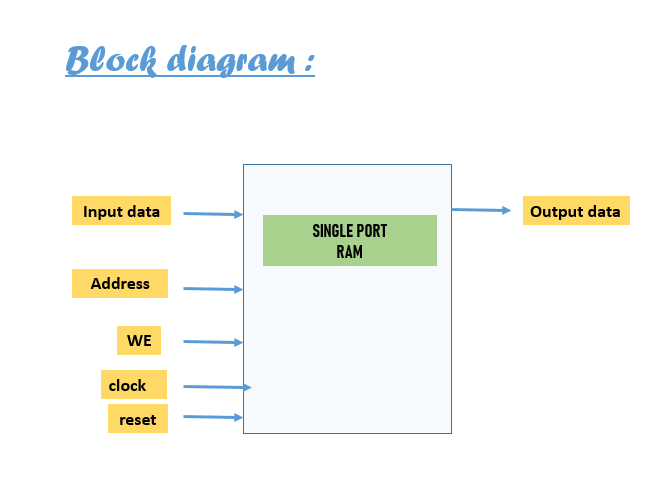
SINGLE PORT RAM

Single Port RAM (Random Access Memory) is a type of computer memory that has a single data input/output port. This means that data can only be accessed from one side of the RAM chip.

Single Port RAM chips are available in various sizes, from a few kilobits to several gigabits, and they can be organized as different types of memory, such as static RAM (SRAM) or dynamic RAM (DRAM). They are typically faster than dual-port or multi-port RAM because they have fewer input/output pins and less circuitry.



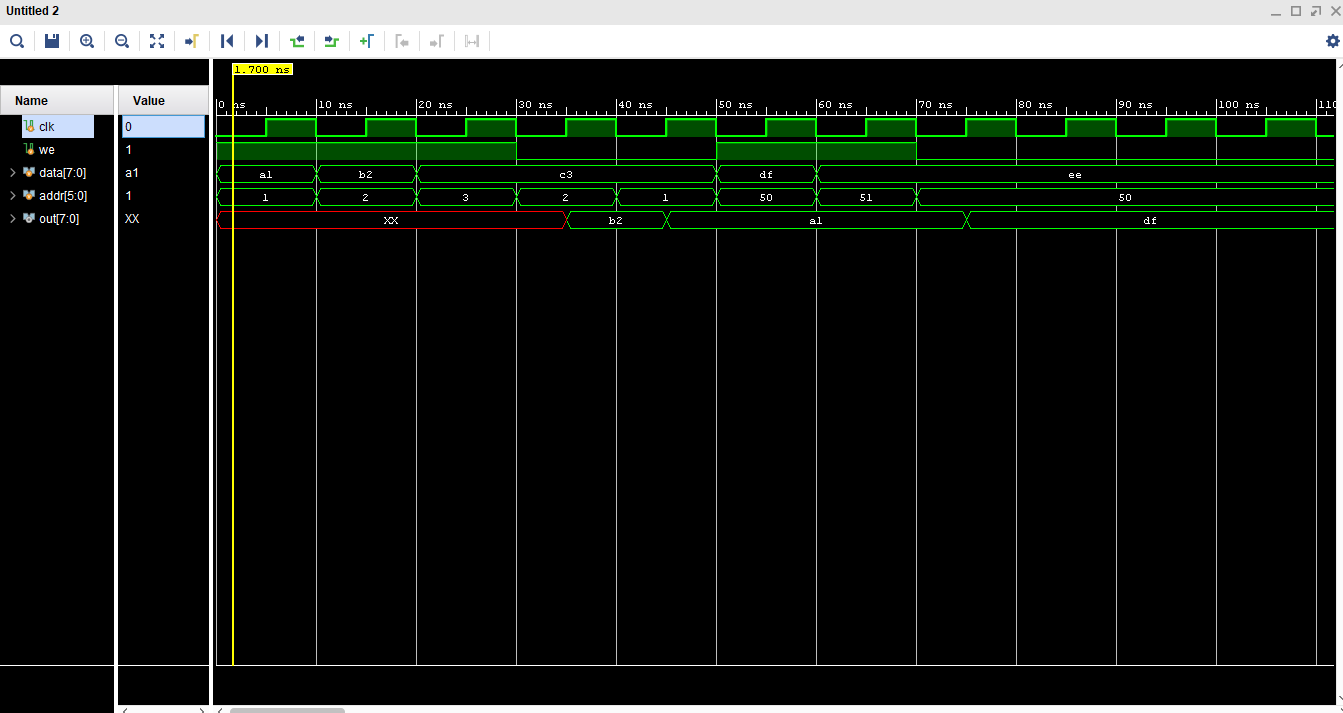
Verilog code :

|  |  |
| --- | --- |
|  | `timescale 1ns / 1ps |
|  | // 8\*64 Bit RAM |
|  | module single\_port\_ram( |
|  | input [7:0] data, //data input |
|  | input [5:0] addr, //address |
|  | input we, //write enable |
|  | input clk, // clock |
|  | output [7:0] out //output |
|  | ); |
|  |  |
|  | reg [7:0]ram[63:0]; /\* 64 memory locations, requires 6-address lines to represent the location and each location has 8 bit of data \*/ |
|  | reg [5:0]addr\_reg; //address register |
|  |  |
|  | always@(posedge clk) |
|  | begin |
|  | if(we) |
|  | ram[addr]<=data; |
|  | else |
|  | addr\_reg <= addr; |
|  | end |
|  | assign out=ram[addr\_reg]; |
|  | endmodule |

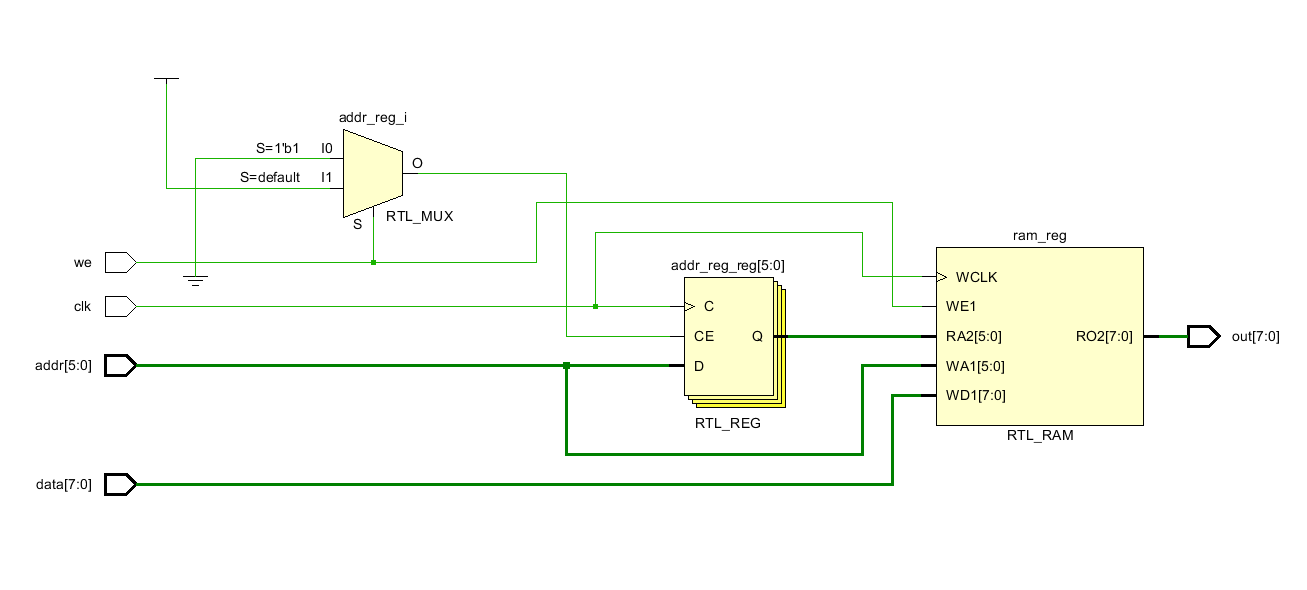
Test Bench :

|  |
| --- |
|  |
|  | `timescale 1ns / 1ps |
|  | module single\_port\_ram\_tb( ); |
|  | reg [7:0]data; |
|  | reg [5:0]addr; |
|  | reg we, clk; |
|  | wire [7:0]out; |
|  |  |
|  | single\_port\_ram uut ( .data(data), .addr(addr), .we(we), .clk(clk), .out(out)); |
|  |  |
|  | always #5 clk=~clk; |
|  | initial |
|  | begin |
|  | clk=0; |
|  | we=1; |
|  |  |
|  | data=8'ha1; |
|  | addr=6'd1; |
|  | #10; |
|  |  |
|  | data=8'hb2; |
|  | addr=6'd2; |
|  | #10; |
|  |  |
|  | data=8'hc3; |
|  | addr=6'd3; |
|  | #10; |
|  |  |
|  | we=0; |
|  | addr=6'd2; |
|  | #10; |
|  |  |
|  | addr=6'd1; |
|  | #10; |
|  |  |
|  | we=1; |
|  | data=8'hdf; |
|  | addr=6'd50; |
|  | #10; |
|  |  |
|  | data=8'hee; |
|  | addr=6'd51; |
|  | #10; |
|  |  |
|  | we=0; |
|  | addr=6'd50; |
|  |  |
|  | #1000; $finish; |
|  | end |
|  |  |
|  | endmodule |

Simulation Ouput

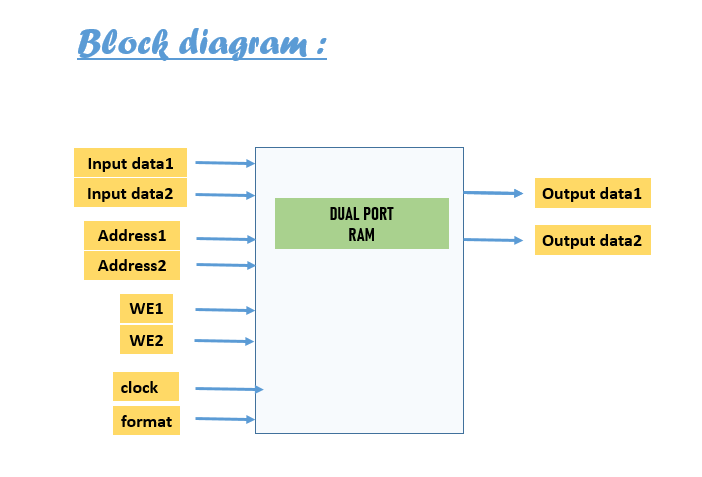


RTL :



DUAL PORT RAM

Dual Port RAM (Random Access Memory) is a type of computer memory that has two independent data input/output ports, which allows two different devices or circuits to read from and write to the RAM at the same time. This makes it useful in applications where two or more devices need to access the same memory simultaneously.



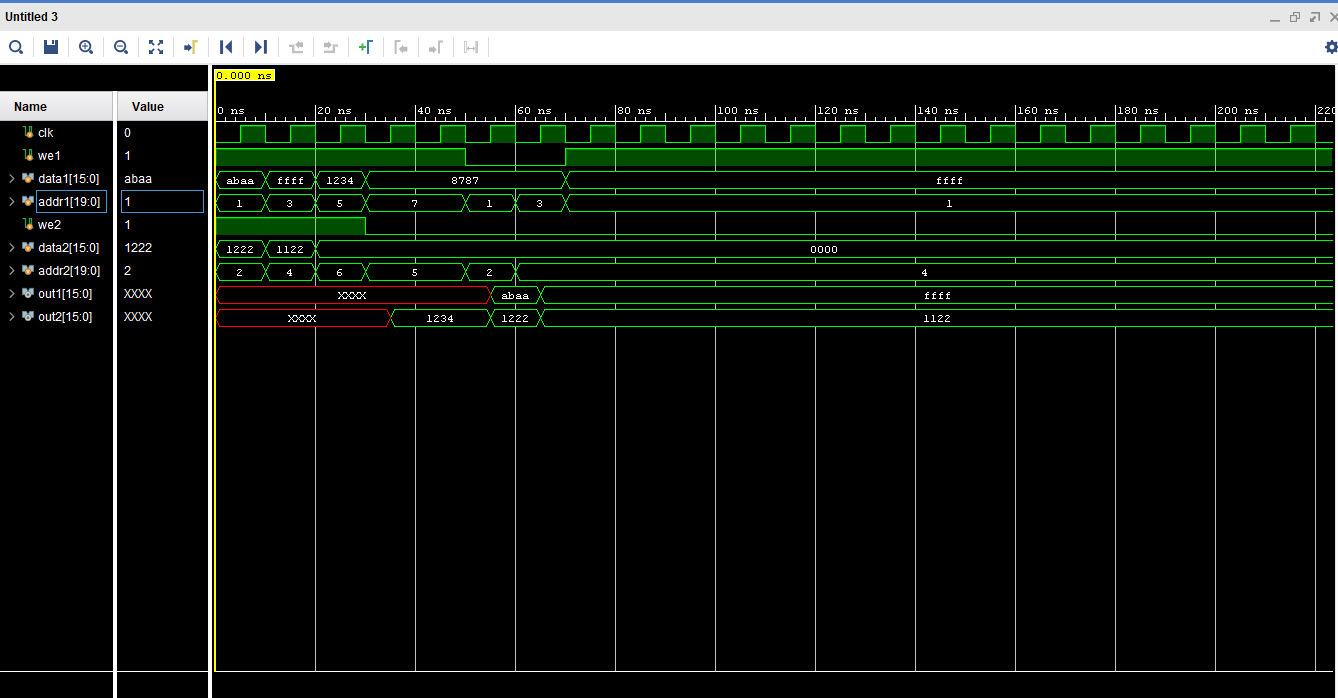
Verilog code:

|  |
| --- |
|  |
|  |
|  | `timescale 1ns / 1ps  //16-data lines and 20-address lines |
|  | //16\*1024 RAM |
|  | module dual\_port\_ram( |
|  | input [15:0]data1, data2, //input data |
|  | input [19:0]addr1, addr2, //addresses |
|  | input we1, we2, //write enables |
|  | input clk, //clock |
|  | output reg[15:0]out1, out2 //output |
|  | ); |
|  |  |
|  | reg [15:0]ram[1023:0]; /\* 1024 memory locations, so we require 20 address lines to represent the 1024 locations and each location has 16-bit data \*/ |
|  |  |
|  | always@(posedge clk) |
|  | if(we1) |
|  | ram[addr1]<=data1; |
|  | else |
|  | out1<=ram[addr1]; |
|  |  |
|  | always@(posedge clk) |
|  | if(we2) |
|  | ram[addr2]<=data2; |
|  | else |
|  | out2<=ram[addr2]; |
|  | endmodule |

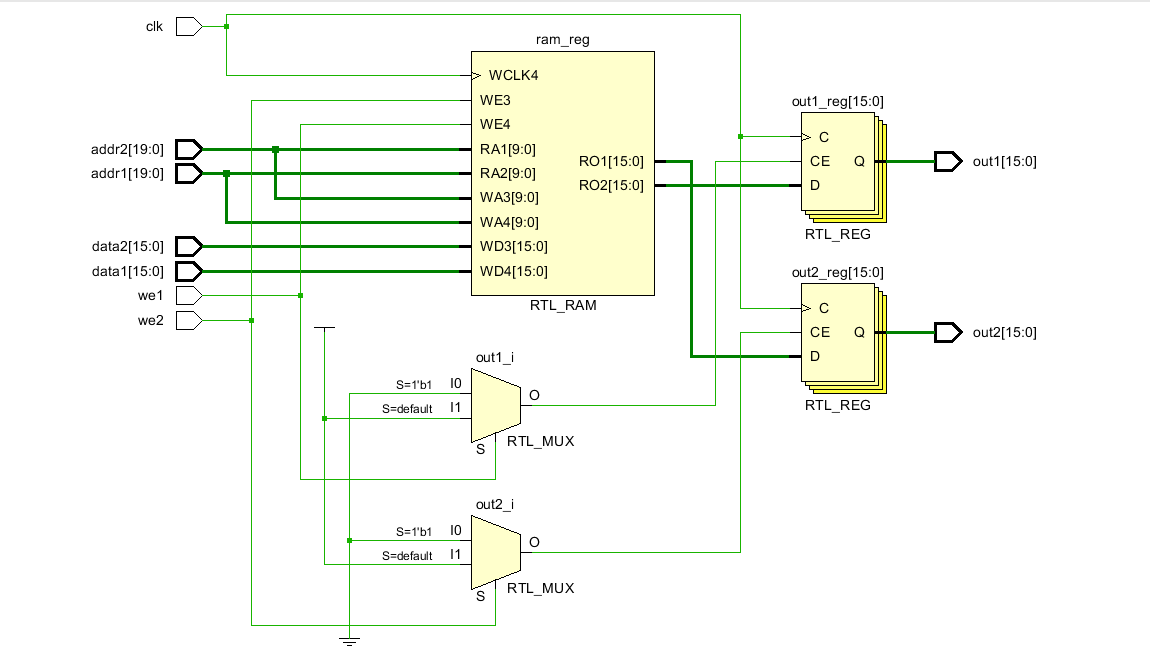
Test Bench:

|  |
| --- |
|  |
|  | `timescale 1ns / 1ps |
|  | module dual\_port\_ram\_tb( ); |
|  | reg [15:0] data1, data2; |
|  | reg [19:0] addr1, addr2; |
|  | reg we1, we2, clk; |
|  | wire [15:0] out1, out2; |
|  |  |
|  | dual\_port\_ram uut( .data1(data1), .data2(data2), .addr1(addr1), .addr2(addr2), .we1(we1), .we2(we2), .clk(clk), .out1(out1), .out2(out2)); |
|  |  |
|  |  |
|  | always #5 clk=~clk; |
|  | initial begin |
|  | clk=0; |
|  | we1=1; we2=1; |
|  |  |
|  | data1=16'habaa; addr1=20'd1; |
|  | data2=16'h1222; addr2=20'd2; |
|  | #10; |
|  |  |
|  | data1=16'hffff; addr1=20'd3; |
|  | data2=16'h1122; addr2=20'd4; |
|  | #10; |
|  |  |
|  | data1=16'h1234; addr1=20'd5; |
|  | data2=16'h0000; addr2=20'd6; |
|  | #10; |
|  |  |
|  | we2=0; |
|  | data1=16'h8787; addr1=20'd7; |
|  | addr2=20'd5; |
|  | #10; |
|  |  |
|  | we2=0; |
|  | data1=16'h8787; addr1=20'd7; |
|  | addr2=20'd5; |
|  | #10; |
|  |  |
|  | we1=0; |
|  | addr1=20'd1; |
|  | addr2=20'd2; |
|  | #10; |
|  |  |
|  | addr1=20'd3; |
|  | addr2=20'd4; |
|  | #10; |
|  |  |
|  | we1=1; |
|  | data1=16'hffff; addr1=20'd1; |
|  | #10; |
|  |  |
|  | #1000; $finish; |
|  | end |
|  | endmodule |

Simulation Output:

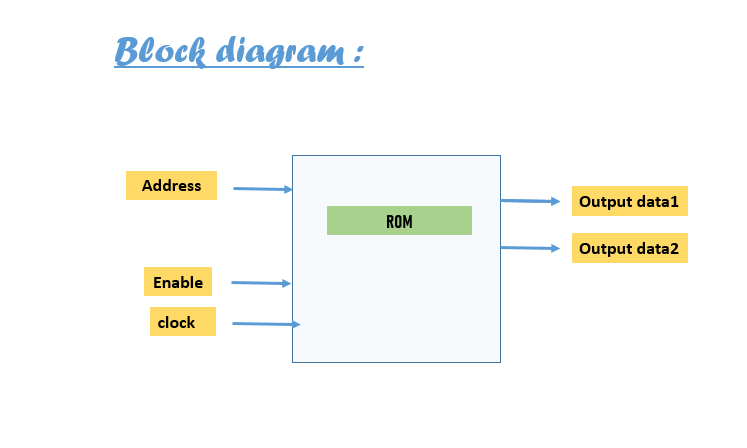


RTL



ROM

Read-Only Memory (ROM) is a type of computer memory that stores data that cannot be modified or erased. The data in ROM is non-volatile, meaning it is retained even when the power to the computer is turned off.



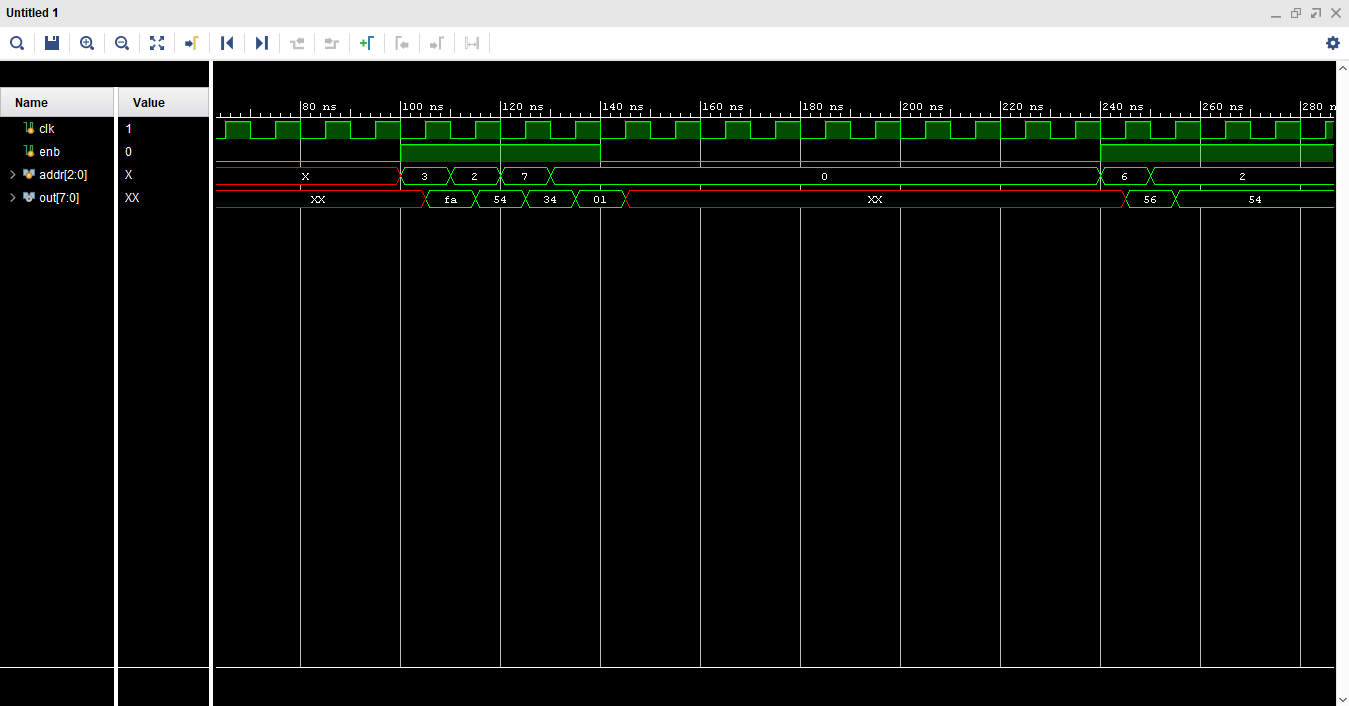
Verilog code:

|  |
| --- |
|  |
|  | `timescale 1ns / 1ps  //8\*8 bit rom |
|  | module rom( |
|  | input clk, enb, |
|  | input [2:0]addr, |
|  | output reg [7:0]out |
|  | ); |
|  |  |
|  | reg[7:0]rom[7:0]; /\* 8 memory locations each having 8-bit data, so it requires 3-address lines to represent \*/ |
|  |  |
|  | always@(posedge clk) |
|  | Begin |
|  | if(enb) |
|  | out<=rom[addr]; |
|  | Else |
|  | out<=8'hxx; |
|  | End |
|  |  |
|  | initial begin |
|  | rom[0]=8'h01; |
|  | rom[1]=8'haa; |
|  | rom[2]=8'h54; |
|  | rom[3]=8'hfa; |
|  | rom[4]=8'he5; |
|  | rom[5]=8'h98; |
|  | rom[6]=8'h56; |
|  | rom[7]=8'h34; |
|  | end |
|  | endmodule |

Test Bench:

|  |
| --- |
|  |
|  | `timescale 1ns / 1ps |
|  | module rom\_tb( ); |
|  | reg clk, enb; |
|  | reg [2:0]addr; |
|  | wire [7:0]out; |
|  |  |
|  | rom uut (.clk(clk), .enb(enb), .addr(addr), .out(out)); |
|  |  |
|  | always #5 clk=~clk; |
|  | initial begin |
|  | clk=0; enb=0; |
|  | #100; |
|  |  |
|  | enb=1; |
|  | addr=3; |
|  | #10; |
|  |  |
|  | addr=2; |
|  | #10; |
|  |  |
|  | addr=7; |
|  | #10; |
|  |  |
|  | addr=0; |
|  | #10; |
|  |  |
|  | enb=0; |
|  | #100; |
|  |  |
|  | enb=1; |
|  | addr=6; |
|  | #10; |
|  |  |
|  | addr=2; |
|  | #10; |
|  |  |
|  | #1000; $finish; |
|  | end |
|  | endmodule |

Simulation Output:



RTL:

