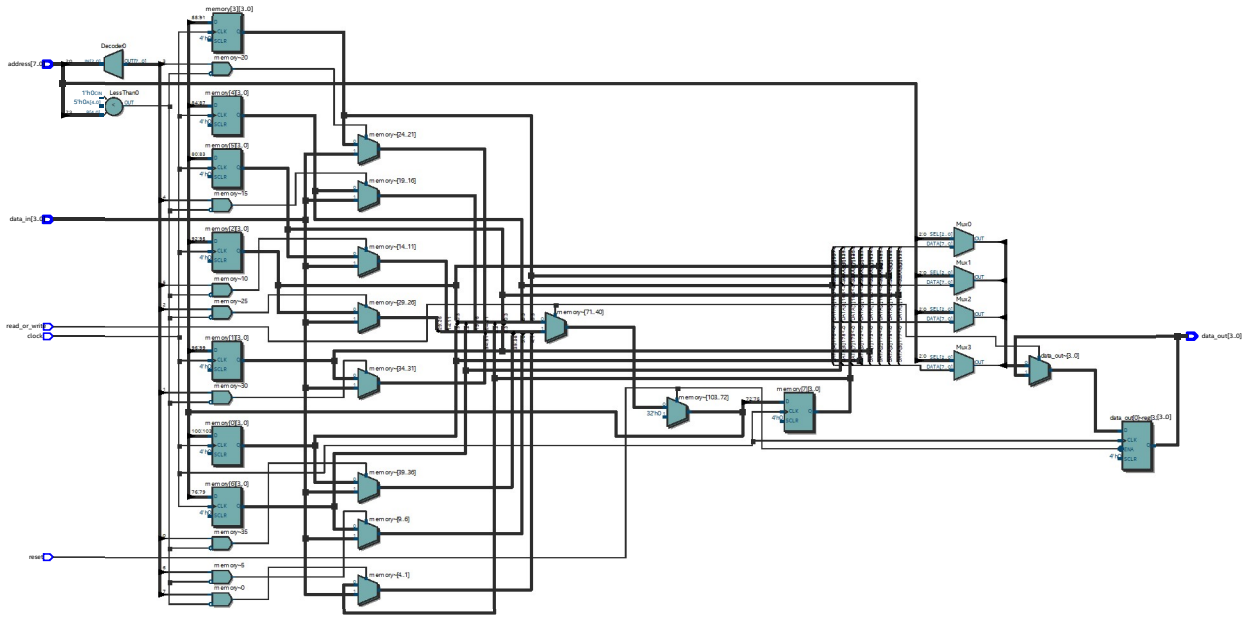
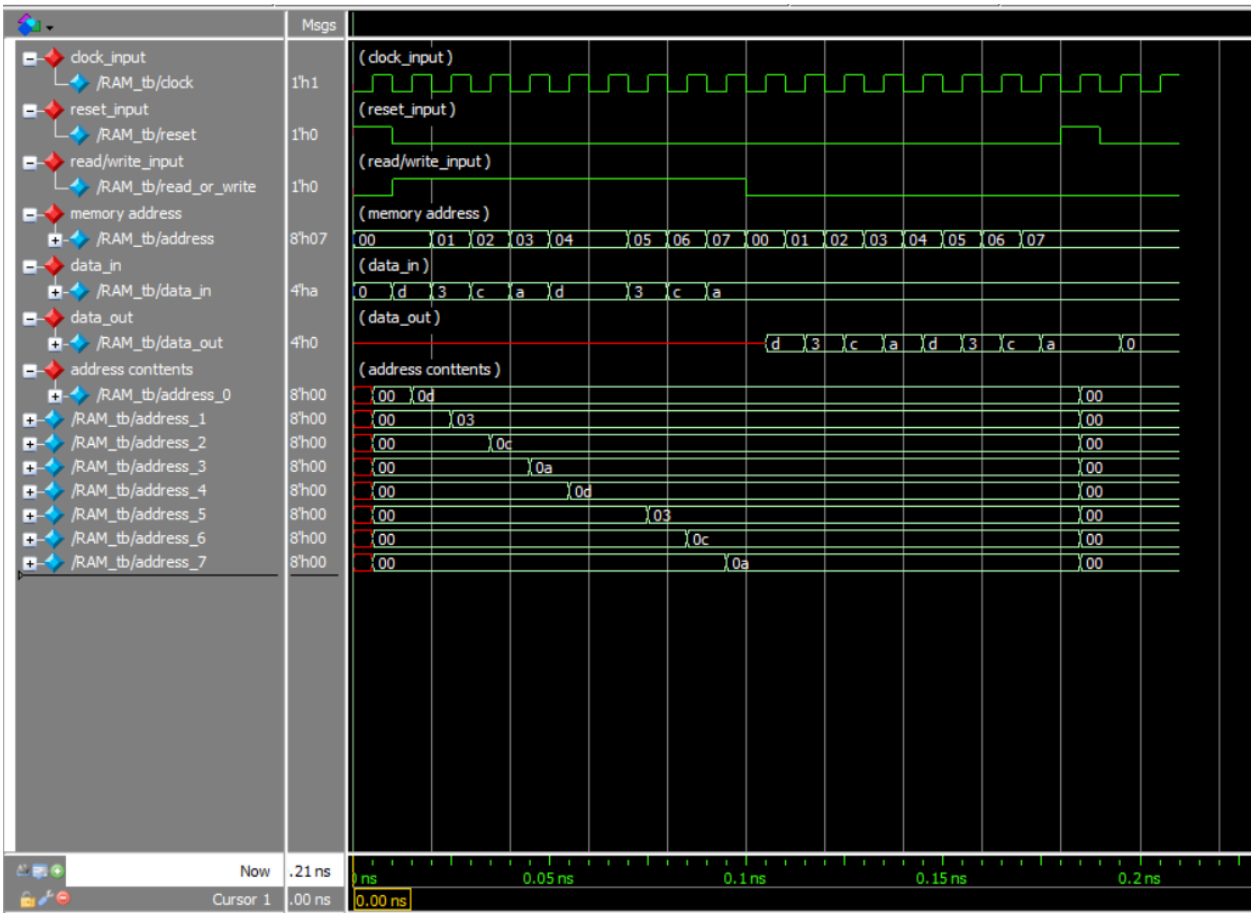
```
1  /*-----------------------------------------------------------------------------------------
   ---------
2  Name Lamin Jammeh
3  CLass: EE417 Summer 2024
4  Lesson 11 HW Question 2
5  Group: Ron Kalin/ Lamin Jammeh
6  Project Description: This is a RAM module where write takes priority when a read_or_write
   signal is high
7  -----------------------------------------------------------------------------------------
   --------*/
8
9  module RAM  #(parameter memory_height = 8,
10               parameter data_width    = 4)
11             (
12              input                                  clock,            //one clock for read
   and write
13              input                                  reset,            //clears memory
14              input         [memory_height -1:0]     address,          //address line
15              input                                  read_or_write,    //enables read or write
16              input         [data_width -1:0]        data_in,          //input data port
17              output  reg   [data_width -1:0]        data_out          //output data port
18              );
19
20 //internal register
21 reg      [data_width -1:0] memory [memory_height -1:0];              // temp register to
   store memory data
22 integer k;
23
24 //Transition logic
25 always   @(posedge clock)
26    if (reset)
27       for (k = 0; k < memory_height; k = k+1)
28          memory[k] <= 4'b0000;               //all address lines in memory are cleared @
   reset high
29    else if (read_or_write)
30          memory[address] <= data_in;      //priority to write if we recieve both read and
   write signal at the same time
31    else if (~read_or_write)
32          data_out    <= memory[address];  // when read_or_write is low data_out=requested
   address line
33 endmodule
```


Decoder0
LessThan0
Mux0
Mux1
Mux2
Mux3
address[7..0]
data_in[3..0]
read_or_write
clock
reset
data_out[3..0]

```
/*-------------------------------------------------------------------------------------------------
---------
Name Lamin Jammeh
CLass: EE417 Summer 2024
Lesson 11 HW Question 2
Group: Ron Kalin/ Lamin Jammeh
Project Description: Testtbench for the RAM Module
-------------------------------------------------------------------------------------------------
--------*/

module RAM_tb ();

parameter   memory_height = 8;
parameter   data_width    = 4;

//define the wires and registers testbench
reg                             clock, reset;
reg                             read_or_write;
reg   [memory_height -1:0]      address;
reg   [data_width    -1:0]      data_in;
wire  [data_width    -1:0]      data_out;

//probe the each address line to determine the content
wire  [memory_height -1:0]      address_0, address_1, address_2, address_3, address_4,
address_5, address_6, address_7;

//instantiate the Unit under test UUT
RAM UUT (clock, reset, address, read_or_write, data_in, data_out);

//assign the probes for the address to the address to lines in the UUT
assign   address_0   = UUT.memory[0];
assign   address_1   = UUT.memory[1];
assign   address_2   = UUT.memory[2];
assign   address_3   = UUT.memory[3];
assign   address_4   = UUT.memory[4];
assign   address_5   = UUT.memory[5];
assign   address_6   = UUT.memory[6];
assign   address_7   = UUT.memory[7];

//intantiate the clock cycle
initial
   begin
      clock   = 0;
      forever #5  clock = ~clock;
   end

//initialize the RAM
initial
   begin
      reset         = 1;
      read_or_write = 0;
      address       = 0;
      data_in       = 4'b0000;



//load the data to the memory
      #10;
               reset         = 1'b0;
               address       = 0;                //load address 0
               data_in       = 4'b1101;
               read_or_write = 1'b1;             // write to memory

      #10;
               data_in       = 4'b0011;
               address       = 1;                //load address 1
               read_or_write = 1'b1;             // write to memory

      #10;
```

```
67                  data_in       = 4'b1100;
68                  address       = 2;                  //load address 2
69                  read_or_write = 1'b1;               // write to memory
70
71          #10;
72                  data_in       = 4'b1010;
73                  address       = 3;                  //load address 3
74                  read_or_write = 1'b1;               // write to memory
75          #10;
76                  address       = 4;                  //load address 4
