30 Days of RTL Coding

-By T Tushar Shenoy

Day 23

Problem Statement: Implementing 4-bit ALU (Arithmetic Logic Unit) in Behavioural Style of Modelling.

Theory:

An ALU, or Arithmetic Logic Unit, is a fundamental component of a central processing unit (CPU) or microprocessor in a computer. It is responsible for performing arithmetic and logic operations on binary data. The primary functions of an ALU include:

1. Arithmetic Operations:
    - Addition: Adding two binary numbers to produce a sum.
    - Subtraction: Subtracting one binary number from another to produce a difference.
    - Multiplication: Multiplying two binary numbers to produce a product.
    - Division: Dividing one binary number by another to produce a quotient and a remainder.
2. Logic Operations:
    - AND: Performing a bitwise AND operation on two binary numbers. The result is 1 if both corresponding bits are 1; otherwise, it's 0.
    - OR: Performing a bitwise OR operation on two binary numbers. The result is 1 if at least one corresponding bit is 1.
    - XOR (Exclusive OR): Performing a bitwise XOR operation on two binary numbers. The result is 1 if the corresponding bits are different; otherwise, it's 0.
    - NOT: Performing a bitwise NOT operation on a binary number. It flips each bit (1 becomes 0, and 0 becomes 1).

3. Comparison Operations:
    - Equal: Comparing two binary numbers to check if they are equal.
    - Not Equal: Checking if two binary numbers are not equal.
    - Less Than: Determining if one binary number is less than another.
    - Greater Than: Determining if one binary number is greater than another.

Only Arithmetic Operation has been implemented.

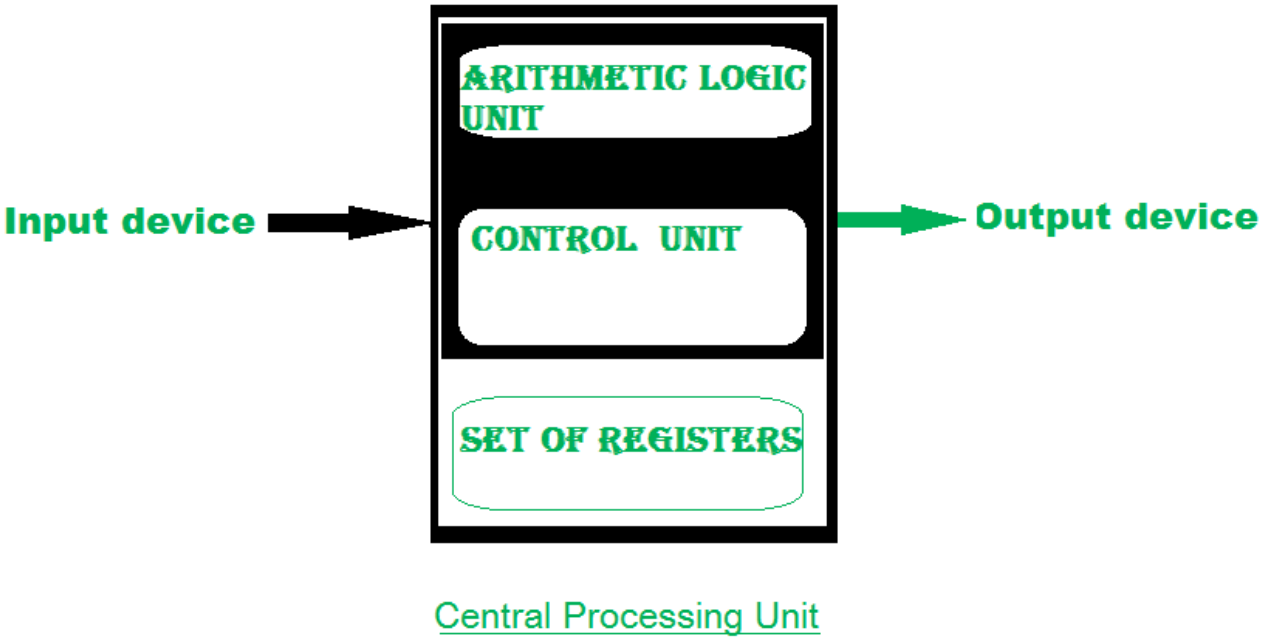


FIG: Block Diagram of Central Processing Unit

Verilog Code:

```
//Verilog Code for ALU
module alu(A, B, select, zero, carry, sign, parity, overflow, out);

  input [3:0] A, B;
  input [1:0] select;

  output reg [3:0] out;
  output reg zero, carry, sign, parity, overflow;

  always @ (A or B)
  begin
    if (select == 0) // Addition
    begin
      {carry,out} = A + B;

    end
    else if (select == 1) // Subtraction
    begin
      {carry,out} = A - B;

    end
    else if (select == 2) // Multiplication
    begin
      {carry,out} = A * B;

    end
```

```
    else if (select == 3) // Division (if A|B != 0)
    begin
      if (~(A | B) == 0)
      begin
        {carry,out} = A / B;


      end
    end

    zero = ~|out;
    sign = out[3];
    parity = ~^out;
    overflow = (A[3] & B[3] & ~out[3]) | (~A[3] & ~B[3] & out[3]);

  end
endmodule
```

Testbench Code:

```
//Testbench Code for ALU
module alu_tb;

reg [3:0] A, B;
reg [1:0] select;
wire [3:0] out;
wire zero, carry, sign, parity, overflow;
```

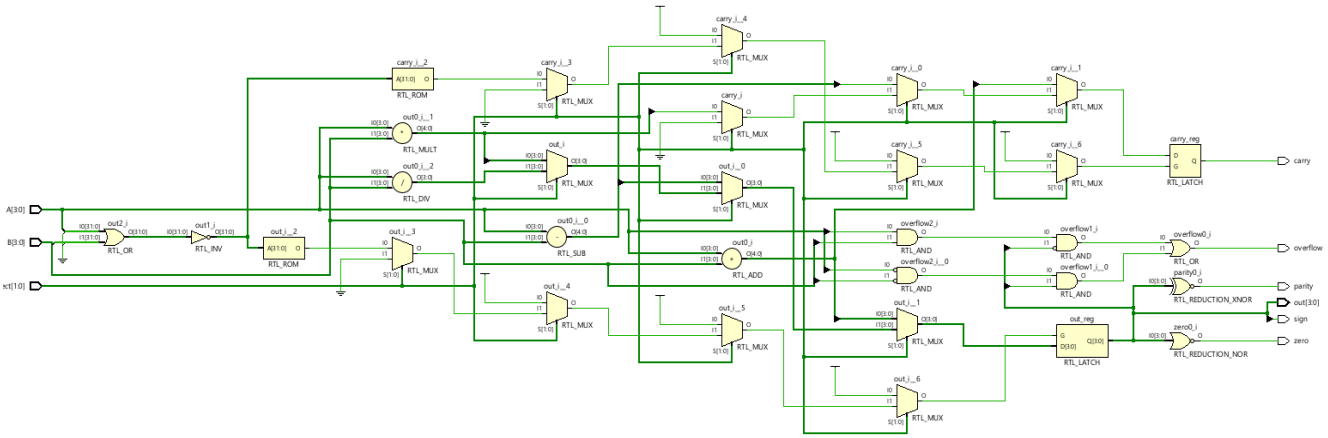
```
alu dut
(.A(A),.B(B),.select(select),.zero(zero),.carry(carry),.sign(sign),.parity(parity
),.overflow(overflow),.out(out));


initial begin
  repeat (100)
  begin
    stimulus();
    #10$display("A = %b, B = %b, select = %b, out = %b, zero = %b, carry =
%b, sign = %b, parity = %b, overflow = %b", A, B, select, out, zero, carry,
sign, parity, overflow);
  end
  $finish;
end


task stimulus;
begin
  A = $random;
  B = $random;
  select = $random;
end
endtask


endmodule
```

Schematic:

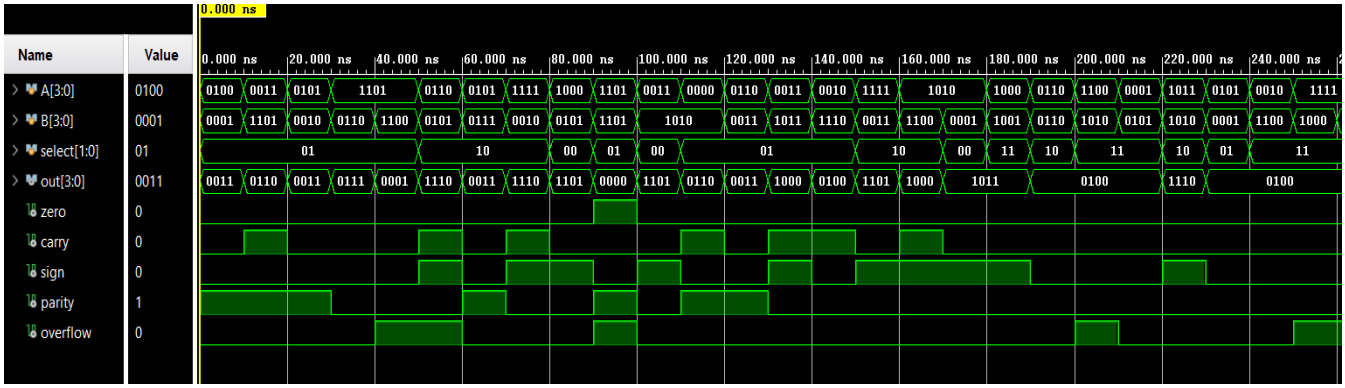


Console Output:

Tcl Console × Messages Log

```
A = 0100, B = 0001, select = 01, out = 0011, zero = 0, carry = 0, sign = 0, parity = 1, overflow = 0
A = 0011, B = 1101, select = 01, out = 0110, zero = 0, carry = 1, sign = 0, parity = 1, overflow = 0
A = 0101, B = 0010, select = 01, out = 0011, zero = 0, carry = 0, sign = 0, parity = 1, overflow = 0
A = 1101, B = 0110, select = 01, out = 0111, zero = 0, carry = 0, sign = 0, parity = 0, overflow = 0
A = 1101, B = 1100, select = 01, out = 0001, zero = 0, carry = 0, sign = 0, parity = 0, overflow = 1
A = 0110, B = 0101, select = 10, out = 1110, zero = 0, carry = 1, sign = 1, parity = 0, overflow = 1
A = 0101, B = 0111, select = 10, out = 0011, zero = 0, carry = 0, sign = 0, parity = 1, overflow = 0
A = 1111, B = 0010, select = 10, out = 1110, zero = 0, carry = 1, sign = 1, parity = 0, overflow = 0
A = 1000, B = 0101, select = 00, out = 1101, zero = 0, carry = 0, sign = 1, parity = 0, overflow = 0
A = 1101, B = 1101, select = 01, out = 0000, zero = 1, carry = 0, sign = 0, parity = 1, overflow = 1
A = 0011, B = 1010, select = 00, out = 1101, zero = 0, carry = 0, sign = 1, parity = 0, overflow = 0
A = 0000, B = 1010, select = 01, out = 0110, zero = 0, carry = 1, sign = 0, parity = 1, overflow = 0
A = 0110, B = 0011, select = 01, out = 0011, zero = 0, carry = 0, sign = 0, parity = 1, overflow = 0
A = 0011, B = 1011, select = 01, out = 1000, zero = 0, carry = 1, sign = 1, parity = 0, overflow = 0
A = 0010, B = 1110, select = 01, out = 0100, zero = 0, carry = 1, sign = 0, parity = 0, overflow = 0
A = 1111, B = 0011, select = 10, out = 1101, zero = 0, carry = 0, sign = 1, parity = 0, overflow = 0
A = 1010, B = 1100, select = 10, out = 1000, zero = 0, carry = 1, sign = 1, parity = 0, overflow = 0
A = 1010, B = 0001, select = 00, out = 1011, zero = 0, carry = 0, sign = 1, parity = 0, overflow = 0
A = 1000, B = 1001, select = 11, out = 1011, zero = 0, carry = 0, sign = 1, parity = 0, overflow = 0
A = 0110, B = 0110, select = 10, out = 0100, zero = 0, carry = 0, sign = 0, parity = 0, overflow = 0
A = 1100, B = 1010, select = 11, out = 0100, zero = 0, carry = 0, sign = 0, parity = 0, overflow = 1
```

Simulation Output:



GitHub Repository URL:https://github.com/tusharshenoy/RTL-Day-23-ALU