INF-245 Experiencia 3

Kevin Reyes
201773091-9

1 Investigación de Conceptos

a.

Es un lenguaje utilizado para describir el comportamiento de circuitos digitales. Utiliza programación de estructuras mediante el uso de: expresiones, declaraciones, estructuras de control; con las cuales se busca simular el comportamiento de circuitos digitales sin la necesidad de recurrir a la estructuración física de estos. Un lenguaje de programación no describe el comportamiento de circuitos digitales, sino que provee un conjunto de instrucciones al procesador para realizar una tarea específica.

b.

- ***wire*:** Esta variable describe una 'compuerta' en un cable que está continuamente escuchando a cualquier cambio en los datos de entrada(otros cables, compuertas lógicas, etc...), por lo que se le llama variable de asignación continua.
- ***reg*:** Comunmente describe un espacio de almacenamiento, el cual sostiene el valor actual hasta que otro dato sea ingresado y el CLK este en canto de subida. Utiliza la asignación de procedimiento.
- ***logic*:** Puede describir tanto asignación continua, como de procedimiento dependiendo del contexto.

c.

- ***'assign'*:** Este operador es del tipo asignación continua, es decir si el lado derecho del operador cambia, entonces cambiará automáticamente el lado izquierdo del operador. Se utiliza fue de declaraciones always.
- **<=:** Este operador es del tipo non-blocking, es decir, pueden ser ejecutados varios en paralelo. Es operado en cada canto de subida del CLK. Se utiliza dentro de declaraciones always.
- ***'='*:** Este operador es del tipo blocking, es decir, bloquea la ejecución de lineas siguientes hasta que sea ejecutada su operación por completo. Ejecuta código secuencialmente dentro de bloques begin/end.

d.

- ***'reg'*:** Tipo de variable.
- ***'[15:0]'*:** Tamaño en bit de la variable, con índice del bit más significativo al menos significativo.
- ***'g'*:** Nombre de la variable.
- ***'='*:** Operador.
- ***'h'*:** Sistema numérico de la variable, en este caso, hexadecimal.
- ***'A6B2'*:** Número en hexadecimal.
- ***';'*:** Necesario para que compilador identifique fin de un statement.
- ***En binario*:** reg [15:0] g = 16'b1010_0110_1011_0010;

e.

Ya que always_ff es un bloque, las asignaciones continuas fuera de este no tendrán efecto hasta que el bloque haya sido ejecutado por completo. Por lo que, el valor almacenado en first cambiará dentro del bloque, pero para que este cambio sea notado por second fuera del bloque es necesario que este sea ejecutado antes, por esto, en el segundo statement del bloque, first y second tienen los mismos valores.

2 Análisis de código

A continuación el ruteo del valor del output 'n' para el valor de entrada $r = 000$:

n	**x**	**y**	**z**	**m**	**s**	**Posedge CLK**
119-01110111	**0-00000000**	**119-01110111**	**0-000**	**1-001**	**0-000**	
	0-00000000	**118-01110110**	**0**		**1**	**Primer C.Sub**
			1-001			
118-01110110				**2-010**		
	0-00000000	**116-01110100**				**Segundo C.Sub**
			2-010			
116-01110100				**3-011**		
	0-00000000	**116-01110100**				**Tercer C.Sub**
			3-011			
116-01110100				**4-100**		
	16-00010000	**116-01110100**				**Cuarto C.Sub**
			4-100			
132-10000100				**5-101**		
	16-00010000	**100-01100100**				**Quinto C.Sub**
			5-101			
116-01110100				**6-110**		
	16-00010000	**68-01000100**				**Sexto C.Sub**
			6-110			
84-1010100				**7-111**		
	16-00010000	**68-01000100**				**Séptimo C.Sub**
			7-111			
84-1010100				**0-000**		
	17-00010001	**196-11000100**				**Octavo C.Sub**
			0-000			
213-11010101				**1-001**		
	19-00010011	**197-11000101**				**Noveno C.Sub**
			1-001			
216-11011000				**2-010**		
	23-00010111	**197-11000101**				**Décimo C.Sub**
			2-010			
220-11011100				**3-011**		

Se obtiene un valor para n en sistema numérico decimal de 220 y en sistema numérico binario de 1101_1100.

3 Diseño de módulo

```
module up_counter // Definimos nuestros módulo contador
(
    output logic [4:0] out, // Output para valor actual del contador
    output logic fibonacci, // Output para valor de verdad de: 'El número actual es fibonacci'
    input logic enable, // Encender el contador
    input logic clk, // Reloj para sincronizar operaciones
    input logic reset // Para generar una simulación permanente
    );
    int f[7]= '{1,2,3,5,8,13,21}; // Arreglo con números de fibonacci dentro del rango de 5 bits
    always_ff @(posedge clk)
        if (reset)
        begin
            out <= 5'b0; // Se inicializa el contador
        end
        else if (enable)
        begin
            out = out +1; // Se opera el contador
            // Se revisa si el output está dentro del arreglo de números de fibonacci
            fibonacci <= (f[0]==out)? 1:((f[1]==out)?1:((f[2]==out)?1:((f[3]==out)?1:((f[4]==out)?1:((f
        end
endmodule
```

A continuación el testbench usado para la simulación:

```
`timescale 1ns/1ps
module up_counter_tb; //definimos nuestro testbench para modulo contador
    logic [4:0] out;
    logic fibonacci;
    logic enable, clk, reset;
    up_counter U8(
    .out (out),
    .fibonacci (fibonacci),
    .enable (enable),
    .clk (clk),
    .reset (reset)
    );
    // Generamos alternación entre reset y enable para así generar una simulación continua
    initial begin
        clk=0;
        reset=1;
        enable=0;
    end
    initial begin
        #15 reset=0;
        enable=1;
    end
    always
    begin
        #10 clk=~clk;
    end
    integer file_out;
    initial begin
```

```
        file_out = $fopen("up_counter.txt","wb");
        $fstrobe(file_out, "time\tclk\treset\tenable\tout\tfibonacci");
    end
    always @(posedge clk)
    begin
        $fstrobe(file_out,"%3d\t%b\t%b\t%b\t%d",$time,clk,reset,enable,out,fibonacci);
    end
endmodule
```

A continuación las imagenes de la simulación en ModelSim: