



Instituto Politécnico Nacional Centro de Investigación en Computación

Lenguajes de descripción de hardware

Tarea 1 - Sumadores

PROFESOR:

M. EN C. OSVALDO ESPINOZA SOSA

Por:

ING. RICARDO ALDAIR TIRADO TORRES

Tabla de contenido

1.	Objetivos	2
2.	Sumador con acarreo de entrada	3
3.	Sumador completo	5
4.	Sumador/restador	7
5.	Conclusiones	9
6.	Anexos	11
	6.1. Descripciones del hardware	11
	6.2 Rancos de pruebas (Test Renches)	14

1. Objetivos

- Aprender acerca de los elementos básicos en los lenguajes de descripción, como lo son los comentarios, las palabras reservadas, los números y bases numéricas y los operadores lógicos, aritméticos y relacionales.
- Entender la operación de concatenación y su importancia en operaciones como la suma.
- Aplicar los conocimientos anteriores para la generación de circuitos combinatorios como el sumador con acarreo de entrada, el sumador completo y el sumador/restador.

2. Sumador con acarreo de entrada

Actividad 1

Completar el código de la lámina 10 para implementar el sumador con acarreo de entrada (Ci) y compilar. Usar el visor RTL para verificar cómo se implementa dicho sumador. ¿Cuántos sumadores se utilizan? Utilizar los dos lenguajes.

La visualización RTL del sumador con acarreo de entrada en VHDL se muestra en la Figura 1 y en Verilog en la Figura 2. Como se observa, si se realiza una descripción por comportamiento, el entorno utiliza dos instancias de sumadores de 2 entradas, ya que en el primero realiza la suma de A y B (ambos de 8 bits), mientras que el segundo sumador realiza la operación entre el resultado anterior y el acarreo de entrada (nótese que al ser el acarreo de entrada de un solo bit, pone los otros 7 bits restantes en cero con GND). Las simulaciones para el código en VHDL se visualizan en la Figura 3 en base binaria y en la Figura 4 en base decimal. En cambio, las simulaciones para el código en Verilog se visualizan en la Figura 5 en base binaria y en la Figura 6 en base decimal. Cabe resaltar que los valores utilizados para los bancos de pruebas fueron obtenidos de un generador de números aleatorios entre 0 y 255 [1].

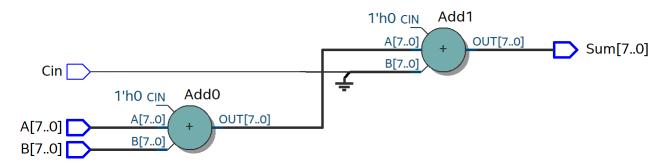


Figura 1: Diagrama RTL del sumador con acarreo de entrada en VHDL.

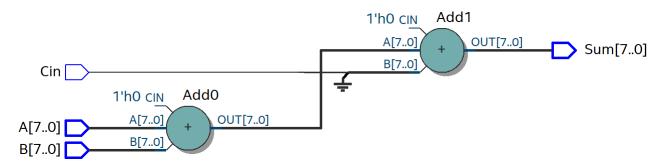


Figura 2: Diagrama RTL del sumador con acarreo de entrada en Verilog.



Figura 3: Simulación del sumador con acarreo de entrada en VHDL con el visor de formas de onda de ModelSim (Base binaria).

/adder_cin_vhdl_vhd_tst/Cin	0												
/adder_cin_vhdl_vhd_tst/A	0	0	135	122	178	170	23	0	135	122	178	170	23
/adder_cin_vhdl_vhd_tst/B	0	0	206	52	18	115	77	0	206	52	18	115	77
/adder_cin_vhdl_vhd_tst/Sum_	0	0	85	174	196	29	100	1	86	175	197	30	101

Figura 4: Simulación del sumador con acarreo de entrada en VHDL con el visor de formas de onda de ModelSim (Base decimal).



Figura 5: Simulación del sumador con acarreo de entrada en Verilog con el visor de formas de onda de ModelSim (Base binaria).

