Pre-Informe 3

Carlos Jara Almendra
201773038-2

Pregunta 1

- A.¿Qué es un lenguaje de descripción de hardware?
  ¿Qué lo diferencia de un lenguaje de programación?

  R: Es un lenguaje de computación que se usa para describir la estructura y el comportamiento de circuitos electrónicos. Éste lenguaje luce casi igual que un lenguje de programación, pero se diferencia de éstos en que HDL incluye la noción de tiempo explícita.

- B. Explique la funcionalidad de los tipos de variable wire, reg y logic, y cuales son sus diferencias. (HINT: Recordar que los registros dependen del CLK)

  R: **Wire** se utiliza para conectar distintos elementos, como si fuera un cable. Sólo se pueden leer o asignar pero no se puede guardar valores en ellos. Se manejan por medio de asignamientos contínuos o desde un puerto del módulo.

  Reg, aunque no siempre corresponde a un registro, representa el almacenamiento de datos en Verilog. Éste retiene su valor hasta que obtiene otro comúnmente por un bloque always() y la influencia de un CLK.

  Logic se puede usar como wire o reg, pero no puede hacer uso de múltiples "drivers" de forma correcta, como se podría hacer por usando wire's y reg's.

- C. ¿Cuál es la diferencia entre los operadores de asignación assign, <= y =, y en que secciones del módulo se utilizan?

  R:
  En verilog se destacan 2 tipos de asignaciones: Contínua o Procedural.

  En la **Contínua** se usa el operador **asssign** en conjunto con **=** de la forma: assign variable = asignación;. Este tipo de asignación no pueede existir dentro de bloques *always* o *initial*

  En la **Procedural** se usa solo el operador **=** de la forma: variable = asignación;

  En cuanto al operador <=, se usa en caso de que se quiera hacer una asignación procedural sin bloqueo, esto quiere decir que el término no se asigna hasta finalizar un determinado instante declarado en el circuito.

- D. Dada la siguiente expresión:

```
reg [15:0]g = 16'hA6B2;
```

Explique su estructura, que significa cada una de sus partes, y cual es el equivalente binario del valor definido.

R: En el código reg es el tipo de variable que corresponde a g; [15:0] indica cuantos bits tiene g; g es la variable; el igual es el operador que indica que es una asignación procedural, 16'hA6B2 es el valor asignado a g; 16 indica de cuantos bits es el número, h indica el tipo de número que en este caso es hexadecimal y A6B2 es el número que en binario es 1010011010110010_2 y en decimal es 42674_{10}.

- E. Si consideramos el siguiente módulo:

```
module Yaled (input logic clk);
reg [3:0] arr = 4'b0101;
reg first = 1'b0;
wire second;
assign second = (~(first));
always_ff@(posedge clk) begin
    first = 1;
    arr [first] <= (~(arr[second]));
    end
endmodule
```

Después de una subida del "clk", el registro "arr" resulta tener el valor binario 0111. Explique porque el delay y las asignaciones dentro del bloque always_ff influyen en este resultado.

R: El delay y las asignaciones influyen, ya que si la línea first = 1; tuviera una asignación no bloqueante de la forma: first <= 1; y tuviera delay, es probable que al pasar a la siguiente línea, first aún tenga el valor anterior (0) y en este caso el output no sería el mismo.

Pregunta 2

Realice un ruteo del valor del output "n" después de 10 subidas del CLK
Defina usted el valor de entrada de 'r', y explicítelo junto al ruteo.

```
module Modulo (input logic [2:0]r,
    input logic clk,
    output logic [7:0]n);
reg [7:0]x = 8'b0;
reg [7:0]y = 8'b01110111;
reg [2:0]z = 3'b0;
wire [2:0]m;
assign n = (x + y)%256;
assign m = (z + 1)%8;
reg s = 1'b0 ;
always_ff@(posedge clk) begin
    if (~(s)) begin
        z = r;
        s <= 1;
        end
    y [z] <= (~(y[m]));
    x [m] <= (~(y[z]));
    z = (z + 1)%8;
    end
endmodule
```

