МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Лабораторна робота №3

з дисципліни «Технології проектування комп’ютерних систем»
на тему: «Оперативний запам'ятовуючий пристрій»

Виконав:
студент 4-го курсу
факультету ІОТ
групи ІО-41
Демчик В. В.
НЗК 4111

Перевірив:
проф. Сергієнко А. М.

Київ 2017

Тема: Оперативний запам'ятовуючий пристрій.

Мета та основні завдання роботи: оволодіти знаннями і практичними навичками з проектування пристроїв пам'яті, таких як ОЗП (RAM). Лабораторна робота також служить для оволодіння навичками програмування і налагодження опису RAM на мові VHDL.

Завдання на лабораторну роботу: розробити блок пам’яті за наведеними нижче умовами:

Розрядність слів – 16 біт.

Об’єм пам’яті – 5120 слів.

Виходів – 1.

Тип виходу – тристабільний буфер.

Виконати описання поведінкової моделі. Провести аналіз отриманих графіків роботи схем.

Хід проектування:

Виконаємо опис входів та виходів RAM:

CLK	Синхросигнал
R	Скидання
WR	Запис даних IO в пам’ять за адресою AD
OE	Сигнал тристабільному буферу на видачу прочитаного слова
AD	Адреса
IO	Вхідні/вихідні дані

Оскільки за варіантом ОЗП повинна вміщувати 5120 слів, то для їх індексації знадобляться (5119_{10}= 0001 0011 1111 1111_2) - 16-бітні адреси.

Код програми:

Поведінкова модель:

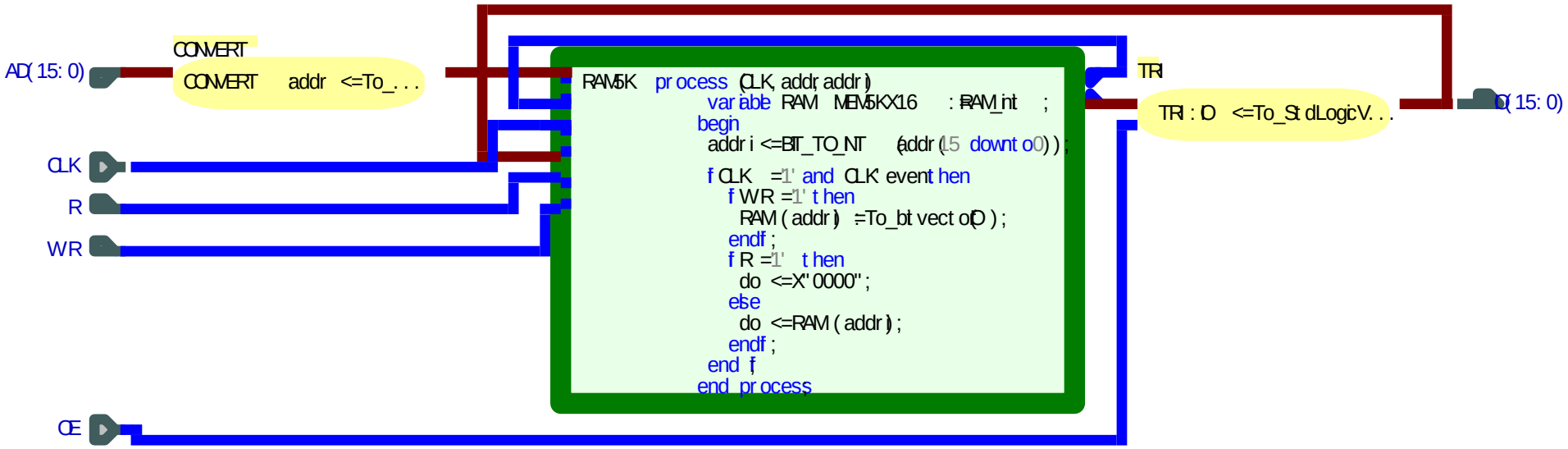
```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
library UNISIM;
use UNISIM.all;
use CNetwork.all;

entity RAM is
port(
 CLK : in BIT; -- synchro
 R : in BIT; -- reset
 WR: in BIT; -- write
 OE: in BIT; -- get read word
 AD : in STD_LOGIC_VECTOR(15 downto 0); -- address
 IO : inout STD_LOGIC_VECTOR(15 downto 0));-- data
```

```
end RAM;

architecture BEH of RAM is
type MEM5KX16 is array(0 to 5119) of BIT_VECTOR(15 downto 0);
constant RAM_init: MEM5KX16:= -- initial memory status
 (X"0000",X"0000",X"0000",X"0000",X"0000",X"0000",X"0000",X"0000",others=> X"0000");
signal addr,do: BIT_VECTOR(15 downto 0);
signal addri : INTEGER;
begin
addr<= To_bitvector(AD);
 ------ Memory block ------------------
RAM1K:process(CLK,addr,addri)
variable RAM: MEM5KX16:= RAM_init;
--variable addri: INTEGER;
 begin
 addri <= BIT_TO_INT(addr(15 downto 0));
 if CLK='1' and CLK'event then
            if WR = '1' then
                        RAM(addri):= To_bitvector(IO); --write
            end if;
            if R='1' then
                        do<= X"0000"; --reset
            else
                        do<= RAM(addri); --read
            end if;
 end if;
end process;
 -- Tristate output buffer ----------------------
 TRI:IO<= To_StdLogicVector(do) when OE='1'
 else "ZZZZZZZZZZZZZZZZ";
end BEH;
```