/Adder_Cin_Verilog_vlg_tst/Cin	-No												
/Adder_Cin_Verilog_vlg_tst/A	-No	0	135	122	178	170	23	0	135	122	178	170	23
/Adder_Cin_Verilog_vlg_tst/B	-No	0	206	52	18	115	77	0	206	52	18	115	77
/Adder_Cin_Verilog_vlg_tst/Sum	-No	0	85	174	196	29	100	1	86	175	197	30	101

Figura 6: Simulación del sumador con acarreo de entrada en Verilog con el visor de formas de onda de ModelSim (Base decimal).

3. Sumador completo

Actividad 2

Codificar el sumador completo (con acarreo de entrada y acarreo de salida) indicado en la lámina 11, en los dos lenguajes y compilar. Alguno presenta algún error durante la compilación. ¿Cómo se soluciona el problema?

Se presenta un error en VHDL, como se observa en la Figura 7, esto surge debido a que se necesita que los operandos sean del mismo tamaño que la variable en donde se almacenará el resultado. Para solucionarlo, unicamente se necesita concatenar un bit, en bajo, en la posición más significativa de ambos operandos. Este error no sucede con Verilog, ya que el propio lenguaje concatena el bit faltante automáticamente.

```
    10344 VHDL expression error at Full_Adder_VHDL.vhd(18): expression has 8 elements, but must have 9 elements
    12153 Can't elaborate top-level user hierarchy
    Quartus Prime Analysis & Synthesis was unsuccessful. 2 errors, 1 warning
    293001 Quartus Prime Full Compilation was unsuccessful. 4 errors, 1 warning
```

Figura 7: Error obtenido al utilizar el código de la lámina 11 sin modificar.

La visualización RTL del sumador completo en VHDL se muestra en la Figura 8 y en Verilog en la Figura 9. Las simulaciones para el código en VHDL se visualizan en la Figura 10 en base binaria y en la Figura 11 en base decimal. En cambio, las simulaciones para el código en Verilog se visualizan en la Figura 12 en base binaria y en la Figura 13 en base decimal. Los valores empleados son los mismos que los de la Actividad 1.

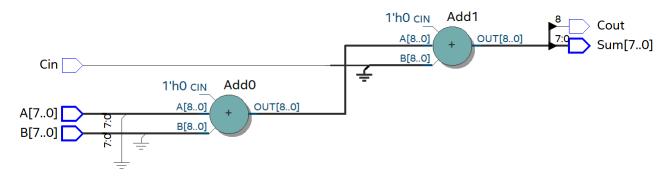


Figura 8: Diagrama RTL del sumador completo en VHDL.

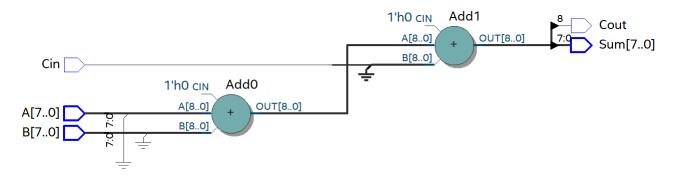


Figura 9: Diagrama RTL del sumador completo en Verilog.

/full_adder_vhdl_vhd_tst/Cin	0												
/full_adder_vhdl_vhd_tst/A	00000000	00000000	(10000111	01111010	(10110010	10101010	00010111	00000000	110000111	01111010	(10110010	10101010	(00010111
	00000000	00000000	(11001110	[00110100	(00010010	[01110011	01001101	00000000	1110011110	00110100	(00010010	01110011	(01001101
	000000000	000000000	(101010101	[010101110	(011000100	100011101	001100100	000000001	101010110	010101111	(011000101	100011110	(001100101
	00000000	00000000	(01010101	10101110	11000100	100011101	01100100	00000001	[01010110	10101111	11000101	00011110	(01100101
	0												

Figura 10: Simulación del sumador completo en VHDL con el visor de formas de onda de ModelSim (Base binaria).

/full_adder_vhdl_vhd_tst/Cin	0												
/full_adder_vhdl_vhd_tst/A	0	0	-121	122	-78	-86	23	0	-121	122	-78	-86	23
/full_adder_vhdl_vhd_tst/B	0	0	-50	52	18	115	77	0	-50	52	18	115	77
/full_adder_vhdl_vhd_tst/i1/Temp	0	0	-171	174	196	-227	100	1	-170	175	197	-226	101
/full_adder_vhdl_vhd_tst/Sum	0	0	85	-82	-60	29	100	1	86	-81	-59	30	101
/full_adder_vhdl_vhd_tst/Cout	0												

Figura 11: Simulación del sumador completo en VHDL con el visor de formas de onda de ModelSim (Base decimal).

