Pre-Informe Experiencia 3

Daniel Torres Guzmán
201604572-4

Investigacion de conceptos

a) Es un lenguaje de programación especializado que se utiliza para definir la estructura, diseño y operación de circuitos electrónicos, y más comúnmente, de circuitos electrónicos digitales. Así, los lenguajes de descripción de hardware hacen posible una descripción formal de un circuito electrónico, y posibilitan su análisis automático y su simulación.
Los lenguajes de descripción de hardware se parecen mucho a otros lenguajes de programación de ordenadores, pero la diferencia radica en que el HDL incluye explícitamente la noción de tiempo.

b) Los wires son usados para conectar diferentes elementos, pueden ser tratados como cables físicos, en cambio las variables reg representa elementos de información almacenada en Verilog, a diferencia de los wires, los cuales no pueden almacenar información.
Dado que reg solo puede correr en bloque procedurales tales como initial o allways mientras que wire solo puede ser usada en sentencias de tipo assign, SystemVerilog introdujo un nuevo tipo de dato de 4 estados llamado logic, el cual se puede usar tanto en bloques procedurales como en sentencias del tipo assign

c) <= indica que la asignación no tiene bloqueo, esto significa que todos se ejecutan en paralelo y no hay seguridad sobre el orden.
= indica bloqueo, cada uno se ejecutara en orden secuencial.
assign indica asignación continua, LHS (Left Hand Side) cambiara si cambia el RHS (Right Hand Side).
Cabe destacar que <= y = debes estar siempre dentro de un bloque de tipo initial o allways.

d) Se almacena una variable de tipo reg de nombre g de 16 bits de tamaño con un valor de tipo hexadecimal A6B2 el cual en binario es 1010011010110010.

e) Dado que luego de la asignacion de bloqueo que se le da a first viene una asignacion de no bloqueo, assign se ejecuta en paralelo con la sentencia no bloqueante dentro de always , por lo tanto first si cambia pero second conserva su valor generando el output 0111.

Análisis de codigo

clk	r	m	n	x	y	z	s
0	000						
				00000000			
					01110111		
						000	
			01110111				
		001					
							0
pedge						000	1
		010	01110110	00000000	01110110	001	
pedge		011	01110100	00000000	01110100	010	
pedge		100	01110100	00000000	01110100	011	
pedge		101	10000100	00010000	01110100	100	
pedge		110	01110100	00010000	01100100	101	
pedge		111	01010100	00010000	01000100	110	
pedge		000	01010100	00010000	01000100	111	
pedge		001	11010101	00010001	11000100	000	
pedge		010	11011000	00010011	11000101	001	
pedge		011	11011100	00010111	11000101	010	

Diseño de modulo

Figure 1: Simulación

```
module Fibo (input logic clk,
                output logic [4:0]n,
                output logic p);
        reg [4:0]x = 5'b0;
        assign p = (~x[4])&(~x[3])&(~x[2])&x[0] |
                (~x[4])&(~x[3])&(~x[2])&x[1] |
                (~x[4]) & x[2] & (~x[1]) & x[0] |
                (~x[3])& x[2] & (~x[1]) & x[0] |
                (~x[4]) & x[3] & (~x[2]) & (~x[1]) & (~x[0]);
        assign n = x;
        always_ff@(posedge clk) begin
                x = x+1;
        end
endmodule
```

Figure 2: Codigo