Pre-Informe 3
INF-245: Arquitectura y Organización de Computadores

Felipe Alberto Narea Gago
201773019-6

Pregunta 1

a) ¿Qué es un lenguaje de descripción de hardware? ¿Qué lo diferencia de un lenguaje de programación?

HDL, (su sigla en inglés), es un lenguaje de programación especializado que se utiliza para definir la estructura, diseño y operación de circuitos electrónicos y electrónicos digitales. Sirve para simular la operación del circuito descrito y asegurar la correcta operación de esta antes de su fabricación.
La diferencia entre HDL y otros lenguajes de programación es que el HDL incluye explícitamente la noción del tiempo (una notación específica), debido a que este es una característica fundamental en los circuitos electrónicos reales.

b) Explique la funcionalidad de los tipos de variable *wire*, *reg* y *logic*, y cuales son sus diferencias.

Wire o en español, "*cable*", es un tipo de variable que conecta a dos componentes, pueden ser usados como entradas o salidas de los componentes, es importante mencionar que no almacenan datos. Suele ser usado solo en modelado combinacional.
Reg es el tipo de variable que puede almacenar valores y la intensidad de esta señal. se utilizan para modelar el almacenamiento de datos.
Los tipos de variable Logic son un tipo de variable que se puede usar como las dos anteriores, ya sea almacenando información o traspasándola entre componentes. Las diferencias están en la capacidad de almacenar datos, wire solo transmite la información, mientras que reg la guarda y la mantiene, logic puede hacer ambas.

c)
- "*assign*": es una asignación continua que cambia el parámetro izquierdo cuando el de la derecha cambia. Es usada para asignar un valor a un wire dentro de un módulo, esta debe realizarse fuera de un procedimiento, nunca en bloques always o initial.
- "=": asignación de bloqueo, significa que la asignación se realiza antes de proceder con la siguiente, se utiliza para la asignación dentro de bloques always o initial. En cuestion de tiempos su asignacion es en serie.
- "<=": asignación no bloqueante, El termino derecho se evalúa en el instante actual, pero no se asigna al derecho hasta finalizar dicho instante, se utiliza para la asignación dentro de bloques always o initial.

d) r e g [1 5 : 0] g = h ' A6B2 ;
reg: Representan variables con capacidad de almacenar información.
[15 : 0] : Registro de 16 bits, donde data[0] es el bit menos significativo
g = es el nombre de la variable
h ' A6B2 = h =base hexadecimal
a6b2 es el valor que toma la base (en binario: 01000001 00110110 01000010 00110010)

e) En el caso es debido a que first al ser asignado como bloqueante, es asignado en orden de instrucción, por lo que first pasa primero a tener el valor de 1, luego de eso se utiliza en el arr[ ], para ubicar la posición [1], si esto no fuese así, y hubiese sido no bloqueante, el valor que arr hubiese recibido seria [0], pues la asignación es en serie y los valores hubiesen cambiado al "mismo tiempo".

Pregunta 2

El ruteo es:

N	clk	m	n	x	y	z
0		001	01110111	00000000	01110111	000
1	1					
						000
				00000000	01110110	
		010	01110110			001
2	1					
				00000000	01110100	
		011	01110100			010
3	1					
				00000000	01110100	
		100	01110100			011
4	1					
				00010000	01110100	
		101	10000100			100
5	1					
				00010000	01100100	
		110	01110100			101
6	1					
				00010000	01000100	
		111	01010100			110
7	1					
				00010000	01000100	
		000	01010100			111
8	1					
				00010001	11000100	
		001	11010101			000
9	1					
				00010011	11000101	
		010	11011000			001
10	1					
				00010111	11000101	
		011	11011100			010

Figure 1: Con r = b’ 0

Pregunta 3

```
module counter(
    input logic clk, reset,
    output reg[4:0] count,
    output logic fib);
initial count = 31;
assign A = count[4];
assign B = count[3];
assign C = count[2];
assign D = count[1];
assign E = count[0];

always@(posedge clk, posedge reset) begin
    if(reset) begin
count <= 0;
end
    count = count + 1;
    if(((~A)&&(~B)&&(~C))||
((~A)&&(~B)&&(~D)&&E)||
((~A)&&(~C)&&(~D)&&(~E))||
((~A)&&(C)&&(~D)&&(E))||
((~B)&&(C)&&(~D)&&(E)))
fib = 1;
    else
fib = 0;
    end
endmodule
```