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

Ciudad de México, 14 de marzo de 2024

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. Bancos de pruebas (*Test Benches*) 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].

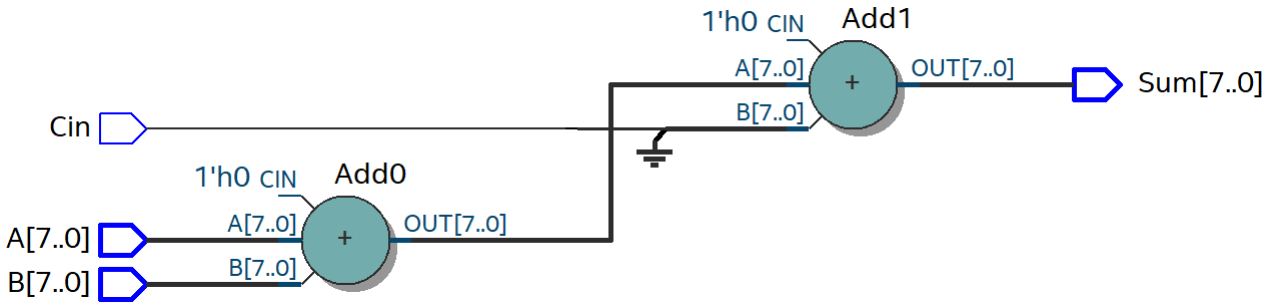

Figura 1: Diagrama RTL del sumador con acarreo de entrada en VHDL.

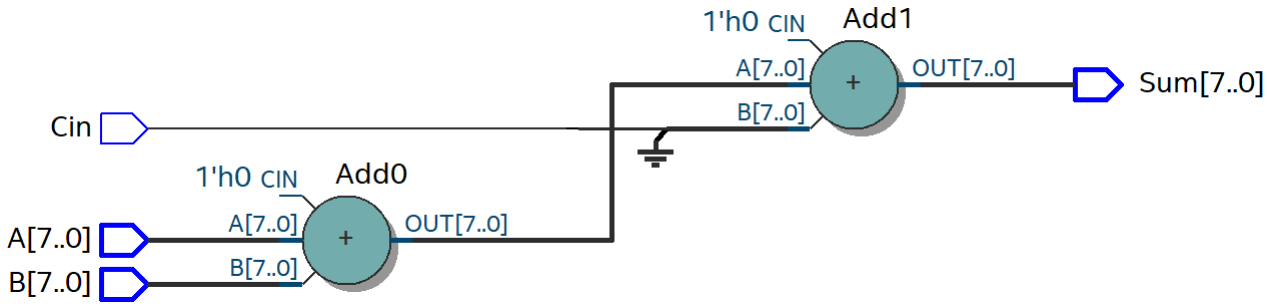

Figura 2: Diagrama RTL del sumador con acarreo de entrada en Verilog.

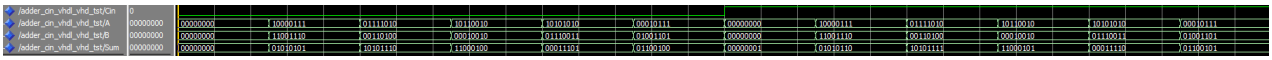

Figura 3: Simulación del sumador con acarreo de entrada en VHDL con el visor de formas de onda de ModelSim (Base binaria).

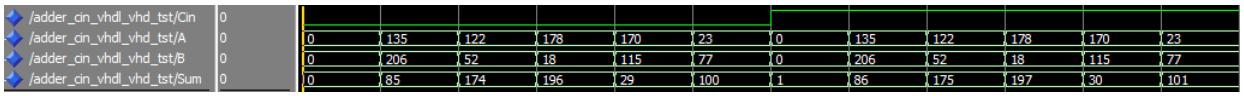

Figura 4: Simulación del sumador con acarreo de entrada en VHDL con el visor de formas de onda de ModelSim (Base decimal).

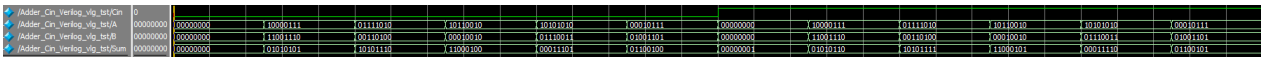

Figura 5: Simulación del sumador con acarreo de entrada en Verilog con el visor de formas de onda de ModelSim (Base binaria).

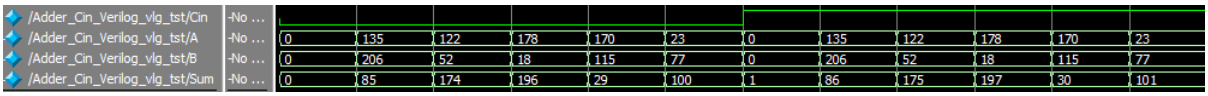

Figura 6: Simulación del sumador con acarreo de entrada en Verilog con el visor de formas de onda de ModelSim (Base decimal).

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.

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

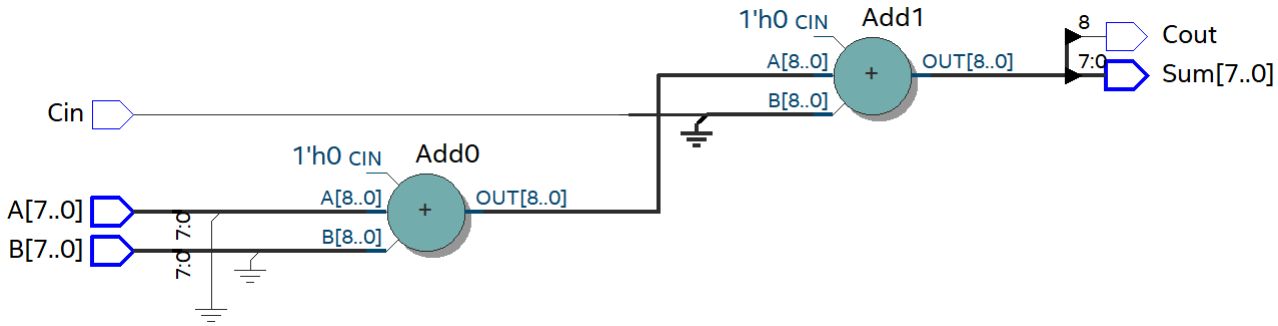

Figura 8: Diagrama RTL del sumador completo en VHDL.

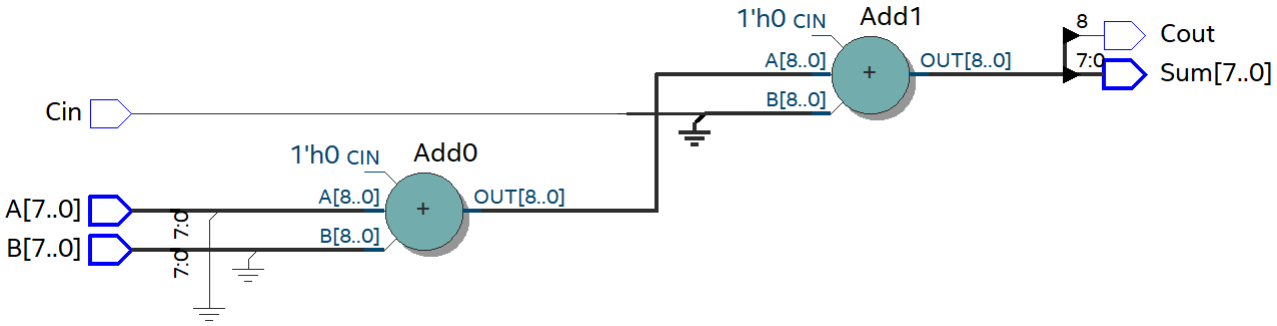

Figura 9: Diagrama RTL del sumador completo en Verilog.

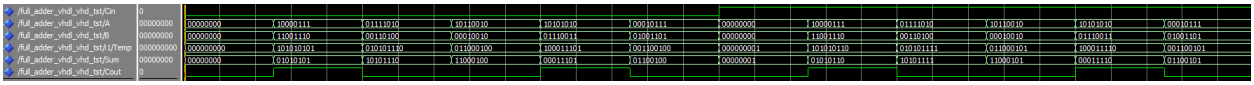

Figura 10: Simulación del sumador completo en VHDL con el visor de formas de onda de ModelSim (Base binaria).

