Instituto Politécnico Nacional

Centro de Investigación en Computación

Lenguajes de descripción de hardware

Tarea 6 - Asignación en Verilog

Profesor:

M. en C. Osvaldo Espinosa Sosa

Por:

Ing. Ricardo Aldair Tirado Torres

Ciudad de México, 16 de mayo de 2024

Tabla de contenido

1. Objetivos

- Implementar la descripción en forma estructural para generar un multiplexor sencillo e implementar la instanciación de un módulo creado por el usuario.
- Aprender sobre el uso de la herramienta *IP Catalog* y como se utiliza para instanciar módulos ya proporcionados por el propietario de la herramienta de Quartus.

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 circuito con múltiples asignaciones, descrito en Verilog, se muestra en la Figura 1. La implementación se hace con la instanciación de un multiplexor, denominado "MyMux", visualizado en el interior del módulo. 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 "?".

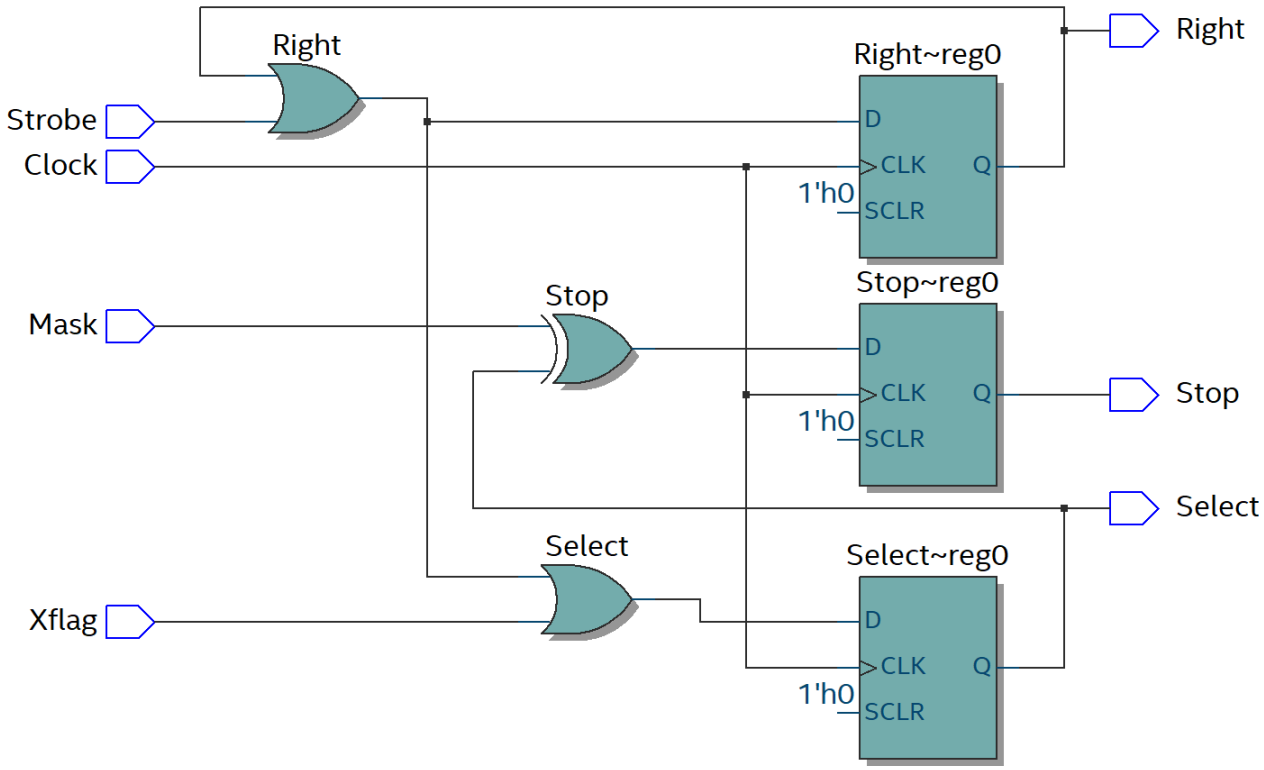

Figura 1: Diagrama RTL del circuito con múltiples asignaciones, descrito en Verilog.

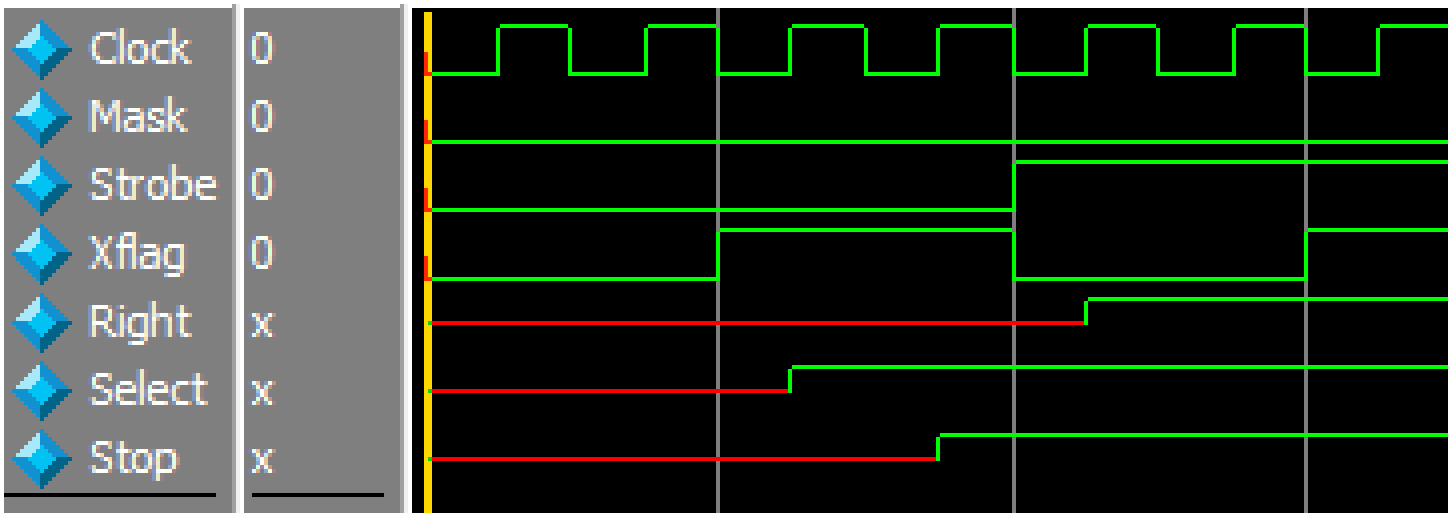

Figura 2: Simulación del circuito con múltiples asignaciones, descrito en Verilog, con el visor de formas de onda de ModelSim.

3. Descripción estructural de un multiplexor con *IP Catalog*

Actividad 2

Describir el multiplexor en forma estructural usando una función creada por el *IP Catalog* siguiendo los pasos descritos en el documento o presentación. Simular el multiplexor.

La visualización RTL del circuito con múltiples asignaciones, descrito en Verilog, se muestra en la Figura 3. La implementación se hace con la instanciación de un multiplexor, denominado "MyMux", visualizado en el interior del módulo. Las simulaciones se visualizan en la Figura 4, 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 "?".

