SRAM

Understanding SRAM: RTL Implementation and Verification
As part of my RTL practice and verification journey, today I explored Static Random Access Memory (SRAM)—a critical component in modern digital systems.

What is SRAM?
SRAM (Static Random Access Memory) is a type of volatile memory that retains data as long as power is supplied, unlike DRAM, which requires periodic refreshing. SRAM stores data using flip-flops, making it faster and more power-efficient for certain applications. It's widely used in cache memory, CPU registers, and small embedded systems due to its low latency.

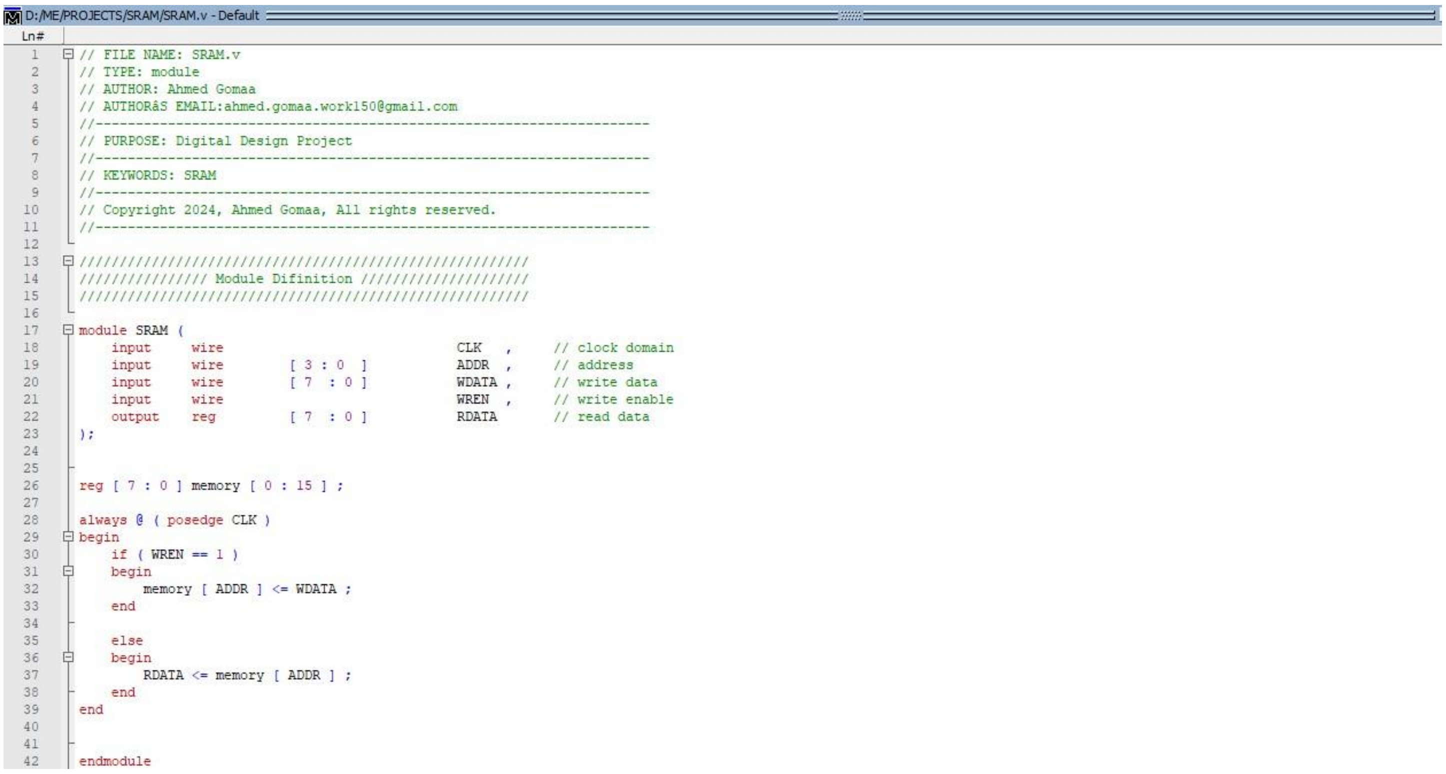
Why is SRAM Important?
SRAM plays a crucial role in high-speed computing environments. Its static nature means it doesn't need to be refreshed periodically, allowing faster data access times, which is essential for applications requiring high-speed processing, such as cache in CPUs or as memory in embedded systems.

How to Implement SRAM in RTL?
When implementing SRAM in RTL, the design starts with defining memory size (number of bits and word size). The typical building block of SRAM is the 6T memory cell, made up of six transistors that store a single bit of data. Here's a high-level approach to its RTL design:

1. Define the memory array – A two-dimensional array where each row is a memory word.
2. Control logic – Write and read operations are controlled by signals like write enable, chip enable, and address lines.
3. Address decoder – Decodes the input address to select a specific memory location.
4. Read/Write data path – The data to be written or read is managed through input/output ports, ensuring the appropriate read/write functionality.

Verifying the SRAM involves simulating different test cases to validate functionality under normal, boundary, and corner cases. Typical verification scenarios include checking write-read operations, power-up data states, and timing checks for setup and hold violations.

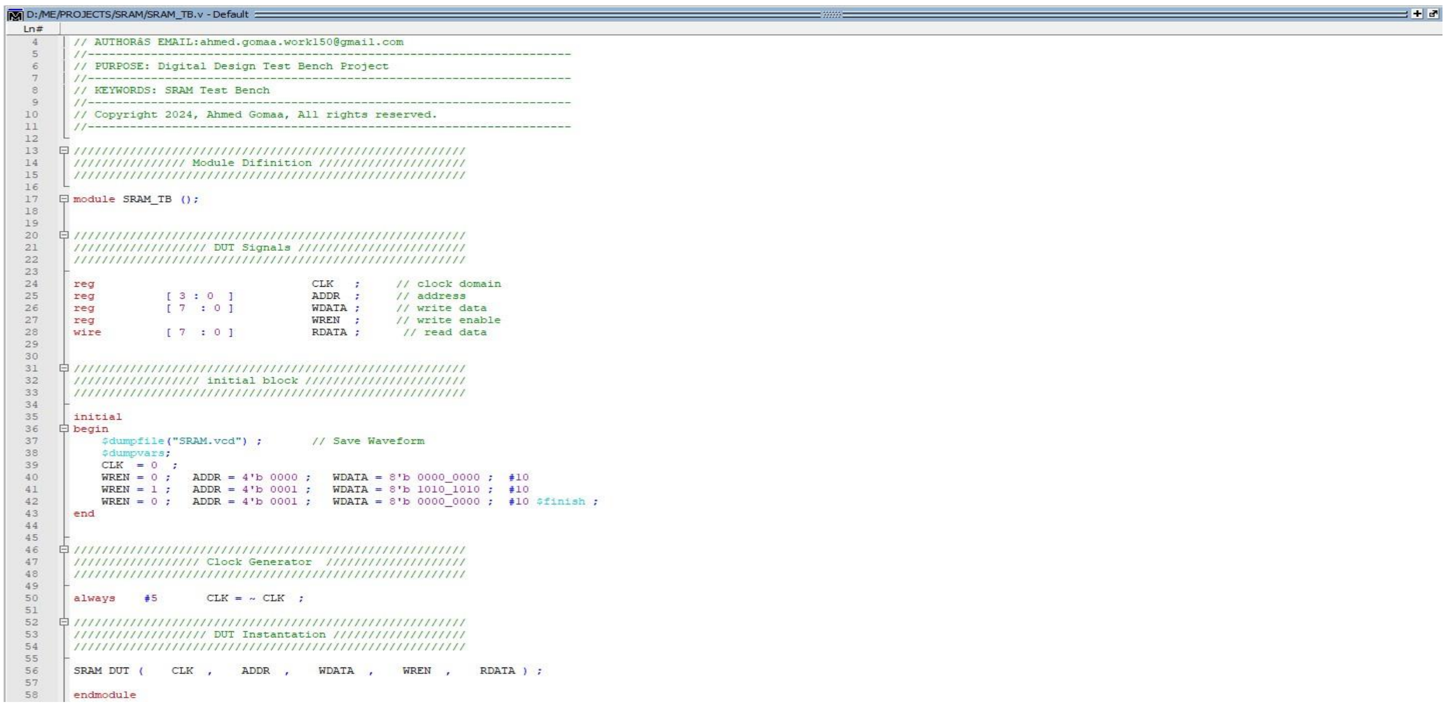
RTL code :



