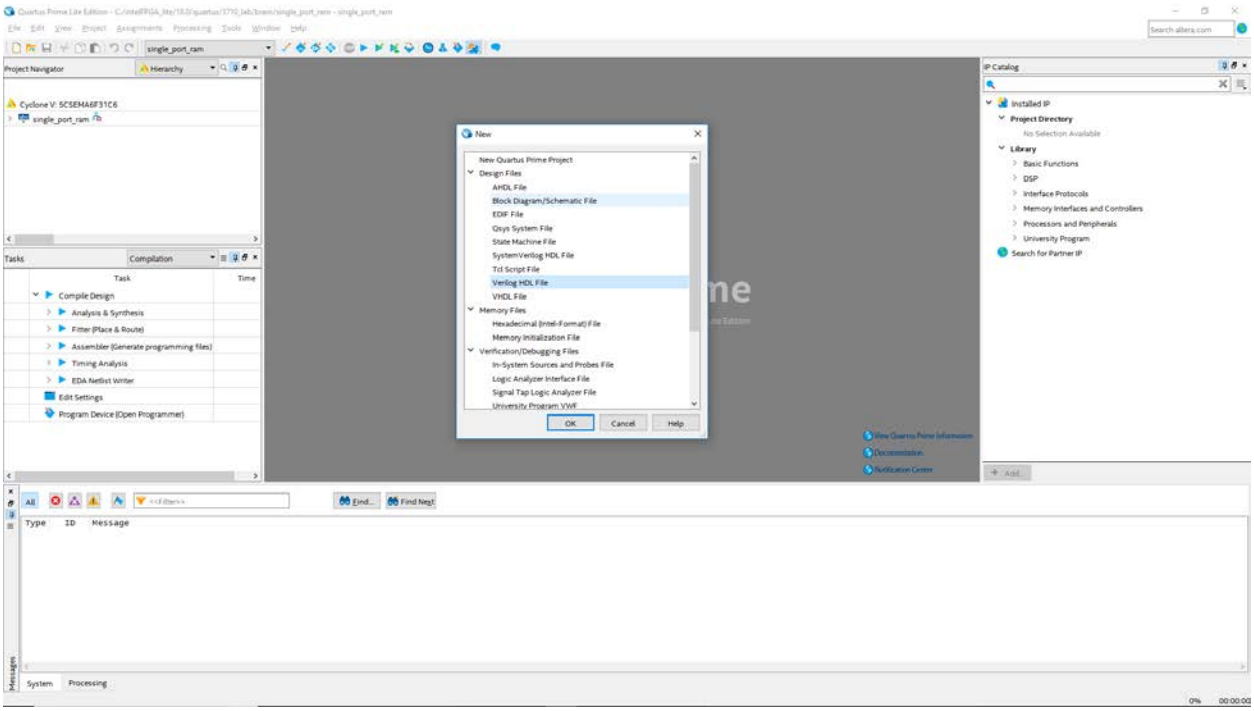
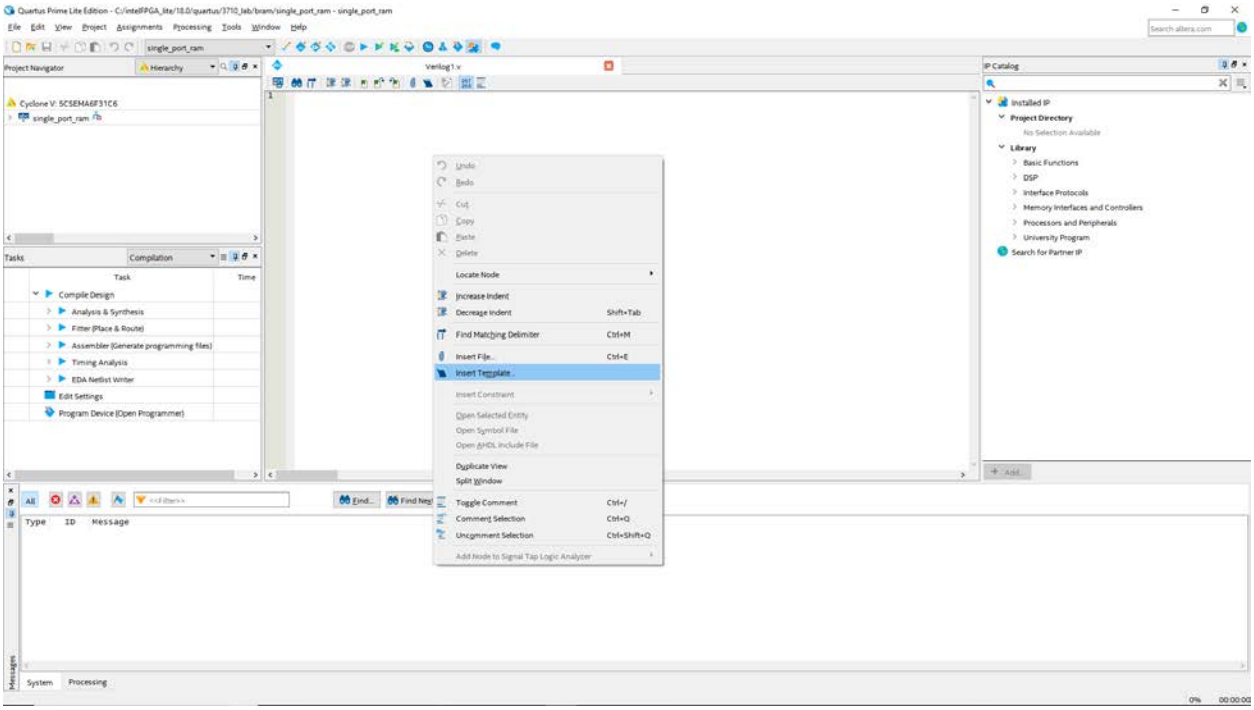
Inferring Block Memory in Quartus –

