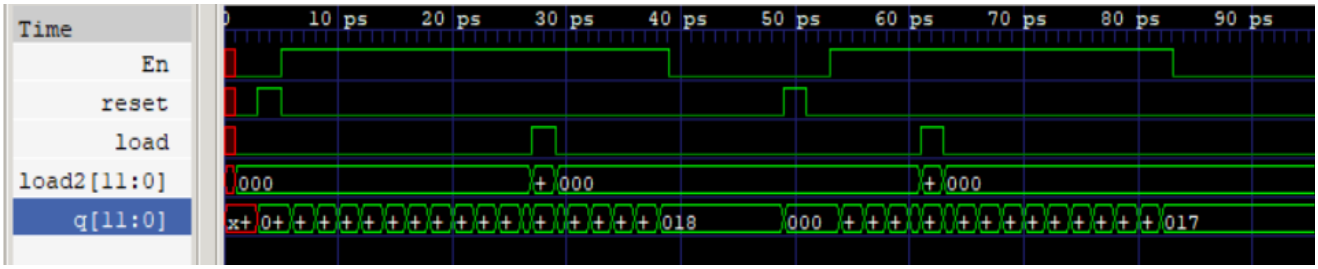
LABORATORIO #8

ELECTRONICA DIGITAL

Ejercicio 1 (CONTADOR)

```
iverilog -o Alejandro-Duarte-19446-seccion21-contador_tb.out -D VCD_OUTPUT=Alejandro-Duar
.v Alejandro-Duarte-19446-seccion21-contador_tb.v
vvp Alejandro-Duarte-19446-seccion21-contador_tb.out
VCD info: dumpfile Alejandro-Duarte-19446-seccion21-contador_tb.vcd opened for output.
reset En load load2 | q
-------------------------
0  0  0   000000000000   | xxxxxxxxxxx
1  0  0   000000000000   | 000000000000
0  1  0   000000000000   | 000000000000
0  1  0   000000000000   | 000000000001
0  1  0   000000000000   | 000000000010
0  1  0   000000000000   | 000000000011
0  1  0   000000000000   | 000000000100
0  1  0   000000000000   | 000000000101
0  1  0   000000000000   | 000000000110
0  1  0   000000000000   | 000000000111
0  1  0   000000000000   | 000000001000
0  1  0   000000000000   | 000000001001
0  1  0   000000000000   | 000000001010
0  1  0   000000000000   | 000000001011
0  1  1   000000010010   | 000000010010
0  1  0   000000000000   | 000000010011
0  1  0   000000000000   | 000000010100
0  1  0   000000000000   | 000000010101
0  1  0   000000000000   | 000000010110
0  1  0   000000000000   | 000000010111
0  1  0   000000000000   | 000000011000
0  0  0   000000000000   | 000000011000
1  0  0   000000000000   | 000000000000
0  0  0   000000000000   | 000000000000
0  1  0   000000000000   | 000000000000
0  1  0   000000000000   | 000000000001
0  1  0   000000000000   | 000000000010
0  1  0   000000000000   | 000000000011
0  1  0   000000000000   | 000000000100
0  1  1   000000001100   | 000000001100
0  1  0   000000000000   | 000000001101
0  1  0   000000000000   | 000000001110
0  1  0   000000000000   | 000000001111
0  1  0   000000000000   | 000000010000
0  1  0   000000000000   | 000000010001
0  1  0   000000000000   | 000000010010
0  1  0   000000000000   | 000000010011
0  1  0   000000000000   | 000000010100
0  1  0   000000000000   | 000000010101
0  1  0   000000000000   | 000000010110
0  1  0   000000000000   | 000000010111
0  0  0   000000000000   | 000000010111
gtkwave Alejandro-Duarte-19446-seccion21-contador_tb.vcd Alejandro-Duarte-19446-seccion21
```



Alejandro-Duarte-19446-seccion21-contador.v

```
module contador(input wire clk,reset,En,load,
input wire [11:0]load2,
output reg [11:0]q);

always @(posedge clk or posedge reset or load or load2) begin
  if(reset)
  q <= 12'b000000000000;
  else if(load)
  q <= load2;
  else if(En)
  q <= q + 1;
  end
endmodule


//jdsjfasdjfjadsfiadsfiodsjfodsjfidssnfiads

```

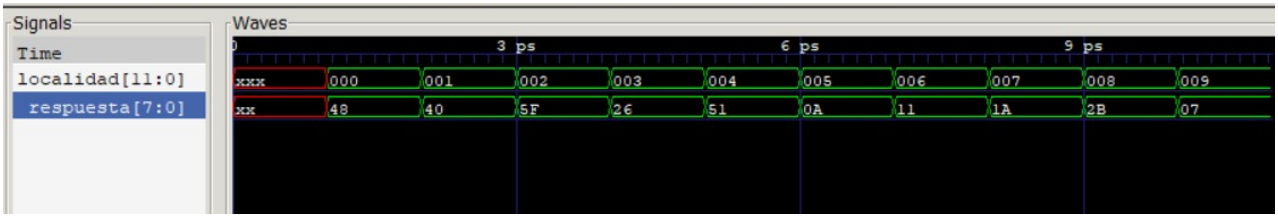
Alejandro-Duarte-19446-seccion21-contador...

```
module testbench();

reg clk,reset,En,load;
reg [11:0]load2;
wire [11:0]q;

contador a(clk,reset,En,load,load2,q);

always
  #1 clk = ~clk;

initial begin
  $display(" reset En load load2 | q");
  $display("-------------------------");
  $monitor("   %b  %b  %b   %b   | %b", reset,En,load,load2, q);
    clk = 0; reset = 0; En = 0; load = 0; load2 = 12'b000000000000;
    #1 reset = 1;
    #1 reset = 0; En = 1;
    #1 reset = 0; En = 1;
    #1 reset = 0; En = 1;
    #1 reset = 0; En = 1;
    #1 reset = 0; En = 1;
    #1 reset = 0; En = 1;
    #1 reset = 0; En = 1;
    #1 reset = 0; En = 1;
    #1 reset = 0; En = 1;
    #1 reset = 0; En = 1;
```

Para el modulo del contador se establecieron 4 variables de 1 bit, en los cuales está el clk, reset, En y el load, también se determinan dos variables con buses de 12 bits (load2 y q). En este módulo se utiliza always con nonblocking para que trabaje en paralelo todas las condiciones implementadas, entre estas esta la condición que cuando load se active se cargaría el valor de load2 a la salida q del contador, también están las condiciones de cuando el ennable suma 1 bit cada tiempo de reloj y cuando se activa el reset vuelvo a valores iniciales la salida q.

Ejercicio 2 (MEMORIA ROM)

```
Localidad    | Respuesta
-------------------------
000000000000 | 01001000
000000000001 | 01000000
000000000010 | 01011111
000000000011 | 00100110
000000000100 | 01010001
000000000101 | 00001010
000000000110 | 00010001
000000000111 | 00011010
000000001000 | 00101011
000000001001 | 00000111
gtkwave Alejandro-Duarte-19446-21-memoria_tb.vcd Alejandro-Duarte-19446-21-memoria_tb.gtkw
```



Alejandro-Duarte-19446-21-memoria.v

```
module memoriaROM(input wire [11:0]localidad, output wire [7:0]respuesta);

reg [7:0] cerebro [0:4095];

  initial begin
    $readmemb("memory.list", cerebro);
  end

assign respuesta = cerebro[localidad];

endmodule
```

Alejandro-Duarte-19446-21-memoria_tb.v

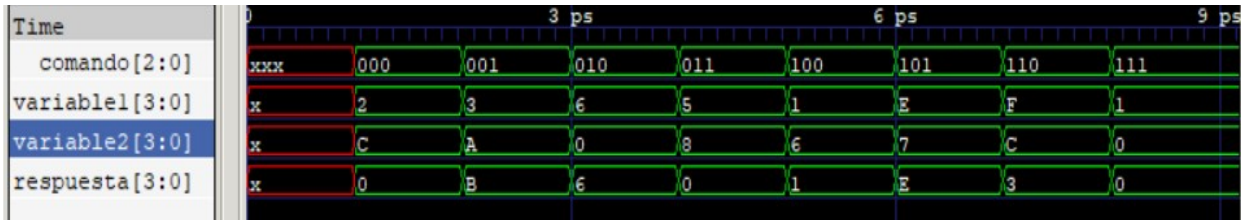
```
module testbench();

reg [11:0]localidad;
wire [7:0]respuesta;

memoriaROM b(localidad,respuesta);

initial begin
  #1
  $display("   Localidad   | Respuesta");
  $display("-------------------------");
  $monitor("   %b | %b", localidad, respuesta);
    localidad = 12'b000000000000;
  #1localidad = 12'b000000000001;
  #1localidad = 12'b000000000010;
  #1localidad = 12'b000000000011;
  #1localidad = 12'b000000000100;
  #1localidad = 12'b000000000101;
  #1localidad = 12'b000000000110;
  #1localidad = 12'b000000000111;
  #1localidad = 12'b000000001000;
  #1localidad = 12'b000000001001;

end

initial
  #11 $finish;

initial begin
  $dumpfile("Alejandro-Duarte-19446-21-memoria_tb.vcd");
```

