Pre-Informe Arquitecturas y Organizacion de Computadores
Experiencia 3

Felipe Cornes
201773060-9

Investigación de Conceptos

a) Un lenguaje de descripción de Hardware es un lenguaje especializado de computador usado para describir estructuras y comportamientos de circuitos electrónicos, y más comúnmente, circuitos electrónicos lógicos.
La principal diferencia entre HDL y los lenguajes de programación es que, como ya mencione, el HDL describe el comportamiento de sistemas digitales, mientras que los lenguajes de programación dan un listado de instrucciones al CPU para realizar una tarea especifica.

b) Wire es usado en asignaciones continuas. Es tratado como un cable, por ende no puede poseer un valor . Puede ser conducido y leido, se usan para conectar distintos modulos. Reg es un elemento de almacenamiento. Puede guardar valores. Retiene el valor hasta la siguiente asignación. Mientras que logic es bastante similar a las mencionadas, la principal diferencia es que logic puede ser conducida por asignaciones continuas o asignaciones de bloqueo y no bloqueo.

c) Los 3 sirven para asignar, la diferencia radica en que:

- assign: asignación continua, ocurre en el orden en que aparece en el archivo
- <= : asignación de bloqueo, ocurre simultaneamente junto a otras asignaciones
- = : asignación de no bloqueo, ocurre en el orden en que aparece en el archivo.

d) Su estructura esta dada por reg, que dice que g almacenara datos, [15:0] da a entender que almacenara un dato de 16 bits, y el h' indica que esta en forma hexadecimal. El numero en binario es 1010011010110010 (en decimal: 42674)

e)

Analisis de Codigo

Usando un r=0 el ruteo fue:

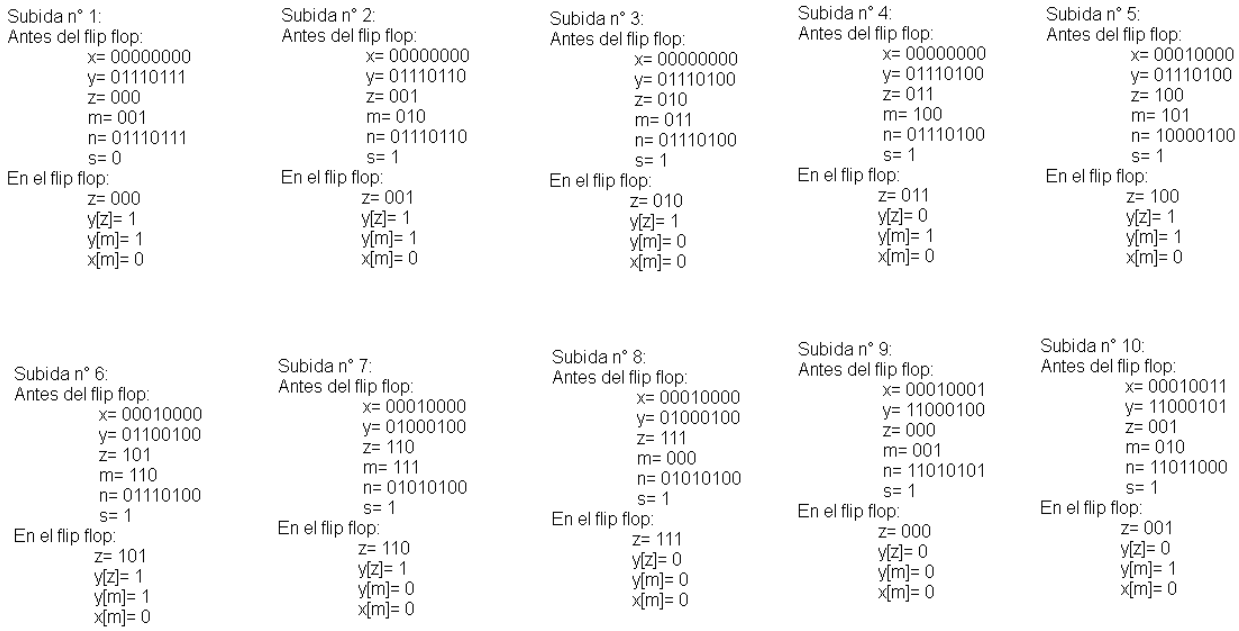

Figure 1: Subidas

Diseño de Modulo

El modulo usado fue:

```
Ln#
 1  module count(input logic clock, reset,
 2                  output logic [4:0] counter,
 3                  output logic fibonacci);
 4
 5          reg [63:0] secu = 64'b11;
 6          reg [31:0] p = 32'b1;
 7          reg [31:0] q = 32'b0;
 8          reg [31:0] act_value = 32'b1;
 9
10
11          always_ff @(posedge clock or posedge reset)
12                  if (reset) counter = 5'b00000;
13                  else counter = counter + 1;
14                  fibonacci = secu[counter];
15          end
16
17          always_ff @(negedge clock) begin
18                  secu[act_value] = 1;
19                  q = p;
20                  p = act_value;
21                  act_value = p + q;
22          end
23
24  endmodule
```

Figure 2: code photo 1

Para testearlo se usó

```
Ln#
 1  `timescale 1ns/1ns;
 2
 3  module run_counter;
 4          logic clock = 0;
 5          reg reset;
 6          wire [4:0] counter;
 7          count beg(clock, reset, counter);
 8
 9          initial begin
10                  reset = 1; #5;
11                  reset = 0;
12          end
13
14          always begin
15                  clock <= 0; #5;
16                  clock <= 1; #5;
17          end
18  endmodule
```

Figure 3: code photo 2

Y el resultado fue:

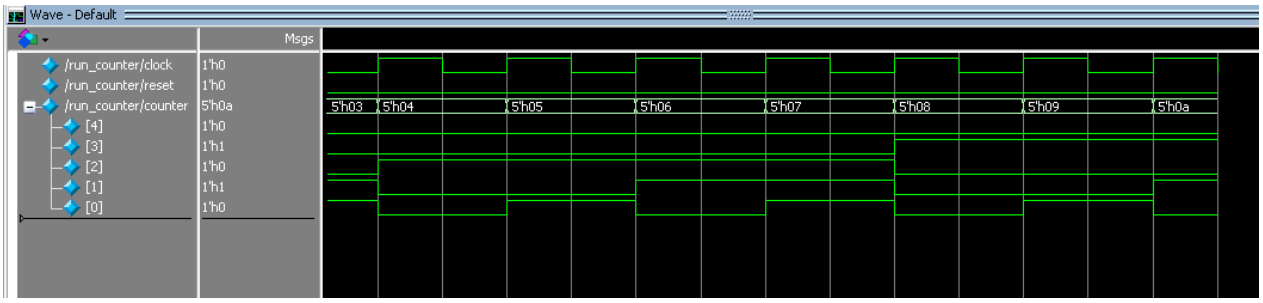

Figure 4: Wave