Pre-Informe 3

Hugo Leyton
201773057-9

Pregunta 1

a) Un lenguaje de descripción de hardware (HDL) es un lenguaje especializado que se utiliza para describir la estructura y el comportamiento de los circuitos electrónicos, y más comúnmente, los circuitos lógicos digitales. Un HDL permite una descripción precisa y formal de un circuito electrónico que permite el análisis automatizado y la simulación de un circuito electrónico.

Un HDL incluye explícitamente la noción de tiempo, esto lo hace diferente a un lenguaje de programación, también el hecho de que la mayoría de las expresiones se "ejecutan" concurrentemente y cada expresión o "instrucción" corresponde a la operación de un bloque de circuito.

b) Las variables de tipo *wire* representan conexiones estructurales entre componentes y no tienen capacidad de almacenamiento. Podemos decir que se usan para conectar módulos y con "assign".

Las variables de tipo *reg* representan variables con capacidad de almacenar información. También se usan en procedimientos ("always").

Las variables de tipo *logic* son idénticas a reg en todos los sentidos.

En cuanto a diferencias, podemos decir lo siguiente:

- ***wire*:** El tipo de dato *wire* se utiliza en las asignaciones continuas o en la lista de puertos. Se trata como un cable, por lo que no puede contener un valor. Se puede conducir y leer. Los cables se utilizan para conectar diferentes módulos.
- ***reg*:** es un elemento de almacenamiento de fecha en System Verilog (depende del CLK). No es un registro de hardware real, pero puede almacenar valores. El registro retiene su valor hasta la siguiente declaración de asignación.
- ***logic*:** System Verilog agregó este tipo de dato adicional que extiende al tipo *rand*, por ejemplo, para que pueda ser controlado por un solo controlador, como una compuerta o un módulo. La principal diferencia entre el tipo de dato *logic* y *wire*/*reg* es que *logic* puede ser manejado tanto por asignación continua como por asignación de bloqueo/no bloqueo.

c) Las diferencias son las siguientes:

- `assign :` Asignación continua para cablear fuera de una declaración *always*. El valor de LHS se actualiza cuando RHS cambia.
- `<= :` Sin bloqueo y se realiza en cada flanco positivo del CLK. Estos se evalúan en paralelo, por lo que no hay garantía de orden. Un ejemplo de esto sería un registro.
- `= :` Asignación de bloqueo, dentro de las declaraciones siempre impone el orden secuencial.

d) Veamos que significa cada parte:

- `reg` : Este es el tipo de dato, en este caso será un registro.
- `[15:0]` : Aquí se establece que se utilizarán múltiples bits, 16 en este caso.

- `g` : Este es el nombre del registro.
- `=` : Este es el operador de asignación.
- `16'hA6B2` : La parte `16'h` define que será un registro hexadecimal con 16 bits. La parte `A6B2` establece el valor en hexadecimal.