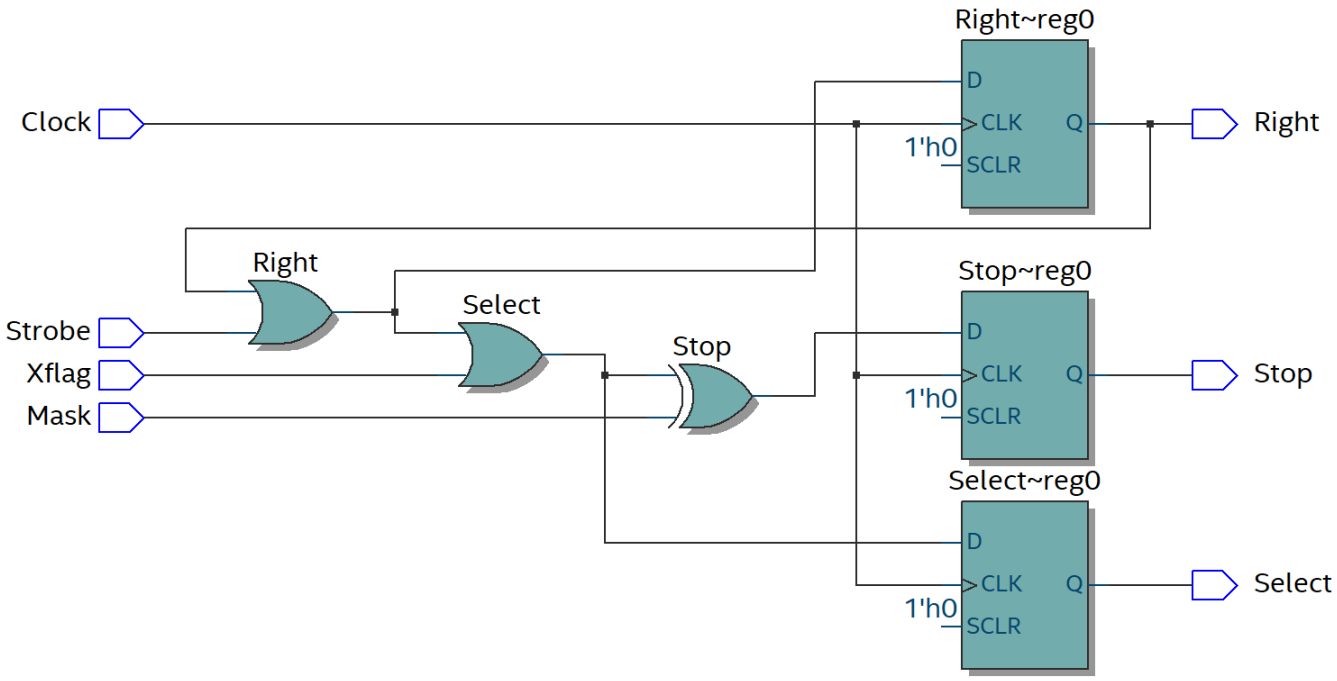

Figura 3: Diagrama RTL del circuito con múltiples asignaciones, descrito en Verilog.

