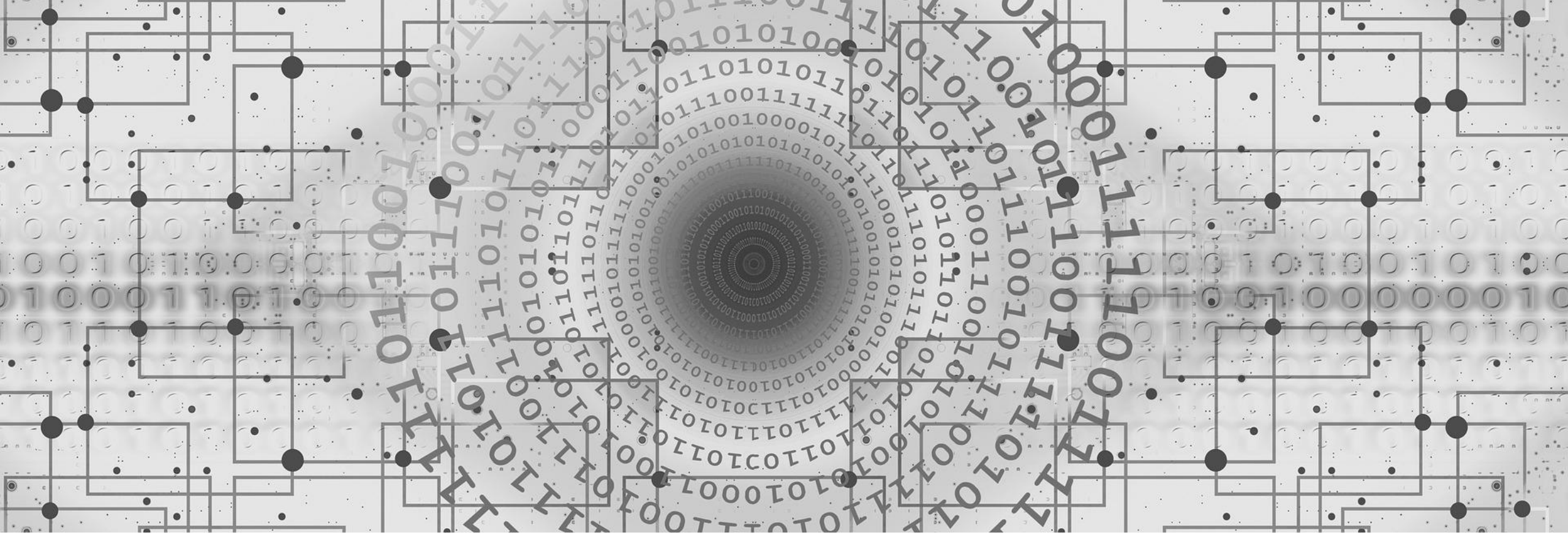


SystemVerilog Bi-directional Port

ECE-111 Advanced Digital Design Projects

Vishal Karna

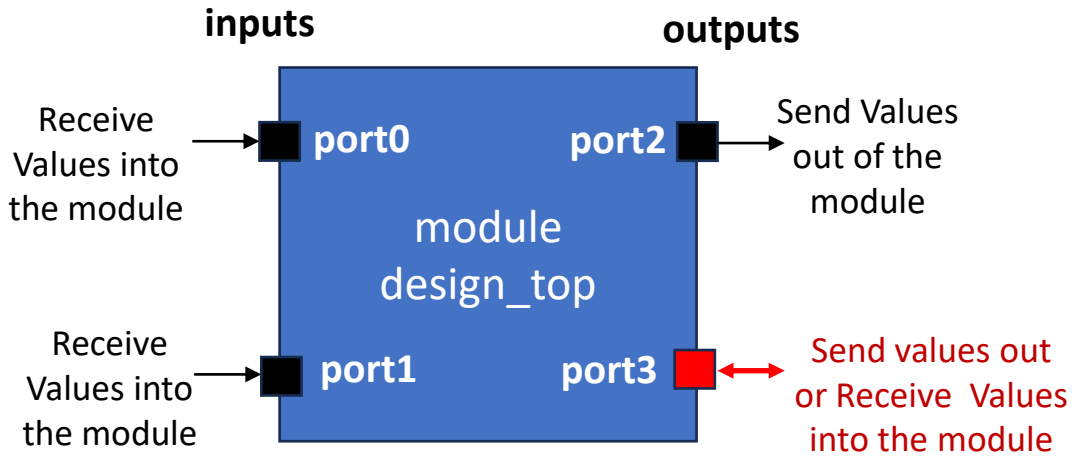
UC San Diego
JACOBS SCHOOL OF ENGINEERING
Electrical and Computer Engineering

Bi-directional Port

What is a Bidirectional Port ?

- Bi-directional ports allow data to flow both into and out of the SystemVerilog module
- Bi-directional port is declared as an **inout** port in primary port list of a module

```
module design_top (
  input logic port0;
  input logic port1;
  output logic port2;
  inout logic port3;  //Bi-directional Port
 );
  .....
   <Implementation>
  .....
endmodule: design_top
```



Rules for inout port :

- **inout** ports **cannot** be a reg type as they cannot store values
- **inout** ports can only be declared as a net type i.e. wire type
- **inout** ports can also be declared as a logic type
- Parent module can only connect to an inout port of a child module using a wire type net !

SystemVerilog Implementation of a Simple RAM with inout data port

```
module ram_with_bidir_data_bus #(  // RAM module implementation with bi-directional data port
  parameter DATA_WIDTH=8, //width of data bus
  parameter ADDR_WIDTH=3  //width of addresses buses
)(
  input logic clk, // clock
  input logic wr_en, //  '1' indicates write to internal memory and '0' indicates read to internal memory
  input logic [ADDR_WIDTH-1:0] addr, // index for write or read operation to memory
  inout logic [DATA_WIDTH-1:0] data_io // if wr_en=1, data_io becomes as input port otherwise it is output port
);
 // create two-dimensional array to represent a memory
 localparam index = 2**ADDR_WIDTH; // indicates 2³ = 8
 logic [DATA_WIDTH-1:0] memory[index-1:0];

 always_ff@(posedge clk) begin
  // when wr_en =1, write incoming values on data_io port is written into memory at address location
  if(wr_en) begin
     memory[addr]  <= data_io;
  end
 end

 // when wr_en =0, memory content at the address location is read and sent onto the data_io port
 assign data_io = (wr_en == 0) ? memory[addr] : 'bz;
endmodule
```

Bi-directional port to send and receive data

Implements a 2D array with 8 locations and each location to store 8-bit data. So, this will result in 8 x 8 = 64-bit RAM

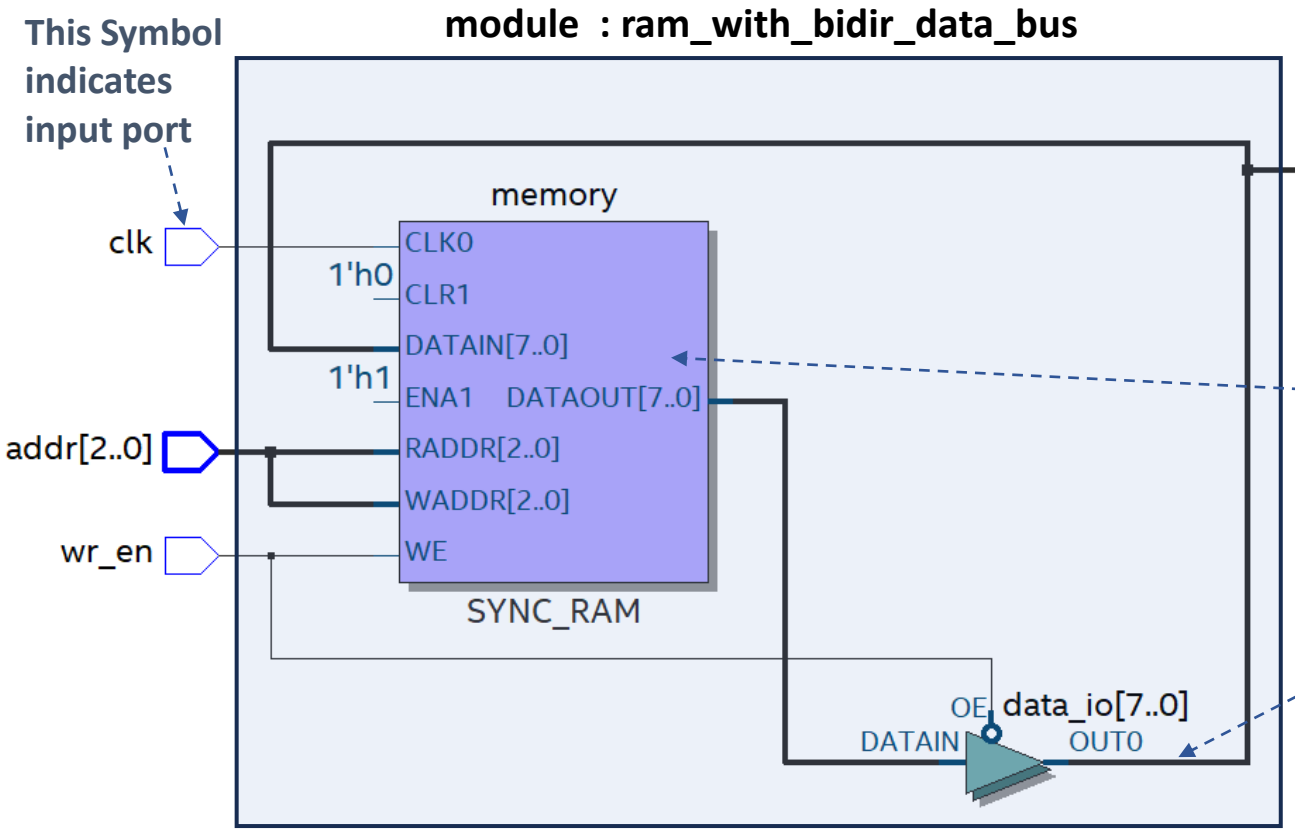
addr	data_io
0	8'b10011111
1	8'b00011100
2	8'b00101001
…	….
7	8'b00000011

Logic to perform synchronous write operation to memory

Implements a tri-state buffer and logic to perform asynchronous read operation to memory

Post Synthesis RTL Netlist View

- **Synthesis tool inferred below mentioned logic for module ram_with_bidir_data_bus :**
  - Block RAM for synchronous write and asynchronous read memory implementation in SystemVerilog code
  - Tri-state buffer for data_io inout port which is controlled by write enable (wr_en) port
    - When wr_en = 1, the tri-state buffer output is disconnected and drives "z" values. Also, data_io port acts like an input port to receive data values which will be written to its internal memory array
    - When wr_en = 0, data_io port acts like an output port to send data values after reading from its internal memory array



This Symbol indicates bi-directional inout port

data_io[7..0]

memory array inside which is indexed using addr value

addr	data_io
0	8'b10011111
1	8'b00011100
2	8'b00101001
...	....
7	8'b00000011

Truth table for nets : when two drivers on a net are driving values simultaneously final resolved value

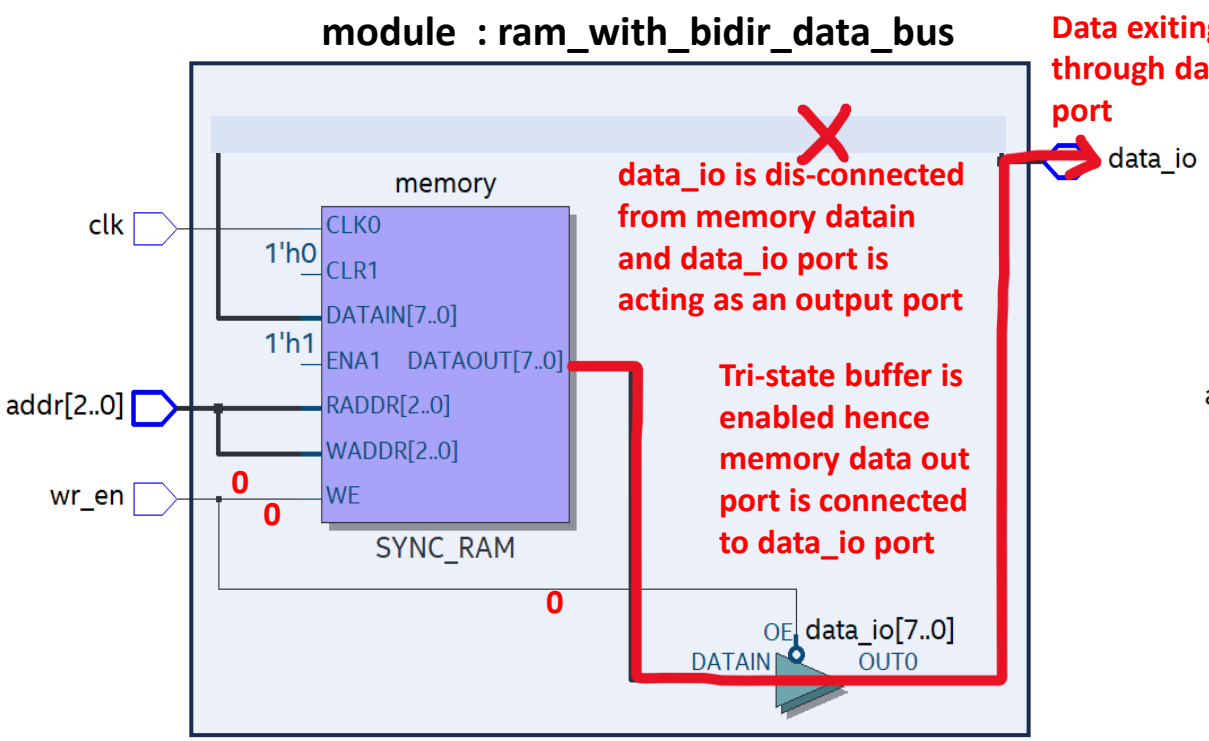
	0	1	x	z
0	0	x	x	0
1	x	1	x	1
x	x	x	x	x
z	0	1	x	z

To avoid multiple driver on data_io port when wr_en=1, the tri-state buffer will drive "z" value on it, so that incoming data values with 1's and 0's on data_io port will override "z" value

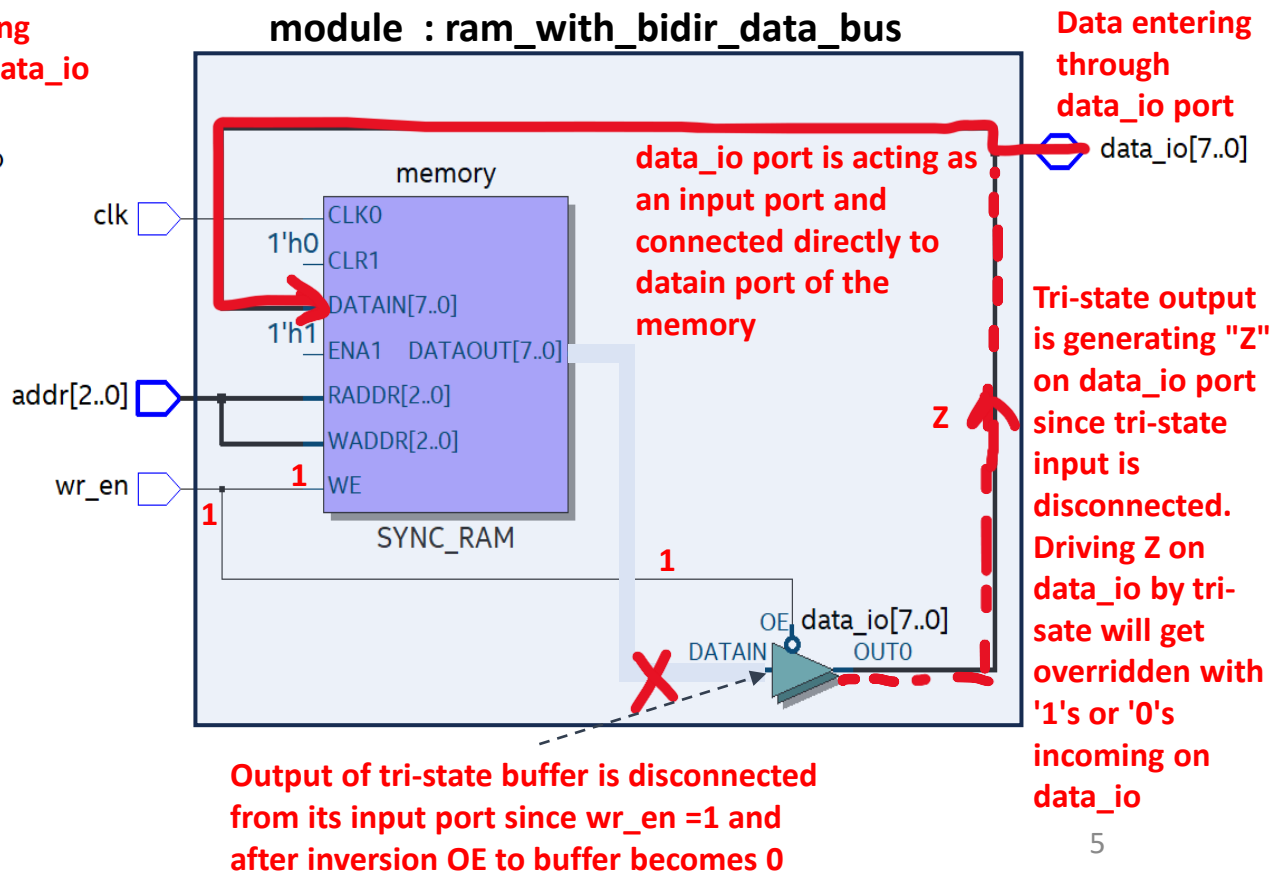
Post Synthesis RTL Netlist View (continued …)

- **inout data_io port view during write and read operation :**
  - Tri-state buffer for data_io inout port which is controlled by write enable (wr_en) port
    - When wr_en = 1, the tri-state buffer output is disconnected and drives "z" values. Also, data_io port acts like an input port to receive data values which will be written to its internal memory array
    - When wr_en = 0, data_io port acts like an output port to send data values after reading from its internal memory array

Memory Read Operation (wr_en = 0)



Memory Write Operation (wr_en = 1)



Testbench Code Review

- **When instantiating RAM design module inside testbench module :**
  - It is mandatory to declare net type variable connecting to inout port of design module

```
1  `timescale 1ns/1ps
2  module ram_wih_bidir_data_bus_testbench();
3  parameter DATA_WIDTH=8;
4  parameter ADDR_WIDTH=3;
5  localparam integer index = 2**ADDR_WIDTH;
6  logic clock, wr_en;
7  // data has to be a wire type since it is getting connected to inout port
8  wire [DATA_WIDTH-1:0] data;
9  logic [DATA_WIDTH-1:0] write_data;
10 logic [DATA_WIDTH-1:0] read_data;
11 logic [ADDR_WIDTH-1:0] addr;
12 logic [DATA_WIDTH-1:0] data_written[index-1:0];
13 logic [DATA_WIDTH-1:0] data_read[index-1:0];
14
15
16 ram_with_bidir_data_bus #(.DATA_WIDTH(DATA_WIDTH), .ADDR_WIDTH(ADDR_WIDTH))
17  design_instance (
18 .clk(clock),
19 .wr_en(wr_en),
20 .addr(addr),
21 .data_io(data)
22 );
23
24 // Tri-state buffer to write and read data using
25 // shared data bus
26 assign data = (wr_en == 1) ? write_data : 8'bz;
27 assign read_data = (wr_en==0) ? data : read_data;
28
29 // Clock Generator
30 always #100 clock = ~clock;
```

"data" must be a wire type and not reg type since it is getting connected to inout port of design module ram_with_bidir_data_bus module

- Since RAM design module has inout type port then a tri-state buffer should be created in testbench to avoid multiple driver when ram design module is sending data value to testbench
- When wr_en=1, testbench is writing data values onto the data_io port of the RAM design module

When wr_en=0, testbech is receiving data values from the design module through its port data_io for read operation

Testbench Code Review (continued…)

```
31  //Stimulus
32  initial
33  begin
34  // Initiliase Input Stimulus
35  //write_data = 0;
36  clock = 0;
37  addr = 0;
38  wr_en = 0;
39  #25;
40
41  // write random data to all locations of memory
42  for(int i=0; i<index; i++) begin
43    write_data = $urandom_range(1, 15);
44    data_written[i] = write_data;
45    wr_en = 1;
46    #200;
47    addr = addr + 1;
48  end
49
50  // Reset address and wr_en values
51  addr  = 0;
52  wr_en = 0;
53
54  // read data written to all locatons of memory
55  for(int i=0; i<index; i++) begin
56    #200
57    wr_en = 0;
58    data_read[i] = read_data;
59    addr = addr + 1;
60  end
61
62  // wait for some delay
63  #200;
64
65  // Compare read data with written data to all memory locations
66  foreach (data_written[idx]) begin
67    if(data_read[idx] === data_written[idx]) begin
68      $display("Read and Write Data Matched !  data_read = %h, data_write = %h, index = %h\n", data_read[idx], data_written[idx], idx);
69    end else begin
70      $display("Read and Write Data Mis-Matched !  data_read = %h, data_write = %h, index = %h\n", data_read[idx], data_written[idx], idx);
71    end
72  end
73  end
74  endmodule
```

