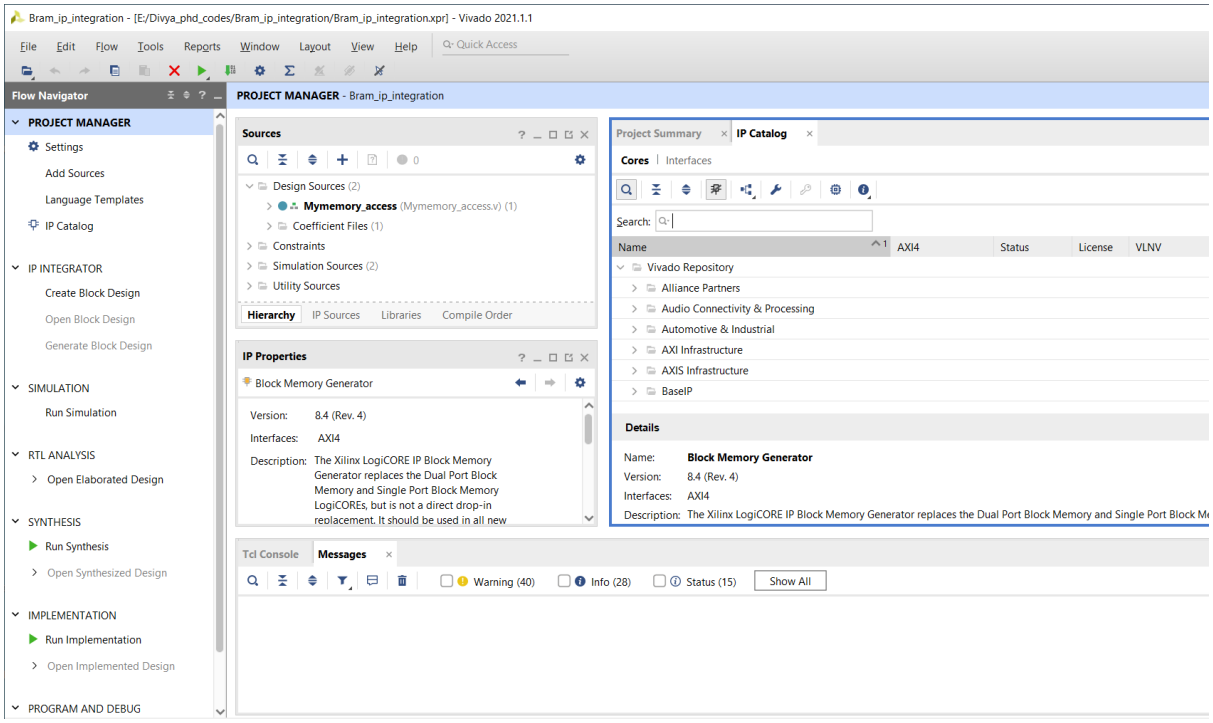
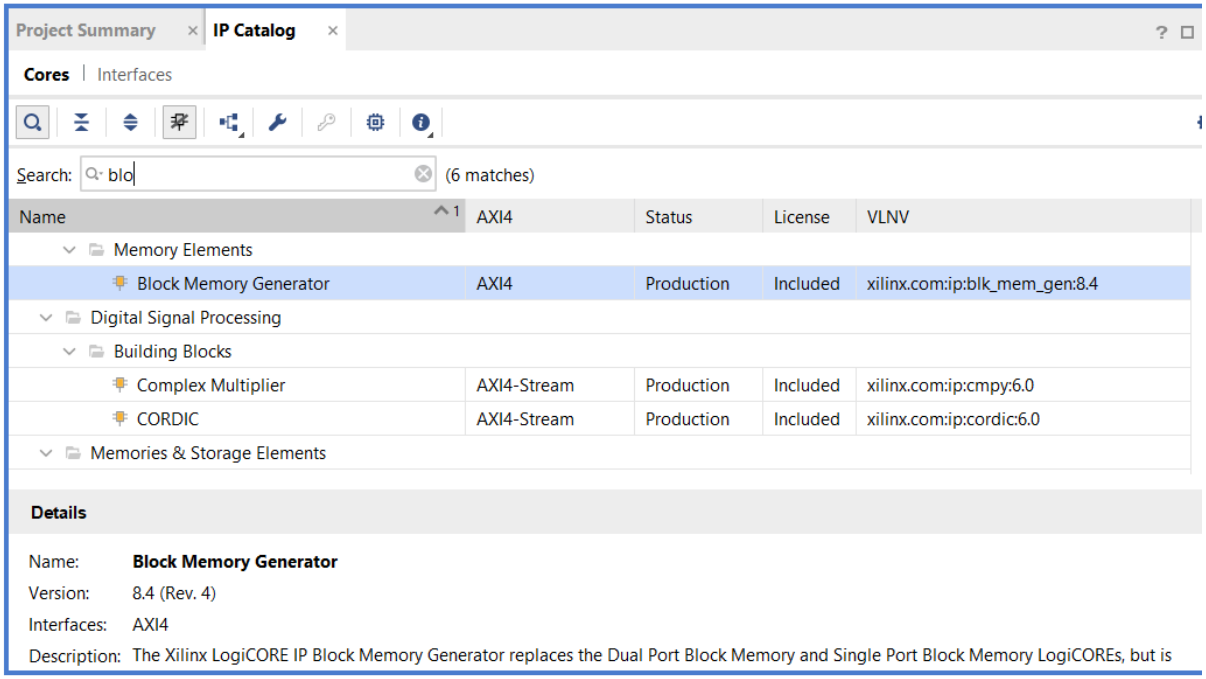
BRAM IP Integration

