Parcial 1

Indice =

Solución

1) Código de implementación

```
module dual_alu_16 (
    input [31:0] A, B, C,   // Entradas de 32 bits
    output [31:0] S,        // Resultado de 32 bits
    output cout             // Acarreo final
);
    wire [15:0] S_L, S_H;
    wire Cout_L, Cout_H;

    // ALU baja (parte de 16 bits)
    alu_16 ALU_16_L (
        .A(A[15:0]),
        .B(B[15:0]),
        .C(C[15:0]),
        .cin(1'b0),
        .S(S_L),
        .cout(Cout_L)
    );

    // ALU alta (parte de 16 bits), sumando el acarreo de la ALU baja
    alu_16 ALU_16_H (
        .A(A[31:16]),
        .B(B[31:16]),
        .C(C[31:16]),
        .cin(Cout_L),
        .S(S_H),
        .cout(Cout_H)
    );

    assign S = {S_H, S_L};
    assign cout = Cout_H; // Acarreo final
endmodule
```

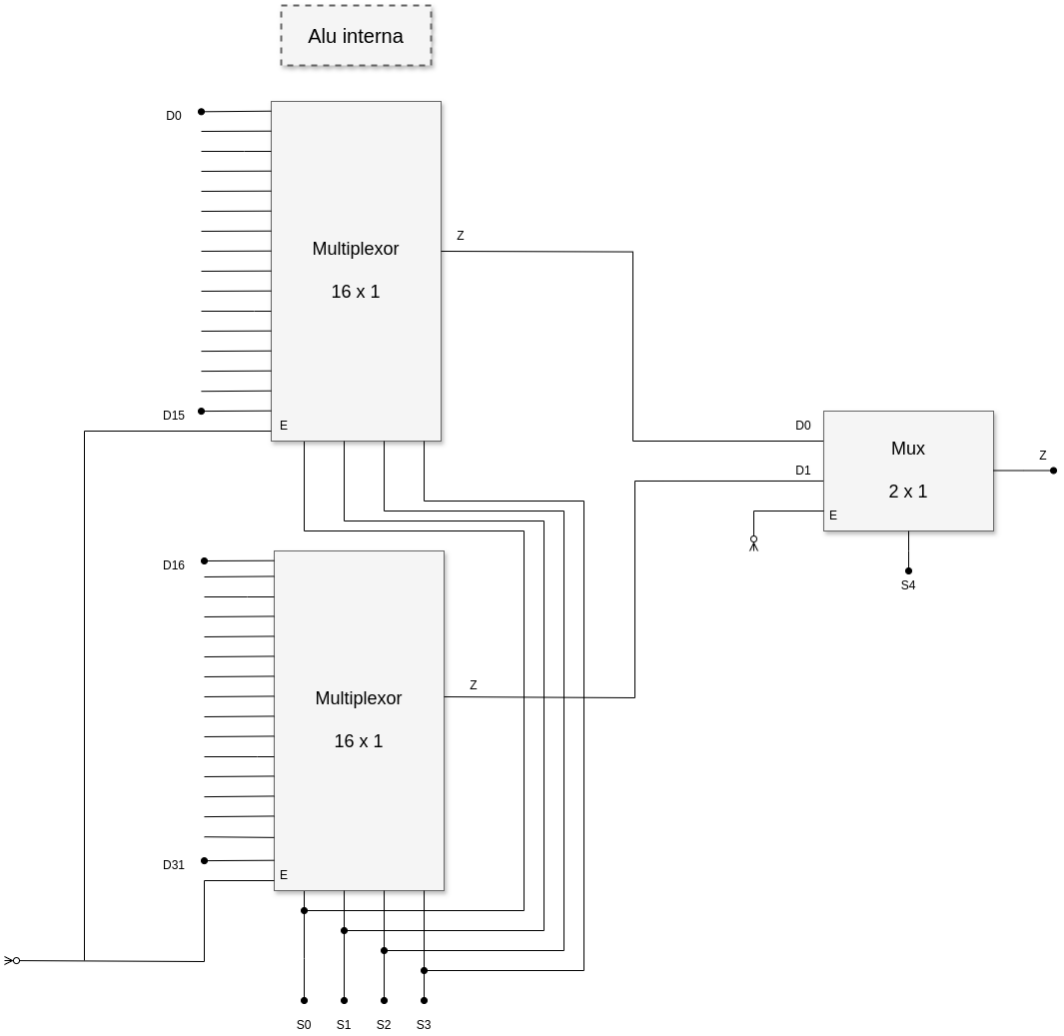
2) Explicación de código

En la operación de 32 bits se divide en 2, de [0, 15] de los 32 bits y lo llamamos parte baja de la ALU, la otra parte de [16,31] de los 32 bits faltantes y lo llamamos parte alta de la ALU.

