Pre-Informe 3

Carlos Rojas
201773013-7

Investigación de Conceptos

a) Un lenguaje de descripcion de hardware esun lenguaje usado para describir y modelar circuitos, la principal diferencia es este no se programan los circuitos, sino que se modelan los circuitos y su comportamiento , mientras que en un lenguaje de programación se crean programas y sus ejecutables,además los HDL pueden ejecutar distintas tareas paralelamente para así dar mejores tiempos de respuesta.

b)
- Wire: Este es un tipo de variable que representa conexiones estructurales entre componentes de nuestro circuito, de ahi su nombre,además no poseen capacidad para almacenar informacion,cabe destacar que este debe ser usado fuera de un bloque always.
- Reg: Este es un tipo de variable que representan aquellas variables de registro que son capaces de guardar informacion, cabe destacar que este puede ser usado dentro de bloques always.
- Logic:Este tipo de variable que, a diferencia de las anteriores, se encuentra en Systemverilog solamente y sirve para declarar cualquier tipo de variables (puede ser visto como una mejor al wire y al reg), pero este al definir muchos tipos de variable se le deben escribir el comportamiento que tendra esa variable, para lo que usamos los difentes tipos de asignaciones que existen tal como assign, <= o =.

c) Las principales diferencias entre <=, assign y = es que mientras que el <= siempre se usara dentro de bloques y no se puede garantizar un orden de asignacion (si hay mas de uno) ya que realizara todas las asignaciones en paralelo una de otro , el = nos garantiza un orden y puede ser usado tanto dentro de bloques como fuera de ellos y por ultimo assign se usa para realizar asignaciones continuas, definen logica, esto quiere decir que al cambiar una variable que este relacionada a una variable que fue asignada con assign , la segunda variable variara de igual forma,además debido a lo anterior assign no puede ser usado dentro de ningun bloque.

d) Se tiene que reg sirve para declarar la variable y decir que este es de tipo registro (puede almacenar informacion), luego definimos el tamaño de este en [15:0] y g es la variable a declarar y definir.Luego esta el = que sirve para definir nuestra variable tal que ahora sea igual a h ' A6B2 , y luego el ; es para terminar la definicion y declaracion de la variable.Se tiene que esta variable esta definida como un hexadecimal tal que A6B2 es igual a 42674, lo cual en binario es 1010 0110 1011 0010.

e) Esto se debe a que se redefine a first primero como 1 y se tiene que hay una relacion continua logica entre first y second tal que second es first , por lo que second deberia pasar a ser 0 , pero debido a que no hubo suficiente tiempo entre la linea first = 1 ; y arr[first] <= ((arr[second]));, por lo que el delay que tuvo el second al cambiar debido al cambio de first fue mas grande que el tiempo entre lineas, por lo que second mantiene aun su valor anterior, por lo que cambia el valor de arr[1] por el del negado de arr[1] en vez del negado de arr[0] y nos da un resultado de arr erroneo para como esta hecho el codigo.

Análisis de Código

Para el siguiente ruteo se tomara un input r=3'b0:

	r	x	y	z	n	m	s
	3'b0	8'b0	8'b01110111	3'b0	8'b01110111	3'b001	1'b0
1							1'b1
			8'b01110110	3'b001			
					8'b01110110	3'b010	
2			8b'01110100	3b'010			
					8'b01110100	3b'011	
3				3'b011			
						3'b100	
4		8'b00010000		3'b100			
					8'b10000100	3'b101	
5			8'b01100100	3'b101			
					8'b01110100	3'b110	
6			8'b01000100	3'b110			
					8b'01010100	3'b111	
7				3'b111			
						3'b000	
8		8'b00010001	8'b11000100	3'b000			
					8'b11010101	3'b001	
9		8'b00010011	8'b11000101	3'b001			
					8'b11011000	3'b010	
10		8'b00010111		3'b010			
					8'b11011100	3'b011	

Diseño de Modulo

- El siguiente codigo en Systemverilog es que el que se uso para la simulacion que se adjuntara realizada en modelsim:

```
module bit_fib(input logic clk,
               output logic [4:0] count,output logic fib);
initial begin
   count <= 5'b0;
end
   assign fib = (~(count[4])&~(count[3])&~(count[2]))|
   (~(count[4])&(count[2])&~(count[1])&count[0])|
   (~(count[4])&~(count[2])&~(count[1])&~(count[0]))|
   (~(count[3])&count[2]&~(count[1])&count[0]);
   always@ (posedge clk) begin
        count++;
   end
endmodule
```

- Nota: La linea en la cual se realiza assign fib fue separada por un salto de linea en los "—" para que así pueda ser completamente mostrado en Latex.

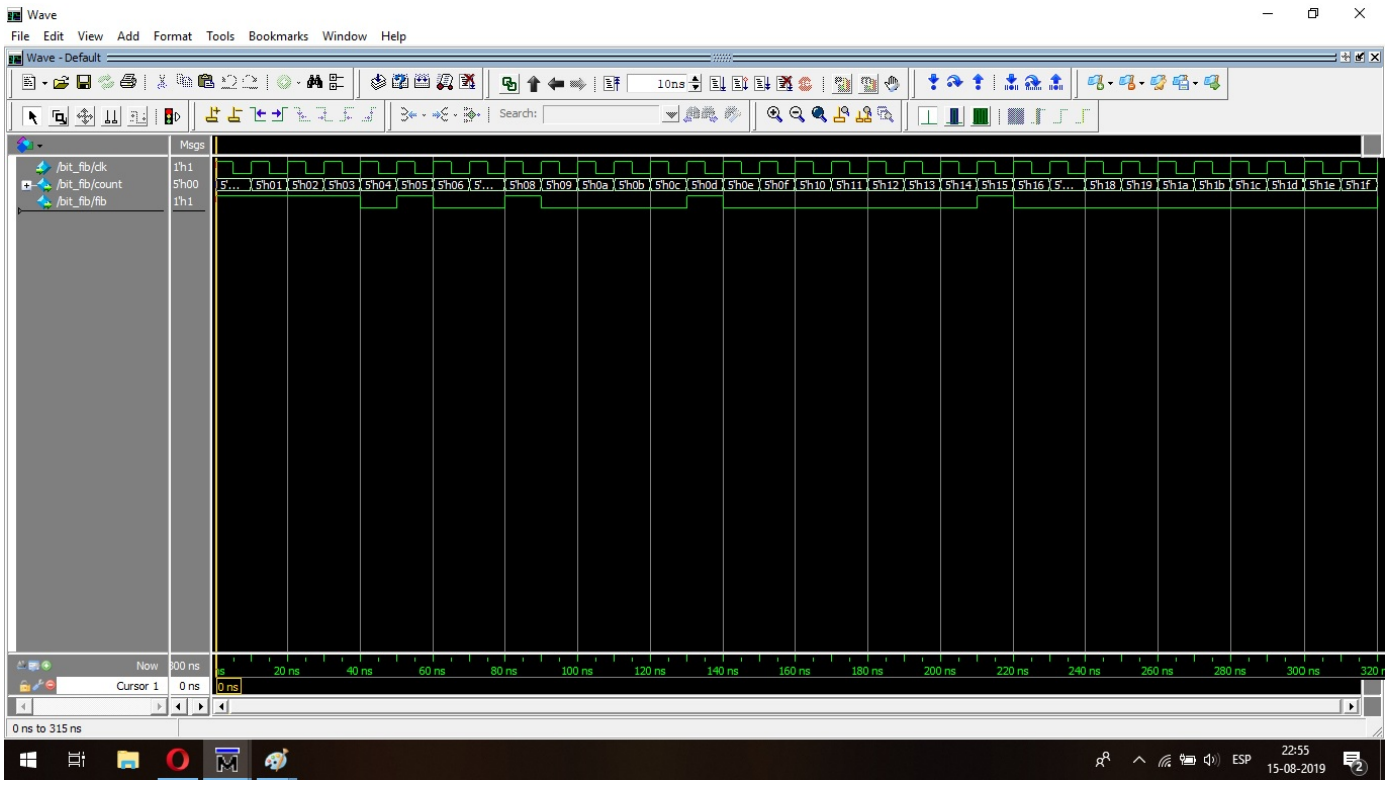

Figure 1: Simulacion

- Nota: La imagen fue ajustada usando Paint para así mostrara de forma correcta todos los numeros que toma nuestro contador.