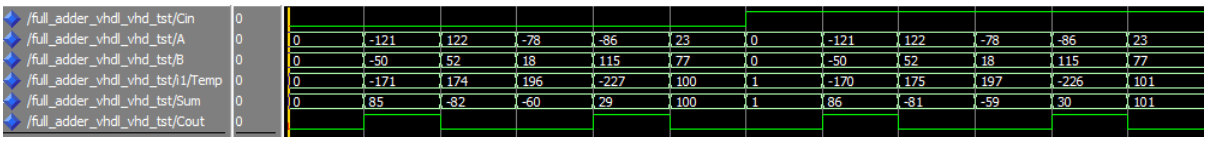

Figura 11: Simulación del sumador completo en VHDL con el visor de formas de onda de ModelSim (Base decimal).

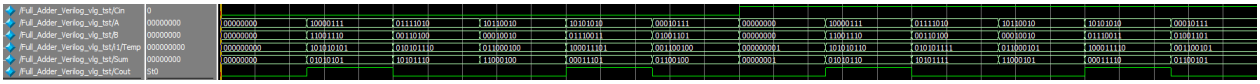

Figura 12: Simulación del sumador completo en Verilog con el visor de formas de onda de ModelSim (Base binaria).

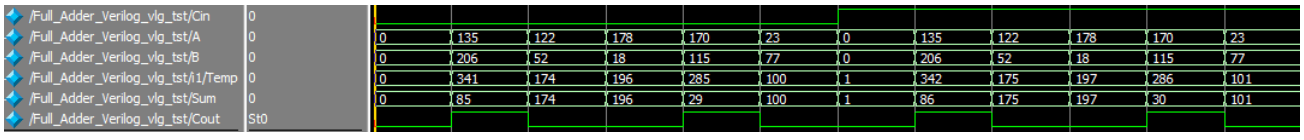

Figura 13: Simulación del sumador completo en Verilog con el visor de formas de onda de ModelSim (Base decimal).

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.

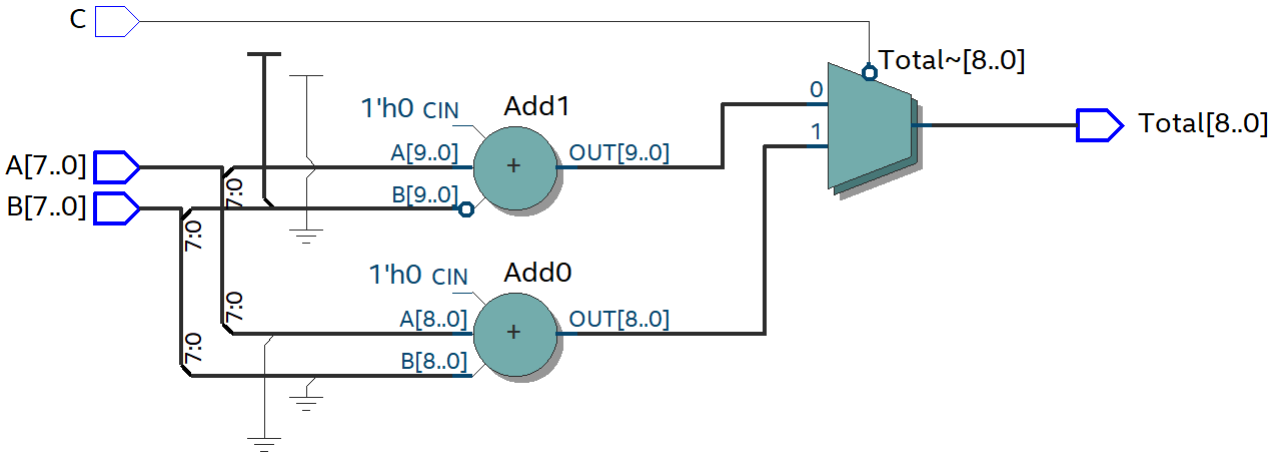

Figura 14: Diagrama RTL del sumador/restador en VHDL.

