



Microtecnología y
Sistemas Embebidos

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, 18 DE MAYO DE 2024

Tabla de contenido

1. Objetivos	2
2. Descripción de circuito con diferentes operadores de asignación	3
3. Descripción de circuito con diferentes operadores de asignación (modificado)	5
4. Estructura	7
5. Conclusiones	9
6. Anexos	10
6.1. Descripciones del hardware	10
6.2. Bancos de pruebas (<i>Test Benches</i>)	11

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 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 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 únicamente es la descripción de un multiplexor sencillo, utilizando el operador condicional “?”.

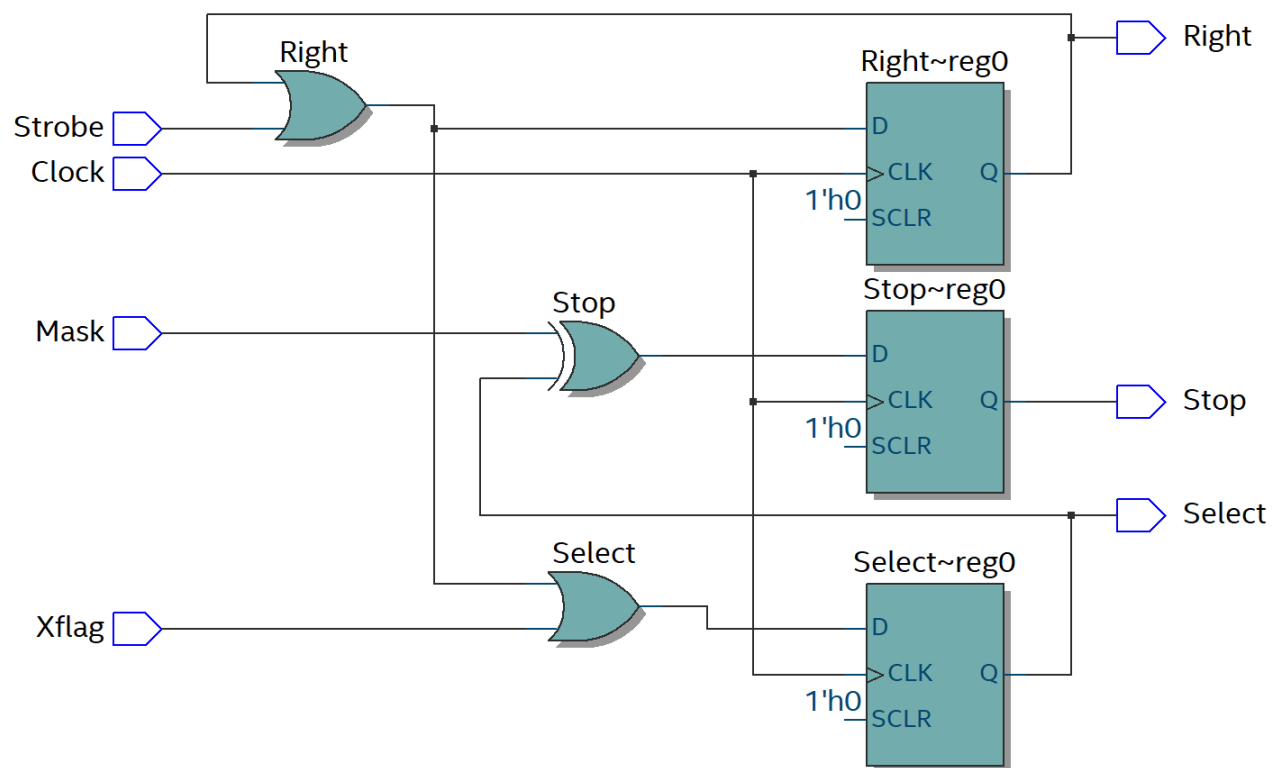


Figura 1: Diagrama RTL del circuito con múltiples asignaciones, descrito en Verilog.