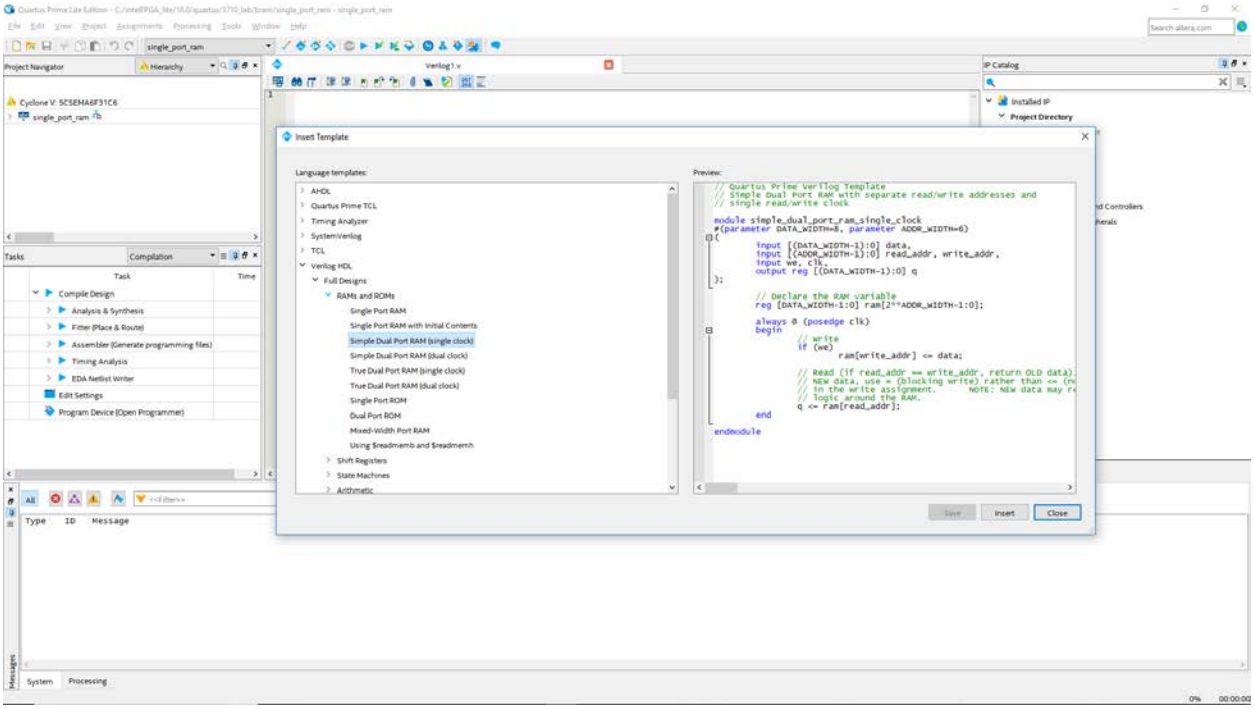
- Once the project options have been setup, navigate to File -> New -> Verilog HDL File.



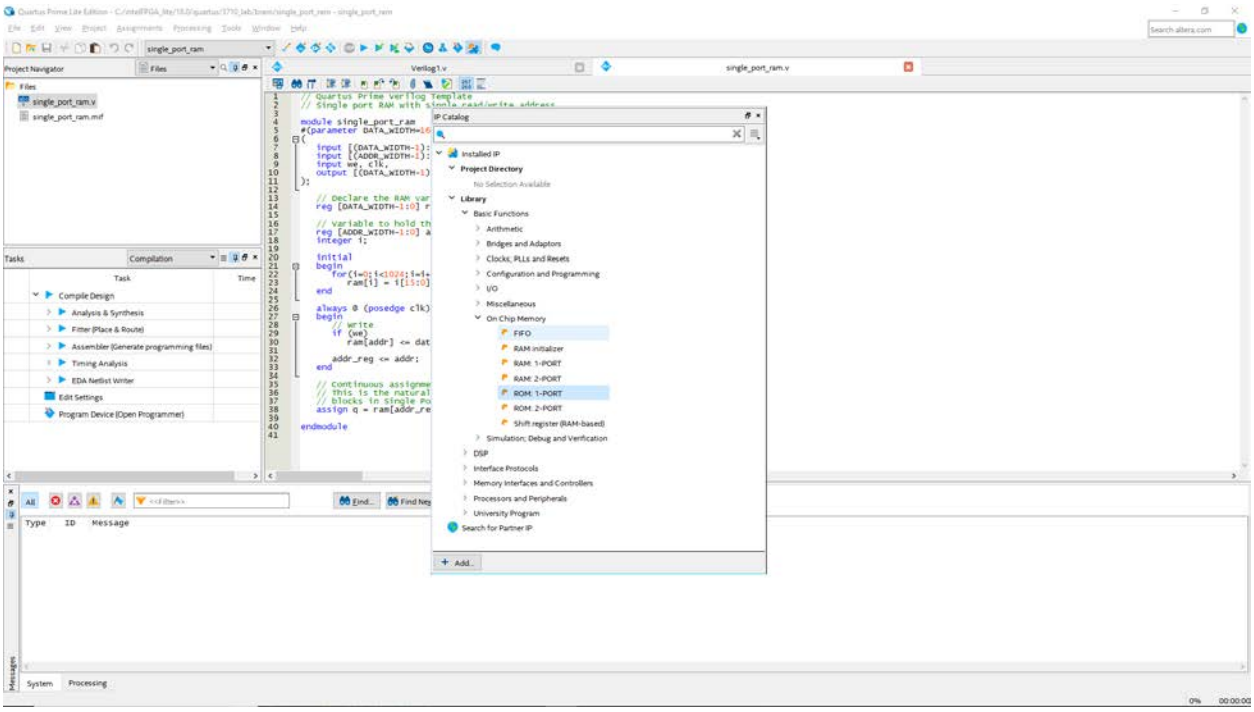
- Right click on the Verilog file editor and navigate to Insert Template option



