Hw #05 { Detalles de asignación en verilog }
Lenguajes de descripción de hardware

Hernández Castellanos César Uriel

1 Sección uno

Cuando se realiza una asignación en el lenguaje verilog, usualmente usamos el operador "=", por ejemplo:

assign salida = a + b;

Pero cuando la asignación se realiza dentro de un bloque de descripción por comportamiento puede ser de dos tipos

Blocking = (lógica combinatoria)

Non blocking <= (lógica secuencial)

El operador blocking solemos usarlo para describir lógica combinatoria y el non blocking lo utilizamos en descripción de lógica secuencial.
En el siguiente código indicar el tipo de operador que debe usarse en la asignación.

```
module ejemplo(A,B,salida);
input [3:0] A,B;
output reg salida;

always @(A,B)
        salida = A&B;
endmodule
```

Indicar el operador que debe utilizarse, ¿Qué circuito resultará después de la síntesis? Utilizar el visor RTL viewer.

1.1 Visor RTL

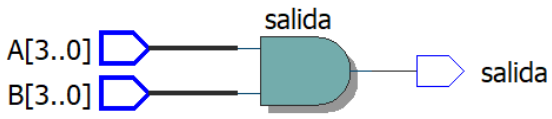

Figure 1: *Resultado del visor RTL*

1.2 Simulación

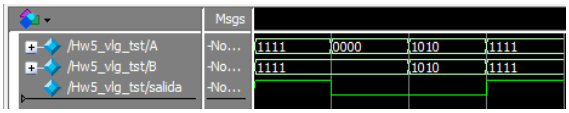

Figure 2: *Resultado de la simulación*

1.3 Sección dos

En el siguiente código indicar el tipo de operador que debe usarse en la asignación

```
module ejemplo(A,B,clock,salida);
input [3:0] A,B;
input clock;
output reg salida;

always @(posedge clock)
        salida <= A&B;
endmodule
```

Indicar el operador que debe utilizarse, ¿Qué circuito resultará después de la síntesis? Utilizar el visor RTL viewer.

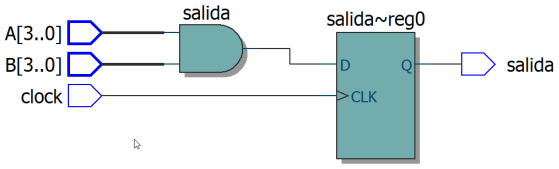

Figure 3: *Resultado del visor RTL*

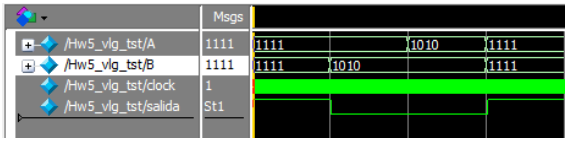

Figure 4: *Resultado de la simulación*

En general, en las asignaciones dentro de un bloque "al-

ways" hemos dicho que para describir lógica combinatoria debemos usar operador blocking y para secuencial non blocking. Cuando en los bloques o en las estructuras se tiene una sola sentencia, el compilador puede inferir el mismo circuito aunque usemos cualquiera de los dos operadores de asignación, sin embargo cuando se tienen bloques de dos o mas sentencias es muy importante estar seguros del operador a utilizar ya que produce un resultado diferente después de la síntesis. Considerar el siguiente código:

```
module circuito(clock,strobe,xflag,
                mask,right, select,
                stop);
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
```

¿Qué resultará de la síntesis del código anterior? Intentar describir el circuito resultante con solo ver el código y sin ver el visor RTL viewer.

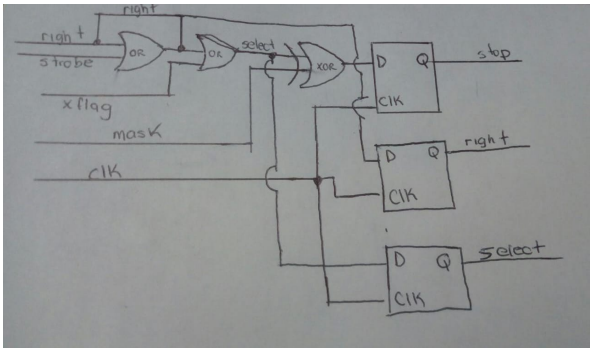

Figure 5: *Descripción uno del circuito*

.

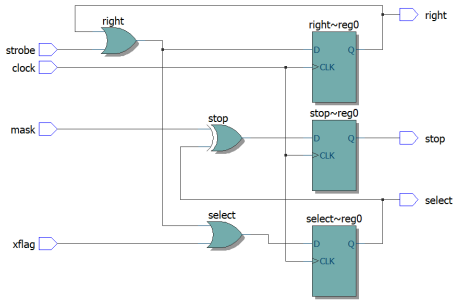

Figure 6: *Descripción dos del circuito*

Después de compilar el código usar el visor RTL viewer, ¿difiere lo mostrado en el visor del circuito descrito por ustedes?

Sí, el circuito difiera en el origen de la retroalimentación de las compuertas lógicas, mi descripción del circuito lo toma directamente de la salida de la compuerta y el visor RTL lo obtiene de los Flip-Flop tipo D.

Cambiar en la línea siguiente del código el operador de asignación:

```
stop <= select ^mask;
```

es decir, usar:

```
stop = select ^mask;
```

Compilar de nuevo el código y comparar el resultado del visor RTL viewer en ambos casos. ¿cuál es la diferencia después de la síntesis en ambos circuitos? Fijarse bien, la diferencia es muy pequeña pero importante. ¿De donde se toma la señal select en la tercera sentencia del código de acuerdo al operador usado?

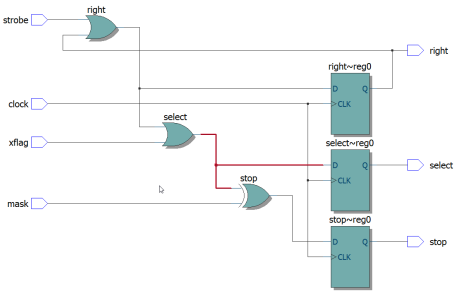

Figure 7: *Descripción tres del circuito*

La señal select que se asigna como entrada a la compuerta xor es tomada directamente de la salida de la compuerta or.