In Vivado , BRAM IP can be added to your design by following steps:

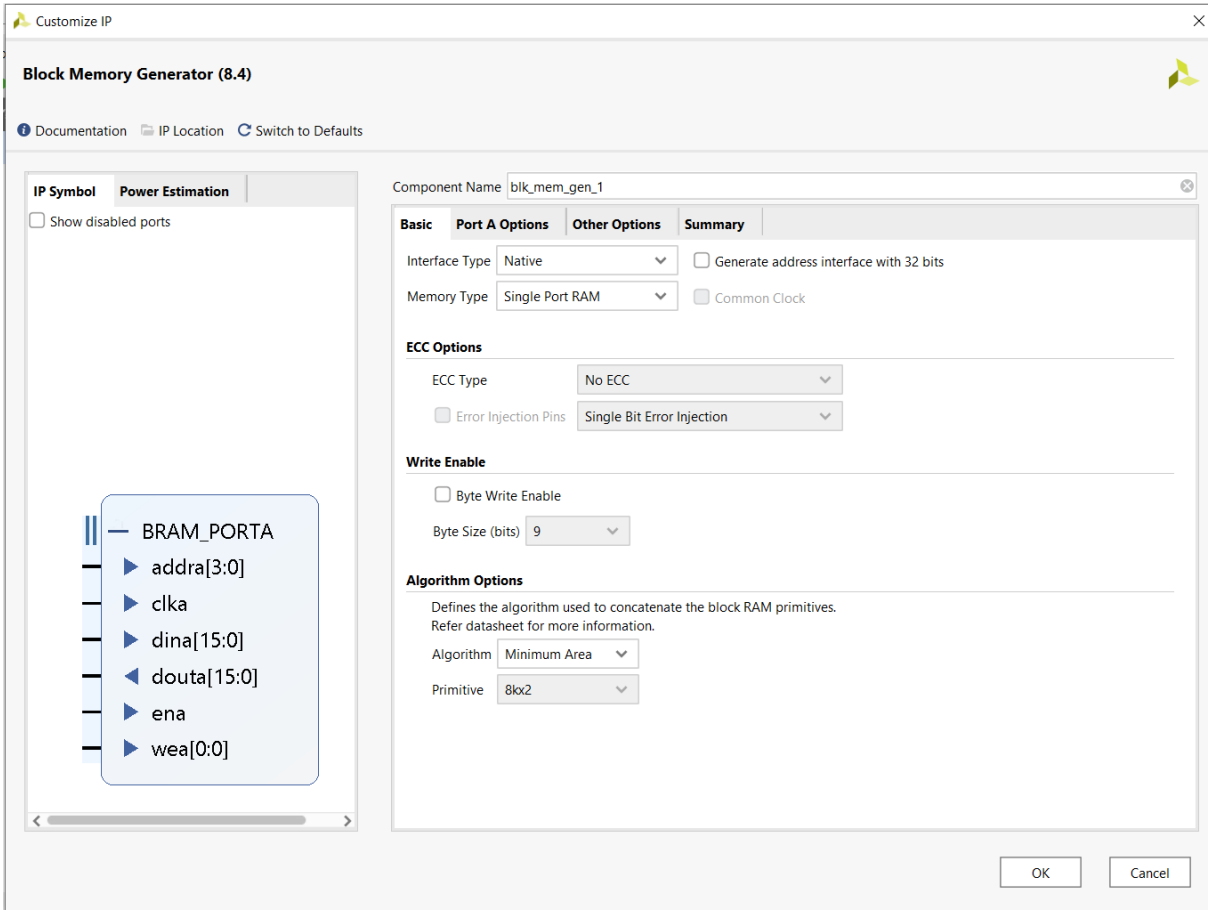
1.After creating a project, click on **IP Catalog** from the Flow Navigator pane.

