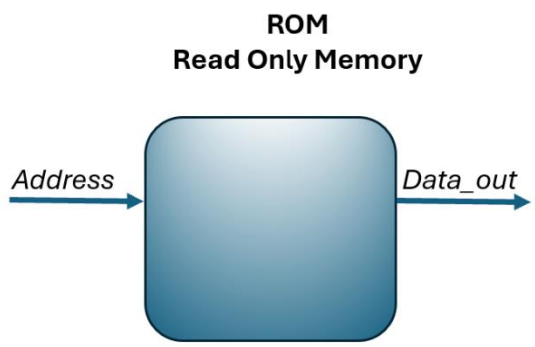
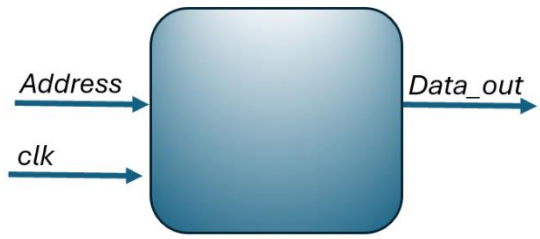
The Design of a ROM in Verilog



Asynchronous ROM design – no clock



Synchronous ROM design – clocked

We will include in this document the design of synchronous ROM modules. The first ROM will be instantiated by listing the data within each memory word register within an initial block in the Verilog module. The second design will use the function **$readmemh** to initialize the ROM. The function reads the data from a **.hex file** generated in **NotePad**. The same testbench file can be used to verify the functionality and data stored in the ROM.

(1) Typing the ROM initialization file in the ROM module within an initial block:

```
module Memory_ROM (ROM_address, data_out, clk);

parameter ROM_depth    = 8;
parameter address_size = $clog2(ROM_depth);   // log2(8) = 3
parameter word_size    = 8;

input        [address_size-1 :0] ROM_address;
input                            clk;
output  reg  [word_size-1    :0] data_out;

reg [word_size-1:0] ROM [0: ROM_depth-1];

initial
begin
   ROM [0] = 8'h00;
   ROM [1] = 8'h01;
   ROM [2] = 8'h20;
   ROM [3] = 8'h03;
   ROM [4] = 8'h40;
   ROM [5] = 8'h05;
   ROM [6] = 8'h60;
   ROM [7] = 8'h07;
end

always @ (posedge clk)
   data_out <= ROM[ROM_address];

endmodule
```

Initializing the data words in the ROM

ROM [address] = data;

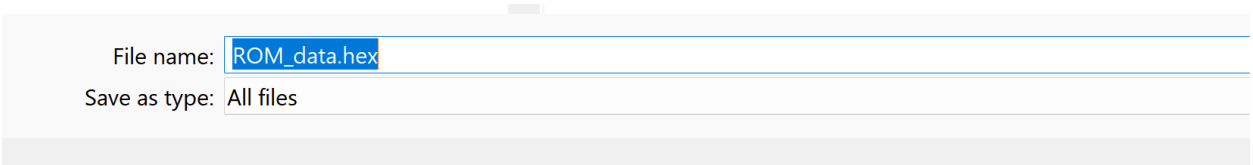
Creating the ROM array registers

[word_size-1:0] defines the word bits with the MSB at the left and the LSB bit 0 on the right. The [0: ROM_depth-1] determines the order of the word addresses in the array with the top one being address 0 and the bottom one being ROM_depth-1

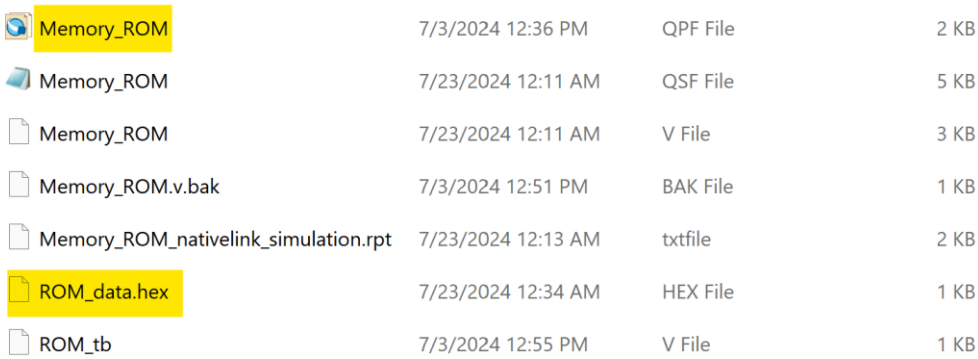
(2) Typing the ROM initialization file in the ROM module within an initial block:

(a) Create the memory initialization file in Notepad (text editor).

