Generating Images for PYNQ Image Overlay Filter

# Generate Image to Display

The images used in the overlay need to be in a particular format. The image overlay filter requires each pixel color to be one of 256 colors defined by its index in the 256-color array. The images are also limited in size due to the amount of BRAM available in the partially reconfigurable region provided on the Zynq-7000 FPGA. The following steps explain the process.

## Resize Image

The image must fit in the image overlay filter’s memory. A maximum of 40,960 pixels can be stored. The default image is 200x200 pixels (40,000 pixels in total).

1. Open your image in your desired image editing program (Paint, GIMP, etc.) and resize it so that it has a maximum of 40,960 pixels.

## Convert Image to Type PPM

The program that generates the files needed to display the image requires an input image of type PPM. This file type specifies the values of each individual pixel.

2. Use a free online tool to convert a JPEG or GIF image to a PPM image.

JPG: https://convertio.co/jpg-ppm/

GIF: https://convertio.co/gif-ppm/

## Generate Files for Display

3. Change directory to ConvertImage

$ cd /path/to/Documentation/ConvertImage

4. Make the convert\_image executable

$ make

5. Run convert\_image on your input image.

$ ./convert\_image /path/to/input.ppm /path/to/output.ppm \

/path/to/image\_name

This will produce 3 separate files. The output.ppm file is the new image that uses only 256 colors\*. The second is a .bin file. This should be copied to the Jupyter notebook so that the new image can be displayed using the python interface. The third is a .coe file. This can be used to change the default image in the filter. See “Change Default Image”.

\*Sometimes the conversion program uses some strange colors when it converts to the 256-color palette (e.g. brown 🡪 green). If this happens, you can open the new image in an image editor (you might have to convert it back to a JPG or GIF) and change the undesirable colors to a better choice (the eyedropper tool is great for this if there is an acceptable substitute color already in the image). Then repeat steps 2-5 with the edited image.

# Display Image

The following instructions assume you are modifying the img\_256color\_filter.ipynb notebook.

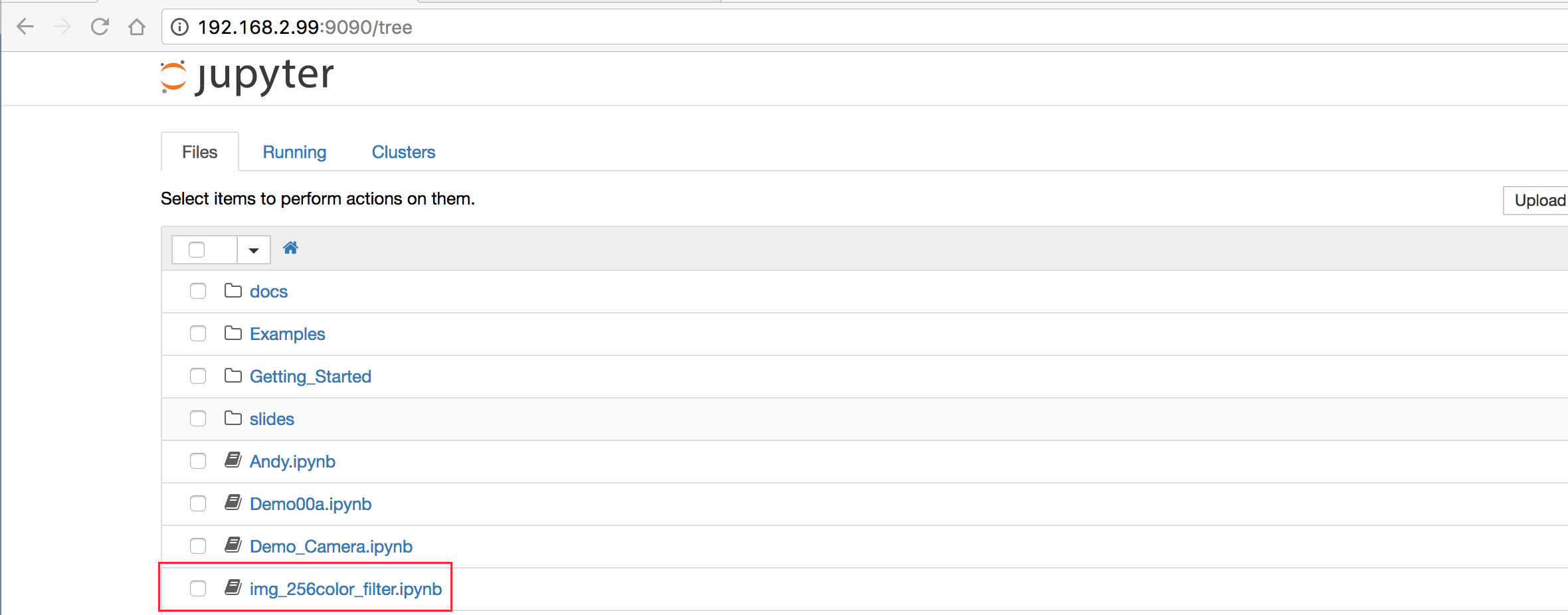
1. Copy the .bin image output files to the PYNQ in the same directory as the img\_256color\_filter.ipynb notebook.

$ sftp xilinx@192.168.2.99

$ cd jupyter\_notebooks

$ put image\_name.bin

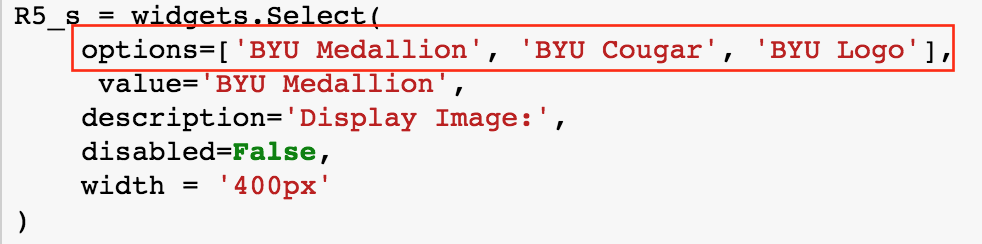
2. Open the img\_256color\_filter.ipynb notebook in your web browser.



Figure

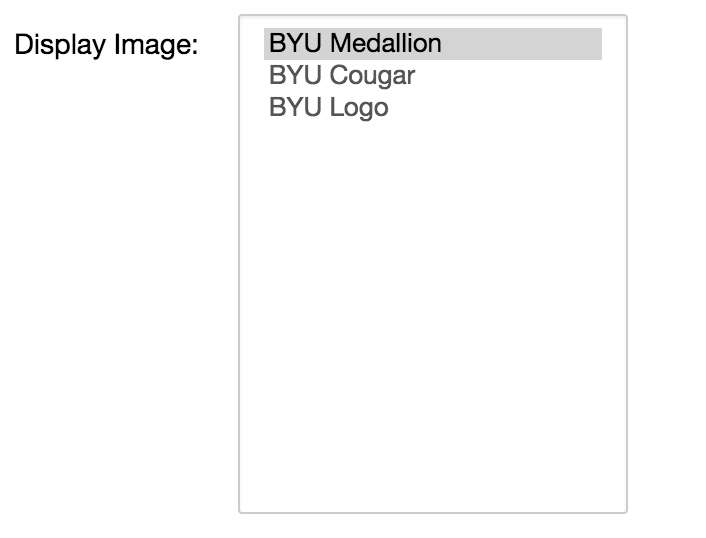
3. Find the code snippet following Step 4.

4. Add another entry to the options list in the R5\_s widget definition. Give it whatever title you want.



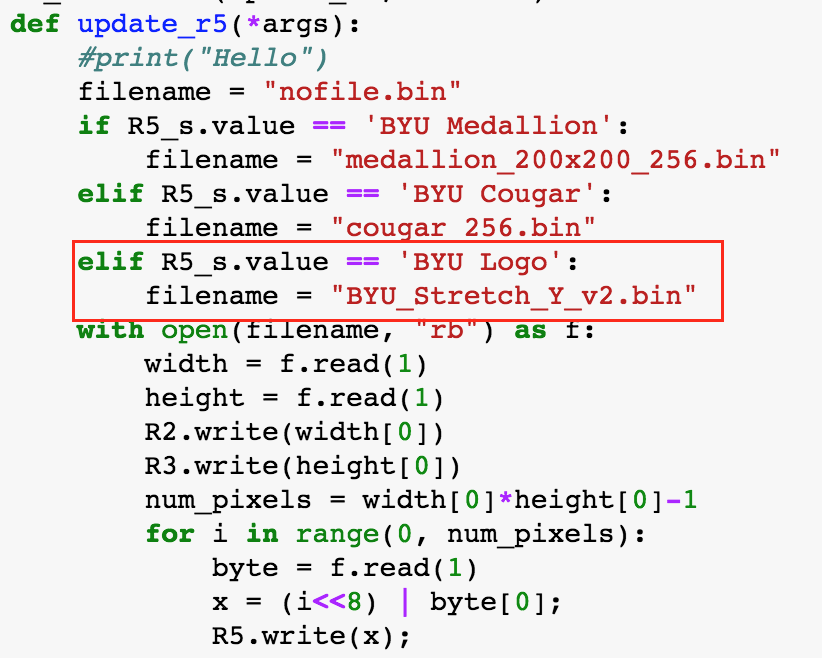
Figure

This will appear in the selection list when you run the notebook.



Figure

5. Add another “elif” statement in the update\_r5 section. Use the title from Step 3 as the value and the image .bin file as the filename.



Figure

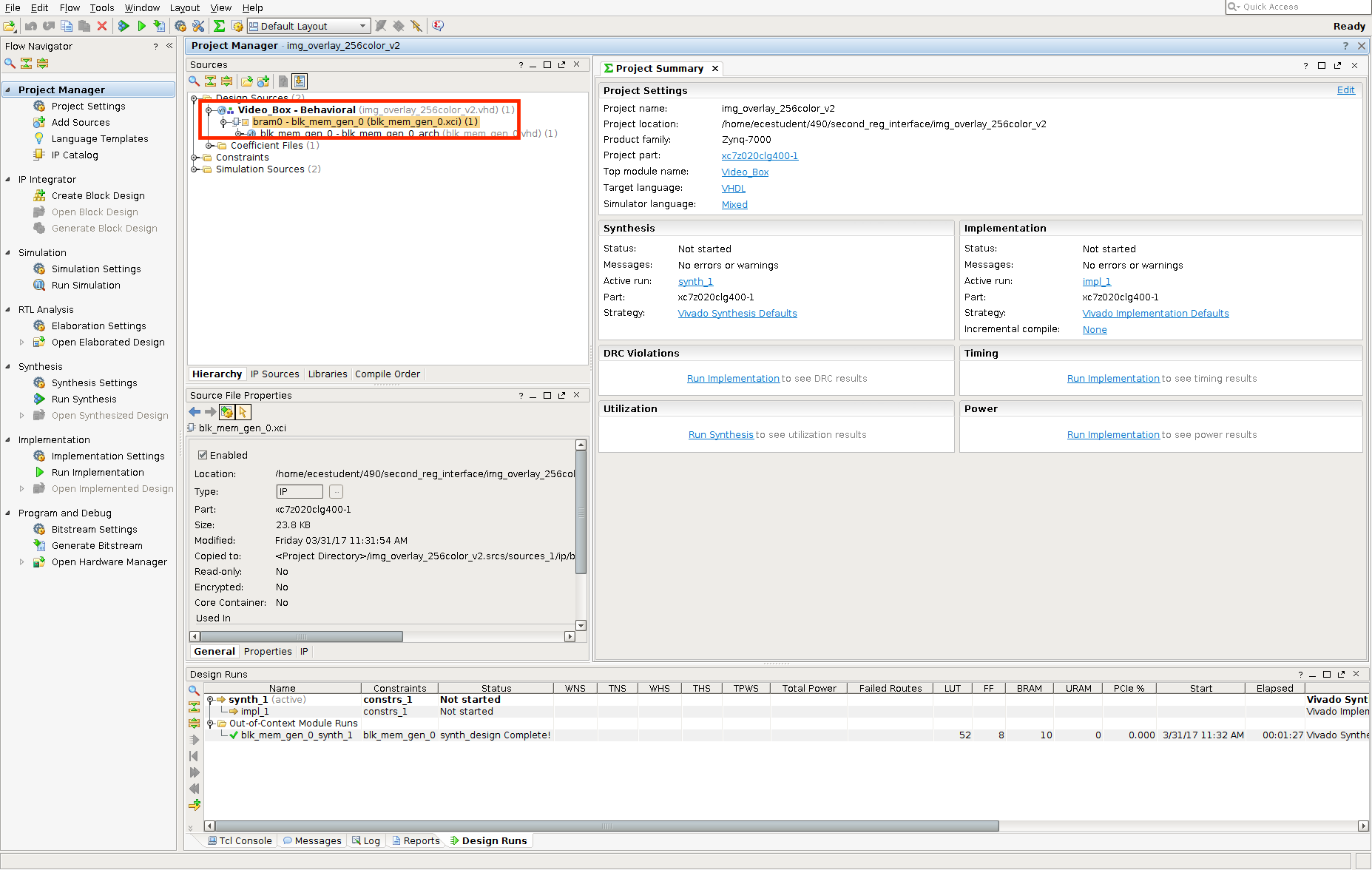
6. Re-run that code block. The new image should appear in the selection list. When you select it, the new image should be displayed.

# Change Default Image

If you want to change the default image, you must rebuild the Vivado project. Assuming you have an existing copy of the project, open the project and follow the instructions below. If not, create a new project with the provided VHDL source and follow the instructions in the “BRAM Instantiation” section.

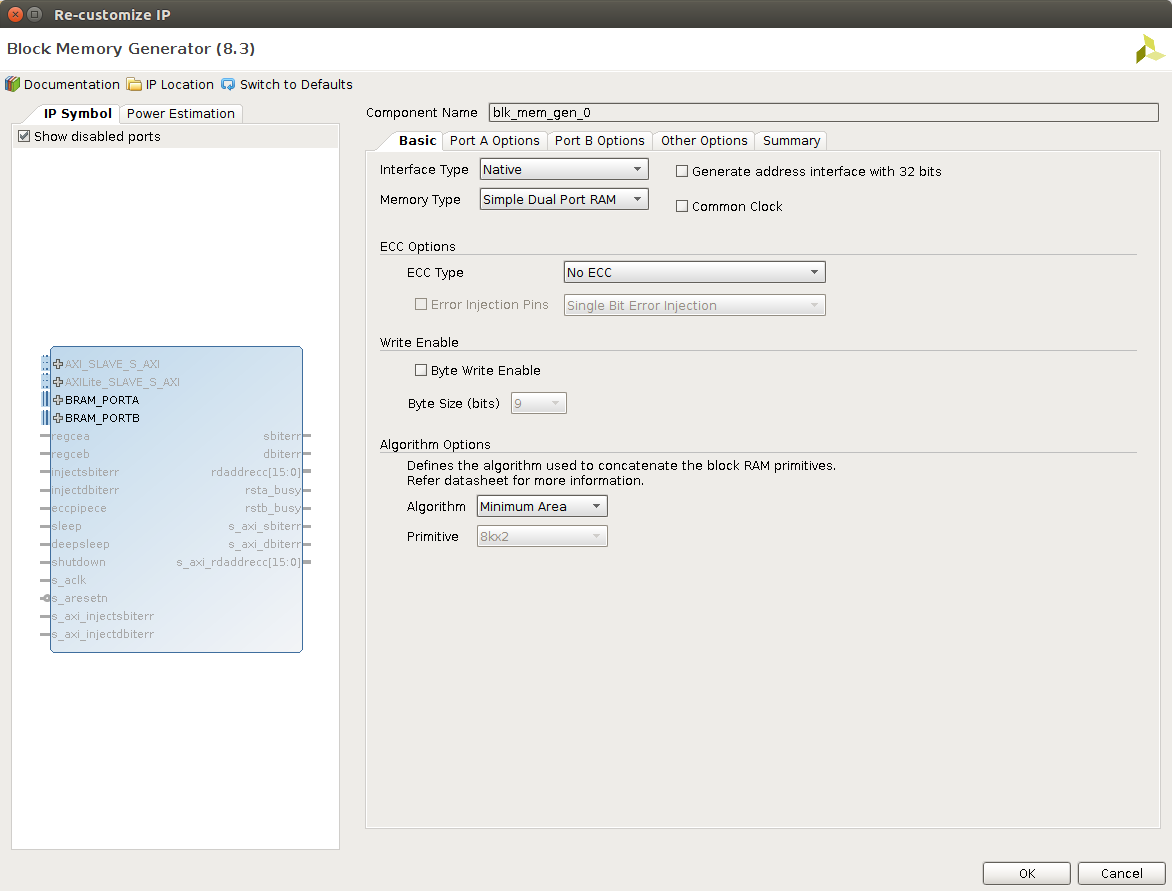
1. Open the bram0 instantiation.

In the Project Manager window, expand the Video\_Box source and double-click on the bram0 instantiation.



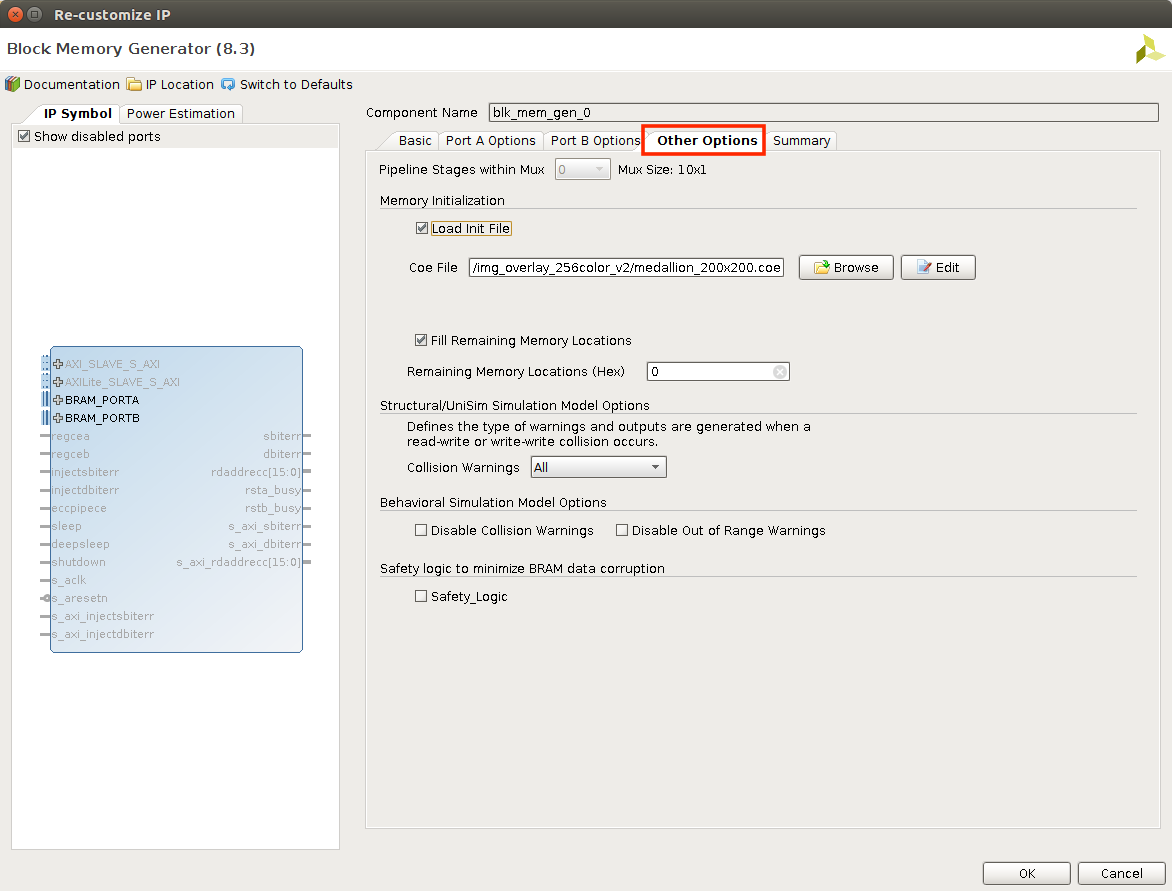
Figure

The following window should pop up.



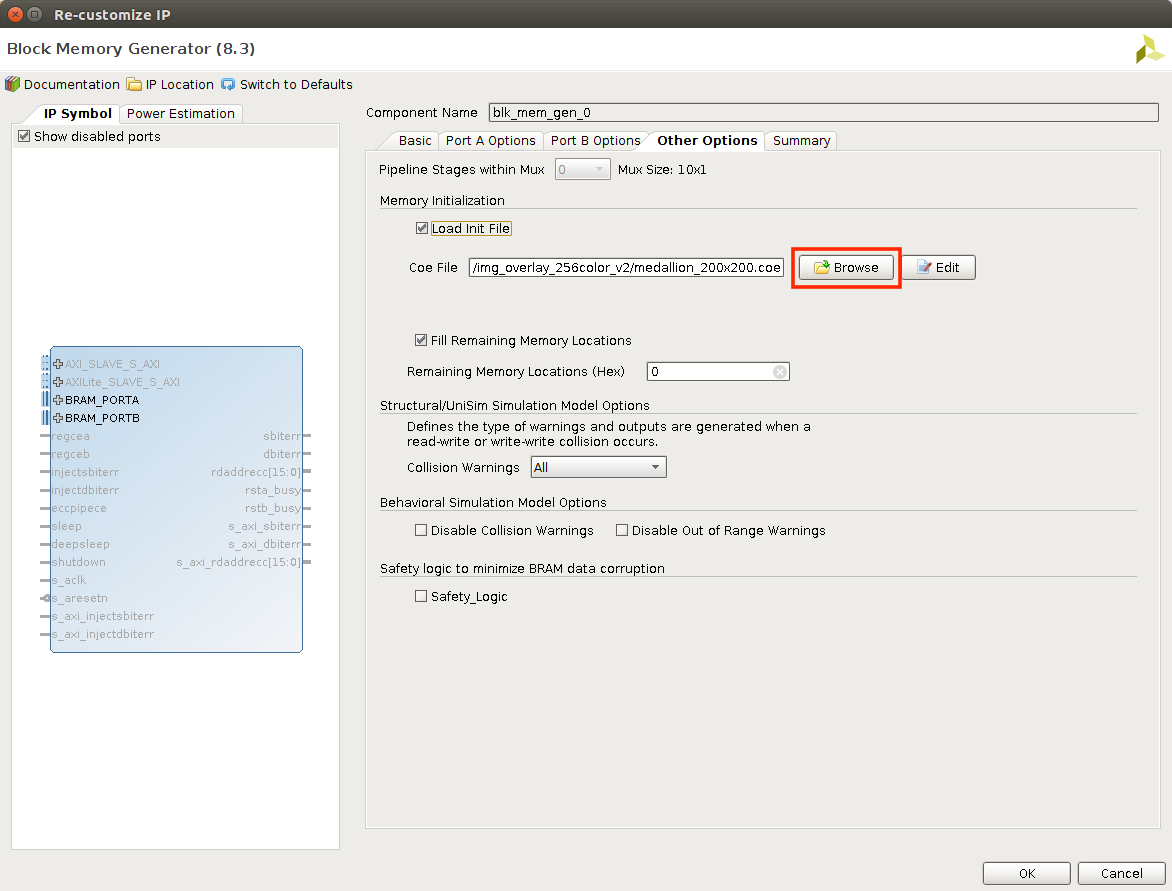
Figure

2. Select the “Other Options” tab.



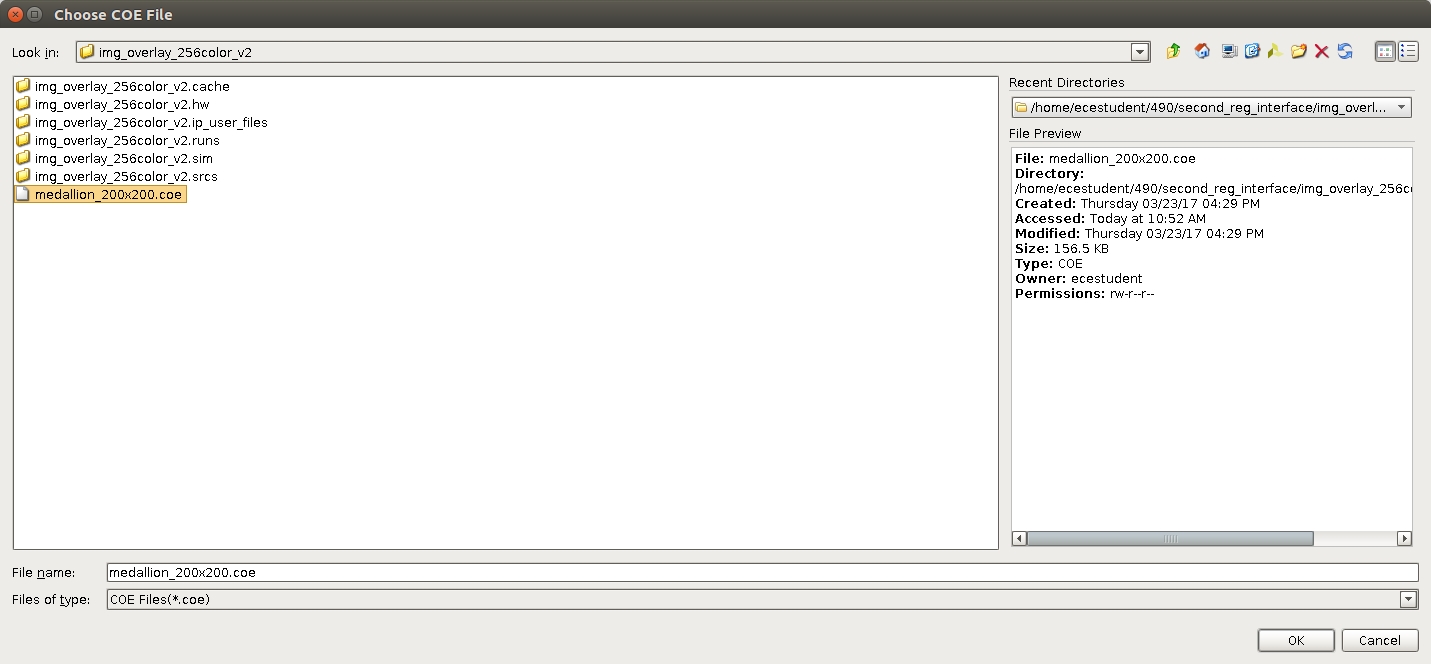
Figure

3. Select the Browse button for the Coe File.



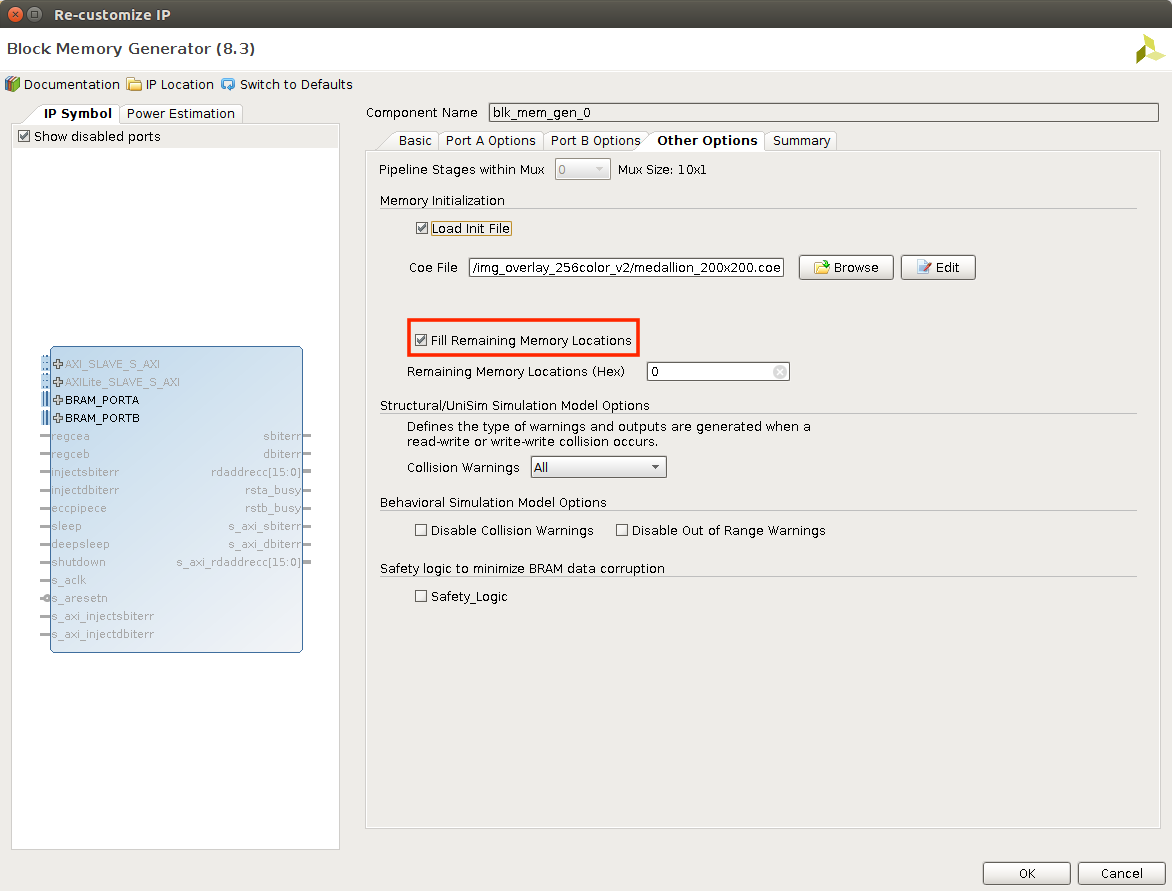
Figure

4. Browse to the COE file you generated from “Generate Image to Display” and click “OK”.



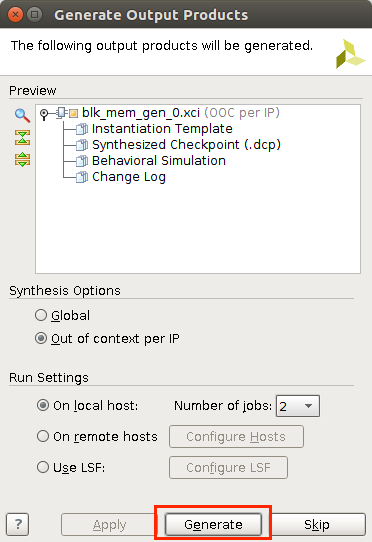
Figure

5. Make sure that “Fill Remaining Memory Locations” is selected.



Figure

7. Click “OK”. Then select “Generate” in the window that pops up.



Figure

8. You will now have to re-synthesize the project and produce a new partial bitstream.

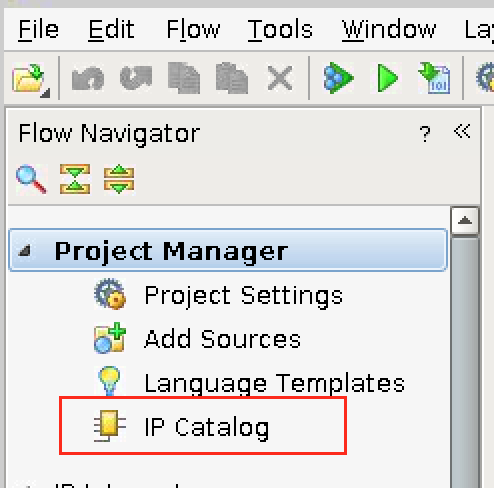
$ source part\_gen.tcl

$ part\_gen img\_overlay\_256color

# BRAM Instantiation

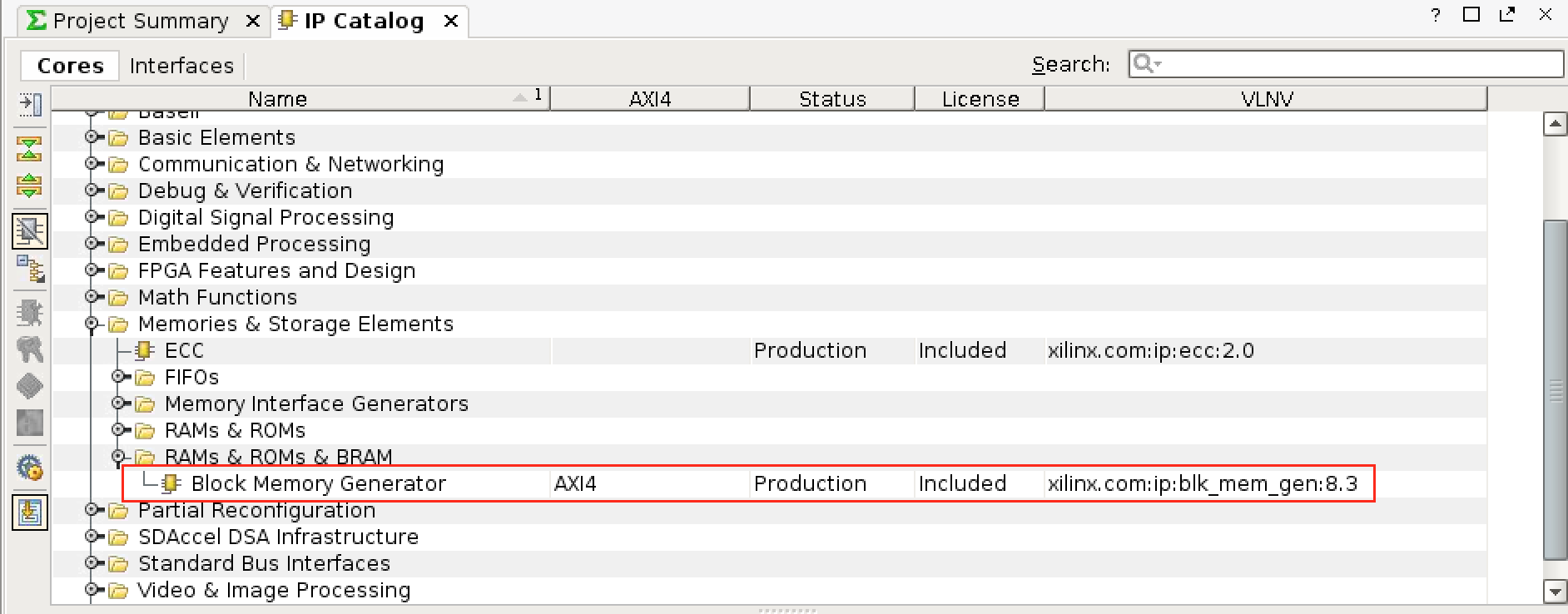
To re-create the project, use the img\_overlay\_256color.vhd file provided as a source file and instantiate the BRAMs as follows.

1. Select IP Catalog from the Project Manager



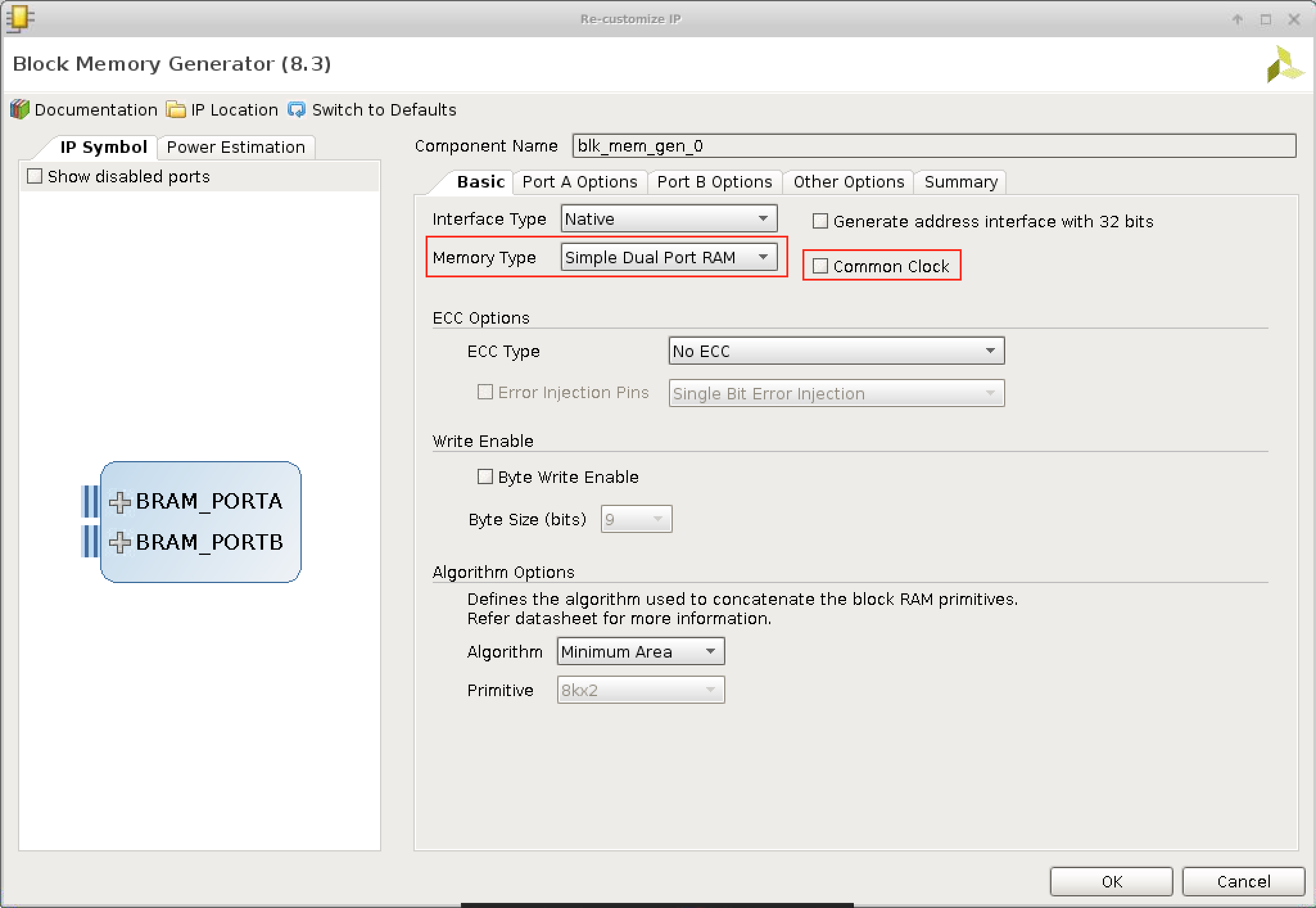
Figure

2. Expand “Memories & Storage Elements”, then expand “RAMs & ROMs & BRAM”. Select the Block Memory Generator.



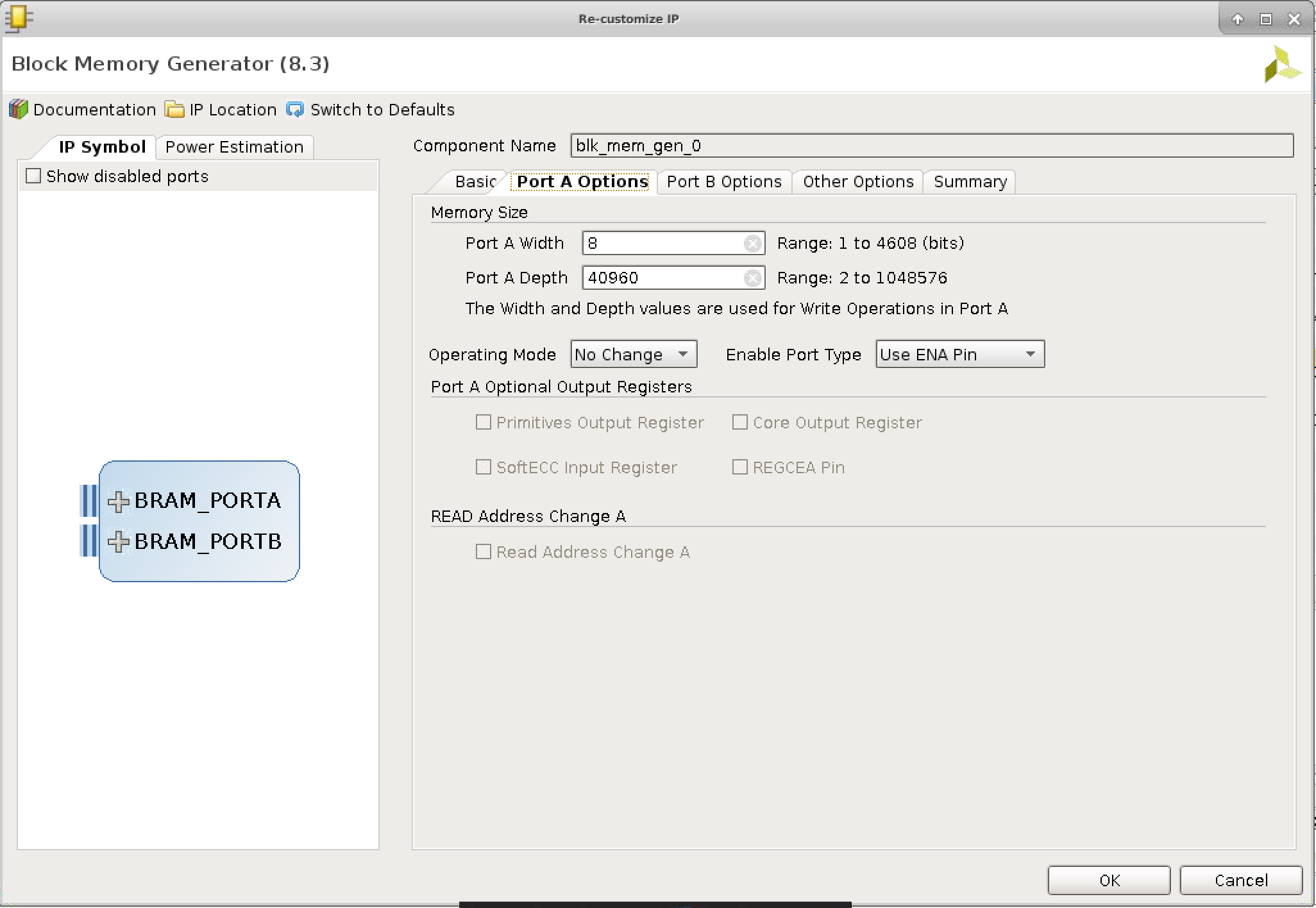
Figure

3. Under the “Basic” tab, select “Simple Dual Port RAM” as the Memory Type. Make sure “Common Clock” is de-selected.



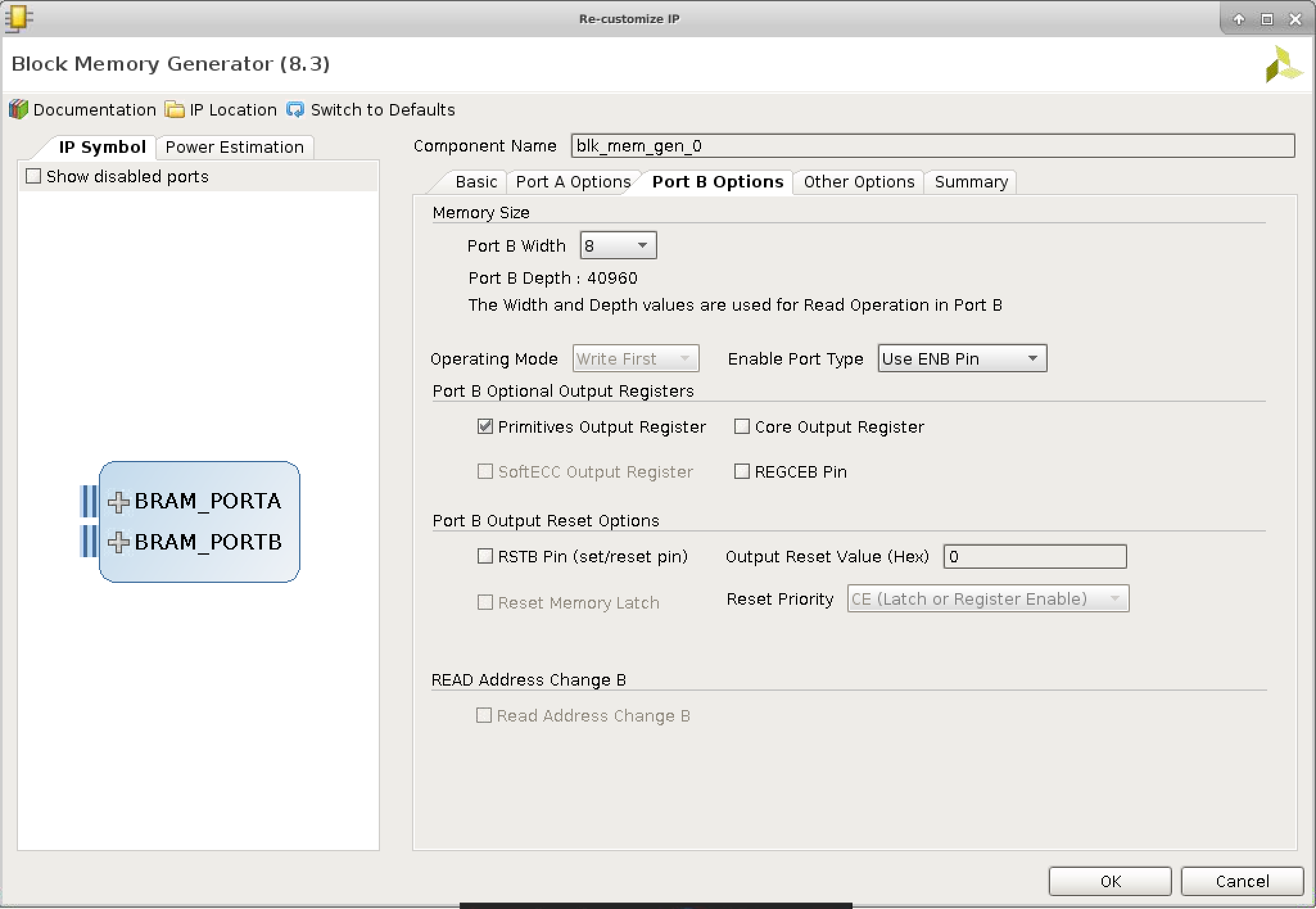
Figure

4. Under the “Port A Options” tab, set the width to 8 and depth to 4096. Use ENA Pin.



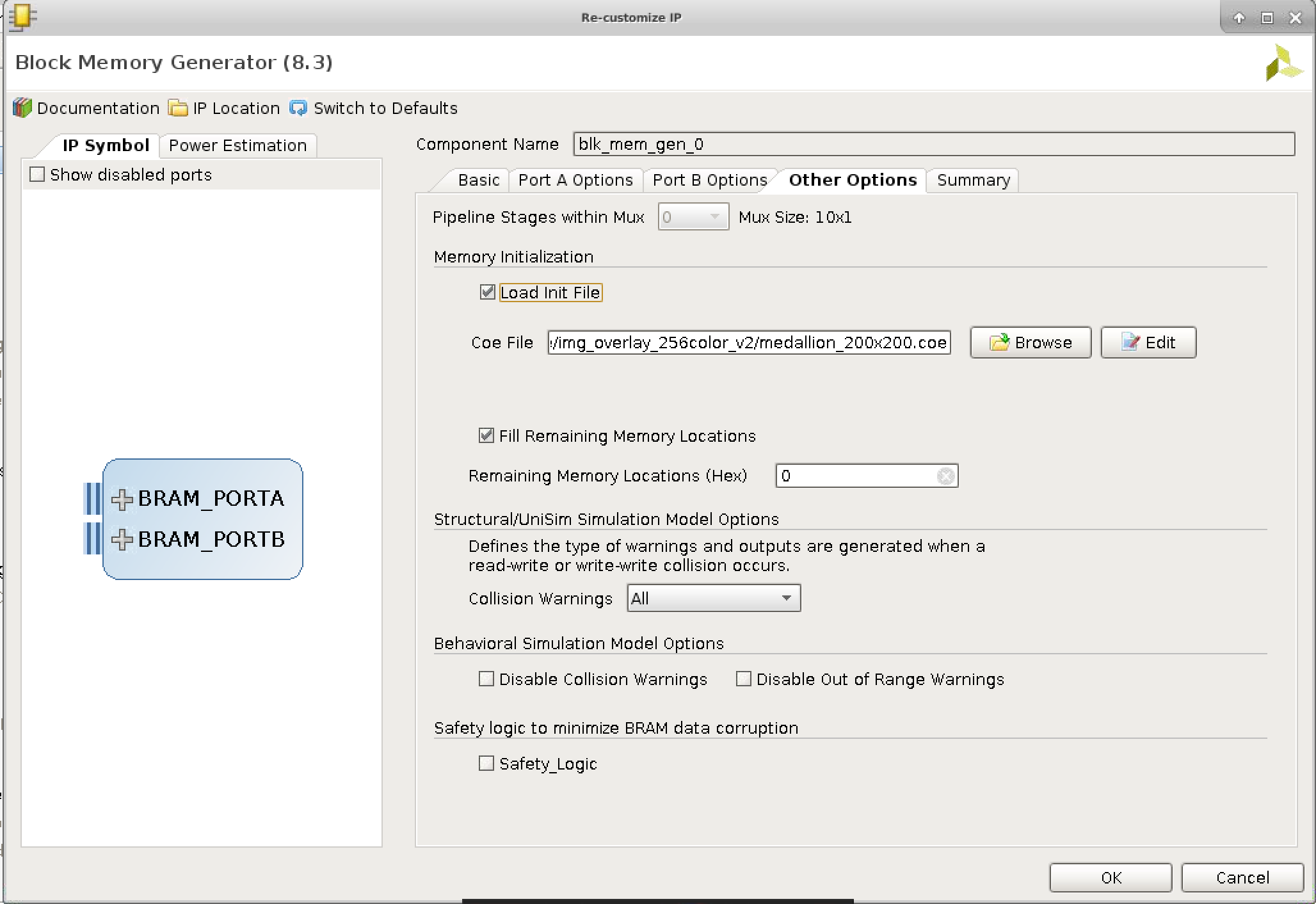
Figure

5. Make the same selections for the “Port B Options.



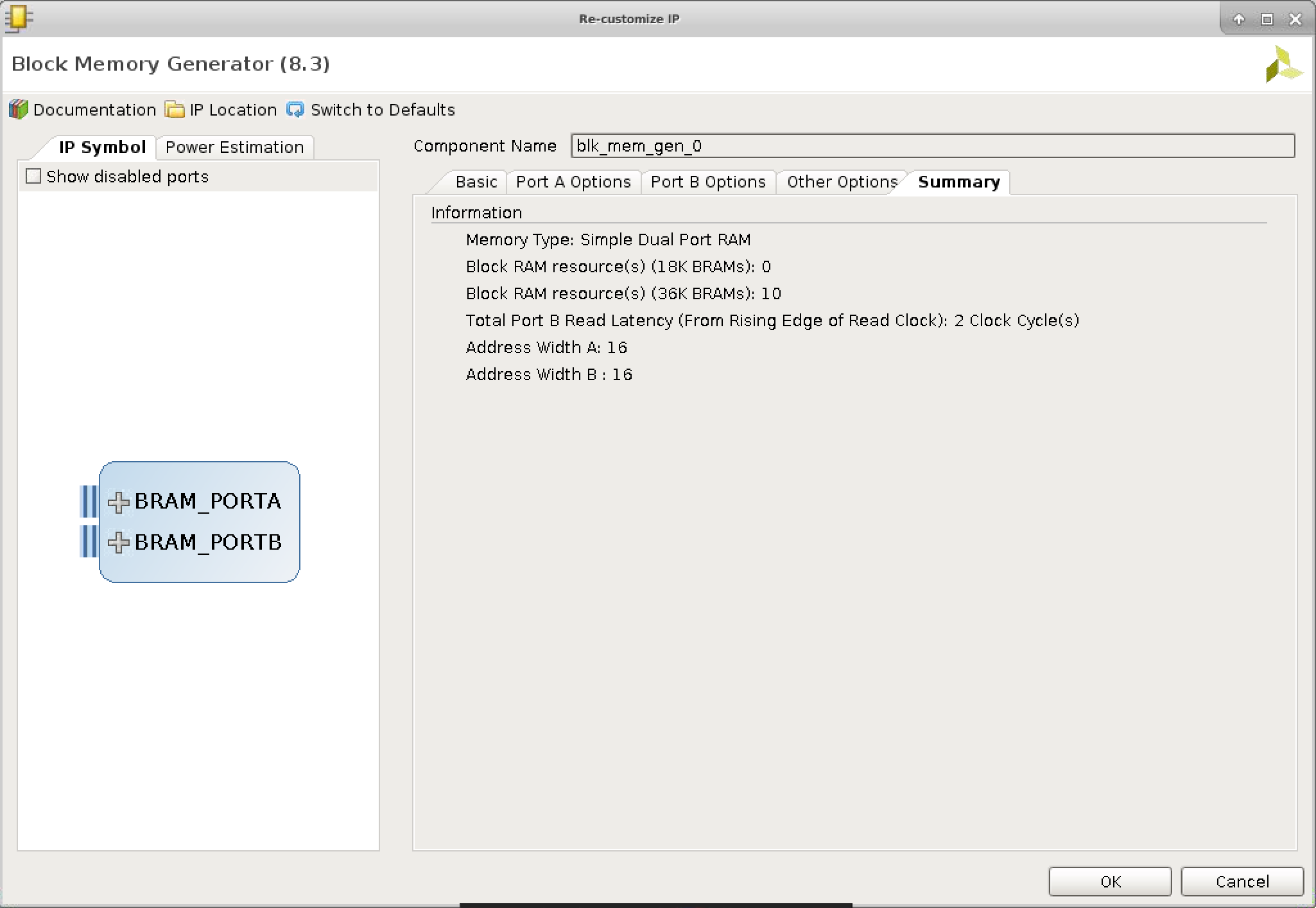
Figure

6. Under “Other Options”, browse to the provided COE file. Select to fill the remaining memory locations with zeroes.



Figure

7. Verify that the summary matches the following:



Figure