ISTANBUL TECHNICAL UNIVERSITY

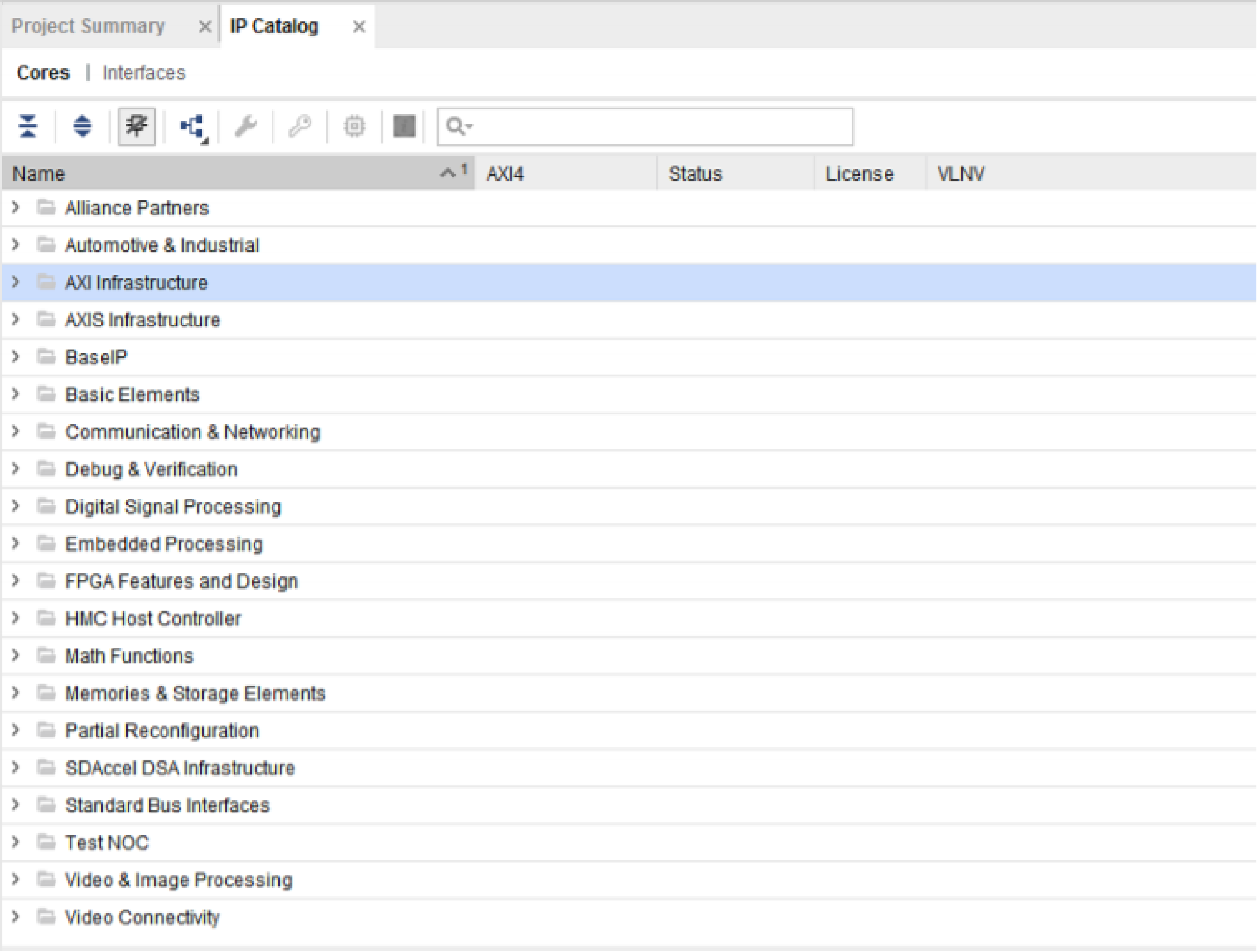
EMBEDDED SYSTEMS DESIGN LABORATORY

Block Memory Generator

Serdar Duran

IP (Intellectual Property)

- **IP core**, or **IP block** refers to preconfigured and reusable unit of a logic circuit design.
- Vivado has a rich library of **IP cores** including:
  - DSP blocks
  - Image Processing blocks
  - Communication & Interfacing blocks
  - Channel Encoding & Decoding
  - Math blocks etc.
- In Vivado, you can customize and add IP cores from the **IP Catalog** into a project.



