Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота № 4. Проектування FM

з дисципліни «Комп’ютерна схемотехніка»

Виконав студент групи ІО-01 *Редько Олександр*

Номер залікової книжки **10310**

# Завдання

На PLMT с параметром построить KF‑канальную FM емкостью  слов разрядностью  бит, где ,  означает: , ,, .

Оценить сложность и быстродействие полученной схемы.

# Визначення варіанту

 ‑ 5 входов.

 ‑ количество слов FM.

 ‑ 15 бит.

 ‑ трехканальная FM .

# Виконання роботи

Блок FM должен содержать 8 регистров размером 15 бит, два мультиплексора считанных данных по каналам B, D и дешифратор записи по каналу Q (рис. Рисунок 3.1).

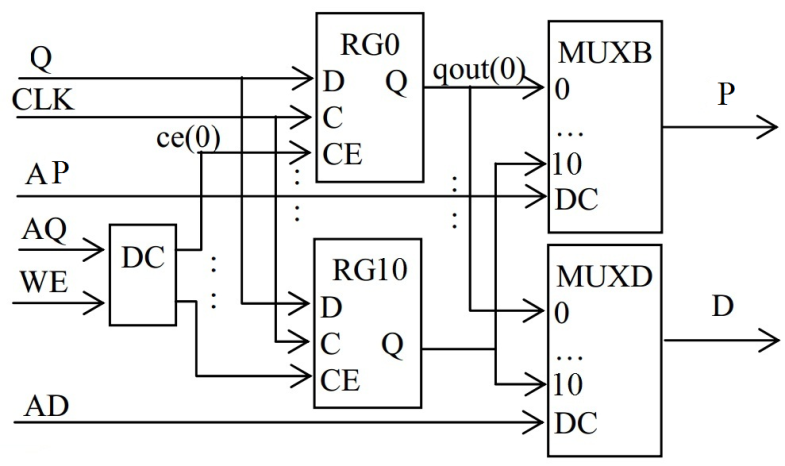


Рисунок 3.1 – Структура FM

Дешифратор DC состоит из десяти PLM, каждая из которых декодирует три разряда адреса. Первая из них описывается следующим образом при D = WE, C =A2, B = A1, A=A0:

architecture PLM4\_DC0 of PLM\_4 is

begin

Y <= (D and not C and not B and not A) after td;

end PLM4\_DC0;

Остальные PLM дешифратора- PLM4\_DC1,…,PLM4\_DC7- описываются аналогично.

Каждый из мультиплексоров состоит из пятнадцати однобитных восьмивходовых мультиплексоров. Восьмивходовый мультиплексор можно представить как два четырехвходовых мультиплексора, выходы которых объединены в двухвходовом мультиплексоре.

Четырехвходовый мультиплексор описывается следующим образом при F=D3, E=D2, D = D1, C = D0, B=A1, A=A0:

architecture PLM6\_MUX of PLM\_6 is

begin

Y <= (C and not B and not A) -- 0-й вход

or(D and not B and A) -- 1-й вход

or(E and B and not A) -- 2- й вход

or(F and B and A) -- 3- й вход

after td; -- задержка элемента

end PLM6\_MUX;

Двухвходовый мультиплексор имеет следующую архитектуру при C = D1, B= D0, A=A0:

architecture PLM3\_MUX of PLM\_3 is

begin

Y <= (B and not A) or (C and A) after td;

end PLM3\_MUX;

Cтруктурная модель восьмивходового мультиплексора описывается в следующем объекте.

library IEEE;

use IEEE.STD\_LOGIC\_1164.all;

entity MUX8 is port(D0, D1, D2, D3, D4, D5, D6, D7: in STD\_LOGIC;

A: in STD\_LOGIC\_VECTOR(2 downto 0);

Q: out STD\_LOGIC);

end MUX8;

architecture Behavioral of MUX8 is

signal mux0, mux1 : STD\_LOGIC;

begin

U\_MUX0: entity PLM\_6(PLM6\_MUX)

port map(F => D3, E => D2, D => D1, C => D0, B => A(1), A => A(0), Y => mux0);

U\_MUX1: entity PLM\_6(PLM6\_MUX)

port map(F => D7, E => D6, D => D5, C => D4, B => A(1), A => A(0), Y => mux1);

