Instituto Politécnico Nacional

Centro de Investigación en Computación

Lenguajes de descripción de hardware

Tarea 4 - Ciclos FOR en descripción por comportamiento

Profesor:

M. en C. Osvaldo Espinosa Sosa

Por:

Ing. Ricardo Aldair Tirado Torres

Ciudad de México, 3 de mayo de 2024

Tabla de contenido

1. Objetivos

- Comprender el funcionamiento de las iteraciones con For-Loop, así como su sintaxis, realizando la operación producto punto de dos vectores.
- Indagar sobre la manera en que se describen las iteraciones For-Loop en Verilog.
- Aprender acerca de la importancia de inicializar variables que se usarán en operaciones aritméticas.

2. For-Loop en VHDL

Actividad 1

Compilar el código VHDL donde se utiliza la construcción for-loop. Observar el resultado en el visor RTL.

La visualización RTL del producto punto de dos vectores, empleando For-Loop en VHDL, se muestra en la Figura 1. Como se observa, la implementación se realiza utilizando 4 instancias de sumadores de 16 bits, 4 instancias de multiplicadores de 8 bits, un decodificador, algunas compuertas lógicas AND para apoyar al decodificador y 64 latches que se utilizan para almacenar cada bit de cada dato ubicado en las localidades de Vector1 y Vector2. Las simulaciones se visualizan en la Figura 2 en base binaria y en la Figura 3 en base decimal, en donde se muestra que la operación producto punto de ambos vectores es correcta.

En los Anexos se localiza la descripción en VHDL de este modulo de producto punto. La implementación se hace declarando las librerías y la entidad, junto con las señales de entradas y salida correspondientes. En la zona declarativa se declara un arreglo de 4 localidades de 8 bits de tamaño y después se crean 2 arreglos de este tipo, denominados Vector1 y Vector2 y se inicializan los valores de cada localidad. Al entrar a la declaración de arquitectura se generan dos lista sensibles:

- La primera, evalúa si esta habilitada la señal WR, en caso de ser así, se escriben los datos de entrada en la dirección indicada por la señal DIR.
- La segunda, genera el cálculo del producto punto, usando una variable de apoyo (Var) para hacer la suma de los productos y otra variable (i) para iterar las operaciones. Cabe señalar que la primer variable mencionada se inicializa en 0, para realizar la suma aritmética de manera correcta. Al final de las iteraciones, se asigna el valor de la variable a la señal de salida.

