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, 19 de mayo de 2024

Tabla de contenido

1. Objetivos

- Compilar una descripción vista en clase, para entender como se implementa la asignación inmediata y no inmediata de las señales de un circuito.
- Analizar, con el visor RTL, la manera en que se conectan las señales, de acuerdo con el tipo de asignación
- Aprender sobre el uso de los Latches, para la asignación de señales de salida y como es que estas señales se pueden habilitar y deshabilitar de forma independiente.
- Diferenciar la forma en la que opera un Latch de un Flip-Flop.

2. Descripción de circuito con diferentes operadores de asignación

Actividad 1

Capturar el código de la lámina 7 de la presentación de clase (Asignación en verilog (6)). Compilar y observar el resultado de la síntesis con el visor RTL.

La visualización RTL del circuito con múltiples asignaciones, descrito en Verilog, se muestra en la Figura 1. La implementación se hace empleando las compuertas lógicas descritas en el código, no obstante, como se utilizaron operadores de asignación bloqueante y no bloqueante, se usan Latches en las señales de salida. Nótese que la señal *Right* como es de asignación inmediata, la salida de la compuerta OR se conecta directamente a la entrada de la compuerta OR, pero de la señal *Select*, en cambio, *Select*, al tener asignación no inmediata, se conecta la salida del Latch a la entrada de la compuerta XOR de la señal *Mask*. Las simulaciones se visualizan en la Figura 2, en donde se muestra que el módulo opera de manera adecuada, siendo que los Latches funcionan con el cambio del nivel lógico de la señal de *Clock* y no con los flancos de subida o bajada, como los Flip-Flop.

En los Anexos se localiza la descripción en Verilog de este módulo. En el código se tiene la declaración de entradas y salidas junto con una lista sensible a los flancos de subida de la señal de *Clock*. Dentro de la estructura *always*, se realiza una asignación inmediata, de la señal *Right*, y dos no inmediatas, de las señales *Select* y *Mask*.

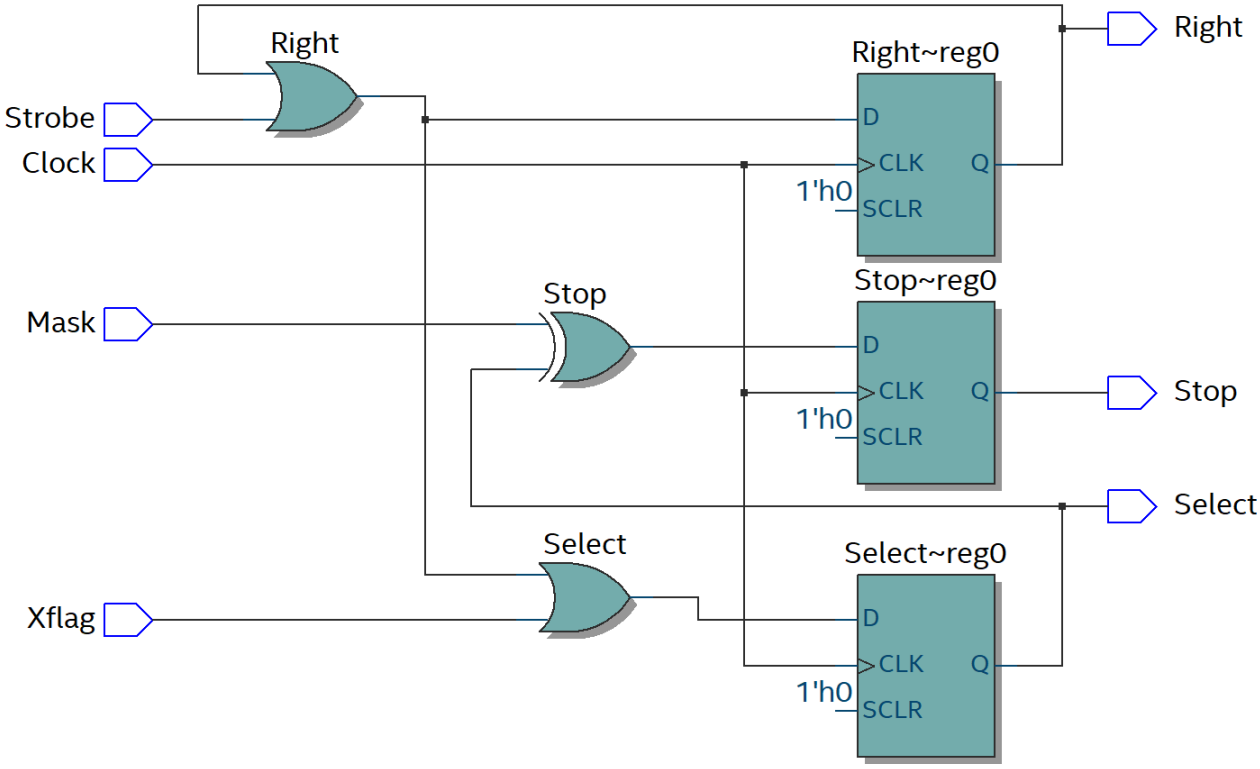

Figura 1: Diagrama RTL del circuito con múltiples asignaciones, descrito en Verilog.

