Brandon Amisael Garrido Ramírez
Carné: 19421
Sección: 20

Ing. Mecatrónica
Electrónica Digital 1

Laboratorio #8

Enlace al repositorio de github: https://github.com/gar19421/Lab-08-Digital.git

Ejercicio 1:

En este ejercicio consistió un contador el cual contara +1 en cada flanco cuando la señal de enable de entrada estuviera en 1 y que precargara un valor cuando la señal de load estuviera en 1, de forma asincrónica.

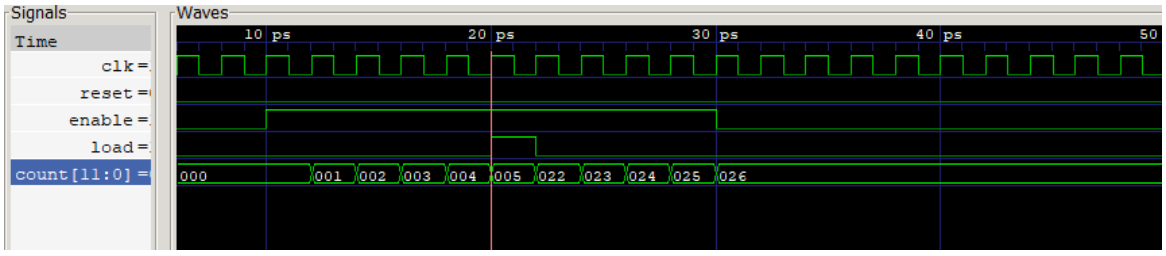
Se colocó dentro de un bloque always para verificar continuamente el valor del reset, el clock, el enable y el load y dependiendo del valor que tomara cada el contador aumentaba en uno cuando el enable estuviera activado, de lo contrario se mantenía en el mismo valor. Precargaba un valor en el contador de 12 bits todo esto se analizaba en cada flanco de reloj.

```
module Counter(input wire clk, reset, enable, load,  output reg[11:0] count);
//contador de 12 bits con entradas de clock, reset, enable y load

  //conteo en cada flanco de reloj
  always @ (posedge clk or posedge reset or posedge enable or posedge load) begin
    if(reset) begin
     count <= 12'd0; // si se resetea empieza en cero
    end
    else if(clk & enable & ~load) begin
        #2 count <= count + 12'd1; // si esta en enable cuenta el contador sino no cuenta
    end
    else if(clk & load & enable) begin
        #2 count <= 12'd34; //precargando el valor
    end
  end

endmodule
```

Como se puede observar en el diagrama de timing el funcionamiento anteriormente descrito funciona adecuadamente, ya que cuando el enable se activa espera para empezar a contar en el siguiente flanco de reloj y va sumando mientras este activado luego se activa load y se puede ver como se carga en el siguiente flanco de reloj un valor en el contador y continúa contando desde allí hasta que el enable se desactiva y deja de contar

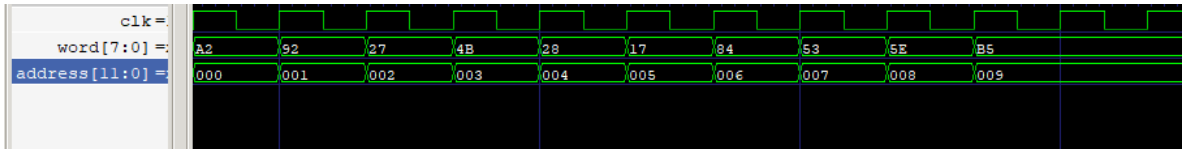


Ejercicio 2:

Este ejercicio consitió en construir en verilog una memoria ROM de la cual se pudiera leer datos de un archivo. Para realizar esto se utilizo el comando $readmemb para leer los datos del archivo memory.list de la memoria de 4kx8 construida, pidiendo una direccion de memoria y devolviendo la informacion contenida dentro de esta. Existe otra variación del comando mencionado anteriormente la cual es $readmemh, la diferencia entre estos dos comandos es que $readmemb sirve para leer archivos con datos en binario mientras que $readmemh para leer datos en hexadecimal.

```
module Memory (input wire [11:0] address, output wire [7:0] word);//memoria rom
  reg [7:0] mem [0:4095];// creacion de arreglo de memoria
  initial begin
      $readmemb("memory.list", mem);//lectura del archivo de datos
  end
  assign word = mem[address];// resultado leido de la memoria
endmodule
```

Que como se puede observar en cada flanco de reloj se obtiene el valor de cada posición de memoria de los 10 datos solicitados para la lectura de la memoria ROM.



