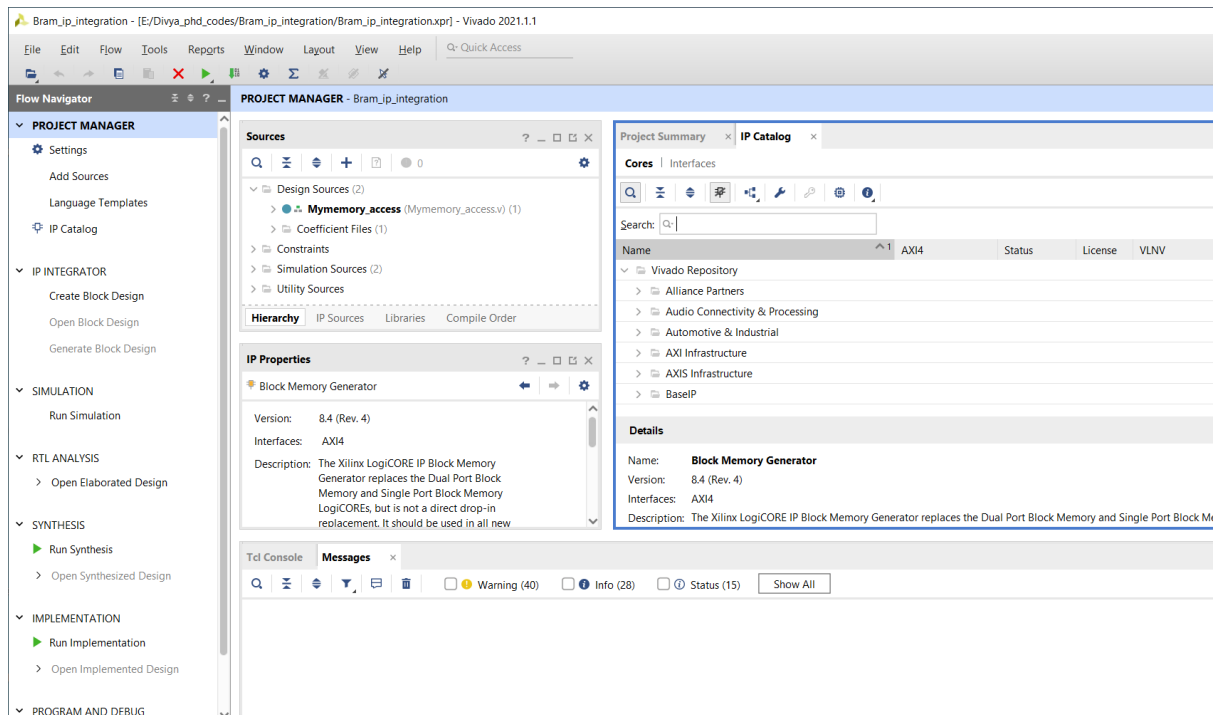


BRAM IP Integration

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:

The screenshot shows the 'IP Catalog' window with the search bar containing 'blk'. The search results show 6 matches. The 'Block Memory Generator' IP is selected, showing its details.

Name	AXI4	Status	License	VLNV
Memory Elements				
Block Memory Generator	AXI4	Production	Included	xilinx.com:ip:blk_mem_gen:8.4
Digital Signal Processing				
Building Blocks				
Complex Multiplier	AXI4-Stream	Production	Included	xilinx.com:ip:cmpy:6.0
CORDIC	AXI4-Stream	Production	Included	xilinx.com:ip:cordic:6.0
Memories & Storage Elements				

Details

Name: **Block Memory Generator**
Version: 8.4 (Rev. 4)
Interfaces: AXI4
Description: The Xilinx LogiCORE IP Block Memory Generator replaces the Dual Port Block Memory and Single Port Block Memory LogiCOREs, but is

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.

The screenshot shows the 'Customize IP' window for the 'Block Memory Generator (8.4)'. The 'Basic' tab is selected, showing the configuration options.

Component Name: blk_mem_gen_1

Basic | Port A Options | Other Options | Summary

Interface Type: Native ☐ Generate address interface with 32 bits
Memory Type: Single Port RAM ☐ Common Clock

ECC Options

ECC Type: No ECC
☐ Error Injection Pins: Single Bit Error Injection

Write Enable

☐ Byte Write Enable
Byte Size (bits): 9

Algorithm Options

Defines the algorithm used to concatenate the block RAM primitives. Refer datasheet for more information.
Algorithm: Minimum Area
Primitive: 8lx2

IP Symbol | Power Estimation

☐ Show disabled ports

BRAM_PORTA

- addra[3:0]
- clka
- dina[15:0]
- douta[15:0]
- ena
- wea[0:0]

OK Cancel

Customize IP

Block Memory Generator (8.4)

Documentation
IP Location
Switch to Defaults

IP Symbol

Power Estimation

Show disabled ports

+ BRAM_PORTA

Component Name

blk_mem_gen_1

Basic

Port A Options

Other Options

Summary

Memory Size

Write Width

4

Range: 1 to 4608 (bits)

Read Width

4

Write Depth

8

Range: 2 to 1048576

Read Depth

8

Operating Mode

Write First

Enable Port Type

Always Enabled

Port A Optional Output Registers

☒ Primitives Output Register
☐ Core Output Register

☐ SoftECC Input Register
☐ REGCEA Pin

Port A Output Reset Options

☐ RSTA Pin (set/reset pin)

Output Reset Value (Hex) 0

☐ Reset Memory Latch

Reset Priority CE (Latch or Register Enable)

READ Address Change A

☐ Read Address Change A

OK

Cancel

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

Customize IP

Block Memory Generator (8.4)

[Documentation](#) [IP Location](#) [Switch to Defaults](#)

