Лабораторна робота 6
Проектування арифметико-логічного пристрою з використанням бібліотеки IEEE

1 **Мета лабораторної роботи**:
оволодіти знаннями і навичками по проектуванню арифметико-логічних пристроїв (АЛП) для сучасних компьютерів. Навчитись використовувати функції і типи з пакетів STD_LOGIC_ARITH і STD_LOGIC_SIGNED бібліотеки IEEE.

Теоретичні відомості

В пакеті std_logic_arith визначені типи unsigned і signed як вектори елементів типу std_logic і підтип small_int типу integer, а також ряд арифметичних функцій, функцій порівняння і зсуву над операндами цих типів, які перезавантажують однойменні функції і операції, які визначені над цілими числами.

Значення типів std_logic_vector, unsigned і signed можуть бути порозрядно рівні між собою. Але для операнда unsigned перезавантажується така функція, що обробляє цей операнд як код числа без знака. Тобто тип unsigned означає позитивні двійкові цілі числа. Аналогічно тип signed означає двійкові цілі числа зі знаком в доповнюючому коді.

В пакеті визначені перезавантажувані арифметичні функції '+', '-','*',**abs**, а також всі функції порівняння для операндів типу unsigned, signed, integer в різних комбінаціях. Причому вектори операндів можуть мати різні діапазони. Наприклад, якщо об′явлені:

```
constant A:signed(4 downto 0) :="11100";  --число -4
constant B:unsigned(3 downto 1):="101";   --число 5
```

то результати різних функцій будуть наступні:

```
A+7 = "00011", A+B = "00001", A-B = "10111", B*3 = "01111",
(A=B) = false, (A<B) = true.
```

Функція SHR виконує зсув вправо без урахування знака операнда типу unsigned або з урахуванням знака операнда типу signed на кількість розрядів, яка задана другим операндом типу unsigned. Аналогічно функція SHL виконує зсув вліво. Наприклад, для констант з попереднього прикладу:

```
SHR(A,B-4) = "11110",    SHL(A,B-3) = "10000".
```

Функція EXT додає до операнда типу std_logic_vector зліва стільки нулів, щоб в результуючому векторі було загальне число розрядів, яке дорівнює другому операнду типу integer. Аналогічно функція SXT додає зліва розряди, які дорівнюють старшому – знаковому розряду операнда. Наприклад:

```
EXT("10",5) = "00010",      SXT("10",5) = "11110".
```

Для взаємного перетворення типів визначені наступні функції. Функція Conv_std_logic_vector пертворює операнд типу integer, unsigned, signed, std_ulogic в результат типу std_logic_vector з числом розрядов, яке задане другим операндом типу integer. Функції Conv_integer, Conv_unsigned і Conv_signed пертворюють такі самі аргументи в результати типу integer, unsigned і signed, відповідно. Наприклад, вищеприведені константи A і B можуть бути об′явлені як:

```
constant A:signed(4 downto 0):= Conv_signed (-4);   --число -4
constant B:unsigned(3 downto 1):= Conv_unsigned (5); --число 5
```

Так як signed, unsigned і std_logic_vector для векторів однакової довжини є тісно зв′язаними типами, то перетворення типів таких векторів виконується як перехід типу. Ті самі константи A і B можуть бути об′явлені як:

```
constant A:signed(4 downto 0):=signed(SXT("100",5));--число - 4
constant B:unsigned(3 downto 1):=unsigned(1-A(2 downto 0));--число 5
```

Пакети std_logic_signed і std_logic_unsigned вміщують, в основному, такі самі функції, як і пакет std_logic_arith, але вони відносяться до операндів типу integer і std_logic_vector. Останній приймається або як число зі знаком, або як число без знака відповідно до назви пакета. Ци пакети використовують в тіх об′єктах проекта, в яких задіяна арифметика або тільки чисел зі знаком (напр. АЛП), або тільки чисел без знаку (напр, лічильники).

2. Завдання для лабораторної роботи:

– розробити функціональну схему АЛП, що виконує задані функції;

– змоделювати роботу АЛП.

Результати виконання оформлюються у вигляді звіту (протоколу). Звіт півинен вміщувати:

- опис заданого варіанта АЛП,
- хід проектування і схему АЛП,
- графіки сигналів, знятих при іспитах АЛП,
- висновки.