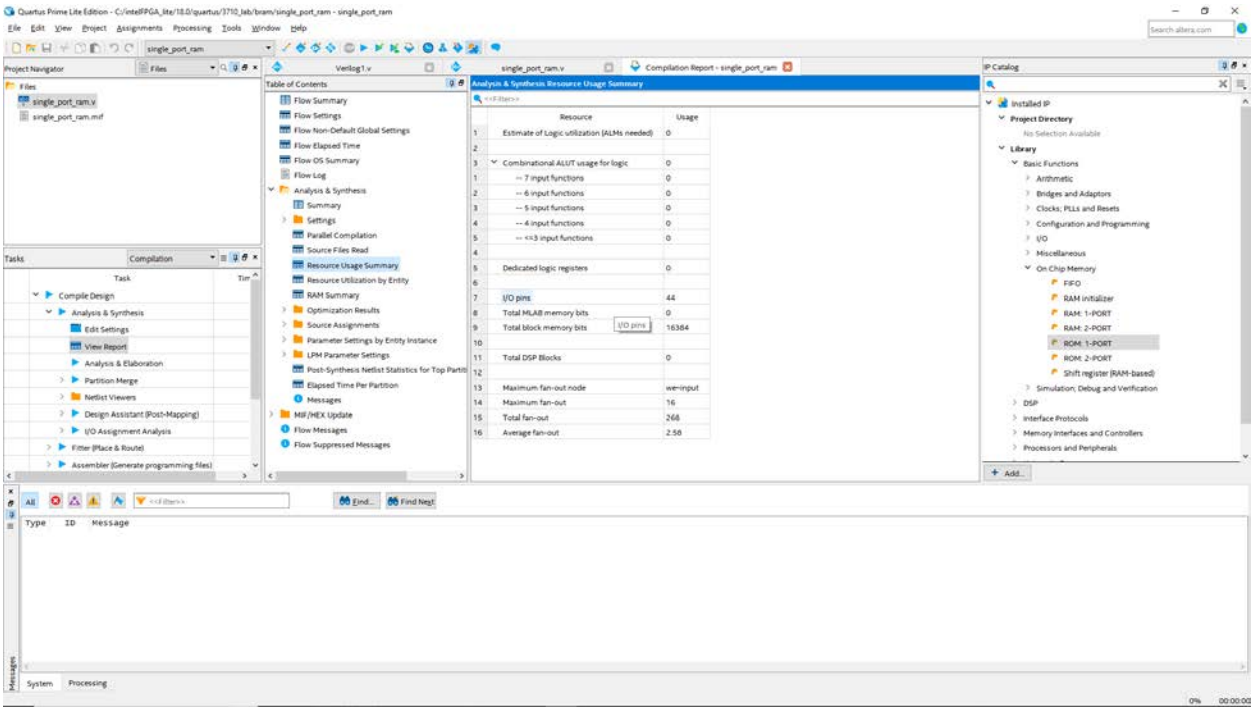
- Navigate to Verilog HDL -> Full Designs -> RAMs and ROMs and select the appropriate design as per your project requirement.



- The alternate way to invoke a RAM is to use the megawizard function IP catalog. Navigate to Tools -> IP catalog -> Library -> Basic Functions -> On Chip Memory and select the appropriate port details to invoke the GUI and select the appropriate parameters to match your project requirements.



- Once the code is synthesized, a good way to make sure the RAM is indeed inferred in the BLOCK RAM (M10K blocks) and not in distributed RAM (MLAB) is to verify the resource utilization summary.



Initializing a RAM/ROM –

- There are multiple ways to initialize a memory file -
    1. The most common way to initialize a memory is to read from a text file through readmemh/readmemb commands.
        - The $readmemb and $readmemh system tasks load the contents of a 2-D array variable from a text file. Quartus Prime supports these system tasks in initial blocks. They may be used to initialize the contents of inferred RAMs or ROMs. They may also be used to specify the power-up value for a 2-D array of registers.
        - Usage:
          ("file_name", memory_name [, start_addr [, end_addr]]);
          ("file_name", memory_name [, start_addr [, end_addr]]);
        - // File Format:
          The text file can contain Verilog whitespace characters, comments, and binary ($readmemb) or hexadecimal ($readmemh) data values. Both types of data values can contain x or X, z or Z, and the underscore character.

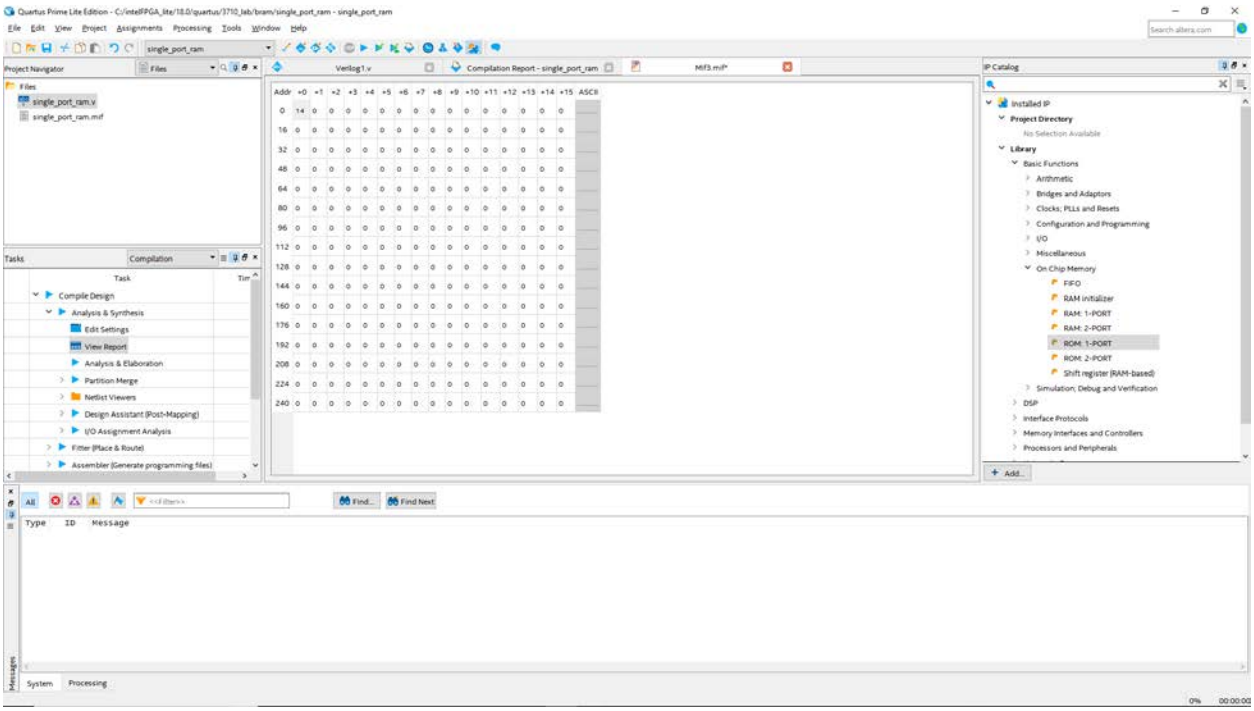
- The data values are assigned to memory words from left to right, beginning at start_addr or the left array bound (the default). The next address to load may be specified in the file itself using @hhhhhh, where h is a hexadecimal character. Spaces between the @ and the address character are not allowed. It shall be an error if there are too many words in the file or if an address is outside the bounds of the array.
- Example:

```
 reg [7:0] ram [0:2];
initial
begin
   $readmemb("init.txt", rom);
end
```

- <init.txt>

```
11110000     // Loads at address 0 by default
10101111
@2 00001111
```

2. Using GUI template – Navigate to File -> New -> Memory Files -> Memory Initialization Files.
    - Specify the Number of Words and Word Size to generate a *.mif file wherein you can specify the values (Decimal) to be initialized at each location.
    - The same initialization file can also be generated in a Hexadecimal-Intel Format file (*.hex) by selecting the appropriate format in Memory Files.



- The option will generate a file (*/mif/*.hex) with the specified value format and the file can be edited as well.

3. Initial statements in memory code
    - The third way to initialize a memory is to write an “initial” block inside the memory code and this automatically invokes a *.mif file with the initialized values in db directory.
    - Example:

```
initial
begin
        for(i=0;i<depth;i=i+1)
        ram[i] = i[word-1:0];
end
```