1. Create new project -> create new block design

2. add these blocks:

a. zynq7 processing system

i. run block automation and do the DDR and FIXED_IO connections

ii. double click PS -- PS-PL configuration -> GP Master AXI Interface -> M AXI GP0 Interface and make sure its checked

b. reset for ps

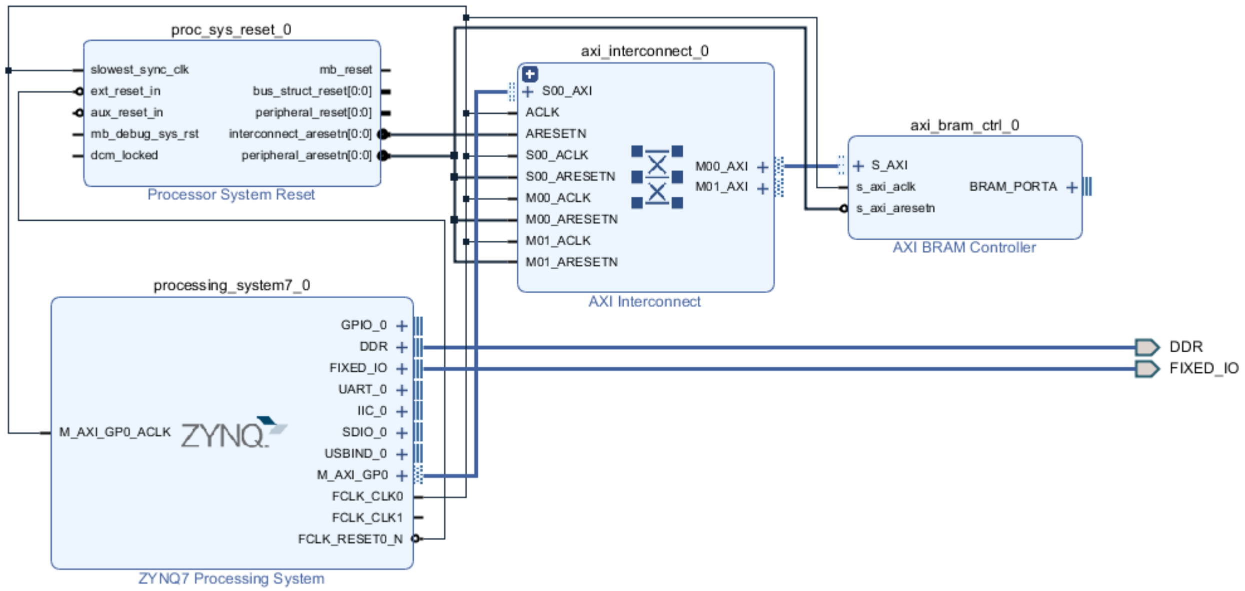
c. axi interconnect

d. axi bram ctrl

i. double click it and changed number of BRAM interfaces to 1 and change to AXI4LITE

3. After all this is added, run connection automation, press regenerate layout to reorder components (*makes it look neater), press validate design and make sure no errors.

4. You should have this after doing everything above, if not, connect all the wires manually and validate design:



5. Tools -> create and package new IP -> new AXI peripheral -> create a name you will remember -> edit IP on the last step and press finish

6. in the package IP vivado -> add all the design sources

7. add this in, the _v1_0.v file, if you are unsure about this step, refer to the github where its uploaded:

a. At line 7:

```
parameter          WIDTH=32, SIZE=512,     //WIDTH is bits per word(shouldn't be changed), SIZE is # of WORDS
    parameter          NUM_COL   = 4,
    parameter          COL_WIDTH = 8,
    localparam integer          LOGSIZE=$clog2(SIZE),
```

b. at line 49 add: `wire ps_reset;`

c. add this inside the instantiation `.ps_reset(ps_reset)`

d. Line 22:

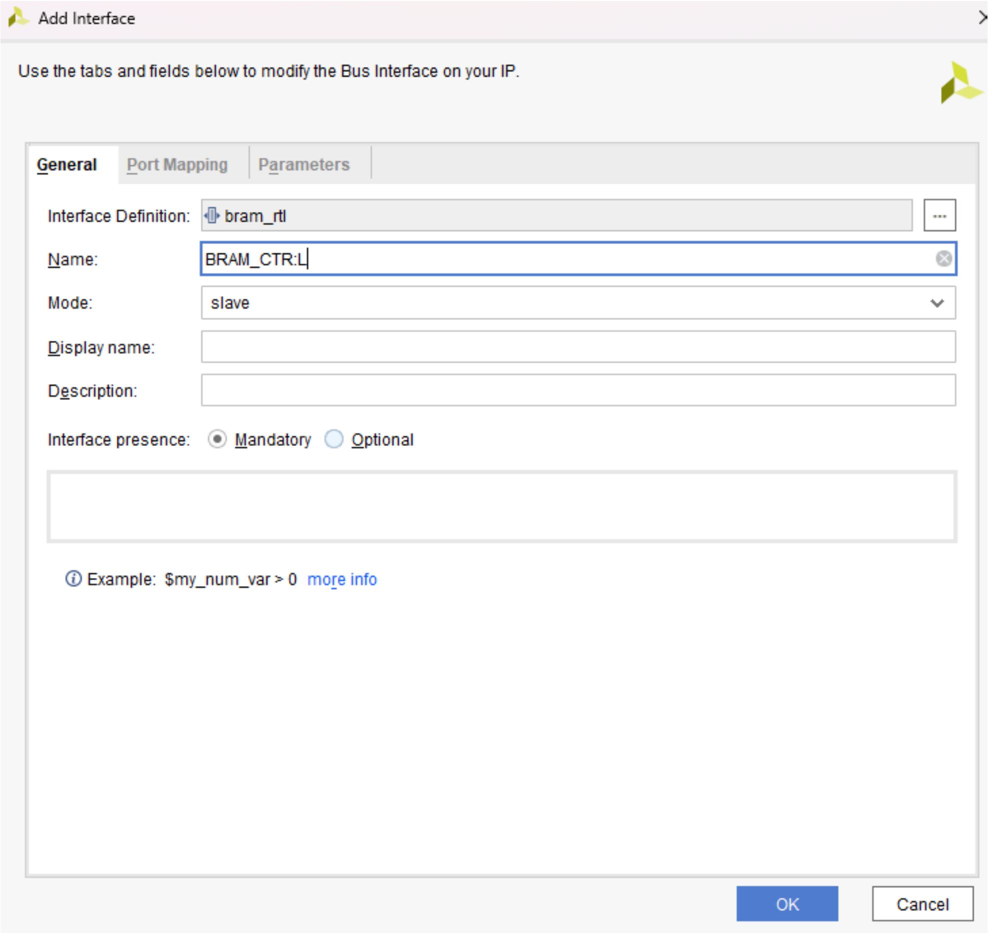
```
    input wire  [LOGSIZE:0] bram_addr,
    input wire  [WIDTH-1:0] bram_wrdata,
    output wire [WIDTH-1:0] bram_rddata,
    input wire  [NUM_COL-1:0]  bram_we,
```

e. Line 81:

```
  pipelined_processor #(.WIDTH(WIDTH), .SIZE(SIZE), .NUM_COL(NUM_COL), .COL_WIDTH(COL_WIDTH)) dut(.clk(s00_axi_aclk), .reset(ps_reset),
            .bram_din(bram_wrdata), .shared_bram_addr(bram_addr),
            .bram_wr_en(bram_we), .bram_dout(bram_rddata));
```

f. inside the v1_0_S00_AXI.v file:

```
output ps_reset,                    -> add at line 18
                assign ps_reset = slv_reg0;          -> add this to line 117
```



Add Interface

Use the tabs and fields below to modify the Bus Interface on your IP.