Ejercicio 3:

En este ejercicio se implementó la ALU de la sección 5.2.4. del libro con un switch case para realizar las distintas acciones del ALU con base al parámetro de entrada controlado por la variable F y se implemento las operaciones propias del ALU con implementación Behavioral.

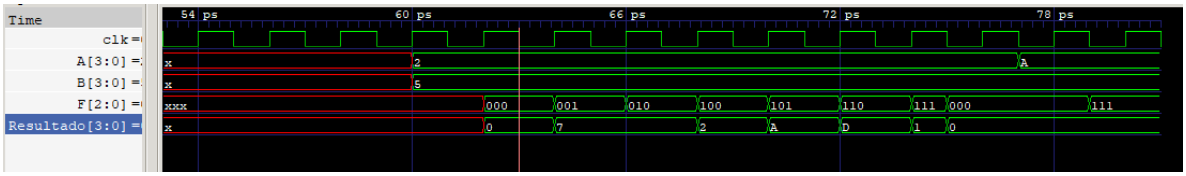
```
module ALU(input wire [3:0] A, B, input wire [2:0] F, output reg [3:0] Resultado);//implementacion de la ALU

// parametros tabla 5.1 libro
  parameter AND1 = 3'd0;
  parameter OR1 = 3'd1;
  parameter ADD = 3'd2;
  parameter AND2 = 3'd4;
  parameter OR2 = 3'd5;
  parameter SUBTRACT = 3'd6;
  parameter SLT = 3'd7;

  always @(*) begin// operaciones a realizar con la ALU con un switch case
    case(F)
    AND1:  assign Resultado = (A & B);
    OR1:   assign Resultado = (A | B);
    ADD:  assign Resultado = A + B;
    AND2: assign Resultado = (A & ~B);
    OR2:  assign Resultado = (A | ~B);
    SUBTRACT: assign Resultado = A - B;
    SLT: assign Resultado = (A > B) ? 4'd0 : 4'd1;
    default: Resultado = 4'd0;
    endcase
  end

endmodule
```

Como se puede observar en el diagrama de timing el resultado obtenido en cada una de las operaciones realizadas es el correcto con las distintas combinaciones de entrada de A y B y también el valor de salida es cero cuando no se realiza ninguna operación.



Otras evidencias(screenshots):

- Pruebas en consola:

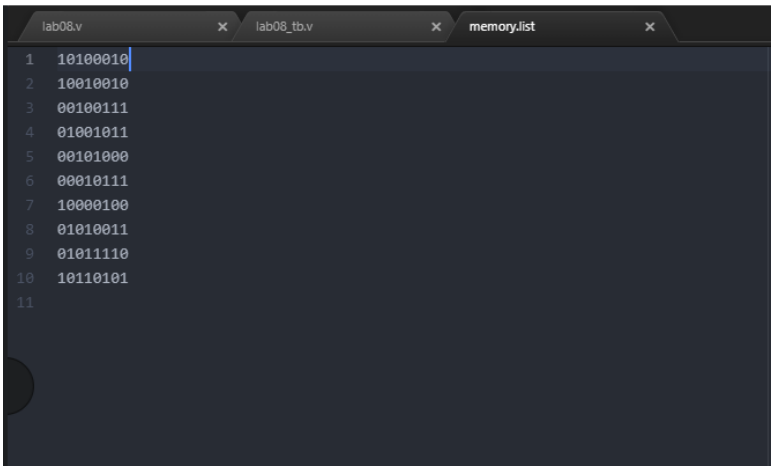
```
CLK Reset | Enable Load | Contador
----------|------------|-------------
VCD info: dumpfile lab08_tb.vcd opened for output.
1     x   |    x     x  | xxxxxxxxxxx
0     x   |    x     x  | xxxxxxxxxxx
1     1   |    x     x  | 000000000000
0     1   |    x     x  | 000000000000
1     0   |    0     0  | 000000000000
0     0   |    0     0  | 000000000000
1     0   |    0     0  | 000000000000
0     0   |    0     0  | 000000000000
1     0   |    0     0  | 000000000000
0     0   |    0     0  | 000000000000
1     0   |    1     0  | 000000000000
0     0   |    1     0  | 000000000000
1     0   |    1     0  | 000000000001
0     0   |    1     0  | 000000000001
1     0   |    1     0  | 000000000010
0     0   |    1     0  | 000000000010
1     0   |    1     0  | 000000000011
0     0   |    1     0  | 000000000011
1     0   |    1     0  | 000000000100
0     0   |    1     0  | 000000000100
1     0   |    1     1  | 000000000101
0     0   |    1     1  | 000000000101
1     0   |    1     0  | 000000100010
0     0   |    1     0  | 000000100010
1     0   |    1     0  | 000000100011
0     0   |    1     0  | 000000100011
1     0   |    1     0  | 000000100100
0     0   |    1     0  | 000000100100
1     0   |    1     0  | 000000100101
0     0   |    1     0  | 000000100101
1     0   |    0     0  | 000000100110
0     0   |    0     0  | 000000100110
1     0   |    0     0  | 000000100110
0     0   |    0     0  | 000000100110
1     0   |    0     0  | 000000100110
0     0   |    0     0  | 000000100110
1     0   |    0     0  | 000000100110
0     0   |    0     0  | 000000100110
```

```
Memoria ROM
Address        |    Word
---------------|-------------
xxxxxxxxxxxx   |   xxxxxxxx
000000000000   |   10100010
000000000001   |   10010010
000000000010   |   00100111
000000000011   |   01001011
000000000100   |   00101000
000000000101   |   00010111
000000000110   |   10000100
000000000111   |   01010011
000000001000   |   01011110
000000001001   |   10110101
```

```
ALU


A        B     | F   |  Y
-----------------------------
0010     0101  | xxx |  xxxx
0010     0101  | 000 |  0000
0010     0101  | 001 |  0111
0010     0101  | 010 |  0111
0010     0101  | 100 |  0010
0010     0101  | 101 |  1010
0010     0101  | 110 |  1101
0010     0101  | 111 |  0001
0010     0101  | 000 |  0000
1010     0101  | 000 |  0000
1010     0101  | 111 |  0000
```

Archivo de datos:



Testbech:


```
initial begin// test de contador
  $display("Contador\n");
  $display("CLK Reset | Enable Load | Contador");
  $display("----------|-------------|--------------");
  $monitor("%b     %b   |    %b     %b  | %b", clock,reset,enable,load,salida);
  #2 reset = 1;
  #2 enable = 0; load = 0; reset = 0;
  #2; #2;
  #2 enable = 1; load = 0;
  #2; #2; #2; #2;
  #2 load = 1;
  #2 load = 0 ;
  #2; #2; #2;
  #2 enable = 0;
  #2; #2; #2;

end
```

```
initial begin//test de memoria
  #38
  $display(" Memoria ROM ");
  $display("Address       |    Word   ");
  $display("--------------|-------------");
  $monitor("%b  |   %b", address, word);

  for (i = 0; i < 10; i++) begin// lectura de los 10 primeros datos de la memoria ROM
    #2 address =  i;
  end
end
```

```
initial begin//test del ALU
    #60
    $display(" \nALU ");
    $display(" A       B    | F   |  Y ");
    $display(" ---------------------------");
    $monitor(" %b    %b  | %b |  %b ", A, B, F, salida_2);

    A[3] = 0; A[2] = 0; A[1] = 1; A[0] = 0; B[3] = 0; B[2] = 1; B[1] = 0; B[0] = 1;
    #2  F[2] = 0; F[1] = 0; F[0] = 0;
    #2  F[2] = 0; F[1] = 0; F[0] = 1;
    #2  F[2] = 0; F[1] = 1; F[0] = 0;
    #2  F[2] = 1; F[1] = 0; F[0] = 0;
    #2  F[2] = 1; F[1] = 0; F[0] = 1;
    #2  F[2] = 1; F[1] = 1; F[0] = 0;
    #2  F[2] = 1; F[1] = 1; F[0] = 1;
    #1  F[2] = 0; F[1] = 0; F[0] = 0;
    #2  A[3] = 1; A[2] = 0; A[1] = 1; A[0] = 0;
    #2  F[2] = 1; F[1] = 1; F[0] = 1;
    #2 $finish;
  end

  initial begin//ejecutar GTKWave para diagramas de timing.
    $dumpfile("lab08_tb.vcd");
    $dumpvars(0, testbench);
  end

endmodule
```