	00000000	0000000	(10000111	01111010	10110010	10101010	00010111	00000000	10000111	01111010	(10110010	10101010	00010111
/Full_Adder_Verilog_vlg_tst/B	00000000	00000000	(11001110	00110100	[00010010	01110011	01001101	00000000	(11001110	00110100	(00010010	01110011	01001101
/Full_Adder_Verilog_vlg_tst/i1/Temp	000000000	00000000	(101010101	010101110	[011000100	100011101	(001100100	000000001	X 101010110	(010101111	(011000101	100011110	(001100101
/Full_Adder_Verilog_vlg_tst/Sum	00000000	00000000	(01010101	10101110	[11000100	00011101	(01100100	00000001	(01010110	10101111	(11000101	00011110	(01100101
/Full_Adder_Verilog_vlg_tst/Cout	St0												

Figura 12: Simulación del sumador completo en Verilog con el visor de formas de onda de ModelSim (Base binaria).

/Full_Adder_Verilog_vlg_tst/Cin	0												
/Full_Adder_Verilog_vlg_tst/A	0	0	135	122	178	170	23	0	135	122	178	170	23
<pre>/Full_Adder_Verilog_vlg_tst/B</pre>	0	0	206	52	18	115	77	0	206	52	18	115	77
/ /Full_Adder_Verilog_vlg_tst/i1/Temp	0	0	341	174	196	285	100	1	342	175	197	286	101
<pre>/Full_Adder_Verilog_vlg_tst/Sum</pre>	0	0	85	174	196	29	100	1	86	175	197	30	101
/Full_Adder_Verilog_vlg_tst/Cout	St0												

Figura 13: Simulación del sumador completo en Verilog con el visor de formas de onda de ModelSim (Base decimal).

4. Sumador/restador

Actividad 3

Codificar el sumador/restador presentado en la lámina 12 y verificar como se implementa utilizando el visor RTL.

La visualización RTL del sumador/restador en VHDL se muestra en la Figura 14 y en Verilog en la Figura 15. Como se observa, la implementación del sumador/restador se hace utilizando un multiplexor que selecciona la salida deseada con la variable C. Las simulaciones para el código en VHDL se visualizan en la Figura 16 en base binaria y en la Figura 17 en base decimal. En cambio, las simulaciones para el código en Verilog se visualizan en la Figura 18 en base binaria y en la Figura 19 en base decimal. Los valores empleados son los mismos que los de la Actividad 1.

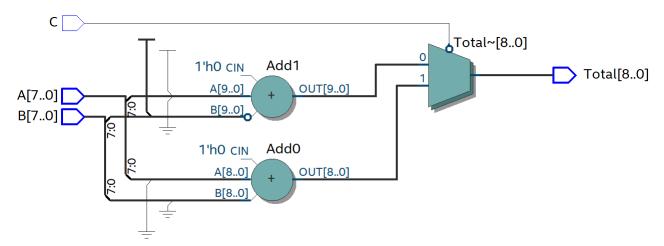


Figura 14: Diagrama RTL del sumador/restador en VHDL.

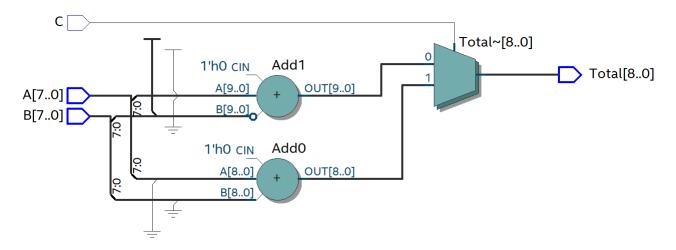


Figura 15: Diagrama RTL del sumador/restador en Verilog.