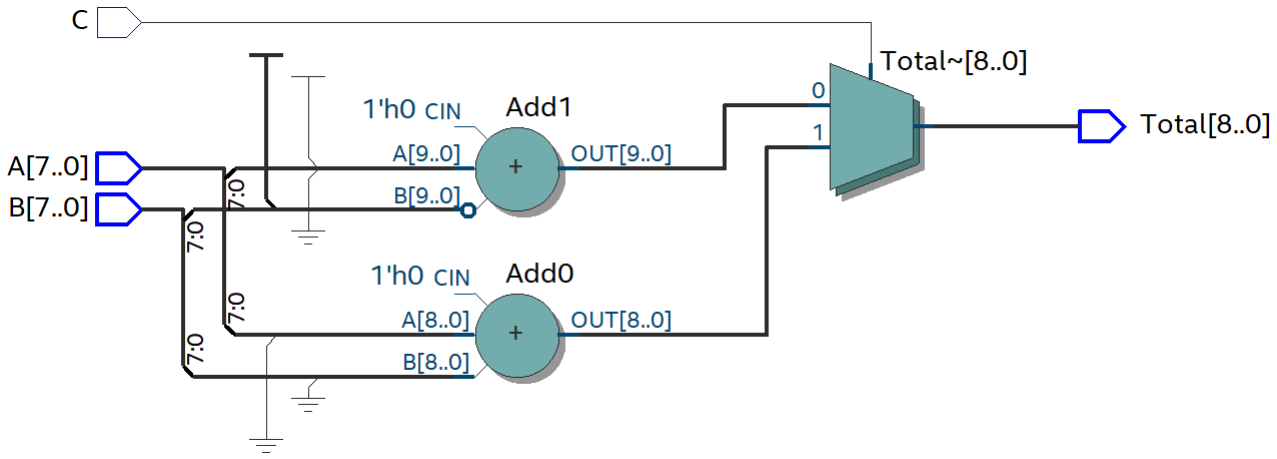

Figura 15: Diagrama RTL del sumador/restador en Verilog.

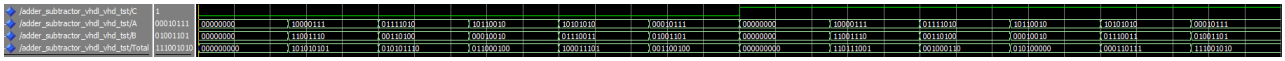

Figura 16: Simulación del sumador/restador en VHDL con el visor de formas de onda de ModelSim (Base binaria).

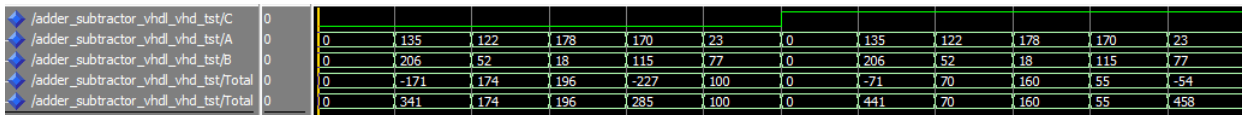

Figura 17: Simulación del sumador/restador en VHDL con el visor de formas de onda de ModelSim (Base decimal).

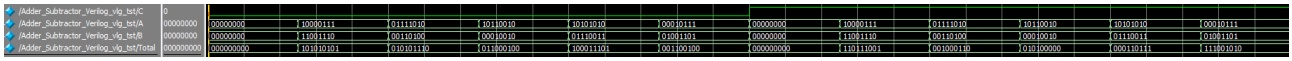

Figura 18: Simulación del sumador/restador en Verilog con el visor de formas de onda de ModelSim (Base binaria).

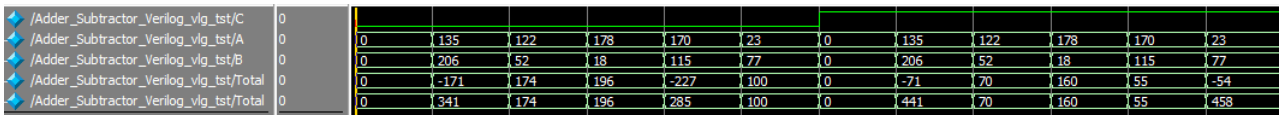

Figura 19: Simulación del sumador/restador en Verilog con el visor de formas de onda de ModelSim (Base decimal).