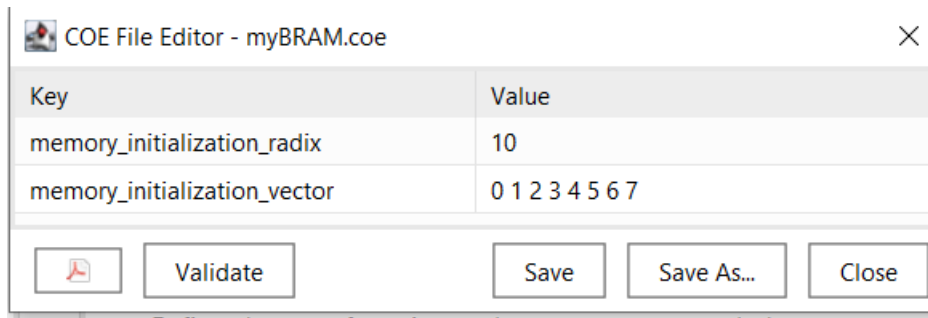
IP Symbol **Power Estimation**

☐ Show disabled ports

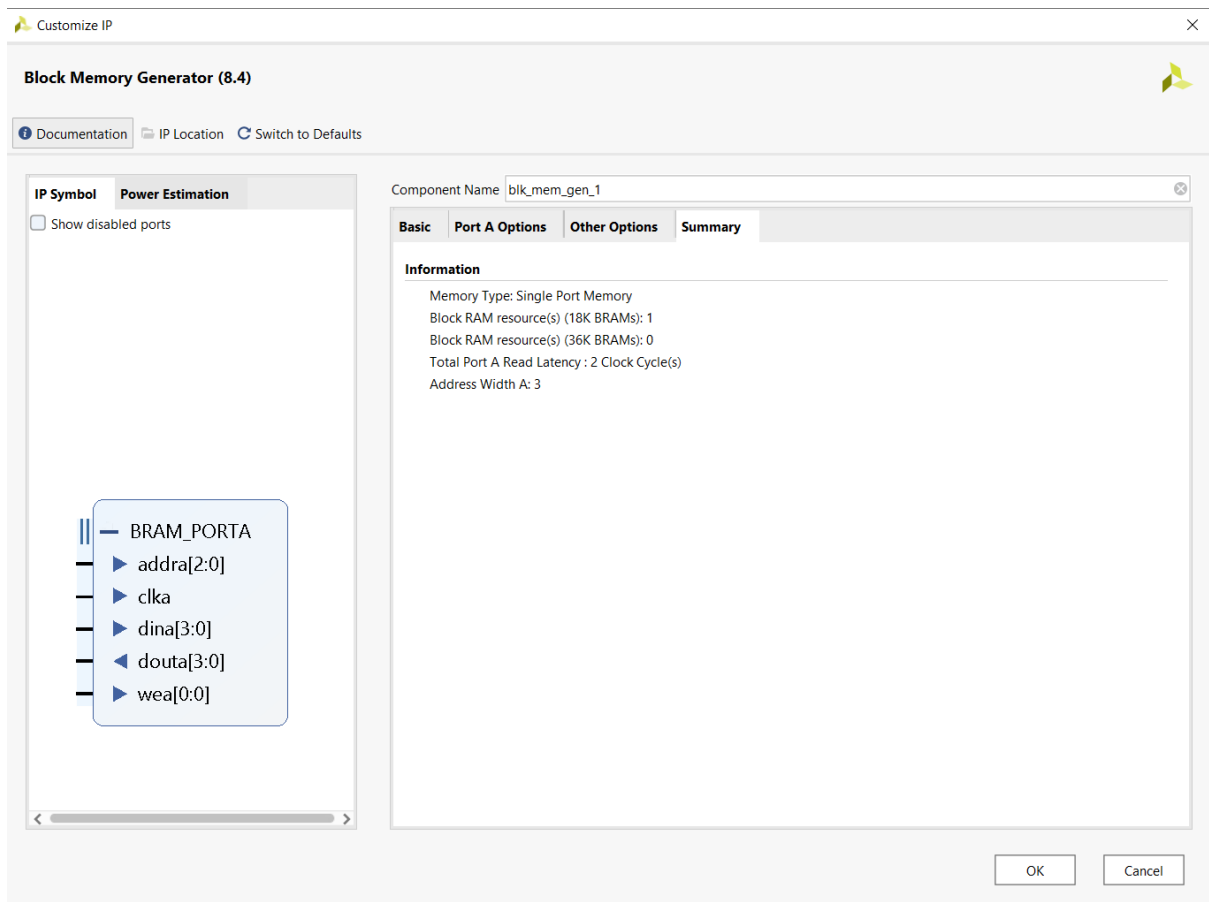
+ BRAM_PORTA

Component Name: blk_mem_gen_1

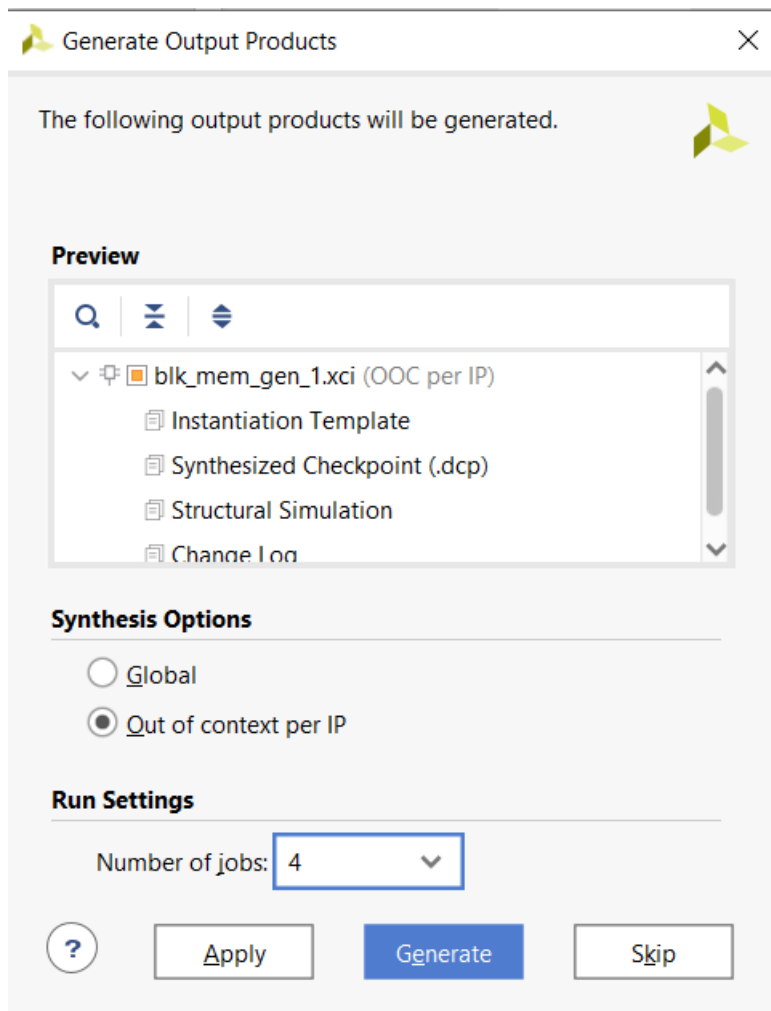
Basic	Port A Options	Other Options	Summary
Pipeline Stages within Mux: 0 Mux Size: 1x1			
Memory Initialization			
<input checked="" type="checkbox"/> Load Init File			
Coe File: integration/Bram_ip_integration.ip_user_files/myBRAM.coe		<input type="button" value="Browse"/>	<input type="button" value="Edit"/>
<input type="checkbox"/> Fill Remaining Memory Locations			
Remaining Memory Locations (Hex): 0			
Structural/UniSim Simulation Model Options			