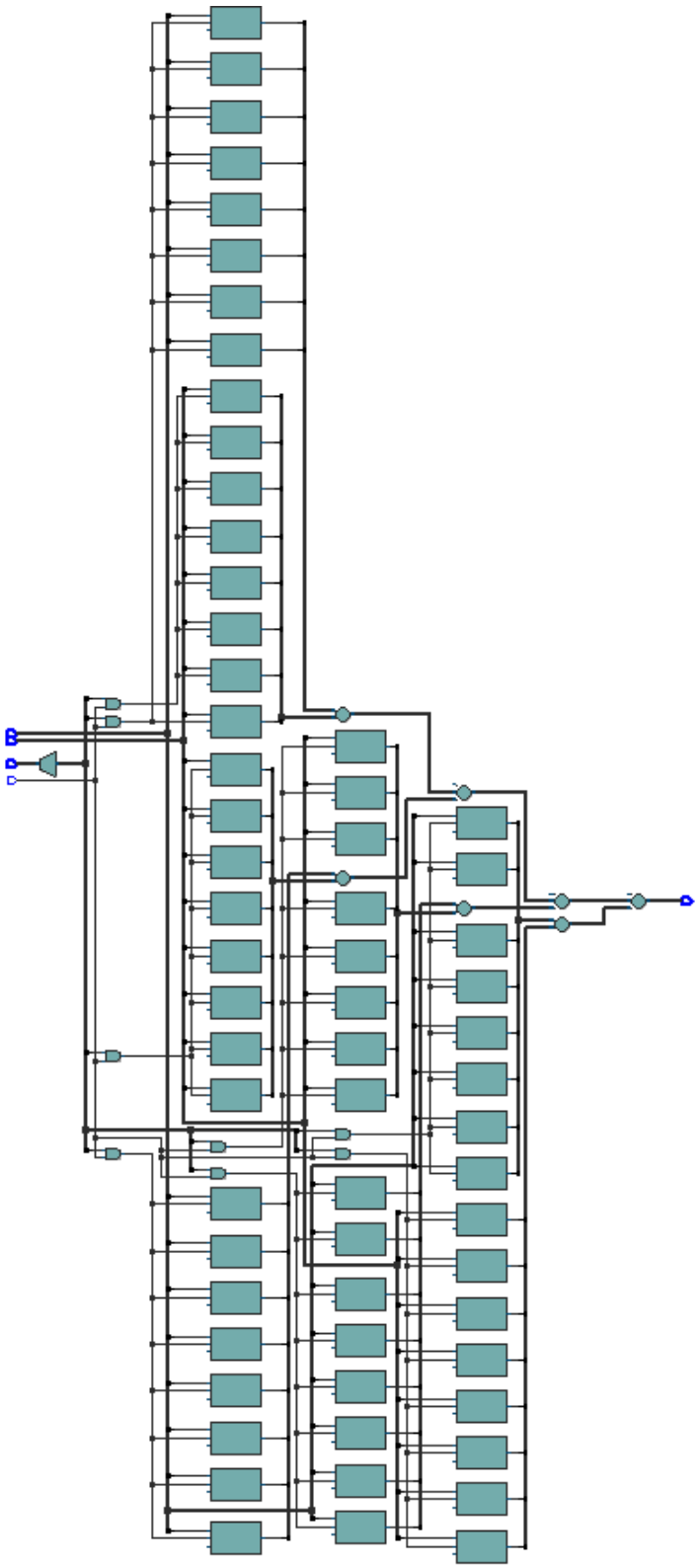

Figura 1: Diagrama RTL del producto punto de dos vectores implementado en VHDL.

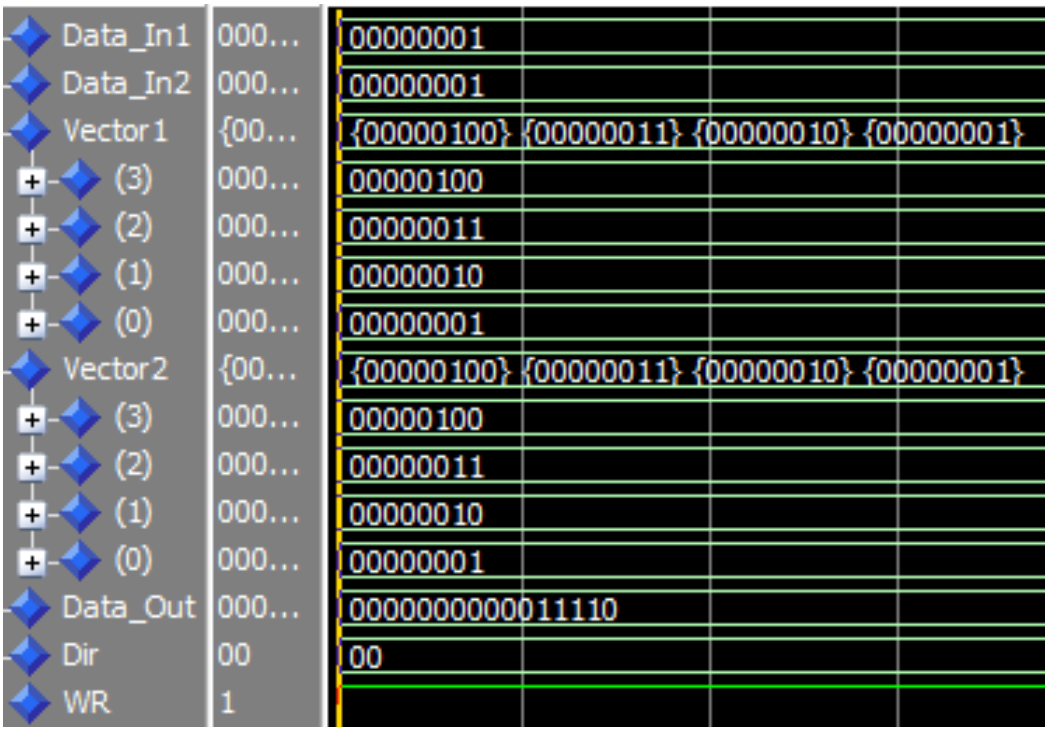

Figura 2: Simulación del producto punto de dos vectores en VHDL con el visor de formas de onda de ModelSim (base binaria).

