Pre-Informe 3
INF-245

Martín Abbott Silva
201773014-5

Pregunta 1

1. Es un lenguaje formal especializado que se utiliza para definir la estructura, diseño y operación de circuitos electrónicos(digitales generalmente). La principal diferencia es que el HDL describe un comportamiento de un sistema digital, mientras que un lenguaje de programación común entrega un set de instrucciones para que la CPU realice una tarea en específico. También en el HDL se incluye explícitamente la noción del tiempo, la mayoría de sus expresiones se ejecutan concurrentemente, y cada expresión o instrucción corresponde a la operación de un bloque de circuito.

2. Un wire es un tipo de dato de red que representa una conexión física entre entidades estructurales, como puertas o módulos, por lo que actúa como un asignamiento continuo
   Un reg es un tipo de dato variable que representa un valor de asignación que puede ir cambiando debido a las variables implicadas, no mediante nuevas asignaciones. Es usado, por ejemplo, para modelar registros de hardware.
   En SystemVerilog se introdujo logic para describir un genérico tipo de dato de 4 estados. En la prácitca, logic declarará una variable, pero las reglas han sido reescritas de tal forma que se puede usar una variable donde se quiera en diseño RTL. Principales diferencias entre estas variables son el tipo, ya que reg y logic son variables y wire es de red. También wire se usa fuera del always block, mientras que reg se usa adentro. (Otras diferencias se aprecian en las descripciones anteriores).

3. La diferencia entre estos operadores de asignación es que el assing "le dice" al Verilog como evaluar la expresión, por lo que actua como un wire; el <= es usado para asignaciones bajo una lógica secuencial y el = es usado para asignaciones bajo una lógica combinacional. El <= es para nonblock assignment, mientras que el = es para blocking assignment (procedural blocks).

4. Esta línea de código representa la asignación de un número hexadecimal a una variable llamada g. Primero esta el reg[15:0], que es la abstracción de guardado de datos, el cual mantiene el valor en un vector de 16 bits; luego viene el nombre de la variable, llamada g; después está =, que viene siendo la asignación del valor; y finalmente está el valor numérico que mantendrá g, el cual es h'A6B2. La h significa que el número está en hexadecimal, y luego de la ' se dá el valor, que representa un 0xA6B2 en hexadecimal. Su equivalente binario es el 1010 0110 1011 0010.

5. La línea de código "arr[first]<=(˜(arr[second]))" dentro del bloque alwaysff es el que nos da la asignación final y necesaria para que 0101 cambie a 0111; la explicación es la siguiente: En cuanto el canto del clk comienza el bloque, first es 1, y second es 1, ya que antes del bloque alwaysff, first era 0, y second es (˜(first)), por lo tanto 1; entonces "arr[first]<=(˜(arr[second]))" queda así "arr[1]<=(˜(arr[1]))", lo que significa que a la posicion 1 se le asignará la negación de arr[1], que es cero, por lo que su negación es 1, así, finalmente, sabemos que nuestra nueva posición 1 es 1, por lo tanto nos queda el valor 0111. El delay se debe a que hay una cantidad considerable de asignaciones secuenciales en el código, lo que significa que deben hacerse una después de la otra.

Pregunta 2

Si nos definimos r=3'b0, entonces el ruteo de n como de las otras variables implicadas que inciden en el valor de n es sl siguiente:(El ruteo de todas las variables se hará con números de base decimal por un motivo

práctico y visual).

Estado actual	r	x	y	z	n	m	s
Inicial	0	0	119	0	119	1	0
Clk 1	0	0	118	1	118	2	1
Clk 2	0	0	116	2	116	4	1
Clk 3	0	0	116	3	116	4	1
Clk 4	0	16	116	4	132	5	1
Clk 5	0	16	100	5	116	6	1
Clk 6	0	16	68	6	84	7	1
Clk 7	0	16	68	7	84	0	1
Clk 8	0	17	196	0	213	1	1
Clk 9	0	19	197	1	216	2	1
Clk 10	0	23	197	2	220	3	1

Así, finalemte, el valor de n será 220, que en binario es 11011100.

Pregunta 3

Además del contador, debemos determinar cuando el otro output será un 1 o un 0. Esto, como dice el enunciado, será determinado por los números que pertenecen a la serie de Fibonacci que esten del 0 al 31(0,1,2,3,5,8,13,21). Para poder canalizar esto a un código, debemos primero formular la ecuación lógica a través de mapas de karnough.