/adder_subtractor_vhdl_vhd_tst/C	£												
/adder_subtractor_vhdl_vhd_tst/A	00010111	00000000) 10000111	(01111010	10110010	10101010	00010111	(00000000	10000111	(01111010	(10110010	(10101010	00010111
/adder_subtractor_vhdl_vhd_tst/B (01001101	00000000	11001110	(00110100	00010010	01110011	01001101	(00000000	11001110	00110100	00010010	01110011	01001101
/adder_subtractor_vhdl_vhd_tst/Total	111001010	000000000	101010101	010101110	011000100	100011101	001100100	000000000	110111001	001000110	010100000	000110111	111001010

Figura 16: Simulación del sumador/restador en VHDL con el visor de formas de onda de ModelSim (Base binaria).

/adder_subtractor_vhdl_vhd_tst/C	0												
- /adder_subtractor_vhdl_vhd_tst/A	0	0	135	122	178	170	23	(O	135	122	178	170	23
- /adder_subtractor_vhdl_vhd_tst/B	0	0	206	52	18	115	77	0	206	52	18	115	77
/adder_subtractor_vhdl_vhd_tst/Total	0	0	-171	174	196	-227	100	0	-71	70	160	55	-54
- /adder_subtractor_vhdl_vhd_tst/Total	0	0	341	174	196	285	100	(o	441	70	160	55	458

Figura 17: Simulación del sumador/restador en VHDL con el visor de formas de onda de ModelSim (Base decimal).

/Adder_Subtractor_Verilog_vlg_tst/C	0												
/Adder_Subtractor_Verilog_vlg_tst/A	00000000	00000000	10000111	[01111010	[10110010	(10101010	00010111	[00000000	[10000111	(01111010	10110010	[10101010	[00010111
// /Adder_Subtractor_Verilog_vig_tst/8	00000000	00000000	11001110	00110100	(00010010	(01110011	01001101	[00000000	[11001110	(00110100	00010010	[01110011	[01001101
/Adder_Subtractor_Verilog_vlg_tst/Total	000000000	000000000	[101010101	010101110	(011000100	100011101	001100100	000000000	[110111001	001000110	010100000	000110111	111001010

Figura 18: Simulación del sumador/restador en Verilog con el visor de formas de onda de ModelSim (Base binaria).

/Adder_Subtractor_Verilog_vlg_tst/C	0												
/Adder_Subtractor_Verilog_vlg_tst/A	0	0	135	122	178	170	23	0	135	122	178	170	23
-4> /Adder_Subtractor_Verilog_vlg_tst/B	0	0	206	52	18	115	77	0	206	52	18	115	77
	0	0	-171	174	196	-227	100	0	-71	70	160	55	-54
	0	0	341	174	196	285	100	0	441	70	160	55	458

Figura 19: Simulación del sumador/restador en Verilog con el visor de formas de onda de ModelSim (Base decimal).

5. Conclusiones

En conclusión, se implementaron los 3 circuitos en ambos lenguajes de manera exitosa.

Para el sumador con acarreo de entrada, se implementó la descripción por comportamiento, usando el operador aritmético "+" y con el visor RTL se entendió como es que el lenguaje interpreta la suma de más de dos variables de entrada.

Para el sumador completo, se comprendió como es que se debe manipular la concatenación para realizar una correcta descripción de operaciones (en este caso, la suma con acarreo de salida) y evitar errores en el tamaño de las variables utilizadas.

Para el sumador/restador, se utilizaron las sentencias *if-else-then* para generar un multiplexor, el cual selecciona que resultado visualizar en la salida, con una variable de control evaluada en las sentencias mencionadas.

En los Anexos se pueden encontrar los códigos implementados junto con sus respectivos bancos de pruebas.

Referencias

 $[1] \ \ PiliApp, \ "Generador de números aleatorios," \ https://es.piliapp.com/random/number/.$

6. Anexos

6.1. Descripciones del hardware

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity Adder_Cin_VHDL is

port( Cin : in std_logic;
    A, B : in std_logic_VECTOR(7 downto 0);
    Sum : out std_logic_VECTOR(7 downto 0));
end Adder_Cin_VHDL;

architecture behavior of Adder_Cin_VHDL is
begin
Sum <= A + B + Cin;
end behavior;</pre>
```

Programa 1: Descripción en VHDL de un sumador de 8 bits con acarreo de entrada.

```
1 library ieee;
use ieee.std_logic_1164.all;
3 use ieee.std_logic_arith.all;
4 use ieee.std_logic_unsigned.all;
6 entity Full_Adder_VHDL is
  port( Cin : in std_logic;
    A, B : in std_logic_vector(7 downto 0);
     Sum : out std_logic_vector(7 downto 0);
    Cout : out std_logic);
11 end Full_Adder_VHDL;
13 architecture behavior of Full_Adder_VHDL is
14 signal
         Temp : std_logic_vector(8 downto 0);
15 begin
process(A, B, Cin)
17 begin
   Temp <= ('0' & A) + ('0' & B) + Cin;
19 end process;
20 Sum <= Temp(7 downto 0);</pre>
```

```
Cout <= Temp(8);
def temp(8);
def temp(8);
def temp(8);
def temp(8);</pre>
```

Programa 2: Descripción en VHDL de un sumador completo de 8 bits.

```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3 use IEEE.STD_LOGIC_ARITH.ALL;
4 use IEEE.STD_LOGIC_UNSIGNED.ALL;
6 entity Adder_Subtractor_VHDL is
      Port ( C : in std_logic;
      A, B : in std_logic_vector(7 downto 0);
      Total : out std_logic_vector(8 downto 0)
      );
11 end Adder_Subtractor_VHDL;
13 architecture Behavioral of Adder_Subtractor_VHDL is
14 begin
   process(A, B, C)
15
   begin
16
   if (C = '0') then
17
     Total <= ('0' & A) + ('0' & B);
18
    else
    Total <= ('0' & A) - ('0' & B);
20
    end if;
21
   end process;
23 end Behavioral;
```

Programa 3: Descripción en VHDL de un sumador-restador de 8 bits.

```
module Adder_Cin_Verilog(
input Cin,
input [7:0] A,
input [7:0] B,
output [7:0] Sum
);

assign Sum = A + B + Cin;
endmodule
```

Programa 4: Descripción en Verilog de un sumador de 8 bits con acarreo de entrada.

```
1 module Full_Adder_Verilog (
      input
               Cin,
3
      input [7:0] A,
      input [7:0] B,
      output [7:0] Sum,
                Cout
      output
7);
   reg [8:0] Temp;
  always @(Cin, A, B)
11 begin
    Temp <= A + B + Cin;
12
   end
   assign Sum = Temp[7:0];
   assign Cout = Temp[8];
17 endmodule
```

Programa 5: Descripción en Verilog de un sumador completo de 8 bits.

```
nodule Adder_Subtractor_Verilog(
   input
            [7:0] A,
3 input
4 input
            [7:0] B,
 output reg [8:0] Total
6);
   always @(A, B, C)
   begin
    if(!C)
10
     Total <= {1'b0, A} + {1'b0, B};
11
12
     Total <= {1'b0, A} - {1'b0, B};
13
   end
14
16 endmodule
```

Programa 6: Descripción en Verilog de un sumador-restador de 8 bits.

6.2. Bancos de pruebas (Test Benches)

```
1 LIBRARY ieee;
USE ieee.std_logic_1164.all;
3 use ieee.std_logic_arith.all;
4 use ieee.std_logic_unsigned.all;
6 ENTITY Adder_Cin_VHDL_vhd_tst IS
7 END Adder_Cin_VHDL_vhd_tst;
9 ARCHITECTURE Adder_Cin_VHDL_arch OF Adder_Cin_VHDL_vhd_tst IS
  SIGNAL Cin : STD_LOGIC:='0';
SIGNAL A : STD_LOGIC_VECTOR(7 downto 0):="000000000";
  SIGNAL B : STD_LOGIC_VECTOR(7 downto 0):="000000000";
   SIGNAL Sum : STD_LOGIC_VECTOR(7 downto 0);
13
  COMPONENT Adder_Cin_VHDL
    PORT (
15
      Cin : IN STD_LOGIC;
16
        : IN STD_LOGIC_VECTOR(7 downto 0);
17
      B : IN STD_LOGIC_VECTOR(7 downto 0);
18
      Sum : OUT STD_LOGIC_VECTOR(7 downto 0)
      );
  END COMPONENT;
22 BEGIN
  i1 : Adder_Cin_VHDL
   PORT MAP (
      Cin => Cin,
         => A,
      В
          => B,
27
      Sum => Sum
      );
29
   init : PROCESS
31
   BEGIN
32
    WAIT;
33
   END PROCESS init;
34
   always : PROCESS
36
   BEGIN
37
    WAIT FOR 50ns;
38
    Cin <= '0'; A <= "10000111"; B <= "11001110"; -- 135 + 206
39
    WAIT FOR 50ns;
40
    Cin <= '0'; A <= "011111010"; B <= "00110100"; -- 122 + 52
41
    WAIT FOR 50ns;
    Cin <= '0'; A <= "10110010"; B <= "00010010"; -- 178 + 18
```