R: Asignando a r el valor 010, el ruteo queda explícito en la siguiente tabla:

ESTADO	y[m]	y[z]	z	n	m	y	x
Q0	-	-	000	01110111	001	01110111	00000000
Q1	0	0	011	01110011	100	01110011	00000000
Q2	0	1	100	10000011	101	01110011	00010000
Q3	0	0	101	01110011	110	01100011	00010000
Q4	0	0	110	01010011	111	01000011	00010000
Q5	1	0	111	01010011	000	01000011	00010000
Q6	0	1	000	01010100	001	01000011	00010001
Q7	0	0	001	01010011	010	01000010	00010001
Q8	1	0	010	01010011	011	01000010	00010001
Q9	1	1	011	01011111	100	01000110	00011001
Q10	1	1	100	01100111	101	01001110	00011001

Pregunta 3

Escriba un módulo en System Verilog para la función siguiente:

Un contador de 5 bits que tenga como outputs la cuenta actual, y un booleano que indique con 1 si es que el numero actual se encuentra en la serie de Fibonacci, o con un 0 si no.

Para el desarrollo de esta pregunta es obligatorio utilizar ModelSim al momento de escribir su código. Debe tomar una captura de pantalla de la simulación una vez el contador alcance su máximo valor, y presentarla junto al modulo escrito.

El código que se usó fué:

```
module fibonacci(input logic [4:0]n,
                 input logic clk,
                 output logic fib);

// lista que tiene los numeros de fibbonacci
reg [31:0]arr = 32'b00000000000100000010000100101111;

reg [4:0]e = 5'b0;
assign fib = arr[e];
reg s = 1'b0;

always_ff@(posedge clk) begin
        if (~(s)) begin
                e = n;
                s <= 1;
                end
        $display("%b",e);
        $display("fibb %b",fib);
        e = e + 1;

        end

endmodule
```

Se hizo uso de clock para hacerlo de forma más sincrónica y poder manejar los displays en consola, en caso de tener que "debuguear".

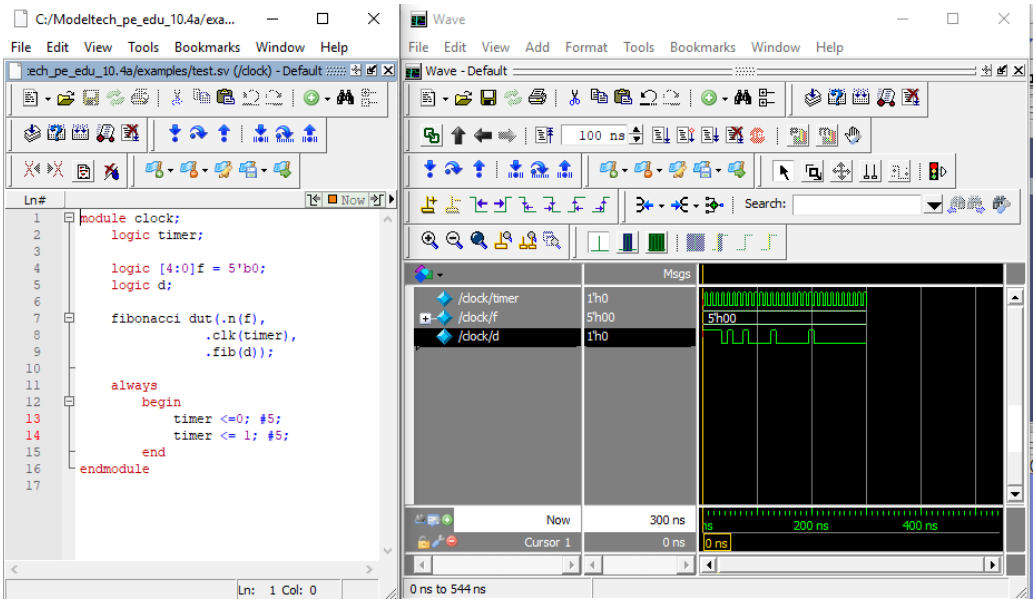

Figure 1: Código usado para correr el contador y el clock