Лабораторна робота 5

Проектування АЛП стилем потоків даних

1 **Мета лабораторної роботи**:

оволодіти знаннями і навичками по проектуванню арифметико-логічних пристроїв (АЛП) для сучасних компьютерів. Ознайомитись з основами розробки проектів для програмованих логічних інтегральних схем (ПЛІС). Вивчити спосіб проектування логічних схем стилем потоків даних. Навчитись складати стенд для іспитів і користуватись ним.

Теоретичні відомості

Оператори паралельного присвоюння, паралельного виклику процедури, умовного і селективного параллельного присвоювання вказують потоки даних між лініями зв′язку, які позначені ідентифікаторами сигналів, а також обробку цих потоків. Модель обчислювача, що описується такими операторами, можна представити відповідним ациклічним графом потоків даних. Тому, якщо в тілі архітектури зустрічаються тільки такі оператори, то говорять, що така архітектура описана стилем потоків даних.

До операторів паралельного присвоювання, що програмують логічні схеми, належать наступні оператори.

Присвоювання логічного виразу, наприклад:

```
Y<= (A and B) or (not A and B);
```

Слід приймати до уваги, що оператор **not** має найвищий пріоритет, а інші логічні оператори мають однаковий пріоритет і тому у виразі з такими операторами повинні бути розставлені дужки відповідним чином.

Умовне присвоюваня сигналу:

```
Y <= A xor B when ( not F and C)  = '1' ,
     A and B when F = '1',
     else A;
```

Тут умови повинні бути булевськими виразами, тобто вони повинні повертати true, якщо при цій умові необхідно присвоїти вираз перед **when**. Множини умов можуть пересікатись, тоді першою відпрацюється умова, що зустрілась раніше при послідовному переборі умов. Умовне присвоювання сигнала повинне закінчуватись безумовним виразом **else.**

Вибіркове присвоювання сигналу:

```
with F select
Y <= A xor B xor C when  "00",
     A and B       when  "01"|"10" -- перелік варіантів,
     A             when  others;--решта
```

При цьому всі варіанти вибору повинні бути перелічені, доки не зустрінеться слово **others** як останній варіант вибору. Як варіант вибору можуть використовуватись діапазони і переліки. Множини варіантів вибору не повинні пересікатися.

2. Завдання для лабораторної роботи:

- розробити функціональну схему n-розрядного АЛП, що виконує задані функції;
- розробити стенд для іспитів, який повинен порівнювати цю модель АЛП з моделлю, розробленою в попередній лабораторній роботі.

– змоделювати роботу АЛП.

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

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

Варіант завдання такий самий, як в лабораторній роботі 4.

3. **Виконання роботи**

За основу беруться результати лабораторної роботи 3, а саме – модель одного розряду АЛП. Ця модель сприймається як компонент в новому проекті.

Результуючу VHDL-програму тестують в САПР ActiveHDL.

Для тестів розробляють стенд для іспитів, в якому АЛП, який тестується включений як компонент. Цей стенд можна підготувати за допомогою утіліти Generate Testbench, яку вибирають в меню Tools симулятора ActiveHDL.

Одержані графіки сигналів заносять в протокол лабораторної роботи.

4 **Приклад виконання роботи**

Розглянемо приклад проектування і- го розряду АЛП, що виконує функції

Y = max(A , B) при F=0 , Y = min(A, B) при F=1.

Нехай розрядність n дорівнює 12. При цьому 12-бітний АЛП повинен складатись з 12 таких розрядів. Його структура показана на рис. 1.

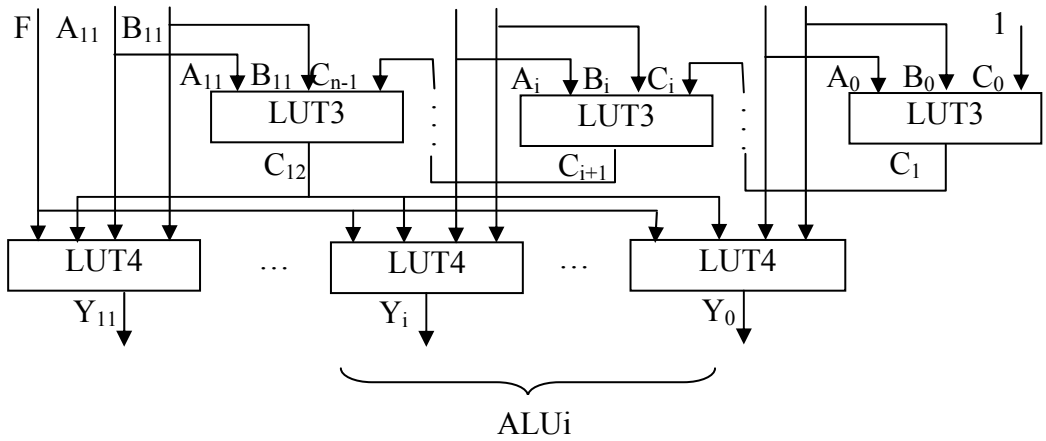

Рис.1. Структура АЛП

Логіка одного розряду ЛТ виражається наступними таблицями істинності

Ai	Bi	Ci	C_{i+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

F	Cn	Ai	Bi	Yi
0	0	0	0	0
		0	1	0
		1	0	1
		1	1	1
	1	0	0	0
		0	1	1
		1	0	0
		1	1	1
1	0	0	0	0
		0	1	1
		1	0	0
		1	1	1
	1	0	0	0
		0	1	0
		1	0	1
		1	1	1