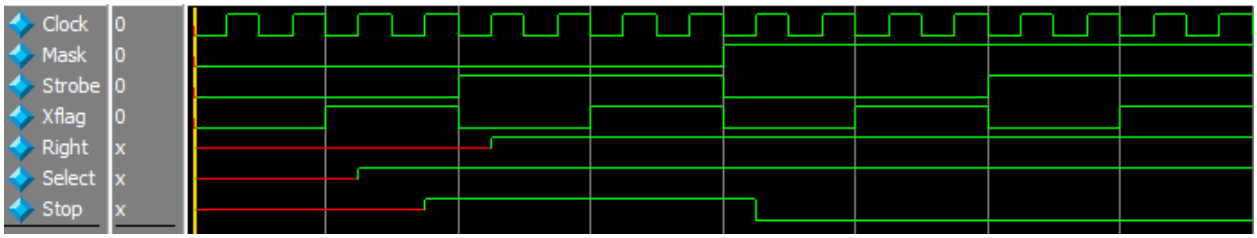

Figura 2: Simulación del circuito con múltiples asignaciones, descrito en Verilog, con el visor de formas de onda de ModelSim.

3. Descripción de circuito con diferentes operadores de asignación (modificado)

Actividad 2

Cambiar el operador de asignación de la segunda sentencia dentro del bloque *"always"*. Compilar y observar el resultado de la síntesis con el visor RTL. Comparar con el resultado del inciso 1.

La visualización RTL del circuito modificado con múltiples asignaciones, descrito en Verilog, se muestra en la Figura 3. La implementación se hace empleando las mismas compuertas lógicas y Latches, no obstante, la conexión es diferente, siendo que, una de las entradas de compuerta XOR proviene de la salida de la compuerta OR de la señal *Select*, y no del Latch dispuesto para la misma señal. Las simulaciones se visualizan en la Figura 4, en donde se muestra que el módulo opera de manera diferente, en comparación con el módulo anterior.

En los Anexos se localiza la descripción en Verilog de este módulo. En el código solo se realiza la modificación de la señal *Select*, utilizando un operador de asignación bloqueante.

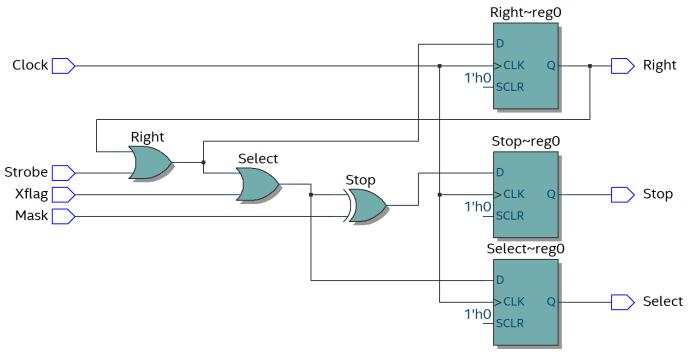

Figura 3: Diagrama RTL del circuito con múltiples asignaciones, descrito en Verilog (versión modificada).

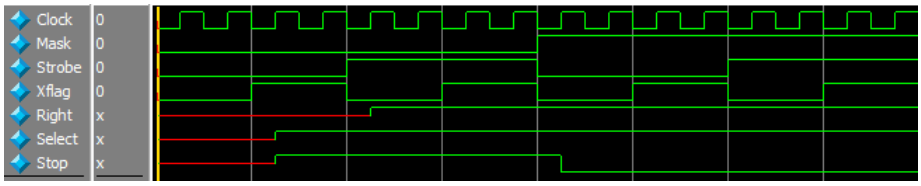

Figura 4: Simulación del circuito con múltiples asignaciones, descrito en Verilog, con el visor de formas de onda de ModelSim (versión modificada).