В усіх варіантах завдань необхідно обчислити деяку функцію Y=F(A,B), біт Z – ознаку нульового результата і біт C_N – перенос із старшого розряду.

Варіант завдання вибирається за номером студента в списку групи з наступної таблиці

№ завдання	Функції Y в залежності від F	№ завдання	Функції Y в залежності від F
1	AND, OR, NAND, ADD, INC, SRA	12	OR, NAND, ADD, ADDC, SRA, INC
2	NOT, AND, XOR, ADD, SUB, SRA	13	AND, XOR, ADD, INC, SRA, DEC
3	NOT, AND, OR, XOR, ADD, SUB	14	AND, XOR, ADD, ADDC, SUB,SRL
4	NOT, AND, XOR, ADD, INC, SRA	15	AND, XOR, ADD, SUB, SRL,INC
5	NOT,AND,ABS, ADD, SUB, SLA	16	AND,ABS,ADDC, INC,SUB, SLA
6	NOT, XOR, ADD, ADDC, SUB, SLA	17	AND, ADD, SUB, SUBB,SRL, INC
7	AND, XOR, ADD, SUB, SUBB, SLA	18	AND, XOR, ADD, SUB, SUBB,SLA
8	OR, ADD, ADDC, SUB, SLA, INC	19	AND, ADD, ADDC, SUB, SRL, DEC
9	NAND,XOR, ADD, ADDC, SUB, SRA	20	OR, XOR, ADD, ADDC, SUB, SRL
10	AND, OR, ADD, ADDC, SUB, SRA	21	AND, NAND, ADD, ADDC, SUB, SRL
11	AND, NOR, ADD, ADDC, SUB, SRA	22	AND, XOR, ADD, ADDC, SUB, SRL

Примітки:

1.В варіантах 1-11 розрядність АЛП N=12, в варіантах 12-21 розрядність АЛП N=16.

2.Функції SRA, SRL, SLA – зсув вправо (R) і вліво (L) арифметичний (SRA) і логічний (SRL) першого операнду на кількість розрядів, яка задається молодшими розрядами другого операнду.

3. Функції інверсії NOT, ABS відносяться до першого операнду.

4. Функції ADDC і SUBB відповідають додаванню з переносом і відніманню з позичкою, які приходять на вхід C_0.

3. **Приклад виконання роботи**

Розглянемо приклад проектування n = 8 –розрядного АЛП, що виконує функції NOT, XOR, ADD, SUB, SUBB, SRL. При цьому крім результата Y, видається біт Z – ознака нульового результата і біт C_8 – перенос із старшого розряду.

Об′єкт АЛП виглядає як наступний

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_arith.all;
entity ALUs is
  port(  C0 : in STD_LOGIC;                          --вхід переносу (позички)
         A : in STD_LOGIC_VECTOR(7 downto 0); --1-й операнд
         B : in STD_LOGIC_VECTOR(7 downto 0);  --2-й операнд
         F : in STD_LOGIC_VECTOR(2 downto 0);  --функція АЛП
         C8 :out STD_LOGIC;      -- вихід переносу
         Z :out STD_LOGIC;       -- вихід нульового результату
         Y : out STD_LOGIC_VECTOR(7 downto 0) ); -- результат
end ALUs;
```

```
architecture ALU6 of ALUs is          --коди операцій
  constant \NOT\: STD_LOGIC_VECTOR(2 downto 0):="000";
  constant \XOR\: STD_LOGIC_VECTOR(2 downto 0):="001";
  constant \SRL\: STD_LOGIC_VECTOR(2 downto 0):="010";
  constant ADD: STD_LOGIC_VECTOR(2 downto 0):="100";
  constant SUB: STD_LOGIC_VECTOR(2 downto 0):="101";
  signal yi:SIGNED(8 downto 0);
  --функція обчислення нульового результату
  function IS_ZERO(y:SIGNED) return STD_LOGIC is
        variable zi:std_logic;
  begin
        zi:='0';
        for i in y'range loop
              zi:=zi or y(i);
        end loop;
        return NOT zi;
  end function;
begin
  ALU: with F select
  yi<=signed('0' & NOT A) when \NOT\,
                signed('0'&(A XOR B)) when \XOR\,
                signed('0'&(SHR(unsigned(A), unsigned(B(2 downto 0))))) when \SRL\,
                signed(SXT(A,9)) + signed(B) when ADD,
                signed(SXT(A,9)) - signed(B) when SUB,
                signed(SXT(A,9)) - signed(B) - C0 when others;