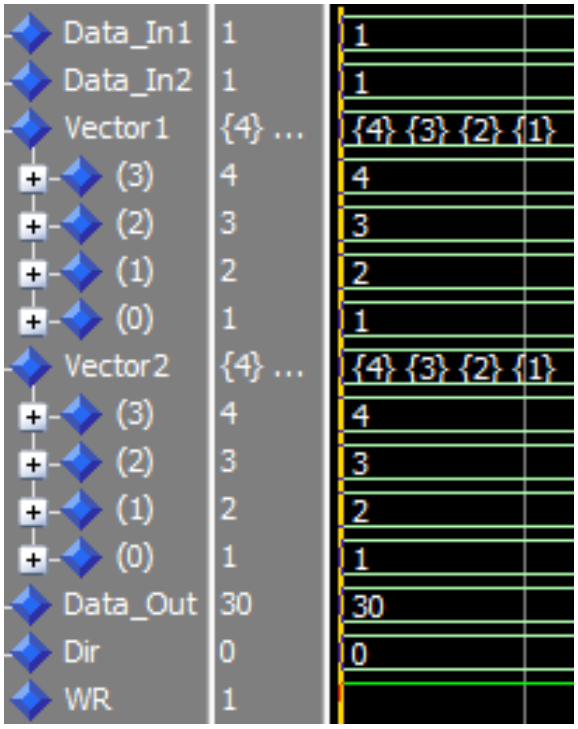

Figura 3: Simulación del producto punto de dos vectores en VHDL con el visor de formas de onda de ModelSim (base decimal).

3. For-Loop sin inicialización de variable

Actividad 2

Compilar de nuevo el código VHDL pero comentando o eliminando la inicialización de la variable *"var"* a cero. Observar el resultado en el visor RTL. ¿Qué diferencia existe respecto al inciso 1?

Lo que se observa en la Figura 4 es que se conectan de manera distinta los componentes del modulo. Al momento de compilar el proyecto se muestra un incremento significativo de las advertencias (Ver Figura 5). Las simulaciones se visualizan en la Figura 6 en base binaria y en la Figura 7 en base decimal, en donde se tiene que el resultado del producto punto es el valor "X" (conocido como "no importa"), ya que no se inicializó la variable y cuando se realiza una operación aritmética con "X", el resultado siempre sera "X". Esto se comprueba con el mensaje que se observa en la Figura 8.

