Electrónica Digital 1
Laboratorio 8
24/10/2020

Luis Pedro Molina (18822), Ingeniería Mecatrónica, UVG

Ejercicio 1 - Contador

Este contador es un contador asíncrono de 12 bits. Es un contador común y corriente hasta cierto punto. El contador comienza a funcionar, luego de resetear el mismo y seguido de esto, que el ENABLE se encuentre en 1. Sin embargo, cuenta con la posibilidad de introducir un número y que el contador cuente a partir de dicho número. Esta función opera en conjunto con su propio "enable" por así decirlo, variable la cual fue llamada en este módulo como "nonblck" por "Non-blocking".

En la figura 1 se pueden apreciar las variables asignadas. La variable "LOAD" es la variable que trabaja en conjunto con la "nonblck".
El primer if el módulo básicamente nos dice que si el reset = 1 la salida Q del contador será de 000000000000.
El primer else if nos dice que cuando ENABLE = 1 el contador comenzará a funcionar, es decir Q+1.
El siguiente else if nos indica que cuando nonblck = 1 se tomará en cuenta el valor introducido en la variable LOAD.

Figura 1 - Módulo contador

En el testbench, se asignaron las variables y se procedió al planteamiento de las operaciones. el clk fue de 1ms. Como primer punto se resetea el contador para asegurarse que comience en 0. Luego no se presiona nada, por lo tanto el contador no realiza nada. Se enciende el enable y el contador comienza a continuar. Al llegar a 2 el enable se apaga y se enciende el nonblck y se introduce el valor 32. Luego se apaga el nonblck y se enciende el enable hasta que el contador llega a 34. Luego se vuelve a encender el nonblck y se carga la variable load con el número 256. Luego se apaga el nonblck y se enciende el enable hasta que llega el contador a 258. Se reinicia el contador y seguido de esto se enciende el enable hasta contar a 2 nuevamente.
En el testbench se utilizó \t en los displays de valores con el propósito de que los valores se desplegarán de forma ordenada y tabulada, en la tabla, valga la redundancia.

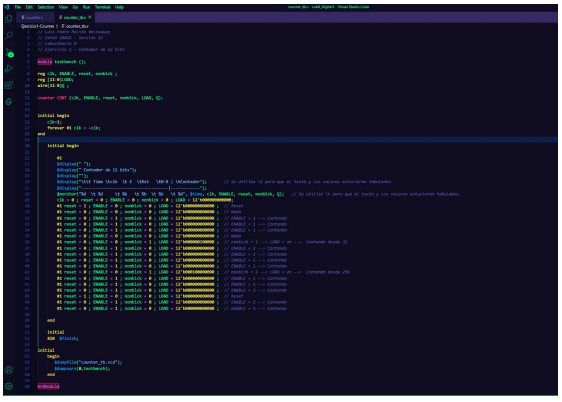


Figura 2 - Módulo testbench contador

Time	clk	E	Rst	N-B	Contador
1	0	0	0	0	x
2	1	0	1	0	0
3	0	0	0	0	0
4	1	1	0	0	1
5	0	1	0	0	1
6	1	1	0	0	2
7	0	0	0	0	2
8	1	0	0	1	32
9	0	1	0	0	32
10	1	1	0	0	33
11	0	1	0	0	33
12	1	1	0	0	34
13	0	0	0	1	256
14	1	1	0	0	257
15	0	1	0	0	257
16	1	1	0	0	258
17	0	0	1	0	0
18	1	1	0	0	1
19	0	1	0	0	1
20	1	1	0	0	2

Figura 3 - Tabla de valores

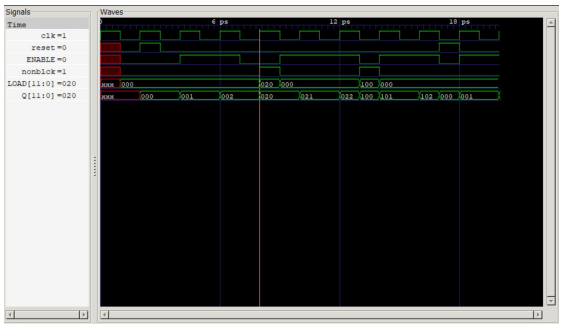


Figura 4 - Diagrama de Timming

Ejercicio 2 - Memoria ROM

En este módulo se definen los inputs y outputs de la memoria. Se creó un archivo externo, cuyó nombre fue "memoria.listLP" el cual fue llamado en el módulo de la memoria ROM. Se mandaron a llamar dichos valores y se almacenaron en otra variable llamada Datos.