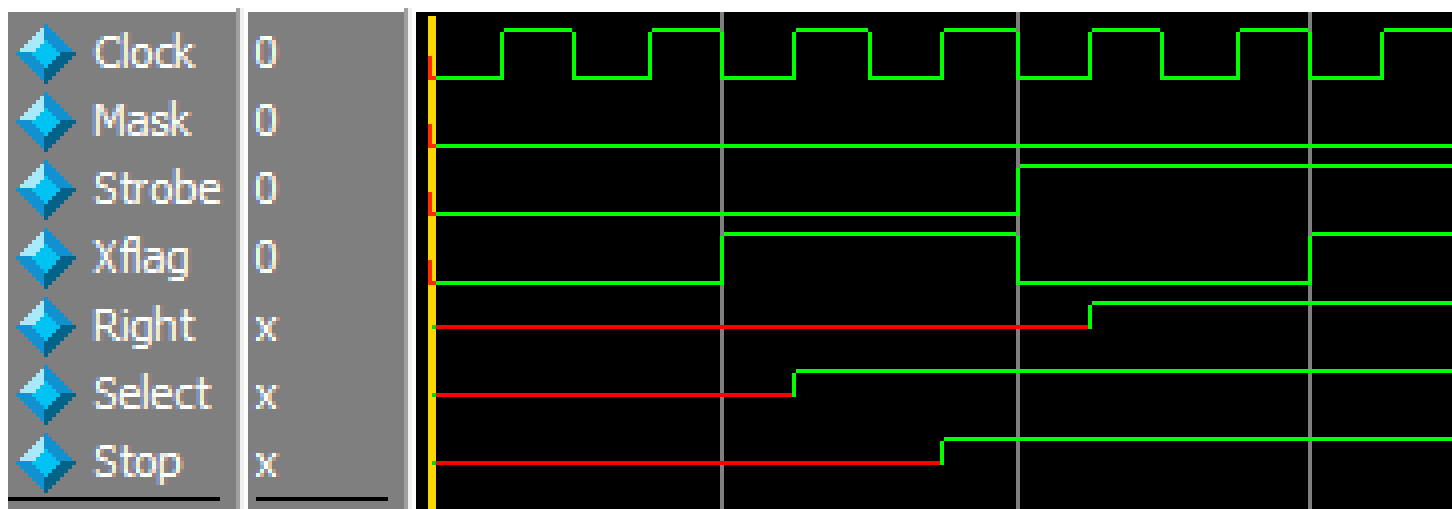


Figura 2: Simulación del circuito con múltiples asignaciones, descrito en Verilog, con el visor de formas de onda de ModelSim.

3. Descripción 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 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 únicamente es la descripción de un multiplexor sencillo, utilizando el operador condicional “?”.