```
WAIT FOR 50ns;
44
    Cin <= '0'; A <= "10101010"; B <= "01110011"; -- 170 + 115
    WAIT FOR 50ns;
46
    Cin <= '0'; A <= "00010111"; B <= "01001101"; -- 23 + 77
47
    WAIT FOR 50ns;
48
    Cin <= '1'; A <= "00000000"; B <= "00000000"; -- 0 + 0 + 1
49
    WAIT FOR 50ns;
50
    Cin <= '1'; A <= "10000111"; B <= "11001110"; -- 135 + 206 + 1
51
    WAIT FOR 50ns;
    Cin <= '1'; A <= "01111010"; B <= "00110100"; -- 122 + 52 + 1
    WAIT FOR 50ns;
54
    Cin <= '1'; A <= "10110010"; B <= "00010010"; -- 178 + 18 + 1
55
    WAIT FOR 50ns;
    Cin <= '1'; A <= "1010101010"; B <= "01110011"; -- 170 + 115 + 1
57
    WAIT FOR 50ns;
58
    Cin <= '1'; A <= "00010111"; B <= "01001101"; -- 23 + 77 + 1
    WAIT;
  END PROCESS always;
62 END Adder_Cin_VHDL_arch;
```

Programa 7: Banco de prueba para el Programa 1.

```
1 LIBRARY ieee;
USE ieee.std_logic_1164.all;
4 ENTITY Full_Adder_VHDL_vhd_tst IS
5 END Full_Adder_VHDL_vhd_tst;
7 ARCHITECTURE Full_Adder_VHDL_arch OF Full_Adder_VHDL_vhd_tst IS
  SIGNAL Cin : STD_LOGIC:='0';
  SIGNAL A : STD_LOGIC_VECTOR(7 downto 0):="000000000";
SIGNAL B : STD_LOGIC_VECTOR(7 downto 0):="000000000";
signal Temp : STD_LOGIC_VECTOR(8 downto 0);
SIGNAL Sum : STD_LOGIC_VECTOR(7 downto 0);
   SIGNAL Cout : STD_LOGIC;
13
   COMPONENT Full_Adder_VHDL
   PORT (
15
      Cin : IN STD_LOGIC;
16
        : IN STD_LOGIC_VECTOR(7 downto 0);
17
         : IN STD_LOGIC_VECTOR(7 downto 0);
18
      Sum : OUT STD_LOGIC_VECTOR(7 downto 0);
19
      Cout : OUT STD_LOGIC
20
      );
 END COMPONENT;
22
23 BEGIN
24 i1 : Full_Adder_VHDL
```