(b) Save the file with the extension **.hex** to the file name (example: "**ROM_data.hex**")

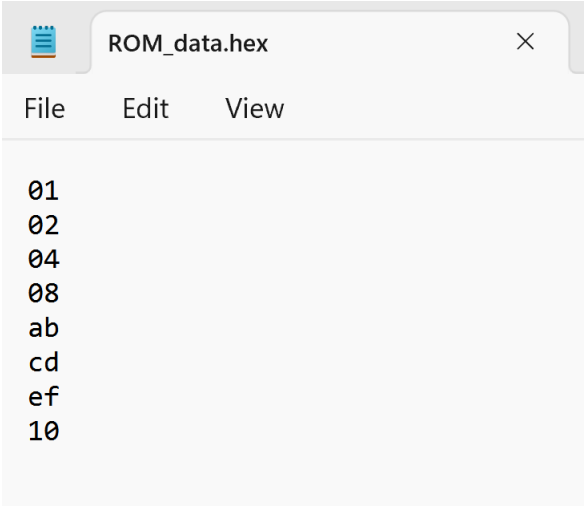


(c) Save it in the same project folder that you have created for your design.



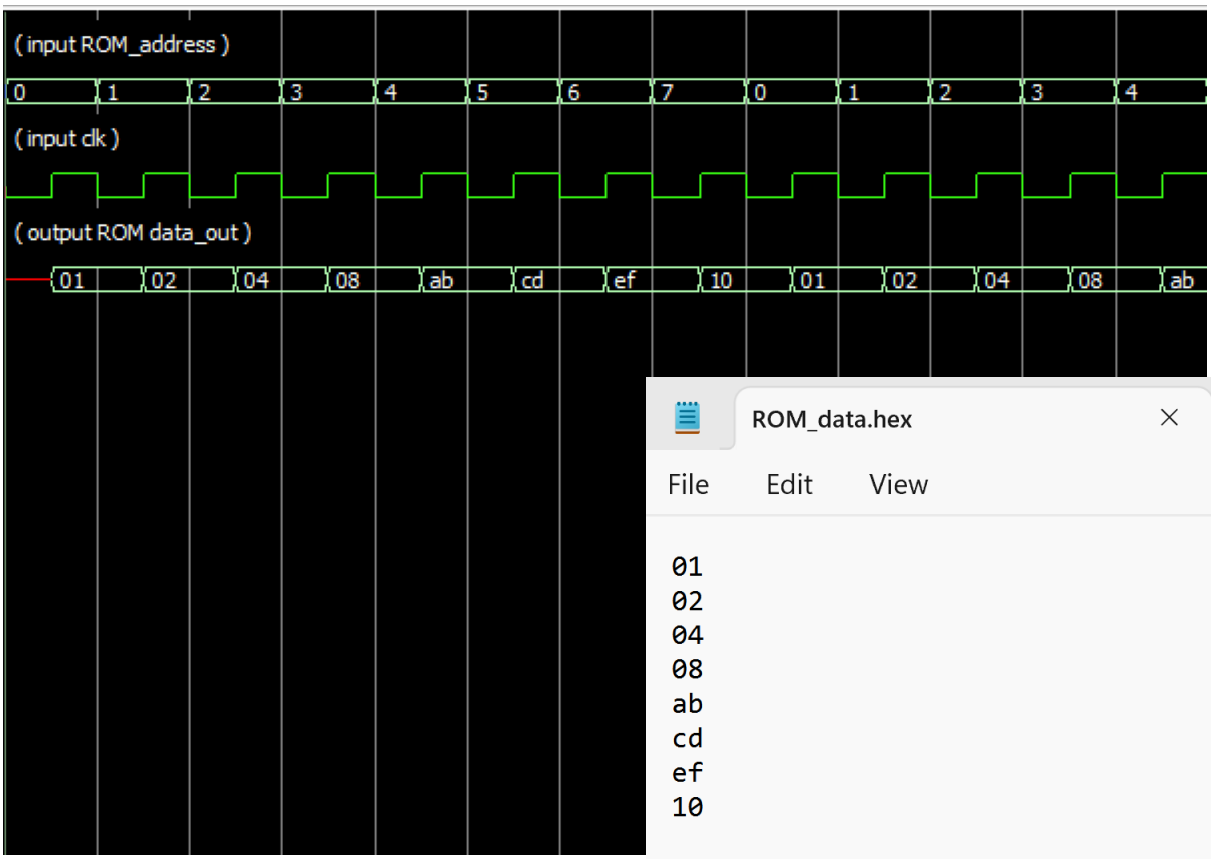
Memory_ROM	7/3/2024 12:36 PM	QPF File	2 KB
Memory_ROM	7/23/2024 12:11 AM	QSF File	5 KB
Memory_ROM	7/23/2024 12:11 AM	V File	3 KB
Memory_ROM.v.bak	7/3/2024 12:51 PM	BAK File	1 KB
Memory_ROM_nativelink_simulation.rpt	7/23/2024 12:13 AM	txtfile	2 KB
ROM_data.hex	7/23/2024 12:34 AM	HEX File	1 KB
ROM_tb	7/3/2024 12:55 PM	V File	1 KB

