NOTA:

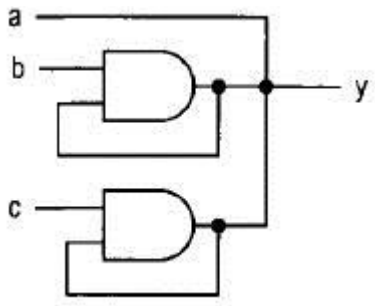
Todos los problemas aquí planteados deberán ser resueltos para ser mostrados el día lunes 24 de agosto.

Se deberá mostrar tanto el código en lenguaje Verilog que describe el circuito que da solución a cada problema como el código de los Test Benches y los resultados obtenidos de la ejecución de los mismos.

Problemas

1) Diseñar un multiplexor de 4 entradas (a,b,c,d) y una salida (y) describiéndolo en nivel RTL.
2) Agregar la posibilidad de reset/set a la salida del multiplexor utilizando una señal llamada 'rs'. El funcionamiento de la misma será: si rs = '0' la salida 'y' será '0' independientemente de la entrada. De igual modo si rs = '1' la salida 'y' será '1'.
3) Diseñar el mismo multiplexor del problema (1) pero con una descripción estructural.
4) Diseñar un circuito half adder de 2 bits.
5) Diseñar un circuito full adder de 2 bits.
6) Diseñar un decodificador de 7 segmentos para un solo display:
    a. El display es anodo común.
    b. Deberá ser capaz de mostrar los números comprendidos entre 0 y 9, para los demás valores deberá mostrar la letra "H". Para esto utilizara 4 bits de entrada donde "0000" → 0 y "1001" → 9;
7)

    El circuito:



Es equivalente a la descripción en verilog:

```
module a (
input wire a,b,c,
output wire y
);
   assign y = a;
   assign y = y & b;
   assign y = y & c;
endmodule
```

☐
```
module b (
input wire a,b,c,
output reg y
);
   always @*
   begin
         y = a;
         y = y & b;
         y  = y & c;
   end
endmodule
```

8) Marque con una cruz las afirmaciones que sean correctas:

☐ En Verilog un “wire” se utiliza para conectar puertos “input” y “output” de un modulo instanciado con otros elementos del diseño.

☐ Los elementos “wire” tienen que ser manejados por algo, y no pueden almacenar un valor sin ser manejados.

☐ Los elementos “wire” pueden ser utilizados en el lado izquierdo de un símbolo “=” o “<=” dentro de un bloque always.

☐ Los elementos “wire” pueden estar en el lado izquierdo solo cuando se encuentra en una asignación continua (utilizando “assign”)

☐ Los elementos “wire” se utilizan tanto para modelar lógica combinacional como lógica secuencial.

☐ Un elemento “reg” puede ser conectado a un puerto “input” de una instanciación de modulo.

☐ Un elemento “reg” no puede ser conectado a un puerto “output” de una instanciación de modulo.

☐ Un elemento “reg” es el único elemento permitido en el lado izquierdo dentro de un bloque always@

☐ Un elemento “reg” puede ser utilizado para crear registros cuando este es utilizado dentro de un bloque always@(posedge clock).

☐ Un elemento “reg” se puede utilizar para crear lógica combinacional y secuencial.

☐ Un elemento “reg” es similar a un elemento “wire” solo que puede almacenar información.

☐ En Verilog un elemento “reg” es siempre un registro de hardware

9) Cuando se realiza una asignación incompleta (es decir, no se contemplan todas las posibilidades) como por ejemplo el siguiente multiplexor donde no se tiene en cuenta el caso en el que sel = '11':

```
always @(a or b or c or sel)
begin
    case (sel)
        2'b00: out = a;
        2'b01: out = b;
        2'b10: out = c;
    endcase
end
```

La herramienta generara:

- ☐ Un Latch para poder contener el valor de "la cuarta señal"
- ☐ La herramienta "entiende" y genera un multiplexor con 3 entradas
- ☐ Ninguna de las anteriores

10) Realizar una Unidad Aritmetica y Lógica (ALU) con las siguientes características:

- La ALU debe ser parametrizable (bus de datos) para
- poder ser utilizada posteriormente en el trabajo final.
- La ALU deberá realizar las siguientes operaciones:

Operacion	Codigo
ADD	100000
SUB	100010
AND	100100
OR	100101
XOR	100110
SRA	000011
SRL	000010
NOR	100111