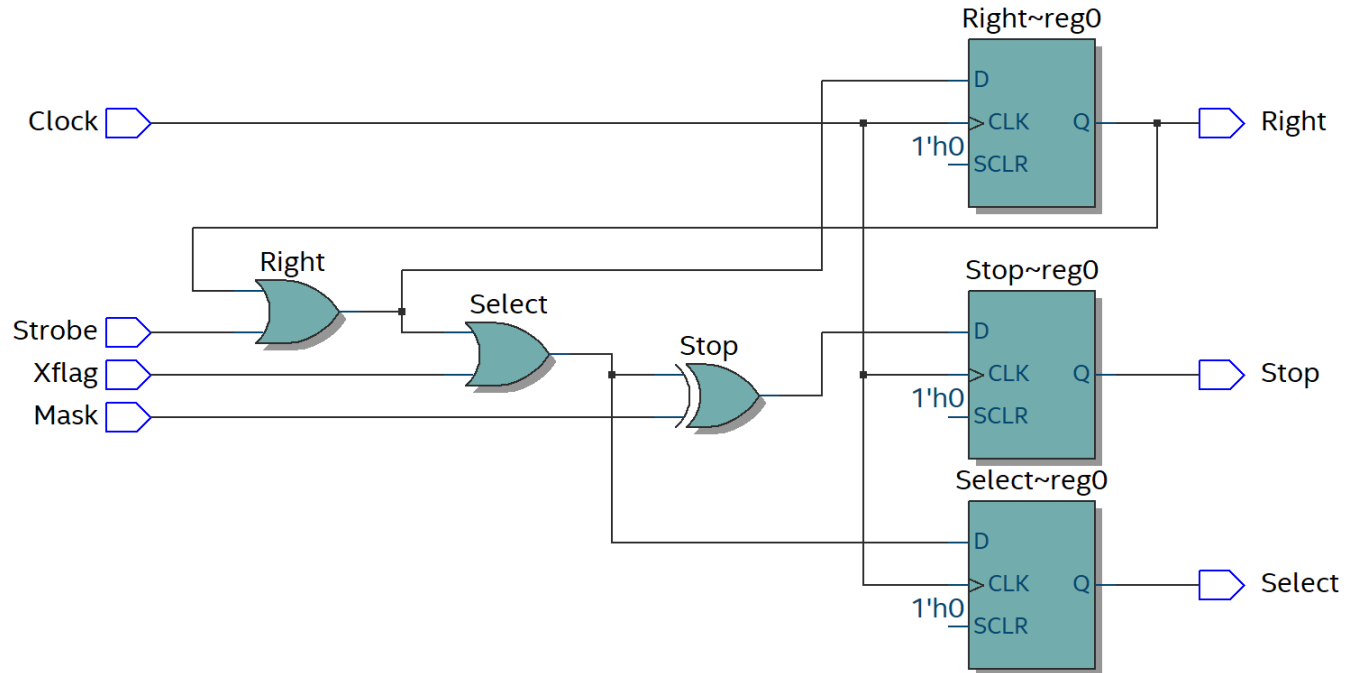


Figura 3: Diagrama RTL del circuito con múltiples asignaciones, descrito en Verilog (versión modificada).

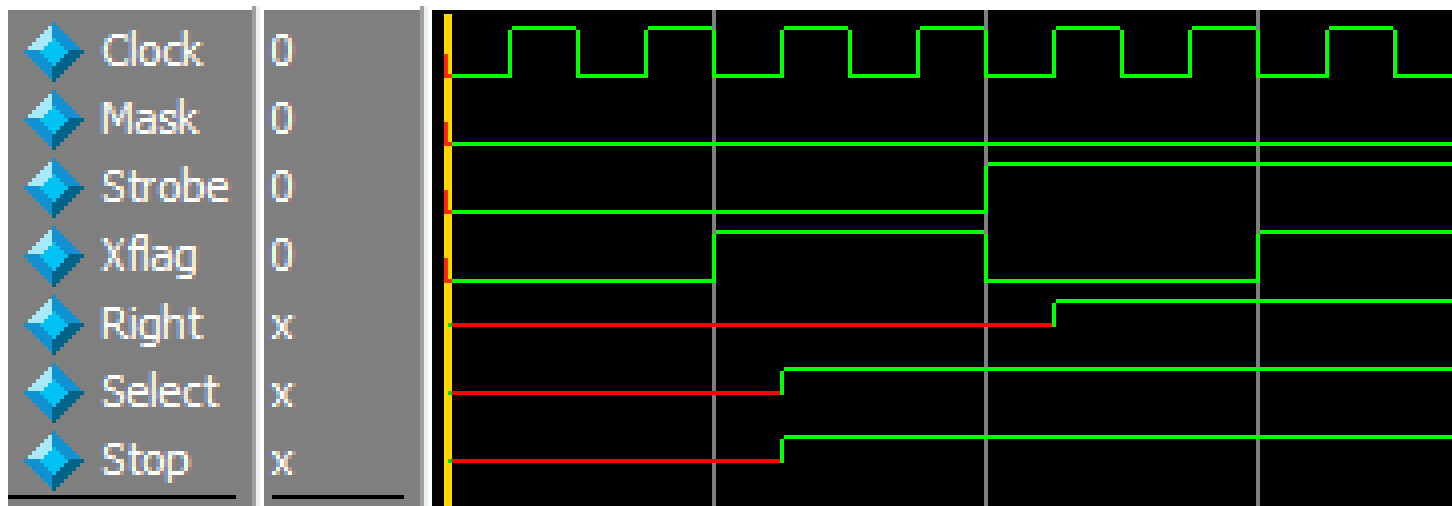


Figura 4: Simulación del circuito con múltiples asignaciones, descrito en Verilog, con el visor de formas de onda de ModelSim (versión modificada).