(d) write the data values in hexadecimal
(e) Have the list of values separated by new line
(f) Don't add anything else other than the hexadecimal values



```
/*-----------------------------------------------------------------------
      ROM Design in Verilog
-----------------------------------------------------------------------

  $readmemh ("file", memory_identifier [,begin_address[,end_address]]) ;

  To read data from a file and store it in memory, use the functions:
  $readmemb and $readmemh. The $readmemb task reads binary data and
  $readmemh reads hexadecimal data. Data has to exist in a text file.
  White space is allowed to improve readability, as well as comments
  in both single line and block. The numbers have to be stored as binary
  or hexadecimal values. The basic form of a memory file contains numbers
  separated by new line characters that will be loaded into the memory.

  When a function is invoked without starting and finishing addresses,
  it loads data into memory starting from the first cell. To load data only
  into a specific part of memory, start and finish addresses have to be used.
  The address can be explicit, given in the file with the @ (at) character
  and is followed by a hexadecimal address with data separated by a space.
  It is very important to remember the range (start and finish addresses)
  given in the file, the argument in function calls have to match each other,
  otherwise an error message will be displayed and the loading process
  will be terminated.

  - Create the memory initialization file in Notepad.
  - Save it in the same project folder
  - add the extension .hex to the file name (example: "ROM_data.hex")
  - write the data values in hexadecimal
  - Have the list of values seperated by new line
  - don't add anything else other than the hexadecimal values

-----------------------------------------------------------------------*/


  module Memory_ROM (ROM_address, data_out, clk);

  parameter ROM_depth    = 8;
  parameter address_size = $clog2(ROM_depth);   // log2(8) = 3
  parameter word_size    = 8;

  input        [address_size-1 :0] ROM_address;
  input                            clk;
  output  reg  [word_size-1    :0] data_out;

  reg [word_size-1:0] ROM [0: ROM_depth-1];

 //Read memory in hexadecimal format and save it in the ROM register:

  initial
  // $readmemh ("file name saved in the project folder", Memory array)
     $readmemh ("ROM_data.hex", ROM);


  always @ (posedge clk)
     data_out <= ROM[ROM_address];

  endmodule
```

```
module ROM_tb ();

parameter address_size = 3;
parameter ROM_size     = 8;
parameter word_size    = 8;

reg        [address_size-1 :0] ROM_address;
reg                            clk;
wire       [word_size-1    :0] data_out;

Memory_ROM  UUT (ROM_address, data_out, clk);

initial
begin
clk = 1'b0;
forever
#5 clk = ~clk; end

initial
begin
ROM_address = 3'b000;
forever
#10  ROM_address = ROM_address + 1;
end

initial
#200 $stop;     //to save on memory and stop the simulation after #200

endmodule
```



ROM_data.hex

File Edit View

```
01
02
04
08
ab
cd
ef
10
```