Pre-Informe 3
Laboratorio de Integración Tecnólogica

Matías Fajardo
201773081-1

Preguntas :

3.1 Investigación de conceptos :

- a)un HDL es un lenguaje de programación especializado, que se utiliza principalmente para definir estructuras, diseño y operaciones de circuitos electrónicos, es por esto, que este lenguaje pueden crear una descripción formar de un circuito electrónico , y posibilitan su análisis automático y su simulación. La principal diferencia consisten en que los lenguajes HDL tienen explicitamente la noción de tiempo

- b)la variable reg, representan variables con capacidad de almacenar información.
  la variable wire, representan conexiones estructurales entre componentes. No tienen capacidad de almacenamiento.
  La variable logic que es de systemverilog a diferencia de las otras dos que son de verilog, tanto reg como wire estan definidos con sus 4 estados (0,1,X,Z), en cambio en systemverilog logic se usará para describir implicitamente que una variable tendrá 4 estados, normalmente se usa para describir inputs,outputs, variables(normalmente en bloques de always), variables nets.

- c)En el caso del " <= ", es un non-blocking statement lo que provocá que dentro de un bloque de always, las líneas de código se ejecuten en paralelo. Es decir, es ejecutado en los cantos de subida del CLK, como por ejemplo una variable reg.
  En cambio el =, es un blocking statement, lo que causa que en un bloque de always se fuerze a seguir la lindea de código secuencialmente, es decir, que la línea de código no se ejecutará si no se ha ejecutado la linea anterior.
  Por otra parte, el assign, como dice la palabra asignará a una variable fuera de los bloques de always, es decir, el lado izquierdo se actualiza siempre y cuando el lado derecho cambia.

- d) Esto es una variable g guardada como registro de 16 bits, y valor almacenado es A6B2, lo cual en binario es 1010 0110 1011 0010.

- e) Los valores iniciales de las variables son :
  $arr = 0101$ $first = 0$ $second = 1$ Despues de la subida de la primera subida del CLK nos queda :
  first = 1
  arr[first] = 0 y el arr[second] = 1, por ende el arr quedará arr = 0111, esto porque de acuerdo a lo visto en clases, el = funciona secuencialmente, entonces first cambiará primero y después esto lo usará para cambiar de valor con el arr[second], pues se trata del mismo indice de bit, ya que el <=, cambia en paralelo y puede existir alguna demora en el tiempo de respuesta lo que hace que cambie el valor de arr, pero como el delay de systemverilog no lo vimos en clases, no puedo argumentar con mayor enfásis.

3.2 Análisis de código

Ruteo para R = 000

- Estados iniciales :
  $y_0 = 01110111$ $x_0 = 00000000$ $z_0 = 000$ $n_0 = 01110111$ $m_0 = 001$ $s_0 = 0$

- Primera subida del clk :
  $y_1 = 01110110$
  $x_1 = 00000000$
  $n_1 = 01100110$

- Segunda subida del clk :
  $y_2 = 01110100$
  $x_2 = 00000000$
  $n_2 = 01100110$

- Tercera subida del clk :
  $y_3 = 01110100$
  $x_3 = 00000000$
  $n_3 = 01110100$

- Cuarta subida del clk :
  $y_4 = 01110100$
  $x_4 = 00010000$
  $n_4 = 10000100$

- Quinta Subida del clk :
  $y_5 = 01100100$
  $x_5 = 00010000$
  $n_5 = 01110100$

- Sexta subida del clk :
  $y_6 = 01000100$
  $x_6 = 00010000$
  $n_6 = 01010100$

- Séptima subida del clk :
  $y_7 = 01000100$
  $x_7 = 00010000$
  $n_7 = 01010100$

- Octava subida del clk :
  $y_8 = 11000100$
  $x_8 = 00010001$
  $n_8 = 11010101$

- Novena Subida del clk :
  $y_9 = 11000101$
  $x_9 = 00010011$
  $n_9 = 11011000$

- Décima subida del clk :
  $y_{10} = 11000101$
  $x_{10} = 00010011$
  $n_{10} = 11011100$

Diseño de modulo

El código para el contador es :

```
module contador(clk,n);//definimos el modulo con las respectivas variables
 input logic clk;//definimos el clock
 reg [4:0] count = 5'd0;
 output reg  n = 1'b1;//output para el valor de fibonacci
 reg [4:0]  x1 = 5'd1; //uno
 reg [4:0]  x2 = 5'd2; //dos
 reg [4:0]  x3 = 5'd3; //tres
 reg [4:0]  x4 = 5'd5; //cinco
 reg [4:0]  x5 = 5'd8; //ocho
 reg [4:0]  x6 = 5'd13; //trece
 reg [4:0]  x7 = 5'd21; //veintiuno
 reg [4:0]  x8 = 5'd0; //cero
 always@(posedge clk)//sentencia para cada subida del clock
        begin
        count<=count+1;
        if((((count+1) == x1) | ((count+1)==x2) | ((count+1)==x3) | ((count+1)==x4) | ((count+1)==x5) | ((count+1)==x6) | ((count+1)==x7)) & n)//if's para el contador y para conocer el estado de n
                begin

                end
        else
                begin
                if(((count+1) == x1) | ((count+1)==x2) | ((count+1)==x3) | ((count+1)==x4) | ((count+1)==x5) | ((count+1)==x6) | ((count+1)==x7) | ((count+1)==x8))//el mismo pero sin n
                        begin
                        n = ~n;
                        end
                else begin
                        n=0;
                        end
                end
        end

endmodule
```

Luego el wave resultado es el sgte, esto hasta el numero mas grande que se puede representar con 5 bits, es decir, 31 :