**PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**KHOA ĐIỆN - ĐIỆN TỬ**

**BỘ MÔN ĐIỆN TỬ VIỄN THÔNG**



**BÁO CÁO MÔN HỌC**

**Thiết kế hệ thống và vi mạch tích hợp**

**Đề Tài: Thiết kế Single port RAM bằng Verilog**

Sinh viên thực hiện:

**BÙI HỮU TÀI**  18161268

**NGUYỄN VĂN TÍNH** 18161287

GVHD: **TS. ĐỖ DUY TÂN**

TP. HỒ CHÍ MINH – 3/2021**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

# **Mục lục**

**[Mục lục](#_Toc68037276)** [1](#_Toc68037276)

**[Danh mục hình ảnh](#_Toc68037277)** [1](#_Toc68037277)

**[CHƯƠNG 1. TỔNG QUAN](#_Toc68037278)** [2](#_Toc68037278)

**[1.1 Đặt ra vấn đề](#_Toc68037279)** [2](#_Toc68037279)

**[1.2 Mục tiêu](#_Toc68037280)** [3](#_Toc68037280)

**[CHƯƠNG 2. CƠ SỞ LÍ THUYẾT](#_Toc68037281)** [4](#_Toc68037281)

**[2.1. Giới thiệu về Memory](#_Toc68037282)** [4](#_Toc68037282)

**[2.1.1 Định nghĩa về RAM](#_Toc68037283)** [5](#_Toc68037283)

**[2.1.2 Lịch sử về RAM](#_Toc68037284)** [7](#_Toc68037284)

**[2.1.3 Phân loại](#_Toc68037285)** [7](#_Toc68037285)

**[CHƯƠNG 3: CÁC BƯỚC THIẾT KẾ VÀ MÔ PHỎNG HỆ THỐNG](#_Toc68037288)** [12](#_Toc68037288)

**[3.1 Sơ Đồ Khối Của bản thiết kế.](#_Toc68037289)** [12](#_Toc68037289)

**[3.2 Mô Tả Thanh Ghi](#_Toc68037290)** [14](#_Toc68037290)

**[3.3 Thiết kế chi tiết](#_Toc68037291)** [15](#_Toc68037291)

**[3.3.1 SINGLE PORT RAM](#_Toc68037292)** [16](#_Toc68037292)

[3.3.2.](#_Toc68037293) **[Mô phỏng qua testbench:](#_Toc68037293)** [17](#_Toc68037293)

**[CHƯƠNG 4: ĐÁNH GIÁ QUA TESTBENCH](#_Toc68037294)** [20](#_Toc68037294)

**[4.1 Mô hình Testbench tổng quát](#_Toc68037295)** [20](#_Toc68037295)

**[4.1.1 Giới thiệu](#_Toc68037296)** [20](#_Toc68037296)

**[4.2 Các thành phần bên trong testbench](#_Toc68037297)** [21](#_Toc68037297)

**[4.3 Kết quả Simulation](#_Toc68037298)** [24](#_Toc68037298)

**[4.4 Các trường hợp Testcase](#_Toc68037299)** [25](#_Toc68037299)

**[4.4.1 Quá trình cho phép đọc ghi dữ liệu.](#_Toc68037300)** [25](#_Toc68037300)

**[4.4.2 Ghi dữ liệu vào RAM](#_Toc68037301)** [27](#_Toc68037301)

**[4.4.3 Đọc dữ liệu từ RAM](#_Toc68037302)** [28](#_Toc68037302)

**[4.5. Kết luận và hướng phát triển](#_Toc68037303)** [29](#_Toc68037303)

# **Danh mục hình ảnh**

**Hình 2.1. CD 5**

**Hình 2.2. Băng từ 5**

**Hình 2.3. Cache 5**

**Hình 2.4. Đĩa cứng 5**

**Hình 2.5. Công nghệ bộ nhớ 6**

**Hình 2.6. SDR RAM 7**

**Hình 1.7. SRAM 8**

**Hình 2.8. Cấu trúc SRAM và cấu trúc cell 8**

**Hình 2.9. DRAM 9**

**Hình 2.10. Cấu trúc DRAM 9**

**Hình 2.11. SDRAM 10**

**Hình 2.12. SDR SDRAM 11**

**Hình 2.13. DDR SDRAM 11**

**Hình 3.1 Sơ đồ In\_Out của Single-port RAM 12**

**Hình 3.2 Sơ đồ chi tiết bên trong khối RAM 13**

**Hình 3.3 Mô tả thanh ghi 14**

**Hình 3.4 Chi tiết bên trong thanh ghi 15**

**Hình 4.1 Mô hình RTL của RAM khi code 20**

**Hình 4.2 waveform từ simulation 25**

***Hình 4.3 Mô phỏng quá trình cho phép đọc-ghi dữ liệu 26***

***Hình 4.4 Mô phỏng quá trình ghi dữ liệu vào RAM 27***

***Hình 4.5 Mô phỏng quá trình đọc dữ liệu ra từ RAM 28***

***Hình 4.6 Mô phỏng quá trình đọc dữ liệu ra từ RAM 28***

# **CHƯƠNG 1. TỔNG QUAN**

## **Đặt ra vấn đề**

RAM-Random Access Memory (Bộ nhớ truy xuất ngẫu nhiên) là một thuật ngữ không quá xa lạ với con người hiện nay. Chip RAM có mặt trên thị trường vào cuối những năm 1960, được sản xuất bởi Intel-một trong những công ty về công nghệ hàng đầu hiện nay, và dần theo thời gian ngành thiết kế bộ nhớ ngày càng được chú trọng, qui mô ngày một rộng mở, các công ty sẵn sàng đầu tư tài chính, hướng mũi ngọn về ngành công nghệ này, nhằm không ngừng tối ưu về kích thước, tốc độ,… như : KIOXIA, Kingston, Renesas,…Vậy, tại sao RAM ngày càng được các “ông lớn” về công nghệ để mắt đến ?

Bởi lẽ, sự phát triển ngày càng nhanh của khoa học kỹ thuật, kéo theo cần có các siêu máy tính với tốc độ xử lí cao, bộ nhớ lớn,… mới đáp ứng được những vấn đề vượt qua khả năng của con người. Vì vậy, các công ty ra sức đầu tư về mặt cơ sở vật chất và nguồn nhân lực lớn mạnh nhất nhằm có được sản phẩm tốt nhất, tiên tiến nhất. Điển hình như Renesas đã đến đại học Sư Phạm Kỹ Thuật để tuyển dụng những kĩ sư tiềm năng cho công ty. Chính vì thế, nhóm em đã chọn đề tài “Nghiên cứu về Memory (RAM)” là tiểu luận môn học Thiết kế vi mạch số và vi mạch tích hợp, qua tiểu luận nhằm tự trang bị cho bản thân kiến thức cơ bản về RAM, cũng như tăng cơ hội nghề nghiệp,…để có thể phát tiển tư duy cá nhân và góp phần cho xã hội ngày càng phát triển .

## **Mục tiêu**

Như đã đề cập ở trên, tiểu luận “Nghiên cứu về Memory (RAM)” nhằm cung cấp cho nhóm cũng như người đọc về:

* Cái nhìn khách quan nhất về lịch sử ra đời và sự phát triển không ngừng qua từng thời kì của RAM
* Trang bị nhưng kiến thức căn bản nhất về nguyên lí hoạt động, cấu tạo,… của RAM
* Tìm hiểu các loại RAM đang hiện hành.
* Rèn luyện khả năng làm việc nhóm cũng như tạo thói quen nghiên cứu khoa học, dịch thuật, tổng hợp tài liệu,…
* Thiết kế và mô phỏng mô hình RAM đơn giản.

