Pre-Informe 3

Fabián Levicán Santibáñez
201603012-3

Pregunta 3.1

a) Un lenguaje de descripción de hardware o HDL (en inglés, hardware description language) es un lenguaje de programación que permite modelar sistemas electrónicos y electrónico-digitales. Permite describir formalmente la operación, estructura y diseño de un circuito, y ayuda a simularlo y a analizarlo de manera automática.
Si bien un HDL es similar a un lenguaje de programación concurrente, su principal diferencia con los lenguajes de programación usuales es su inclusión explícita del concepto de tiempo.

b) Una nota previa: *logic* fue creado para reemplazar a *reg* como un tipo de variable más general, debido a las confusiones que generaba el uso de estos tres tipos de variable en versiones más antiguas de *Verilog*.
reg se utiliza para registros. Son variables cuyo valor se quiere guardar. En general, estas variables se utilizan en el lado izquierdo de asignaciones dentro de bloques *always*. Se le suelen asignar valores al ocurrir un evento como subir el clk.
wire es un tipo de dato de "red". Se utiliza para variables que representan conexiones físicas. En general, estas variables se utilizan en asignaciones *assign* fuera de los bloques *always*. Se le suelen asignar valores continuamente.
logic es el tipo de dato más general, y así se puede utilizar como registro o como tipo de dato de "red". Su uso principal es como el tipo de dato de las variables de entrada (*input*) y de salida (*output*).

c) *assign* se utiliza para asignar continuamente la expresión del lado derecho (del símbolo =) al *wire* del lado izquierdo. Se utiliza fuera de un *always*.
<= se utiliza para asignación no bloqueante. Esto significa que todas las asignaciones de este tipo se ejecutan en paralelo. Se utiliza dentro de un *always* o un *initial*.
= se utiliza para asignación bloqueante. Esto significa que se fuerza un orden secuencial. Se utiliza dentro de un *always* o un *initial*.

d) *reg* significa que se está declarando un registro, $[15:0]$ significa que el registro es de 16 bits indizados del 15 (más significativo) al 0 (menos significativo), g es el identificador del registro, = indica que el registro se está declarando con el valor inicial del lado derecho, y $h'A6B2$ es un número en hexadecimal (h') de tamaño indefinido (se puede indicar el número de bits antes del h'). El equivalente binario del valor definido es 1010_0110_1011_0010.

e) Antes de subir el *clk*, *first* se declara con el valor 0, y a *second* se asigna el valor 1. Una vez que sube el *clk*, ninguna señal que cambie dentro del *always* puede salir de este bloque hasta que se llegue a su instrucción *end*. Por lo tanto, primero *first* toma el valor 1 (asignación bloqueante), pero *second* no toma inmediatamente el valor 0. Luego, el bit 1 de *arr* toma el valor del bit 1 de *arr* negado, o sea, 1, y así *arr* toma el valor binario indicado.

Pregunta 3.2

Utilicé $r = 101$.

En la Subida clk no. 1 se entra al if, pues s es 0, y z toma el valor de r. Luego, el valor de s cambia de 0 a 1 lo que implica que no se vuelve a entrar al if, ni a usar r. El valor final de n está destacado en negrita.

Estados iniciales:
$y_0 = 01110111$
$x_0 = 00000000$
$z_0 = 000$

$n_0 = 01110111$
$m_0 = 001$

Subida clk no. 1

$z = 101$
$\neg y[m] = 0, \neg y[z] = 0$
y nuevo $= 01010111$
x nuevo $= 00000000$
z nuevo $= 110$

n nuevo $= 01010111$
m nuevo $= 111$

Subida clk no. 2

$\neg y[m] = 1, \neg y[z] = 0$
y nuevo $= 01010111$
x nuevo $= 00000000$
z nuevo $= 111$

n nuevo $= 01010111$
m nuevo $= 000$

Subida clk no. 3

$\neg y[m] = 0, \neg y[z] = 1$
y nuevo $= 01010111$
x nuevo $= 00000001$
z nuevo $= 000$

n nuevo $= 01011000$
m nuevo $= 001$

Subida clk no. 4

$\neg y[m] = 0, \neg y[z] = 0$
y nuevo $= 01010110$
x nuevo $= 00000001$
z nuevo $= 001$

n nuevo $= 01010111$
m nuevo $= 010$

Subida clk no. 5

$\neg y[m] = 0, \neg y[z] = 0$
y nuevo $= 01010100$
x nuevo $= 00000001$
z nuevo $= 010$

n nuevo $= 01010101$
m nuevo $= 011$

Subida clk no. 6

$\neg y[m] = 1, \neg y[z] = 0$
y nuevo $= 01010100$
x nuevo $= 00000001$
z nuevo $= 011$

n nuevo $= 01010101$
m nuevo $= 100$

Subida clk no. 7

$\neg y[m] = 0, \neg y[z] = 1$
y nuevo $= 01010100$
x nuevo $= 00010001$
z nuevo $= 100$

n nuevo $= 01100101$
m nuevo $= 101$

Subida clk no. 8

$\neg y[m] = 1, \neg y[z] = 0$
y nuevo $= 01010100$
x nuevo $= 00010001$
z nuevo $= 101$

n nuevo $= 01100101$
m nuevo $= 110$

Subida clk no. 9

$\neg y[m] = 0, \neg y[z] = 1$
y nuevo $= 01010100$
x nuevo $= 01010001$
z nuevo $= 110$

n nuevo $= 10100101$
m nuevo $= 111$

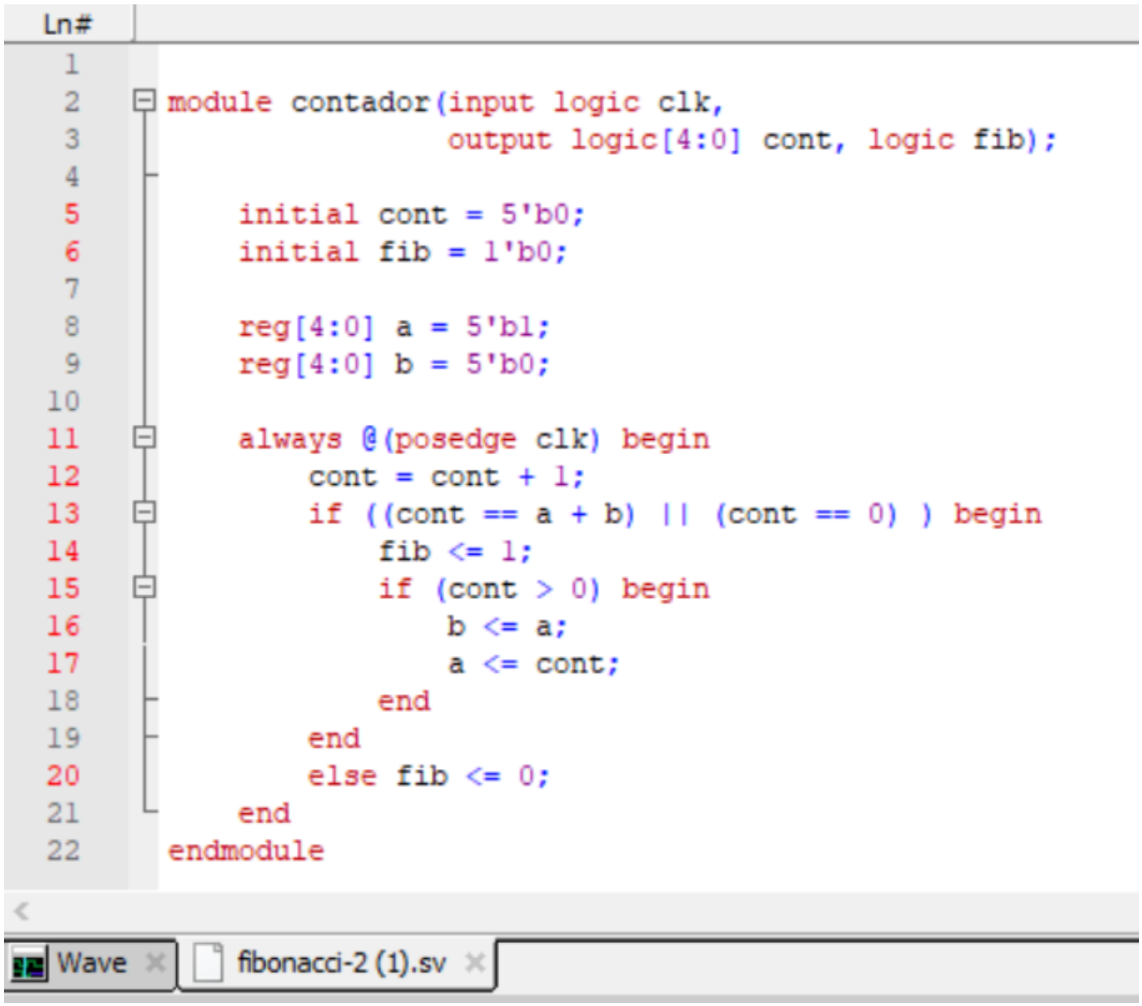
Subida clk no. 10

$\neg y[m] = 1, \neg y[z] = 0$
y nuevo $= 01010100$
x nuevo $= 01010001$
z nuevo $= 111$

n nuevo = 10100101
m nuevo $= 000$

Pregunta 3.3

A continuación, primero la captura de pantalla del código, y luego la captura de pantalla del estado final de la simulación.



```
Ln#
 1
 2  module contador(input logic clk,
 3                  output logic[4:0] cont, logic fib);
 4
 5      initial cont = 5'b0;
 6      initial fib = 1'b0;
 7
 8      reg[4:0] a = 5'b1;
 9      reg[4:0] b = 5'b0;
10
11      always @(posedge clk) begin
12          cont = cont + 1;
13          if ((cont == a + b) || (cont == 0) ) begin
14              fib <= 1;
15              if (cont > 0) begin
16                  b <= a;
17                  a <= cont;
18              end
19          end
20          else fib <= 0;
21      end
22  endmodule
```

Wave | fibonacci-2 (1).sv

Figure 1: Código.