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

```vhdl
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;
```

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

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

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);
end Full_Adder_VHDL;

architecture behavior of Full_Adder_VHDL is
signal  Temp  :  std_logic_vector(8 downto 0);
begin
 process(A, B, Cin)
 begin
  Temp <= ('0' & A) + ('0' & B) + Cin;
 end process;
 Sum  <= Temp(7 downto 0);
```

```
 Cout <= Temp(8);
end behavior;
```

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

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

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)
    );
end Adder_Subtractor_VHDL;

architecture Behavioral of Adder_Subtractor_VHDL is
begin
 process(A, B, C)
 begin
  if (C = '0') then
   Total <= ('0' & A) + ('0' & B);
  else
   Total <= ('0' & A) - ('0' & B);
  end if;
 end process;
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.

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

 reg [8:0] Temp;
 always @(Cin, A, B)
 begin
  Temp <= A + B + Cin;
 end
 assign Sum  = Temp[7:0];
 assign Cout = Temp[8];

endmodule
```

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

```
module Adder_Subtractor_Verilog(
 input  C,
 input    [7:0]  A,
 input    [7:0]  B,
 output reg [8:0]  Total
);

 always @(A, B, C)
 begin
  if(!C)
   Total <= {1'b0, A} + {1'b0, B};
  else
   Total <= {1'b0, A} - {1'b0, B};
 end

endmodule
```

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

6.2. Bancos de pruebas (*Test Benches*)

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

ENTITY Adder_Cin_VHDL_vhd_tst IS
END Adder_Cin_VHDL_vhd_tst;

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):="00000000";
 SIGNAL  B   : STD_LOGIC_VECTOR(7 downto 0):="00000000";
 SIGNAL  Sum : STD_LOGIC_VECTOR(7 downto 0);
 COMPONENT Adder_Cin_VHDL
  PORT  (
    Cin  : IN STD_LOGIC;
    A    : IN STD_LOGIC_VECTOR(7 downto 0);
    B    : IN STD_LOGIC_VECTOR(7 downto 0);
    Sum  : OUT STD_LOGIC_VECTOR(7 downto 0)
    );
 END COMPONENT;
BEGIN
 i1 : Adder_Cin_VHDL
 PORT MAP (
    Cin  => Cin,
    A    => A,
    B    => B,
    Sum  => Sum
    );

 init : PROCESS
 BEGIN
  WAIT;
 END PROCESS init;

 always : PROCESS
 BEGIN
  WAIT FOR 50ns;
  Cin <= '0'; A <= "10000111";  B <= "11001110"; -- 135 + 206
  WAIT FOR 50ns;
  Cin <= '0'; A <= "01111010";  B <= "00110100"; -- 122 + 52
  WAIT FOR 50ns;
  Cin <= '0'; A <= "10110010";  B <= "00010010"; -- 178 + 18
```

```
  WAIT FOR 50ns;
  Cin <= '0'; A <= "10101010";  B <= "01110011"; -- 170 + 115
  WAIT FOR 50ns;
  Cin <= '0'; A <= "00010111";  B <= "01001101"; -- 23 + 77
  WAIT FOR 50ns;
  Cin <= '1'; A <= "00000000";  B <= "00000000"; -- 0 + 0 + 1
  WAIT FOR 50ns;
  Cin <= '1'; A <= "10000111";  B <= "11001110"; -- 135 + 206 + 1
  WAIT FOR 50ns;
  Cin <= '1'; A <= "01111010";  B <= "00110100"; -- 122 + 52 + 1
  WAIT FOR 50ns;
  Cin <= '1'; A <= "10110010";  B <= "00010010"; -- 178 + 18 + 1
  WAIT FOR 50ns;
  Cin <= '1'; A <= "10101010";  B <= "01110011"; -- 170 + 115 + 1
  WAIT FOR 50ns;
  Cin <= '1'; A <= "00010111";  B <= "01001101"; -- 23 + 77 + 1
  WAIT;
 END PROCESS always;
END Adder_Cin_VHDL_arch;
```

Programa 7: Banco de prueba para el Programa 1.

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

ENTITY Full_Adder_VHDL_vhd_tst IS
END Full_Adder_VHDL_vhd_tst;

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):="00000000";
 SIGNAL  B   : STD_LOGIC_VECTOR(7 downto 0):="00000000";
 SIGNAL  Temp : STD_LOGIC_VECTOR(8 downto 0);
 SIGNAL  Sum : STD_LOGIC_VECTOR(7 downto 0);
 SIGNAL  Cout : STD_LOGIC;
 COMPONENT Full_Adder_VHDL
  PORT  (
    Cin   : IN  STD_LOGIC;
    A     : IN  STD_LOGIC_VECTOR(7 downto 0);
    B     : IN  STD_LOGIC_VECTOR(7 downto 0);
    Sum   : OUT  STD_LOGIC_VECTOR(7 downto 0);
    Cout   : OUT  STD_LOGIC
    );
 END COMPONENT;
BEGIN
 i1 : Full_Adder_VHDL
```

```
 PORT MAP (
    Cin  => Cin,
    A    => A,
    B    => B,
    Sum  => Sum,
    Cout => Cout
    );

 init : PROCESS
 BEGIN
  WAIT;
 END PROCESS init;

 always : PROCESS
 BEGIN
  WAIT FOR 50ns;
  Cin <= '0'; A <= "10000111";  B <= "11001110"; -- 135 + 206
  WAIT FOR 50ns;
  Cin <= '0'; A <= "01111010";  B <= "00110100"; -- 122 + 52
  WAIT FOR 50ns;
  Cin <= '0'; A <= "10110010";  B <= "00010010"; -- 178 + 18
  WAIT FOR 50ns;
  Cin <= '0'; A <= "10101010";  B <= "01110011"; -- 170 + 115
  WAIT FOR 50ns;
  Cin <= '0'; A <= "00010111";  B <= "01001101"; -- 23 + 77
  WAIT FOR 50ns;
  Cin <= '1'; A <= "00000000";  B <= "00000000"; -- 0 + 0 + 1
  WAIT FOR 50ns;
  Cin <= '1'; A <= "10000111";  B <= "11001110"; -- 135 + 206 + 1
  WAIT FOR 50ns;
  Cin <= '1'; A <= "01111010";  B <= "00110100"; -- 122 + 52 + 1
  WAIT FOR 50ns;
  Cin <= '1'; A <= "10110010";  B <= "00010010"; -- 178 + 18 + 1
  WAIT FOR 50ns;
  Cin <= '1'; A <= "10101010";  B <= "01110011"; -- 170 + 115 + 1
  WAIT FOR 50ns;
  Cin <= '1'; A <= "00010111";  B <= "01001101"; -- 23 + 77 + 1
  WAIT;
 END PROCESS always;
END Full_Adder_VHDL_arch;
```

Programa 8: Banco de prueba para el Programa 2.

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

```

```
ENTITY Adder_Subtractor_VHDL_vhd_tst IS
END Adder_Subtractor_VHDL_vhd_tst;

ARCHITECTURE Adder_Subtractor_VHDL_arch OF Adder_Subtractor_VHDL_vhd_tst
    IS
 SIGNAL C   : STD_LOGIC:='0';
 SIGNAL  A  : STD_LOGIC_VECTOR(7 DOWNTO 0):="00000000";
 SIGNAL  B  : STD_LOGIC_VECTOR(7 DOWNTO 0):="00000000";
 SIGNAL  Total : STD_LOGIC_VECTOR(8 DOWNTO 0);
 COMPONENT Adder_Subtractor_VHDL
  PORT  (
    C   : IN  STD_LOGIC;
    A   : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
    B   : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
    Total : OUT STD_LOGIC_VECTOR(8 DOWNTO 0)
    );
 END COMPONENT;
BEGIN
 i1 : Adder_Subtractor_VHDL
 PORT MAP (
    C  => C,
    A   => A,
    B   => B,
    Total => Total
    );

 init : PROCESS
 BEGIN
  WAIT;
 END PROCESS init;

 always : PROCESS
 BEGIN
  WAIT FOR 50ns;
  C <= '0'; A <= "10000111";  B <= "11001110"; -- 135 + 206
  WAIT FOR 50ns;
  C <= '0'; A <= "01111010";  B <= "00110100"; -- 122 + 52
  WAIT FOR 50ns;
  C <= '0'; A <= "10110010";  B <= "00010010"; -- 178 + 18
  WAIT FOR 50ns;
  C <= '0'; A <= "10101010";  B <= "01110011"; -- 170 + 115
  WAIT FOR 50ns;
  C <= '0'; A <= "00010111";  B <= "01001101"; -- 23 + 77
  WAIT FOR 50ns;
  C <= '1'; A <= "00000000";  B <= "00000000"; -- 0 - 0
```