U\_MUX3: entity PLM\_3(PLM3\_MUX)

port map(C => mux1, B => mux0, A => A(2), Y => Q);

end Behavioral;

Тело архитектуры регистровой памяти выглядит так:

library IEEE;

use IEEE.STD\_LOGIC\_1164.all;

architecture FM\_BEH of FM is

type FMARR is array(7 downto 0, 14 downto 0) of STD\_LOGIC;

signal y: FMARR;

signal ce: STD\_LOGIC\_VECTOR(7 downto 0);

constant gnd: STD\_LOGIC := '0';

component MUX8 is port(D0, D1, D2, D3, D4, D5, D6, D7: in STD\_LOGIC; -- входы

A: in STD\_LOGIC\_VECTOR(2 downto 0); -- адрес

Q: out STD\_LOGIC); -- выход данного

end component;

component FDRE is port(Q: out STD\_LOGIC; -- триггер

D : in STD\_LOGIC;

C : in STD\_LOGIC;

CE: in STD\_LOGIC;

R : in STD\_LOGIC);

end component;

begin

-- дешифратор адреса

U\_DC0: entity PLM\_4(PLM4\_DC0)

port map(D => WR, C => AQ(2), B => AQ(1), A => AQ(0), Y => ce(0));

U\_DC1: entity PLM\_4(PLM4\_DC1)

port map(D => WR, C => AQ(2), B => AQ(1), A => AQ(0), Y => ce(1));

U\_DC2: entity PLM\_4(PLM4\_DC2)

port map(D => WR, C => AQ(2), B => AQ(1), A => AQ(0), Y => ce(2));

U\_DC3: entity PLM\_4(PLM4\_DC3)

port map(D => WR, C => AQ(2), B => AQ(1), A => AQ(0), Y => ce(3));

U\_DC4: entity PLM\_4(PLM4\_DC4)

port map(D => WR, C => AQ(2), B => AQ(1), A => AQ(0), Y => ce(4));

U\_DC5: entity PLM\_4(PLM4\_DC5)

port map(D => WR, C => AQ(2), B => AQ(1), A => AQ(0), Y => ce(5));

U\_DC6: entity PLM\_4(PLM4\_DC6)

port map(D => WR, C => AQ(2), B => AQ(1), A => AQ(0), Y => ce(6));

U\_DC7: entity PLM\_4(PLM4\_DC7)

port map(D => WR, C => AQ(2), B => AQ(1), A => AQ(0), Y => ce(7));

-- массив регистров

U\_FM: for i in 0 to 7 generate

U\_RG: for j in 0 to 14 generate

U\_TT: FDRE port map(D => Q(j), -- входное данное

C => CLK, -- синхросигнал

CE => ce(i), -- разрешение записи

R => gnd, -- сброс не используется

Q => y(i,j)); -- выходы триггеров

end generate;

end generate;

-- выходные мультиплексоры

U\_MUX: for i in 0 to 14 generate

MUXD: MUX8 port map(D0 => y(0,i), D1 => y(1,i), D2 => y(2,i), D3 => y(3,i),

D4 => y(4,i), D5 => y(5,i), D6 => y(6,i), D7 => y(7,i),

A => AD, -- адрес

Q => D(i)); -- адрес канала D

MUXP: MUX8 port map(D0 => y(0,i), D1 => y(1,i), D2 => y(2,i), D3 => y(3,i),

D4 => y(4,i), D5 => y(5,i), D6 => y(6,i), D7 => y(7,i),

A => AP, -- адрес

Q => P(i)); -- выход канала P

end generate;

end FM\_BEH;

Результирующая схема, полученная с помощью утилиты Code2Graphics, приведена на рис. Рисунок 3.2.



Рисунок 3.2

Рассчитаем сложность полученной схемы.

N = (8\*3) \* 24=504 PLMT

Рассчитаем быстродействие. При операциях чтения:

T = τ\_dc + τ\_vid = 1нс, так как операции извлечения результата из триггера и разрешения его выдачи выполняются параллельно.

При операции записи:

T = τ\_dc + τ\_vid + τ\_vid = 1нс + 1нс +1нс = 3нс.