D:/ME/PROJECTS/SRAM/SRAM.v - Default

```verilog
// FILE NAME: SRAM.v
// TYPE: module
// AUTHOR: Ahmed Gomaa
// AUTHORâS EMAIL:ahmed.gomaa.work150@gmail.com
//-----------------------------------------------------------------------
// PURPOSE: Digital Design Project
//-----------------------------------------------------------------------
// KEYWORDS: SRAM
//-----------------------------------------------------------------------
// Copyright 2024, Ahmed Gomaa, All rights reserved.
//-----------------------------------------------------------------------

////////////////////////////////////////////////////////
/////////////// Module Difinition ////////////////////
////////////////////////////////////////////////////////

module SRAM (
    input     wire        [ 3 : 0  ]           CLK   ,     // clock domain
    input     wire        [ 3 : 0  ]           ADDR  ,     // address
    input     wire        [ 7  : 0 ]           WDATA ,     // write data
    input     wire                             WREN  ,     // write enable
    output    reg         [ 7  : 0 ]           RDATA       // read data
);


reg [ 7 : 0 ] memory [ 0 : 15 ] ;

always @ ( posedge CLK )
begin
    if ( WREN == 1 )
    begin
        memory [ ADDR ] <= WDATA ;
    end

    else
    begin
        RDATA <= memory [ ADDR ] ;
    end
end


endmodule
```

Test bench :



D:/ME/PROJECTS/SRAM/SRAM_TB.v - Default

```verilog
// AUTHORâS EMAIL:ahmed.gomaa.work150@gmail.com
//-----------------------------------------------------------------------
// PURPOSE: Digital Design Test Bench Project
//-----------------------------------------------------------------------
// KEYWORDS: SRAM Test Bench
//-----------------------------------------------------------------------
// Copyright 2024, Ahmed Gomaa, All rights reserved.
//-----------------------------------------------------------------------

////////////////////////////////////////////////////////
/////////////// Module Difinition ////////////////////
////////////////////////////////////////////////////////

module SRAM_TB ();


////////////////////////////////////////////////////////
////////////////// DUT Signals ///////////////////////
////////////////////////////////////////////////////////

reg                                  CLK   ;     // clock domain
reg          [ 3 : 0  ]              ADDR  ;     // address
reg          [ 7  : 0 ]              WDATA ;     // write data
reg                                  WREN  ;     // write enable
wire         [ 7  : 0 ]              RDATA ;      // read data


////////////////////////////////////////////////////////
////////////////// initial block /////////////////////
////////////////////////////////////////////////////////

initial
begin
    $dumpfile("SRAM.vcd") ;       // Save Waveform
    $dumpvars;
    CLK  = 0  ;
    WREN = 0 ;   ADDR = 4'b 0000 ;   WDATA = 8'b 0000_0000 ;  #10
    WREN = 1 ;   ADDR = 4'b 0001 ;   WDATA = 8'b 1010_1010 ;  #10
    WREN = 0 ;   ADDR = 4'b 0001 ;   WDATA = 8'b 0000_0000 ;  #10 $finish ;
end


////////////////////////////////////////////////////////
////////////////// Clock Generator  //////////////////
////////////////////////////////////////////////////////

always    #5       CLK = ~ CLK  ;

////////////////////////////////////////////////////////
/////////////////// DUT Instantation /////////////////
////////////////////////////////////////////////////////

SRAM DUT (    CLK  ,    ADDR  ,    WDATA  ,    WREN  ,    RDATA ) ;

endmodule
```

Applications of SRAM

SRAM is ubiquitous in digital design for applications requiring fast data access, such as:

- **Processor Cache Memory**: Often used for L1, L2, and L3 caches in CPUs and GPUs due to its low access time.
- **Networking Devices**: Used in high-speed routers and switches for buffer storage. **Embedded Systems**: Common in microcontrollers and FPGAs as on-chip memory for critical, frequently accessed data.
- **Graphics Cards**: Utilized in frame buffers for quick rendering of images.
-

Conclusion

Working with SRAM in RTL design and verification is a valuable exercise in understanding memory architecture. Its wide use in high-performance systems makes it an essential component to master for any digital design engineer.