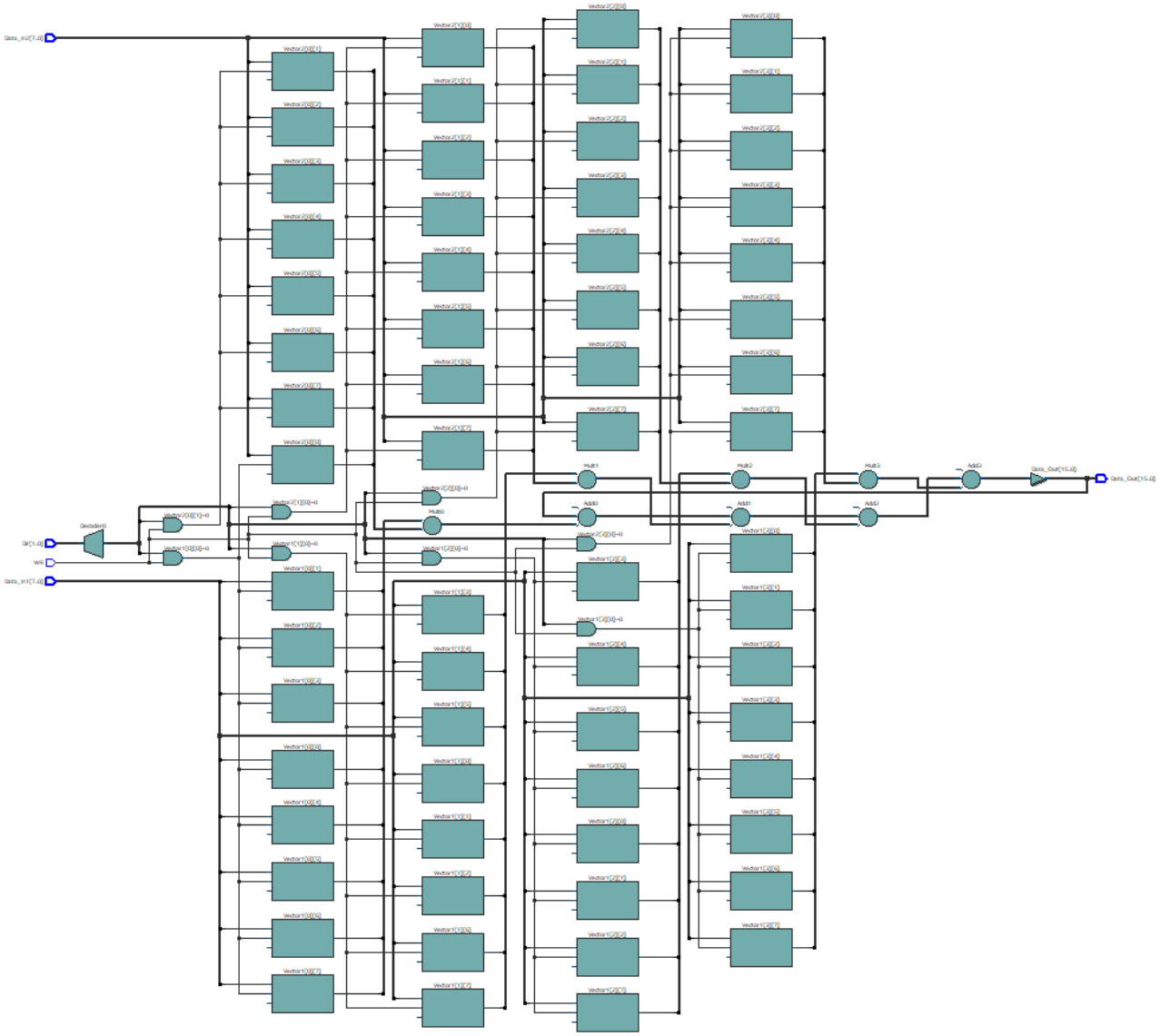

Figura 4: Diagrama RTL del producto punto de dos vectores implementado en VHDL, sin inicializar la variable de suma de productos.

```
Quartus Prime EDA Netlist Writer was successful. 0 errors, 1 warning
293000 Quartus Prime Full Compilation was successful. 0 errors, 243 warnings
```

Figura 5: Notorio incremento de los mensajes de advertencia al momento de compilar el proyecto debido a que la variable de suma de productos no se inicializó.

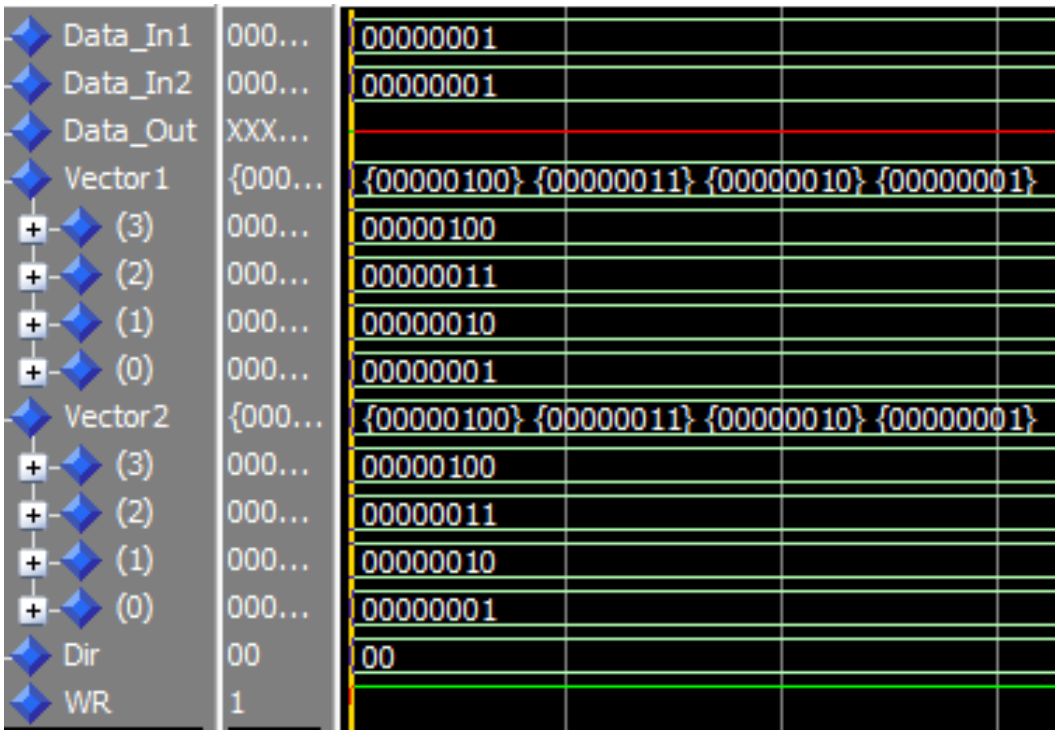

Figura 6: Simulación del producto punto de dos vectores en VHDL, sin inicializar la variable de suma de productos, con el visor de formas de onda de ModelSim (base binaria).

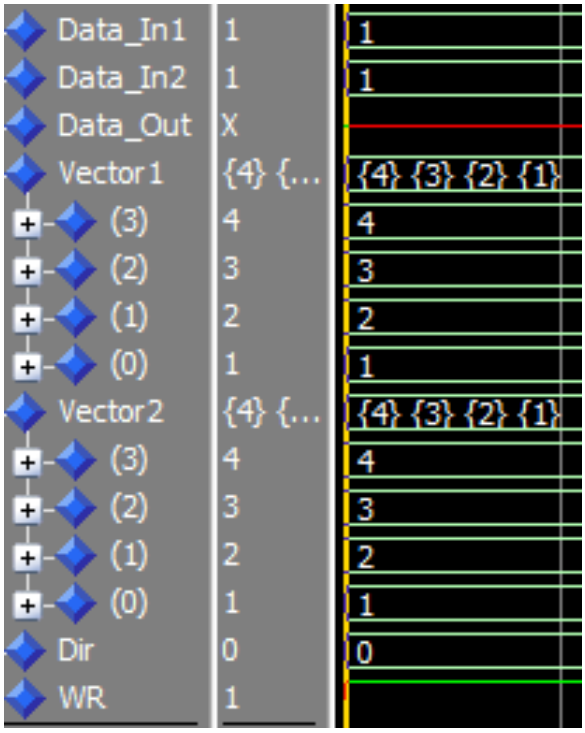

Figura 7: Simulación del producto punto de dos vectores en VHDL, sin inicializar la variable de suma de productos, con el visor de formas de onda de ModelSim (base decimal).

```
# ** Warning: There is an 'U'|'X'|'W'|'Z'|'-' in an arithmetic operand, the result will be 'X'(es).
#    Time: 0 ps  Iteration: 0  Instance: /for_loop1_vhdl_vhd_tst/i1
# ** Warning: There is an 'U'|'X'|'W'|'Z'|'-' in an arithmetic operand, the result will be 'X'(es).
#    Time: 0 ps  Iteration: 0  Instance: /for_loop1_vhdl_vhd_tst/i1
# ** Warning: There is an 'U'|'X'|'W'|'Z'|'-' in an arithmetic operand, the result will be 'X'(es).
#    Time: 0 ps  Iteration: 0  Instance: /for_loop1_vhdl_vhd_tst/i1
# ** Warning: There is an 'U'|'X'|'W'|'Z'|'-' in an arithmetic operand, the result will be 'X'(es).
#    Time: 0 ps  Iteration: 0  Instance: /for_loop1_vhdl_vhd_tst/i1
# ** Warning: There is an 'U'|'X'|'W'|'Z'|'-' in an arithmetic operand, the result will be 'X'(es).
#    Time: 0 ps  Iteration: 2  Instance: /for_loop1_vhdl_vhd_tst/i1
# ** Warning: There is an 'U'|'X'|'W'|'Z'|'-' in an arithmetic operand, the result will be 'X'(es).
#    Time: 0 ps  Iteration: 2  Instance: /for_loop1_vhdl_vhd_tst/i1
# ** Warning: There is an 'U'|'X'|'W'|'Z'|'-' in an arithmetic operand, the result will be 'X'(es).
#    Time: 0 ps  Iteration: 2  Instance: /for_loop1_vhdl_vhd_tst/i1
# ** Warning: There is an 'U'|'X'|'W'|'Z'|'-' in an arithmetic operand, the result will be 'X'(es).
#    Time: 0 ps  Iteration: 2  Instance: /for_loop1_vhdl_vhd_tst/i1
```

Figura 8: Mensaje del simulador ModelSim debido a que la variable de suma de productos no se inicializó.

4. For-Loop en Verilog

Actividad 1

Codificar el código del inciso 1, pero escribiéndolo en verilog. Observar el resultado con el visor RTL.

La visualización RTL del producto punto de dos vectores, empleando For-Loop en Verilog, se muestra en la Figura 9. Como se observa, la implementación se realiza utilizando 4 instancias de sumadores de 16 bits, 4 instancias de multiplicadores de 8 bits, un decodificador y 64 latches que se utilizan para almacenar cada bit de cada dato ubicado en las localidades de Vector1 y Vector2. Las simulaciones se visualizan en la Figura 10 en base binaria y en la Figura 11 en base decimal, en donde se muestra que la operación producto punto de ambos vectores es correcta.

