Pre-Informe 3
Laboratorio de Integración Tecnológica

Carlos Daniel Abello Allende
201773052-8

Pregunta 1

a) Son lenguajes especializados que se utilizan para definir la estructura, diseño y operación de circuitos electrónicos y circuitos electrónicos digitales, permitiendo hacer una descripción formal de un circuito, posibilitando su análisis y simulación. A diferencia de un lenguaje de programación, la mayoría de las expresiones se ejecutan simultaneamente, y cada una corresponde a la operación de un bloque de circuito.

b) "wire" se utiliza para conectar elementos distintos y no tienen memoria, por lo que no guardan un valor, por lo que deben ser asignados de forma continua
"reg" representa un valor asignado que guarda hasta hasta que se le asigne un nuevo valor.
Tanto "wire" como "reg" son tipos de datos de 4 estados (1,0,x,z).
"logic" hace referencia a un tipo de dato genérico de 4 estados, pudiendo referirse a tanto a un "wire" como a un "reg".

c) "assign" es una asignación que sea realiza de forma continua y ocurre en paralelo con el resto de las operaciones.
"<=" se utiliza en lógica secuencial, se ejecutan en simultáneo y son leídos antes de que comience la operación.
"=" se utiliza en lógica combinacional, son leídos secuencialmente y siempre son ejecutados antes que el código que lo sigue.

d) "reg" hace referencia al tipo de dato.
"[15:0]" hace referencial al número de bits utilizados, en este caso son 16 (del 0 al 15).
"g" es el nombre de la variable.
"=" es el operador de asignación.
La última parte es el valor asignado, donde el primer número hace referencia al número de bits que puede utilizar, en este caso son 16 bits. El primer caracter luego del "'" identifica en que base está representado el número, en este caso la letra "h" identifica que es un número hexadecimal. El resto representa el número en sí, en este caso A6B2, que representado como número decimal, equivale a 42674, y en binario es 1010011010110010.

e) Las asignaciones dentro del always_ff influyen porque hay una asignación "=", lo que evita que se ejecute antes la asignación "<=", por lo que el valor de "first" con que se ejecuta es el valor actualizado "1" y no el valor inicialmente asignado "0", y por lo tanto se modifica el segundo bit de "arr" y no el primero.

Pregunta 2

Se define el valor de entrada $r = 0$.

Valores iniciales: $x = 00000000$ $y = 01110111$ $z = 000$ $n = 01110111$ $m = 001$

clk 1: $x = 00000000$ $y = 01110110$ $z = 001$ $n = 01110110$ $m = 010$

clk 2: $x = 00000000$ $y = 01110100$ $z = 010$ $n = 01110100$ $m = 011$

clk 3: $x = 00000000$ $y = 01110100$ $z = 011$ $n = 01110100$ $m = 100$

clk 4: $x = 00010000$ $y = 01110100$ $z = 100$ $n = 10000100$ $m = 101$

clk 5: $x = 00010000$ $y = 01100100$ $z = 101$ $n = 01110100$ $m = 110$

clk 6: $x = 00010000$ $y = 01000100$ $z = 110$ $n = 01010100$ $m = 111$

clk 7: $x = 00010000$ $y = 01000100$ $z = 111$ $n = 01010100$ $m = 000$

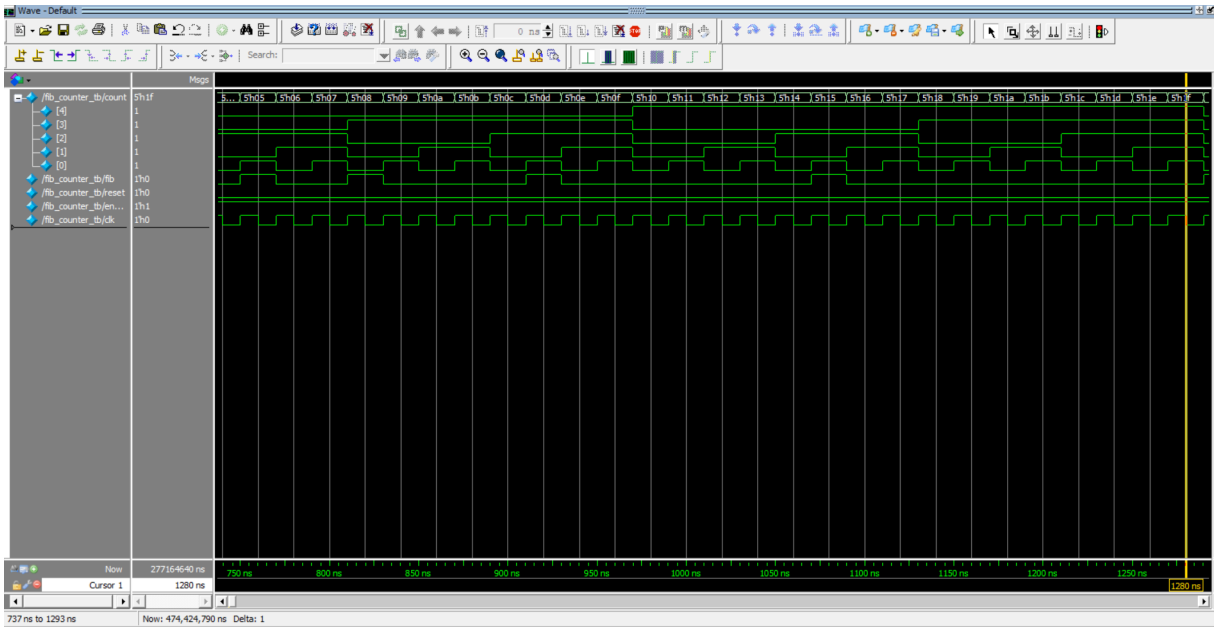
clk 8: $x = 00010001$ $y = 11000100$ $z = 000$ $n = 11010101$ $m = 001$

clk 9: $x = 00010011$ $y = 11000101$ $z = 001$ $n = 11011000$ $m = 010$

clk 10: $x = 00010111$ $y = 11000101$ $z = 010$ $n = 11011100$ $m = 011$

Por lo tanto el valor final del output "n" es 11011100.

Pregunta 3



Módulo desarrollado:

```
module fib_counter(
input wire clk,
input wire reset,
input wire enable,
output logic fib,
output logic [4 : 0] count
);
always_ff @(posedgeclk)
if (reset)
begin
count <= 5'b00000;
fib <= 1'b1;
end
else if (enable)
begin
count <= count + 1;
if (count == 5'b11111|count + 1 == 5'b00001|count + 1 == 5'b00010|count + 1 == 5'b00011
|count + 1 == 5'b00101|count + 1 == 5'b01000|count + 1 == 5'b01101|count + 1 == 5'b10101)
begin
fib <= 1'b1;
end
else
begin
fib <= 1'b0;
end
end
endmodule
```

Probado usando el testbench:

```
module fib_counter_tb;
logic [4 : 0] count;
logic fib;
logic reset, enable, clk;
fib_counter U0(
.count (count),
.reset (reset),
.fib (fib),
.enable (enable),
.clk (clk)
);
initial begin
clk = 0;
reset = 1;
enable = 0;
end
initial begin
#15 reset = 0;
enable = 1;
end
always
begin
#10 clk =~ clk;
end
endmodule
```