Національний університет “Львівська політехніка”

Кафедра СКС

Звіт

до лабораторної роботи №2

з дисципліни: “Дослідження і проєктування надвеликих інтегральних схем”

на тему: “Розробка структури спеціалізованого перемножувача на константу з оптимізацією”

Варіант 12

Виконав: ст. гр. кіск-13

Романів В. А.

Прийняв: д.т.н., професор

Клим Г.І.

Львів 2024

Мета: згідно заданого варіанту розробити структурну схему перемножувача на константу з оптимізацією, та за допомогою мови VHDL описати розроблену структуру пристрою та провести функціональну симуляція проекту.

Послідовність виконання роботи

1. Ознайомитись з теоретичним матеріалом.
2. Нарисувати структурну схему перемножуваа на константу з оптимізацією згідно заданого варіанту.
3. За допомогою мови VHDL описати розроблену схему пристрою.
4. Провести моделювання на функціональному рівні розробленого VHDL - проекту.
5. Скласти звіт та захистити його.

Дані згідно варіанту

Таб. 1

Значення константи згідно варіанту

№	**2-ове представлення константи**	**16-ове представлення константи**
12	0,01011011001110110000011100001010	5B3B070A

Хід виконання

1. **Ознайомитись з теоретичним матеріалом**

Вивчення методів зменшення кількості базових операцій, зокрема у контексті конвеєрних обчислень, відкриває нові можливості для підвищення ефективності апаратних пристроїв. Одним із ключових аспектів цієї оптимізації є мінімізація кількості одиниць у двійковому представленні констант, що безпосередньо впливає на кількість необхідних додавань під час множення.

Згідно з формулою A*B=A*(B-1)+A , можна ефективно зменшити кількість одиничних бітів у константі шляхом переписування чисел у такій формі, яка дозволяє уникати надмірної кількості одиниць. Цей підхід не тільки зменшує кількість додавань, але й зменшує вимоги до апаратних ресурсів, що особливо важливо для вбудованих систем та інших обчислювальних пристроїв з обмеженими можливостями.

Як видно з наведених прикладів у теоретичних відомо, застосування методу мінімізації одиниць дозволяє представити складні числові значення у спрощеній формі, зменшуючи кількість одиниць до мінімуму. Зокрема, групування одиниць, яке створює можливість віднімання менших чисел від більших, сприяє підвищенню ефективності множення.

1. **Нарисувати структурну схему перемножувача на константу з оптимізацією згідно заданого варіанту**

Згідно варіанту моя константа у двійковому вигляді рівна - 0, 0101 1011 0011 1011 0000 0111 0000 1010.

Користуючись правилом A*B=A*(B-1)+A, і позначенням $\underline{1}$ = -1, мінімізуємо кількість одиниць в константі.

1. 01011011001110110000011100001010
2. 01011011001110110000100$\underline{1}$00001010
3. 01011011001111$\underline{1}$0000100$\underline{1}$00001010
4. 01011011010000$\underline{1}$0$\underline{1}$0000100$\underline{1}$00001010
5. 0101110$\underline{1}$01000$\underline{1}$0$\underline{1}$0000100$\underline{1}$00001010
6. 01100$\underline{1}$0$\underline{1}$01000$\underline{1}$0$\underline{1}$0000100$\underline{1}$00001010

Таким чином,

Y = x.L1 + x.L3 - x.L8 + x.L11 - x.L16 - x.L18 + x.L22 - x.L24 - x.L26 + x.L29 + x.L30 = (x.L1 + x.L3) + (x.L11 + x.L22) + (x.L29 + x.L30) - ((x.L8 + x.L16) + (x.L18 + x.L24) + x.L26)

