Pre-Informe 3

Florencia Valladares
201773007-2

Pregunta 1

a. Un HDL corresponde a un lenguaje de computación especializado, utilizado para describir la estructura y comportamiento de circuitos electrónicos y digitales. Son muy similares a un lenguaje de programación como C, sin embargo la mayor diferencia es que el HDL incluye explícitamente la noción del tiempo.

b. El tipo de variable reg representa variables con capacidad de almacenar información, mientras que wire representa conexiones estructurales entre componentes y no tienen capacidad de almacenamiento. Por otra parte, logic fue introducido para evitar confusiones entre wire y reg, ya que es capaz de interpretar el comportamiento que debe tener la variable según su uso.

c. Existen dos tipos de asignaciones: continua y procedural. La asignación continua utiliza assign en su sintaxis, y se utiliza exclusivamente para modelar lógica combinacional, además la variable a la que se realiza la asignación sólo puede ser declarada de tipo wire. En la asignación procedural la variable a la que se le asigna el valor puede ser de cualquier tipo, además existen dos tipo: con y sin bloqueo. El operador = se utiliza para asignaciones con bloqueo, esto significa que se ejecuta antes de proceder con la siguiente línea, en un módulo always significa que una linea se ejecuta después de otra, como un ciclo while cualquiera . El operador <=, en cambio, corresponde a una asignación sin bloqueo, se suelen ocupar dentro de módulos always y las lineas con este tipo de asignaciones se ejecutan paralelamente.

d. Corresponde a una declaración y asignación. Primero se declara una variable de tipo reg, llamada "g", que ocupa 16 bits de almacenamiento (por [15:0]). Luego, se le asigna el valor "16'hA6B2" que corresponde a un número hexadecimal dada la notacion " 'h ", de 16 bits pues es "16'h" y cuyo valor numérico es "A6B2", es decir, 1010011010110010 en binario.

e. Las asignacion de "first=1" dentro del módulo always es con bloqueo, por lo que se evalua y aplica antes de pasar a la siguiente linea, por lo que "arr[first]" será "arr[1]". Los assign se actualizan al final de cada ciclo "always", por lo que se produce un delay al actualizar el valor de second y pese a haber cambiado first a 1, la operación "arr[first] = (∼(arr[second]))" se ejecuta como "arr[1] = (∼(arr[1])), devolviendo así el binario "011".

Pregunta 2

Sea $r = 000$, la siguiente tabla represante los valores de cada variable (si es que cambia) después de cada subida del clk:

(r=000)	Variables					
Tras subida n°	x	y	z	n	m	s
0	00000000	01110111	000	01110111	001	0
1	00000000	01110110	001	01110110	010	1
2	00000000	01110100	010	01110100	011	
3	00000000	01110100	011	01110100	100	
4	00010000	01110100	100	10000100	101	
5	00010000	01100100	101	01110100	110	
6	00010000	01000100	110	01010100	111	
7	00010000	01000100	111	01010100	000	
8	00010001	11000100	000	11010101	001	
9	00010011	11000101	001	11011000	010	
10	00010111	11000101	010	11011100	011	

Pregunta 3

Código contador de 5 bits, donde out el el número de 5 bits y b el booleano que indica si el número pertenece a la serie de Fibonacci o no (como son números de 5 bits, aquellos que pertencen a Fibonacci son: 0, 1, 2, 3, 5, 8, 13 y 21).

```
module counter ( out, clk);
//----------Output--------------
    output [4:0] out;
//----------Input--------------
    input  clk;
//----------Internal Variables--------
    reg [4:0] out = 5'b0;
    reg b = 1'b1;

always @(posedge clk) begin
  out = out + 1;
  if ((out == 5'b1)||(out == 5'b00010)||(out == 5'b00011)||(out == 5'b00101)||(out == 5'b01000)||
      (out == 5'b01101)||(out == 5'b10101))
    b <= 1;
  else
    b <= 0;
end
endmodule
```

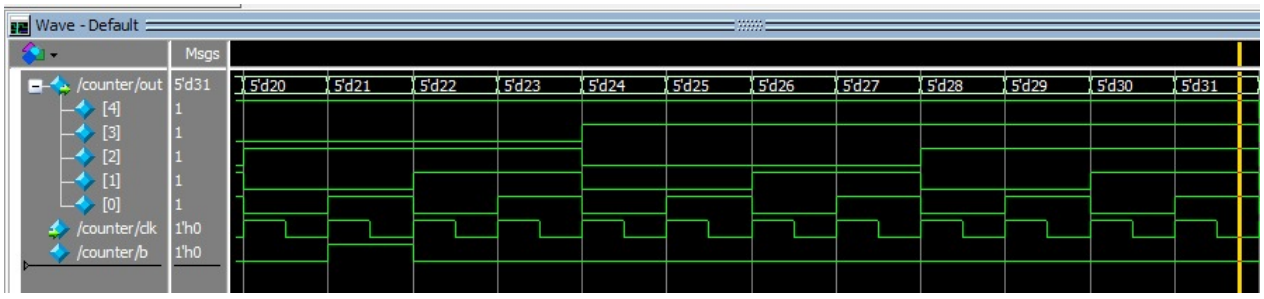

Figure 1: Captura de pantalla tras haber llegado a 31, máximo número de 5 bits.