# **CHƯƠNG 2. CƠ SỞ LÍ THUYẾT**

## **2.1. Giới thiệu về Memory [2]**

Bộ nhớ máy tính bao gồm các [bộ nhớ điện tĩnh](https://vi.wikipedia.org/wiki/B%E1%BB%99_nh%E1%BB%9B_%C4%91i%E1%BB%87n_t%C4%A9nh" \o "Bộ nhớ điện tĩnh) (*non-volatile memory*) để lưu trữ được dữ liệu của [máy tính](https://vi.wikipedia.org/wiki/M%C3%A1y_t%C3%ADnh" \o "Máy tính) một cách lâu dài (khi kết thúc một phiên làm việc của [máy tính](https://vi.wikipedia.org/wiki/M%C3%A1y_t%C3%ADnh" \o "Máy tính) thì dữ liệu không bị mất đi), hoặc [bộ nhớ điện động](https://vi.wikipedia.org/wiki/B%E1%BB%99_nh%E1%BB%9B_%C4%91i%E1%BB%87n_t%C4%A9nh" \o "Bộ nhớ điện tĩnh) (*volatile memory*) để lưu dữ liệu tạm thời trong quá trình làm việc của máy tính (khi kết thúc một phiên làm việc của máy tính thì bộ nhớ này bị mất hết [dữ liệu](https://vi.wikipedia.org/wiki/D%E1%BB%AF_li%E1%BB%87u" \o "Dữ liệu)).

Các thiết bị lưu trữ [dữ liệu](https://vi.wikipedia.org/wiki/D%E1%BB%AF_li%E1%BB%87u" \o "Dữ liệu) cho bộ nhớ lâu dài bao gồm: [Đĩa cứng](https://vi.wikipedia.org/wiki/%E1%BB%94_%C4%91%C4%A9a_c%E1%BB%A9ng" \o "Ổ đĩa cứng), [Đĩa mềm](https://vi.wikipedia.org/wiki/%C4%90%C4%A9a_m%E1%BB%81m" \o "Đĩa mềm), [Đĩa quang](https://vi.wikipedia.org/wiki/%C4%90%C4%A9a_quang" \o "Đĩa quang), Băng từ, [ROM](https://vi.wikipedia.org/wiki/B%E1%BB%99_nh%E1%BB%9B_ch%E1%BB%89_%C4%91%E1%BB%8Dc" \o "Bộ nhớ chỉ đọc), các loại bút nhớ...

Các thiết bị lưu trữ [dữ liệu](https://vi.wikipedia.org/wiki/D%E1%BB%AF_li%E1%BB%87u" \o "Dữ liệu) tạm thời trong quá trình làm việc: [RAM](https://vi.wikipedia.org/wiki/RAM" \o "RAM) máy tính, [Cache](https://vi.wikipedia.org/wiki/Cache" \o "Cache)..

**Hình 2.4. Đĩa cứng [7]**

**Hình 2.2. Băng từ [5]**

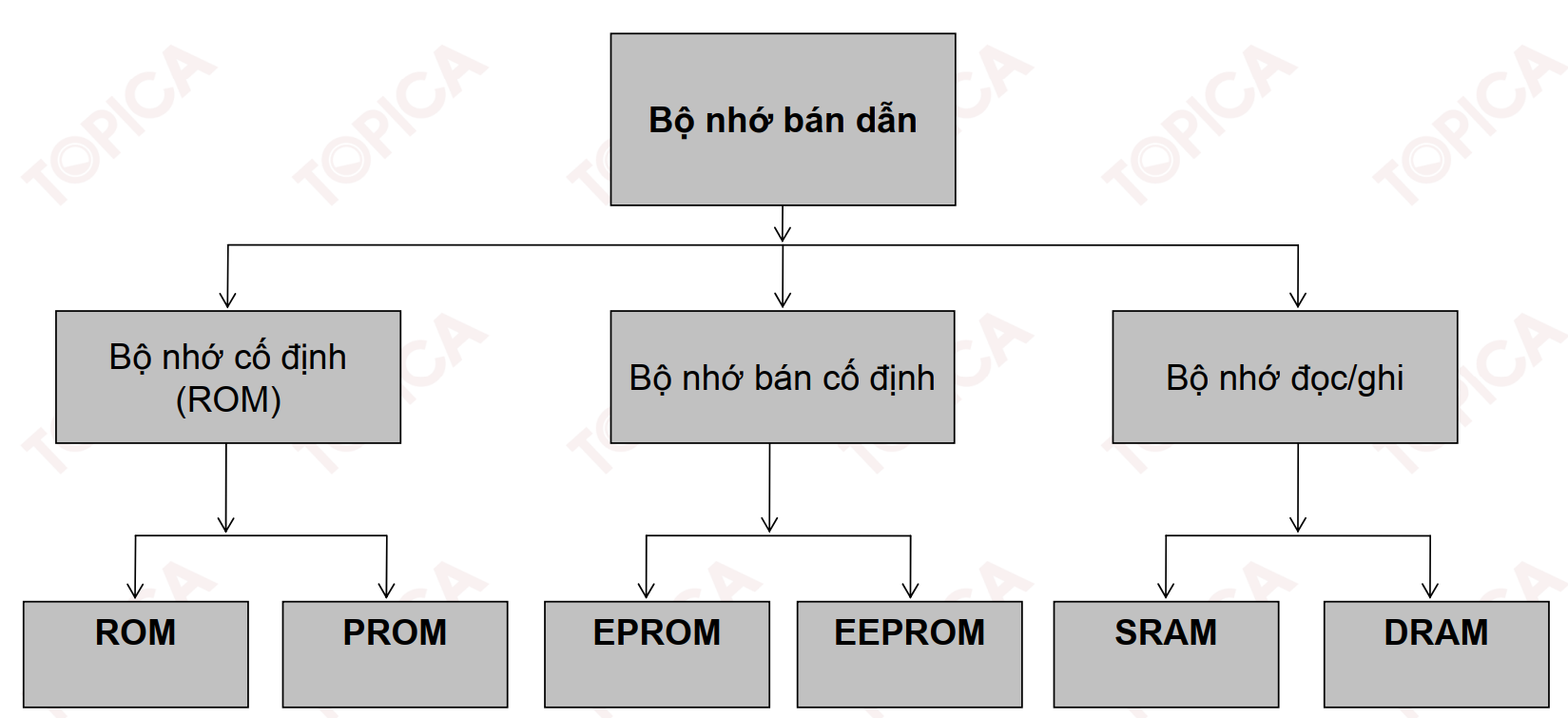
**Hình 2.3. Cache [6]**

**Hình 2.1. CD [4]**

## **2.1.1 Định nghĩa về RAM**

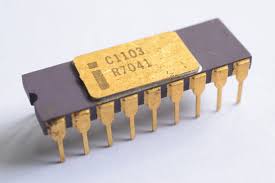
RAM là một bộ nhớ có khả năng đọc ghi và là một tập hợp các ô lưu trữ giống như các thanh ghi kết nối với nhau cùng với các mạch giải mã địa chỉ. Các ô nhớ RAM được tổ chức dạng 8 bits (byte), 16 bits (word), …trên mỗi ô. Có thể truy cập các ô nhớ để tiến hành chuyển giao dữ liệu đến hoặc đi một cách ngẫu nhiên. Vì vậy, nó có tên là bộ nhớ truy cập ngẫu nhiên (Random Access Memory). Tuy nhiên, thông tin lưu trên RAM chỉ là tạm thời và có thể bị mất khi mất điện.