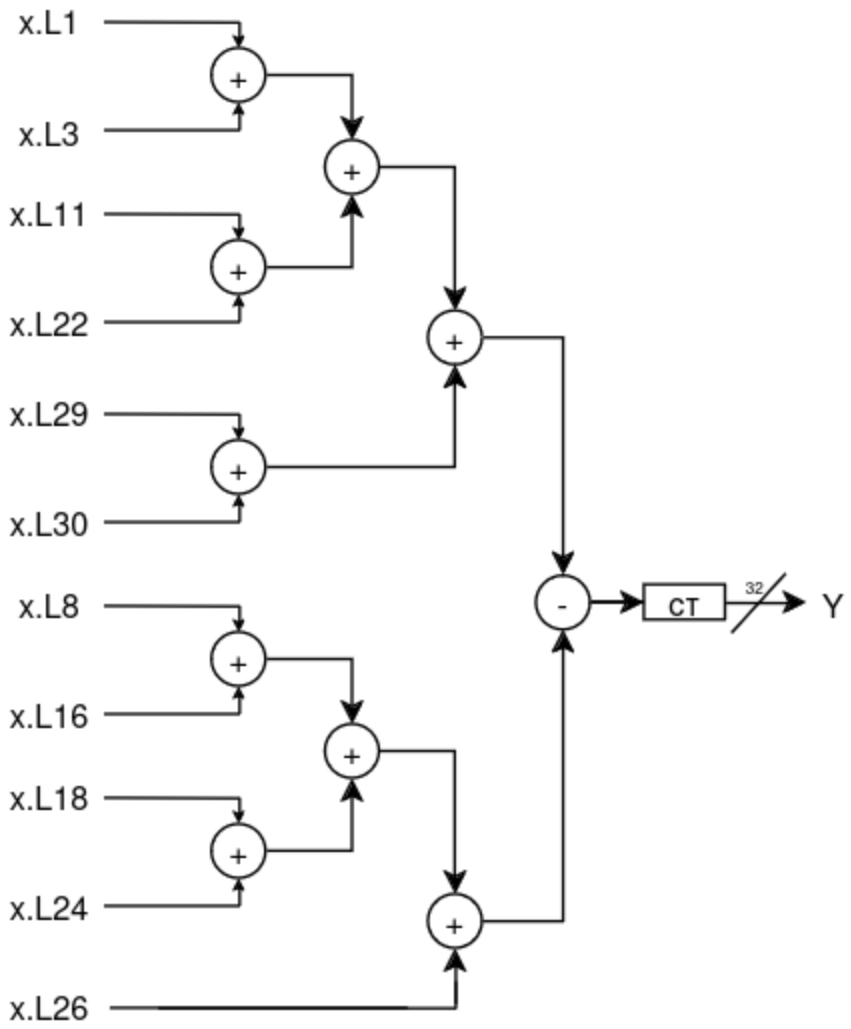

Рис. 1. Оптимізована структурна схема перемножувача на константу згідно заданого варіанту

Апаратна складність пристрою:

- Кількість одиниць - 11
- Кількість суматорів - 10

2. **За допомогою мови VHDL описати розроблену схему пристрою**

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;



entity my_mult is

    port(

        DIN  : in  STD_LOGIC_VECTOR(31 downto 0);

        DOUT : out STD_LOGIC_VECTOR(31 downto 0)

    );

end my_mult;



architecture my_mult of my_mult is

    signal X_L1, X_L3, X_L8, X_L11, X_L16, X_L18, X_L22, X_L24, X_L26, X_L29,
X_L30 : STD_LOGIC_VECTOR(63 downto 0) := (others => '0');

    signal X_L1_PLUS_X_L3, X_L11_PLUS_X_L22, X_L29_PLUS_X_L30,
X_L8_PLUS_X_L16, X_L18_PLUS_X_L24 : STD_LOGIC_VECTOR(63 downto 0) := (others => '0');

    signal X_L1_PLUS_X_L3_PLUS_X_L11_PLUS_X_L22,
X_L8_PLUS_X_L16_PLUS_X_L18_PLUS_X_L24 : STD_LOGIC_VECTOR(63 downto 0) := (others =>
'0');

    signal X_L1_PLUS_X_L3_PLUS_X_L11_PLUS_X_L22_PLUS_X_L29_PLUS_X_L30,
X_L8_PLUS_X_L16_PLUS_X_L18_PLUS_X_L24_PLUS_X_L26 : STD_LOGIC_VECTOR(63 downto 0) :=
(others => '0');

    signal
X_L1_PLUS_X_L3_PLUS_X_L11_PLUS_X_L22_PLUS_X_L29_PLUS_X_L30_MINUS_X_L8_PLUS_X_L16_PLUS
_X_L18_PLUS_X_L24_PLUS_X_L26 : STD_LOGIC_VECTOR(63 downto 0) := (others => '0');

begin

    X_L1  <= "00000000000000000000000000000000" & DIN & "0";

    X_L3  <= "00000000000000000000000000000000" & DIN & "000";

    X_L8  <= "00000000000000000000000000" & DIN & "00000000";

    X_L11 <= "00000000000000000000" & DIN & "00000000000";
```

```
            X_L16 <= "0000000000000000" & DIN & "0000000000000000";

            X_L18 <= "00000000000000" & DIN & "00000000000000000000";

            X_L22 <= "0000000000" & DIN & "00000000000000000000000000";

            X_L24 <= "00000000" & DIN & "00000000000000000000000000";

            X_L26 <= "000000" & DIN & "00000000000000000000000000";

            X_L29 <= "000" & DIN & "00000000000000000000000000000";

            X_L30 <= "00" & DIN & "00000000000000000000000000000";


            X_L1_PLUS_X_L3 <= X_L1 + X_L3;

            X_L11_PLUS_X_L22 <= X_L11 + X_L22;

            X_L29_PLUS_X_L30 <= X_L29 + X_L30;