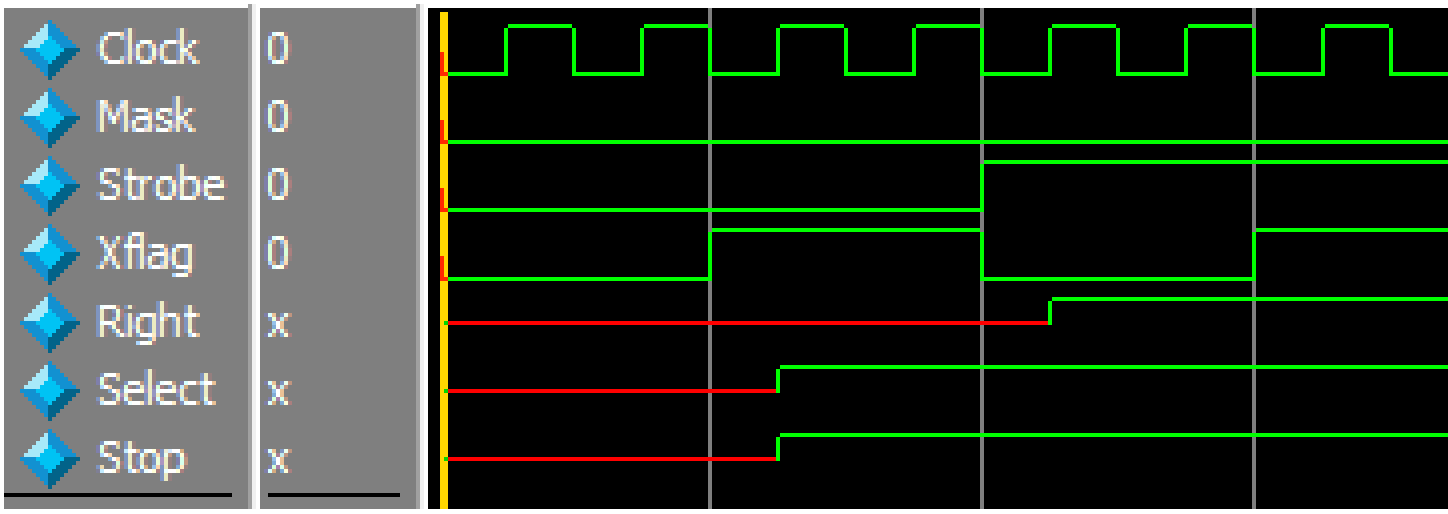

Figura 4: Simulación del circuito con múltiples asignaciones, descrito en Verilog, con el visor de formas de onda de ModelSim.

4. Descripción estructural de un multiplexor con *IP Catalog*

Actividad 2

Describir el multiplexor en forma estructural usando una función creada por el *IP Catalog* siguiendo los pasos descritos en el documento o presentación. Simular el multiplexor.

La visualización RTL del circuito con múltiples asignaciones, descrito en Verilog, se muestra en la Figura 5. La implementación se hace con la instanciación de un multiplexor, denominado "MyMux", visualizado en el interior del módulo. Las simulaciones se visualizan en la Figura 6, 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 "?".

