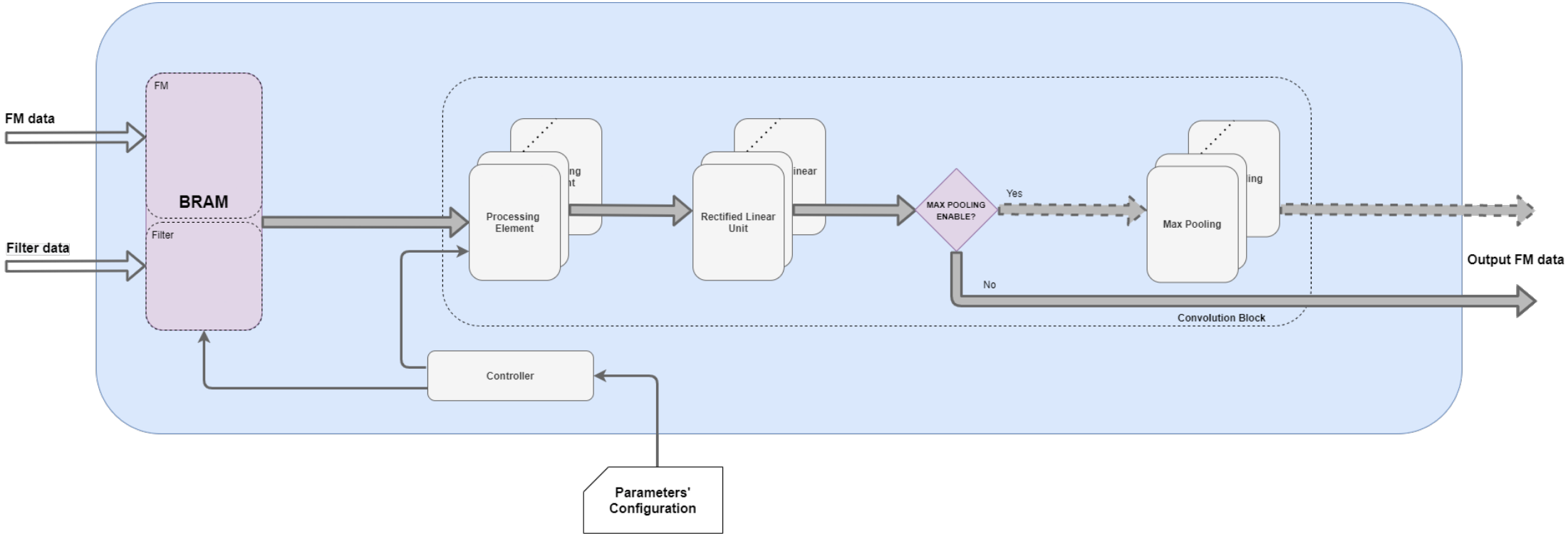
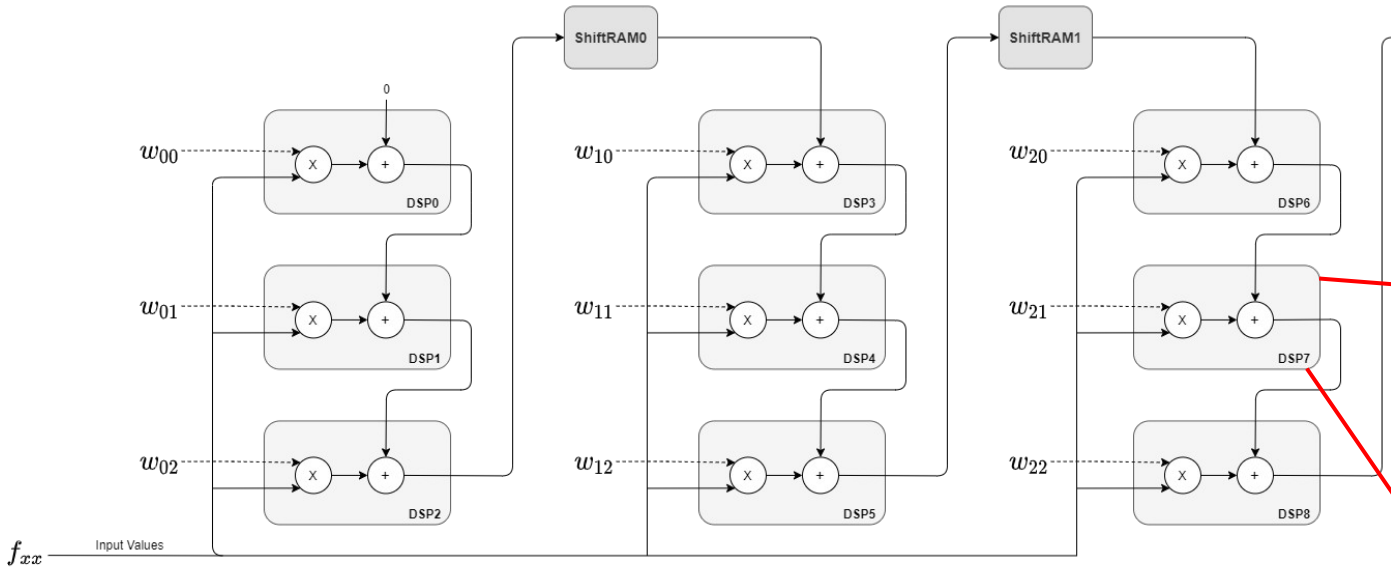
Convolution Module

Module architecture

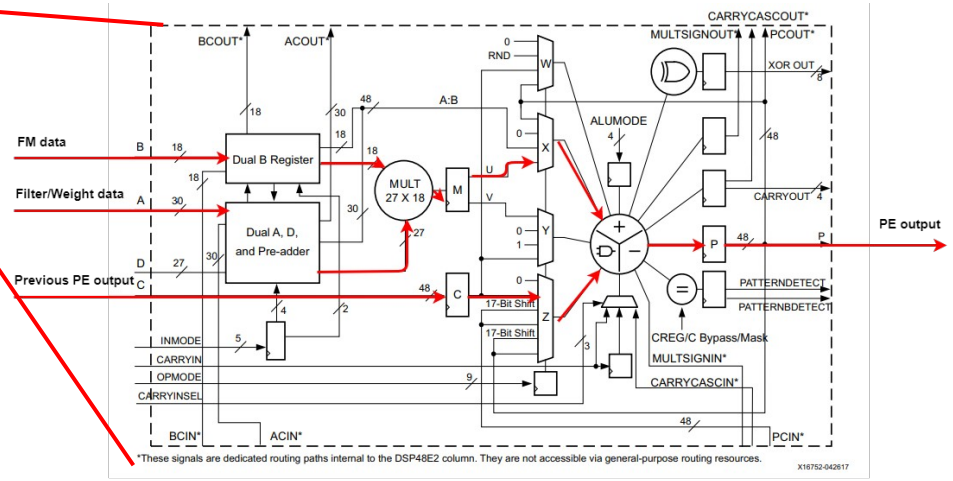


Processing Element

- 3x3 filter



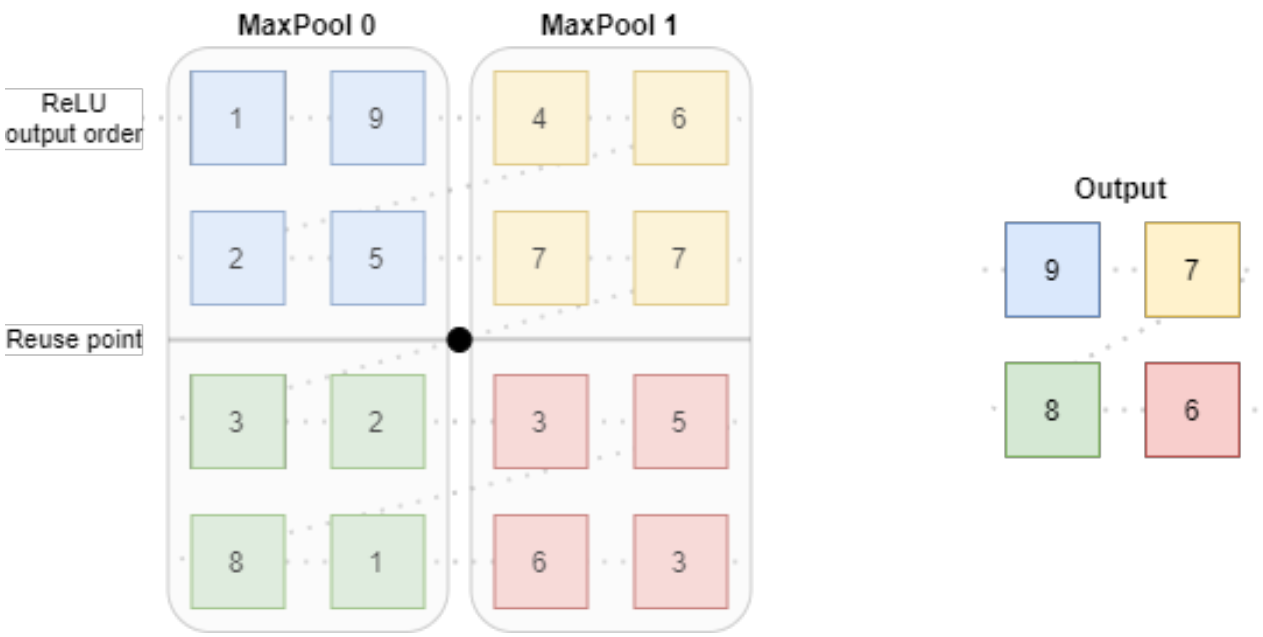
Processing Element configuration for a filter application of 3x3



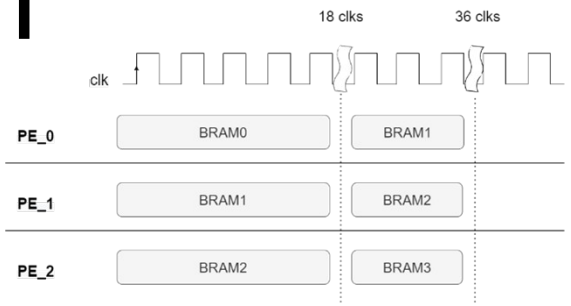
DSP Internal Block Diagram

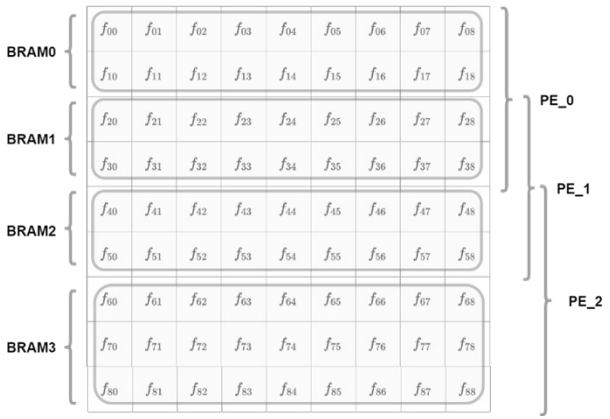
Max Pooling

- Number of MaxPooling blocks depend on the IFM size
- In this example, the output of the ReLU is a 4x4 FM, so 2 MaxP blocks are instantiated
- Both blocks are reused after the first output line have been processed (values 9 and 7)



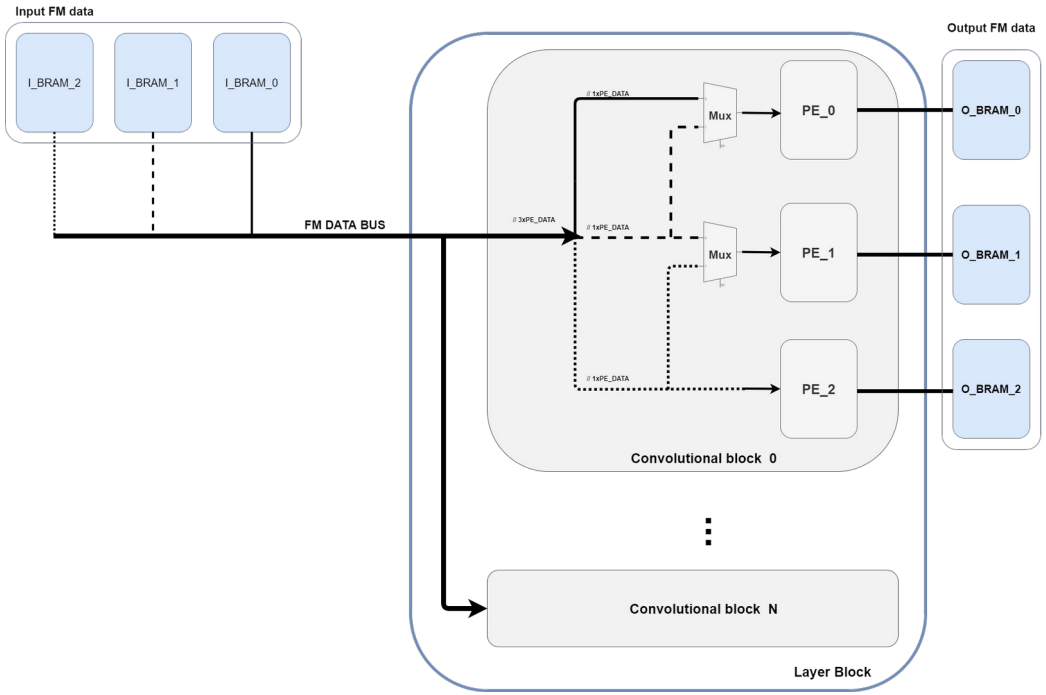
BRAM distribution





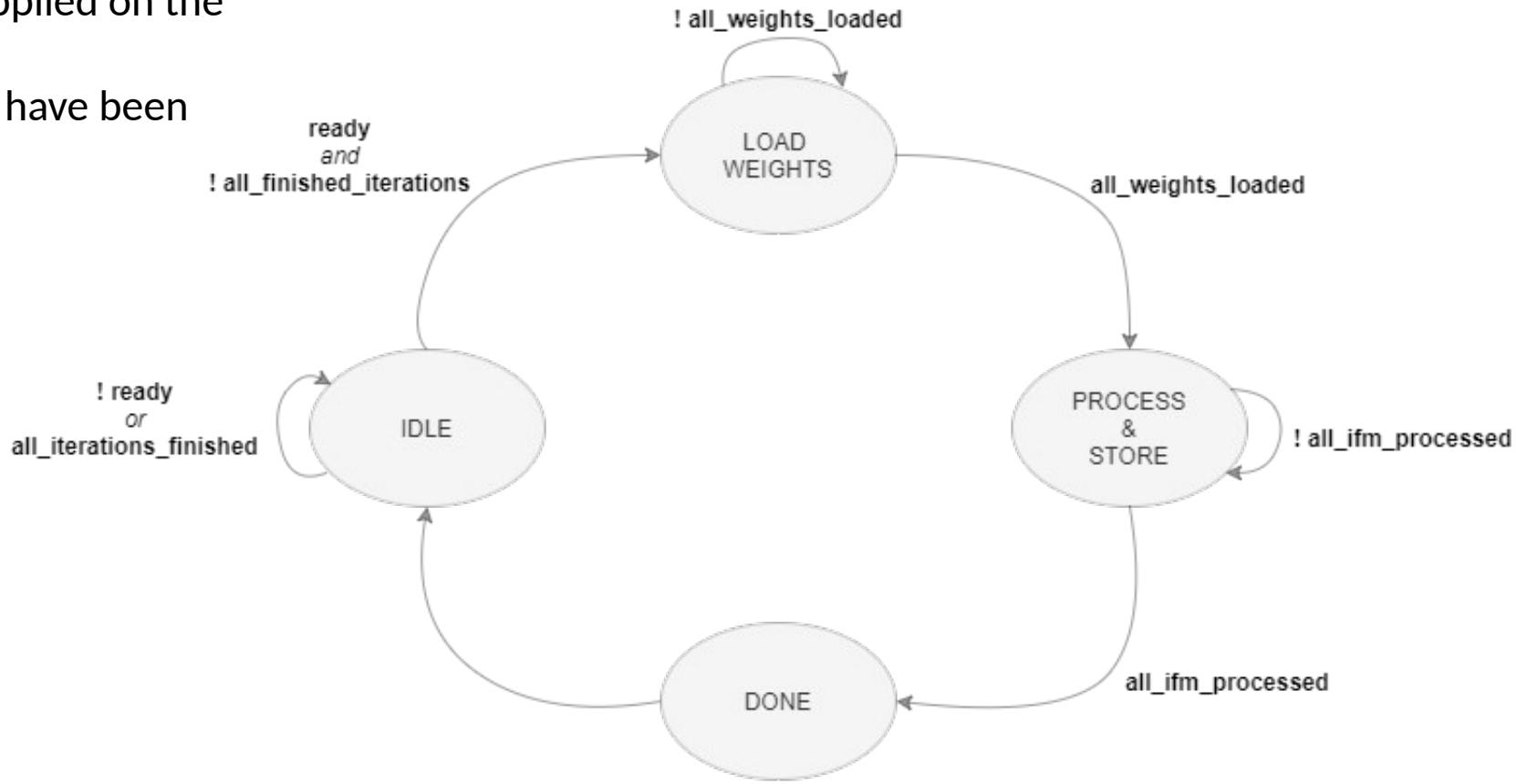
A FM of 9x9 with a filter application of 3x3, stride of 1.

- Figure shows maximum parallelism application
- Three PE blocks in parallel
- After 18 clks, each PE reads BRAM values from the following BRAM block
- Requires a MUX to control BRAM data flow



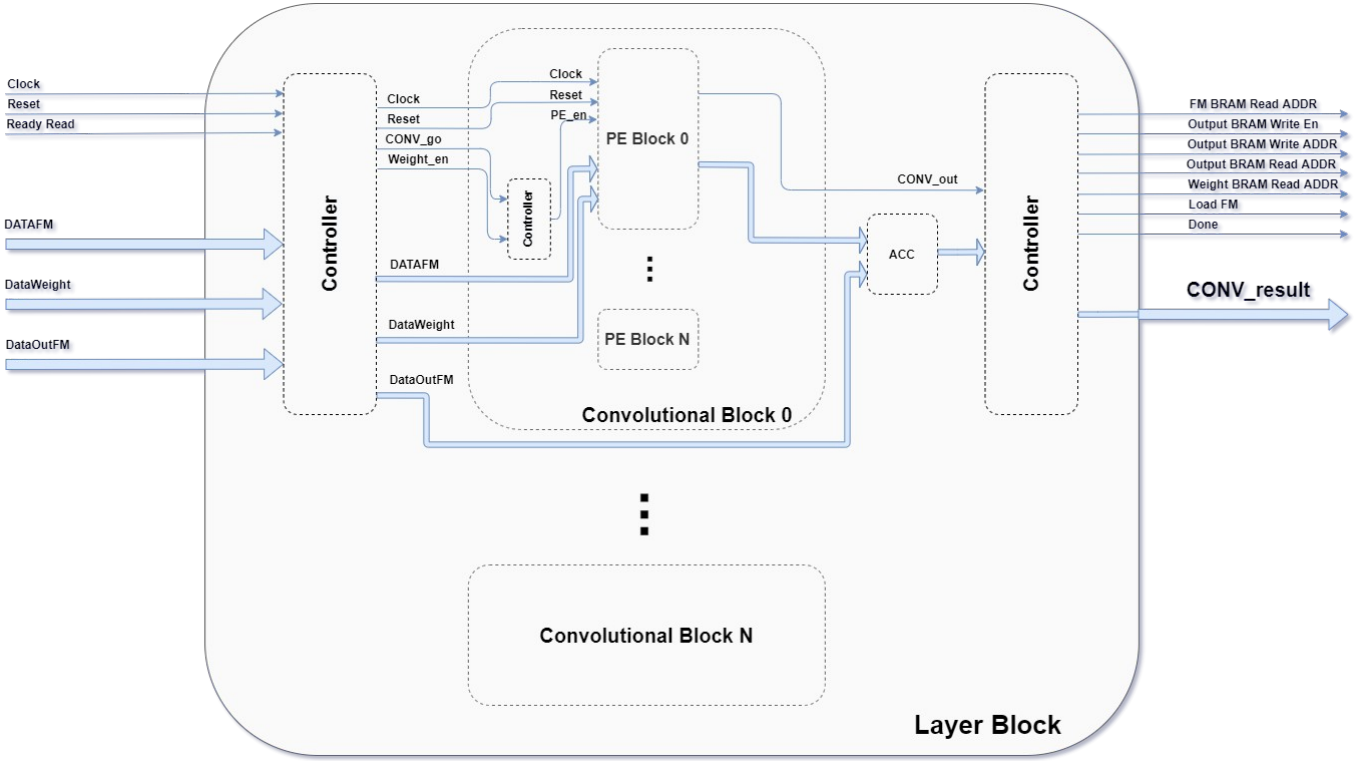
FSM

- According to the available resources, the processing required is divided into several iterations.
- Each iteration a new set of filters are applied on the IFM.
- The processing finished when all filters have been applied to all IFM



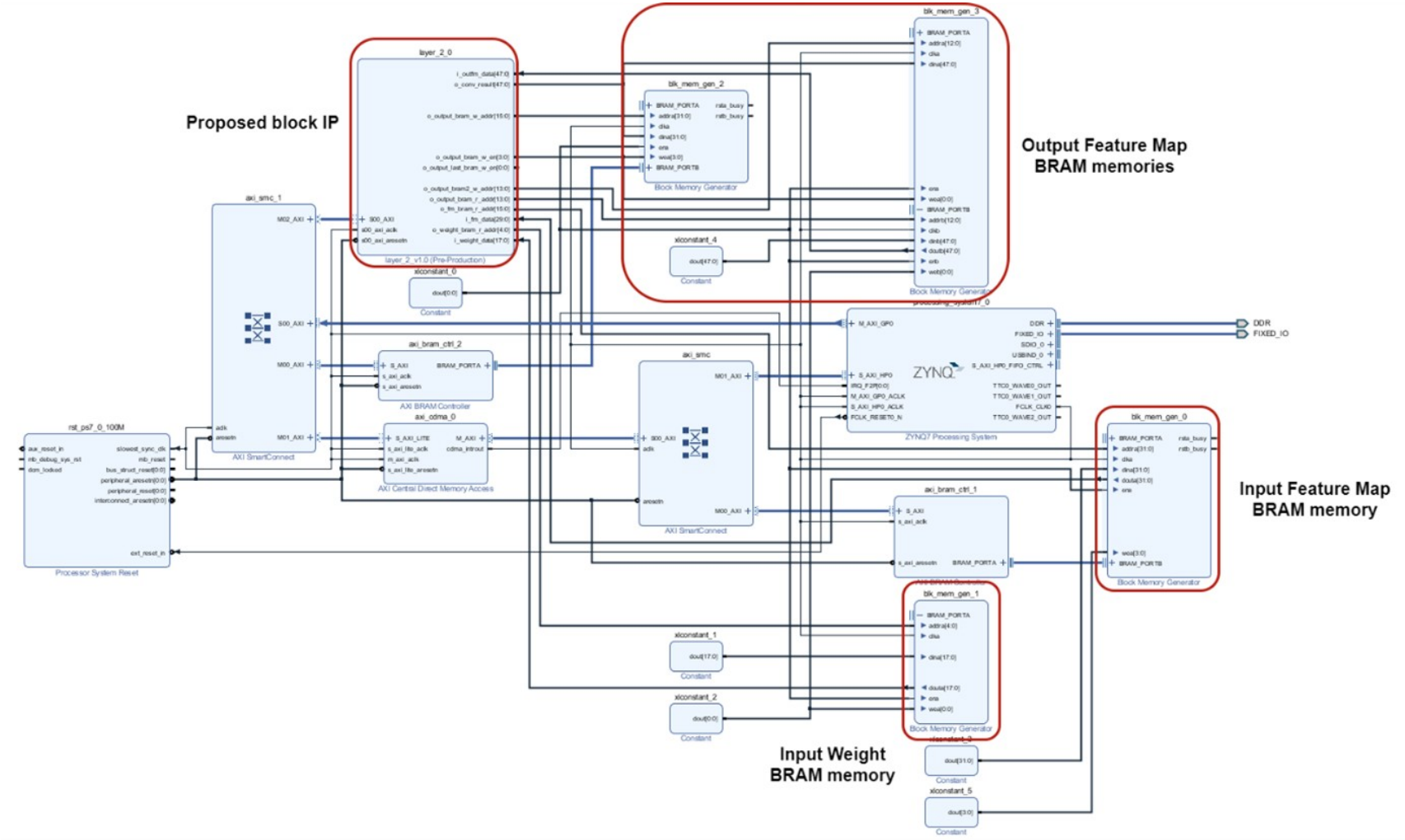
Layer block

- Inside the Layer block are possible to instantiate more than one Conv modules.
- Each Conv module is associated with a different filter.
- All inputs and outputs must be correctly assigned to BRAM blocks, otherwise data collision may occur.



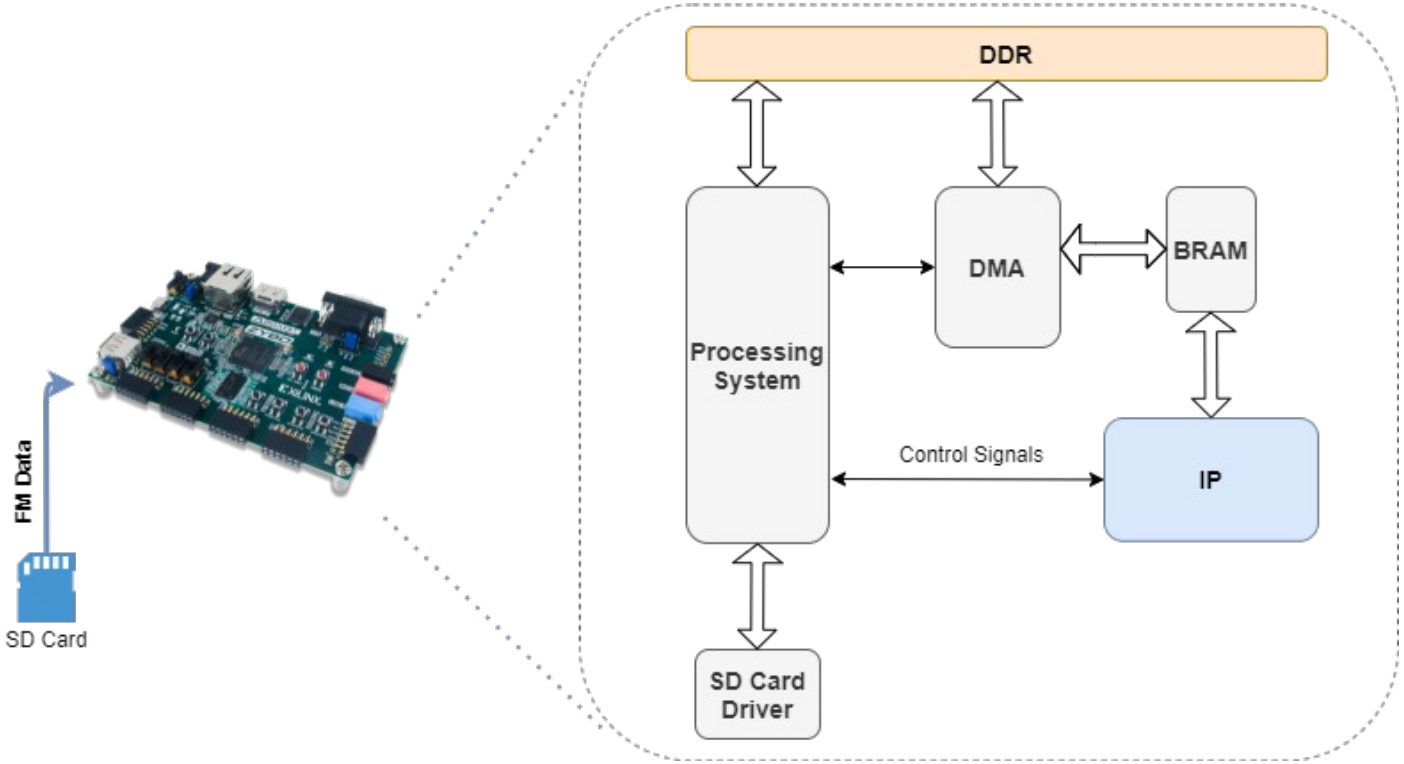
How to use - Block Design for validation

- 3 Block Rams (Filter, IFM, and OFM)
- Developed IP connected to all BRAMS
- OFM BRAMS with two ports (convolution with each IFM channel is added)



Test Setup

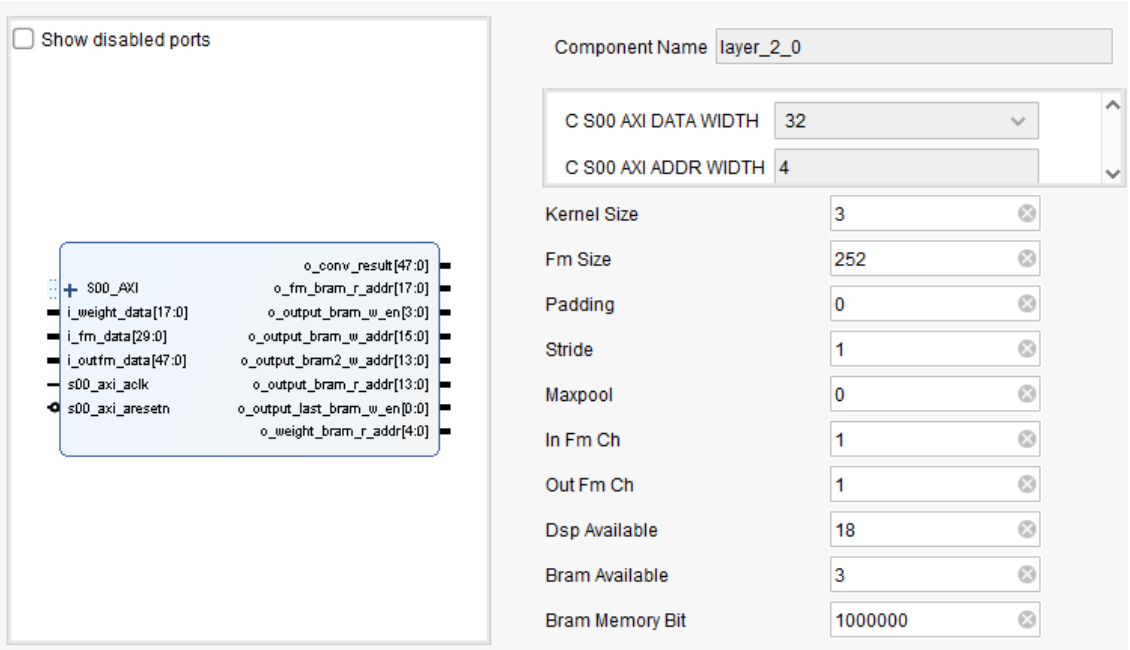
- The input image (FM) is stored in an SD card.
- The image is sent to DDR through a DMA module.
- Then when a signal is sent from the PS to our IP, using an AXI interconnect, the state machine collects data from BRAM block which was previously stored using the DMA module.
- After finishing the convolution process, the output FM is sent to SD card.



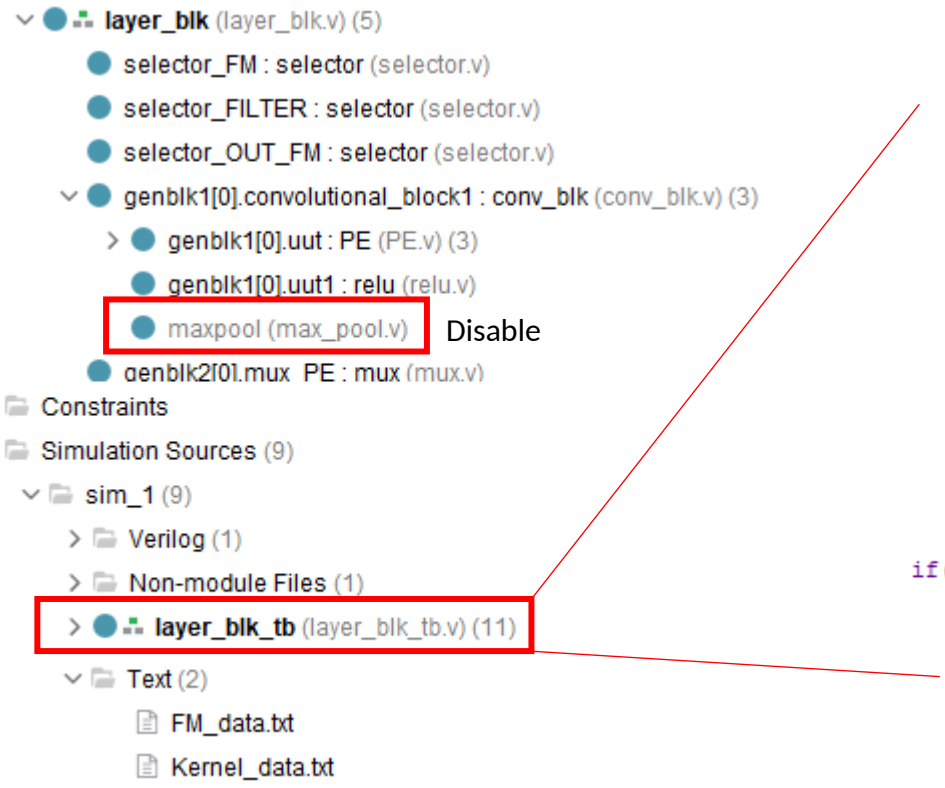
Layer block – IP interface

- The parameters Kernel Size, Fm Size, Padding, Stride, Maxpool, In Fm Ch and Out Fm Ch are directly associated with Convolution Neural Network architecture background.
- The parameters DSP available, BRAM available and memory bit are related with hardware resources.
- Different paremeters values allows higher level of parallelism consuming more resources and increasing computation.

- Maxpool parameter is enabled/disabled with 1/0.
- In Fm Ch refers to the number of input channels for a given convolution.
- Out Fm Ch refers to the number of different filters applying during a convolution.
- “Bram Available”: Blocks of Memory available
- “Bram Memory Bit”: Capacity (bits) of each block memory



Vivado Simulation



In test bench file, change parameters values

```
module layer_blk_tb #(
 parameter KERNEL_SIZE     = 3,
 parameter FM_SIZE         = 252,
 parameter PADDING         = 0,
 parameter STRIDE          = 1,
 parameter MAXPOOL         = 0,
 parameter DSP_AVAILABLE   = 36,
 parameter IN_FM_CH        = 1,
 parameter OUT_FM_CH       = 1,
 parameter BRAM_AVAILABLE  = 12,
 parameter BRAM_MEMORY_BIT  = 1000000,
```

```
if(((output_bram_r_addr < (BRAM_SIZE*OUT_SIZE)+1) && out_file == 0) || (((output_bram_r_addr
   if(output_bram_r_addr > 0) begin
       $fwrite(out_data,"%0d\n", output_bram_o_data[out_file*48 + 48*PE_TO_USE*0 +: 48]);
```

- Each $fwrite saves the convolution output from one filter application.
- The output file is located on .../.sim/sim_x/.. dir from vivado project.

Vivado Simulation

- layer_blk (layer_blk.v) (5)
  - selector_FM : selector (selector.v)
  - selector_FILTER : selector (selector.v)
  - selector_OUT_FM : selector (selector.v)
  - genblk1[0].convolutional_block1 : conv_blk (conv_blk.v) (3)
    - genblk1[0].uut : PE (PE.v) (3)
    - genblk1[0].uut1 : relu (relu.v)
    - maxpool (max_pool.v)
  - genblk2[0].mux_PE : mux (mux.v)
- Constraints
- Simulation Sources (9)
  - sim_1 (9)
    - Verilog (1)
    - Non-module Files (1)
    - layer_blk_tb (layer_blk_tb.v) (11)
    - Text (2)
      - FM_data.txt
      - Kernel_data.txt

FM_data.txt | Kernel_data.txt

C:/Tese/Vivado/DSP_cascade/DSP_casca

```
1  a9
2  a7
3  a6
4  a8
5  a9
6  a7
7  a6
```

FM data in hex format

FM_data.txt | Kernel_data.txt

C:/Tese/Vivado/DSP_cascade/DSP_casca

```
1  0
2  0
3  0
4  0
5  1
6  0
7  0
8  0
9  0
10 3FFFF
11 0
12 3FFFF
13 5
14 3FFFF
15 0
16 3FFFF
17 0
18 0
```

Identity

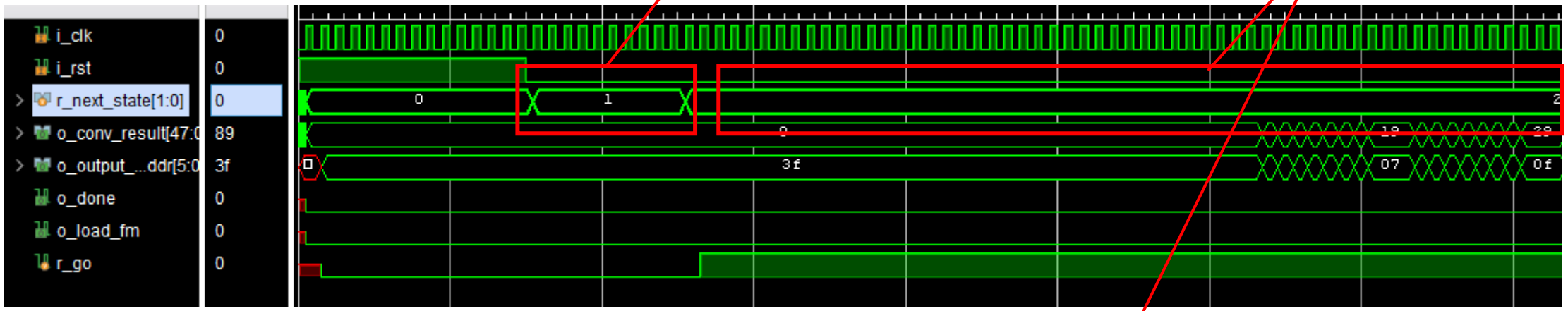
Sharpen

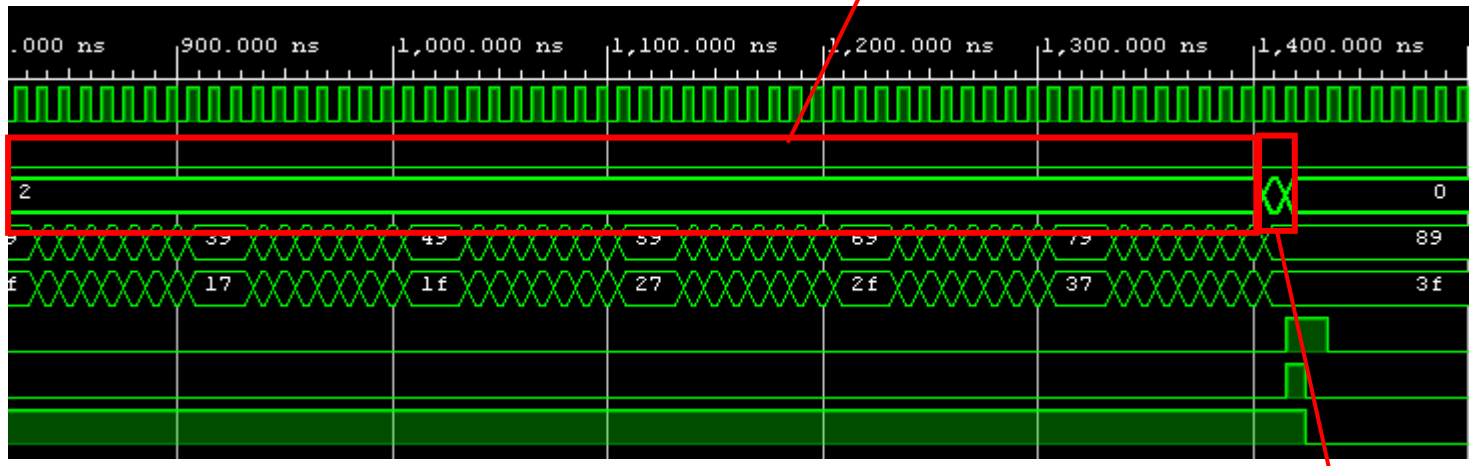
Two different filters, first 9 values for identity filter and following for sharpen filter.

Vivado Simulation

Load Weights

Process and store



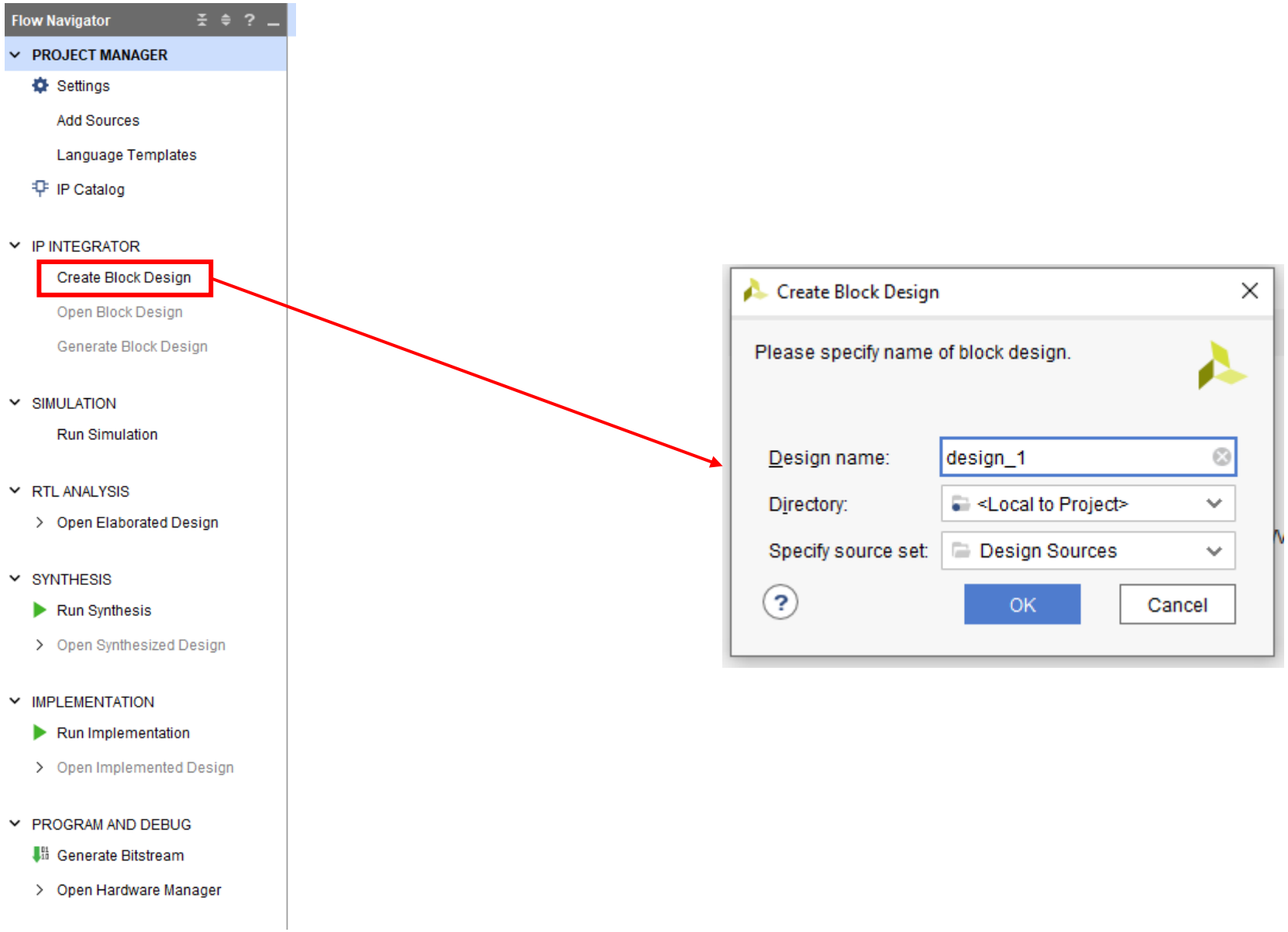


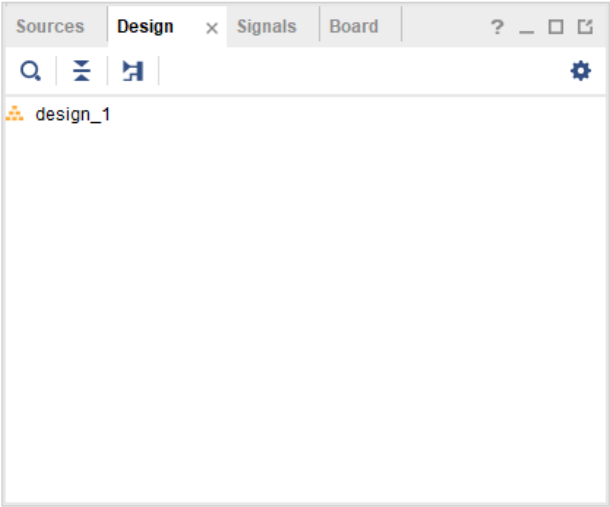
FM of 10x10, filter 3x3 with stride 1

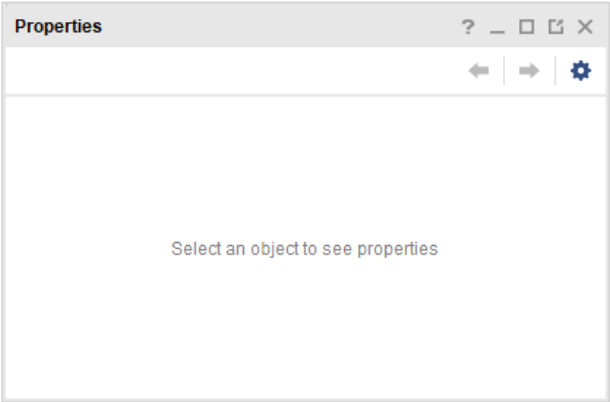
R_go enables convolution process

O_done signals the end of processing

Done


Flow Navigator
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Open Elaborated Design
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
PROGRAM AND DEBUG
Generate Bitstream
Open Hardware Manager
Create Block Design
Please specify name of block design.
Design name:
design_1
Directory:
<Local to Project>
Specify source set:
Design Sources
OK
Cancel


Sources
Design
Signals
Board
design_1


Properties
Select an object to see properties

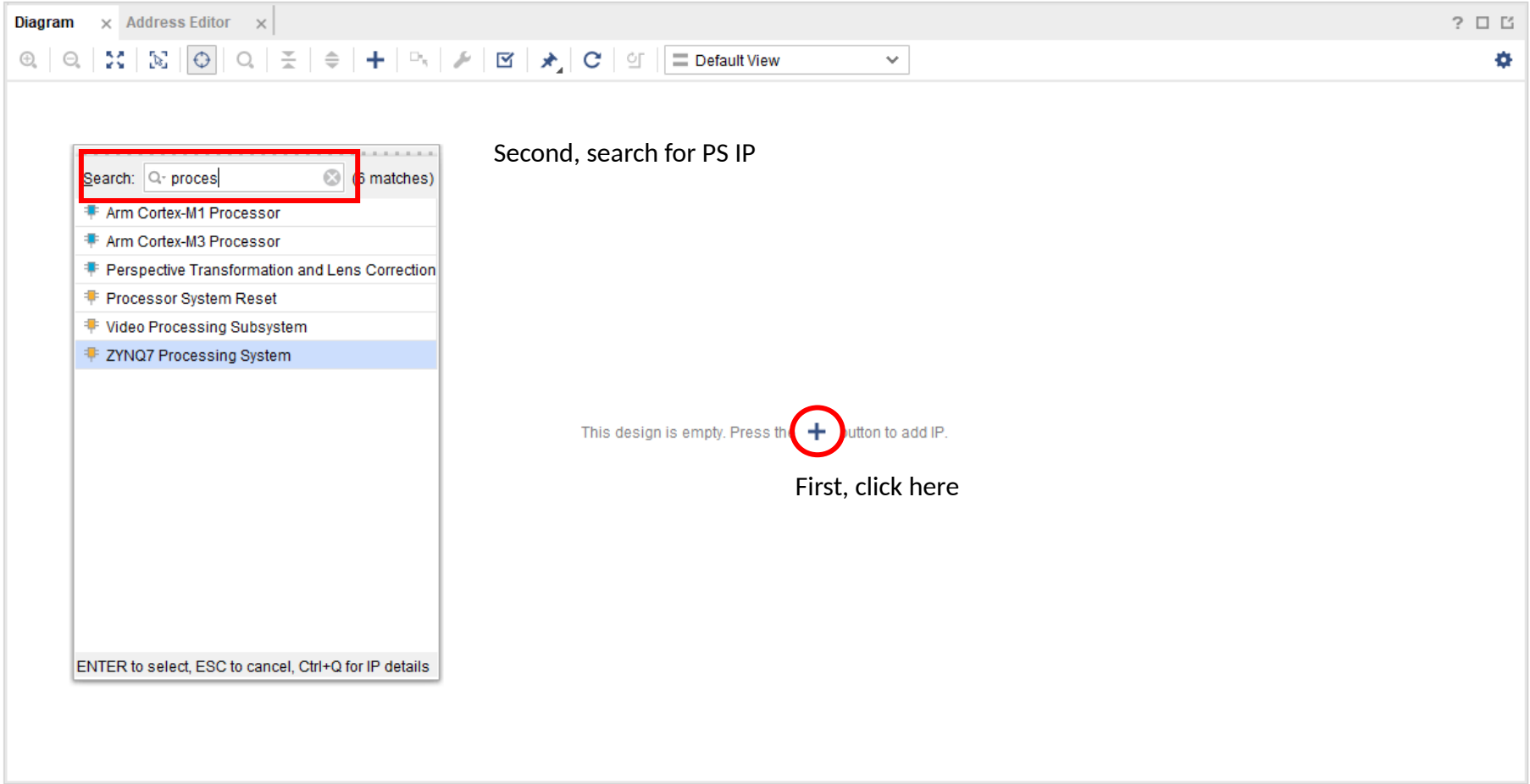
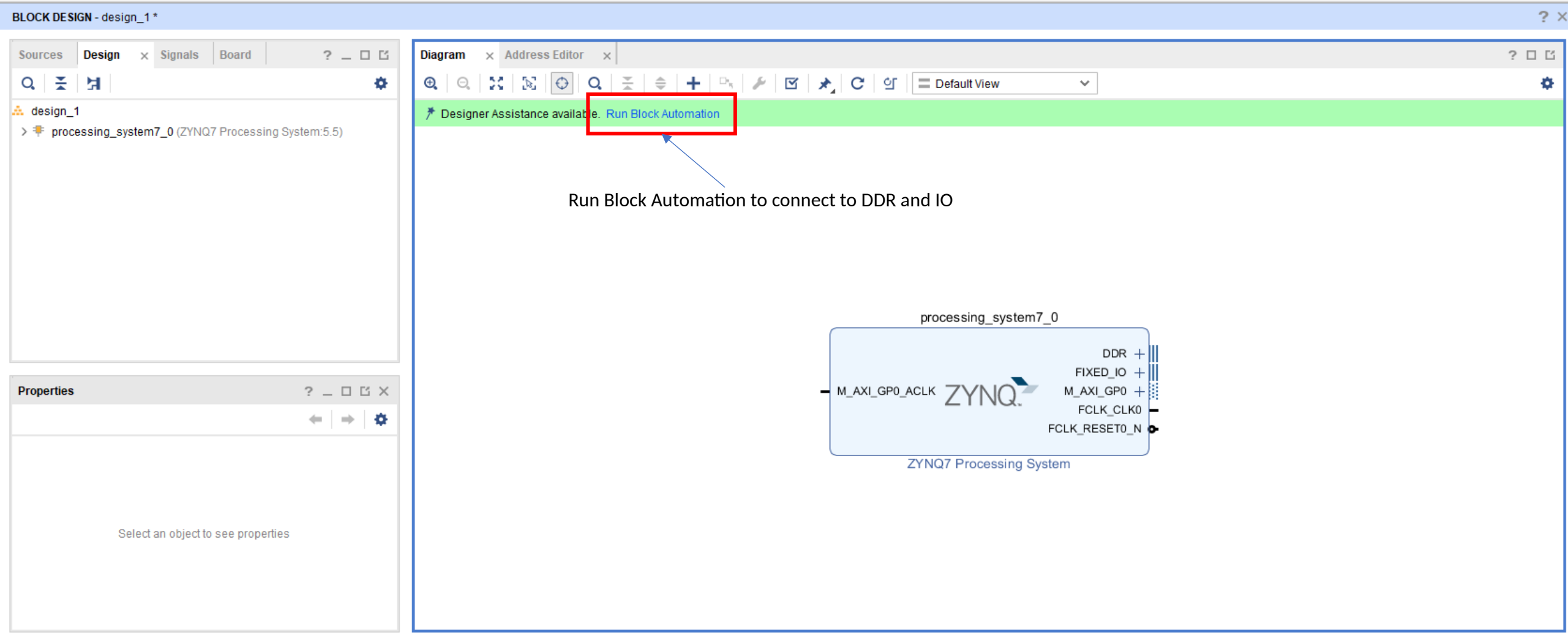


Diagram
Address Editor
Default View
Search: proces
(6 matches)
Arm Cortex-M1 Processor
Arm Cortex-M3 Processor
Perspective Transformation and Lens Correction
Processor System Reset
Video Processing Subsystem
ZYNQ7 Processing System
ENTER to select, ESC to cancel, Ctrl+Q for IP details
Second, search for PS IP
This design is empty. Press the button to add IP.
First, click here


BLOCK DESIGN - design_1 *
Sources
Design
Signals
Board
design_1
processing_system7_0 (ZYNQ7 Processing System:5.5)
Properties
Select an object to see properties
Diagram
Address Editor
Default View
Designer Assistance available. Run Block Automation
Run Block Automation to connect to DDR and IO
processing_system7_0
M_AXI_GP0_ACLK
ZYNQ
DDR
FIXED_IO
M_AXI_GP0
FCLK_CLK0
FCLK_RESET0_N
ZYNQ7 Processing System

BLOCK DESIGN - design_1 *

Sources | Design | Signals | Board

design_1
- External Interfaces
- Interface Connections
- processing_system7_0 (ZYNQ7 Processing System:5.5)

Block Properties

processing_system7_0

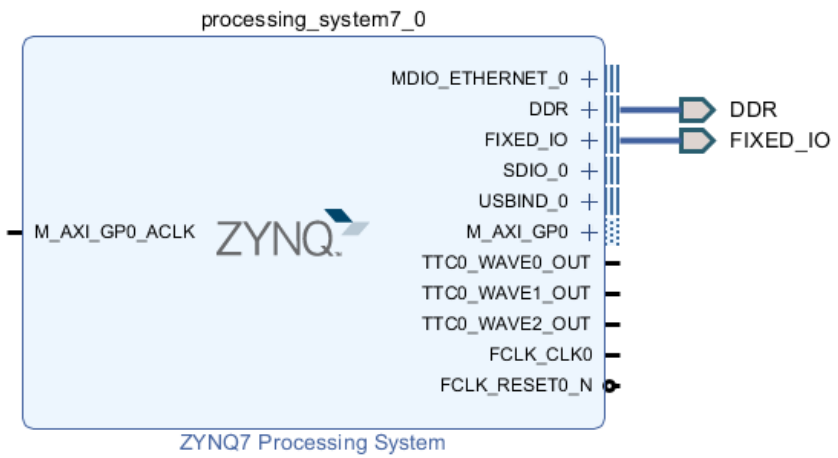
Name: processing_system7_0

Parent name: design_1

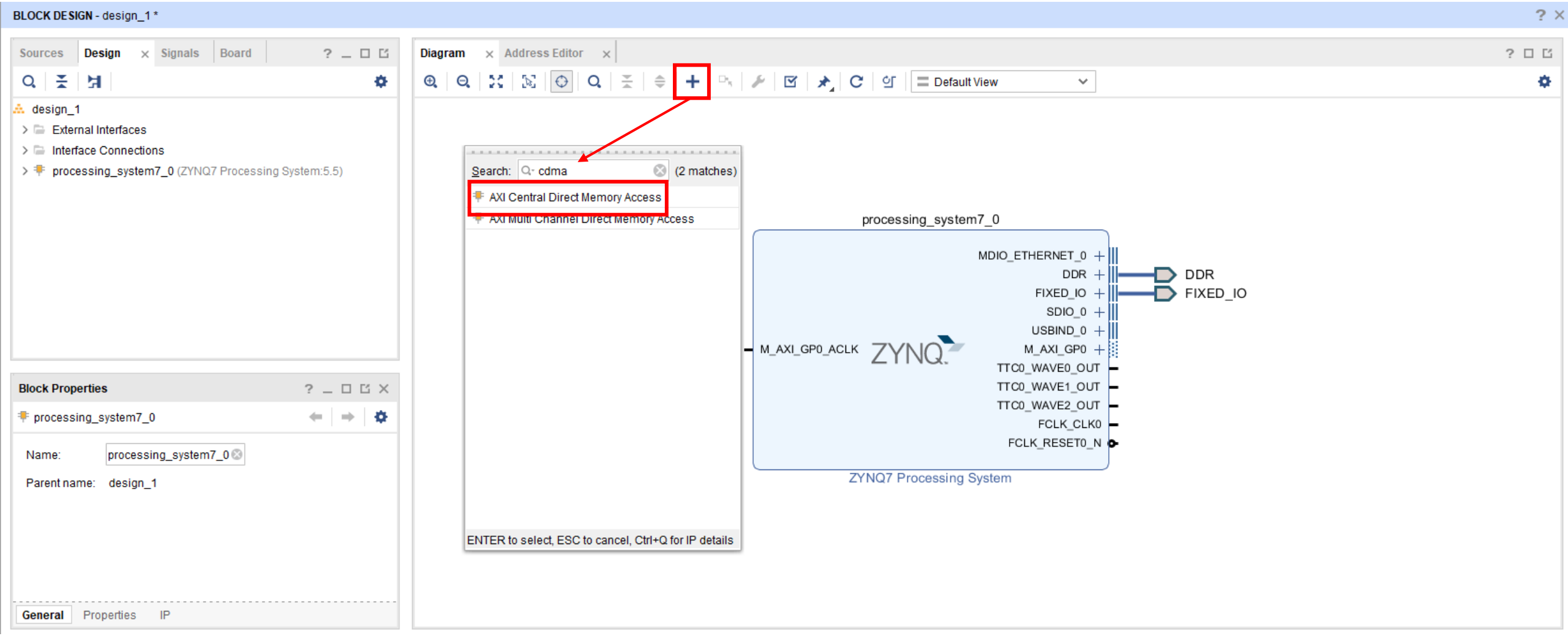
General | Properties | IP

Diagram | Address Editor

Default View



Add CDMA Module to read from input BRAM



BLOCK DESIGN - design_1 *

Sources | Design | Signals | Board

design_1
- External Interfaces
- Interface Connections
- axi_cdma_0 (AXI Central Direct Memory Access:4.1)
- processing_system7_0 (ZYNQ7 Processing System:5.5)

Block Properties

processing_system7_0

Name: processing_system7_0

Parent name: design_1

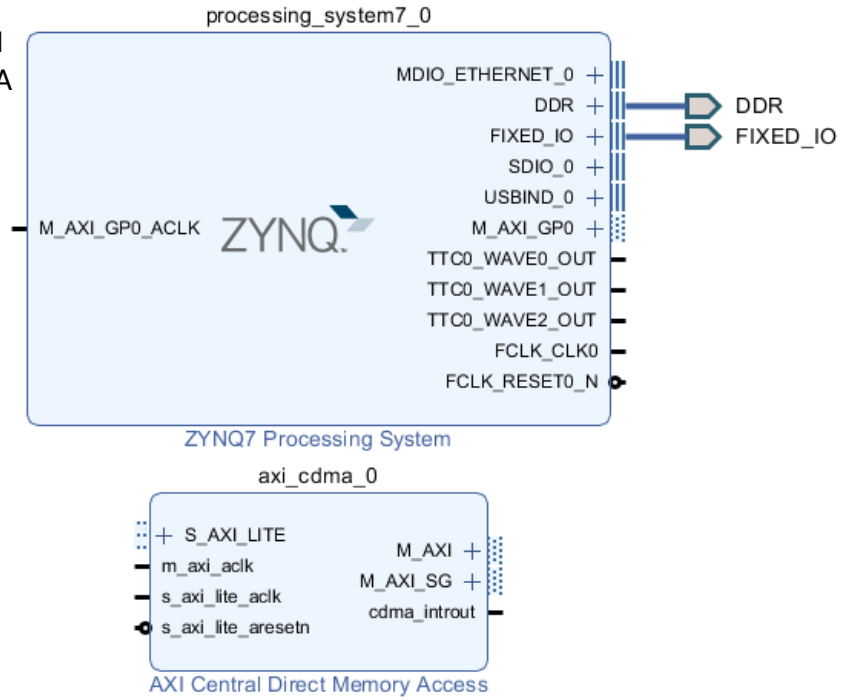
General | Properties | IP

Diagram | Address Editor

Default View

Designer Assistance available. Run Connection Automation

Run Connection Automation to Instantiate System Reset and an AXI Interconnect between PS and CDMA



BLOCK DESIGN - design_1 *

Sources | Design | Signals | Board

design_1
- External Interfaces
- Interface Connections
  - processing_system7_0_DDR
  - processing_system7_0_FIXED_IO
  - processing_system7_0_M_AXI_GP0
  - ps7_0_axi_periph_M00_AXI
- Nets
- axi_cdma_0 (AXI Central Direct Memory Access:4.1)
  - M_AXI
  - M_AXI_SG
  - S_AXI_LITE
  - cdma_introut
  - m_axi_aclk
  - s_axi_lite_aclk

Block Interface Properties

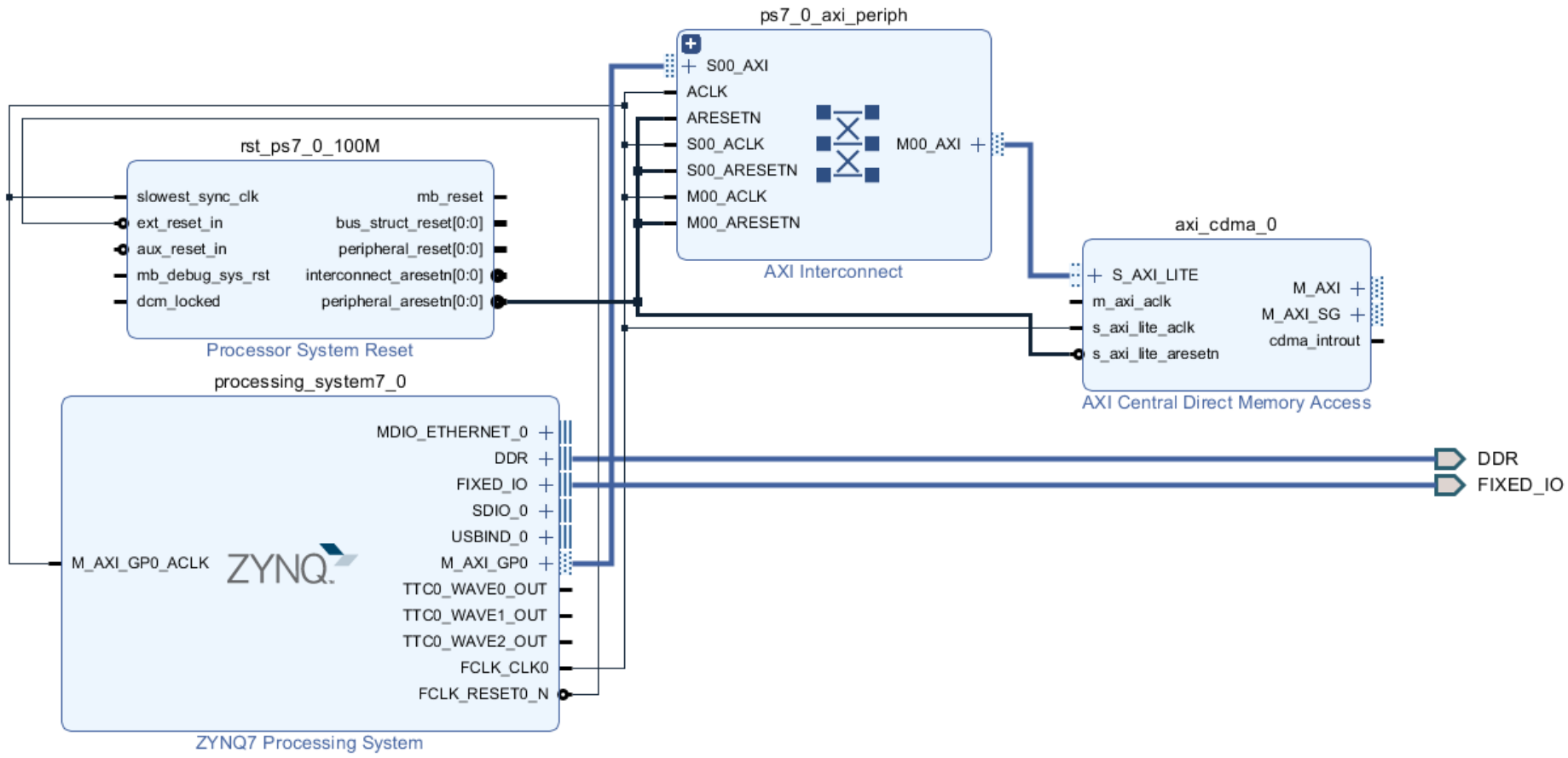
S_AXI_LITE

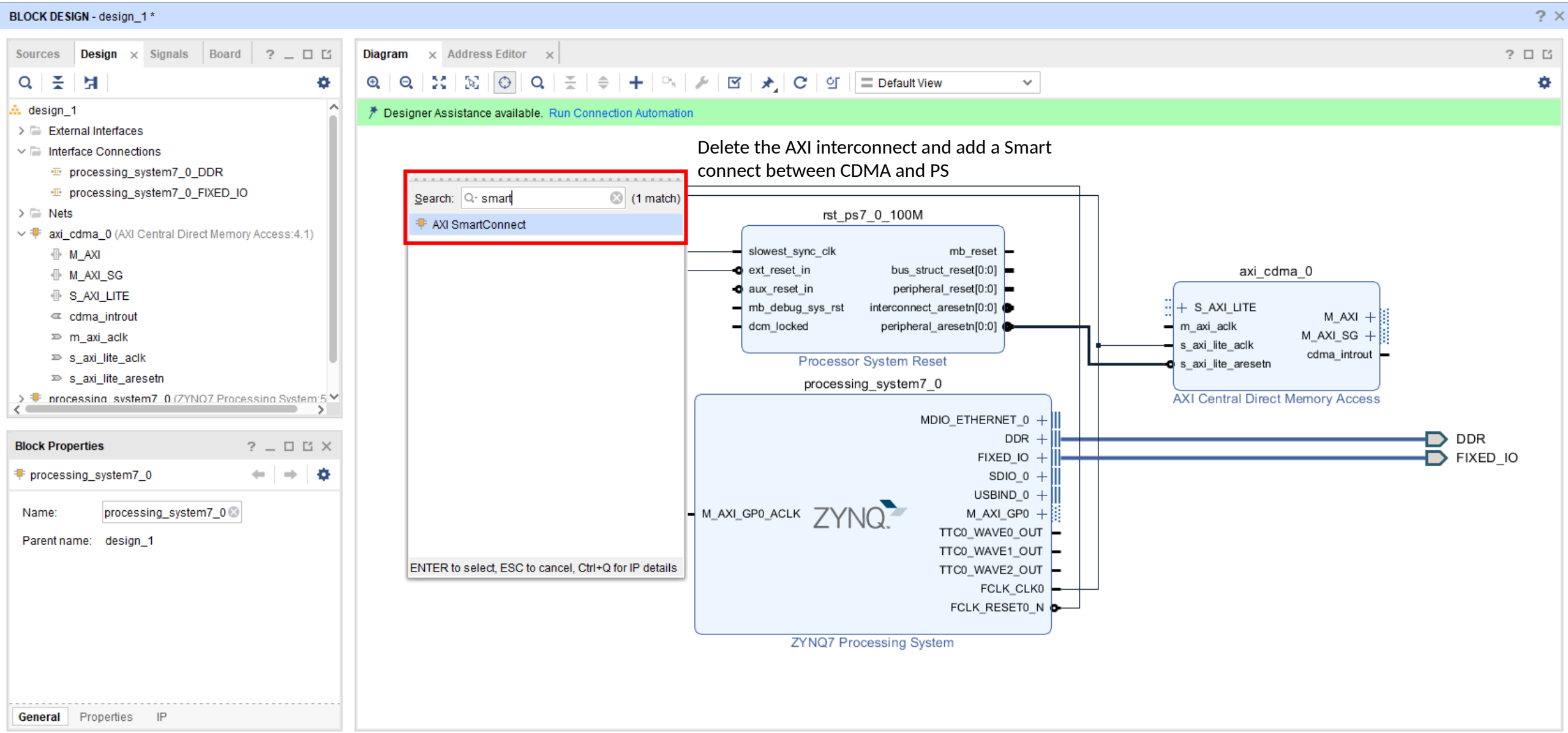
Name:	S_AXI_LITE
Mode:	SLAVE
Connection	ps7_0_axi_periph_M00_AXI
Associated clock:	s_axi_lite_aclk

General | Properties

Diagram | Address Editor

Default View




BLOCK DESIGN - design_1 *
Sources
Design
Signals
Board
design_1
External Interfaces
Interface Connections
processing_system7_0_DDR
processing_system7_0_FIXED_IO
Nets
axi_cdma_0 (AXI Central Direct Memory Access:4.1)
M_AXI
M_AXI_SG
S_AXI_LITE
cdma_introut
m_axi_aclk
s_axi_lite_aclk
s_axi_lite_aresetn
processing_system7_0 (ZYNQ7 Processing System:5
Block Properties
processing_system7_0
Name: processing_system7_0
Parent name: design_1
General
Properties
IP
Diagram
Address Editor
Default View
Designer Assistance available. Run Connection Automation
Delete the AXI interconnect and add a Smart connect between CDMA and PS
Search: smart
(1 match)
AXI SmartConnect
ENTER to select, ESC to cancel, Ctrl+Q for IP details
rst_ps7_0_100M
slowest_sync_clk
ext_reset_in
aux_reset_in
mb_debug_sys_rst
dcm_locked
mb_reset
bus_struct_reset[0:0]
peripheral_reset[0:0]
interconnect_aresetn[0:0]
peripheral_aresetn[0:0]
Processor System Reset
axi_cdma_0
S_AXI_LITE
m_axi_aclk
s_axi_lite_aclk
s_axi_lite_aresetn
M_AXI
M_AXI_SG
cdma_introut
AXI Central Direct Memory Access
processing_system7_0
M_AXI_GP0_ACLK
ZYNQ
MDIO_ETHERNET_0
DDR
FIXED_IO
SDIO_0
USBIND_0
M_AXI_GP0
TTC0_WAVE0_OUT
TTC0_WAVE1_OUT
TTC0_WAVE2_OUT
FCLK_CLK0
FCLK_RESET0_N
ZYNQ7 Processing System
DDR
FIXED_IO

BLOCK DESIGN - design_1 *

Sources | Design | Signals | Board

design_1
- External Interfaces
- Interface Connections
- Nets
- axi_cdma_0 (AXI Central Direct Memory Access:4.1)
- processing_system7_0 (ZYNQ7 Processing System:5.5)
- rst_ps7_0_100M (Processor System Reset:5.0)
- smartconnect_0 (AXI SmartConnect:1.0)

Block Properties

smartconnect_0

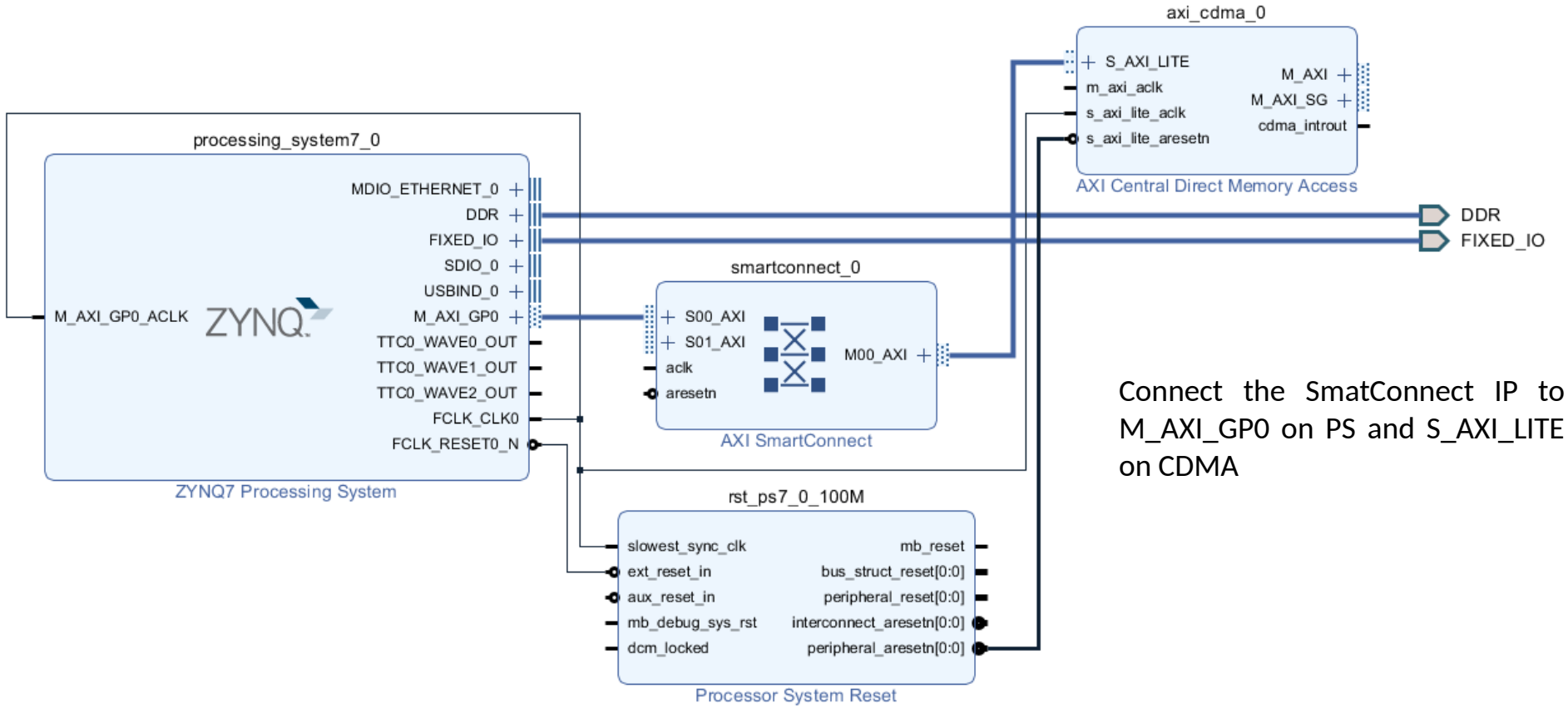
Name: smartconnect_0

Parent name: design_1

General | Properties | IP

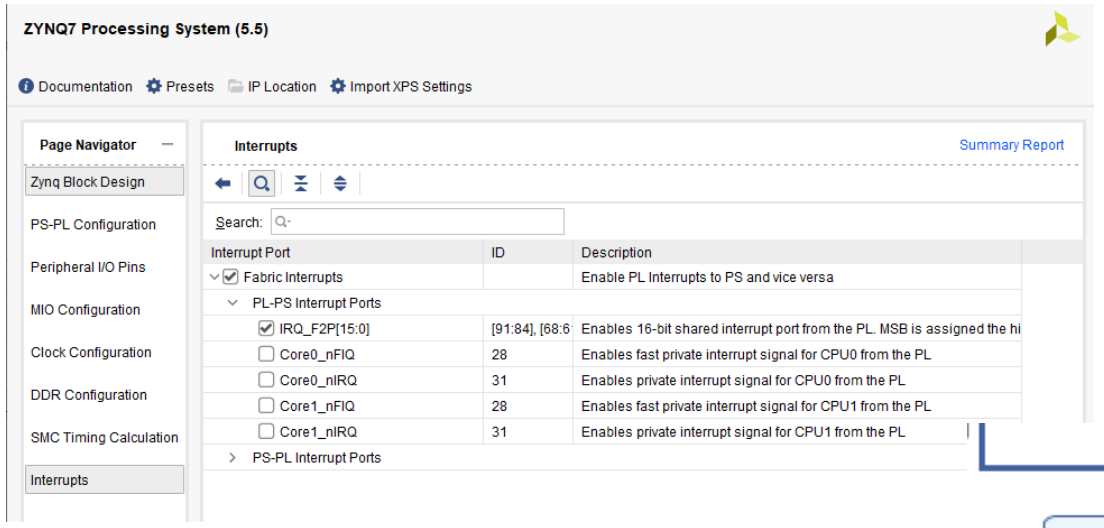
Diagram | Address Editor

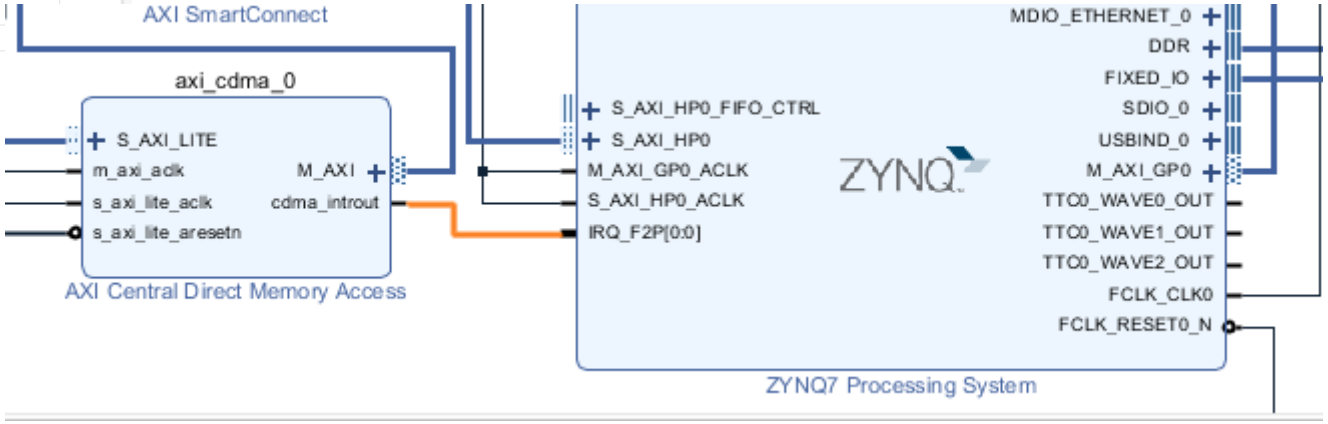
Default View



Connect the SmatConnect IP to M_AXI_GP0 on PS and S_AXI_LITE on CDMA

Enable Interrupt on PS an connect to CMDA





ADD BRAM controller and input BRAM IP

BLOCK DESIGN - design_1 *

Sources | Design | Signals | Board

- design_1
  - External Interfaces
  - Interface Connections
  - Nets
  - axi_bram_ctrl_0 (AXI BRAM Controller:4.1)
  - axi_cdma_0 (AXI Central Direct Memory Access:4.1)
  - blk_mem_gen_0 (Block Memory Generator:8.4)
  - processing_system7_0 (ZYNQ7 Processing System:5.5)
  - rst_ps7_0_100M (Processor System Reset:5.0)
  - smartconnect_0 (AXI SmartConnect:1.0)
  - smartconnect_1 (AXI SmartConnect:1.0)

Block Properties

axi_cdma_0

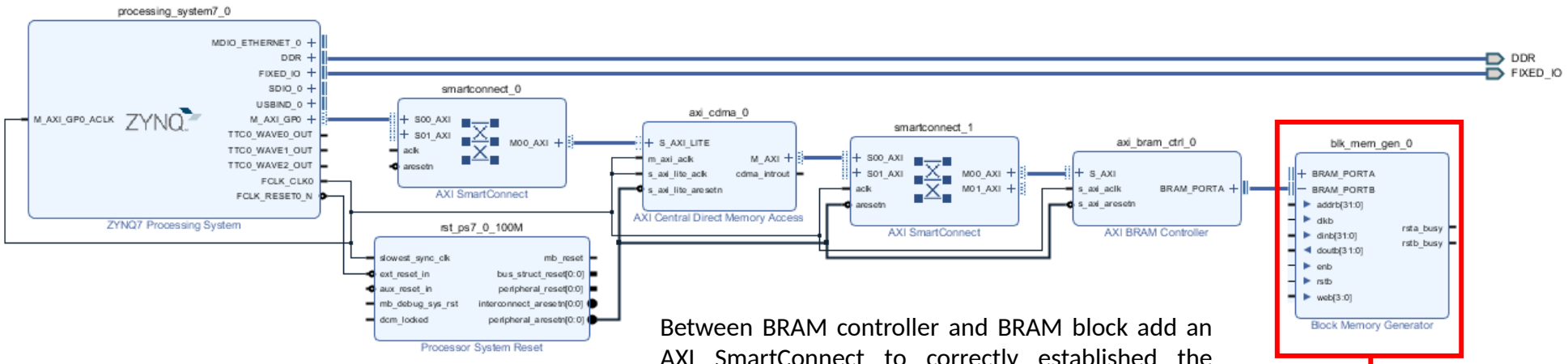
Name: axi_cdma_0

Parent name: design_1

General | Properties | IP

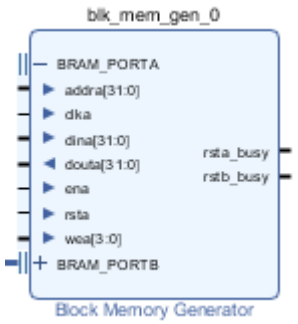
Diagram | Address Editor

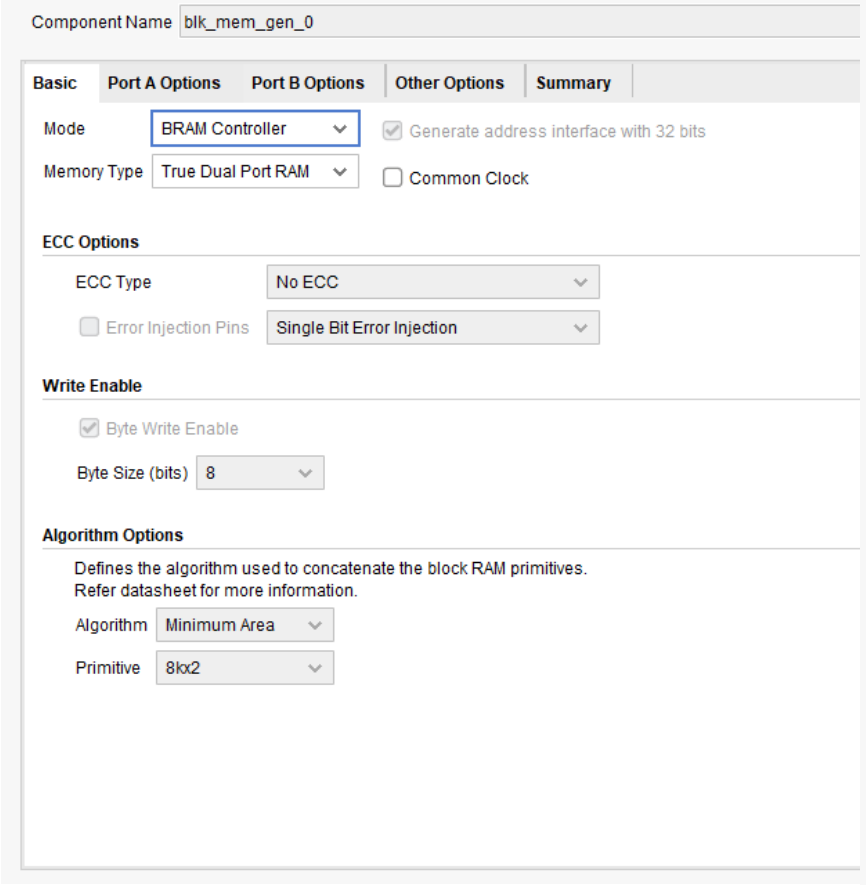
Default View

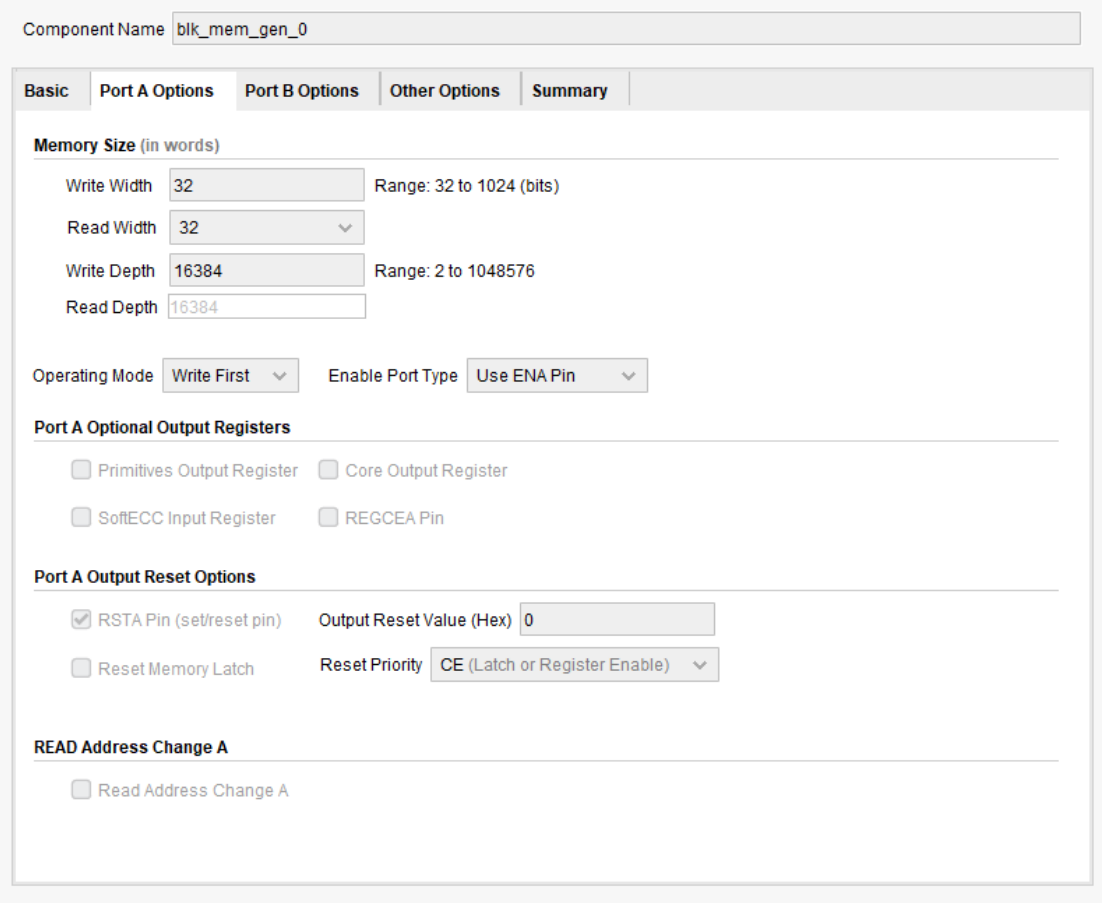


Between BRAM controller and BRAM block add an AXI SmartConnect to correctly established the connection to CDMA

Change the BRAM parameters and BRAM controller for desired example



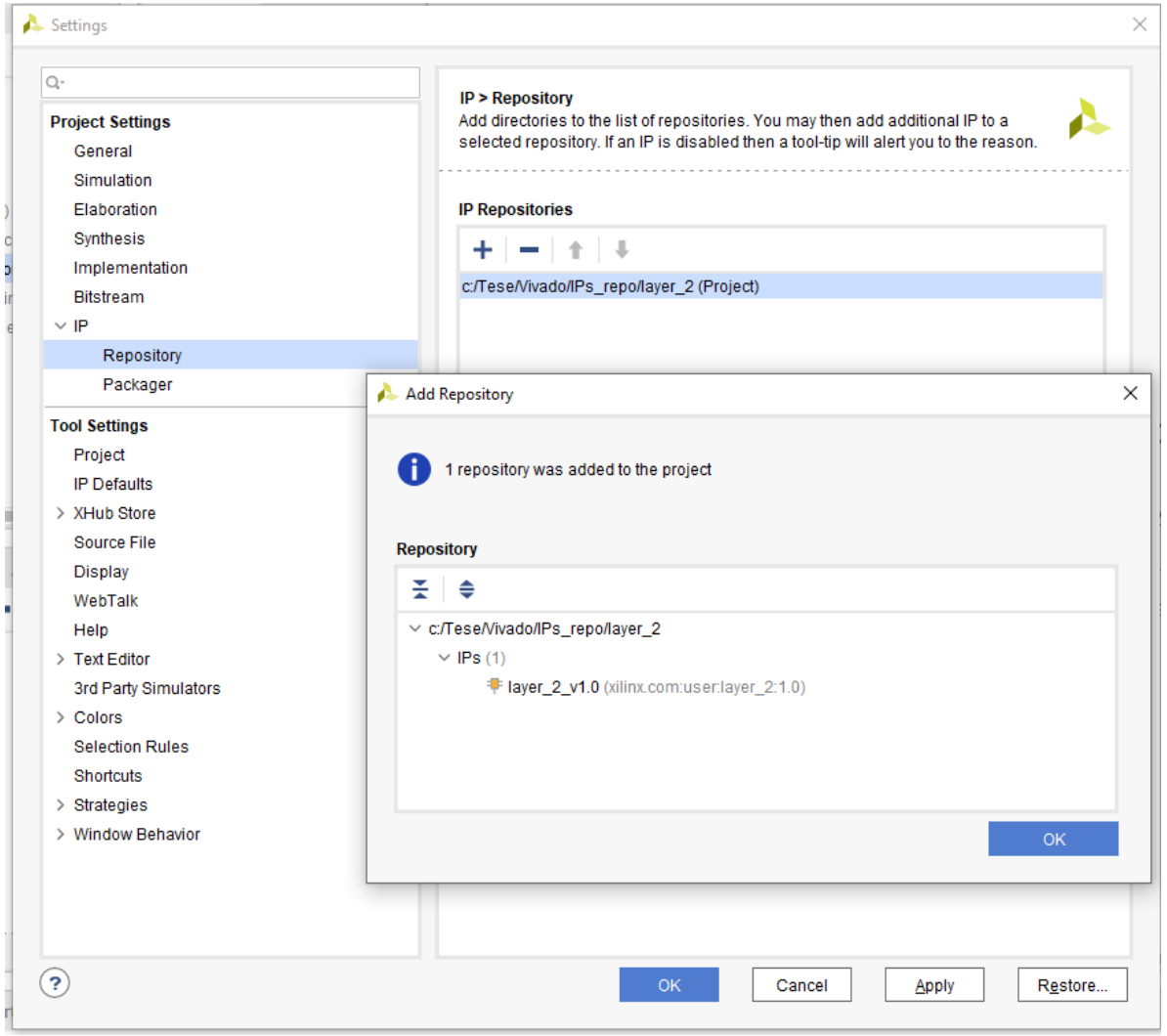




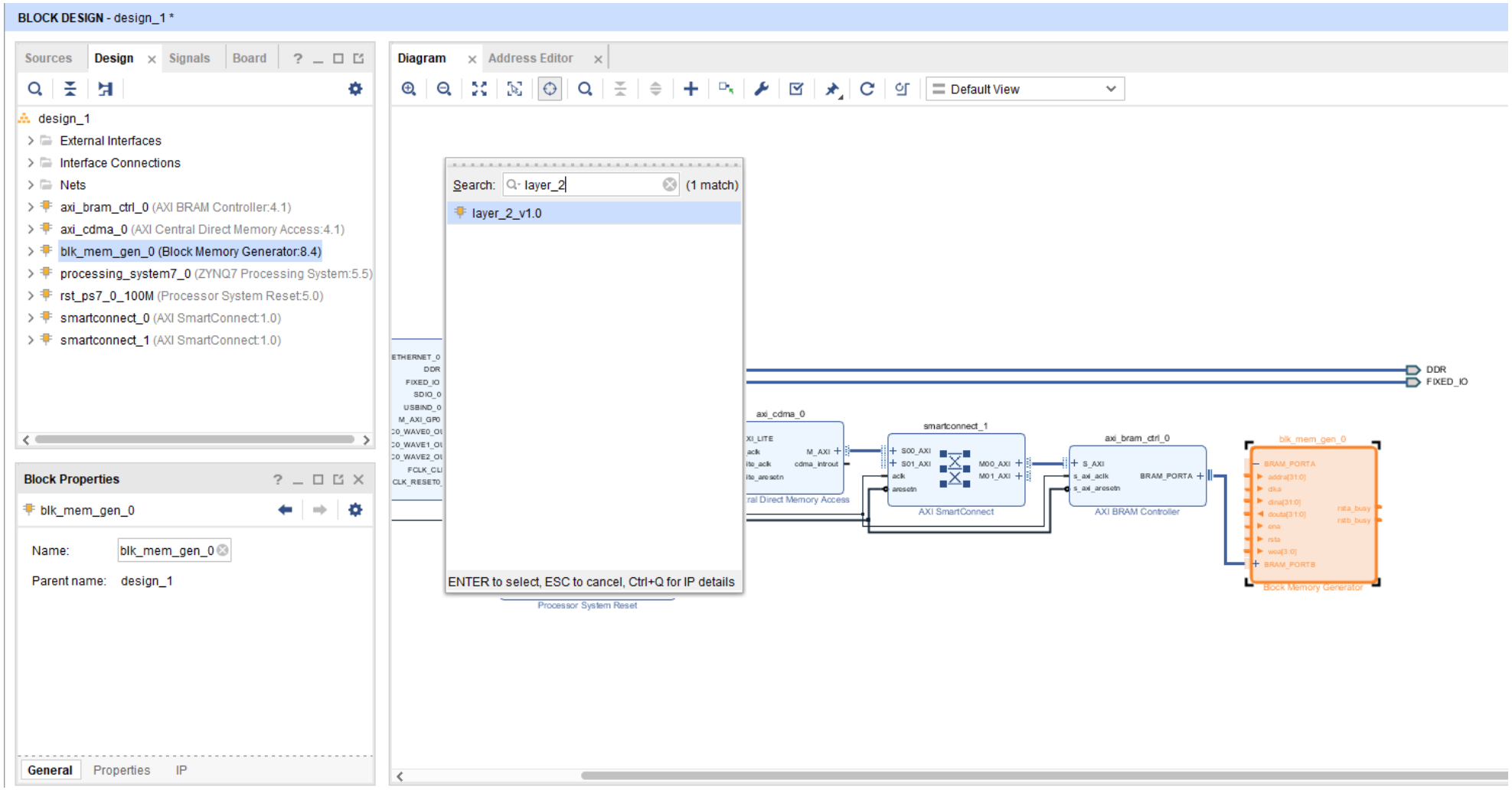
In this case is using a True Dual port, Port B is connected to CMDA and Port A will be connected to our IP.

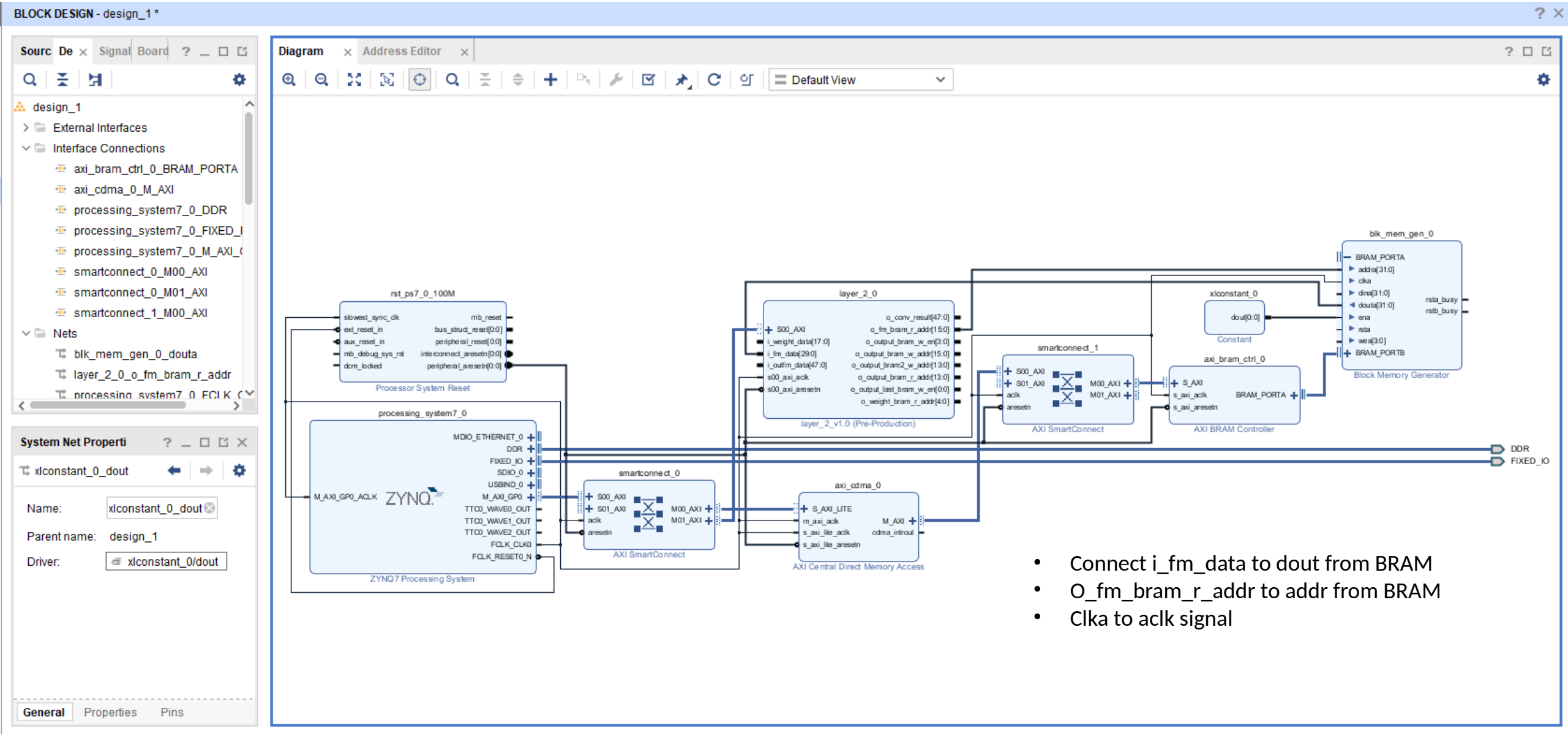
To add a custom IP go to Setting and..

- Choose correctly the path where the IP is located
- After, add it to IP repository of the current project



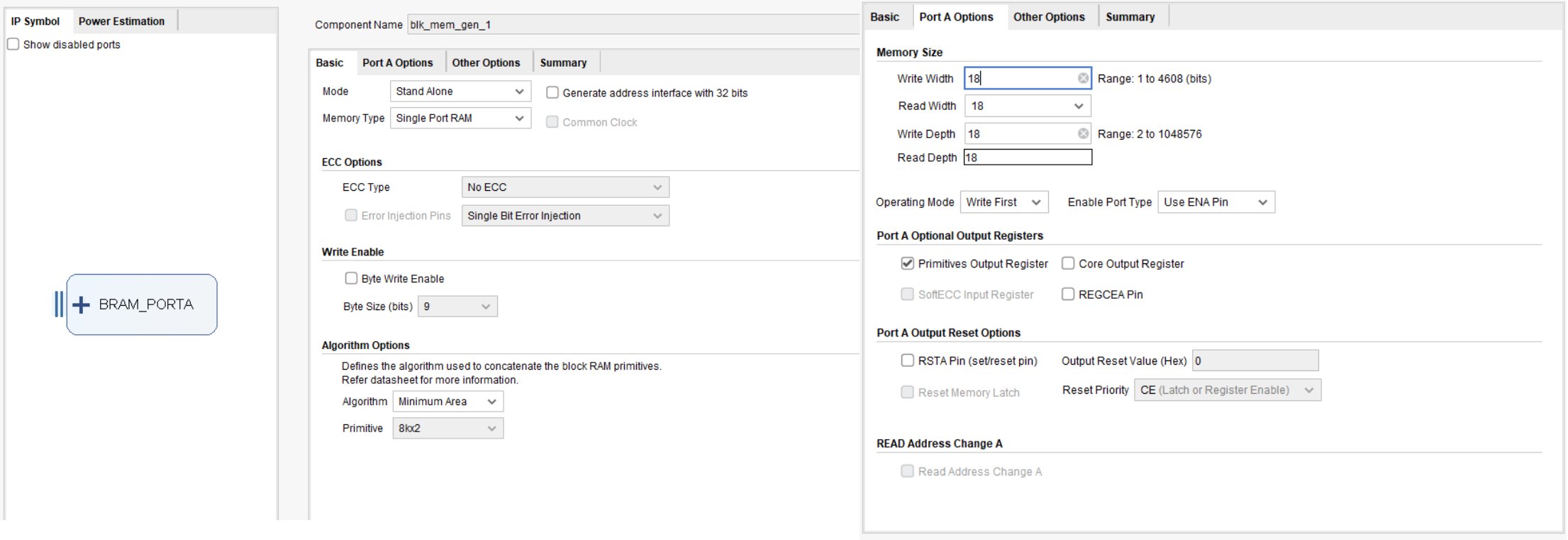
Search for IP on Block Design and add it





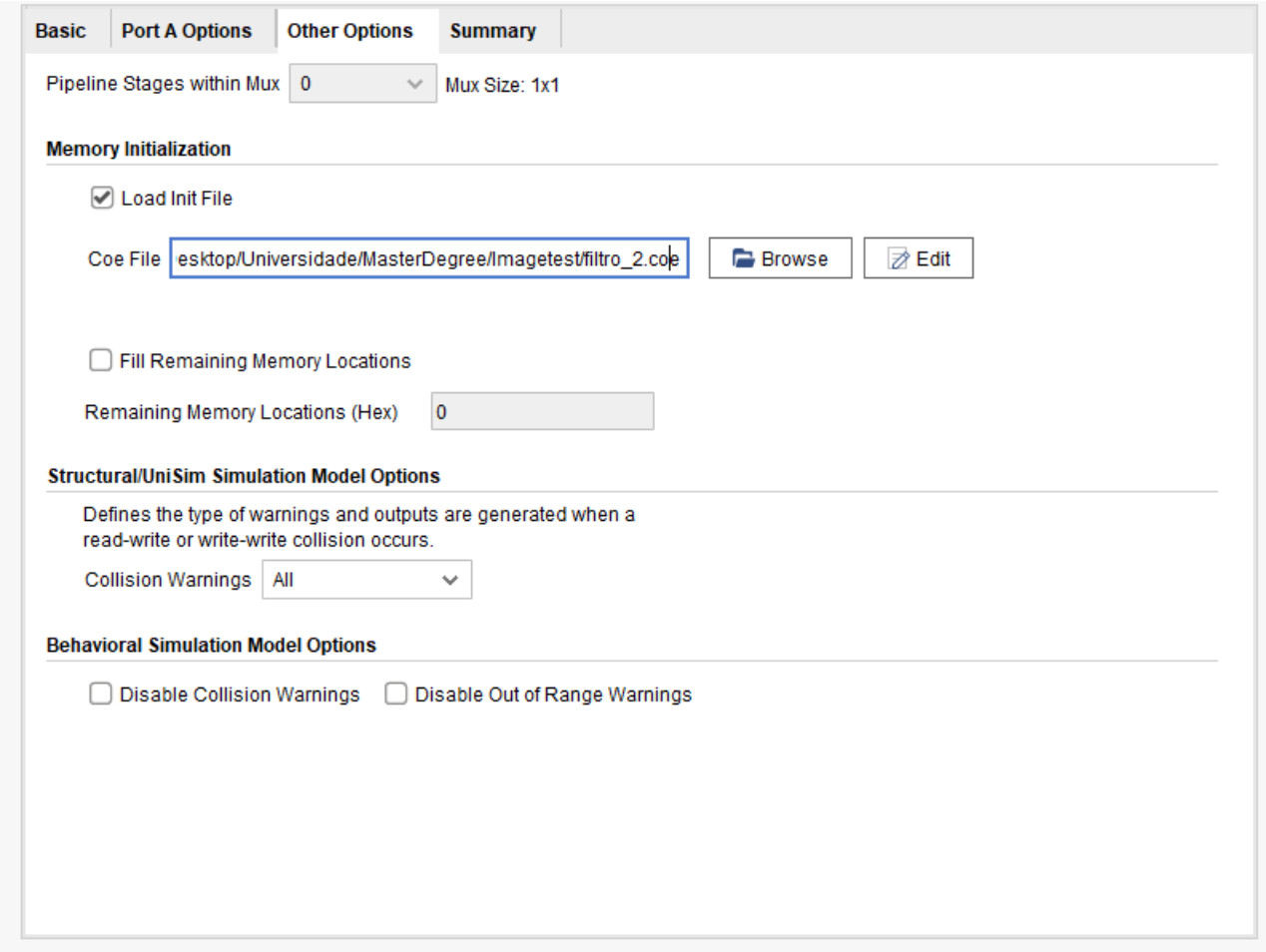
- Connect i_fm_data to dout from BRAM
- O_fm_bram_r_addr to addr from BRAM
- Clka to aclk signal

Add a BRAM to hold weight values



For a straightforward implementation, use a COE file to preload the BRAM memory with weight values

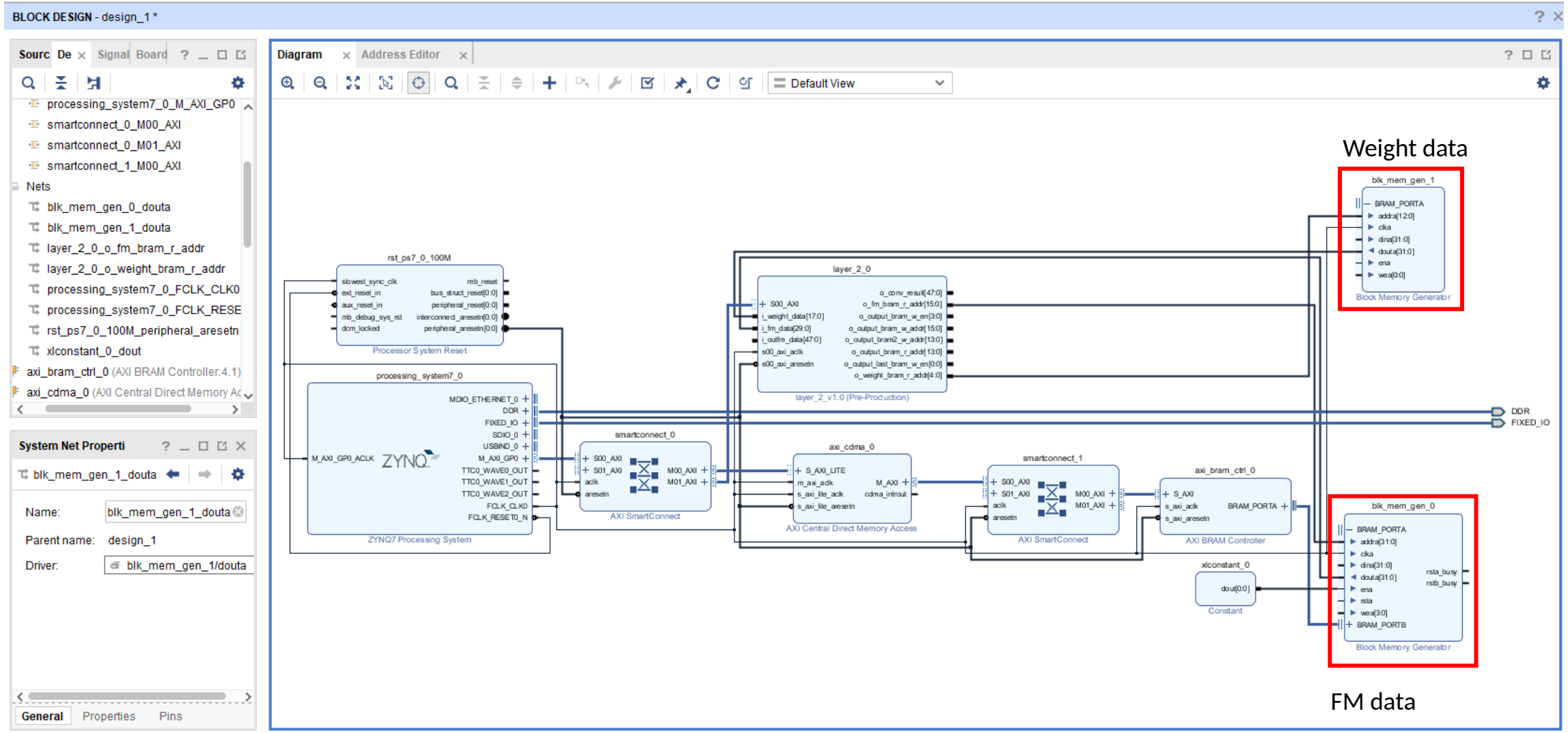
Add a BRAM to hold weight values




```
memory_initialization_radix=2;
memory_initialization_vector=0 0 0 0 1 0 0 0 0;
```

Identity filter used in filtro_2.coe file

Do the connections to layer IP

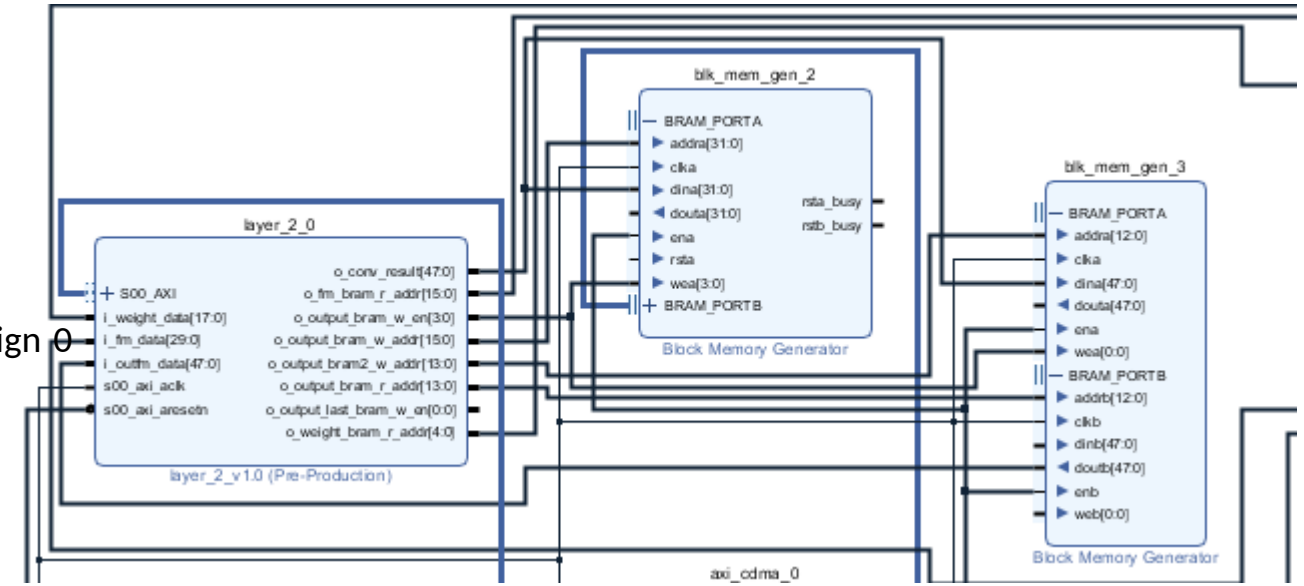


Add two output BRAMs

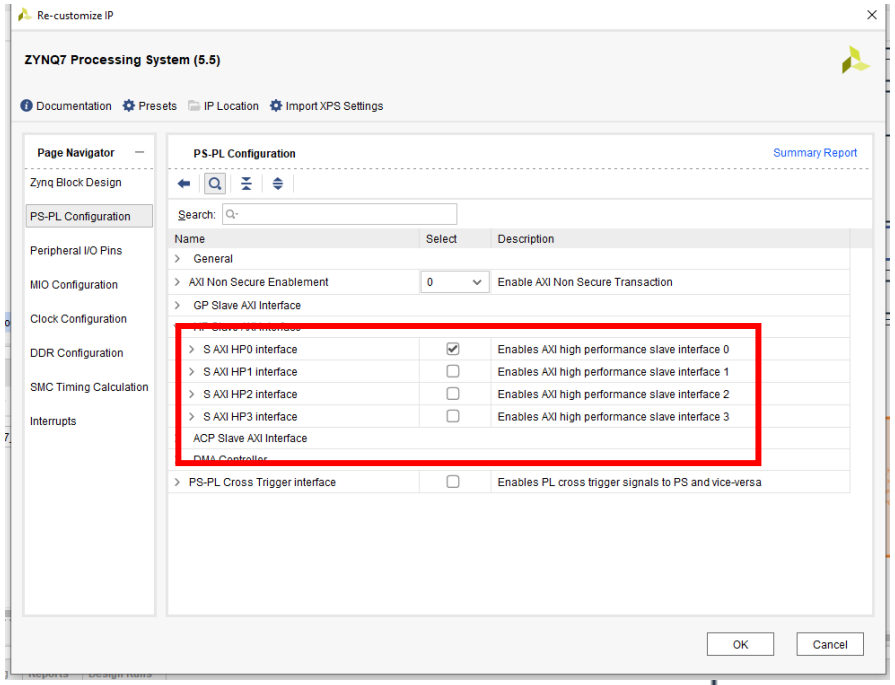
For a correct work of convolution process it is only necessary one BRAM block, in this case to read back and write on SD card it is required a connection from BRAM block and PS using a BRAM controller.

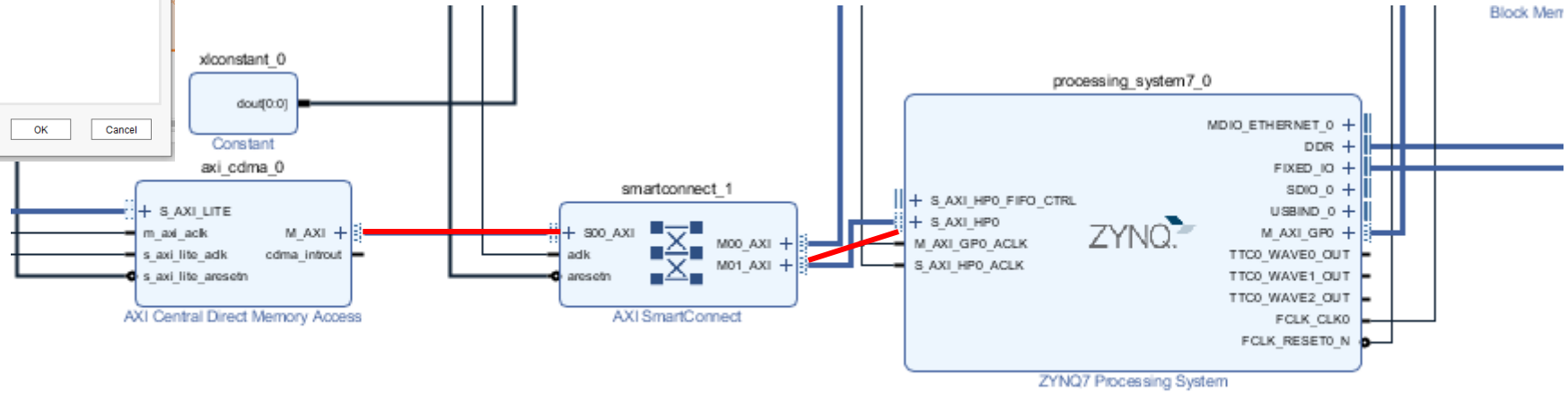
Correctly connect each input/output from layer ip with BRAM modules.

In case of incorrect work of BRAM blocks, use const IP to assign 0 value to dinb on blk_mem_gen3



Connect CDMA to High Performance on PS

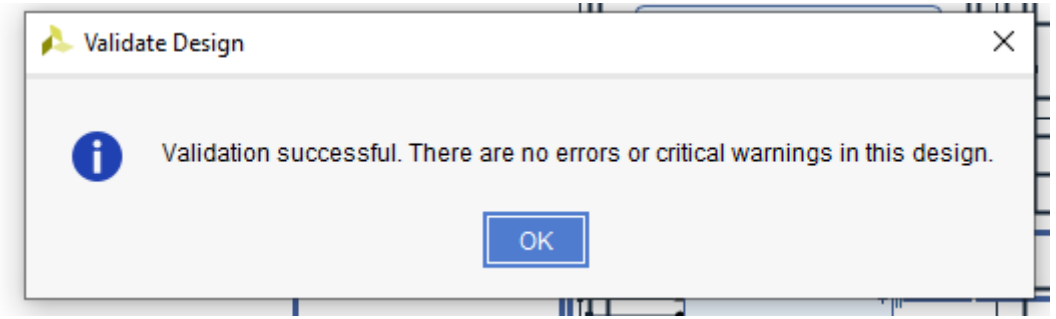




Run Validate Desing

Validate Design (F6)

Name	Interface	Slave Segment	Master Base Address	Range	Master High Address
Network 0					
/axi_cdma_0					
/axi_cdma_0/Data (32 address bits : 4G)					
/axi_bram_ctrl_0	S_AXI	Mem0	0xC000_0000	64K	0xC000_FFFF
/processing_system7_0	S_AXI_HP0	HP0_DDR_LOW	0x0000_0000	512M	0x1FFF_FFFF
/processing_system7_0					
/processing_system7_0/Data (32 address bits : 0x40000000 [1G])					
/axi_bram_ctrl_1	S_AXI	Mem0	0x4000_0000	64K	0x4000_FFFF
/axi_cdma_0	S_AXI_LITE	Reg	0x7E20_0000	64K	0x7E20_FFFF
/layer_2_0	S00_AXI	S00_AXI_reg	0x43C0_0000	64K	0x43C0_FFFF



Should appear not errors, if so check address editor tab for possible addresses collisions.

The range value should be updated for different sizes of image being processed. Larger images requires more data consumption.

After that..

- Generate output products
- Create HDL Wrapper
- Run synthesis
- Run implementation
- Generate bitstream

“Export SDK” after bitstream has been generated

File Edit Flow Tools Reports Window Layout View Help Quick Access

Implementation Complete

Default Layout

- Project
- Add Sources... Alt+A
- Close Project
- Save Block Design Ctrl+S
- Save Block Design As...
- Close Block Design
- Constraints
- Simulation Waveform
- Checkpoint
- IP
- Text Editor
- Import
- Export
  - Export Hardware...
  - Export Block Design...
  - Export Bitstream File...
  - Export Simulation...
- Print... Ctrl+P
- Exit

Select an object to see properties

Diagram | Address Editor

Assigned (5) Unassigned (0) Excluded (0) Hide All

Name	Interface	Slave Segment	Master Base Address	Range	Master High Address
Network 0					
/axi_cdma_0					
/axi_cdma_0/Data (32 address bits : 4G)					
/axi_bram_ctrl_1	S_AXI	Mem0	0x4110_0000	64K	0x4110_FFFF
/processing_system7_0	S_AXI_HP0	HP0_DDR_LOWOCM	0x0000_0000	512M	0x1FFF_FFFF
/processing_system7_0					
/processing_system7_0/Data (32 address bits : 0x40000000 [1G])					
/axi_bram_ctrl_2	S_AXI	Mem0	0x4200_0000	64K	0x4200_FFFF
/axi_cdma_0	S_AXI_LITE	Reg	0x7E20_0000	64K	0x7E20_FFFF
/myip_0	S00_AXI	S00_AXI_reg	0x43C0_0000	64K	0x43C0_FFFF

Tcl Console | Messages | Log | Reports | Design Runs

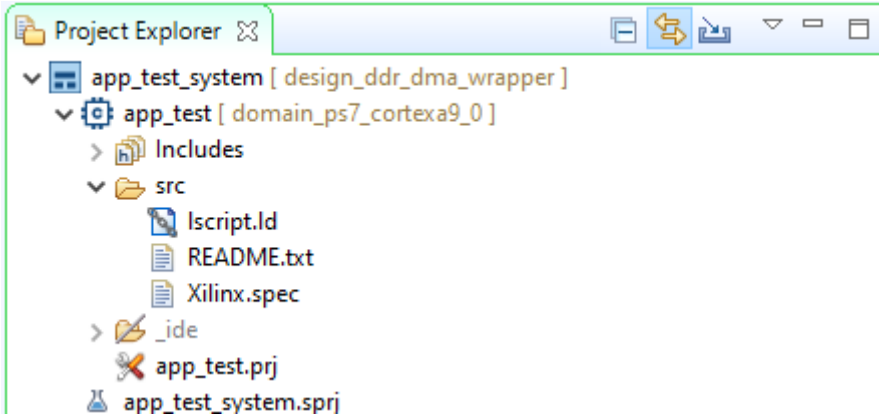
Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAM	URAM	DSP	Start	Elapsed	Run Strategy	Report Strategy
synth_1 (active)	**constrs_1**	**synth_design Complete!**								**0**	**0**	**0.0**	**0**	**0**	**7/28/21, 3:26 PM**	**00:01:15**	**Vivado Synthesis Defaults (Vivado Synthesis 2020)**	**Vivado Synthesis Default Reports (Vivado S**
impl_1	constrs_1	route_design Complete!	1.308	0.000	0.023	0.000	0.000	1.812	0	6621	8190	56.0	0	9	7/28/21, 3:27 PM	00:05:08	Vivado Implementation Defaults (Vivado Implementation 2020)	Vivado Implementation Default Reports (Viva
synth_1_copy_1	constrs_1	Not started															Vivado Synthesis Defaults* (Vivado Synthesis 2020)	Vivado Synthesis Default Reports (Vivado Sy
Out-of-Context Module Runs																		
design_1		Submodule Runs Complete													3/6/21, 12:39 PM	3,457:46:33		

Export hardware platform

- Platform type = Fixed
- Output = Include bitstream
- Choose the name of the hardware platform type and the location where the xsa file will be stored – default option is viable
- Finish to export the platform

Xilinx Vitis

- Create new application project
- Select the .xsa file previously generated
- Write the name of the application
- For the template, select the “Empty Application” option.
- The application structure should look similar to this:



Add source files

- From path: “C:\Xilinx\Vitis\2020.1\data\embeddedsw\XilinxProcessorIPLib\drivers\axicdma_v4_7\examples” copy to src the file: “xaxicdma_example_simple_intr.c”
- This file is used since some functions are already implemented
- Nonetheless we need to edit this file.
- “xparameters.h” file is where all the macros with the base address for the DDR memory and the BRAM controllers are defined.

Includes

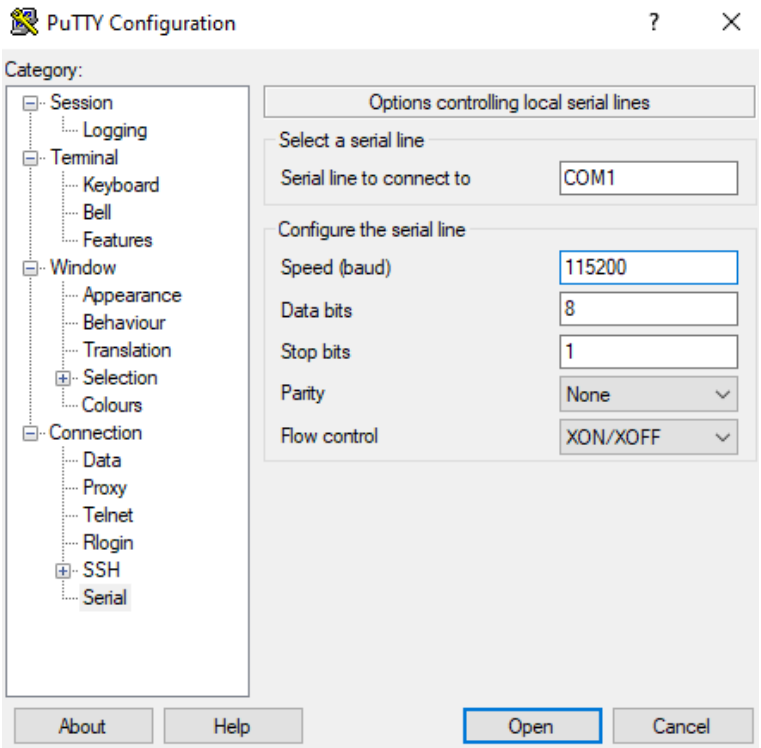
- Make sure to include this set of libraries. As we pretend to transfer data from the Sdcard to the DDR memory, some functions will be used for that purpose.
- Depending on the data size used inside the hardware module, the functions may need to be edited to work with a certain data width.

```
#include "xparameters.h"
#include "SDCARD.h"
#include "CONFIGS.h"
#include "stdio.h"
#include "math.h"
```

```
int sdCardDriverInit();
int writeFramesToSDCard(char* SD_File, u8 *FrameBuffer, u32 SizeBuffer, u32 offset);
int readFramesFromSDCard(char* SD_File, u16 n_frames, u8 *FrameBuffer, u32 SizeBuffer);
int read8FramesFromSDCard(char* SD_File, u8 *FrameBuffer, u32 SizeBuffer, u32 offset);
```

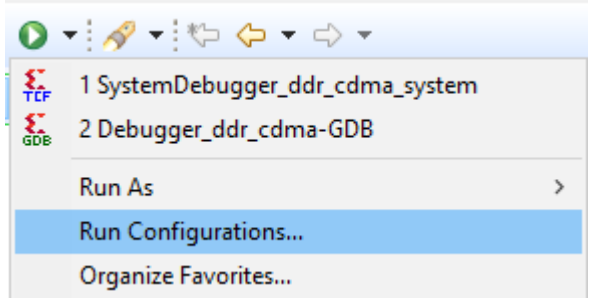
Program the FPGA

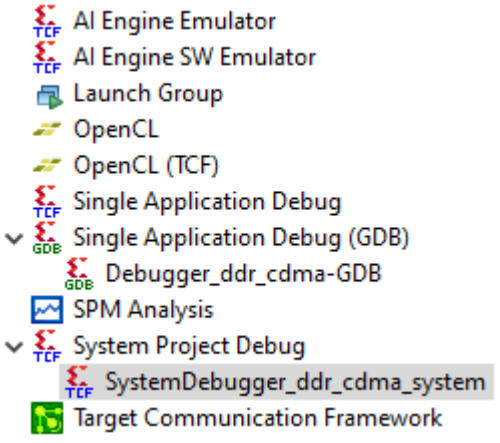
- Select “Program the FPGA”
- Select the correct bitstream and then select “Program”
- Before Run the application you may want to setup the connection to the target platform first.
- In this case, PuTTY was used
- Set the port used and the correct baudrate

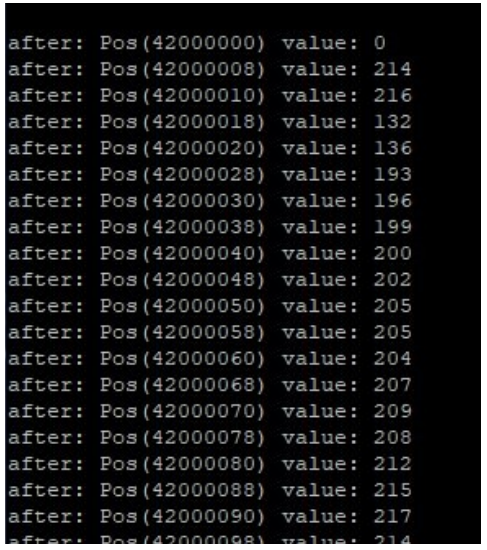


Run the application

- Click on the created application
- Select “Run configurations” from the top-bar
- Select “System debugger”
- Hit run. In the terminal the values sent from the DDR to the BRAM will be displayed as well as the output of processing.
- Depending on the IFM provided, the output values will vary.







```
after: Pos(42000000) value: 0
after: Pos(42000008) value: 214
after: Pos(42000010) value: 216
after: Pos(42000018) value: 132
after: Pos(42000020) value: 136
after: Pos(42000028) value: 193
after: Pos(42000030) value: 196
after: Pos(42000038) value: 199
after: Pos(42000040) value: 200
after: Pos(42000048) value: 202
after: Pos(42000050) value: 205
after: Pos(42000058) value: 205
after: Pos(42000060) value: 204
after: Pos(42000068) value: 207
after: Pos(42000070) value: 209
after: Pos(42000078) value: 208
after: Pos(42000080) value: 212
after: Pos(42000088) value: 215
after: Pos(42000090) value: 217
```