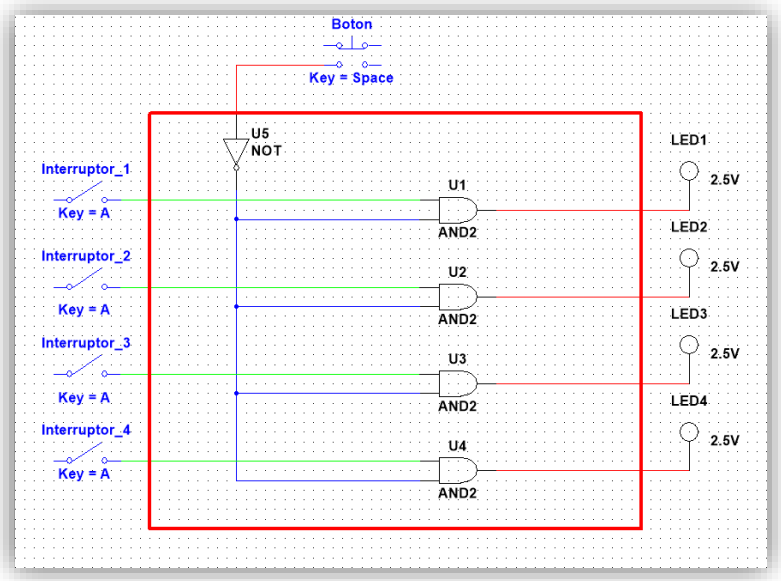
Documentación Ejercicio 1:

El siguiente código busca controlar 16 LEDs en una tarjeta FPGA mediante los interruptores presentes en esta misma. Además, estas 16 interacciones se subdividen en cuatro grupos de acciones iguales. Esto con el fin que cada grupo sea gobernado por un botón de la FPGA. Cuando este botón este presionado ninguno de los cuatro LEDs del grupo puede ser encendido. Para lograr esto primero se presenta una tabla de verdad que representa esta interacción.

s1	s2	s3	s4	botón	l1	l2	l3	l4
x	x	x	x	1	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	1
0	0	1	0	0	0	0	1	0
0	1	0	0	0	0	1	0	0
1	0	0	0	0	1	0	0	0

De la tabla anterior de llama a cada interruptor “sn” donde n es el número y lo mismo ocurre con los LEDs, “ln”. El comportamiento derivado de la tabla es cuando se enciende un interruptor se enciende el LED correspondiente. Esto ocurre siempre y cuando el botón que gobierna esta zona este en 0. Si el botón está en 1, los interruptores no importan y todos los LEDs deben apagarse. Con la tabla es fácil identificar la relación entre los interruptores y los LEDs. Para cumplir que el LED se encienda cuando el interruptor correspondiente se encienda se presenta una operación de AND con el interruptor y la negación del botón. El botón se niega ya que por defecto cuando se presiona entrega un valor de cero. Esta interacción se puede apreciar en el siguiente diagrama:



Al contar con una idea clara del funcionamiento del modulo encargado del comportamiento principal del programa se presenta el código del módulo en SystemVerilog:

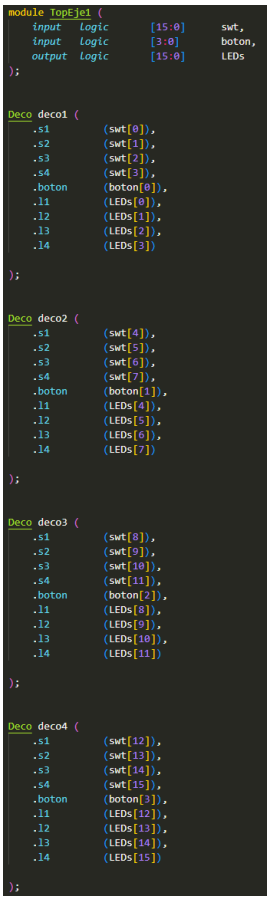
```
module Deco(
    input       Logic s1,
    input       Logic s2,
    input       Logic s3,
    input       Logic s4,
    input       Logic boton,
    output      Logic l1,
    output      Logic l2,
    output      Logic l3,
    output      Logic l4
);

always_comb begin : deco
    l1 = s1 && ~boton;
    l2 = s2 && ~boton;
    l3 = s3 && ~boton;
    l4 = s4 && ~boton;
end


endmodule
```

Transformando el diseño propuesto a un modulo de SystemVerilog se puede apreciar como este tiene cinco señales de entrada las cuales son cuatro por cada interruptor y una para el botón. Similarmente las señales de salida son cuatro ya que es una para cada LED presente. Dentro del always combinacional se puede apreciar el comportamiento del AND presente. Ya que se le asigna al LED de salida el valor de la operación resultante entre el valor del interruptor y la negación del botón. Lo cual logra que el valor del interruptor sea transferido al LED siempre y cuando no se esté presionando del botón.

Ya que el diseño este hecho para un grupo de cuatro interruptores, cuatro LEDs y un botón. Pero el programa busca controlar 16 LEDS, 16 interruptores y 4 botones, se crea un archivo TOP que contenga 4 instancias idénticas del modulo descrito anteriormente.



```
module TopEje1 (
    input   logic       [15:0]      swt,
    input   logic       [3:0]       boton,
    output  logic       [15:0]      LEDs
);


Deco deco1 (
    .s1         (swt[0]),
    .s2         (swt[1]),
    .s3         (swt[2]),
    .s4         (swt[3]),
    .boton      (boton[0]),
    .l1         (LEDs[0]),
    .l2         (LEDs[1]),
    .l3         (LEDs[2]),
    .l4         (LEDs[3])

);


Deco deco2 (
    .s1         (swt[4]),
    .s2         (swt[5]),
    .s3         (swt[6]),
    .s4         (swt[7]),
    .boton      (boton[1]),
    .l1         (LEDs[4]),
    .l2         (LEDs[5]),
    .l3         (LEDs[6]),
    .l4         (LEDs[7])

);


Deco deco3 (
    .s1         (swt[8]),
    .s2         (swt[9]),
    .s3         (swt[10]),
    .s4         (swt[11]),
    .boton      (boton[2]),
    .l1         (LEDs[8]),
    .l2         (LEDs[9]),
    .l3         (LEDs[10]),
    .l4         (LEDs[11])

);


Deco deco4 (
    .s1         (swt[12]),
    .s2         (swt[13]),
    .s3         (swt[14]),
    .s4         (swt[15]),
    .boton      (boton[3]),
    .l1         (LEDs[12]),
    .l2         (LEDs[13]),
    .l3         (LEDs[14]),
    .l4         (LEDs[15])

);
```

Con la implementación del TOP se pueden generar la interacción solicitada de controlar los 16 LEDs mediante 16 interruptores que. Este código de TOP se puede apreciar como la entrada es una señal de 16 bits (uno por cada interruptor), una señal de entrada de 4 bits (uno por cada botón) y una señal de salida de 16 bits (uno por cada LED). Al asignar cada bit a cada instancia correcta se logra el comportamiento deseado. Además, el código presentado es sintetizable lo cual significa que esta listo para ser programado en una FPGA y ser probado de forma física.