En los Anexos se localiza la descripción en Verilog de este modulo de producto punto. La implementación se hace declarando el módulo, junto con las señales de entradas y salida correspondientes. Se declaran 2 arreglos de 4 localidades de 8 bits de tamaño, denominados Vector1 y Vector2 y se inicializan los valores de cada localidad. Después se generan dos lista sensibles:

- La primera, evalúa si esta habilitada la señal WR, en caso de ser así, se escriben los datos de entrada en la dirección indicada por la señal DIR.
- La segunda, genera el cálculo del producto punto, usando una variable de apoyo (Var) para hacer la suma de los productos y otra variable (i) para iterar las operaciones. Cabe señalar que antes del *always*, se deben declarar a las variables de apoyo mencionadas y dentro del bloque se debe inicializar a Var en 0, para realizar la suma aritmética de manera correcta. Al final de las iteraciones, se asigna el valor de la variable a la señal de salida.

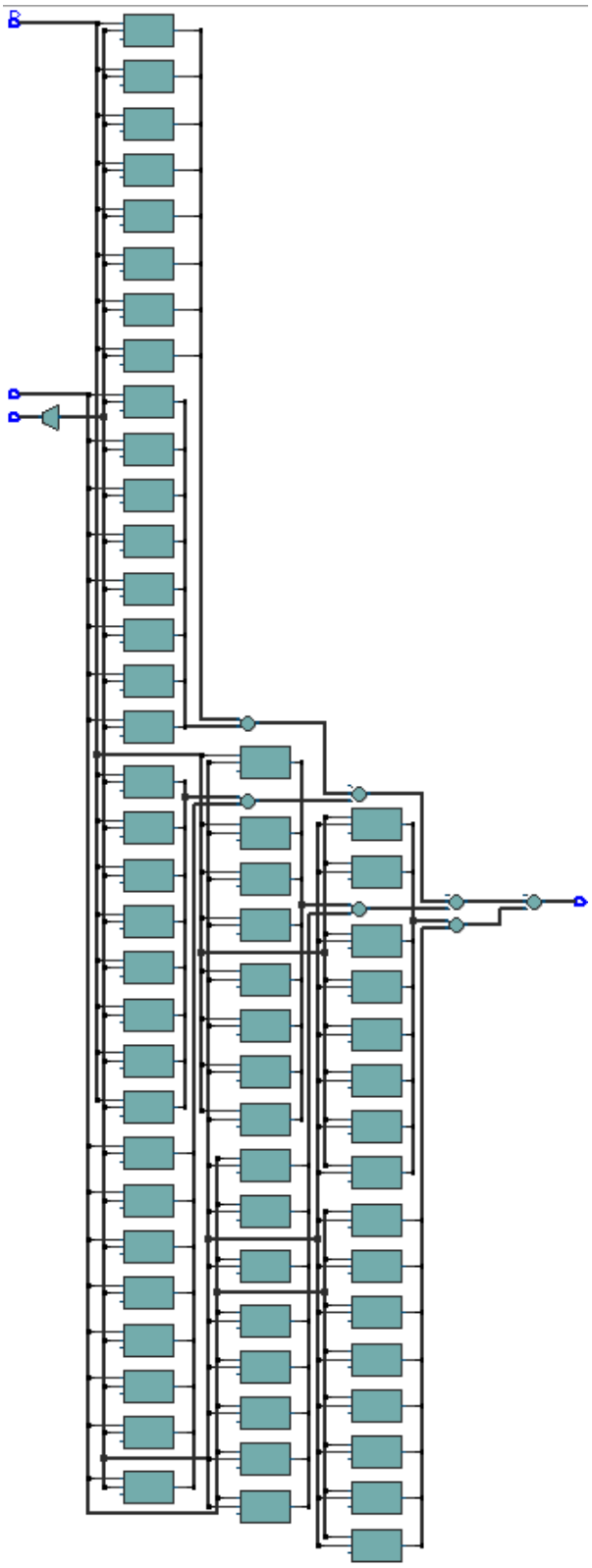

Figura 9: Diagrama RTL del producto punto de dos vectores implementado en Verilog.