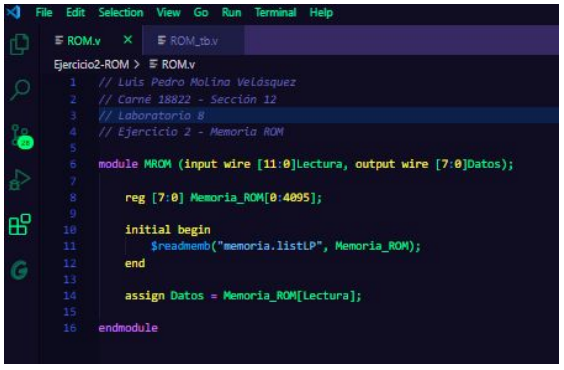


Figura 5 - Módulo Memoria ROM

En el testbench se hicieron las pruebas de dicho módulo, en el cual se procedió a llamar los 10 valores asignados en la memoria.listLP. Dichos valores se pueden observar en la figura 8 y 9.

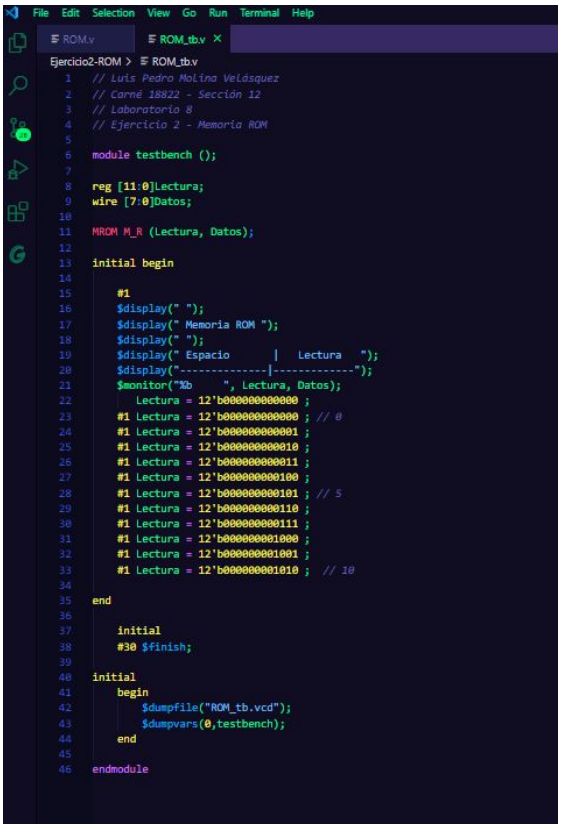


Figura 6 - Módulo testbench ROM

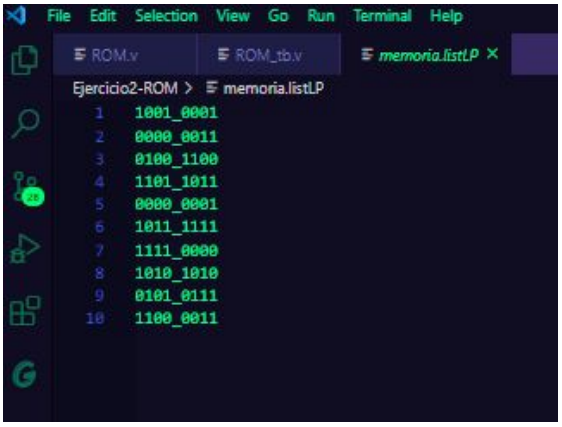


Figura 7 - Lista de memoria a llamar

Memoria ROM

Espacio	Lectura
000000000000	145
000000000001	3
000000000010	76
000000000011	219
000000000100	1
000000000101	191
000000000110	240
000000000111	170
000000001000	87
000000001001	195
000000001010	x

Figura 8 - Tabla valores

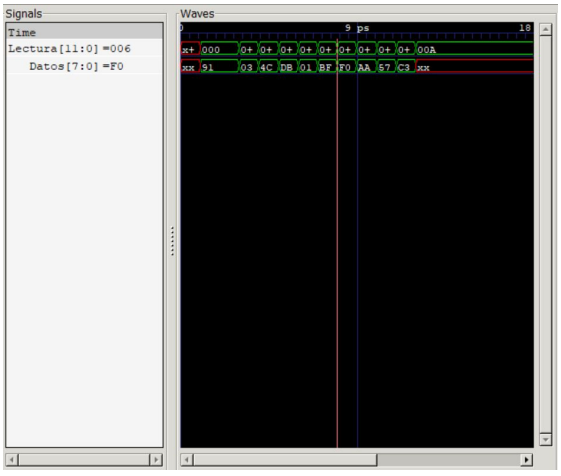
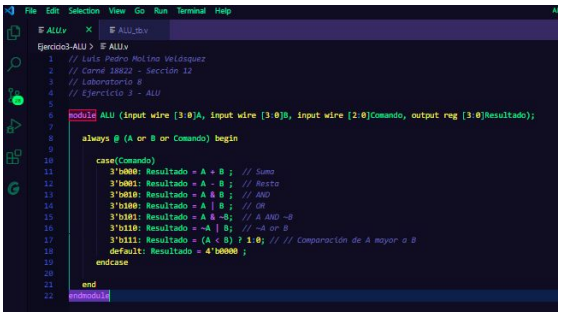


Figura 9 - Diagrama de Timming

Ejercicio 3 - ALU

La ALU es básicamente una máquina capaz de realizar distintas operaciones. Se asignaron 2 variables de 4 bits, 1 variable de 3 bits en la cual se almacenaron las distintas operaciones posibles, como suma, resta etc y se asignó una variable que desplegará el resultado de la operación realizada, de 4 bits.

