Instituto Politécnico Nacional

Centro de Investigación en Computación

Lenguajes de descripción de hardware

Tarea 5 - Descripción estructural y IPCATALOG

Profesor:

M. en C. Osvaldo Espinosa Sosa

Por:

Ing. Ricardo Aldair Tirado Torres

Ciudad de México, 9 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. Descripción estructural de un multiplexor

Actividad 1

Capturar el código para describir en forma estructural el multiplexor en el lenguaje de su elección. Instanciar el módulo o componente que se encuentre en el mismo archivo. Simular el multiplexor.

La visualización RTL del multiplexor usando descripción estructural en Verilog se muestra en la **??**. Como se observa, la implementación se hace con la instanciación de un multiplexor, denominado "MyMux", visualizado en el interior del módulo (Ver **??**). Las simulaciones se visualizan en la **??**, en donde se muestra que el multiplexor descrito opera de manera correcta.

En los Anexos se localiza la descripción en Verilog de este multiplexor. En el código se tienen dos módulos, siendo el primero el de la jerarquía más alta y en donde se realiza la declaración de las entradas y la salida, para luego instanciar al módulo llamado "MyMux" con la etiqueta "u0". Cabe señalar que los argumentos de la instancia se deben colocar en el orden correcto. El segundo módulo unicamente es la descripción de un multiplexor sencillo, utilizando el operador condicional "?".

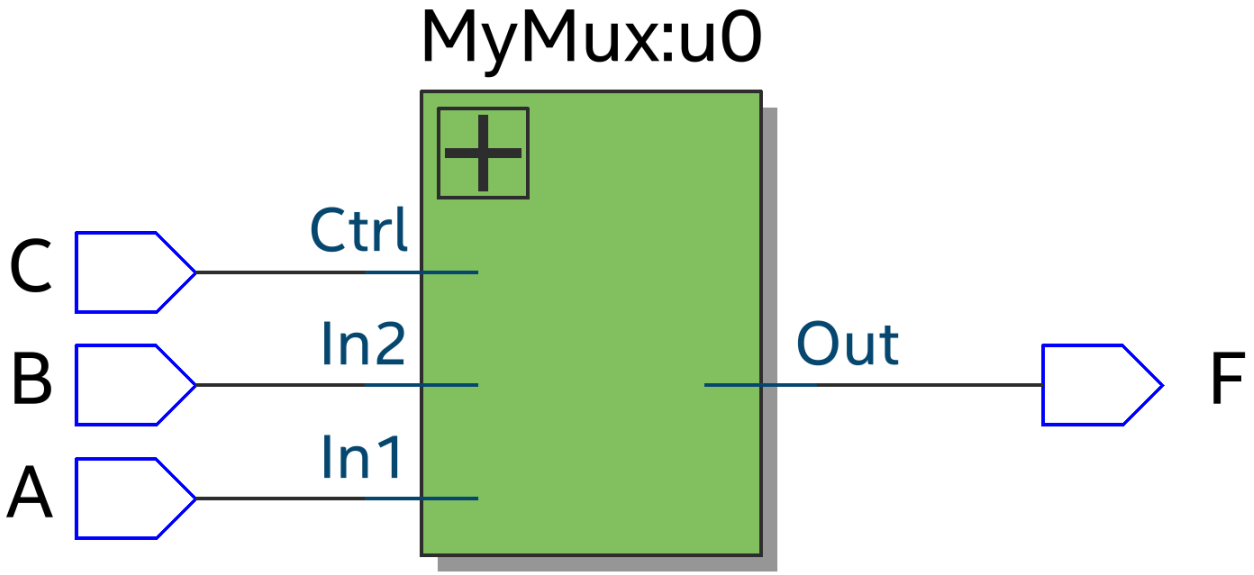

Figura 1: Diagrama RTL del multiplexor descrito en forma estructural en Verilog.