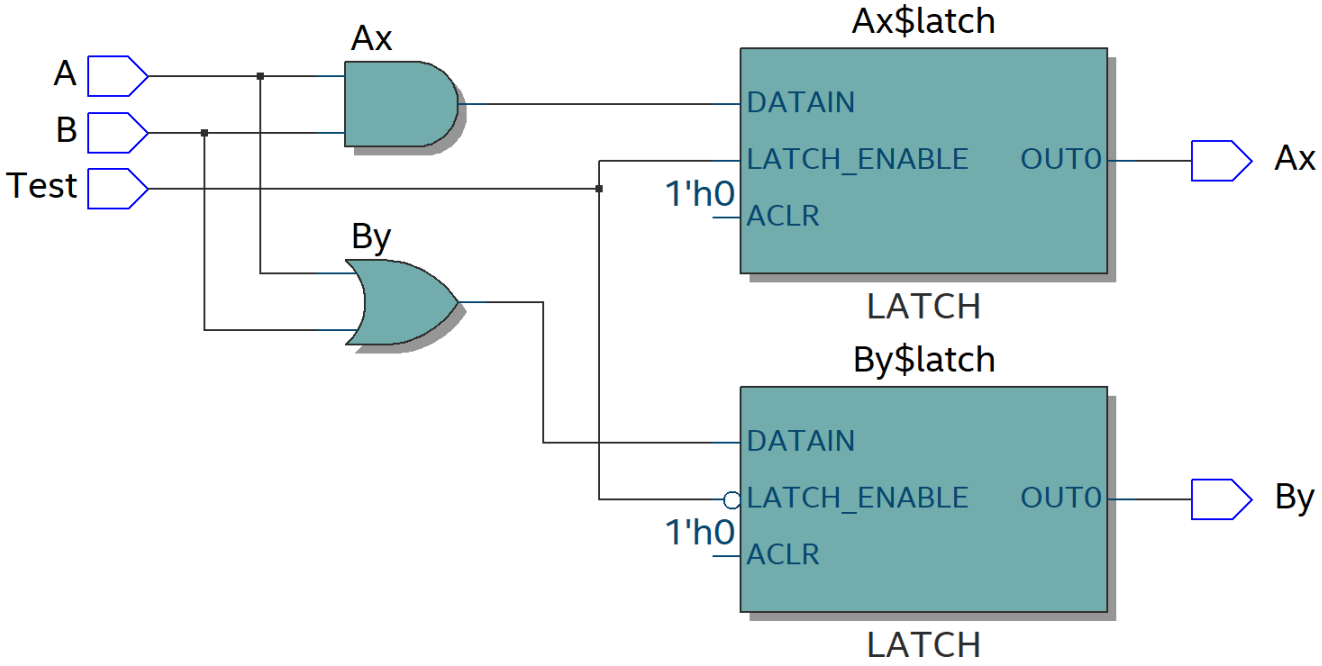

Figura 5: Diagrama RTL del circuito con múltiples asignaciones, descrito en Verilog.

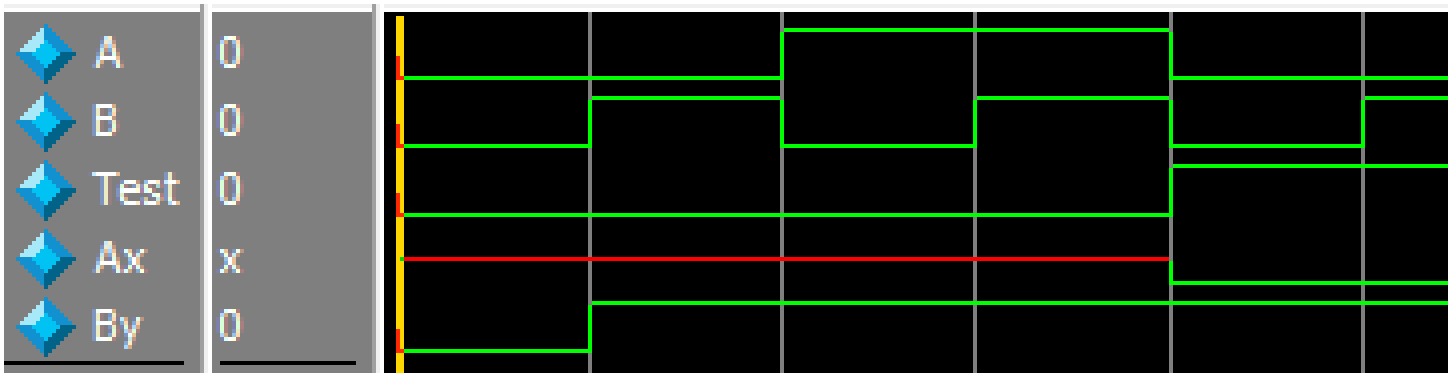

Figura 6: Simulación del circuito con múltiples asignaciones, descrito en Verilog, con el visor de formas de onda de ModelSim.

5. Conclusiones

En conclusión, se implementó el circuito del multiplexor, en Verilog, de manera correcta.

Se comprendió como se utiliza la descripción estructural para instanciar módulos ya creados por el usuario, con el fin de generar hardware más complejo.

Se comprendió la función de la herramienta *IP Catalog* y se instanció un módulo de multiplexor simple como parte de otro módulo de mayor jerarquía.

En ambos casos se implementó un multiplexor sencillo con descripción estructural y se observó con el visor RTL a los circuitos instanciados dentro del módulo principal, y por medio de las simulaciones de forma de onda en ModelSim, se visualizó la correcta operación del dispositivo.