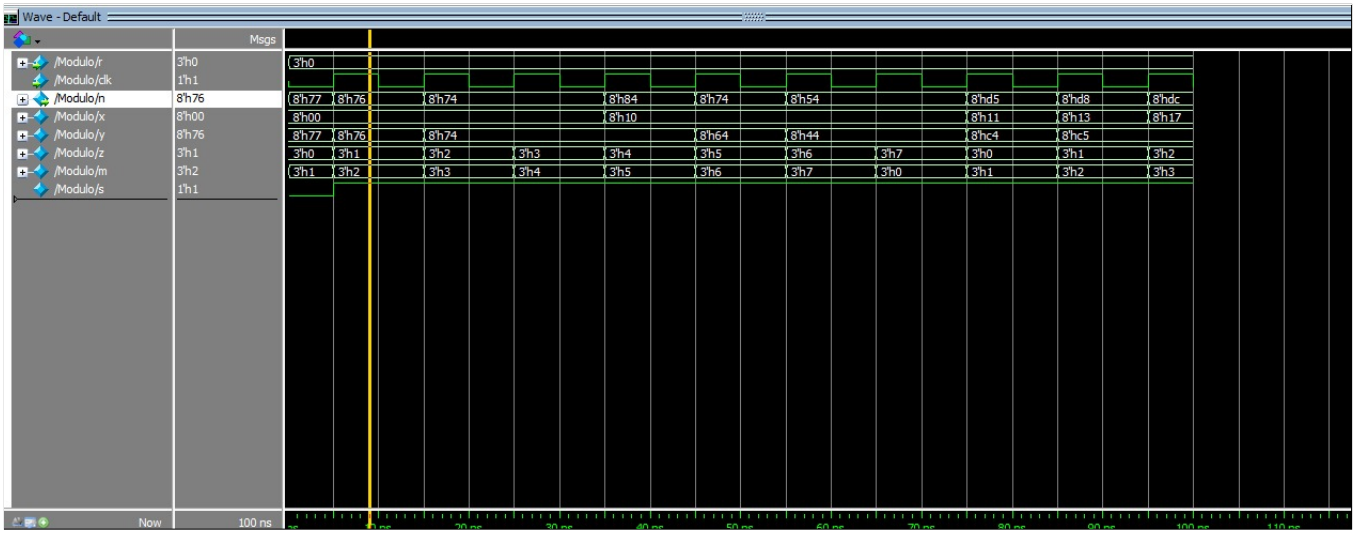
El equivalente en binario sería `reg [15:0] g = 16'b1010011010110010`.

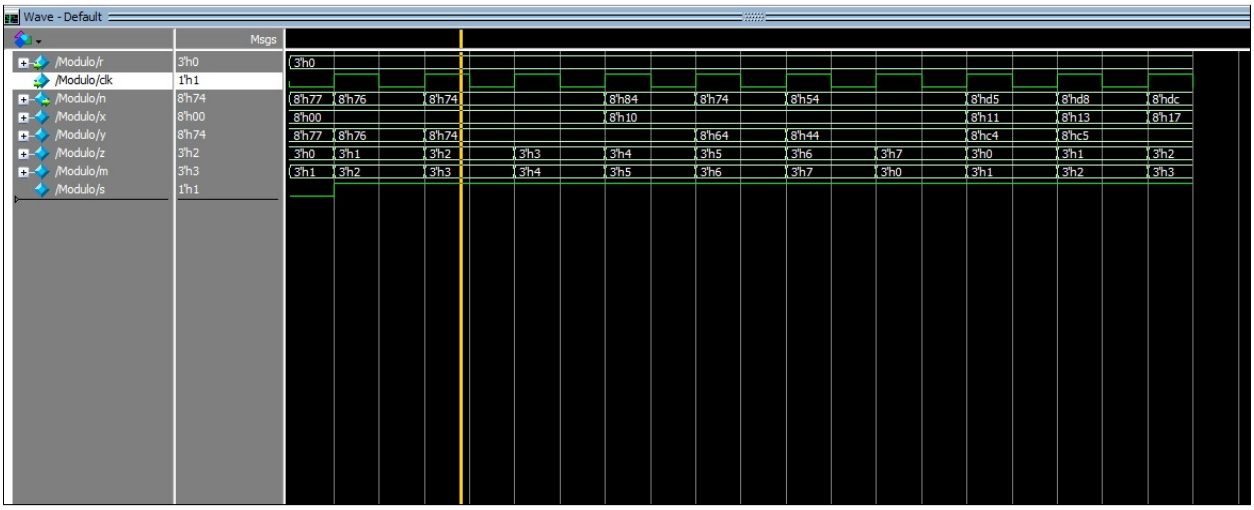
e) El delay influye producto de que el módulo se ejecuta una sola vez. Las asignaciones efectuadas están comentadas a continuación y reflejan en conclusión cómo es que el registro `arr` resulta tener el valor binario 0111.