4. Módulo con estructura *if-else* incompleta

Actividad 3

¿Cuál será el resultado de compilar la estructura *if"* de la última lámina de la presentación? Las estructuras *if"* donde en cada rama se asigna valor a una señal diferente son válidas en los lenguajes de descripción. Completar el código y compilar. Observar el resultado de la síntesis con el visor RTL. Comentar.

La visualización RTL del circuito con estructura *if-else*, descrito en Verilog, se muestra en la Figura 5. La implementación se hace empleando las compuertas lógicas descritas en el código, junto con un Latch en cada salida. Se observa que la señal *Test*, se conecta a la terminal de habilitación de los Latches, por lo que, de esta forma, se realiza la asignación de una sola señal de salida, dependiendo de esta señal de control (nótese que la terminal de habilitación para la señal *By*, toma a la señal de control negada). Las simulaciones se visualizan en la Figura 6, en donde se muestra que el módulo opera de manera adecuada, asignando unicamente el valor a una de las dos salidas, dependiendo del valor lógico de *Test*.

En los Anexos se localiza la descripción en Verilog de este módulo. En el código se tiene la declaración de entradas y salidas junto con una lista sensible a los cambios en las señales de entrada. Dentro de la estructura *always*, se realiza la asignación inmediata de una señal u otra, dependiendo del valor del *Test*, usando para ello una estructura *if-else*.

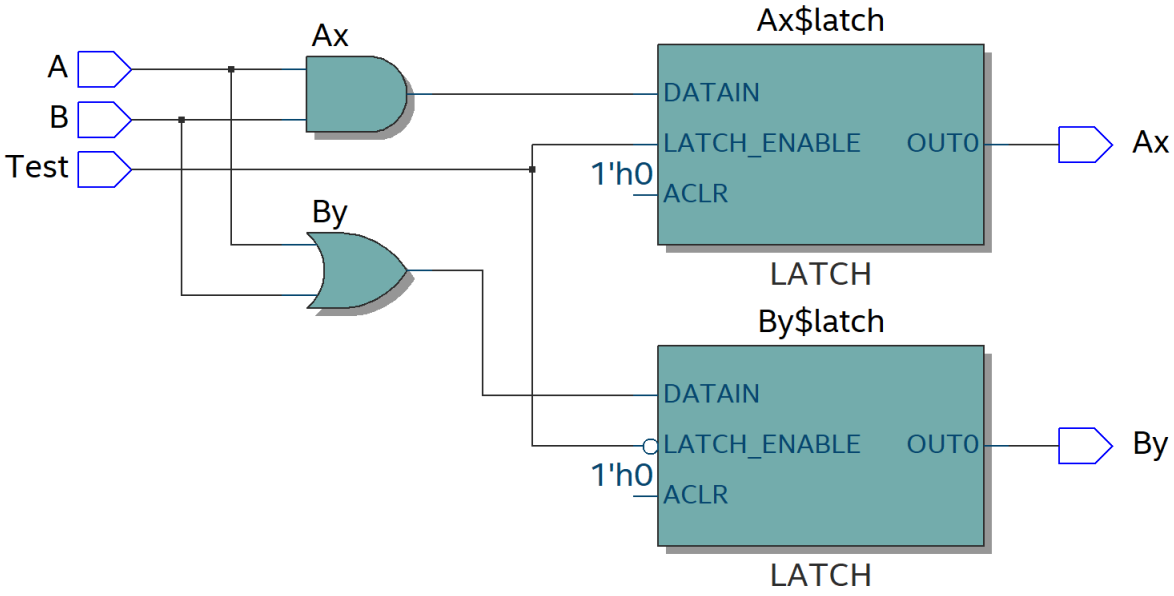

Figura 5: Diagrama RTL del circuito con estructura *if-else*, descrito en Verilog.

Figura 6: Simulación del circuito con estructura *if-else*, descrito en Verilog, con el visor de formas de onda de ModelSim.

5. Conclusiones

En conclusión, se implementaron los tres circuitos de manera adecuada.

Se comprendió como se utilizan los Latches para las asignaciones no inmediatas en un circuito sencillo.

Se diferenciaron a las señales asignadas con operadores bloqueantes y no bloqueantes, gracias a que se visualizó con el visor RTL la manera en que se conectan y operan.

Se comprendió la diferencia de usar un Latch y un Flip-Flop, siendo que el primero opera con niveles lógicos, mientras que el segundo lo hace con flancos de subida o de bajada.

Se entendió un uso interesante de los Latches para habilitar o deshabilitar señales de salida de un circuito.

En todos los casos, se observó con el visor RTL la manera en que se conectan los circuitos, y por medio de las simulaciones de forma de onda en ModelSim, se visualizó la correcta operación de cada 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.