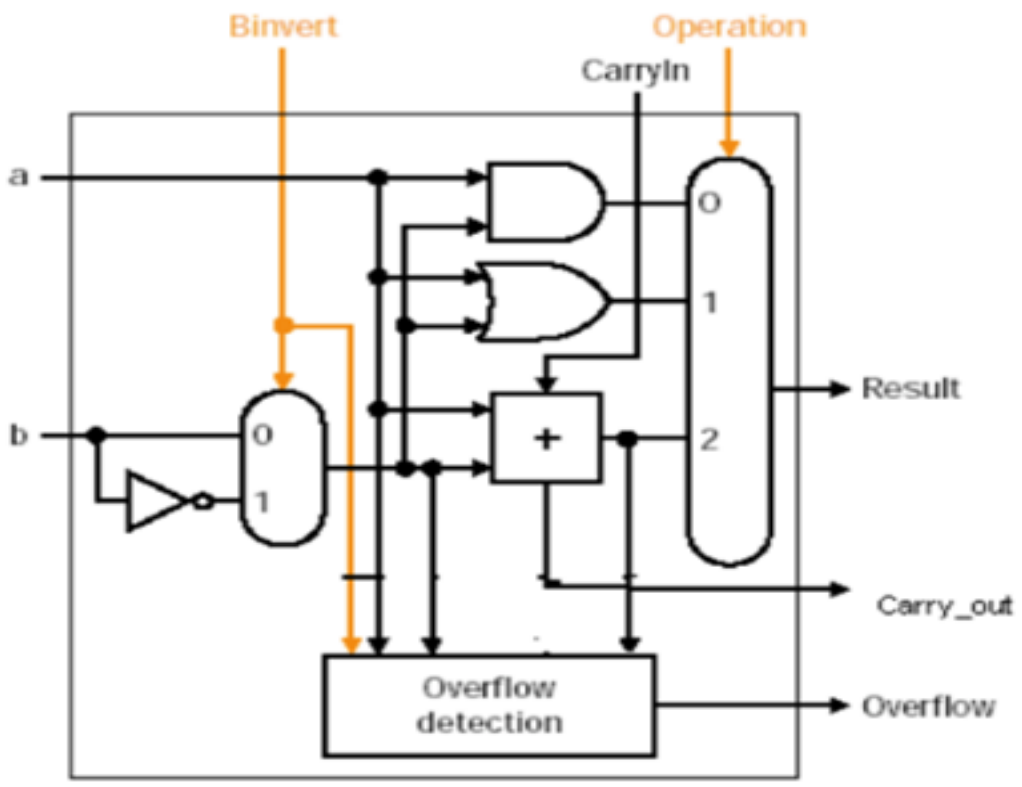
Después de la dividirlo, realizamos la suma o resta de la parte baja, luego se crea el acarreo de la parte baja donde guardamos la operación en un espacio de memoria que en la segunda parte alta tome ese valor para ejecutarlo en la siguiente operación, luego realizamos la suma o resta de la parte alta, al final ensamblamos los resultados y donde miramos y juntamos los 2 resultados con un overflow para mostrar el resultado final de la operación con un numero de 32 bits nuevamente.

Cada operando de 32 bits (A, B, C) se divide en dos partes (A_H | A_L).
Se realiza la suma de 16 bits de las partes bajas.
Si hay acarreo (Cout_L), se suma en la siguiente etapa.
Se realiza la suma de 16 bits de las partes altas + Cout_L.
Se ensamblan los resultados (S = S_H | S_L).
Si hay un Cout_H, significa que hubo overflow en la suma de 32 bits.

3) Diagrama de bloque



4) Diseño de la ALU



5) Tabla de verdad

c	b	a	f(a,b,c)
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

6) Lógica de control

La lógica implementada para el control de las operaciones fue pensando en la primera idea de leer el bit completo y hacer o mostrar la respuesta al final, primero fue la creación de partir el bit en 2, uno con la operación inicial y luego la segunda partición toma el resultado de la primera operación, para controlar la operación hacemos como un controlador donde nos permita primero hacer las operaciones sin que nos haga la segunda operación ya que no tenemos el resultado de la primera operación, luego al final teniendo el resultado 1 y el 2, hacemos un MUX para permitirnos sacar los 2 resultados, unificarlos para sacar el resultado de nuevo con los 32 bits.