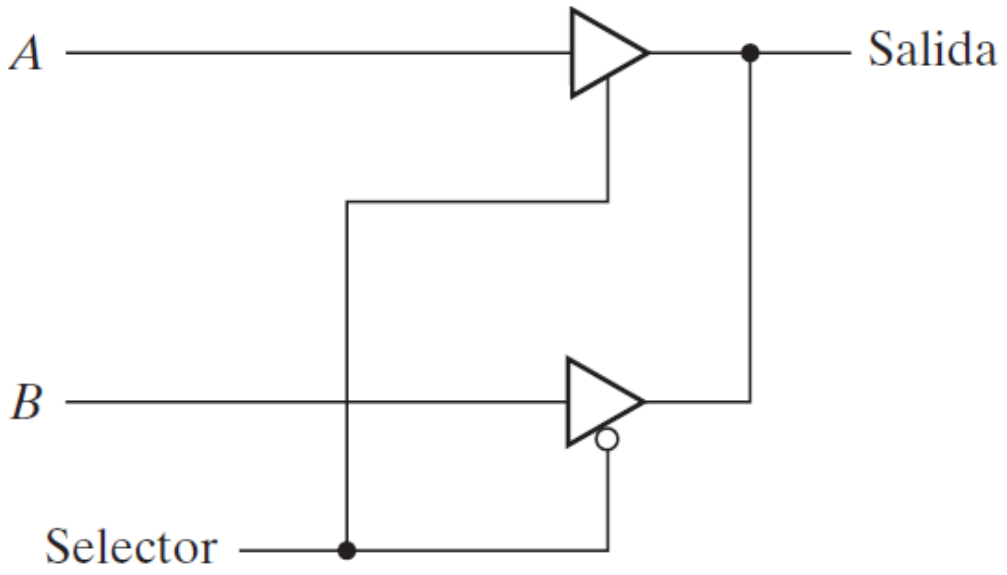


FIGURA 4-32
Multiplexor de 2 líneas a 1 con búferes de tres estados

Lenguajes de Descripción de Hardware
Modelado a nivel de Compuertas

- Lógica con tercer estado.

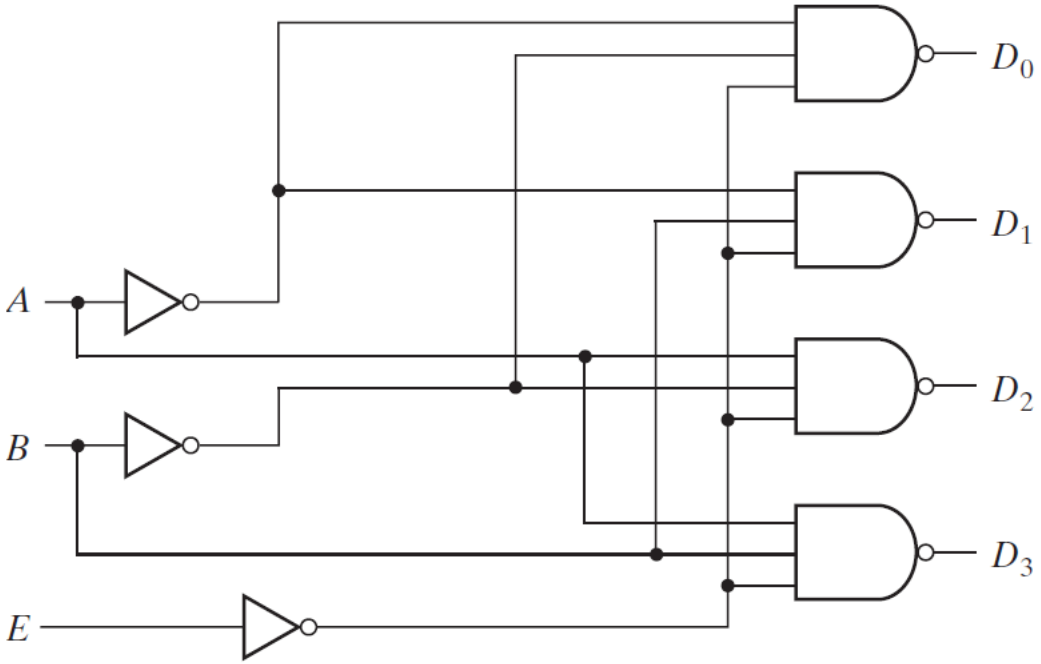
```
module muxtri (A,B,select,OUT);
    input A,B,select;
    output OUT;
    tri OUT;
    bufif1 (OUT,A,select);
    bufif0 (OUT,B,select);
endmodule
```