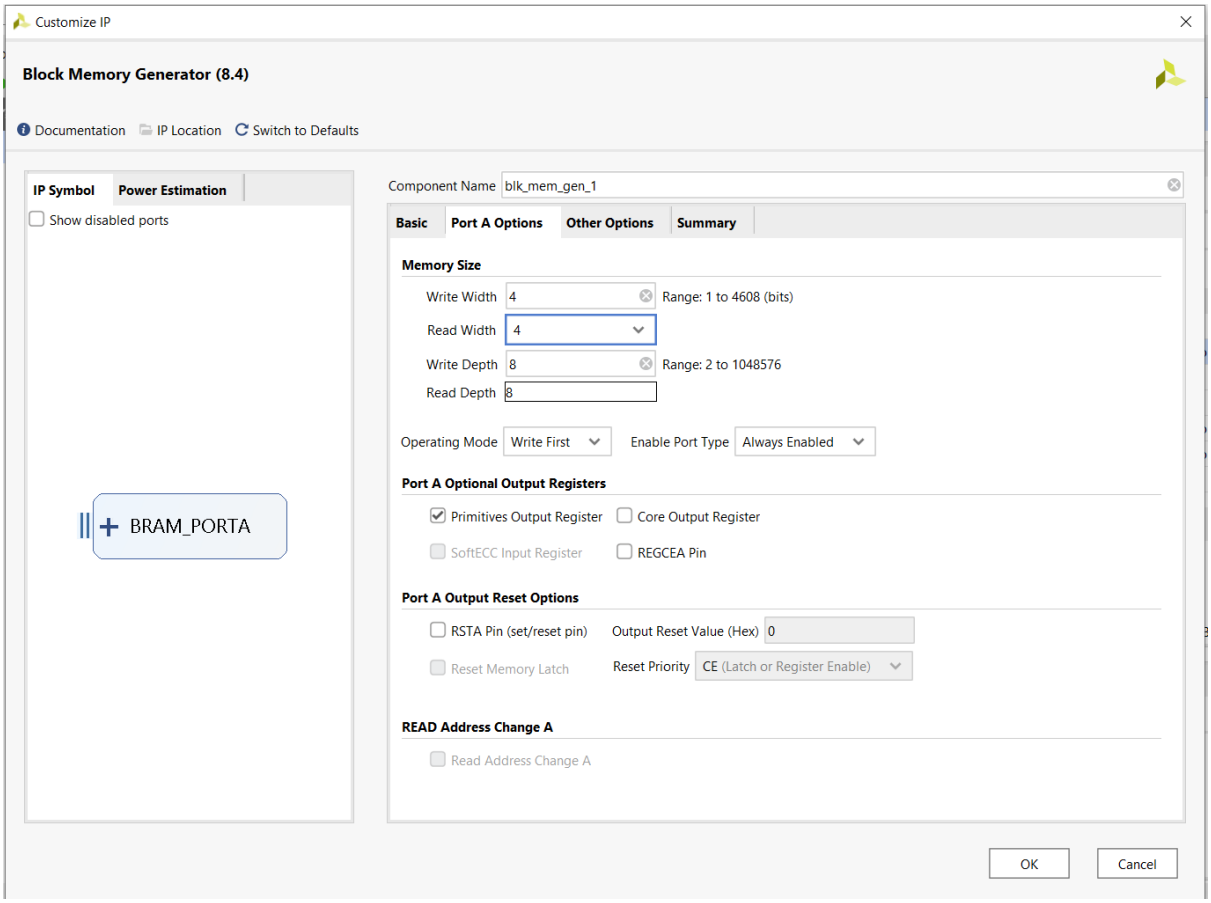


2.From IP Catalog select Block Memory Generator IP as shown below:



3.Customize the IP according to the required design. Below , it is customized to create a Single Port Bram with width=4 and depth=8.





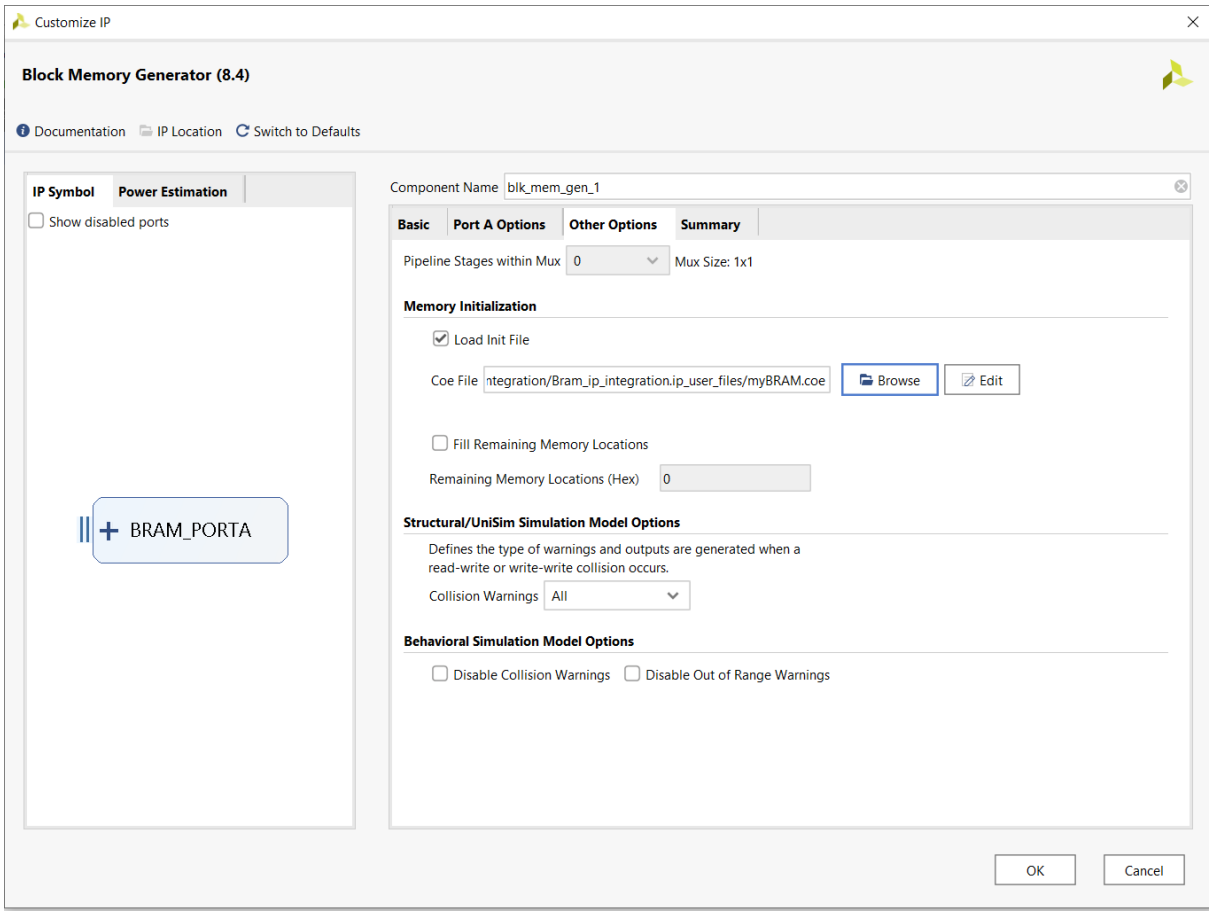
4.The initial data in the BRAM can be loaded either by a mem or a coe file. Below, it is loaded using a coe file.

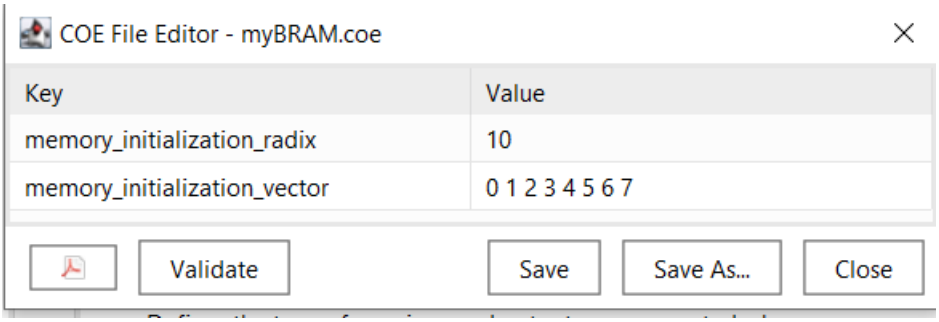
myBRAM.coe - Notepad

File Edit Format View Help

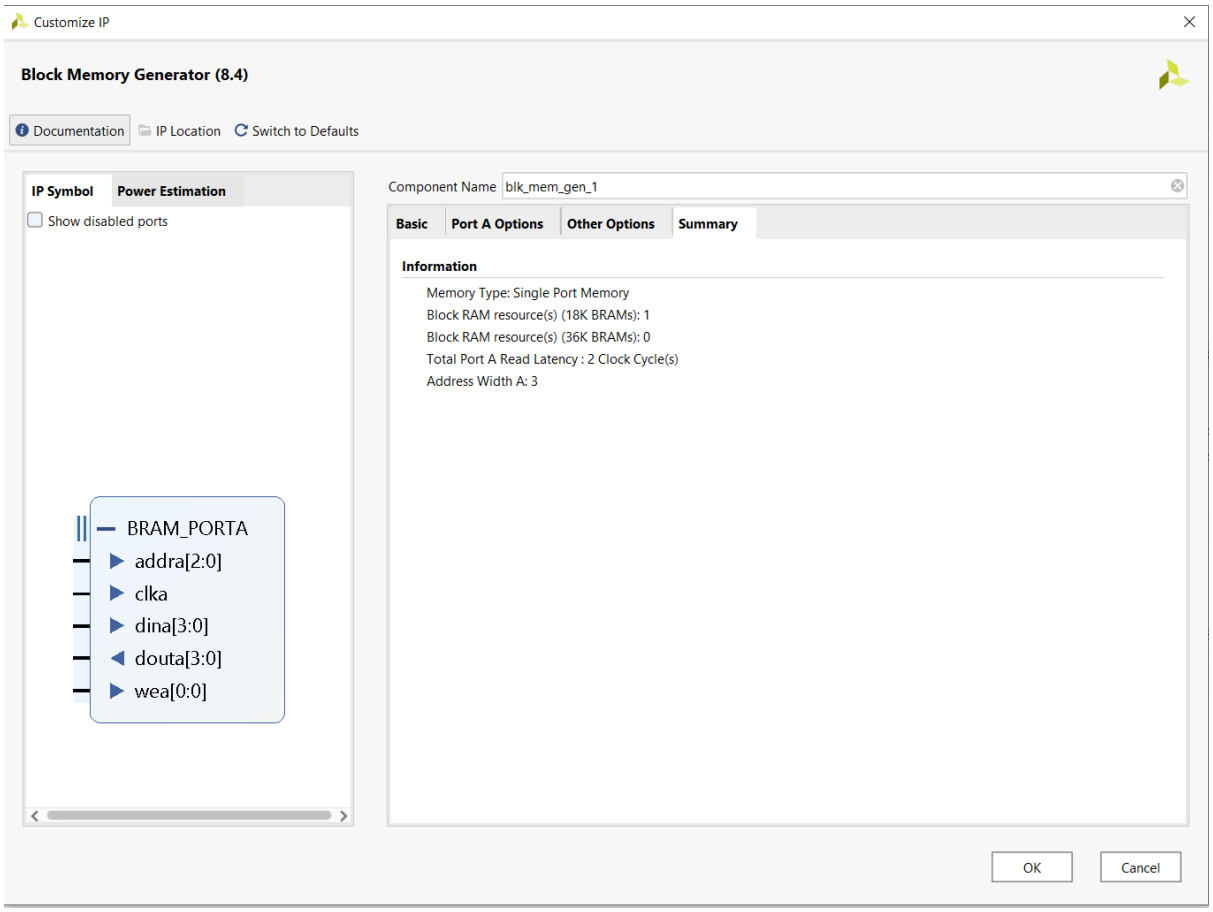
```
memory_initialization_radix=10;
memory_initialization_vector=0 1 2 3 4 5 6 7 ;
```

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

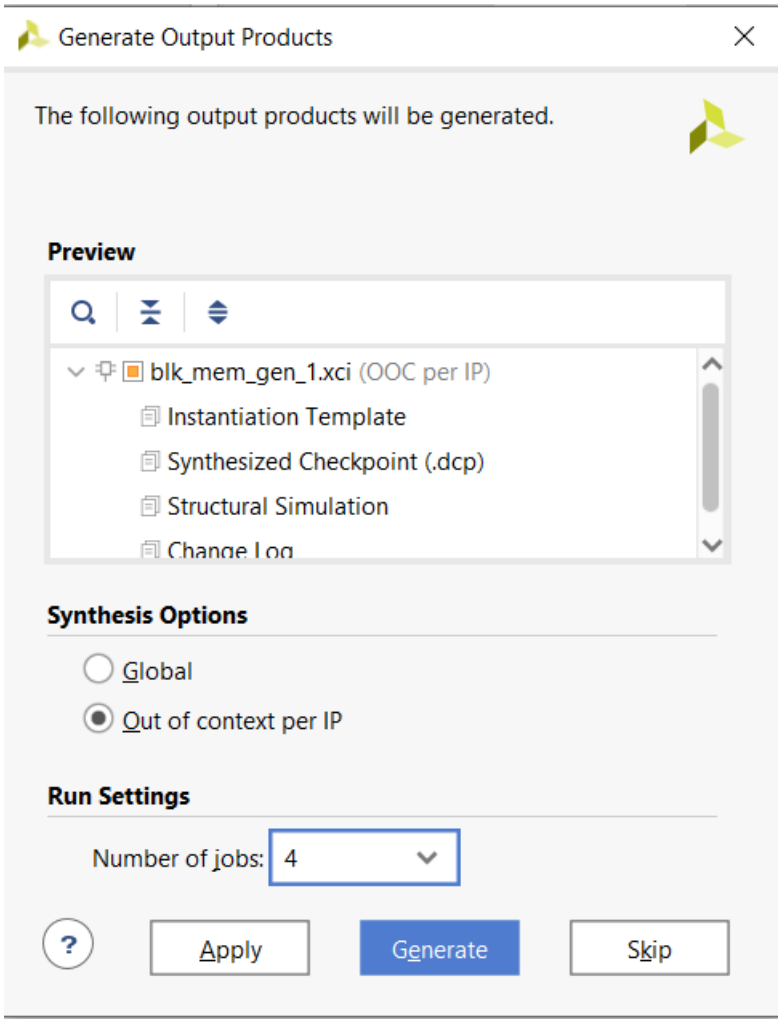




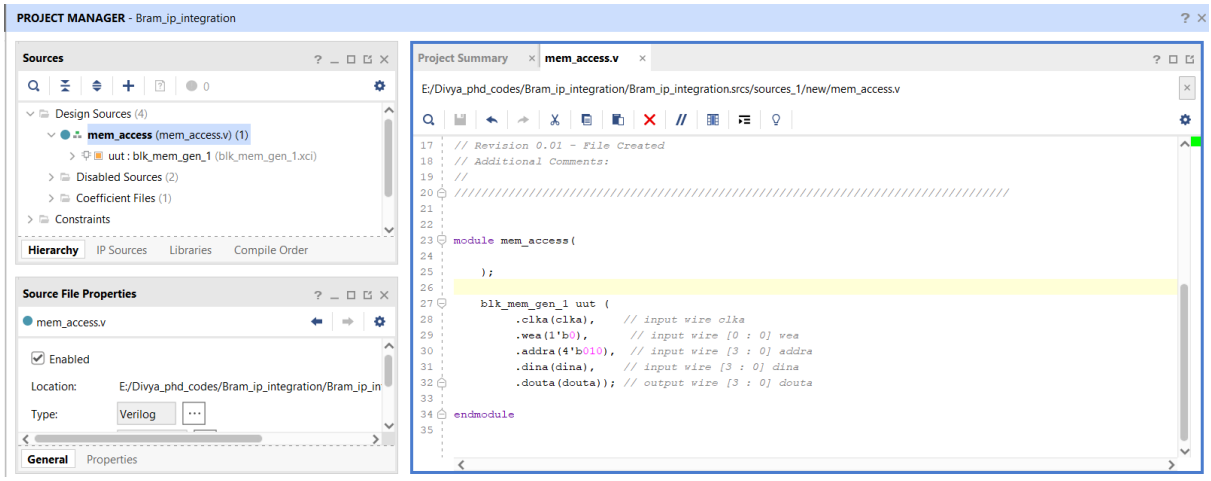
5.The summary of the customized IP can be seen as follows:



6. After clicking on OK , Generate Output Products dialogue box will appear. Click on Generate.



7. Now the generated BRAM of desired type, width and depth can be instantiated in the main module as follows:



Q1. You are required to design a Verilog module which has three Block RAMs:

- **RAM_A** → 16-bit wide, depth 128 (addresses 0–127).
- **RAM_B** → 16-bit wide, depth 256 (addresses 0–255).
- **RAM_F** → 1-bit wide, depth 256 (addresses 0–255).

The design must perform the following tasks:

Part A: Reading Data and Control

1. Read data sequentially from **RAM_A[0…127]**.
2. Read data in reverse order from **RAM_B[127…0]**.
3. For each index i, read the **mode bit** from **RAM_F[i]** (addresses 0–127).

If mode = 0:

Result[i]=RAM_A[i]+RAM_B[127−i]

If mode = 1:

Result[i]=RAM_A[i]−RAM_B[127−i]

Part B: Writing Results

1. Store the **computed result** into **RAM_B[128+i]** for each i (addresses 128–255).
2. Detect **overflow or underflow**:
    - For addition, overflow occurs if the result exceeds 16 bits (carry out of MSB).
    - For subtraction, underflow occurs if the result is negative (borrow).
3. Write the detected **overflow/underflow flag (1-bit)** into **RAM_F[128+i]** (addresses 128–255).

Part C: Processing Sequence

- Repeat the operation for all 128 values.
- At the end of the process:
    - **RAM_B [128…255]** contains results.
    - **RAM_F [128…255]** contains corresponding overflow/underflow flags.
- Write a testbench to show the outcomes.