Національний університет “Львівська політехніка”

Кафедра СКС

Звіт

до лабораторної роботи №1

з дисципліни: “Дослідження і проєктування надвеликих інтегральних схем”

на тему: “Розробка структури спеціалізованого перемножувача на константу без оптимізації”

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 без оптимізації, 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 перемножувача на константу без оптимізації 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

Алгоритм множення числа на константу базується на принципах двійкового множення, де один із співмножників зсувається вправо відповідно до значення бітів іншого співмножника. Якщо необхідно

отримати добуток двох восьмирозрядних чисел, результатом буде сума чисел, які утворюються при зсуванні одного з чисел на кількість розрядів, що відповідає бітам іншого числа.

Множення числа на константу спрощується завдяки тому, що розряди константи відомі заздалегідь. Це дозволяє зменшити кількість операцій, що виконуються апаратно. Наприклад, при множенні на константу H(2) = 10011000 результат Y можна отримати, додавши лише ті зсуви числа X, що відповідають одиничним бітам константи.

2. **Нарисувати структурну схему перемножувача на константу без оптимізації згідно заданого варіанту**

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

Таким чином,

Y = x.L1 + x.L3 + x.L8 + x.L9 + x.L10 + x.L16 + x.L17 + x.L19 + x.L20 + x.L21 + x.L24 + x.L25 + x.L27 + x.L28 + x.L30

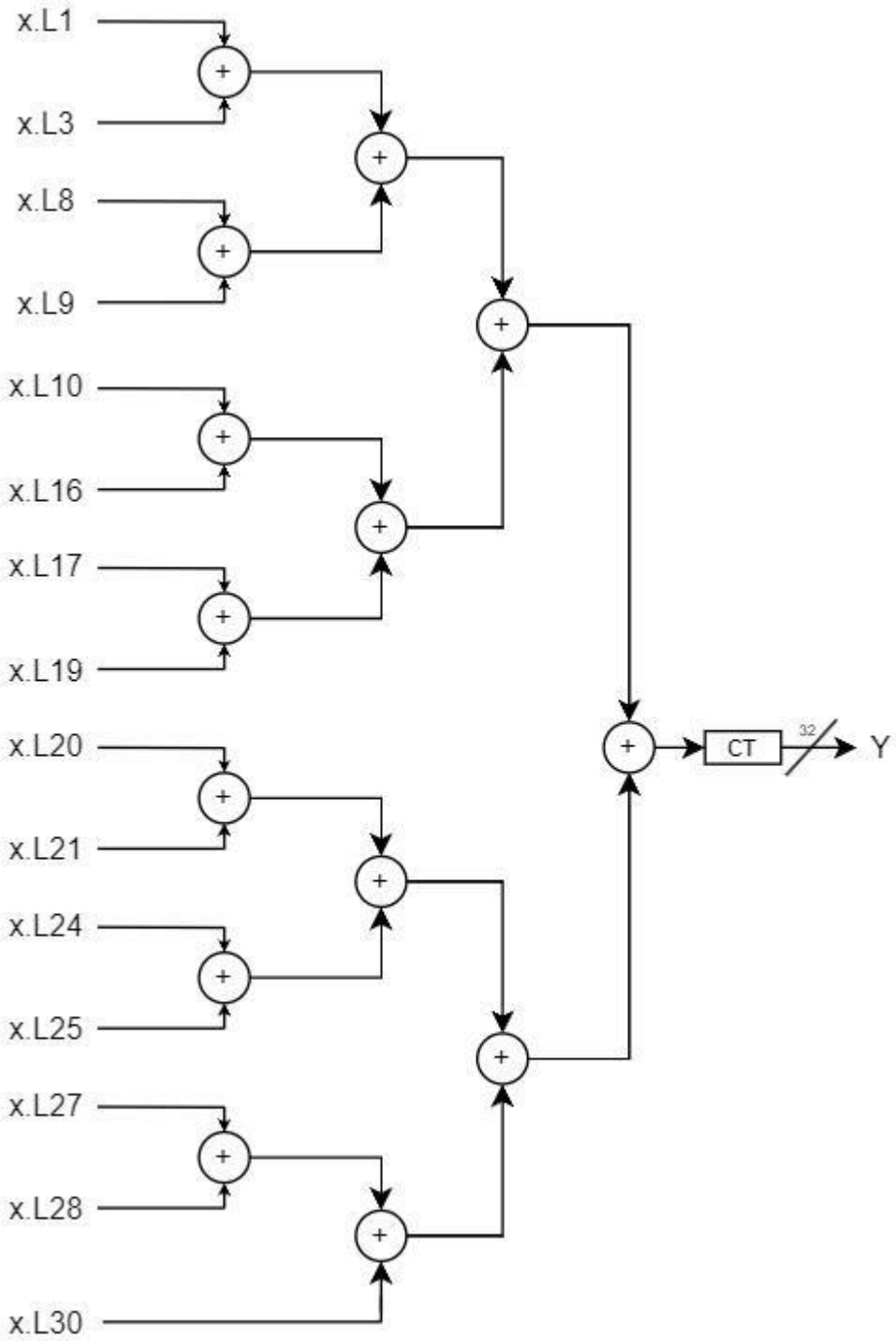

Рис. 1. Структурна схема перемножувача на константу без оптимізації згідно заданого варіанту

Апаратна складність пристрою визначається кількістю ’1’ в константі.