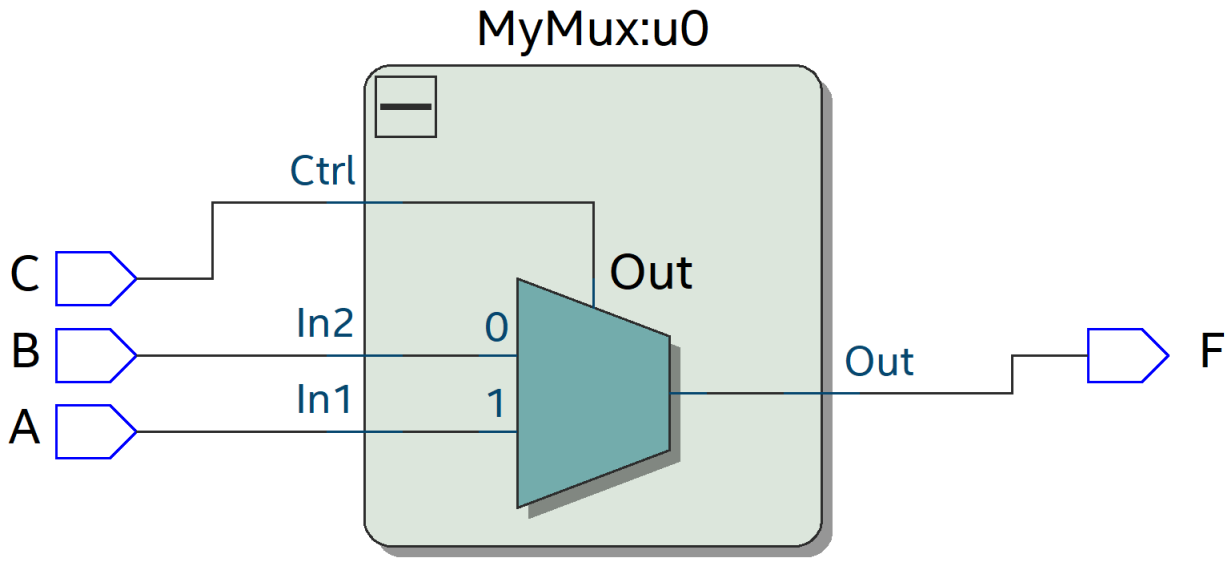

Figura 2: Simulación del multiplexor descrito en forma estructural en Verilog (visualización de la instancia interna).

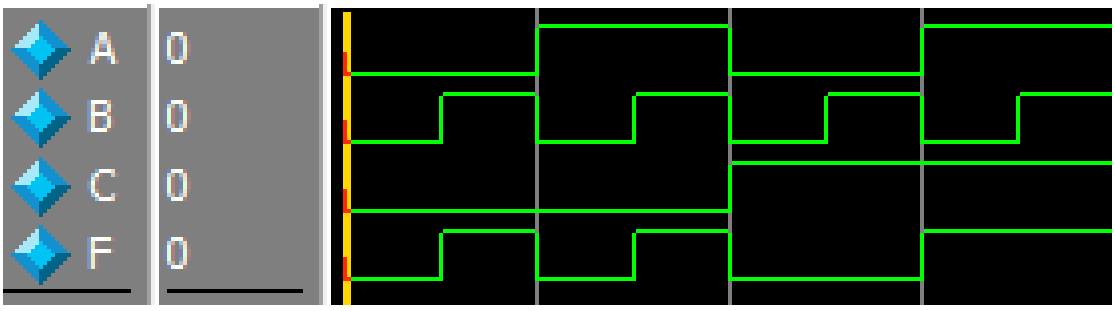

Figura 3: Simulación del multiplexor descrito en forma estructural en Verilog con el visor de formas de onda de ModelSim.

3. 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.

4. Anexos

4.1. Descripciones del hardware

```
module Mux_Structural_Description(
 input  A, B, C,
 output  F);

MyMux u0(A, B, C, F);

endmodule

//Modulo a instanciar
module MyMux(
 input  In1, In2, Ctrl,
 output  Out);

assign Out = Ctrl ? In1 : In2;

endmodule
```

Programa 1: Descripción en Verilog del multiplexor en forma estructural.

```
module Mux_IP_Catalog(
 input  A, B, C,
 output  F);

MyMux u0(A, B, C, F);

endmodule
```

Programa 2: Descripción en Verilog del multiplexor en forma estructural, instanciando un archivo creado con IP Catalog.

4.2. Bancos de pruebas (*Test Benches*)

```
`timescale 1 ns/ 1 ps
module Mux_Structural_Description_vlg_tst();
 reg  A;
 reg  B;
 reg  C;
 wire  F;

 Mux_Structural_Description i1 (
```

```
  .A(A),
  .B(B),
  .C(C),
  .F(F)
 );

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

 always
 begin
  #10; A = 0; B = 1; C = 0;
  #10; A = 1; B = 0; C = 0;
  #10; A = 1; B = 1; C = 0;
  #10; A = 0; B = 0; C = 1;
  #10; A = 0; B = 1; C = 1;
  #10; A = 1; B = 0; C = 1;
  #10; A = 1; B = 1; C = 1;
 end

endmodule
```

Programa 3: Banco de prueba para el Programa 1.

```
`timescale 1 ns/ 1 ps
module Mux_IP_Catalog_vlg_tst();
 reg  A;
 reg  B;
 reg  C;
 wire  F;

 Mux_IP_Catalog i1 (
  .A(A),
  .B(B),
  .C(C),
  .F(F)
 );

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

```
20
21  always
22  begin
23   #10; A = 0; B = 1; C = 0;
24   #10; A = 1; B = 0; C = 0;
25   #10; A = 1; B = 1; C = 0;
26   #10; A = 0; B = 0; C = 1;
27   #10; A = 0; B = 1; C = 1;
28   #10; A = 1; B = 0; C = 1;
29   #10; A = 1; B = 1; C = 1;
30  end
31
32 endmodule
```

Programa 4: Banco de prueba para el Programa 2.