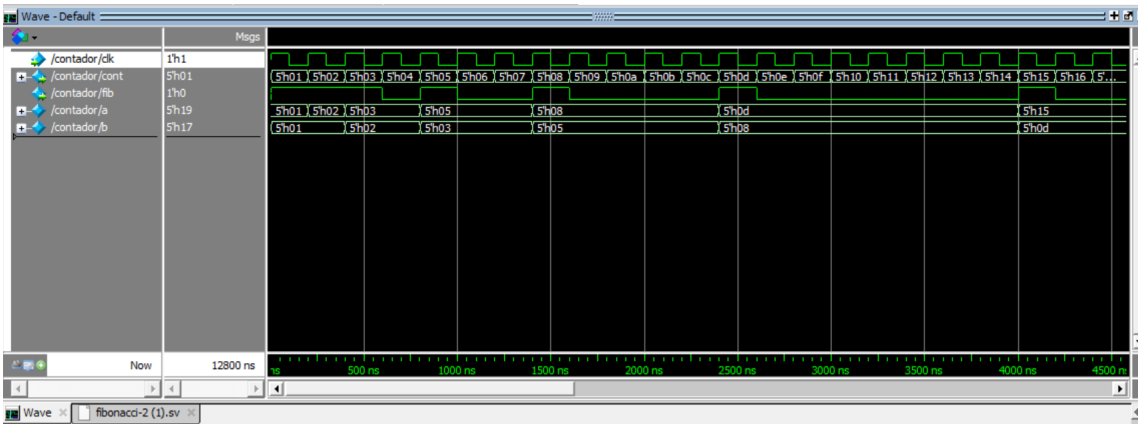


Figure 2: Estado final de la simulación.

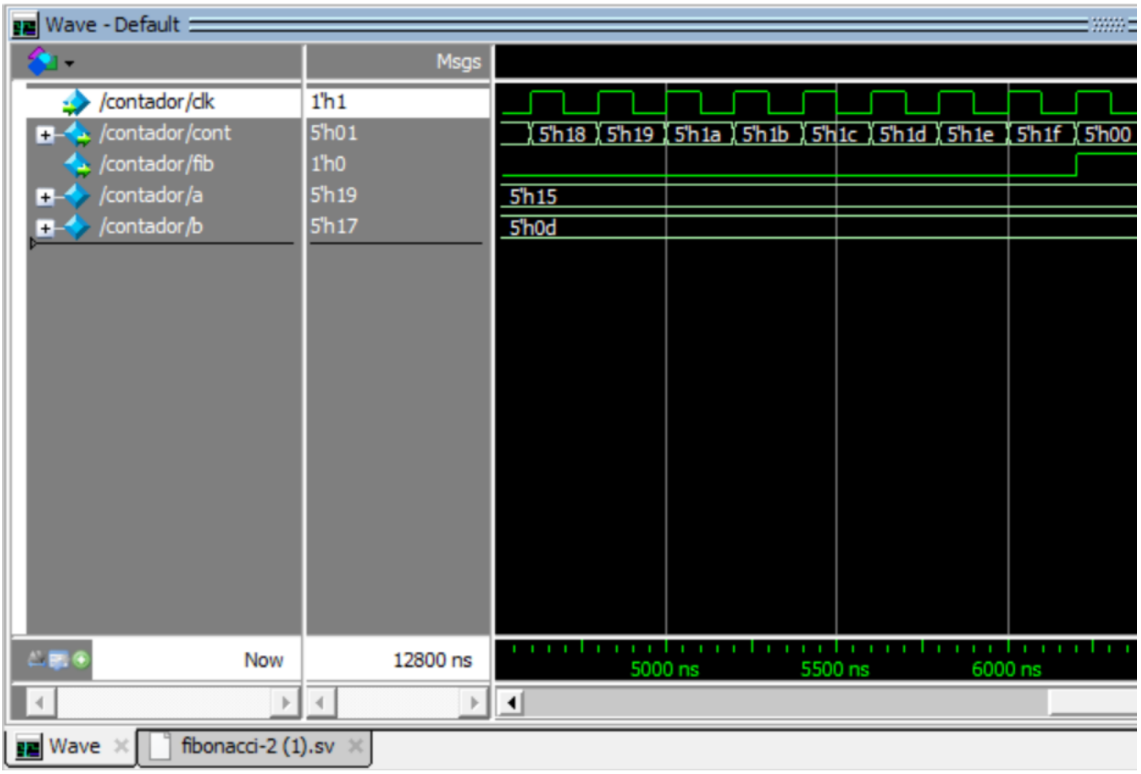


Figure 3: Estado final de la simulación.