- Кількість одиниць - 15
- Кількість суматорів - 14

3. 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_L9, X_L10, X_L16, X_L17, X_L19, X_L20, X_L21,
X_L24, X_L25, X_L27, X_L28, X_L30 : STD_LOGIC_VECTOR(63 downto 0) := (others =>
'0');

	signal X_L1_L3, X_L8_L9, X_L10_L16, X_L17_L19, X_L20_L21, X_L24_L25,
X_L27_L28 : STD_LOGIC_VECTOR(63 downto 0) := (others => '0');

	signal X_L1_L3_L8_L9, X_L10_L16_L17_L19, X_L20_L21_L24_L25, X_L27_L28_L30 :
STD_LOGIC_VECTOR(63 downto 0) := (others => '0');

	signal X_L1_L3_L8_L9_L10_L16_L17_19, X_L20_L21_L24_L25_L27_L28_L30 :
STD_LOGIC_VECTOR(63 downto 0) := (others => '0');

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

begin

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

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

```
X_L8  <= "00000000000000000000000" & DIN & "00000000";

X_L9  <= "0000000000000000000000" & DIN & "000000000";

X_L10 <= "000000000000000000000" & DIN & "0000000000";

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

X_L17 <= "000000000000000" & DIN & "00000000000000000";

X_L19 <= "0000000000000" & DIN & "000000000000000000";

X_L20 <= "000000000000" & DIN & "000000000000000000";

X_L21 <= "00000000000" & DIN & "000000000000000000";

X_L24 <= "00000000" & DIN & "00000000000000000000000";

X_L25 <= "0000000" & DIN & "000000000000000000000000";

X_L27 <= "00000" & DIN & "0000000000000000000000000";

X_L28 <= "0000" & DIN & "00000000000000000000000000";

X_L30 <= "00" & DIN & "000000000000000000000000000000";


X_L1_L3 <= X_L1 + X_L3;

X_L8_L9 <= X_L8 + X_L9;

X_L10_L16 <= X_L10 + X_L16;

X_L17_L19 <= X_L17 + X_L19;

X_L20_L21 <= X_L20 + X_L21;

X_L24_L25 <= X_L24 + X_L25;

X_L27_L28 <= X_L27 + X_L28;


X_L1_L3_L8_L9 <= X_L1_L3 + X_L8_L9;

X_L10_L16_L17_L19 <= X_L10_L16 + X_L17_L19;

X_L20_L21_L24_L25 <= X_L20_L21 + X_L24_L25;

X_L27_L28_L30 <= X_L27_L28 + X_L30;


X_L1_L3_L8_L9_L10_L16_L17_19 <= X_L1_L3_L8_L9 + X_L10_L16_L17_L19;

X_L20_L21_L24_L25_L27_L28_L30 <= X_L20_L21_L24_L25 + X_L27_L28_L30;
```

```
	Y <= X_L1_L3_L8_L9_L10_L16_L17_19 + X_L20_L21_L24_L25_L27_L28_L30;
	DOUT <= Y(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_2 is
end my_mult_tb_2;

architecture Behavioral of my_mult_tb_2 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";  -- Example value
wait for 10 ns;  -- Wait for the output to stabilize
assert (DOUT = X"00000000") report "Test case 1 failed" severity error;

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

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

-- Test case 4
DIN <= X"CDAABCDC";  -- Example value
wait for 20 ns;
assert (DOUT = X"494B1D23") report "Test case 4 failed" severity error;
```

```
        -- End of simulation

        wait;

        end process;

end Behavioral;
```

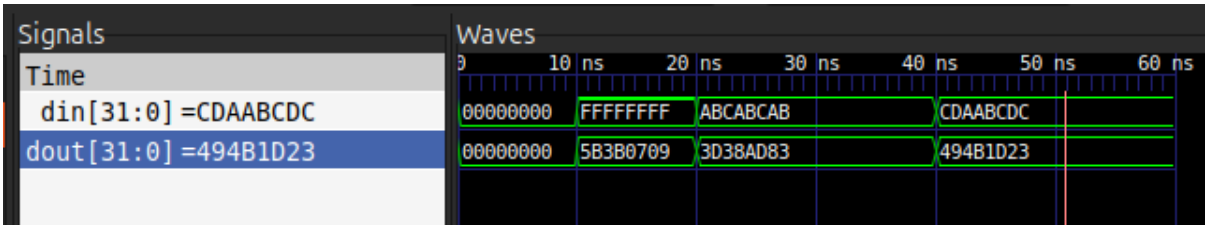

Рис. 2. Результат виконання програми

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