            X_L8_PLUS_X_L16 <= X_L8 + X_L16;

            X_L18_PLUS_X_L24 <= X_L18 + X_L24;


            X_L1_PLUS_X_L3_PLUS_X_L11_PLUS_X_L22 <= X_L1_PLUS_X_L3 + X_L11_PLUS_X_L22;

            X_L8_PLUS_X_L16_PLUS_X_L18_PLUS_X_L24 <= X_L8_PLUS_X_L16 +
X_L18_PLUS_X_L24;



            X_L1_PLUS_X_L3_PLUS_X_L11_PLUS_X_L22_PLUS_X_L29_PLUS_X_L30 <=
X_L1_PLUS_X_L3_PLUS_X_L11_PLUS_X_L22 + X_L29_PLUS_X_L30;

            X_L8_PLUS_X_L16_PLUS_X_L18_PLUS_X_L24_PLUS_X_L26 <=
X_L8_PLUS_X_L16_PLUS_X_L18_PLUS_X_L24 + X_L26;



X_L1_PLUS_X_L3_PLUS_X_L11_PLUS_X_L22_PLUS_X_L29_PLUS_X_L30_MINUS_X_L8_PLUS_X_L16_PLUS
_X_L18_PLUS_X_L24_PLUS_X_L26 <=
X_L1_PLUS_X_L3_PLUS_X_L11_PLUS_X_L22_PLUS_X_L29_PLUS_X_L30 -
X_L8_PLUS_X_L16_PLUS_X_L18_PLUS_X_L24_PLUS_X_L26;



            DOUT <=
X_L1_PLUS_X_L3_PLUS_X_L11_PLUS_X_L22_PLUS_X_L29_PLUS_X_L30_MINUS_X_L8_PLUS_X_L16_PLUS
_X_L18_PLUS_X_L24_PLUS_X_L26(63 downto 32);
```

```
end my_mult;
```

Також був написаний **test bench** на мові VHDL, щоб перевірити правильність виконання програми:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.all;

entity my_mult_tb is
end my_mult_tb;

architecture Behavioral of my_mult_tb is

    -- Component Declaration for the DUT
    component my_mult
    port(
        DIN  : in  STD_LOGIC_VECTOR(31 downto 0);
        DOUT : out STD_LOGIC_VECTOR(31 downto 0)
    );
end component;

    -- Signals for connecting to the DUT
signal DIN  : STD_LOGIC_VECTOR(31 downto 0);
signal DOUT : STD_LOGIC_VECTOR(31 downto 0);

begin
```

```
-- Instantiate the DUT
uut: my_mult
port map (
    DIN => DIN,
    DOUT => DOUT
);

-- Stimulus process
stim_proc: process
begin
    -- Test case 1
    DIN <= X"00000000";
    wait for 10 ns;
    assert (DOUT = X"00000000") report "Test case 1 failed" severity error;

    -- Test case 2
    DIN <= X"FFFFFFFF";
    wait for 10 ns;
    assert (DOUT = X"5B3B0709") report "Test case 2 failed" severity error;

    -- Test case 3
    DIN <= X"ABCABCAB";
    wait for 20 ns;
    assert (DOUT = X"3D38AD83") report "Test case 3 failed" severity error;

    -- Test case 4
    DIN <= X"CDAABCDC";
```

```
        wait for 20 ns;
        assert (DOUT = X"494B1D23") report "Test case 4 failed" severity error;


        -- End of simulation
        wait;
    end process;


end Behavioral;
```

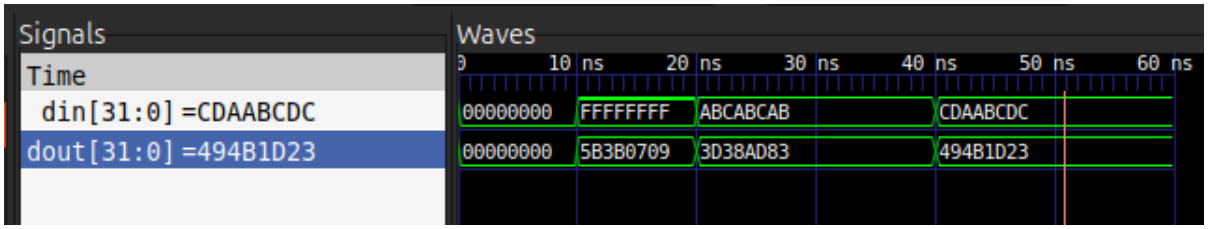

Рис. 2. Результат виконання програми

Висновок: під час виконання даної лабораторної роботи мною було розроблено оптимізовану структурну схему перемножувача на константу згідно варіанту та за допомогою мови VHDL описано розроблену структуру пристрою та проведено функціональну симуляція проекту.