77                  data_in       = 4'b1101;
78                  read_or_write = 1'b1;               // write to memory
79
80          #20;
81                  data_in       = 4'b0011;
82                  address       = 5;                  //load address 5
83                  read_or_write = 1'b1;               // write to memory
84
85          #10;
86                  data_in       = 4'b1100;
87                  address       = 6;                  //load address 6
88                  read_or_write = 1'b1;               // write to memory
89
90          #10;
91                  data_in       = 4'b1010;
92                  address       = 7;                  //load address 7
93                  read_or_write = 1'b1;               // write to memory
94
95    //read from data from memory
96          #10;
97                  address = 0;
98                  read_or_write = 1'b0;
99          #10;
100                 address = 1;
101                 read_or_write = 1'b0;
102         #10;
103                 address = 2;
104                 read_or_write = 1'b0;
105         #10;
106                 address = 3;
107                 read_or_write = 1'b0;
108         #10;
109                 address = 4;
110                 read_or_write = 1'b0;
111         #10;
112                 address = 5;
113                 read_or_write = 1'b0;
114         #10;
115                 address = 6;
116                 read_or_write = 1'b0;
117         #10;
118                 address = 7;
119                 read_or_write = 1'b0;
120   //check the reset function
121         #10;
122                reset = 1;
123         #10;
124                reset = 0;
125         #20    $stop;
126      end
127   endmodule
```


Msgs
clock_input
/RAM_tb/clock
1'h1
reset_input
/RAM_tb/reset
1'h0
read/write_input
/RAM_tb/read_or_write
1'h0
memory address
/RAM_tb/address
8'h07
data_in
/RAM_tb/data_in
4'ha
data_out
/RAM_tb/data_out
4'h0
address conttents
/RAM_tb/address_0
8'h00
/RAM_tb/address_1
8'h00
/RAM_tb/address_2
8'h00
/RAM_tb/address_3
8'h00
/RAM_tb/address_4
8'h00
/RAM_tb/address_5
8'h00
/RAM_tb/address_6
8'h00
/RAM_tb/address_7
8'h00
(clock_input)
(reset_input)
(read/write_input)
(memory address)
(data_in)
(data_out)
(address conttents)
Now
.21 ns
Cursor 1
.00 ns
0.05 ns
0.1 ns
0.15 ns
0.2 ns
0.00 ns