Pre-Informe 4

Rolando Albarnez Sepulveda
Rol:201673066-4

Pregunta 3.1

- Un lenguaje de descripción de hardware permite documentar las interconexiones y el comportamiento de un circuito electrónico, sin utilizar diagramas esquemáticos. Es un lenguaje especializado que se utiliza para definir la estructura, diseño y operación de circuitos electrónicos, y más comúnmente, de circuitos electrónicos digitales.

  La diferencia entre un HDL y lenguaje de programacion es que en unlenguaje de programacion todo se ejecuta de manera secuencial mientras que en HDL todo se ejecuta de manera paralela.

- – wire: Representan conexiones estructurales (fisicas) entre componentes. No tienen capacidad de almacenamiento de información.
  – reg: Representan variables con capacidad de almacenar información.
  – logic: Tipo de dato de cuatro estados (0, 1, x y z), donde x y z pueden no tener valores establecidos. posee propiedades semejantes a reg .

  La diferencia entre reg y wire, por lo descrito es que uno posee la capacidad de almacenar informacion en tantro el otro representa conexiones fisicas y respecto de logic, este se introdujo como un nuevo tipo de dato que puede actuar tanto como reg y wire. La herramienta interpretará automáticamente su comportamiento de acuerdo con su uso.

- – <=: sin bloqueo y se realiza en cada borde positivo del reloj. estos se evalúan en paralelo, por lo que no hay garantía de orden. Un ejemplo de esto sería un registro.
  – assign : asignación continua para wire fuera de un always statement. El valor de LHS se actualiza cuando RHS cambia.
  – = : asignación de bloqueo, dentro de always statements y las declaraciones imponen orden secuencial.

- – reg [15:0] g: Indica que g es un registro o palabra de 16 bits.
  – h 'A6B2: indica que al registro g almacena el valor en hexadecimal A6B2.

  Su equivalente en binario seria: reg [15:0] g = 16'b1010011010110010 .

- Se puede decir respecto a las aginaciones principalmente a la signacion sin bloqueo (<=) dentro del bloque always_ff garantiza obtener el valor anterior para la palabra/registro arr esto ya que hay un retraso (delay) en las 'lecturas' para un paso de tiempo particular respecto la asignacion sin bloqueo, las se llevan a cabo antes de cualquiera de las 'escrituras', por lo que los valores se pueden intercambiar de manera segura llevando asi a que se intercambien los valores y obteniendo el valor 0111

Pregunta 3.2

Para una mejor comprension del ruteo los valores se expresaran en hexadecimal, se informara el valor de n en binario y decimal.

- Valores antes de cualquier subida: x = 00, y = 77, z = 0, n = 77, m = 1, s = 0 y el valor elegido para r = 1.

- Primera subida: $\overline{s} = 1$, entonces entra en el bloque if (es la unica vez que entra en este bloque). z = 1, s = 1 (no volvera a cambiar de valor), m = 2.
  Luego de las asignaciones y = 76, x = 0, n = 76
- Segunda subida: z = 2 , m = 3, y = 74, x = 00, n = 74.
- Tercera subida: z = 3 , m = 4, y = 74, x = 00, n = 74.
- Cuarta subida: z = 4 , m = 5, y = 74, x = 10, n = 84.
- Quinta subida: z = 5 , m = 6, y = 64, x = 10, n = 74.
- Sexta subida: z = 6 , m = 7, y = 44, x = 10, n = 54.
- Septima subida: z = 7 , m = 0, y = 44, x = 10, n = 54.
- Octava subida: z = 0 , m = 1, y = C4, x = 11, n = D5.
- Novena subida: z = 1 , m = 2, y = C5, x = 13, n = D8.
- Decima subida: z = 2 , m = 3, y = C5, x = 17, n = DC.

Pregunta 3

El codigo seria

C:/modeltech64_10.5/examples/fib.sv (/fib) - Default

```
Ln#
1  module fib(input logic clock,
2          output logic [1:0] is_fibo = 1b0;,
3          output logic [5:0] value);
4
5  reg [4:0] counter = 5'b00000;
6  reg [1:0] A = 1'b0;
7  reg [1:0] B = 1'b0;
8  reg [1:0] C = 1'b0;
9  reg [1:0] D = 1'b0;
10 reg [1:0] E = 1'b0;
11
12 always @(posedge clock)begin
13
14         A <= counter[4];
15         B <= counter[3];
16         C <= counter[2];
17         D <= counter[1];
18         E <= counter[0];
19         if ( (~A & ~B & ~C & D & ~E) | (A & ~C & ~D & ~E) | (~A & B & ~C & ~D) | (~A & B & ~D & E) | (~A & B & ~C & E) ) assign is_fibo = 1;
20         else is_fibo <=0 ;
21
22
23         counter <= counter +1;
24         value <= counter +1;
25     end
26
27 endmodule
28
```

No alcanzo a adjuntar la foto de la simulacion ni diagrama de Karnaugh por tiempo.