- This part of the testbench code implements driver to perform write operation to RAM design module
- Random data is generated and sent to the design module data_io inout port along with wr_en and addr values on respective ports
- Random data generated is also stored in data_written array for later comparison

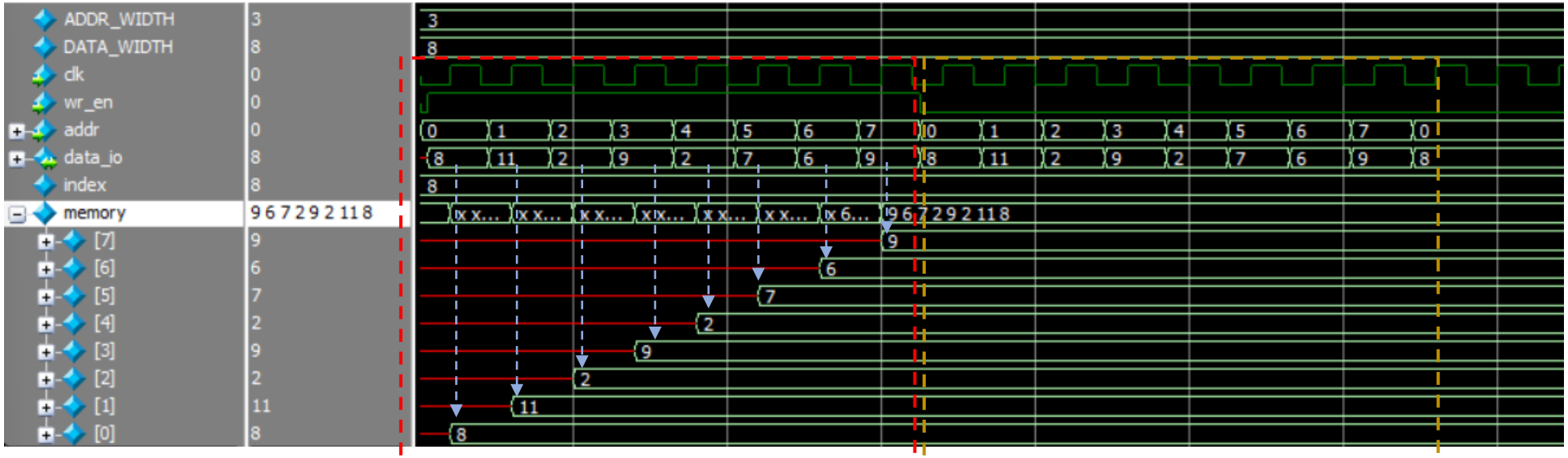
- This part of the testbench code implements driver to perform read operation to RAM design module
- Incoming data through data_io port of RAM design module is read and stored in data_read array for comparison purpose

This part of testbench is for comparing data written to the RAM with the data read from the RAM

Simulation Result

❑ Simulation result of RAM with bi-directional data port :

- Write to RAM is performed at the posedge of clk (i.e. write is synchronous to clock)
- Read to RAM is performed immediately whenever addr value changes (i.e read is asynchronous without waiting for clock edge)



- **During this period in red colored box, write to memory is performed since "wr_en" = 1**
- **During this duration data_io port acts like an input port**
- **As seen above, memory[0] to memory[7] location are filled each clock cycle with incoming data values on "data_io" port.**
- **Note : Due to synchronous write, data values are written in memory array at the posdedge of clk**

- **During this period in golden colored box, read to memory is performed since "wr_en" = 0**
- **During this duration data_io port acts like an output port**
- **Note : Due to asynchronous read, data values are read from memory array immediately each time addr value is changes and wr_en = 0**