Pre-Informe 3

Bryan Masias
201773108-7

Pregunta 3.1.a

Es un lenguaje especializado que se utiliza para definir la estructura, diseño y operación de circuitos electrónicos, y además circuitos electrónicos digitales. En cuanto a lo que diferencia la mayoría de las expresiones se "ejecutan" concurrentemente y cada expresión o "instrucción" corresponde a la operación de un bloque de circuito.

Pregunta 3.1.b

Wire: Se utilizan para conectar diferentes elementos, los cuales puedes ser leídos o asignados pero no pueden ser guardados en el, ademas deben ser controlados por una instrucción de asignación continua o desde un puerto de un módulo.
Reg: Representan variables que tienen capacidad de almacenar informacion, la cual a diferencia de wire puede ser usado en circuitos combinacionales o secuenciales.
Logic: No permite múltiples controladores, ademas solo se considera la ultima asignacion. La diferencia con las anteriores es que logic puede ser ambas y controlada por el bloque de asignación,

Pregunta 3.1.c

assign es utilizado para modelar logica combinacional, tan solo puede ser wire, solo puede ser declarada fuera de cualquier proceso y nunca dentro de bloques always o initial. Utilizado en modulos de logica combinacional.(continuous assignment)
<= Es una asignación sin bloqueo, utilizada en logica secuencial.
= Es una asignacion de bloqueo usada para describir logica combinacional.

Pregunta 3.1.d

reg[15:0]g define g como la salida de un registro de 16 bits, 15 es el mas significativo, el 0 el menos significativo.
16h' hace referencia a que es un numero en Base hexadecimal de 16 bits.
A6B2 es el numero hexadecimal almacenado, en binario seria 1010011010110010 y el numero decimal 42674.

Pregunta 3.1.e

El delay ocurre debido a que cuando usamos el = dentro de un always la asignacion se realiza enseguida, mientras que la asignacion de second donde se utiliza <= se asigna una vez terminado el always y por lo tanto el <= second queda con el valor que tenia antes, de esta forma ((arr[second])) queda como 1.

Pregunta 3.2

Tenemos los estados iniciales utilizando $r = 000$:
$y = 01110111$
$x = 00000000$
$z = 000$
$n = 01110111$
$m = 001$
Subida del clock numero 1:
con $(\sim (y[m])) = 0$ y $(\sim (y[z])) = 0$
$y = 01110110$
$x = 00000000$
$z = 001$
$n = 01110110$
$m = 010$
Subida del clock numero 2:
con $(\sim (y[m])) = 0$ y $(\sim (y[z])) = 0$
$y = 01110100$
$x = 00000000$
$z = 010$
$n = 01110100$
$m = 011$
Subida del clock numero 3:
con $(\sim (y[m])) = 1$ y $(\sim (y[z])) = 0$
$y = 01110100$
$x = 00000000$
$z = 011$
$n = 01110100$
$m = 100$
Subida del clock numero 4:
con $(\sim (y[m])) = 0$ y $(\sim (y[z])) = 1$
$y = 01110100$
$x = 00010000$
$z = 100$
$n = 10000100$
$m = 101$
Subida del clock numero 5:
con $(\sim (y[m])) = 0$ y $(\sim (y[z])) = 0$
$y = 01100100$
$x = 00010000$
$z = 101$
$n = 01110100$
$m = 110$
Subida del clock numero 6:
con $(\sim (y[m])) = 0$ y $(\sim (y[z])) = 0$
$y = 01000100$
$x = 00010000$
$z = 110$
$n = 01010100$
$m = 111$

Subida del clock numero 7:
con $(\sim (y[m])) = 1$ y $(\sim (y[z])) = 0$
$y = 01000100$
$x = 00010000$
$z = 111$
$n = 01010100$
$m = 000$
Subida del clock numero 8:
con $(\sim (y[m])) = 1$ y $(\sim (y[z])) = 1$
$y = 11000100$
$x = 00010001$
$z = 000$
$n = 11010101$
$m = 001$
Subida del clock numero 9:
con $(\sim (y[m])) = 1$ y $(\sim (y[z])) = 1$
$y = 11000101$
$x = 00010011$
$z = 001$
$n = 11011000$
$m = 010$
Subida del clock numero 10:
con $(\sim (y[m])) = 0$ y $(\sim (y[z])) = 1$
$y = 11000101$
$x = 00010111$
$z = 010$
$n = 11011100$
$m = 011$
De esta manera el valor de n es 11011100 que en decimal es 220

Pregunta 3.3

```
module counter    (
  output reg [5:0] out     ,  // Output del contador
input  wire      enable  ,  // enable para el contador
input  wire      clk     ,  // clock
input  wire      reset      // reset
);

always_ff @(posedge clk)
if (reset) begin
  out <= 5'b0 ;
end else if (enable) begin
  out ++;
  case(out)
    5'b0 : $display ("1");
    5'b00001 : $display ("1");
    5'b00010 : $display ("1");
    5'b00011 : $display ("1");
    5'b00101 : $display ("1");
    5'b01000 : $display ("1");
    5'b01101 : $display ("1");
    5'b10101 : $display ("1");
    endcase
end

endmodule
```