En el módulo testbench se pueden apreciar las operaciones realizadas. Se realizaron 2 pruebas distintas para cada operación previamente almacenada en la variable de 3 bits. Los resultados se pueden apreciar en la figura 12 y 13.



```
// Luis Pedro Molina Velásquez
// Carné 18822 - Sección 12
// Laboratorio 8
// Ejercicio 3 - ALU

module ALU (input wire [3:0]A, input wire [3:0]B, input wire [2:0]Comando, output reg [3:0]Resultado);

   always @ (A or B or Comando) begin

      case(Comando)
         3'b000: Resultado = A + B ;  // Suma
         3'b001: Resultado = A - B ;  // Resta
         3'b010: Resultado = A & B ;  // AND
         3'b100: Resultado = A | B ;  // OR
         3'b101: Resultado = A & ~B;  // A AND ~B
         3'b110: Resultado = ~A | B;  // ~A or B
         3'b111: Resultado = (A < B) ? 1:0; // // Comparación de A mayor a B
         default: Resultado = 4'b0000 ;
      endcase

   end
endmodule
```

Figura 10 - Módulo ALU

```
// Luis Pedro Molina Velásquez
// Carné 18822 - Sección 12
// Laboratorio 8
// Ejercicio 3 - ALU

module testbench ();

reg [3:0]A ; reg [3:0]B ; reg [2:0]Comando ;
wire [3:0]Resultado ;

ALU alu(A, B, Comando, Resultado);

initial begin

    #1
    $display(" ");
    $display(" ALU ");
    $display(" ");
    $display(" A       B     COM   | RESULTADO");
    $display(" ------------------|-----------");
    $monitor(" %b    %b    %b      %b     ", A, B, Comando, Resultado);
       A = 4'b0000 ; B = 4'b0000 ; Comando = 3'b000 ;
    #1 A = 4'b0001 ; B = 4'b0010 ; Comando = 3'b000 ;  // 1 + 2 = 3
    #1 A = 4'b1000 ; B = 4'b0110 ; Comando = 3'b000 ;  // 8 + 6 = 14
    #1 A = 4'b1111 ; B = 4'b0011 ; Comando = 3'b001 ;  // 15 - 3 = 12
    #1 A = 4'b0011 ; B = 4'b0001 ; Comando = 3'b001 ;  // 3 - 1 = 2
    #1 A = 4'b1011 ; B = 4'b0010 ; Comando = 3'b010 ;  // A & B = 0010
    #1 A = 4'b0110 ; B = 4'b1111 ; Comando = 3'b010 ;  // A & B = 0110
    #1 A = 4'b0001 ; B = 4'b0010 ; Comando = 3'b100 ;  // A | B = 0011
    #1 A = 4'b0000 ; B = 4'b0100 ; Comando = 3'b100 ;  // A | B = 0100
    #1 A = 4'b1010 ; B = 4'b0101 ; Comando = 3'b101 ;  // A &~B = 1010
    #1 A = 4'b1111 ; B = 4'b0010 ; Comando = 3'b101 ;  // A &~B = 1101
    #1 A = 4'b1011 ; B = 4'b0000 ; Comando = 3'b110 ;  // ~A| B = 0100
    #1 A = 4'b1000 ; B = 4'b0001 ; Comando = 3'b110 ;  // ~A| B = 0111
    #1 A = 4'b1000 ; B = 4'b0000 ; Comando = 3'b111 ;  // A < B = 0000
    #1 A = 4'b0010 ; B = 4'b1000 ; Comando = 3'b111 ;  // A < B = 0001

end

    initial
    #15 $finish;

initial
    begin
        $dumpfile("ALU_tb.vcd");
        $dumpvars(0,testbench);
    end

endmodule
```

Figura 11 - Módulo testbench ALU

ALU

A	B	COM	RESULTADO
0000	0000	000	0000
0001	0010	000	0011
1000	0110	000	1110
1111	0011	001	1100
0011	0001	001	0010
1011	0010	010	0010
0110	1111	010	0110
0001	0010	100	0011
0000	0100	100	0100
1010	0101	101	1010
1111	0010	101	1101
1011	0000	110	0100
1000	0001	110	0111
1000	0000	111	0000
0010	1000	111	0001

Figura 12 - Tabla de valores

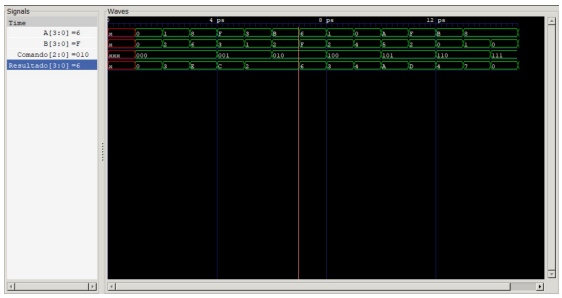


Figura 12 - Diagrama de Timming