Lenguajes de Descripción de Hardware
Modelado de Flujo de Datos

Tabla 4-10
Operadores de Verilog HDL

Símbolo	Operación
+	Suma binaria
−	Resta binaria
&	AND bit por bit
|	OR bit por bit
∧	XOR bit por bit
~	NOT bit por bit
= =	Igualdad
>	Mayor que
<	Menor que
{ }	Concatenación
? :	Condicional

Lenguajes de Descripción de Hardware
Modelado de Flujo de Datos



a) Diagrama lógico

E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

b) Tabla de verdad

FIGURA 4-19

Lenguajes de Descripción de Hardware
Modelado de Flujo de Datos

Ejemplo HDL 4-3

```
//Descripción de flujo de datos del decodificador de 2 a 4
//Véase la figura 4-9
module decoder_df (A,B,E,D);
    input A,B,E;
    output [0:3] D;
    assign D[0] = ~(~A & ~B & ~E),
           D[1] = ~(~A & B & ~E),
           D[2] = ~(A & ~B & ~E),
           D[3] = ~(A & B & ~E);
endmodule
```

Lenguajes de Descripción de Hardware
Modelado de Flujo de Datos

Ejemplo HDL 4-4

```
//Descripción de flujo de datos de un sumador de 4 bits
module binary_adder (A,B,Cin,SUM,Cout);
    input [3:0] A,B;
    input Cin;
    output [3:0] SUM;
    output Cout;
    assign {Cout,SUM} = A + B + Cin;
endmodule
```

Lenguajes de Descripción de Hardware
Modelado de Flujo de Datos

Ejemplo HDL 4-5

```
//Descripción de flujo de datos de un comparador de 4 bits
module magcomp (A,B,ALSB,AGTB,AEQB);
    input [3:0] A,B;
    output ALTB,AGTB,AEQB;
    assign ALTB=(A < B),
           AGTB = (A > B),
           AEQB = (A == B);
endmodule
```

Lenguajes de Descripción de Hardware
Modelado de Flujo de Datos

Ejemplo HDL 4-6

```
//Descripción de flujo de datos del multiplexor 2 a 1
module mux2x1_df (A,B,select,OUT);
    input A,B,select;
    output OUT;
    assign OUT = select ? A : B;
endmodule
```

Lenguajes de Descripción de Hardware
Modelado de Comportamiento

Ejemplo HDL 4-7

```
//Descripción de comportamiento del multiplexor 2 a 1
module mux2x1_bh(A,B,select,OUT);
    input A,B,select;
    output OUT;
    reg OUT;
    always @ (select or A or B)
          if (select == 1) OUT = A;
          else OUT = B;
endmodule
```

Lenguajes de Descripción de Hardware
Modelado de Comportamiento

Ejemplo HDL 4-8

```
//Descripción de comportamiento del multiplexor 4 a 1
//Describe la tabla de función de la figura 4-25b).
module mux4x1_bh (i0,i1,i2,i3,select,y);
    input i0,i1,i2,i3;
    input [1:0] select;
    output y;
    reg y;
    always @ (i0 or i1 or i2 or i3 or select)
              case (select)
                 2'b00: y = i0;
                 2'b01: y = i1;
                 2'b10: y = i2;
                 2'b11: y = i3;
              endcase
endmodule
```

TEST BENCHS

module nombreprueba.

Declarar identificadores locales **reg** y **wire.**

Crear ejemplares del módulo de diseño a probar.

Generar estímulos con enunciados **initial** y **always.**

Exhibir la respuesta de salida.