  Z<=IS_ZERO(yi(7 downto 0));-- нульовий результат
  C8<=yi(8); -- вихід переносу
  Y<=CONV_STD_LOGIC_VECTOR(yi,8); --сам результат АЛП
end ALU6;
```

АЛП можна перевірити за допомогою наступного стенду для іспитів

```
library ieee;
use ieee.std_logic_arith.all;
use ieee.std_logic_1164.all;
use ieee.math_real.all;
entity alus_tb is
end alus_tb;

architecture TB_ARCHITECTURE of alus_tb is
  component alus port( C0 : in std_logic;
                A : in std_logic_vector(7 downto 0);
                B : in std_logic_vector(7 downto 0);
                F : in std_logic_vector(2 downto 0);
                C8 : out std_logic;
                Z : out std_logic;
                Y : out std_logic_vector(7 downto 0) );
  end component;
  signal C0 : std_logic;
  signal A,B,Y : std_logic_vector(7 downto 0);
  signal F : std_logic_vector(2 downto 0);
  signal C8,Z : std_logic;
```

```
begin
  F<="000", "001" after 20 ns,  "010" after 40 ns,
        "100" after 60 ns,  "101" after 100 ns, "110" after 120 ns;

  RANDOM_GEN:process                -- генератор випадкових вхідних даних
          variable s1,s2:integer:=12345;
          variable r:real;
          variable ci:STD_LOGIC_VECTOR(0 to 1);
  begin
          UNIFORM(s1,s2,r);
          A<=CONV_STD_LOGIC_VECTOR(integer(r*128.0),8);
          UNIFORM(s1,s2,r);
          B<=CONV_STD_LOGIC_VECTOR(integer(r*128.0),8);
          UNIFORM(s1,s2,r);
          Ci:=CONV_STD_LOGIC_VECTOR(integer(r*128.0),2);
          C0<=ci(1);
          wait for 10 ns;
  end process;

  UUT : alus port map (       C0 => C0,
          A => A,             B => B,
          F => F,             C8 => C8,
          Z => Z,             Y => Y        );
end TB_ARCHITECTURE;
```

В стенді для іспитів вхідні дані генеруються як випадкові. При цьому кожен виклик функції UNIFORM(s1,s2,r) з пакету ieee.math_real повертає випадкове реальне число в діапазоні (0;1). Функцією CONV_STD_LOGIC_VECTOR це змасштабоване випадкове число перетворюється в вектор довжини 8.

Результати моделювання АЛП у вигляді графіків сигналів показані на рис.2.

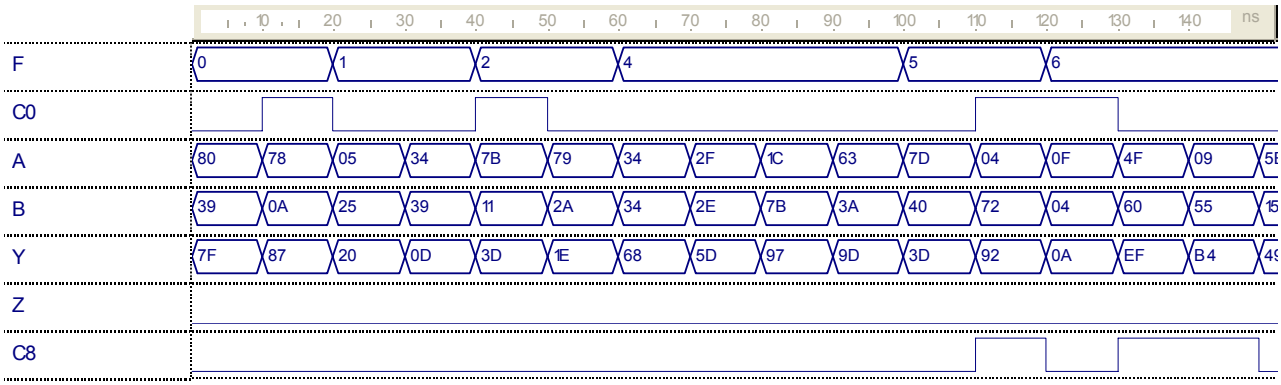

Рис.2. Графіки вхідних і вихідних сигналів АЛП