```
PORT MAP (
      Cin => Cin,
          => A,
27
          => B,
      В
28
      Sum => Sum,
29
      Cout => Cout
30
      );
31
32
   init : PROCESS
   BEGIN
34
    WAIT;
   END PROCESS init;
36
   always : PROCESS
38
   BEGIN
39
40
    WAIT FOR 50ns;
    Cin <= '0'; A <= "10000111"; B <= "11001110"; -- 135 + 206
41
    WAIT FOR 50ns;
    Cin <= '0'; A <= "01111010"; B <= "00110100"; -- 122 + 52
43
    WAIT FOR 50ns;
    Cin <= '0'; A <= "10110010"; B <= "00010010"; -- 178 + 18
45
    WAIT FOR 50ns;
46
    Cin <= '0'; A <= "10101010"; B <= "01110011"; -- 170 + 115
    WAIT FOR 50ns;
48
    Cin <= '0'; A <= "00010111"; B <= "01001101"; -- 23 + 77
49
    WAIT FOR 50ns;
50
    Cin <= '1'; A <= "00000000"; B <= "000000000"; -- 0 + 0 + 1
    WAIT FOR 50ns;
52
    Cin <= '1'; A <= "10000111"; B <= "11001110"; -- 135 + 206 + 1
53
    WAIT FOR 50ns;
    Cin <= '1'; A <= "01111010"; B <= "00110100"; -- 122 + 52 + 1
55
    WAIT FOR 50ns;
    Cin <= '1'; A <= "10110010"; B <= "00010010"; -- 178 + 18 + 1
57
    WAIT FOR 50ns;
58
    Cin <= '1'; A <= "1010101010"; B <= "01110011"; -- 170 + 115 + 1
59
    WAIT FOR 50ns;
60
    Cin <= '1'; A <= "00010111"; B <= "01001101"; -- 23 + 77 + 1
61
    WAIT;
   END PROCESS always;
64 END Full_Adder_VHDL_arch;
```

Programa 8: Banco de prueba para el Programa 2.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
```

```
4 ENTITY Adder_Subtractor_VHDL_vhd_tst IS
5 END Adder_Subtractor_VHDL_vhd_tst;
7 ARCHITECTURE Adder_Subtractor_VHDL_arch OF Adder_Subtractor_VHDL_vhd_tst
8 SIGNAL C : STD_LOGIC:='0';
  SIGNAL A : STD_LOGIC_VECTOR(7 DOWNTO 0):="000000000";
10 SIGNAL B : STD_LOGIC_VECTOR(7 DOWNTO 0):="000000000";
   SIGNAL Total : STD_LOGIC_VECTOR(8 DOWNTO 0);
  COMPONENT Adder_Subtractor_VHDL
12
   PORT (
13
      C : IN STD_LOGIC;
14
     A : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
      B : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
16
      Total : OUT STD_LOGIC_VECTOR(8 DOWNTO 0)
17
      );
18
  END COMPONENT;
19
20 BEGIN
  i1 : Adder_Subtractor_VHDL
   PORT MAP (
22
      C => C,
23
      A => A
24
      B \Rightarrow B,
      Total => Total
26
      );
27
28
  init : PROCESS
  BEGIN
30
    WAIT;
31
   END PROCESS init;
32
33
   always : PROCESS
34
   BEGIN
35
    WAIT FOR 50ns;
36
    C \le '0'; A \le "10000111"; B \le "11001110"; -- 135 + 206
37
    WAIT FOR 50ns;
38
    C \le '0'; A \le "01111010"; B \le "00110100"; -- 122 + 52
39
    WAIT FOR 50ns;
40
    C \le '0'; A \le "10110010"; B \le "00010010"; -- 178 + 18
41
    WAIT FOR 50ns;
42
    C \le '0'; A \le "1010101010"; B \le "01110011"; -- 170 + 115
    WAIT FOR 50ns;
44
    C \le '0'; A \le "00010111"; B \le "01001101"; -- 23 + 77
45
    WAIT FOR 50ns;
46
    C \le '1'; A \le "00000000"; B \le "00000000"; -- 0 - 0
47
```

```
WAIT FOR 50ns;
48
    C <= '1'; A <= "10000111"; B <= "11001110"; -- 135 - 206
    WAIT FOR 50ns;
50
    C \le '1'; A \le "01111010"; B \le "00110100"; --122 - 52
51
    WAIT FOR 50ns;
52
    C <= '1'; A <= "10110010"; B <= "00010010"; -- 178 - 18
53
    WAIT FOR 50ns;
    C \le '1'; A \le "1010101010"; B \le "01110011"; -- 170 - 115
55
    WAIT FOR 50ns;
    C \le '1'; A \le "00010111"; B \le "01001101"; -- 23 - 77
57
    WAIT;
   END PROCESS always;
60 END Adder_Subtractor_VHDL_arch;
```