4. Estructura

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 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 únicamente es la descripción de un multiplexor sencillo, utilizando el operador condicional “?”.

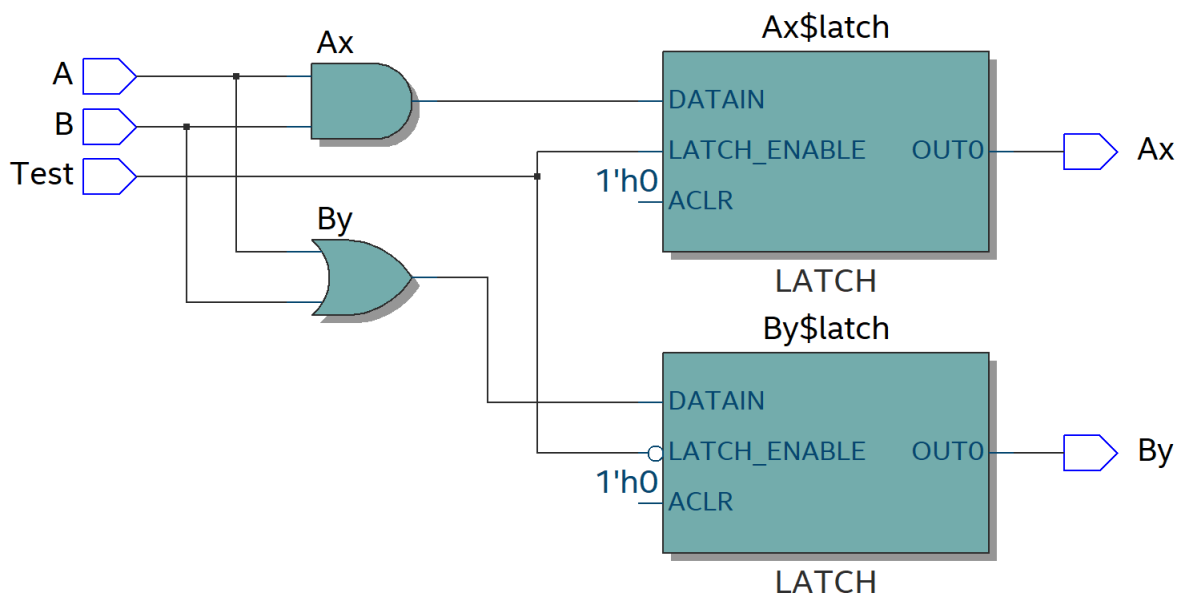


Figura 5: Diagrama RTL del circuito con múltiples asignaciones, descrito en Verilog.



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

```
1 module Assignment_Circuit1(  
2 input    Clock, Strobe, Xflag, Mask,  
3 output reg  Right, Select, Stop);  
4  
5 always @(posedge Clock)  
6 begin  
7     Right = Right | Strobe;  
8     Select <= Right | Xflag;  
9     Stop <= Select ^Mask;  
10 end  
11 endmodule
```

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

```
1 module Assignment_Circuit2(  
2 input    Clock, Strobe, Xflag, Mask,  
3 output reg  Right, Select, Stop);  
4  
5 always @(posedge Clock)  
6 begin  
7     Right = Right | Strobe;  
8     Select = Right | Xflag;  
9     Stop <= Select ^Mask;  
10 end  
11 endmodule
```

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

```
1 module Assignment_Circuit3(  
2 input    A, B, Test,  
3 output reg Ax, By);  
4  
5 always @(*)  
6 begin  
7     if (Test)  
8         Ax = A & B;  
9     else  
10        By = A | B;  
11 end
```

```
12 endmodule
```