endmodule

TEST BENCHS

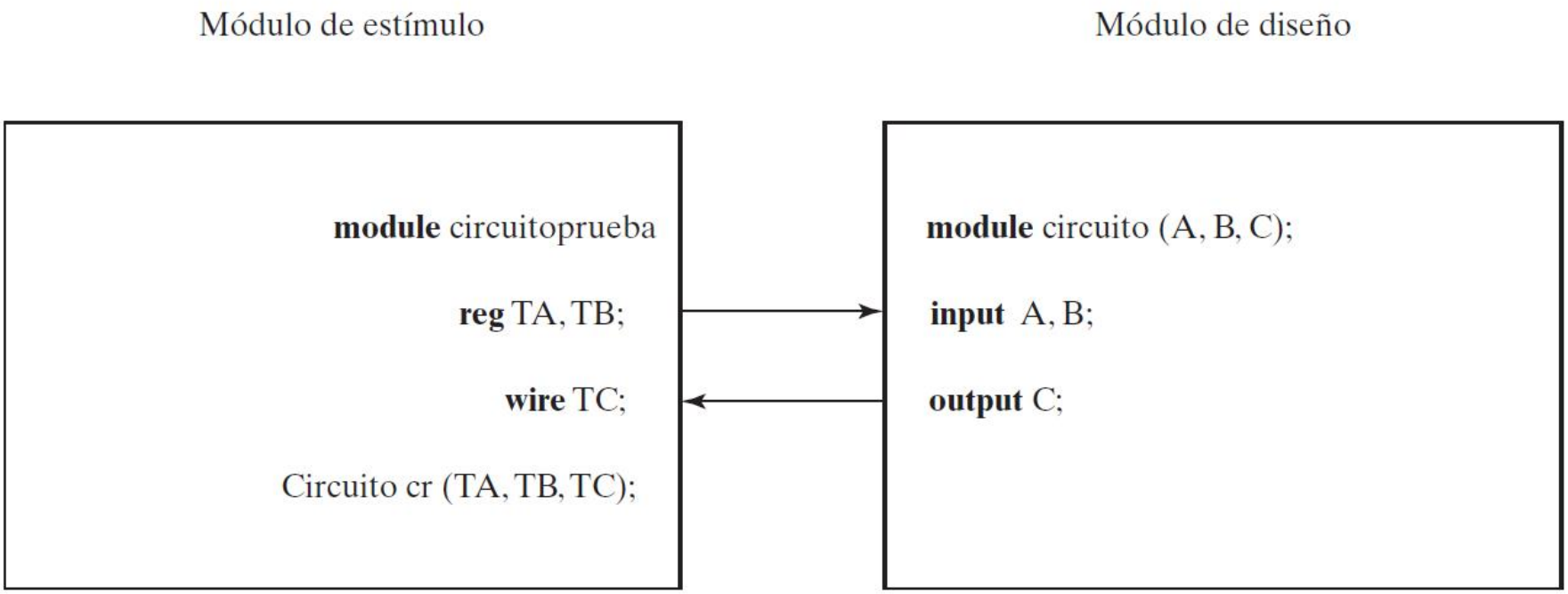


FIGURA 4-33
Interacción de los módulos de estímulo y de diseño

TEST BENCHS

$display—exhibir una vez el valor de variables o cadenas con un retorno de fin de línea,
$write**—igual que **$display pero sin pasar a la siguiente línea,
$monitor—exhibe variables cada vez que un valor cambia durante una simulación,
$time—muestra el tiempo de simulación,
$finish—termina la simulación.

La sintaxis de **$display, $write** y **$monitor** tiene la forma

Nombre-tarea (especificación de formato, lista argumentos);

```
$display (%d %b %b, C,A,B);
```

$display ("tiempo = %0d A = %b B = %b", **$tiempo**,A,B);

```
tiempo = 3   A = 10   B = 1
```

TEST BENCHS

```
//Estímulo para mux2x1_df.
module testmux;
  reg TA,TB,TS;  //entradas para mux
  wire Y;        //salida de mux
  mux2x1_df mx (TA,TB,TS,Y);  // crear un ejemplar mux
     initial
        begin
              TS = 1; TA = 0; TB = 1;
          #10 TA = 1; TB = 0;
          #10 TS = 0;
          #10 TA = 0; TB = 1;
        end
     initial
      $monitor("select = %b A = %b B = %b OUT = %b time = %0d",
               TS, TA, TB, Y, $time);
endmodule
```

TEST BENCHS

```
//Descripción de flujo de datos de multiplexor de 2 a 1
//del ejemplo 4-6
module mux2x1_df (A,B,select,OUT);
    input A,B,select;
    output OUT;
    assign OUT = select ? A : B;
endmodule

Simulation log:

select = 1 A = 0 B = 1 OUT = 0 tiempo = 0
select = 1 A = 1 B = 0 OUT = 1 tiempo = 10
select = 0 A = 1 B = 0 OUT = 0 tiempo = 20
select = 0 A = 0 B = 1 OUT = 1 tiempo = 30
```

TEST BENCHS

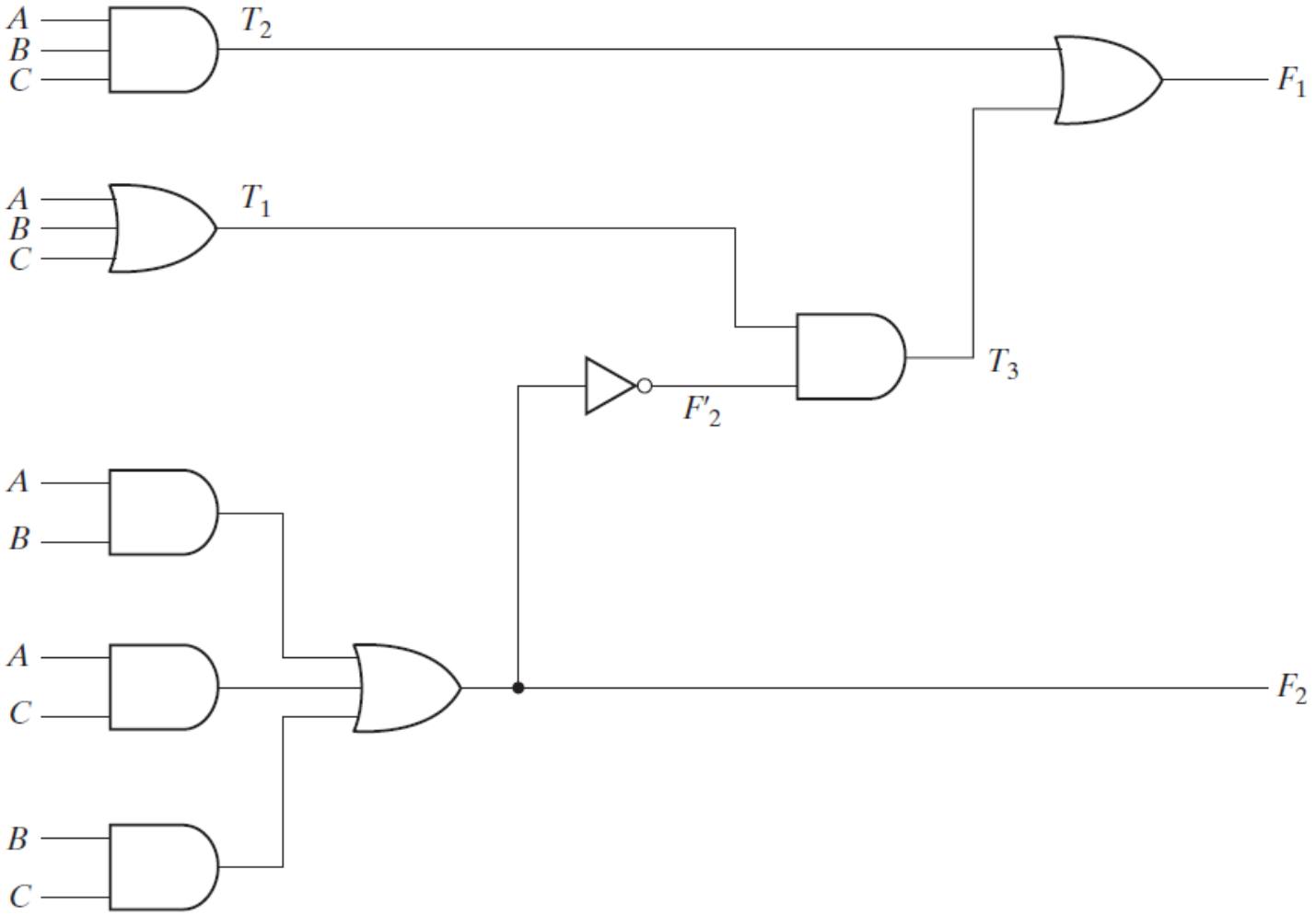


FIGURA 4-2
Diagrama lógico para el ejemplo de análisis

TEST BENCHS

Ejemplo HDL 4-10

```
//Descripción a nivel de compuertas del circuito de la figura 4-2
module analysis (A,B,C,F1,F2);
   input   A,B,C;
   output  F1,F2;
   wire    T1,T2,T3,F2not,E1,E2,E3;
   or  g1 (T1,A,B,C);
   and g2 (T2,A,B,C);
   and g3 (E1,A,B);
   and g4 (E2,A,C);
   and g5 (E3,B,C);
   or  g6 (F2,E1,E2,E3);
   not g7 (F2not,F2);
   and g8 (T3,T1,F2not);
   or  g9 (F1,T2,T3);
endmodule
```

TEST BENCHS

```
//Estímulo para analizar el circuito
module test_circuit;
  reg [2:0]D;
  wire F1,F2;
  analysis fig42(D[2],D[1],D[0],F1,F2);
  initial
    begin
     D = 3'b000;
     repeat(7)
      #10 D = D + 1'b1;
    end
  initial
     $monitor ("ABC = %b F1 = %b F2 =%b ",D, F1, F2);
endmodule
```

TEST BENCHS

```
Simulation log:

ABC = 000 F1 = 0 F2 =0
ABC = 001 F1 = 1 F2 =0
ABC = 010 F1 = 1 F2 =0
ABC = 011 F1 = 0 F2 =1
ABC = 100 F1 = 1 F2 =0
ABC = 101 F1 = 0 F2 =1
ABC = 110 F1 = 0 F2 =1
ABC = 111 F1 = 1 F2 =1
```