Programa 9: Banco de prueba para el Programa 3.

```
'timescale 1 ns/ 1 ps
2 module Adder_Cin_Verilog_vlg_tst();
          Cin;
3 reg
4 reg [7:0] A;
5 reg [7:0] B;
   wire [7:0] Sum;
   Adder_Cin_Verilog i1 (
    .Cin(Cin),
    .A(A),
10
    .B(B),
11
    .Sum(Sum)
12
   );
14
   initial
15
   begin
16
    Cin = 0; A = 0; B = 0;
17
    $display("Running testbench at CIC");
18
   end
19
   always
21
   begin
    \#50; Cin = 0; A = 135; B = 206; // 135 + 206
23
    #50; Cin = 0; A = 122; B = 52; // 122 + 52
24
    \#50; Cin = 0; A = 178; B = 18; // 178 + 18
25
    \#50; Cin = 0; A = 170; B = 115; // 170 + 115
26
    \#50; Cin = 0; A = 23; B = 77; // 23 + 77
    #50; Cin = 1; A = 0;
                            B = 0;
                                     // 0 + 0 + 1
28
    \#50; Cin = 1; A = 135; B = 206; // 135 + 206
    #50; Cin = 1; A = 122; B = 52; // 122 + 52
```

```
#50; Cin = 1; A = 178; B = 18; // 178 + 18

#50; Cin = 1; A = 170; B = 115; // 170 + 115

#50; Cin = 1; A = 23; B = 77; // 23 + 77

end

sendmodule
```

Programa 10: Banco de prueba para el Programa 4.

```
1 'timescale 1 ns/ 1 ps
2 module Full_Adder_Verilog_vlg_tst();
          Cin;
   reg
   reg [7:0]
   reg [7:0]
               В;
   wire [7:0] Sum;
   wire
             Cout;
   Full_Adder_Verilog i1 (
9
    .Cin(Cin),
10
    .A(A),
11
    .B(B),
12
    .Sum(Sum),
    .Cout(Cout)
14
   );
15
16
   initial
17
   begin
18
    Cin = 0; A = 0; B = 0;
19
    $display("Running testbench at CIC");
   end
21
22
   always
23
   begin
24
    \#50; Cin = 0; A = 135; B = 206; // 135 + 206
25
    #50; Cin = 0; A = 122; B = 52;
                                     // 122 + 52
26
    #50; Cin = 0; A = 178; B = 18;
                                     // 178 + 18
27
    \#50; Cin = 0; A = 170; B = 115; // 170 + 115
28
    #50; Cin = 0; A = 23;
                             B = 77;
                                     // 23 + 77
29
    #50; Cin = 1; A = 0;
                             B = 0;
                                      // 0 + 0 + 1
30
    \#50; Cin = 1; A = 135; B = 206; // 135 + 206
31
    #50; Cin = 1; A = 122; B = 52;
                                      // 122 + 52
32
    #50; Cin = 1; A = 178; B = 18;
                                      // 178 + 18
33
    #50; Cin = 1; A = 170; B = 115; // 170 + 115
    #50; Cin = 1; A = 23;
                            B = 77; // 23 + 77
35
   end
37
```

38 endmodule

Programa 11: Banco de prueba para el Programa 5.

```
'timescale 1 ns/ 1 ps
2 module Adder_Subtractor_Verilog_vlg_tst();
   reg
        [7:0]
               Α;
        [7:0]
               B;
   reg
   wire [8:0] Total;
   Adder_Subtractor_Verilog i1 (
    .C(C),
9
    .A(A),
    .B(B),
11
    .Total(Total)
12
   );
13
14
15
   initial
   begin
16
    C = 0; A = 0; B = 0;
17
    $display("Running testbench at CIC");
18
   end
20
   always
21
   begin
22
    #50; C = 0; A = 135; B = 206; // 135 + 206
23
    #50; C = 0; A = 122; B = 52; // 122 + 52
    #50; C = 0; A = 178; B = 18;
                                   // 178 + 18
25
    #50; C = 0; A = 170; B = 115; // 170 + 115
    #50; C = 0; A = 23; B = 77; // 23 + 77
27
    #50; C = 1; A = 0;
                                    // 0 - 0
                          B = 0;
28
    #50; C = 1; A = 135; B = 206; // 135 - 206
29
    #50; C = 1; A = 122; B = 52; // 122 - 52
30
    #50; C = 1; A = 178; B = 18; // 178 - 18
31
    #50; C = 1; A = 170; B = 115; // 170 - 115
32
    #50; C = 1; A = 23; B = 77; // 23 - 77
   end
34
  endmodule
```

Programa 12: Banco de prueba para el Programa 6.