SUM of PRODUCTS

Map

	$\overline{D}.\overline{E}$	$\overline{D}.E$	$D.E$	$D.\overline{E}$
$\overline{A}.\overline{B}.\overline{C}$	1	1	1	1
$\overline{A}.\overline{B}.C$	0	1	0	0
$\overline{A}.B.C$	0	1	0	0
$\overline{A}.B.\overline{C}$	1	0	0	0
$A.\overline{B}.\overline{C}$	0	0	0	0
$A.\overline{B}.C$	0	1	0	0
$A.B.C$	0	0	0	0
$A.B.\overline{C}$	0	0	0	0

Map Layout

	$\overline{D}.\overline{E}$	$\overline{D}.E$	$D.E$	$D.\overline{E}$
$\overline{A}.\overline{B}.\overline{C}$	0	1	3	2
$\overline{A}.\overline{B}.C$	4	5	7	6
$\overline{A}.B.C$	12	13	15	14
$\overline{A}.B.\overline{C}$	8	9	11	10
$A.\overline{B}.\overline{C}$	16	17	19	18
$A.\overline{B}.C$	20	21	23	22
$A.B.C$	28	29	31	30
$A.B.\overline{C}$	24	25	27	26

Groups

(0,1,2,3)	$\overline{A}.\overline{B}.\overline{C}$
(0,8)	$\overline{A}.\overline{C}.\overline{D}.\overline{E}$
(5,13)	$\overline{A}.C.\overline{D}.E$
(5,21)	$\overline{B}.C.\overline{D}.E$

y = A'B'C' + A'C'D'E' + A'CD'E + B'CD'E

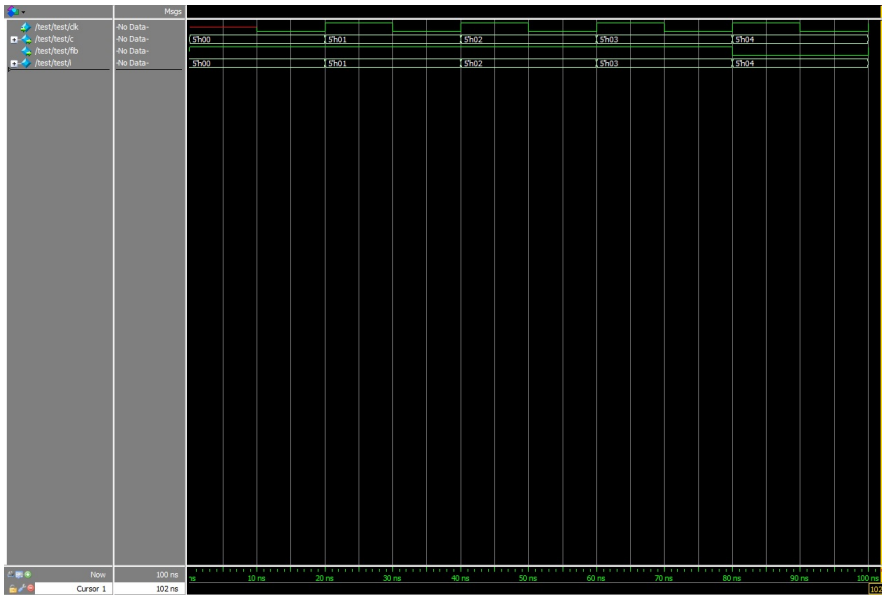
Por lo tanto, nuestro código será el siguiente.

```
Ln#
 1  module countfib(clk,c,fib);
 2          input logic clk;
 3          output logic [4:0] c;
 4          output logic fib ;
 5          reg [4:0]i=0;
 6          assign c=i;
 7          assign fib=((~c[4])&(~c[3])&(~c[2]))|
 8                     ((~c[4])&(~c[2])&(~c[1])&(~c[0]))|
 9                     ((~c[4])&(c[2])&(~c[1])&(c[0]))|
10                     ((~c[3])&(c[2])&(~c[1])&(c[0]));
11          always_ff@(posedge clk) begin
12                  i<=i+1'b1;
13          end
14  endmodule
15
16
```

Y nuestro test bench será:

```
Ln#
 1  module test;
 2          logic clk;
 3          countfib test (clk);
 4          initial begin
 5                  forever begin
 6                  #10 clk=0;
 7                  #10 clk=1;
 8                  end
 9          end
10  endmodule
```

Lo cual al simularlo tenemos:


Msgs
/test/test/clk
/test/test/c
/test/test/lfb
/test/test/l
-No Data-
-No Data-
-No Data-
-No Data-
5'h00
5'h01
5'h02
5'h03
5'h04
5'h00
5'h01
5'h02
5'h03
5'h04
Now
100 ns
Cursor 1
102 ns
10 ns
20 ns
30 ns
40 ns
50 ns
60 ns
70 ns
80 ns
90 ns
100 ns