Згенерована схема на основі описаної архітектури:



Design Unit Header

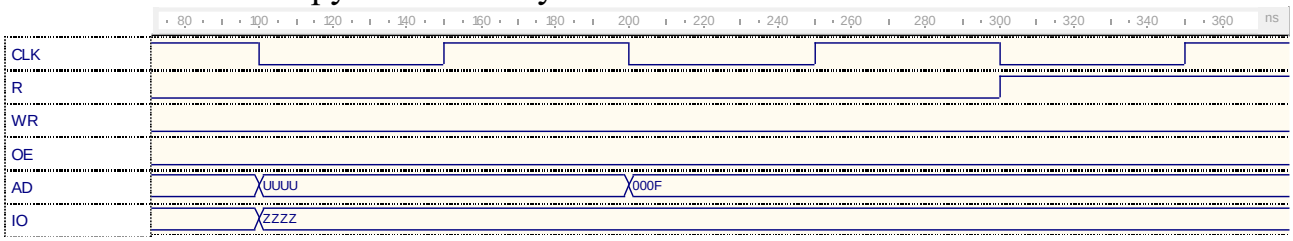
```
library ram ;
    use ram cnet work al ;
library ieee ;
    use ieee std_logic_1164 all ;
```

Architecture Declaration

```
constant RAM_int MEM5KX16 : =(X" 0000", X" 0000", X" 0000", X" 0000", X" 0000", X" 0000", X" 0000", X" 0000", others => X" 0000") ;
-- Added by Active-HDL. Do not change code inside this section.
type MEM5KX16 is array (0 to 5119) of BIT_VECTOR (15 downto 0) ;
-- End of extra code.
```

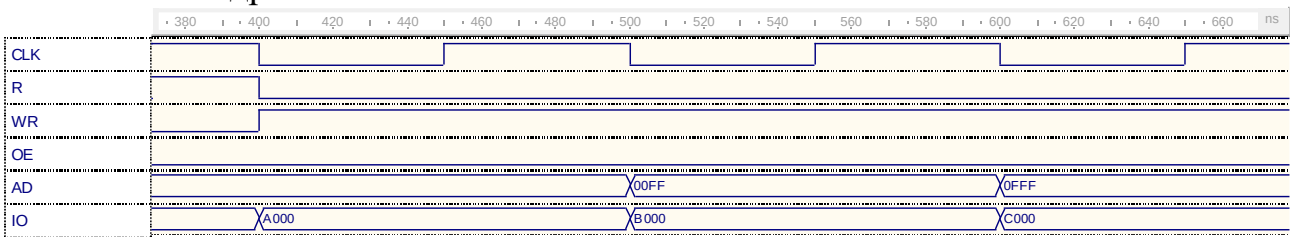
Результати симуляції:

Для початку перевіримо роботу тристабільного буферу на «холостому ході», встановивши всі керуючі біти в нульове положення:

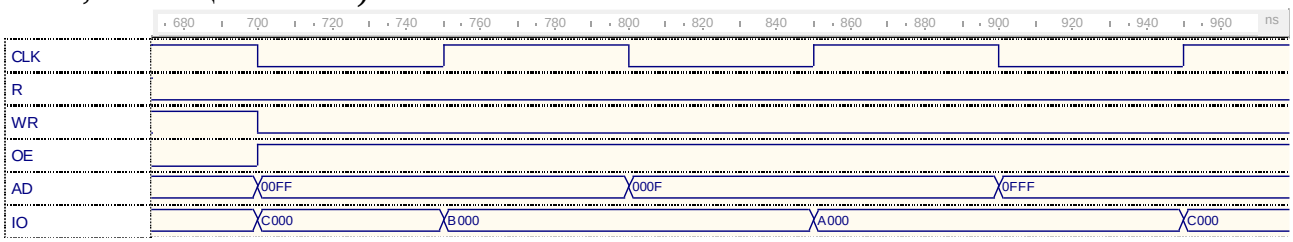


Як бачимо, незалежно від того, надходить адреса чи ні, а також чи активне скидання (R) чи ні, на виході ми маємо високоомний стан.

Запишемо за адресою 000F слово A000,
за адресою 00FF слово B000,
за адресою 0FFF слово C000.



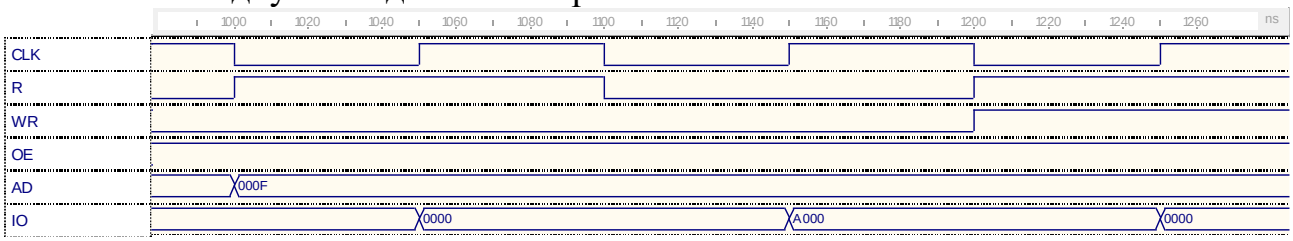
Тепер прочитаємо ці слова в тасованому порядку (спочатку за 00FF, потім за 000F, і в кінці за 0FFF):



Як бачимо, головна перевага роздільних входів для даних та для адреси в тому, що запис за поточною адресою відбувається за один такт, а не за два, як було б у випадку спільної шини адреси та даних.

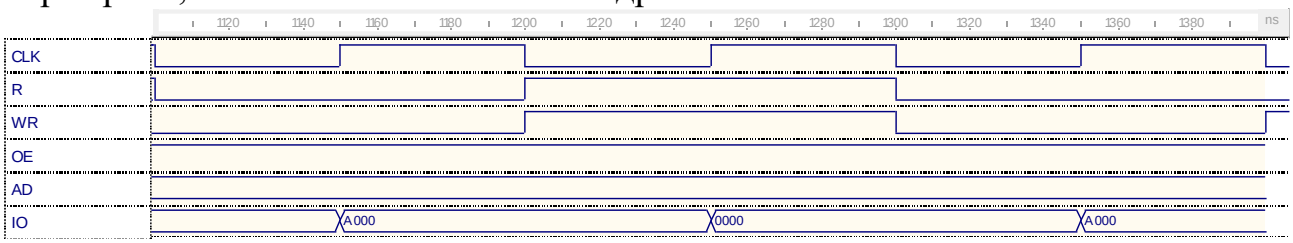
Відповідно головний недолік – неможливість конвеєрного принципу роботи, коли за один такт відбувається запам'ятовування адреси для нових даних, та видача дані за старою адресою. Проте, за роздільних шин адреси та даних такий принцип роботи взагалі не є необхідним.

Перевіримо видачу нульового слова по випадковій адресі (000F) та сигналу R, а також чи не відбувся подальший перезапис слова:



Як бачимо, перезапису не відбулося, після встановлення R назад в «0» за адресою 000F було видано слово A000.

Спробуємо подати одночасно сигнали R та W, а потім відімкнути їх та перевірити, чи не змінилося слово за адресою 000F.



Як бачимо, перезапису не відбулося.

Зробимо перезапис за адресою 00FF. Запишемо туди D000, і через один такт здійснимо зчитування за цією ж адресою.