RAM khác với các khối thiết bị bộ nhớ tuần tự (CD-RW, DVD, ổ đĩa cứng,…) khi muốn lấy dữ liệu, phải tiến hành quét cả khối dữ liệu rồi mới tiến hành truy xuất, gây mất nhiều thời gian.



**Hình 2.5. Công nghệ bộ nhớ[3]**

## **2.1.2 Lịch sử về RAM**

Sản phẩm đầu tiên được cho ra đời bởi Intel đó là DRAM vào tháng 10 năm 1970. Những chiếc vi tính thế hệ đầu được sản xuất vào cuối những thập niên 90 được tích hợp các RAM SDR tuy nhiên tốc độ và bộ nhớ cũng rất hạn chế.

Hình 2.6. SDR RAM [8]

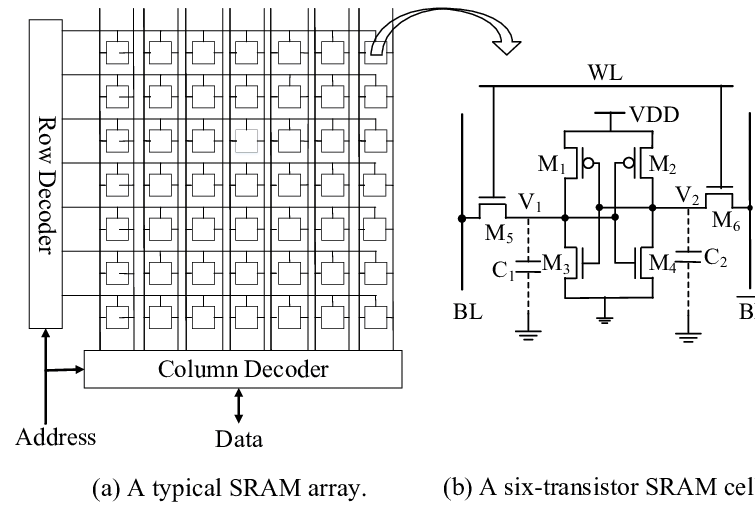
Để cải thiện vấn đề tốc độ cũng như dung lượng, người ta đã tiến hành thiết kế dòng RAM tiếp theo vào đầu những năm 2000 với tên gọi là DDR (Double Data Rate), với tốc độ gấp đôi tốc độ của SDR bằng việc nạp dữ liệu trên cả cạnh lên và cạnh xuống của chu kì xung clock thay vì chỉ nạp 1 lần trong mỗi chu kì xung nhịp, tuy nhiên tốc độ vẫn còn rất chậm.

Với sự ra đời của các thế hệ máy tính mới cùng với hệ điều hành ra đời, làm cho DDR cũng dần trở nên ít được sử dụng và thay thế bằng DDR2 với tốc độ và bộ nhớ lớn hơn. Sau đó là các dòng DDR3 và DDR4 với hàng loạt cải tiến về gia tăng số tùy chọn xung nhịp (clock) và chu kì (timing), giảm điện năng tiêu thụ (power saving) và giảm độ trễ (delay).

## **2.1.3 Phân loại**

Hiện nay, RAM gồm 2 loại chính: RAM tĩnh (Static RAM) và RAM động (Dynamic RAM)

1. **RAM tĩnh (SRAM):** là một loại bộ nhớ sử dụng công nghệ bán dẫn. Từ “tĩnh” nghĩa là bộ nhớ vẫn lưu dữ liệu nếu có điện, không như RAM động cần được nạp lại thường xuyên. [Nguyên lí hoạt động của SRAM](file:///C:\\Users\\Admin\\Desktop\\ảnh%20cho%20Verilog\\SRAM.mp4)-(Ctrl + Click to watch)

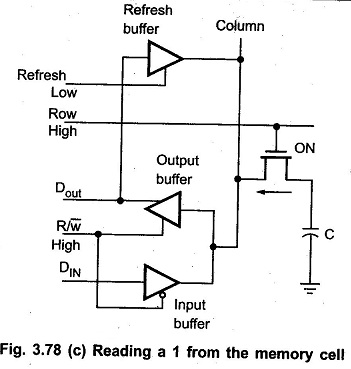


**Hình 2.8. Cấu trúc SRAM và cấu trúc cell**

**Hình 2.7. SRAM [9]**

**Đặc điểm**: SRAM tiêu thụ **điện năng thấp** và cho**tốc độ truy cập nhanh hơn DRAM**. Tuy nhiên, dung lượng bộ nhớ của SRAM thấp và có**chi phí sản xuất cao**. Vì thế, SRAM thường được sử dụng trong: bộ đệm CPU (ví dụ: L1, L2, L3), bộ đệm/bộ nhớ ổ cứng, và bộ chuyển đổi tín hiệu số sang tương tự (DAC) trên thẻ video.

1. **RAM động (DRAM):** cần một nguồn năng lượng “sạc theo định kì” để hoạt động. RAM động lưu mỗi bit dữ liệu trong một tụ điện riêngbiệt trên một mạch tích hợp. Vì thông tin được lưu trữ trên các tụ điện có xu hướng mất trong một khoảng thời gian do sự rò rỉ, nên các tụ điện phải được sạc lại định kỳ để duy trì việc sử dụng dữ liệu. Ngoài ra, RAM động cũng là một loại bộ nhớ được truy cập ngẫu nhiên, và **dữ liệu sẽ mất đị khi nguồn điện bị mất.** [Nguyên lí hoạt động của DRAM](file:///C:\\Users\\Admin\\Desktop\\ảnh%20cho%20Verilog\\DRAM.mp4)-(Ctrl + Click to watch)



**Hình 2.10. Cấu trúc DRAM [12]**

**Hình 2.9. DRAM [11]**

**Đặc điểm:** DRAM có chi phí sản xuất thấp và dung lượng bộ nhớ lớn hơn SRAM. Tuy nhiên, RAM động lại bị hạn chế về tốc độ truy cập chậm hơn và tiêu thụ điện năng cao hơn so với RAM tĩnh, nên thường được sử dụng trong bộ nhớ hệ thống, bộ nhớ đồ họa video.

1. **RAM động đồng bộ (SDRAM):** SDRAM (**Synchronous Dynamic RAM**) là loại Ram được phát triển từ RAM động (DRAM), để **hoạt động đồng bộ với đồng hồ CPU**. Nó chờ tín hiệu đồng hồ trước khi phản hồi đầu vào dữ liệu (như giao diện người dùng chẳng hạn). Tương tự, với chức năng không đồng bộ, nó sẽ đáp ứng ngay lập tức với dữ liệu đầu vào.