Цю логіку можна описати наступним булевським виразом

$$C_{i+1} = A_i \cdot C_i \vee \overline{B}_i \cdot C_i \vee A_i \cdot \overline{B}_i;$$

і рівнянням:

$$Y_i = \begin{cases} A_i \quad при \quad F = 0 \quad i \quad C_n = 0 \quad або \quad F = 1 \quad i \quad C_n = 1; \\ B_i \quad при \quad F = 0 \quad i \quad C_n = 1 \quad або \quad F = 1 \quad i \quad C_n = 0. \end{cases}$$

Тоді опис моделі багаторозрядного суматора буде наступний

```
architecture dataflow of ALU12 is
 signal    c:STD_LOGIC_Vector(12 downto 0);
 signal    ff:STD_LOGIC_Vector(1 downto 0);
begin

 c(0)<='1';
 ff<= f & c(12);

 U_ALU:for i in 0 to 11 generate

--булевське рівняння
      c(i+1)<= (a(i)and c(i))or(not b(i)and c(i))
                or(a(i) and not b(i));

 -- Вибіркове присвоювання сигналу:
      with ff select
           y(i)<= a(i) when “00”|”11”,
                  b(i) when others;

 end generate;

end dataflow;
```

Слід відмітити, що ім′я даної архітектури відмінне від такої, що приведена в лабораторній роботі 4 і відображає її суть. Для цієї архітектури об′єкт проекта не приводиться, так як він вже приведений і зкомпільований в проекті за лабораторною роботою 4 .

Стенд для іспитів слід використати такий:

```
library ieee;
use ieee.std_logic_1164.all;
entity alu12_tb is
end alu12_tb;

architecture TB_ARCHITECTURE of alu12_tb is
 signal a : std_logic_vector(11 downto 0);
 signal b : std_logic_vector(11 downto 0);
 signal f : std_logic;
 signal c12 : std_logic;
 signal y1,y2,err : std_logic_vector(11 downto 0);
```

```
begin

  UUT1 : entity alu12(alu12)
       port map (
            a => a,
            b => b,
            f => f,
            c12 => c12,
            y => y1);

UUT2 : entity alu12(dataflow)
       port map (
            a => a,
            b => b,
            f => f,
            c12 => c12,
            y => y2);

  -- Add your stimulus here ...
  f<='0','1' after 100 ns;
  a<=X"000", X"111" after 40 ns, X"aaa" after 80 ns,
             X"333" after 120 ns,X"bbb" after 160 ns;
  b<=X"000", X"555" after 20 ns, X"666" after 60 ns,
             X"777" after 100 ns,X"888" after 140 ns;

-- перевірка коректності
       err<=y1 xor y2;

end TB_ARCHITECTURE;
```

Можна відзначити, що компонент в декларативній частині проекту не приведено, проте в описній частині двічі вставлено один і той самий об′єкт, але з різними архітектурами. Сигнал err призначений для фіксації відміни в моделюванні двох архітектур. Якщо в якийсь момент цей сигнал не буде нульовим, то це означає, що на відповідних тестових векторах модель, що перевіряється, не відповідає еталонній моделі.

Згенеровані графіки сигналів показані на рис.2.

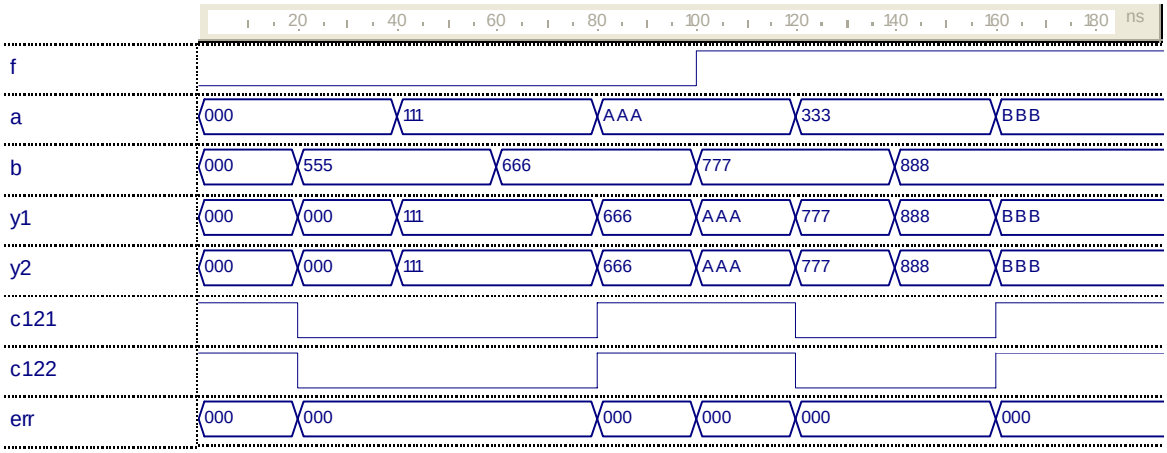

Рис.2. Графіки тестування АЛП

Аналіз графіків показує, що АЛП функціонує корректно – сигнал err завжди нульовий.