Participación 5

La actividad se realizó en equipo, nos tocó programar la ALU con bloqueo y sin bloqueo. La asignación con bloqueo se ejecuta de forma secuencial, indicando que la siguiente instrucción no se ejecutará hasta que la asignación actual haya sido completada. Dentro del código, se puede observar con el signo de = solamente, mientras la que sin bloqueo utiliza <=. La asignación sin bloqueo se ejecuta de manera combinacional, significa que la variable no se actualiza inmediatamente, sino que se programa para actualizarse en el próximo ciclo de simulación.

ALU con Bloqueo:

C:\Users\1234\Documents\Trabajos Arqui\ALU\ALUB.v - Default

```
Ln#
 1
 2  module AluB(
 3      input [31:0] A,
 4      input [31:0] B,
 5      input CLK,
 6      input [2:0] op,
 7      output reg [31:0] Res,
 8      output reg Zflag
 9  );
10
11  always @(posedge CLK) begin
12      case(op)
13          3'b000: Res = A + B;       // Suma
14          3'b001: Res = A & B;       // AND
15          3'b010: Res = A | B;       // OR
16          3'b011: Res = A - B;       // Resta
17          3'b100: Res = A * B;       // Multiplicación
18          3'b111: Res = (A < B) ? 32'd1 : 32'd0;  // Ternaria
19
20          default: Res = 32'd0;
21      endcase
22
23      Zflag = (Res == 32'd0) ? 1'b1 : 1'b0;
24  end
25
26  endmodule
```

ALU sin Bloqueo:

C:\Users\1234\Documents\Trabajos Arqui\ALU\ALUNB.v - Default

```
Ln#
 1  module AluNB(
 2      input [31:0] A,
 3      input [31:0] B,
 4      input CLK,
 5      input [2:0] op,
 6      output reg [31:0] Res,
 7      output reg Zflag
 8  );
 9
10  initial begin
11      Res <= 32'd0;
12      Zflag <= 1'b0;
13  end
14
15  always @(posedge CLK or op) begin
16      case (op)
17          3'b000: Res <= A + B;
18          3'b001: Res <= A & B;
19          3'b010: Res <= A | B;
20          3'b011: Res <= A - B;
21          3'b100: Res <= A * B;
22          3'b101: Res <= (A < B) ? 32'd1 : 32'd0;
23          default: Res <= 32'd0;
24      endcase
25
26      if (Res == 32'd0)
27          Zflag <= 1'b1;
28      else
29          Zflag <= 1'b0;
30  end
31
32  endmodule
```

Testbench

C:\Users\1234\Documents\Trabajos Arqui\ALU\Testbench.v - Default

```
Ln#
 1  module ALU_TB();
 2      reg [31:0] ATB, BTB;
 3      reg CLKTB;
 4      reg [2:0] SELTB;
 5      wire [31:0] RTB_NB, RTB_B;
 6      wire ZFLAGTB_NB, ZFLAGTB_B;
 7
 8
 9      // Instanciamos la ALU con bloqueo (ALUB)
10      AluB alu_b ( .A(ATB), .B(BTB), .op(SELTB), .Res(RTB_B), .CLK(CLKTB), .Zflag(ZFLAGTB_B));
11
12      // Instanciamos la ALU sin bloqueo (ALUNB)
13      AluNB alu_nb ( .A(ATB), .B(BTB), .op(SELTB), .Res(RTB_NB), .CLK(CLKTB), .Zflag(ZFLAGTB_NB));
14
15
16      always #50 CLKTB = ~CLKTB;
17
18      initial begin
19          CLKTB = 0; // Inicializa el reloj
20          ATB = 32'd300;
21          BTB = 32'd100;
22
23          SELTB = 3'b000; #100;  // Suma
24          SELTB = 3'b001; #100;  // AND
25          SELTB = 3'b010; #100;  // OR
26          SELTB = 3'b011; #100;  // Resta
27          SELTB = 3'b100; #100;  // Multiplicacion
28          SELTB = 3'b101; #100;  //Ternaria
29
30
31          $stop;
32      end
33  endmodule
```

Wave