**Hình 2.11. SDRAM [13]**

**Đặc điểm**: hoạt động đồng bộ với CPU để **xử lý song song các lệnh chồng chéo**, nghĩa là khả năng nhận (đọc) một lệnh mới trước khi lệnh trước đó được giải quyết hoàn toàn ( chức năng ghi thông tin). Việc xử lý một lệnh đọc và một lệnh ghi trên mỗi chu kỳ xung nhịp, làm cho**tốc độ truyền và hiệu năng CPU tổng thể được cao hơn.**

1. **RAM động đồng bộ tốc độ dữ liệu đơn (SDR SDRAM):** viết tắt của **Single Data Rate Synchronous Dynamic RAM,** chức năng của loại ram này, chính là cách mà **Bộ nhớ xử lý "một" lệnh đọc và "một" lệnh ghi trên mỗi chu kỳ xung nhịp**

****

**Hình 2.12. SDR SDRAM [14]**

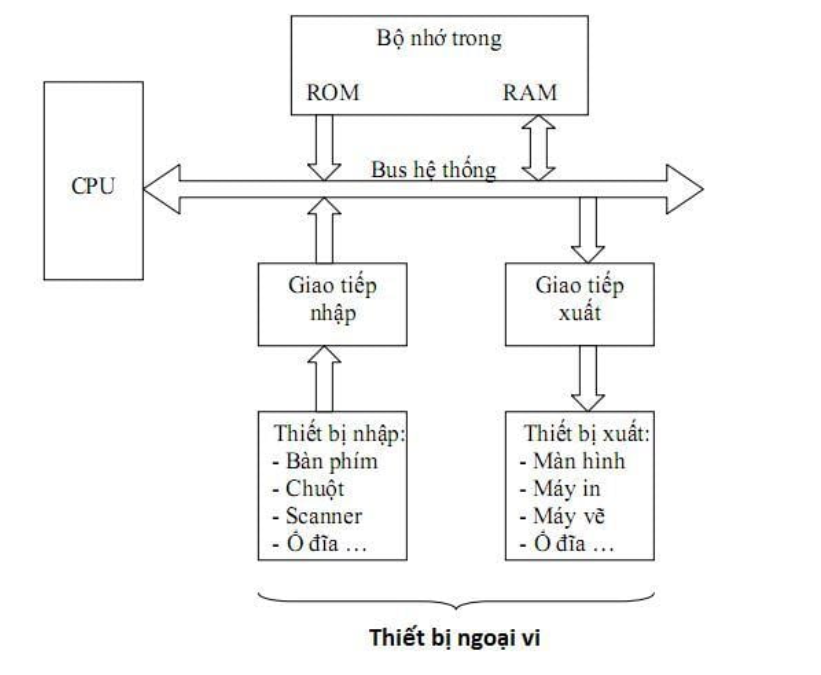
1. **RAM động đồng bộ tốc độ dữ liệu kép (DDR SDRAM):** DDR SDRAM (**Double Data Rate Synchronous Dynamic RAM**), cơ bản **hoạt động giống như SDR SDRAM** nhưng có **tốc độ nhanh gấp đôi.** Nghĩa là DDR SDRAM có khả năng **xử lý "hai" hướng dẫn đọc và "hai" lần ghi trên mỗi chu kỳ xung nhịp**



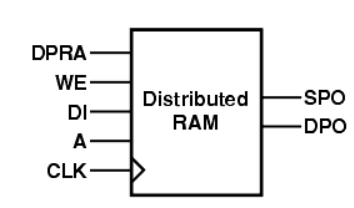
**Hình 2.13. DDR SDRAM [15]**

# **CHƯƠNG 3: CÁC BƯỚC THIẾT KẾ VÀ MÔ PHỎNG HỆ THỐNG**

**3.1 Vị tri RAM trong hệ thống máy tính**



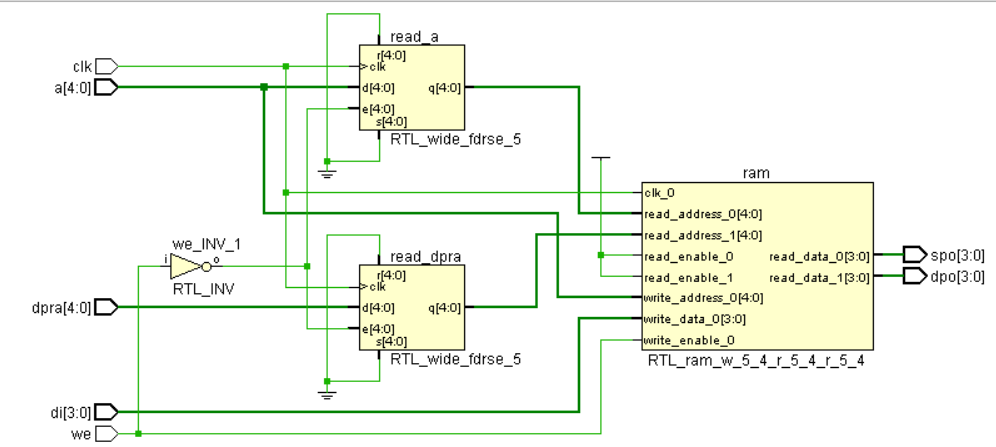
## **3.2 Sơ Đồ của bản thiết kế.**



**Hình 3.1 Sơ đồ 1In\_2Out của Single-port RAM[1]**

|  |  |
| --- | --- |
| **IN\_OUT PIN** | **Mô tả** |
| **clk** | **Xung clock tác động cạnh lên** |
| **we** | **Cho phép đọc dữ liệu đồng bộ ở mức cao** |
| **di** | **Ngõ vào dữ liệu** |
| **a** | **Ghi địa chỉ/ đọc 4 bit địa chỉ thấp** |
| **dpra** | **Đọc địa 4 bit địa chỉ cao** |
| **spo** | **Ngõ ra 4 bit thấp** |
| **dpo** | **Ngõ ra 4 bit cao** |

**Bảng trên mô tả các chân I/O và chức năng của chúng trong RAM**

**Hình 3.2 Sơ đồ chi tiết bên trong khối RAM**

Như trong hình trên, RAM bao gồm các khối sau:

1**. Khối read\_a**: gồm các flip flop D 5bit dùng để cấp phát địa chỉ khi có yêu cầu đọc dữ liệu ra khi có xung clk tác động cạnh lên sau đó đưa đến ngõ ra spo.

2.**Khối read\_dpra**: gồm các flip flop D 5bit dùng để cấp phát địa chỉ khi có yêu cầu đọc dữ liệu ra khi có xung clk tác động cạnh lên sau đó đưa đến ngõ ra dpo.

3.**Khối ram**: Dùng để đọc/ghi dữ liệu khi có yêu cầu.

## **3.3 Thiết kế chi tiết**

Các tín hiệu điều khiển thao tác đối với RAM bao gồm: Tín hiệu CLK cho phép quá trình RAM đọc - ghi . Nếu WE – write enable bằng 1 nếu cần ghi dữ liệu vào RAM. WE – write enable bằng 0 nếu cần đọc dữ liệu ra RAM.

### **3.3.1 RAM**

***Module chính để tạo khối Single Port Ram: [1]***

module In1\_Out2(

input clk,

input we,

input [4:0] a, //khai báo ngõ vào a 5 bit

input [4:0] dpra, // khai báo ngõ vào dpra 5 bit

input [3:0] di, // khai báo ngõ vào di 4 bit

output [3:0] spo, // khai báo ngõ ra spo 4 bit

output [3:0] dpo ); // khai báo ngõ ra dpo 4 bit

reg [3:0] ram [31:0]; // khai báo ram dạng thanh ghi mảng 2 chiều 4bit x 32bit

reg [4:0] read\_a; // khai báo thanh ghi lưu trữ địa chỉ read\_a 5bit

reg [4:0] read\_dpra; // khai báo thanh ghi lưu trữ địa chỉ read\_dpra 5bit

always @(posedge clk)

begin

if (we) //we=1 cho phép ghi

ram[a] <= di; // ghi dữ liệu di vào ram tại địa chỉ a

else //we=0 cho phép đọc

begin

read\_a<=a; //gán giá trị tại địa chỉ a cho read\_a

read\_dpra<=dpra; // gán giá trị tại địa chỉ dpra cho read\_dpra

end

end

assign spo = ram[read\_a]; // xuất dữ liệu từ Ram tại địa chỉ read\_a ra spo

assign dpo = ram[read\_dpra]; // xuất dữ liệu từ Ram tại địa chỉ read\_dpra ra dpo

endmodule

### 3.3.2. **Mô phỏng qua testbench:**

***Code module testbench***

module Testbench;

module ad;