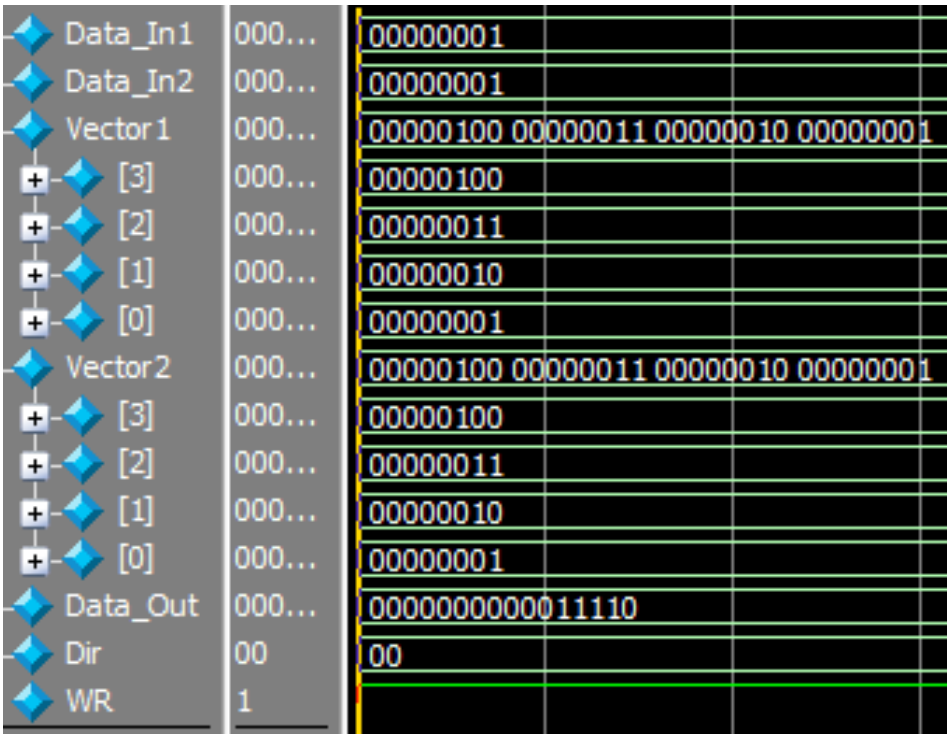

Figura 10: Simulación del producto punto de dos vectores en Verilog con el visor de formas de onda de ModelSim (base binaria).

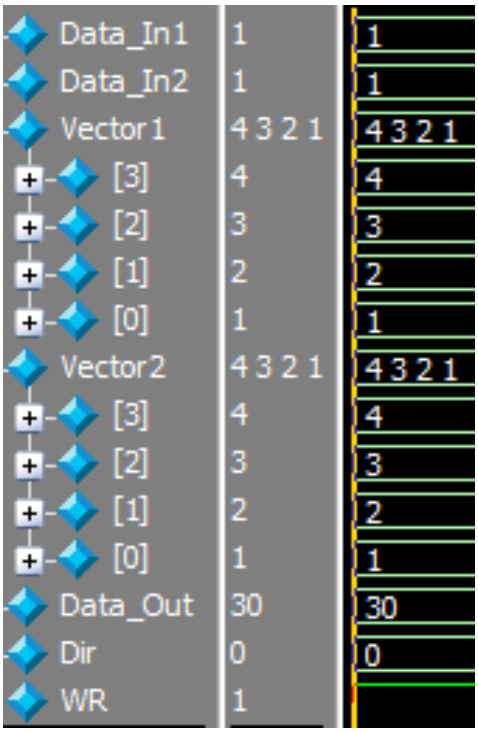

Figura 11: Simulación del producto punto de dos vectores en Verilog con el visor de formas de onda de ModelSim (base decimal).

5. Conclusiones

En conclusión, se implementó el circuito en VHDL y en Verilog de forma correcta.

Se comprendió como funcionan las iteraciones For-Loop para generar circuitos, utilizando descripción por comportamiento. De igual forma se entendió la importancia de estos ciclos iterativos para evitar repetir código y hacerlo más legible.

Se investigó en que manera se implementa el For-Loop en Verilog y se diferenció su sintaxis con la de VHDL.

Se diferenciaron los resultados de inicializar o no, las variables que se usan en operaciones aritméticas.

Se implementó la descripción del producto punto de dos vectores empleando el For-Loop y se observaron con el visor RTL a los circuitos instanciados por el ciclo iterativo, y por medio de las simulaciones de forma de onda en ModelSim se visualizó la correcta operación del módulo.

