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

Guillermo Meza
201773034-k

Pregunta 1: Investigación de conceptos

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

R) Un lenguaje de descripción de hardware (LDH) es un lenguaje de programación especializado en el diseño, estructura y operación de circuitos electrónicos, haciendo posible la descripción de un circuito electrónico y su simulación. La principal diferencia entre LDH y un lenguaje de programación (LP) es que LDH se usa para describir el comportamiento de los sistemas electrónicos de manera digital, mientras que un LP se usa para dar instrucciones a la CPU para que realice una tarea específica.

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

R) Estas variables se definen como:

- wire: son variables que conectan los componentes de un circuito, no pueden almacenar información.
- reg: son variables de tipo registro que almacenan información.
- logic: son un tipo de variable que se puede usar como las dos anteriores, ya sea almacenando informacion o traspasandola entre componentes.
  Las diferencias estan en la capacidad de almacenar datos, wire solo recibe y entrega, mientras que reg los guarda y los mantiene, logic puede hacer ambas.

c) ¿Cuál es la diferencia entre los operadores de asignación *assign*, <= y =, y en que secciones del modulo se utilizan?

R) Las asignaciones que se pueden usar son:

- "<=" es una asignación que no bloquea el circuito, esta se ejecuta en paralelo cuando el reloj sube, y por lo mismo no asegura un orden en el circuito.
- "=" es una asignación de bloqueo, esta detiene el circuito y asegura el cambio en el valor para luego realizar las demás operaciones.
- "*assign*" es una asignacion continua que cambia el parametro izquierdo cuando el de la derecha cambia.

Tanto el "<=" como el "=" se usan dentro de un bloque `always` o `initial`.

d) Dada la siguiente expresión:

```
reg [15:0]g = h'A6B2;
```

Explique su estructura, que significa cada una de sus partes, y cual es el equivalente binario del valor definido.
R) Separando por partes la linea tenemos:

- reg es el tipo de variable, en este caso de registro, que almacena los datos ingresados.

15:0 es la cantidad de bits que se le están asignando a la variable en este caso son 16.

- g es el nombre de la variable.
- = es el operador de asignación de bloqueo.
- h' es la base del numero que se esta ingresando, h es hexadecimal, b binario, d decimal, etc.
- A6B2 es el numero en la base dada anteriormente.

El equivalente binario de A6B2 es 1010 0110 1011 0010, este es el numero que se almacenará en los 16 bits que se le asignaron a la variable.

e) Si consideramos el siguiente modulo:

```
module Yaled(input logic clk);
    reg[3:0] arr = 4'b0101;
    reg first = 1'b0;
    wire second;
    assign second = (~(first));
    always_ff@(posedge clk) begin
        first = 1
        arr[first] <= (~(arr[second]));
        end
    endmodule
```

Después de una subida del "clk", el registro "arr" resulta tener el valor binario 0111. Explique porque el delay y las asignaciones dentro del bloque `always_ff` influyen en este resultado.

R) El first tiene una asignacion de bloqueo, que obliga a cambiar esta variable, dado que esta operacion se ejecutan antes que el arreglo cambie de valor, por otro lado second no varia hasta que termine el ciclo del always, por lo que mantiene el valor inicial que era 1, que nos da el arr[1]=0, que será el mismo valor que se cambiará a uno por la negacion, osea, arr[1] ¡= arr[1] lo que nos cambia de 0101 a 0111.

Pregunta 2: Análisis de código

Realice un ruteo del valor del output "n" después de 10 subidas del CLK. Defina usted el valor de entrada de "r", y explicítelo junto al ruteo.

R) Sea r = 000, tenemos los siguientes valores de la función solicitada. Siendo n en la decima subida del reloj n=11011100.

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

Pregunta 3: Diseño de Módulo

Escriba un módulo en System Verilog para la función siguiente: Un contador de 5 bits que tenga como outputs la cuenta actual, y un booleano que indique con 1 si es que el numero actual se encuentra en la serie de Fibonacci, o con un 0 si no.

R) Se creo un contador simple usando always y aprovechando los cantos de subida del reloj, para implementar la serie de Fibonacci se creo un mapa de Karnaugh que finalmente se aplicó al codigo, se le asignó a los 5 bits del contador los nombre de A,B,C,D,E con los que se hizo la ecuacion para hacer fib 1 o 0 dependiendo el numero.

```
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
```