*// Inputs*

reg clk;

reg we;

reg [4:0] a;

reg [4:0] dpra;

reg [3:0] di;

// Outputs

wire [3:0] spo;

wire [3:0] dpo;

// Instantiate the Unit Under Test (UUT)

In1\_Out2 uut (

.clk(clk),

.we(we),

.a(a),

.dpra(dpra),

.di(di),

.spo(spo),

.dpo(dpo)

);

initial begin

// Initialize Inputs

clk = 1'b1;

forever #10 clk = ~clk; //tạo xung clk với chu kỳ 20ns

end

initial begin

we = 1; //cho phép ghi

a = 0; // tại vị trí a=0

di = 8; // ghi dữ liệu bằng di =8 hay 1000

#20; delay 20ns

a = 1; // tại vị trí a =1

di = 9; // ghi dữ liệu bằng di =9 hay 1001

#20;

a = 2; // tại vị trí a =2

di = 10; // ghi dữ liệu bằng di =10 hay 1010

#20;

a = 3; // tại vị trí a =3

di = 11; // ghi dữ liệu bằng di =11 hay 1011

#20;

// READ //

a = 0; //gán a =0

dpra = 1; //gán dpra=1

we = 0; // cho phép đọc

#20;

a = 30; //gán a = 30

dpra = 2; // gán dpra=2

#20;

a = 2; // gán a =2

dpra = 3; //gán dpra =3

#20;

a = 3; // gán a =3

dpra = 0; // gán dpra =0

#20;

// Wait 100 ns for global reset to finish

end

// Add stimulus here

endmodule

# 

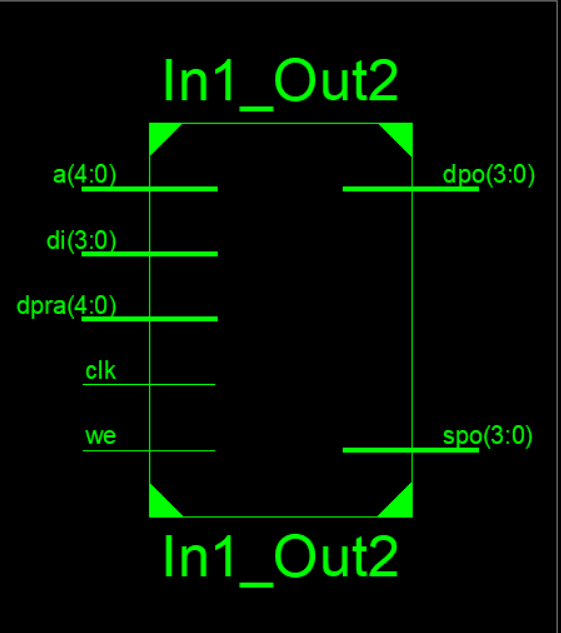
# **CHƯƠNG 4: ĐÁNH GIÁ QUA TESTBENCH**

## **4.1 Mô hình Testbench tổng quát**

### **4.1.1 Giới thiệu**

*Testbench là gì:*

* Hiểu một cách đơn giản, testbench là một phương tiện dùng để kiểm tra logic của một thiết kế cụ thể.
* Hiểu theo ngôn ngữ RTL, testbench là một module, dùng để cấp các tín hiệu đầu vào cho module được test, và thu thập các giá trị đầu ra của module đó, đồng thời, kiểm chứng được tính đúng đắn của logic từ các giá trị đầu ra đó.



**Hình 4.1 Mô hình RTL của RAM khi code**

Phần testbench sẽ có phần điều khiển là những tín hiệu nối với input của RAM, ở hình trên là CLK và We. Phần này cung cấp các trường hợp đầu vào theo yêu cầu. Giá trị đầu ra sẽ được lấy thông qua phần đọc ghi tín hiệu của các khối bên trong RAM, gồm những tín hiệu nối với đầu ra của q[7:0].

## **4.2 Các thành phần bên trong testbench**

Về cấu trúc code, testbench bao gồm những phần cơ bản như sau:

module ad;

*// Inputs*

reg clk;

reg we;

reg [4:0] a;

reg [4:0] dpra;

reg [3:0] di;

// *điều khiển mức đọc-ghi , giá trị we cao là cho phép ghi còn thấp cho phép đọc*

// Outputs

wire [3:0] spo;

wire [3:0] dpo;

// Instantiate the Unit Under Test (UUT)

In1\_Out2 uut (

.clk(clk),

.we(we),

.a(a),

.dpra(dpra),

.di(di),

.spo(spo),

.dpo(dpo)

);

initial begin

// Initialize Inputs

clk = 1'b1;

forever #10 clk = ~clk; //tạo xung clk với chu kỳ 20ns

end

initial begin

we = 1; //cho phép ghi

a = 0; // tại vị trí a=0

di = 8; // ghi dữ liệu bằng di =8 hay 1000

#20; delay 20ns

a = 1; // tại vị trí a =1

di = 9; // ghi dữ liệu bằng di =9 hay 1001

#20;

a = 2; // tại vị trí a =2

di = 10; // ghi dữ liệu bằng di =10 hay 1010

#20;

a = 3; // tại vị trí a =3

di = 11; // ghi dữ liệu bằng di =11 hay 1011

#20;

// READ //

a = 0; //gán a =0

dpra = 1; //gán dpra=1

we = 0; // cho phép đọc

#20;

a = 30; //gán a = 30

dpra = 2; // gán dpra=2

#20;

a = 2; // gán a =2

dpra = 3; //gán dpra =3

#20;

a = 3; // gán a =3

dpra = 0; // gán dpra =0

#20;

// Wait 100 ns for global reset to finish

end

// Add stimulus here

endmodule

**Khai báo các tín hiệu của phần điều khiển RAM.**

    - Tất cả các tín hiệu điều khiển ở testbench sẽ được khai báo ở dạng reg, để có thể được gán giá trị để truyền vào bên trong module test.

    - Tín hiệu q thường được khai báo dạng wire, đây là phần tử thụ động, chỉ nhận giá trị từ output của module được test. Các phần khác sẽ dùng giá trị từ tín hiệu này để đánh giá, so sánh và đưa ra kết luận về tính đúng đắn của chức năng.

* **Phần gọi module được test. Phần này hay còn gọi là "instance"**

    - Cấu trúc của instance được viết như trên.

    - Tên các tín hiệu ở testbench có thể giống với bên trong test module.

* **Phần khởi tạo tín hiệu điều khiểu.**

    - Tại thời điểm t = 0, các simulator hoặc phần mềm mô phỏng sẽ không hiểu được giá trị của các tín hiệu điều khiển này là bao nhiêu. Thông thường nó sẽ gán cho các tín hiệu này giá trị "X". Đây là giá trị thể hiện sự không xác định của tín hiệu. Nếu như không khởi tạo, các giá trị X này sẽ lan truyền vào bên trong module được test, sẽ gây ra các hoạt động không mong muốn.

    - Để loại bỏ sự không xác định trên, chúng ta tốt nhất nên khởi tạo các đầu vào này theo một giá trị cụ thể là 0, hoặc 1.

* **Phần thực thi chính của simulation.**

    - Đây là nơi chứa các sự vận hành của việc kiểm tra.

    - Thông thường, người chạy simulation phải có một kế hoạch hoặc là một ý tưởng test nào đó rõ ràng. Sau đó, các tín hiệu điều khiển liên quan sẽ được viết theo trình tự thời gian tương ứng với ý tưởng đó.

## **4.3 Kết quả Simulation**

Kết quả của simulation có thể tồn tại ở 3 loại thông tin sau:

**a, Waveform:**

    - Đây là dạng thông tin cơ bản nhất mà bạn nhận được khi chạy simulation. Bạn phải hiểu rõ các giá trị của tín hiệu theo từng thời điểm. Việc phân tích waveform đa phần là bằng mắt thường. Do đó, đòi hỏi phải có kỹ năng nhìn nhận waveform một các tinh tế.

**b, Các thông báo của Tool trong quá trình chạy simulation:**

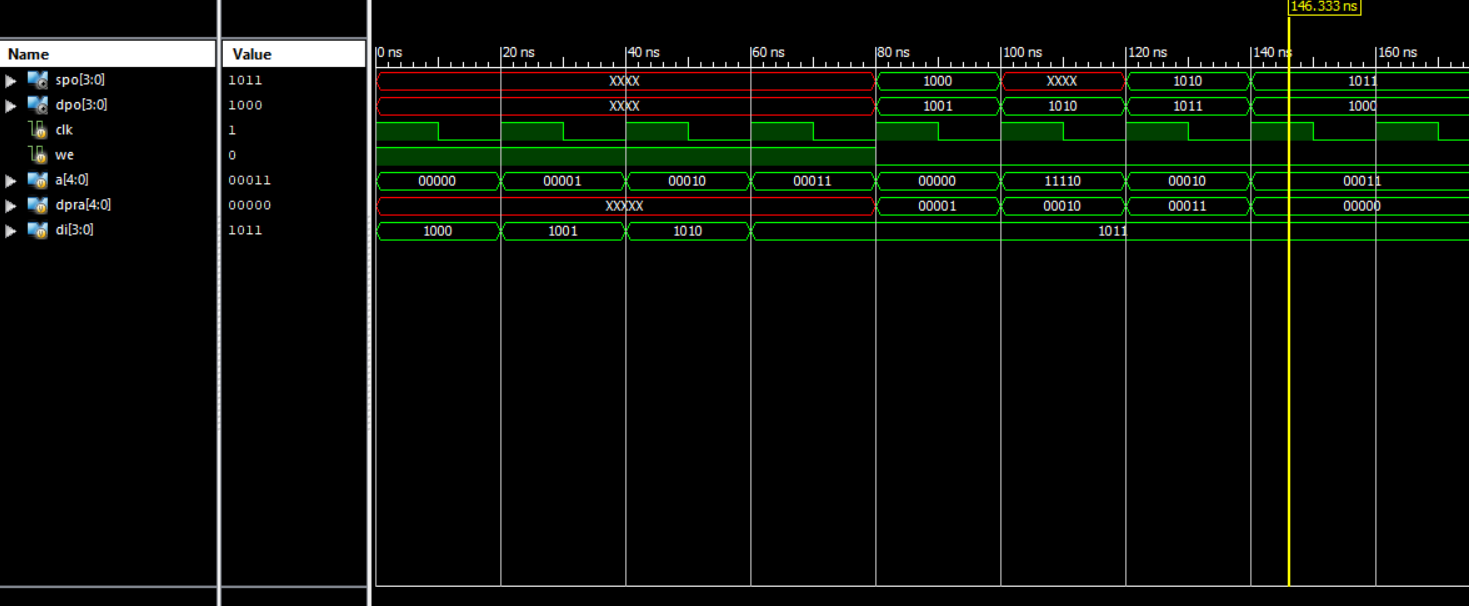
    - Các tool dùng để simulation thường sẽ có phần compile dành cho testbench. Do đó, nó có thể thông báo cho người dùng các vấn đề của code, syntax v.v. Người chạy simulation phải đọc hiểu và đánh giá cũng như có các động tác khắc phục lỗi, cảnh báo của tool.

**c, Các loại output khác từ người dùng:**

    - Các loại này là tùy biến từ mong muốn của người dùng.

    - Nó có thể bao gồm các thông báo dạng text, hoặc một file lưu các giá trị đầu ra, hoặc tín hiệu nào đó cụ thể.

Đây là waveform từ bước simulation của testbench phía trên:



**Hình 4.2 waveform từ simulation**

Từ module thiết kế, các bạn có thể xác nhận lại logic của module có chính xác như những gì mình mong muốn hay chưa.

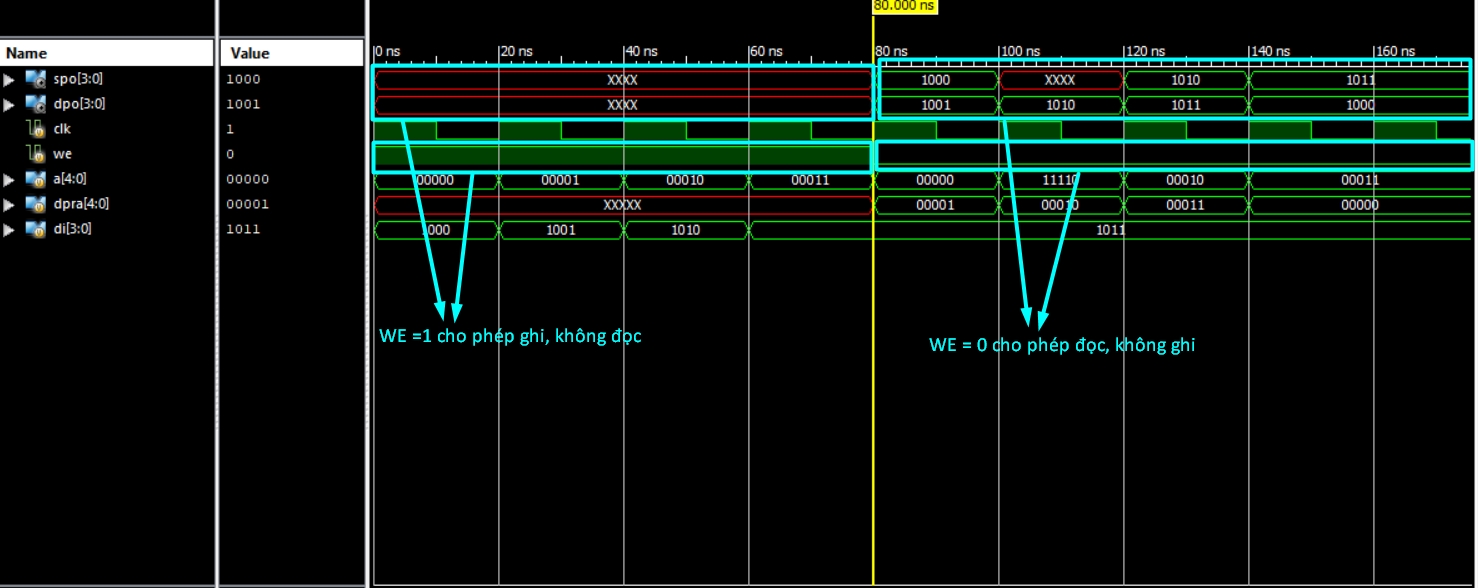
Đây chỉ là ví dụ để các bạn hiểu như thế nào là testbench và instance 1 design vào testbench.

## **4.4 Các trường hợp Testcase**

Với các tín hiệu đã khởi tạo ở phần testbench ta có các trường hợp đọc ghi dữ liệu vào bộ nhớ Ram theo quy luật như các quá trình sau:

## **4.4.1 Quá trình cho phép đọc ghi dữ liệu.**

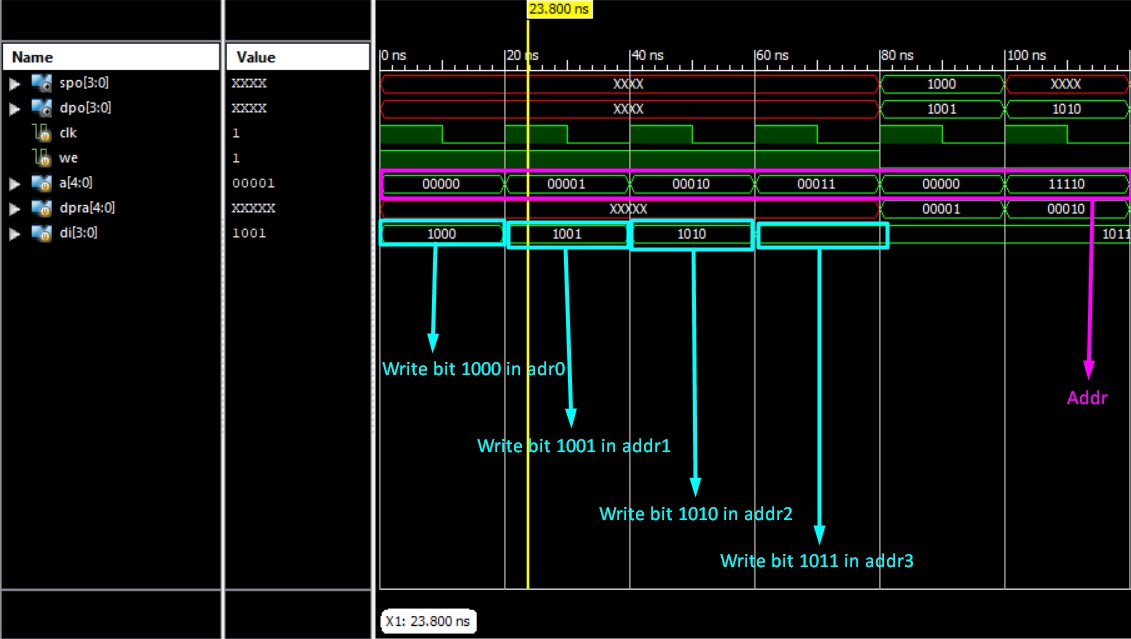
Dựa vào tín hiệu we mà ta có 2 trường hợp đọc và ghi dữ liệu. Khi we ở mức cao dữ liệu được ghi còn we mức thấp thì cho phép đọc dữ liễu ở ngõ ra, cụ thể 2 trường hợp như sau:

****

***Hình 4.3 Mô phỏng quá trình cho phép đọc-ghi dữ liệu***

* Theo hình trên ta thấy bên trái đường màu vàng tín hiệu điều khiển we đang ở mức cao chỉ cho phép ghi dữ liệu vào RAM còn quá trình đọc bị cấm. Tiếp đến bên phải đường màu vàng, lúc này tín hiệu điều khiển we ở mức thấp cho phép quá trình đọc dữ liệu được diễn ra.

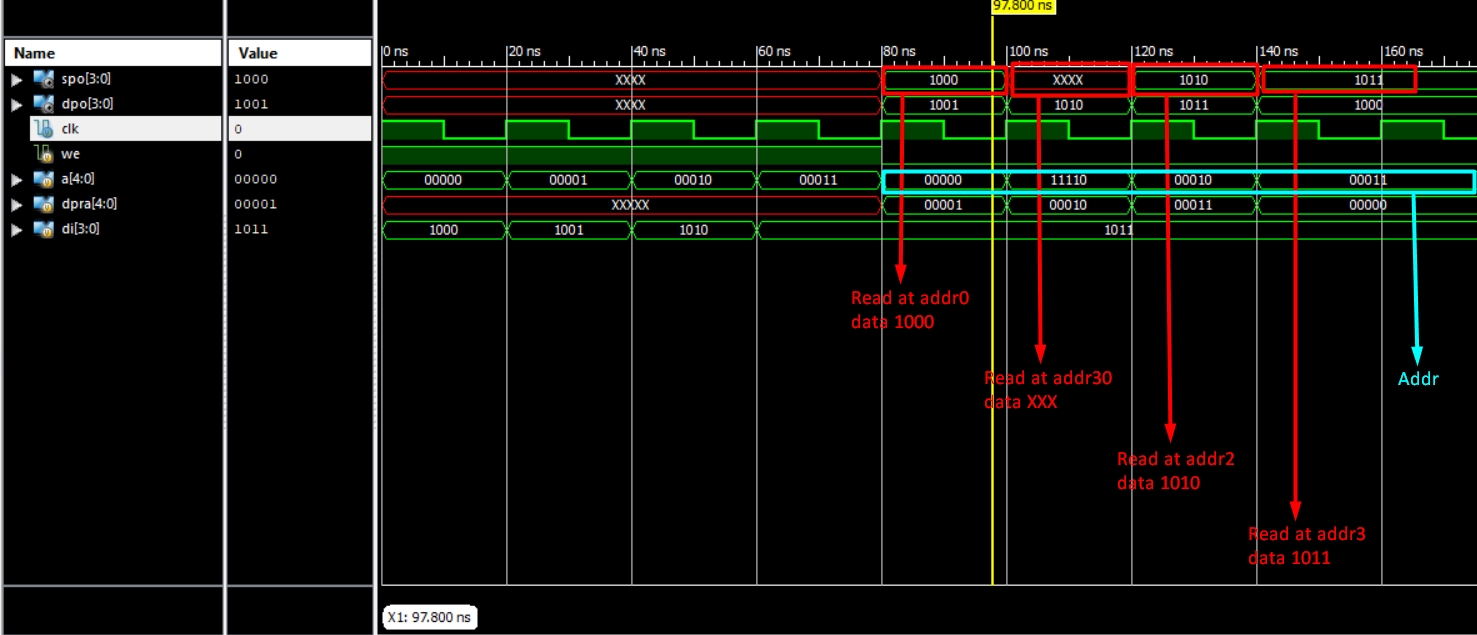
## **4.4.2 Ghi dữ liệu vào RAM**

****

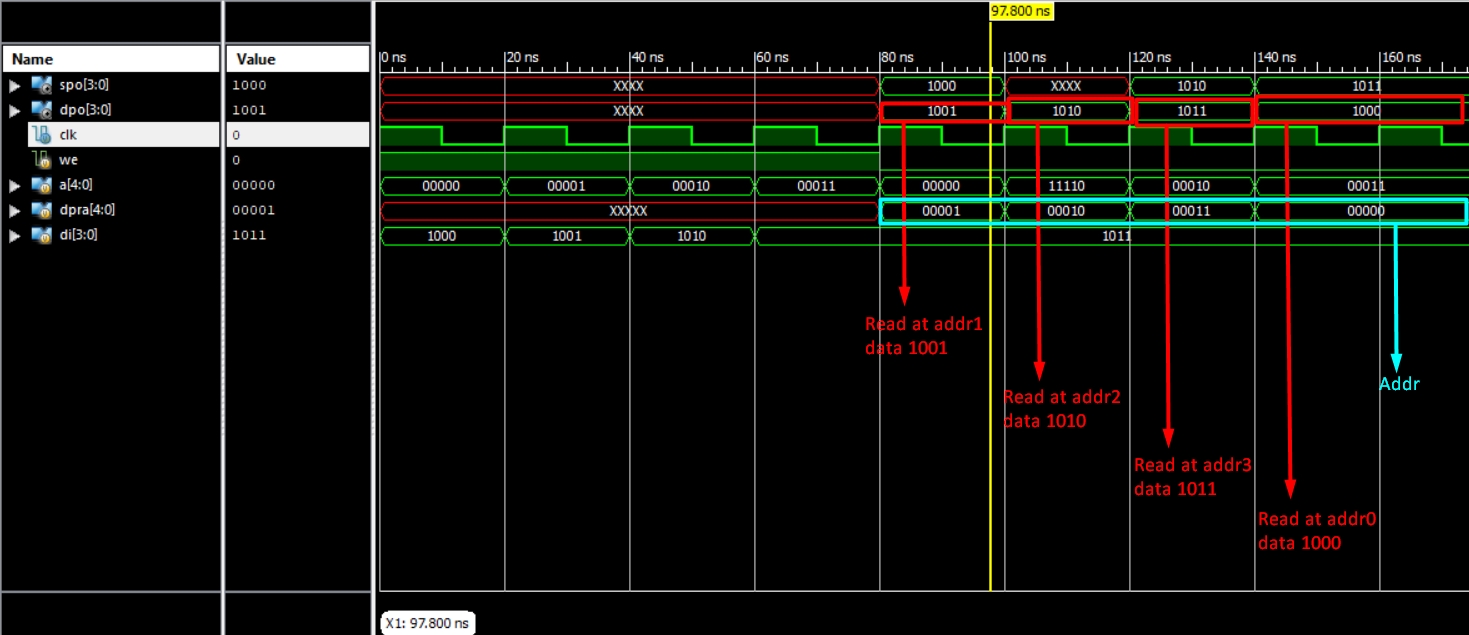
***Hình 4.4 Mô phỏng quá trình ghi dữ liệu vào RAM***

* Theo hình vẽ trên tại vị trí thanh màu vàng ta thấy có xung CLK tác động cạnh lên và tín hiệu cho phép ghi we ở mức cao nên lúc này RAM thực hiện chức năng ghi dữ liệu vào. Giá trị ghi vào được là “1000” tại địa chỉ “00000”. Tương tự tại những vị trí xung CLK tiếp theo quá trình ghi tiếp tục diễn ra, giá trị ghi vào “1001” tại địa chỉ “00001”, “1010” tại địa chỉ “00010”, ….

## **4.4.3 Đọc dữ liệu từ RAM**



***Hình 4.5 Mô phỏng quá trình đọc dữ liệu ra từ RAM***

******

***Hình 4.6 Mô phỏng quá trình đọc dữ liệu ra từ RAM***

* Theo hình vẽ trên tại vị trí thanh màu vàng ta thấy có xung CLK tác động cạnh lên và tín hiệu cho phép đọc we ở mức thấp nên lúc này RAM thực hiện chức năng đọc dữ liệu ra
* Với chức năng một ghi và 2 ngõ đọc dữ liệu thì khi ta truy xuất dữ liệu tại địa chỉ a [4:0] thì ta nhận các giá trị tương ứng tại thanh ghi mà ta đã ghi cụ thể tại địa chỉ “00000” dữ liệu ra là “1000” , tại “11110” ta không ghi ở địa chỉ này nên giá trị xuất ra sẽ là XXX. Tương tự với địa chỉ “00010” data sẽ là “1010” và cuối cùng là tại địa chỉ “00011” dữ liệu là “1011”, như vậy đã đúng với phần testbench mà ta đã giao nhiệm vụ đọc tại a. Đối với tại địa chỉ dpra cũng tương tự như đọc ở địa chỉ tại a, và dữ liệu cũng tương đương với dữ liệu ta đã ghi.

## **4.5. Kết luận và hướng phát triển**

Từ kết quả mô phỏng trên ta thấy rằng giá trị và địa chỉ mà RAM ghi vào thì trùng với giá trị và địa chỉ RAM đọc ra. Vậy bài thiết kế single RAM đã cho ra kết quả đúng.

Hiện tại, nhóm chúng em chỉ mới có thể thiết kế được RAM ở mức cơ bản nhất là đọc và ghi. Trong tương lai, hi vọng nhóm chúng em có thể thiết kế được các RAM khó hơn và phức tạp hơn.

*Tài liệu tham khảo*

**[1]**<http://www.csit-sun.pub.ro/courses/Masterat/Xilinx%20Synthesis%20Technology/toolbox.xilinx.com/docsan/xilinx4/data/docs/xst/hdlcode14.html?fbclid=IwAR0qJ01axpfZ_9Bq96ZU5nApmPMpkqO0UY9t9nXaMRSocGhUdGZ42mIXke8>

**[2]**<https://blogchiasekienthuc.com/dan-cong-nghe/ram-ddr-ddr-2-ddr-3-ddr-4-khac-nhau-nhu-the-nao.html/amp?fbclid=IwAR3lSPoRxd2Pv-JrP9WOJ3wMEX0W2OzMvhjJJ02rix7YXZJJ1TJAblz3asM>

**[3]**[http://eldata11.topica.edu.vn/HocLieu/TH102/PDF\_Slide/TH102\_Bai3\_v1.0014103207.pdf\](http://eldata11.topica.edu.vn/HocLieu/TH102/PDF_Slide/TH102_Bai3_v1.0014103207.pdf\\)

[4] <https://maytinhtrangia.com/dia-cd-neo-viet-nam-loc-50-chiec/>

[5]<https://www.google.com/imgres?imgurl=https%3A%2F%2Fcdn.stereo.vn%2Fuploads%2F2016%2F07%2Fbang-tu-bang-vhs.jpg&imgrefurl=https%3A%2F%2Fstereo.vn%2Ftin-tuc%2Fxu-huong%2Fvinh-biet-bang-tu-bang-vcr--4150.html&tbnid=IQ_UWLWyMjxz2M&vet=12ahUKEwj-lYmKw_DvAhUMvJQKHXzxD24QMygDegUIARC9AQ..i&docid=cJK3Nh1XLsv1OM&w=660&h=371&q=b%C4%83ng%20t%E1%BB%AB&ved=2ahUKEwj-lYmKw_DvAhUMvJQKHXzxD24QMygDegUIARC9AQ>

[6] <https://bizfly.vn/techblog/web-cache.html>

[7]https://phongvu.vn/o-cung-hdd-pc-seagate-barracuda-1tb-3-5-sata-st1000dm010-s1201747.html

[8]<https://maybovanphong.com/ram-may-tinh-la-gi-lich-su-phat-trien-ram-may-tinh.html>

[9] <https://en.wikipedia.org/wiki/Static_random-access_memory>

[10] <https://www.researchgate.net/figure/Simplified-architecture-of-an-SRAM-array-and-a-six-transistor-SRAM-cell_fig1_331453568>

[11] <https://www.mccourier.com/global-dynamic-random-access-memory-dram-market-2021-consumer-needs-samsung-electronics-co-ltd-sk-hynix-inc-micron-technology-inc-nanya-technology-corporation/>

[12] <https://www.eeeguide.com/dynamic-ram/>

[13] <https://thegioilaptopvn.com/tim-hieu-chi-tiet-ve-sdram/>

[14] <https://www.wikidata.org/wiki/Q6116127>

[15]<https://en.bmstu.wiki/index.php?title=DDR_SDRAM&mobileaction=toggle_view_desktop>