Para el modulo de memoria se implementa un arreglo de 8 bits por, aproximadamente, cuatro mil localidades. En este lee el documento nombrado memory.list que tiene números binarios de 8 bits que pasan a la variable cerebro que este en el arreglo establecido para luego establecer los datos del archivo externo en la variable respuesta de 8 bits.

Ejercicio 3 (ALU)

Comando	Variable1	Varaiable2	Respuesta
000	0010	1100	0000
001	0011	1010	1011
010	0110	0000	0110
011	0101	1000	0000
100	0001	0110	0001
101	1110	0111	1110
110	1111	1100	0011
111	0001	0000	0000

```
gtkwave Alejandro-Duarte-19446-21-ALU_tb.vcd Alejandro-Duarte-19446-21-ALU_tb.gtkw
```



Alejandro-Duarte-19446-21-ALU.v

```
module ALU(
  input wire [3:0]variable1,
  input wire [3:0]variable2,
  input wire [2:0]comando,
  output reg [3:0]respuesta
  );
  always @ (variable1,variable2,comando) begin
    case(comando)
        3'b000: respuesta = variable1 & variable2;
        3'b001: respuesta = variable1 | variable2;
        3'b010: respuesta = variable1 + variable2;
        3'b011: respuesta = 4'b0000;
        3'b100: respuesta = variable1 & ~variable2;
        3'b101: respuesta = variable1 | ~variable2;
        3'b110: respuesta = variable1 - variable2;
        3'b111: respuesta = (variable1 < variable2) ? 1:0;
        default: respuesta = 4'b0000;
    endcase
  end
endmodule
```

Alejandro-Duarte-19446-21-ALU_tb.v

```
module testbench();

reg [3:0]variable1;
reg [3:0]variable2;
reg [2:0]comando;
wire [3:0]respuesta;

ALU c(variable1,variable2,comando,respuesta);

initial begin
  #1
  $display("|---------|-----------|------------|-----------|");
  $display("| Comando | Variable1 | Varaiable2 | Respuesta |");
  $display("|---------|-----------|------------|-----------|");
  $monitor("|   %b    |    %b     |     %b     |    %b     |", comando,variable1,varia
       comando = 3'b000; variable1 = 4'b0010; variable2 = 4'b1100;
    #1 comando = 3'b001; variable1 = 4'b0011; variable2 = 4'b1010;
    #1 comando = 3'b010; variable1 = 4'b0110; variable2 = 4'b0000;
    #1 comando = 3'b011; variable1 = 4'b0101; variable2 = 4'b1000;
    #1 comando = 3'b100; variable1 = 4'b0001; variable2 = 4'b0110;
    #1 comando = 3'b101; variable1 = 4'b1110; variable2 = 4'b0111;
    #1 comando = 3'b110; variable1 = 4'b1111; variable2 = 4'b1100;
    #1 comando = 3'b111; variable1 = 4'b0001; variable2 = 4'b0000;

end
initial
  #10 $finish;

initial begin
  $dumpfile("Alejandro-Duarte-19446-21-ALU_tb.vcd");
```

Para este modulo se implemento la ALU que esta descrita en el libro, en el cual se implentan 8 diferentes comandos que se le pueden ver en el código. Para la realización de este se implementaron tres variables de buses de 4 bits y una variable de 3 bits que será la que determine que comando se utilizará para efectuar sobre las variables 1 y 2 para que dicho resultado se guarde en la variable respuesta.