```
  WAIT FOR 50ns;
  C <= '1'; A <= "10000111";  B <= "11001110"; -- 135 - 206
  WAIT FOR 50ns;
  C <= '1'; A <= "01111010";  B <= "00110100"; -- 122 - 52
  WAIT FOR 50ns;
  C <= '1'; A <= "10110010";  B <= "00010010"; -- 178 - 18
  WAIT FOR 50ns;
  C <= '1'; A <= "10101010";  B <= "01110011"; -- 170 - 115
  WAIT FOR 50ns;
  C <= '1'; A <= "00010111";  B <= "01001101"; -- 23 - 77
  WAIT;
 END PROCESS always;
END Adder_Subtractor_VHDL_arch;
```

Programa 9: Banco de prueba para el Programa 3.

```
`timescale 1 ns/ 1 ps
module Adder_Cin_Verilog_vlg_tst();
 reg    Cin;
 reg [7:0] A;
 reg [7:0] B;
 wire  [7:0] Sum;

 Adder_Cin_Verilog i1 (
  .Cin(Cin),
  .A(A),
  .B(B),
  .Sum(Sum)
 );

 initial
 begin
  Cin = 0; A = 0; B = 0;
  $display("Running testbench at CIC");
 end

 always
 begin
  #50; Cin = 0; A = 135; B = 206; // 135 + 206
  #50; Cin = 0; A = 122; B = 52;  // 122 + 52
  #50; Cin = 0; A = 178; B = 18;  // 178 + 18
  #50; Cin = 0; A = 170; B = 115; // 170 + 115
  #50; Cin = 0; A = 23;  B = 77;  // 23 + 77
  #50; Cin = 1; A = 0;   B = 0;   // 0 + 0 + 1
  #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

endmodule
```

Programa 10: Banco de prueba para el Programa 4.

```
`timescale 1 ns/ 1 ps
module Full_Adder_Verilog_vlg_tst();
 reg    Cin;
 reg [7:0]  A;
 reg [7:0]  B;
 wire [7:0]  Sum;
 wire     Cout;

 Full_Adder_Verilog i1 (
  .Cin(Cin),
  .A(A),
  .B(B),
  .Sum(Sum),
  .Cout(Cout)
 );

 initial
 begin
  Cin = 0; A = 0; B = 0;
  $display("Running testbench at CIC");
 end

 always
 begin
  #50; Cin = 0; A = 135; B = 206; // 135 + 206
  #50; Cin = 0; A = 122; B = 52;  // 122 + 52
  #50; Cin = 0; A = 178; B = 18;  // 178 + 18
  #50; Cin = 0; A = 170; B = 115; // 170 + 115
  #50; Cin = 0; A = 23;  B = 77;  // 23 + 77
  #50; Cin = 1; A = 0;   B = 0;   // 0 + 0 + 1
  #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

```

```
endmodule
```

Programa 11: Banco de prueba para el Programa 5.

```
`timescale 1 ns/ 1 ps
module Adder_Subtractor_Verilog_vlg_tst();
 reg  C;
 reg  [7:0]  A;
 reg  [7:0]  B;
 wire  [8:0]  Total;

 Adder_Subtractor_Verilog i1 (
  .C(C),
  .A(A),
  .B(B),
  .Total(Total)
 );

 initial
 begin
  C = 0; A = 0; B = 0;
  $display("Running testbench at CIC");
 end

 always
 begin
  #50; C = 0; A = 135; B = 206; // 135 + 206
  #50; C = 0; A = 122; B = 52;  // 122 + 52
  #50; C = 0; A = 178; B = 18;  // 178 + 18
  #50; C = 0; A = 170; B = 115; // 170 + 115
  #50; C = 0; A = 23;  B = 77;  // 23 + 77
  #50; C = 1; A = 0;   B = 0;   // 0 - 0
  #50; C = 1; A = 135; B = 206; // 135 - 206
  #50; C = 1; A = 122; B = 52;  // 122 - 52
  #50; C = 1; A = 178; B = 18;  // 178 - 18
  #50; C = 1; A = 170; B = 115; // 170 - 115
  #50; C = 1; A = 23;  B = 77;  // 23 - 77
 end

endmodule
```

Programa 12: Banco de prueba para el Programa 6.