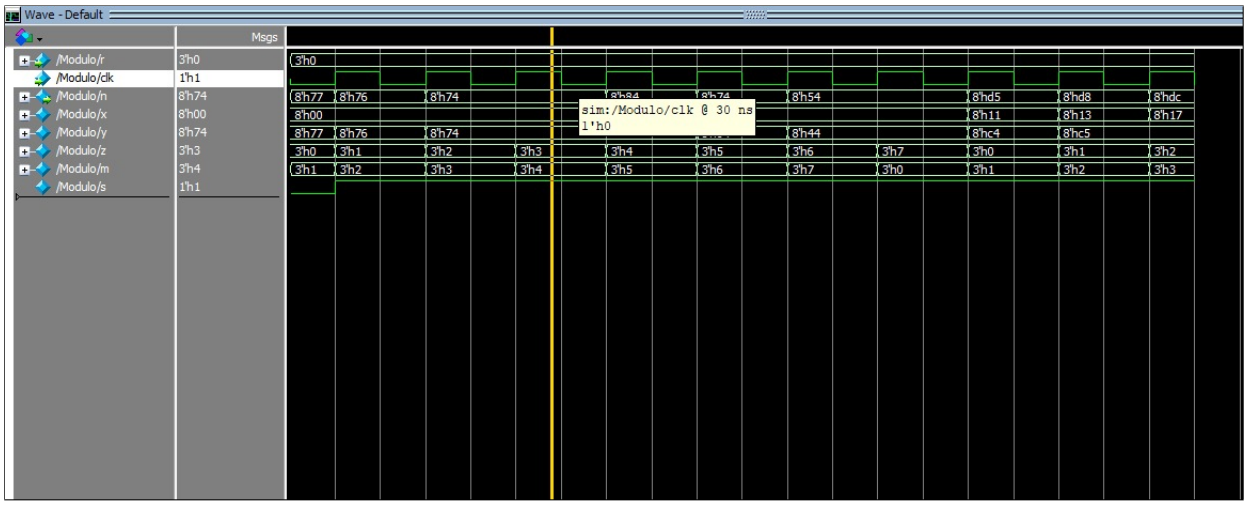
```
module Yaled (input logic clk);
    reg [3:0] arr = 4'b0101;
    reg first = 1'b0; // first = 0
    wire second;
    assign second = (~(first)); // second = ~0 = 1
    always_ff@ (posedge clk) begin
        first = 1;
        arr [first] <= (~(arr [second])); // arr [1] <= ~(arr [1]) = ~0 = 1
  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

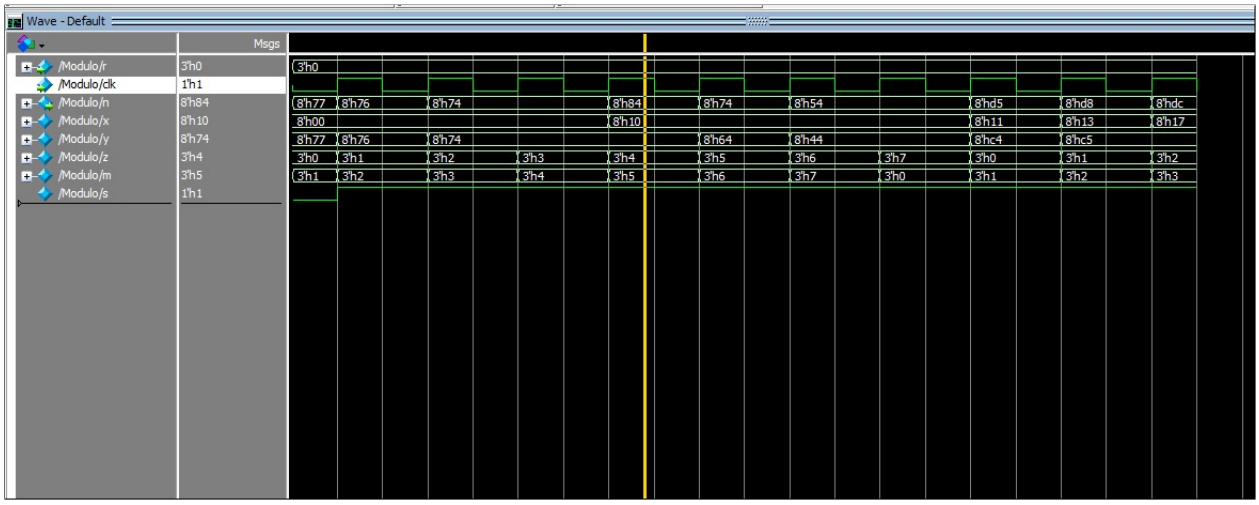
Pregunta 2

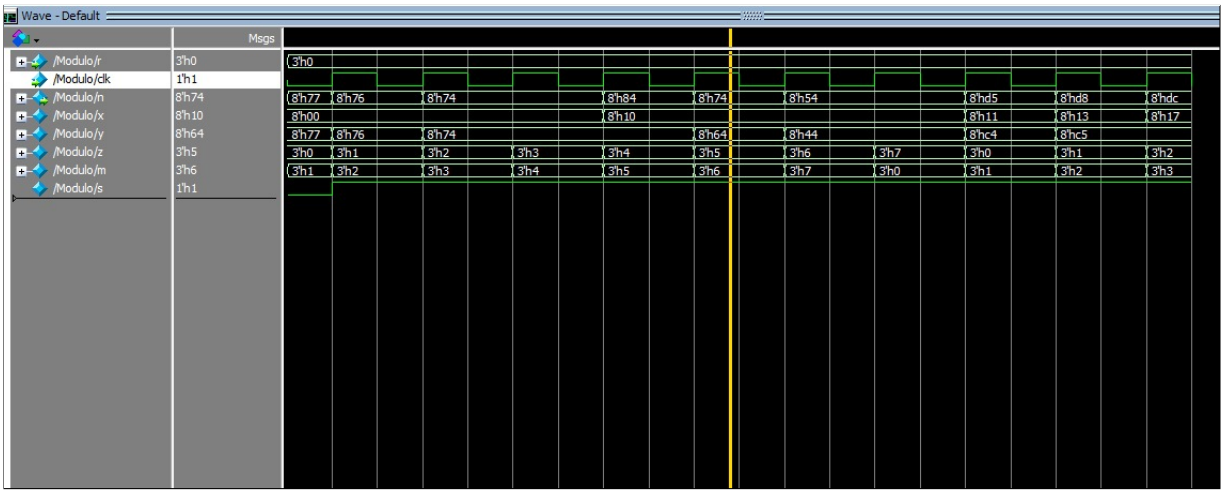
El ruteo queda representado por la siguiente simulación:

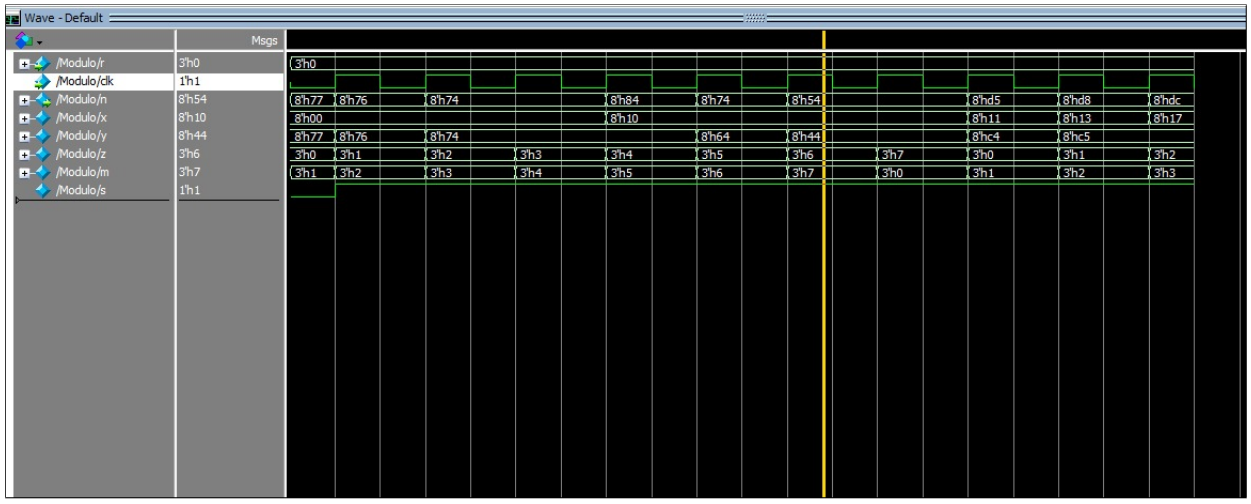


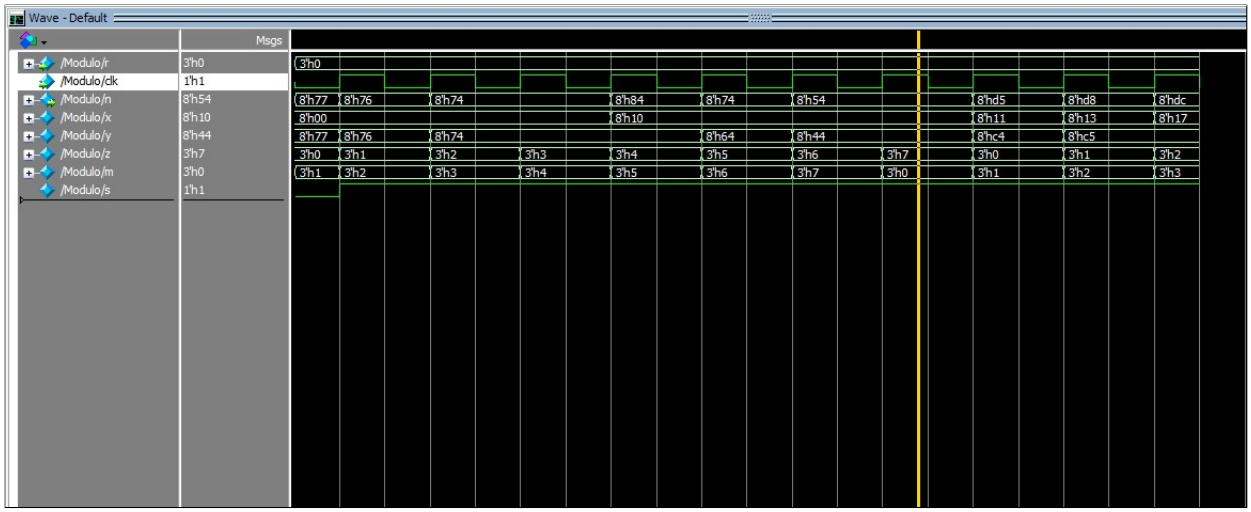

Wave - Default
Msgs
/Modulo/r
/Modulo/clk
/Modulo/n
/Modulo/x
/Modulo/y
/Modulo/z
/Modulo/m
/Modulo/s

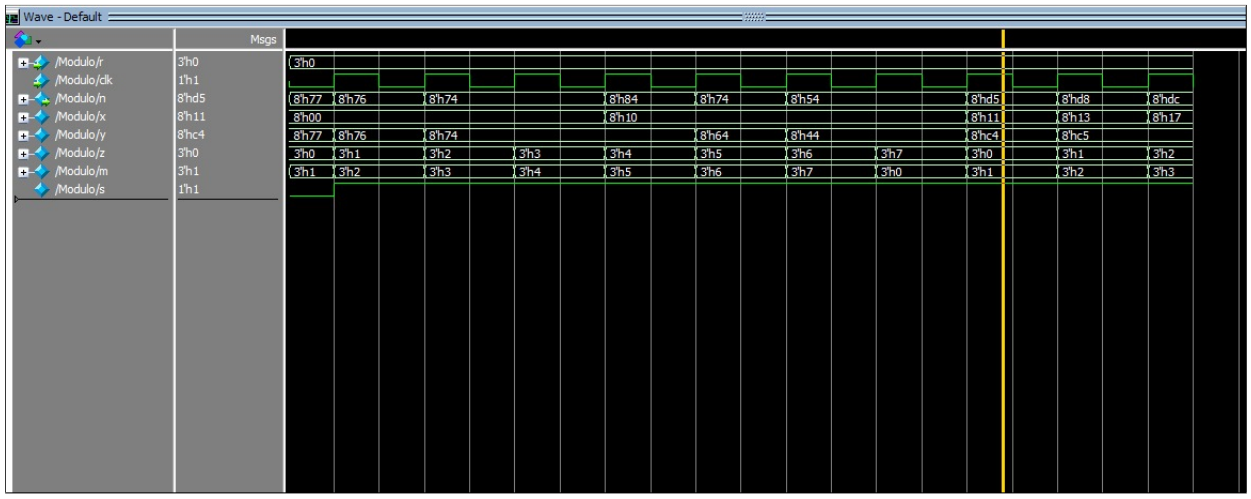

Wave - Default
Msgs
/Modulo/r
/Modulo/clk
/Modulo/n
/Modulo/x
/Modulo/y
/Modulo/z
/Modulo/m
/Modulo/s
sim:/Modulo/clk @ 30 ns
1'h0

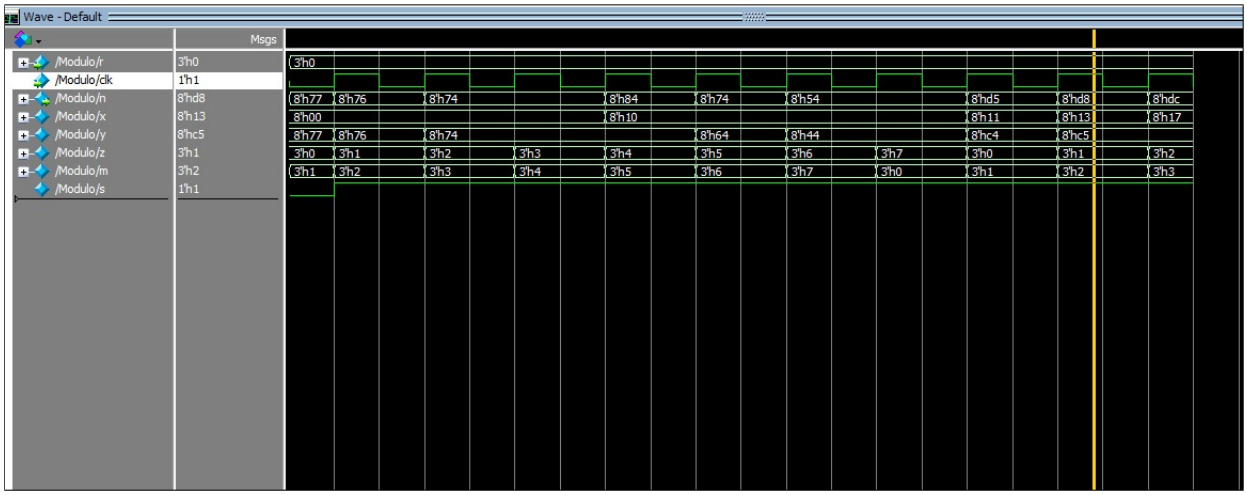

Wave - Default
Msgs
/Modulo/r
/Modulo/clk
/Modulo/n
/Modulo/x
/Modulo/y
/Modulo/z
/Modulo/m
/Modulo/s

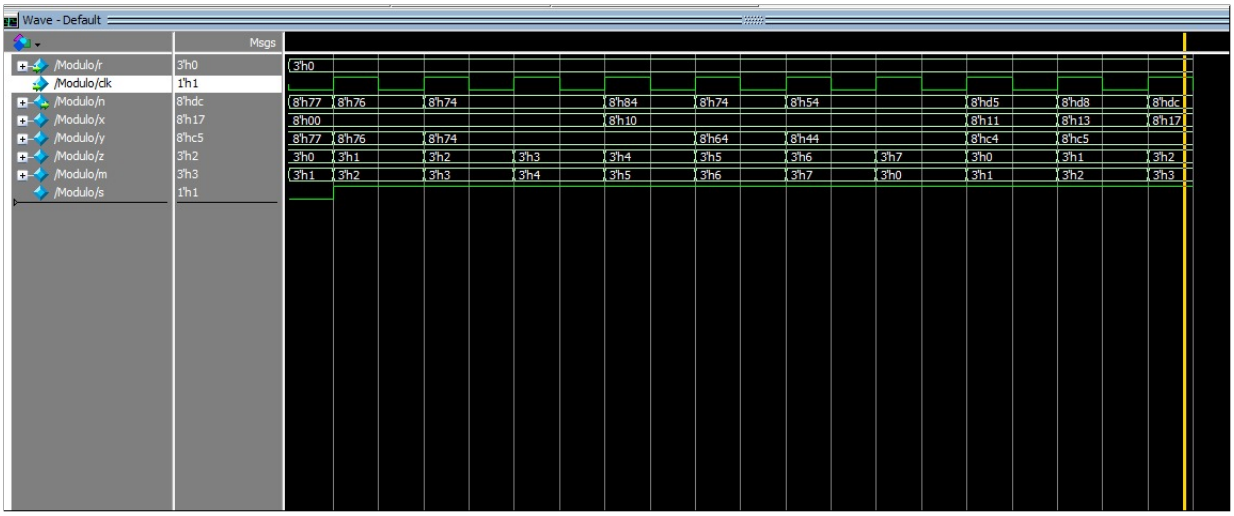

