```vhdl
library ieee;
use ieee.std_logic_1164.all;
use IEEE.numeric_std.ALL;
use std.textio.all;

Entity EN_ExecUnit is
    Generic ( N : natural := 64 );
    Port ( A, B : in std_logic_vector( N-1 downto 0 );
    FuncClass, LogicFN, ShiftFN : in std_logic_vector( 1 downto 0 );
    AddnSub, ExtWord : in std_logic := '0';
    Y : out std_logic_vector( N-1 downto 0 );
    Zero, AltB, AltBu : out std_logic );
End Entity EN_ExecUnit;



-- Architecture using ripple-carry adder, and a barrel shifter
architecture RTL of EN_ExecUnit is
    -- component declrations

    component EN_Logic is
        generic (N : natural := 64);
        port (
            A, B : in std_logic_vector(N-1 downto 0);
            LogicFN : in std_logic_vector(1 downto 0);
            Y : out std_logic_vector(N-1 downto 0)
        );
    end component;
    component EN_Shift is
        generic (N : natural := 64);
        port (
            A : in std_logic_vector(N-1 downto 0);
            ShiftCount: in std_logic_vector(5 downto 0);
            Y_LL, Y_RL, Y_RA : out std_logic_vector(N-1 downto 0)
        );
    end component;
    component EN_Adder is
        generic (N : natural := 64);
        port (
            A, B  : in std_logic_vector (N-1 downto 0);
            S : out std_logic_vector (N-1 downto 0);
            Cin : in std_logic;
            Cout, Ovfl : out std_logic
        );
    end component;

    -- intermediate signals
    signal Y_logic, Y_slt, Y_sltu, Y_ShiftOrArith : std_logic_vector (N-1 downto 0);
    signal B_adder,S : std_logic_vector (N-1 downto 0);
    signal Cout, Ovfl : std_logic;
    signal Y_LL, Y_RL, Y_RA, Y_LorS, Y_R, Y_LorS_Ext, Y_R_Ext : std_logic_vector (N-1 downto 0);
    constant N_half : integer := N / 2;
    begin
    -- final value selection mux
    with std_logic_vector(FuncClass) select
        Y <= Y_ShiftOrArith when "00",
             Y_logic        when "01",
             Y_slt          when "10",
             Y_sltu         when "11",
             (others => 'X')when others;

    -- see if were adding or subtracting and choose Correct B
    B_adder <= B when AddnSub = '0' else not B;

    -- Shift muxes
    Y_LorS          <= S            when ShiftFN(0) = '0'   else Y_LL;
    Y_R             <= Y_RL         when ShiftFN(0) = '0'   else Y_RA;
    Y_LorS_Ext      <= Y_LorS       when ExtWord = '0'      else (N-1 downto N_half => Y_LorS(N_half - 1)) & Y_LorS(N_half-1 downto 0);
    Y_R_Ext         <= Y_R          when ExtWord = '0'      else (N-1 downto N_half => Y_R(N_half - 1)) & Y_R(N_half-1 downto 0);
    Y_ShiftOrArith  <= Y_LorS_Ext   when ShiftFN(1) = '0'   else Y_R_Ext;

    -- Adding Subsystem
    Add : EN_Adder
        generic map (N => N)
        port map (
            A    => A,
            B    => B_adder,
            S    => S,
            Cin  => AddnSub,
            Cout => Cout,
            Ovfl => Ovfl
        );

    -- Output flags
    AltB        <= Ovfl xor S(N-1);
    Y_slt   <= (N-1 downto 1 => '0' ) & AltB;
    AltBu   <= not Cout;
    Y_sltu  <= (N-1 downto 1 => '0' ) & AltBu;
    Zero        <= '1' when unsigned(S) = 0 else '0';

    -- Logic Subsystem
    Logic : entity work.EN_Logic(RTL)
        generic map (N => N)
        port map (
            A           => A,
            B           => B,
            Y           => Y_logic,
            LogicFN     => LogicFN
        );

    -- Shift Subsystem
    Shift : EN_Shift
        generic map (N => N)
        port map (
            A               => A,
            ShiftCount      => B (5 downto 0),
            Y_LL            => Y_LL,
            Y_RL            => Y_RL,
            Y_RA            =>Y_RA
        );

end RTL;

```