Single-Port RAM (16x8)

Random Access Memory (RAM)

Random Access Memory (RAM) is a fast, volatile memory that stores data temporarily for quick access by the processor. Unlike permanent storage, RAM loses data when power is turned off but enables rapid read/write operations. It comes in two main types: SRAM (faster, used in cache) and DRAM (denser, used in main memory). RAM organizes data in addressable cells, accessed via address buses and controlled by read/write signals. Its speed and capacity (e.g., 8GB, 16GB) directly impact system performance. Modern RAM, like DDR SDRAM, transfers data on both clock edges for higher speeds. RAM is essential for multitasking, gaming, and real-time applications. Future advancements include non-volatile RAM and 3D-stacked memory for better efficiency. In summary, RAM bridges the gap between processors and storage, ensuring smooth computing operations.

Single Port RAM (16x8)

A Single Port RAM (Random Access Memory) is a volatile memory that enables data storage and retrieval through a single address and data port. This project focuses on designing a 64×8 Single Port RAM, featuring:

- 64 memory locations (addressable using 6 bits, since 26=6426=64)
- 8-bit data width (each location stores 8 bits/1 byte)

This RAM is essential for digital systems requiring temporary data storage, such as microcontrollers, data buffers, and FPGA-based applications, where moderate-speed read/write operations are sufficient

The 64×8 Single Port RAM comprises the following key components:

- Address Input (6-bit) – Selects one of the 64 memory locations.
- Data Input (8-bit) – Data to be written into RAM.
- Data Output (8-bit) – Data read from RAM (bidirectional in some designs).
- Write Enable (WE) – Controls write operations (active high/low).
- Chip Select (CS) – Enables/disables the RAM.
- Clock (CLK) – Synchronizes read/write operations (for synchronous RAM).

Working Principle

Read Operation

1. The **address bus** selects the memory location.
2. **Write Enable (WE)** is set to 0 (read mode).
3. The data stored at the selected address appears on the **data output** after a small propagation delay (for asynchronous RAM) or at the next clock edge (for synchronous RAM).

Write Operation

1. The **address bus** selects the memory location.
2. **Write Enable (WE)** is set to 1 (write mode).
3. The **data input** is written into the selected memory location at the rising/falling edge of the clock (for synchronous RAM) or immediately (for asynchronous RAM).

Verilog Code

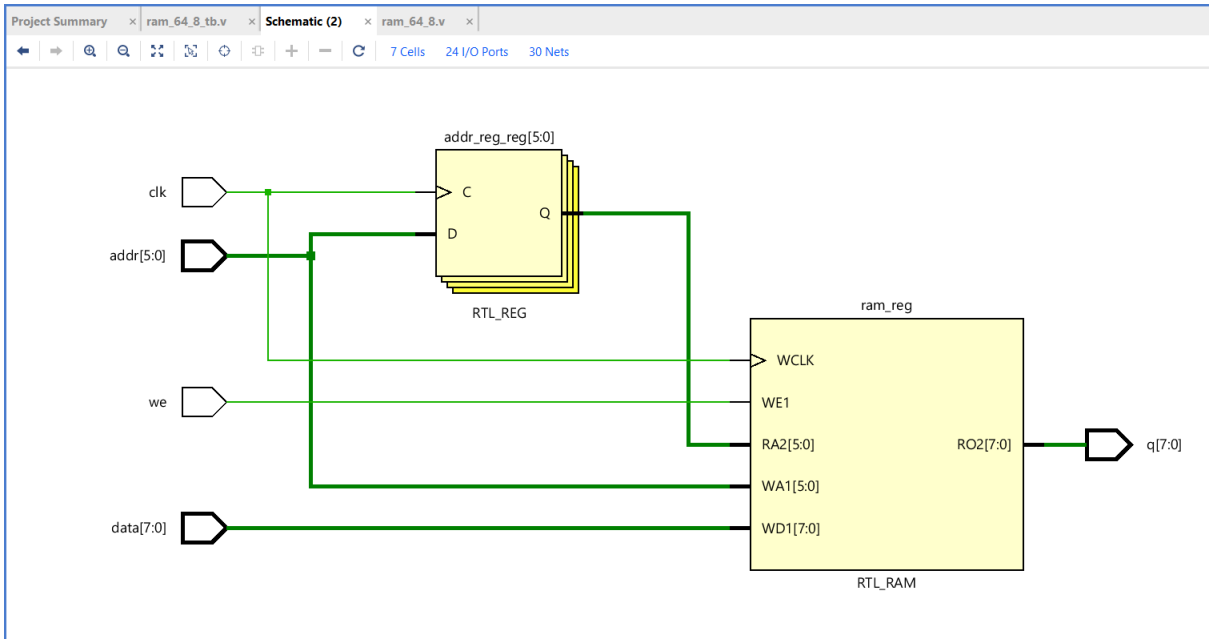
```
module ram_64_8(
   input [7:0] data,      //We storing 8 bit data in 64 loctions
   input [5:0] addr,     //6-bits are sufficient to address (2^6=64)
   input we,             //we (Write Enable),if it is high(1) we are writing in a RAM, if Low (0) reading from RAM
   input clk,
   output [7:0] q          //8-bit data port for output
);
   reg [7:0] ram[63:0];
   reg [5:0] addr_reg;
   always @(posedge clk) begin
      if (we) begin
         ram[addr] <= data;  // Write operation
         addr_reg <= addr;
      end
      else begin
         addr_reg <= addr;
      end
   end
   assign q = ram[addr_reg];
endmodule
```