Wave - Default
Msgs
/Modulo/r
/Modulo/clk
/Modulo/n
/Modulo/x
/Modulo/y
/Modulo/z
/Modulo/m
/Modulo/s
3'h0
1'h1
8'h74
8'h10
8'h64
3'h5
3'h6
1'h1
8'h77 8'h76 8'h74 8'h84 8'h74 8'h54 8'hd5 8'hd8 8'hdc
8'h00 8'h10 8'h11 8'h13 8'h17
8'h77 8'h76 8'h74 8'h64 8'h44 8'hc4 8'hc5
3'h0 3'h1 3'h2 3'h3 3'h4 3'h5 3'h6 3'h7 3'h0 3'h1 3'h2
3'h1 3'h2 3'h3 3'h4 3'h5 3'h6 3'h7 3'h0 3'h1 3'h2 3'h3


Wave - Default
Msgs
/Modulo/r
/Modulo/clk
/Modulo/n
/Modulo/x
/Modulo/y
/Modulo/z
/Modulo/m
/Modulo/s
3'h0
1'h1
8'h54
8'h10
8'h44
3'h6
3'h7
1'h1
8'h77 8'h76 8'h74 8'h84 8'h74 8'h54 8'hd5 8'hd8 8'hdc
8'h00 8'h10 8'h11 8'h13 8'h17
8'h77 8'h76 8'h74 8'h64 8'h44 8'hc4 8'hc5
3'h0 3'h1 3'h2 3'h3 3'h4 3'h5 3'h6 3'h7 3'h0 3'h1 3'h2
3'h1 3'h2 3'h3 3'h4 3'h5 3'h6 3'h7 3'h0 3'h1 3'h2 3'h3


Wave - Default
Msgs
/Modulo/r
/Modulo/clk
/Modulo/n
/Modulo/x
/Modulo/y
/Modulo/z
/Modulo/m
/Modulo/s
3'h0
1'h1
8'h54
8'h10
8'h44
3'h7
3'h0
1'h1
8'h77 8'h76 8'h74 8'h84 8'h74 8'h54 8'hd5 8'hd8 8'hdc
8'h00 8'h10 8'h11 8'h13 8'h17
8'h77 8'h76 8'h74 8'h64 8'h44 8'hc4 8'hc5
3'h0 3'h1 3'h2 3'h3 3'h4 3'h5 3'h6 3'h7 3'h0 3'h1 3'h2
3'h1 3'h2 3'h3 3'h4 3'h5 3'h6 3'h7 3'h0 3'h1 3'h2 3'h3







Pregunta 3

El código del módulo elaborado es el siguiente:

```
module counter(input clk, output [4:0] data, output logic y);
    wire clk;
    reg [4:0] data = 0;
    always @(posedge clk) begin
        data <= data + 1;
        case (data)
            5'b00000, 5'b00001, 5'b00011, 5'b00101, 5'b01000, 5'b01101, 5'b10101:  begin
                assign y = 1'b1;
            end
            default:
                assign y = 1'b0;
        endcase
    end
endmodule
```

La simulación es la siguiente: