Memory_verilog_IP

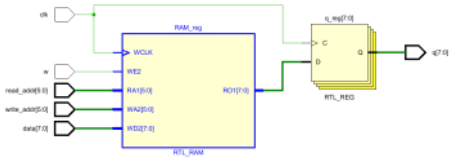
17 July 2025 08:10

Single port RAM

So here we can perform

- 1R
- 1W
- 1R & 1W at the same time

This is a module for 64*8bit reg which is a 64 bytes memory can be expanded since I have written a parameterized code



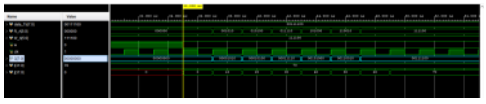
In the TB we have set a program to write at 7 locations and than we read those 7 locations to check the performance here is the results

TCL console

```
# run 1000ns
Write | t=13000 | W_A=0  | Data=00000000
Write | t=15000 | W_A=10 | Data=00001010
Write | t=17000 | W_A=20 | Data=00010100
Write | t=19000 | W_A=30 | Data=00011110
Write | t=21000 | W_A=40 | Data=00101000
Write | t=23000 | W_A=50 | Data=00110010
Write | t=25000 | W_A=60 | Data=00111100
Read  | t=37000 | R_A=0  | Q=00000000
Read  | t=39000 | R_A=10 | Q=00001010
Read  | t=41000 | R_A=20 | Q=00010100
Read  | t=43000 | R_A=30 | Q=00011110
Read  | t=45000 | R_A=40 | Q=00101000
Read  | t=47000 | R_A=50 | Q=00110010
Read  | t=49000 | R_A=60 | Q=00111100
INFO: [USF-XSim-96] XSim completed. Design snapshot 'Single_Port_RAM_Tb_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
```

In the first part of loop the 7 locations are written by the loops, Q remains at 8'b00(initialised at that value) and rest of values change also the value of W=1 for this time duration

Finally W=0 to perform Read operation in the second half



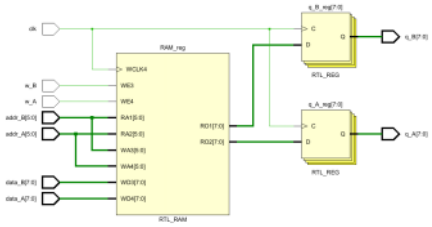
```
module Single_Port_RAM(data, read_addr, write_addr, w, clk, q);
    parameter DATA_WIDTH = 7;
    parameter ADDR_WIDTH = 5;
    parameter RAM_LOC = 63;
    input [DATA_WIDTH :0] data;
    input [ADDR_WIDTH:0] read_addr, write_addr;
    input w, clk;
    output reg [7:0] q;


    reg [DATA_WIDTH:0] RAM [RAM_LOC:0];  // 64 locations

    always @(posedge clk) begin
        if (w)
            RAM[write_addr] <= data;  // Write
        q <= RAM[read_addr];          // Read
    end

endmodule
```

code is also attached and updated in github repo

Dual port ram

Here the operations which can be performed are,

- 2W & 2R(IN_pos) where previous value is fetched out to reg.
  - //this feature is used in neural network as previous layer values are needed in new layer
- 1W & 1R optimal config where we can read data from port A while we write data from B, This is the case for which we will design TB



Dual port ram fully parameterized

TB design,

- Write data using port_A from [0:30] with a step of 3, all odd location
- Fork join to have 2 parallel loops to demonstrate port_A Write and port_R Read in 1 clk cycle
  - Read from port_B all [0:30] with a step of 1, all location in that region

```
# run 1000ns
Write to RAM |At time t=3000 |addr_A= 0 | data_A=00000010
Write to RAM |At time t=5000 |addr_A= 3 | data_A=00000101
Write to RAM |At time t=7000 |addr_A= 6 | data_A=00001000
Write to RAM |At time t=9000 |addr_A= 9 | data_A=00001011
Write to RAM |At time t=11000 |addr_A=12 | data_A=00001110
Write to RAM |At time t=13000 |addr_A=15 | data_A=00010001
Write to RAM |At time t=15000 |addr_A=18 | data_A=00010100
Write to RAM |At time t=17000 |addr_A=21 | data_A=00010111
Write to RAM |At time t=19000 |addr_A=24 | data_A=00011010
Write to RAM |At time t=21000 |addr_A=27 | data_A=00011101
Write to RAM |At time t=23000 |addr_A=30 | data_A=00100000
Write function over, R/W starts from here t=23000
```

- //we should encounter don't care at prime number memory location

```
Read from RAM | t=45000 | addr_B=10 | q_B=00001101
Write to RAM |At time t=45000 |addr_A=20 | data_A=00010111
Write to RAM |At time t=47000 |addr_A=22 | data_A=00011001
Read from RAM | t=47000 | addr_B=11 | q_B=xxxxxxxx
Read from RAM | t=49000 | addr_B=12 | q_B=00001111
Write to RAM |At time t=49000 |addr_A=24 | data_A=00011011
Write to RAM |At time t=51000 |addr_A=26 | data_A=00011101
Read from RAM | t=51000 | addr_B=13 | q_B=xxxxxxxx
Read from RAM | t=53000 | addr_B=14 | q_B=00010001
Write to RAM |At time t=53000 |addr_A=28 | data_A=00011111
Write to RAM |At time t=55000 |addr_A=30 | data_A=00100001
Read from RAM | t=55000 | addr_B=15 | q_B=00010001
```

-

```
Read from RAM | t=57000 | addr_B=16 | q_B=00010011
Read from RAM | t=59000 | addr_B=17 | q_B=xxxxxxxx
Read from RAM | t=61000 | addr_B=18 | q_B=00010101
Read from RAM | t=63000 | addr_B=19 | q_B=xxxxxxxx
Read from RAM | t=65000 | addr_B=20 | q_B=00010111
Read from RAM | t=67000 | addr_B=21 | q_B=00010111
Read from RAM | t=69000 | addr_B=22 | q_B=00011001
Read from RAM | t=71000 | addr_B=23 | q_B=xxxxxxxx
Read from RAM | t=73000 | addr_B=24 | q_B=00011011
Read from RAM | t=75000 | addr_B=25 | q_B=xxxxxxxx
Read from RAM | t=77000 | addr_B=26 | q_B=00011101
Read from RAM | t=79000 | addr_B=27 | q_B=00011101
Read from RAM | t=81000 | addr_B=28 | q_B=00011111
```

  - Write from port_A to write data to even port [0:30] with a step of 2, all even location