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

6.2. Bancos de pruebas (*Test Benches*)

```
1  `timescale 1 ns/ 1 ps
2  module Assignment_Circuit1_vlg_tst();
3      reg Clock;
4      reg Mask;
5      reg Strobe;
6      reg Xflag;
7      wire Right;
8      wire Select;
9      wire Stop;
10
11      Assignment_Circuit1 i1 (
12          .Clock(Clock),
13          .Mask(Mask),
14          .Right(Right),
15          .Select(Select),
16          .Stop(Stop),
17          .Strobe(Strobe),
18          .Xflag(Xflag)
19      );
20
21      initial
22      begin
23          Clock = 0;  Mask = 0; Strobe = 0; Xflag = 0;
24          #20;  Mask = 0; Strobe = 0; Xflag = 1;
25          #20;  Mask = 0; Strobe = 1; Xflag = 0;
26          #20;  Mask = 0; Strobe = 1; Xflag = 1;
27          #20;  Mask = 1; Strobe = 0; Xflag = 0;
28          #20;  Mask = 1; Strobe = 0; Xflag = 1;
29          #20;  Mask = 1; Strobe = 1; Xflag = 0;
30          #20;  Mask = 1; Strobe = 1; Xflag = 1;
31          $display("Running testbench at CIC");
32      end
33
34      always
35      begin
36          #5; Clock = ~Clock;
37      end
38
```

```
39 endmodule
```

Programa 4: Banco de prueba para el Programa 1.

```
1  `timescale 1 ns/ 1 ps
2  module Assignment_Circuit2_vlg_tst();
3      reg Clock;
4      reg Mask;
5      reg Strobe;
6      reg Xflag;
7      wire Right;
8      wire Select;
9      wire Stop;
10
11      Assignment_Circuit2 i1 (
12          .Clock(Clock),
13          .Mask(Mask),
14          .Right(Right),
15          .Select(Select),
16          .Stop(Stop),
17          .Strobe(Strobe),
18          .Xflag(Xflag)
19      );
20
21      initial
22      begin
23          Clock = 0; Mask = 0; Strobe = 0; Xflag = 0;
24          #20; Mask = 0; Strobe = 0; Xflag = 1;
25          #20; Mask = 0; Strobe = 1; Xflag = 0;
26          #20; Mask = 0; Strobe = 1; Xflag = 1;
27          #20; Mask = 1; Strobe = 0; Xflag = 0;
28          #20; Mask = 1; Strobe = 0; Xflag = 1;
29          #20; Mask = 1; Strobe = 1; Xflag = 0;
30          #20; Mask = 1; Strobe = 1; Xflag = 1;
31          $display("Running testbench at CIC");
32      end
33
34      always
35      begin
36          #5; Clock = ~Clock;
37      end
38
39 endmodule
```

Programa 5: Banco de prueba para el Programa 2.

```

1 'timescale 1 ns/ 1 ps
2 module Assignment_Circuit3_vlg_tst();
3   reg  A;
4   reg  B;
5   reg  Test;
6   wire Ax;
7   wire By;
8
9   Assignment_Circuit3 i1 (
10    .A(A),
11    .Ax(Ax),
12    .B(B),
13    .By(By),
14    .Test(Test)
15  );
16
17  initial
18  begin
19    Test = 0; A = 0; B = 0;
20    $display("Running testbench at CIC");
21  end
22
23  always
24  begin
25    #10; Test = 0; A = 0; B = 1;
26    #10; Test = 0; A = 1; B = 0;
27    #10; Test = 0; A = 1; B = 1;
28    #10; Test = 1; A = 0; B = 0;
29    #10; Test = 1; A = 0; B = 1;
30    #10; Test = 1; A = 1; B = 0;
31    #10; Test = 1; A = 1; B = 1;
32  end
33
34 endmodule

```

Programa 6: Banco de prueba para el Programa 3.