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

6. Anexos

6.1. Descripciones del hardware

```
module Assignment_Circuit1(
input    Clock, Strobe, Xflag, Mask,
output reg  Right, Select, Stop);

always @(posedge Clock)
begin
   Right = Right | Strobe;
   Select <= Right | Xflag;
   Stop <= Select ^Mask;
end
endmodule
```

Programa 1: Descripción en Verilog del código presentado en clase.

```
module Assignment_Circuit2(
input    Clock, Strobe, Xflag, Mask,
output reg  Right, Select, Stop);

always @(posedge Clock)
begin
   Right = Right | Strobe;
   Select = Right | Xflag;
   Stop <= Select ^Mask;
end
endmodule
```

Programa 2: Descripción en Verilog del código presentado en clase (modificando la segunda asignación).

```
module Assignment_Circuit3(
input    A, B, Test,
output reg Ax, By);

always @(*)
begin
 if (Test)
  Ax = A & B;
 else
  By = A | B;
end
```

```
endmodule
```

Programa 3: Descripción en Verilog del código con estructura *if-else*

6.2. Bancos de pruebas (*Test Benches*)

```
`timescale 1 ns/ 1 ps
module Assignment_Circuit1_vlg_tst();
 reg Clock;
 reg Mask;
 reg Strobe;
 reg Xflag;
 wire Right;
 wire Select;
 wire Stop;

 Assignment_Circuit1 i1 (
  .Clock(Clock),
  .Mask(Mask),
  .Right(Right),
  .Select(Select),
  .Stop(Stop),
  .Strobe(Strobe),
  .Xflag(Xflag)
 );

 initial
 begin
  Clock = 0;  Mask = 0; Strobe = 0; Xflag = 0;
  #20;   Mask = 0; Strobe = 0; Xflag = 1;
  #20;   Mask = 0; Strobe = 1; Xflag = 0;
  #20;   Mask = 0; Strobe = 1; Xflag = 1;
  #20;   Mask = 1; Strobe = 0; Xflag = 0;
  #20;   Mask = 1; Strobe = 0; Xflag = 1;
  #20;   Mask = 1; Strobe = 1; Xflag = 0;
  #20;   Mask = 1; Strobe = 1; Xflag = 1;
  $display("Running testbench at CIC");
 end

 always
 begin
  #5; Clock = ~Clock;
 end

```

```
endmodule
```

Programa 4: Banco de prueba para el Programa 1.

```
`timescale 1 ns/ 1 ps
module Assignment_Circuit2_vlg_tst();
 reg Clock;
 reg Mask;
 reg Strobe;
 reg Xflag;
 wire Right;
 wire Select;
 wire Stop;

 Assignment_Circuit2 i1 (
  .Clock(Clock),
  .Mask(Mask),
  .Right(Right),
  .Select(Select),
  .Stop(Stop),
  .Strobe(Strobe),
  .Xflag(Xflag)
 );

 initial
 begin
  Clock = 0;  Mask = 0; Strobe = 0; Xflag = 0;
  #20;   Mask = 0; Strobe = 0; Xflag = 1;
  #20;   Mask = 0; Strobe = 1; Xflag = 0;
  #20;   Mask = 0; Strobe = 1; Xflag = 1;
  #20;   Mask = 1; Strobe = 0; Xflag = 0;
  #20;   Mask = 1; Strobe = 0; Xflag = 1;
  #20;   Mask = 1; Strobe = 1; Xflag = 0;
  #20;   Mask = 1; Strobe = 1; Xflag = 1;
  $display("Running testbench at CIC");
 end

 always
 begin
  #5; Clock = ~Clock;
 end

endmodule
```

Programa 5: Banco de prueba para el Programa 2.

```
`timescale 1 ns/ 1 ps
module Assignment_Circuit3_vlg_tst();
 reg   A;
 reg   B;
 reg   Test;
 wire   Ax;
 wire   By;

 Assignment_Circuit3 i1 (
  .A(A),
  .Ax(Ax),
  .B(B),
  .By(By),
  .Test(Test)
 );

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

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

endmodule
```

Programa 6: Banco de prueba para el Programa 3.