Defines the type of warnings and outputs are generated when a read-write or write-write collision occurs.			
Collision Warnings: All			
Behavioral Simulation Model Options			
<input type="checkbox"/> Disable Collision Warnings <input type="checkbox"/> Disable Out of Range Warnings			



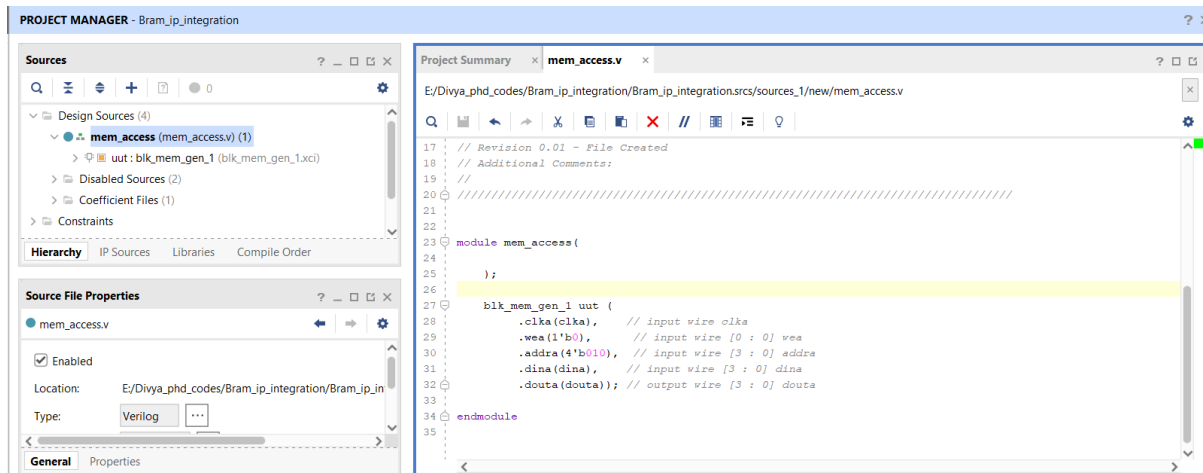
5.The summary of the customized IP can be seen as follows:



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:

$$\text{Result}[i] = \text{RAM_A}[i] + \text{RAM_B}[127-i]$$

If mode = 1:

$$\text{Result}[i] = \text{RAM_A}[i] - \text{RAM_B}[127-i]$$

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.