Testbench

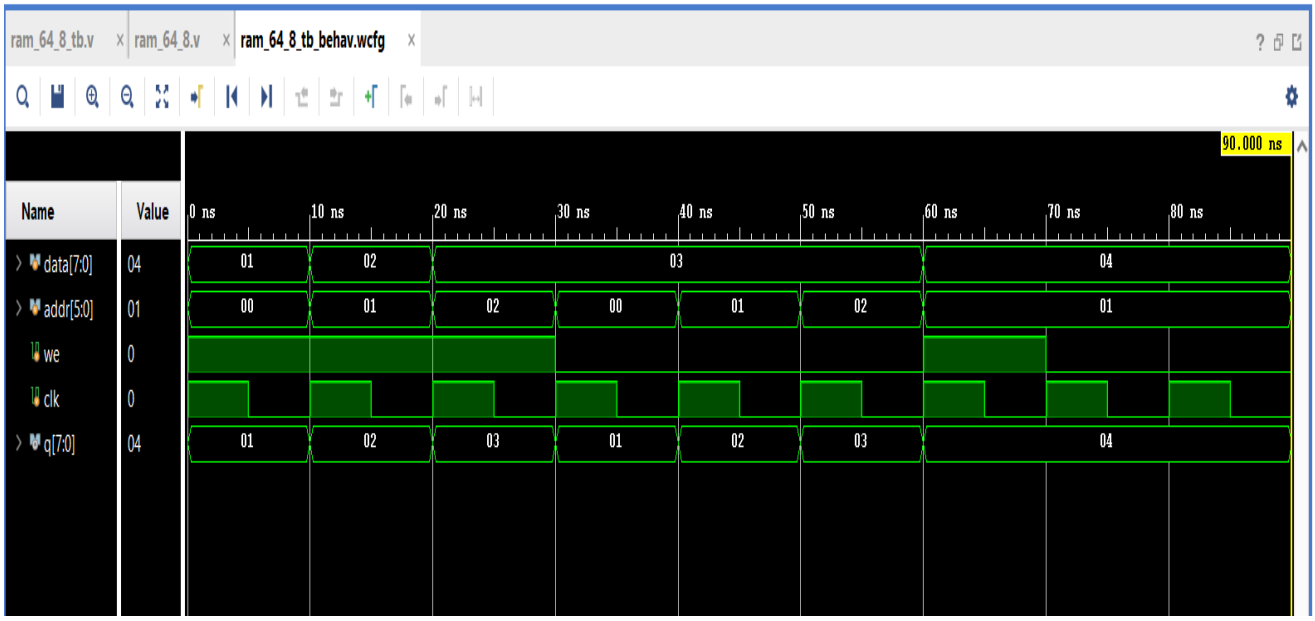
```
// Single Port RAM testbench
module ram_64_8_tb;
 reg [7:0] data;  // Input data
 reg [5:0] addr;  // Address
 reg we;          // Write enable
 reg clk;        // Clock
 wire [7:0] q;    // Output data

 // Instantiate the RAM module
 ram_64_8 uut (
  .data(data),
  .addr(addr),
  .we(we),
  .clk(clk),
  .q(q)
 );
 // Clock generation (100 MHz)
 initial begin
  clk = 1'b1;
  forever #5 clk = ~clk;
 end
 initial begin
  $dumpfile("dump.vcd");
  $dumpvars(1, ram_64_8_tb);

  // Test 1: Write values to addresses 0, 1, 2
  data = 8'h01;
  addr = 6'd0;
  we = 1'b1;
```

```
    #10;
    data = 8'h02;
    addr = 6'd1;
    #10;
    data = 8'h03;
    addr = 6'd2;
    #10;
    we = 1'b0;
    addr = 6'd0;
    #10;
    $display("Addr 0: q = %h (Expected: 01)", q);
    addr = 6'd1;
    #10;
    $display("Addr 1: q = %h (Expected: 02)", q);
    addr = 6'd2;
    #10;
    $display("Addr 2: q = %h (Expected: 03)", q);
    we = 1'b1;
    data = 8'h04;
    addr = 6'd1;
    #10;
    we = 1'b0;
    #10;
    $display("Addr 1 (after write): q = %h (Expected: 04)", q);
    #10 $finish;
  end
  initial begin
    $monitor("Time=%0t: addr=%d data=%h we=%b q=%h",
            $time, addr, data, we, q);
  end
endmodule
```

Schematic



Functional Simulation Results



Conclusion

The **64×8 Single Port RAM** provides an efficient memory solution for digital systems requiring moderate-capacity storage with simple control. Its 6-bit address and 8-bit data width make it suitable for embedded applications, data buffering, and FPGA-based designs. While limited by single-port access, it offers a low-power and cost-effective approach for sequential read/write operations. Future enhancements could include pipelining for faster throughput or error-correction features for improved reliability.