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

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_signed.all;

entity For_Loop1_VHDL is
 port( Data_In1, Data_In2:  in  std_logic_vector(7 downto 0);
   Dir:      in  std_logic_vector(1 downto 0);
   WR:      in  std_logic;
   Data_Out:    out std_logic_vector(15 downto 0));
end For_Loop1_VHDL;

architecture behavior of For_Loop1_VHDL is

type ram_type is array (3 downto 0) of std_logic_vector(7 downto 0);
signal Vector1: ram_type := (
    0 => "00000000",
    1 => "00000010",
    2 => "00000011",
    3 => "00000100"
);
signal Vector2: ram_type := (
    0 => "00000000",
    1 => "00000010",
    2 => "00000011",
    3 => "00000100"
);

begin

 process(Data_In1, Data_In2, WR)
 begin
  if WR = '1' then
   Vector1(conv_integer(Dir)) <= Data_In1;
   Vector2(conv_integer(Dir)) <= Data_In2;
  end if;
 end process;

 process(Vector1, Vector2)
 variable Var : std_logic_vector(15 downto 0);
```

```
 begin
  --Var := "0000000000000000";
  for i in 0 to 3 loop
   Var := Var + (Vector1(i)*Vector2(i));
  end loop;
  Data_Out <= Var;
 end process;

end behavior;
```

Programa 1: Descripción en VHDL del producto punto de dos vectores, empleando For-Loop.

```
module For_Loop_Verilog (
 input     [7:0]  Data_In1,
 input     [7:0]  Data_In2,
 input     [1:0]  Dir,
 input        WR,
 output reg  [15:0]  Data_Out
);

reg [7:0] Vector1 [3:0];
reg [7:0] Vector2 [3:0];

initial
begin
 Vector1[0] = 0;
 Vector1[1] = 2;
 Vector1[2] = 3;
 Vector1[3] = 4;
 Vector2[0] = 0;
 Vector2[1] = 2;
 Vector2[2] = 3;
 Vector2[3] = 4;
end

always @(WR)
begin
 Vector1[Dir] <= Data_In1;
 Vector2[Dir] <= Data_In2;
end

reg [15:0] Var;
integer i;

always @(*)
begin
```

```
 Var = 0;
 for (i = 0; i < 4; i = i + 1)
 begin
  Var = Var + Vector1[i] * Vector2[i];
 end
 Data_Out = Var;
end
endmodule
```

Programa 2: Descripción en Verilog del producto punto de dos vectores, empleando For-Loop.

6.2. Bancos de pruebas (*Test Benches*)

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

entity For_Loop1_VHDL_vhd_tst IS
end For_Loop1_VHDL_vhd_tst;

architecture For_Loop1_VHDL_arch OF For_Loop1_VHDL_vhd_tst is
signal Data_In1  :   std_logic_vector(7 downto 0);
signal Data_In2  :   std_logic_vector(7 downto 0);
signal Data_Out  :   std_logic_vector(15 downto 0);
signal Dir    :   std_logic_vector(1 downto 0);
signal WR    :   std_logic;

component For_Loop1_VHDL
 port (Data_In1 : in   std_logic_vector(7 downto 0);
   Data_In2 : in   std_logic_vector(7 downto 0);
   Data_Out : buffer std_logic_vector(15 downto 0);
   Dir   : in   std_logic_vector(1 downto 0);
   WR   : in   std_logic);
end component;
begin
 i1 : For_Loop1_VHDL
 port map ( Data_In1 => Data_In1,
     Data_In2 => Data_In2,
     Data_Out => Data_Out,
     Dir   => Dir,
     WR   => WR);

 init : process
 begin
  wait;
 end process init;
```

```
33
34  always : process
35  begin
36   Data_In1 <= "00000001"; Data_In2 <= "00000001"; Dir <= "00"; WR <= '1';
      -- 1 + 4 + 9 + 16 = 30
37   wait for 10ns;
38  end process always;
39 end For_Loop1_VHDL_arch;
```

Programa 3: Banco de prueba para el Programa 1.

```
1  `timescale 1 ns/ 1 ps
2  module For_Loop_Verilog_vlg_tst();
3   reg   [7:0]   Data_In1;
4   reg   [7:0]   Data_In2;
5   reg   [1:0]   Dir;
6   reg      WR;
7   wire   [15:0] Data_Out;
8
9   For_Loop_Verilog i1 (
10   .Data_In1(Data_In1),
11   .Data_In2(Data_In2),
12   .Data_Out(Data_Out),
13   .Dir(Dir),
14   .WR(WR)
15  );
16
17  initial
18  begin
19   Data_In1 = 1; Data_In2 = 1; Dir = 0; WR = 1;
20   $display("Running testbench at CIC");
21  end
22
23  always
24  begin
25  #100; WR = 0;
26  end
27
28 endmodule
```

Programa 4: Banco de prueba para el Programa 2.