Interface Definition: bram_rtl

Name: BRAM_CTR:L

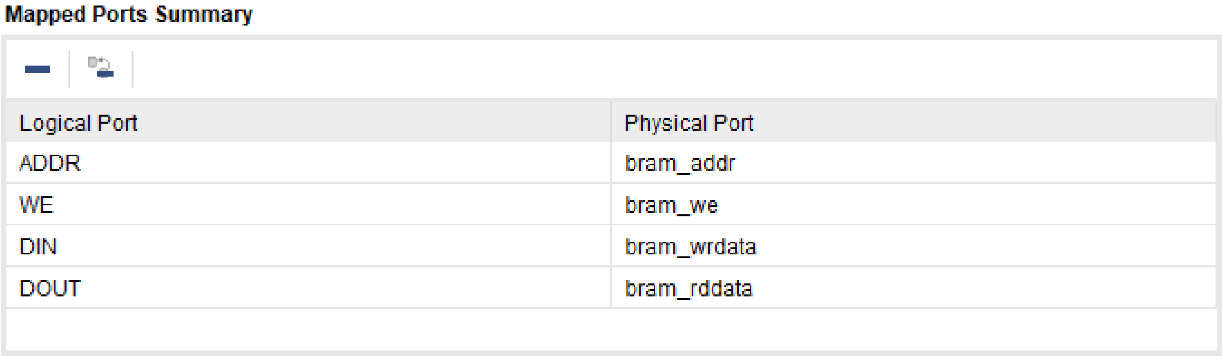
Mode: slave

Display name:

Description:

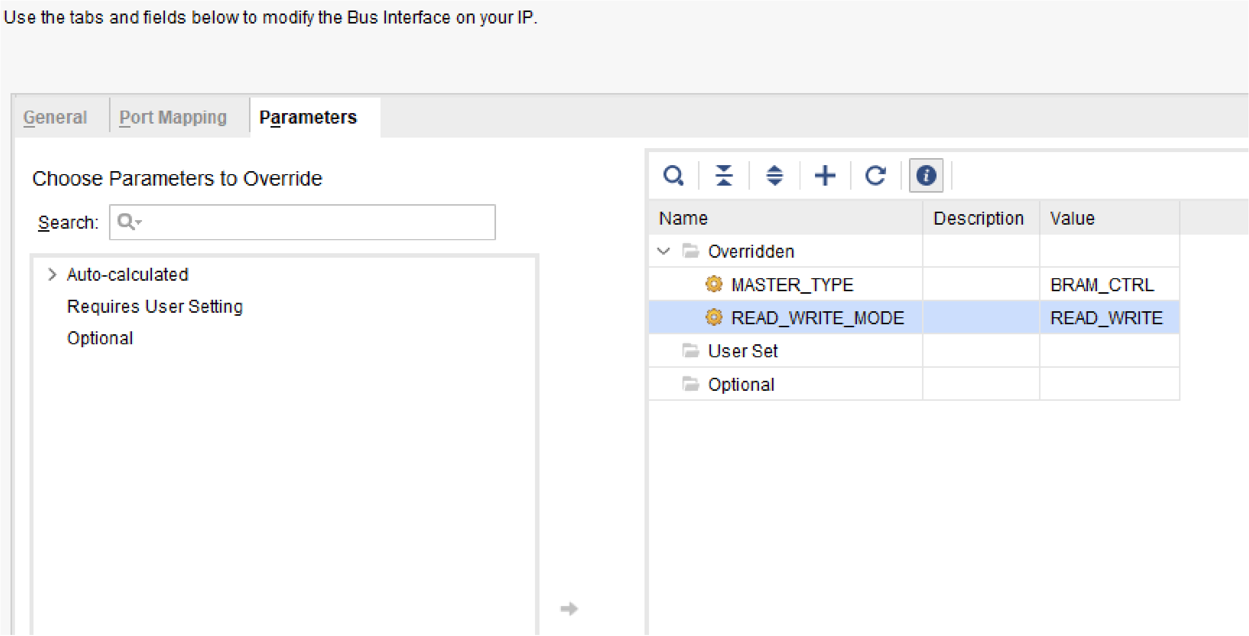
Interface presence: ● Mandatory ○ Optional

ⓘ Example: $my_num_var > 0 more info



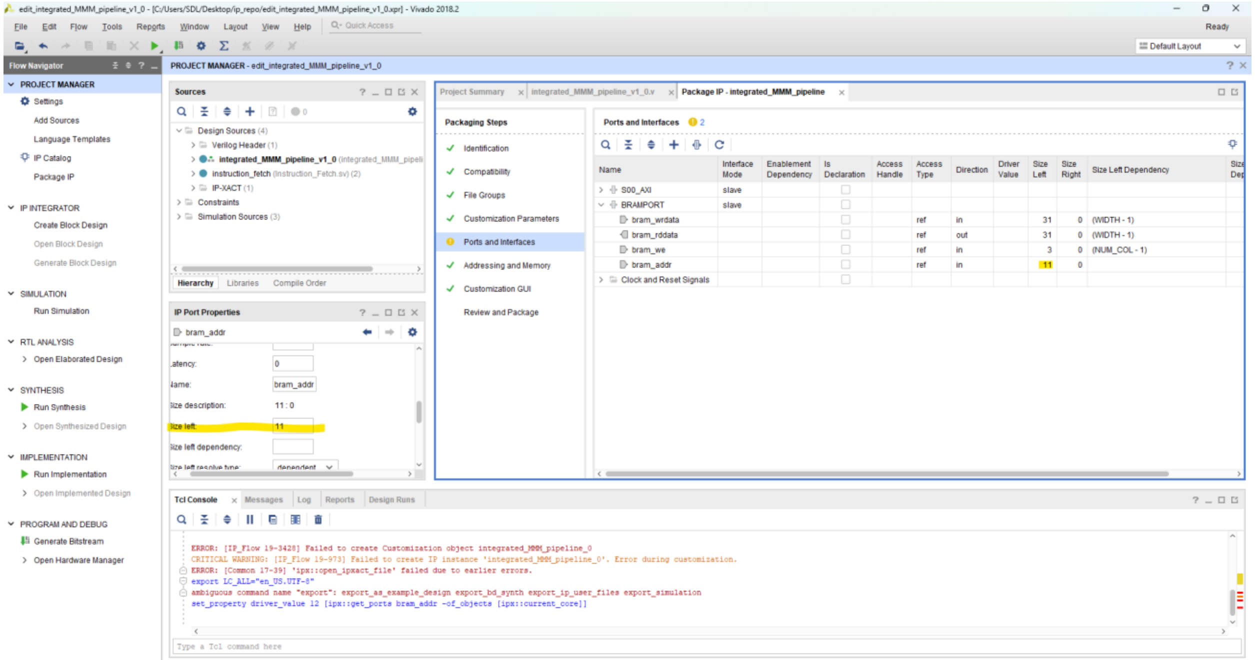
Mapped Ports Summary

Logical Port	Physical Port
ADDR	bram_addr
WE	bram_we
DIN	bram_wrdata
DOUT	bram_rddata



Use the tabs and fields below to modify the Bus Interface on your IP.

Name	Description	Value
Overridden		
MASTER_TYPE		BRAM_CTRL
READ_WRITE_MODE		READ_WRITE
User Set		
Optional		



Name	Interface Mode	Enablement Dependency	Is Declaration	Access Handle	Access Type	Direction	Driver Value	Size Left	Size Right	Size Left Dependency
S00_AXI	slave		□							
BRAMPORT	slave		□							
bram_wrdata			□		ref	in		31	0	(WIDTH - 1)
bram_rddata			□		ref	out		31	0	(WIDTH - 1)
bram_we			□		ref	in		3	0	(NUM_COL - 1)
bram_addr			□		ref	in		11	0	
Clock and Reset Signals			□							

IP Port Properties

bram_addr

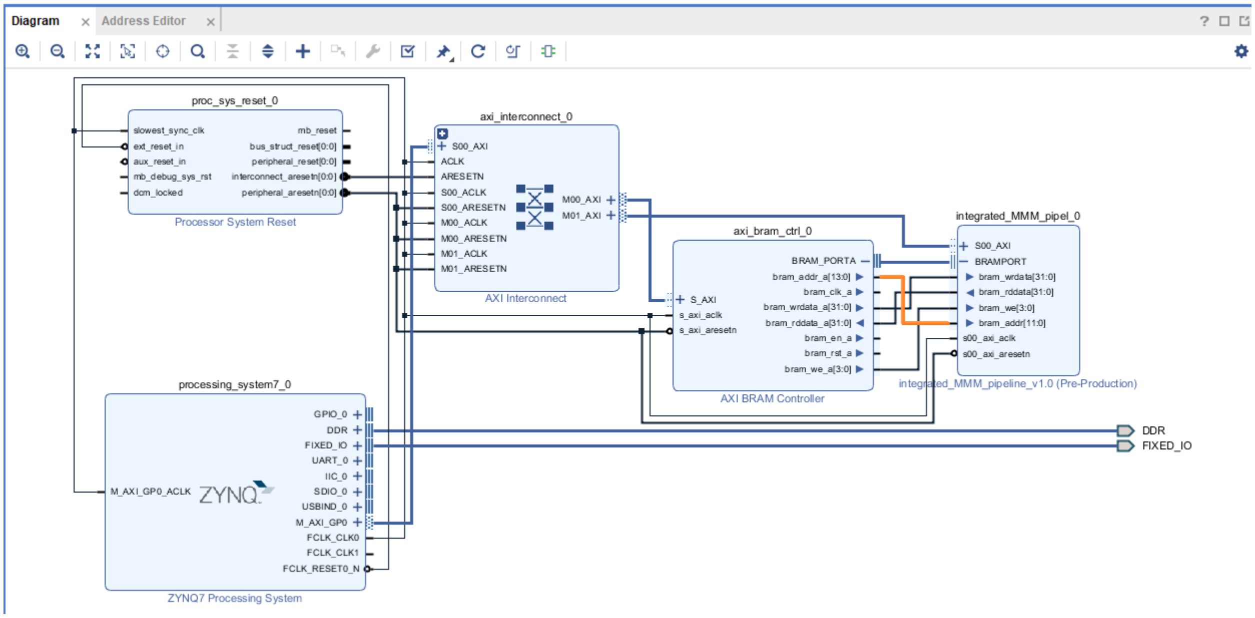
Latency:	0
Name:	bram_addr
Size description:	11 : 0
Size left:	11
Size left dependency:	
Size left resolve type:	dependent

```
ERROR: [IP_Flow 19-3428] Failed to create Customization object integrated_MMM_pipeline_0
CRITICAL WARNING: [IP_Flow 19-973] Failed to create IP instance 'integrated_MMM_pipeline_0'. Error during customization.
ERROR: [Common 17-39] 'ipx::open_ipxact_file' failed due to earlier errors.
export LC_ALL="en_US.UTF-8"
ambiguous command name "export": export_as_example_design export_bd_synth export_ip_user_files export_simulation
set_property driver_value 12 [ipx::get_ports bram_addr -of_objects [ipx::current_core]]
```

Edit bram_addr in ip port properties for size left and make it 11.

8. merge changes in the packaging steps tabs and package IP.

9. By now, Block diagram should look like this:



Validate design to make sure there is no errors.

Lastly, create HDL wrapper

SDK STUFF

File -> export hardware -> launch sdk

In the SDK:

Change linker in the application project under the source (src) folder, open lscript.ld (LINKER SCRIPT).

Under, section to memory region mapping, Change all region memory fields from ps7_DDR_0 to PS7_RAM_0

In case there are weird errors such as an assembly loop or when trying to running the program (AXI error):

1. Create new Application project called “FSBL”, choose zynq FSBL project as the template. Hit Finish.
2. Select the FSBL in project explorer on the left side, in top menu toolbar -> Xilinx-> create boot image-> on the bottom (boot image partitions) -> click add -> browse file path -> navigate to original application project (./Project.sdk/application_project_name/debug/application_name.elf) -> open elf and hit OK

3. Create image
4. With FSBL still highlighted, program and run. Then run application project and proceed as normal.

<u>For UART and SDK testing</u>

RUN THE PYTHON AND THEN THE C FILE IN THE SDK