Project Summary | IP Catalog

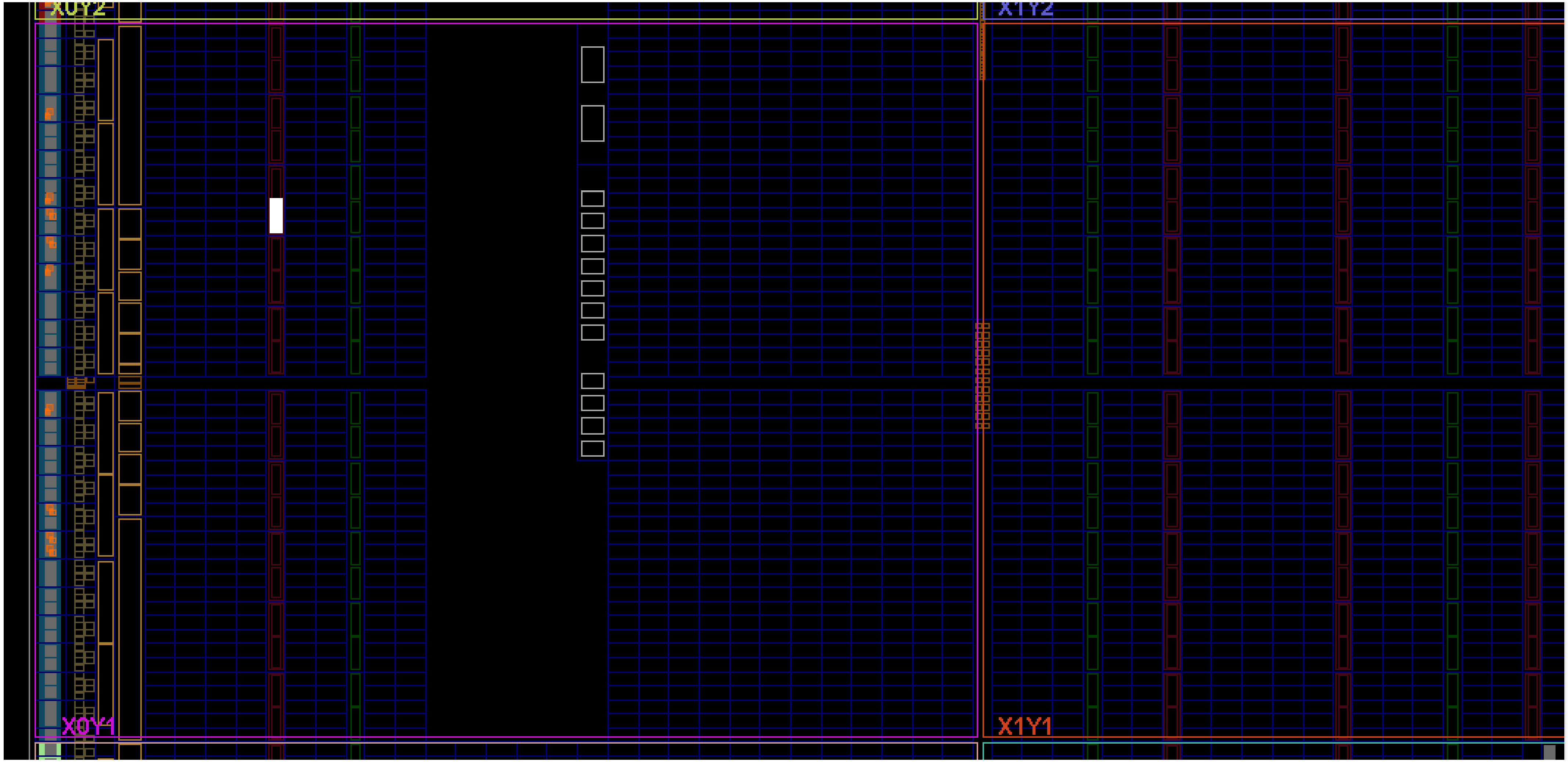
Cores | Interfaces

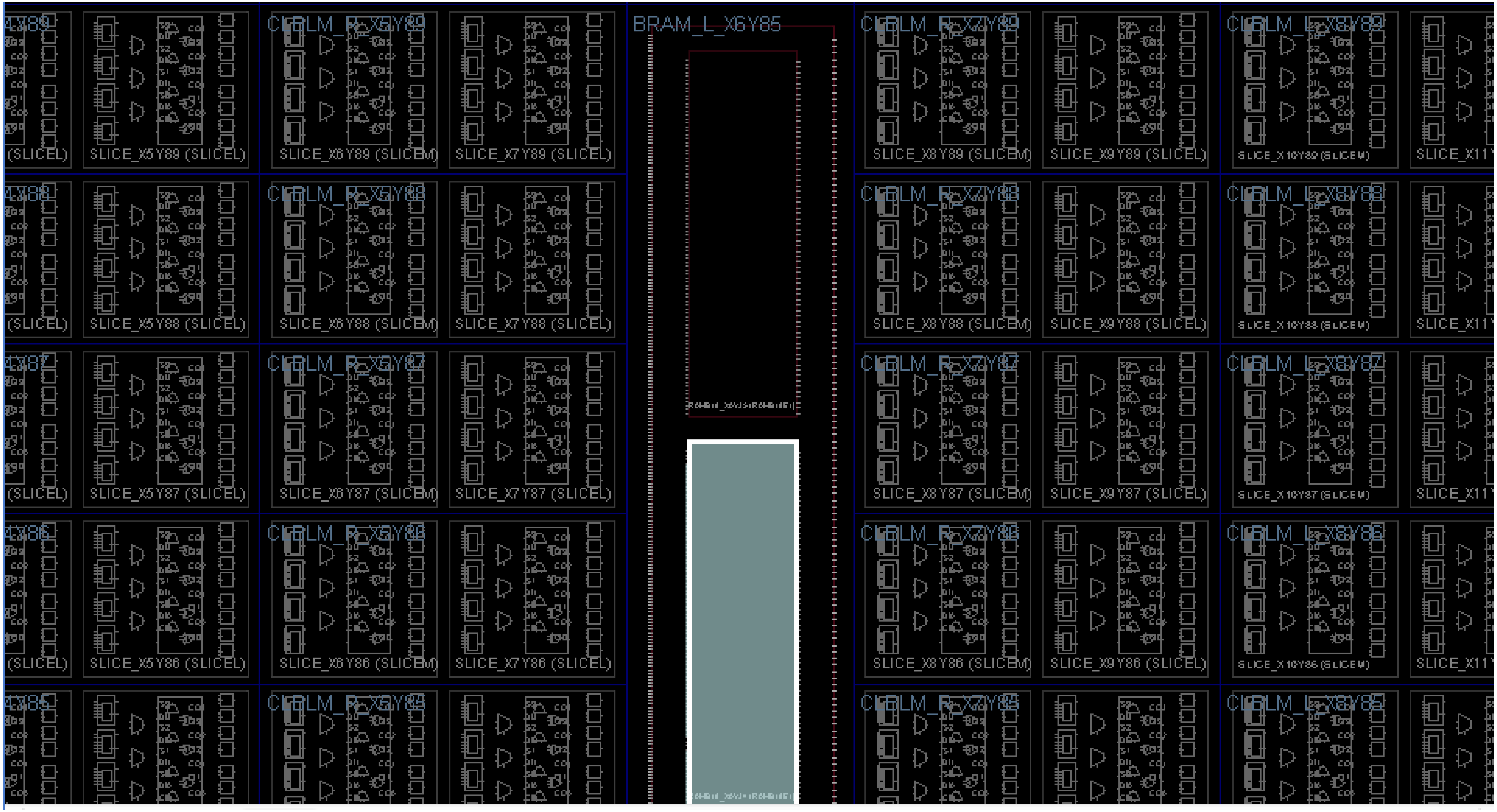
Name	AXI4	Status	License	VLNV
Alliance Partners				
Automotive & Industrial				
AXI Infrastructure				
AXIS Infrastructure				
BaseIP				
Basic Elements				
Communication & Networking				
Debug & Verification				
Digital Signal Processing				
Embedded Processing				
FPGA Features and Design				
HMC Host Controller				
Math Functions				
Memories & Storage Elements				
Partial Reconfiguration				
SDAccel DSA Infrastructure				
Standard Bus Interfaces				
Test NOC				
Video & Image Processing				
Video Connectivity				

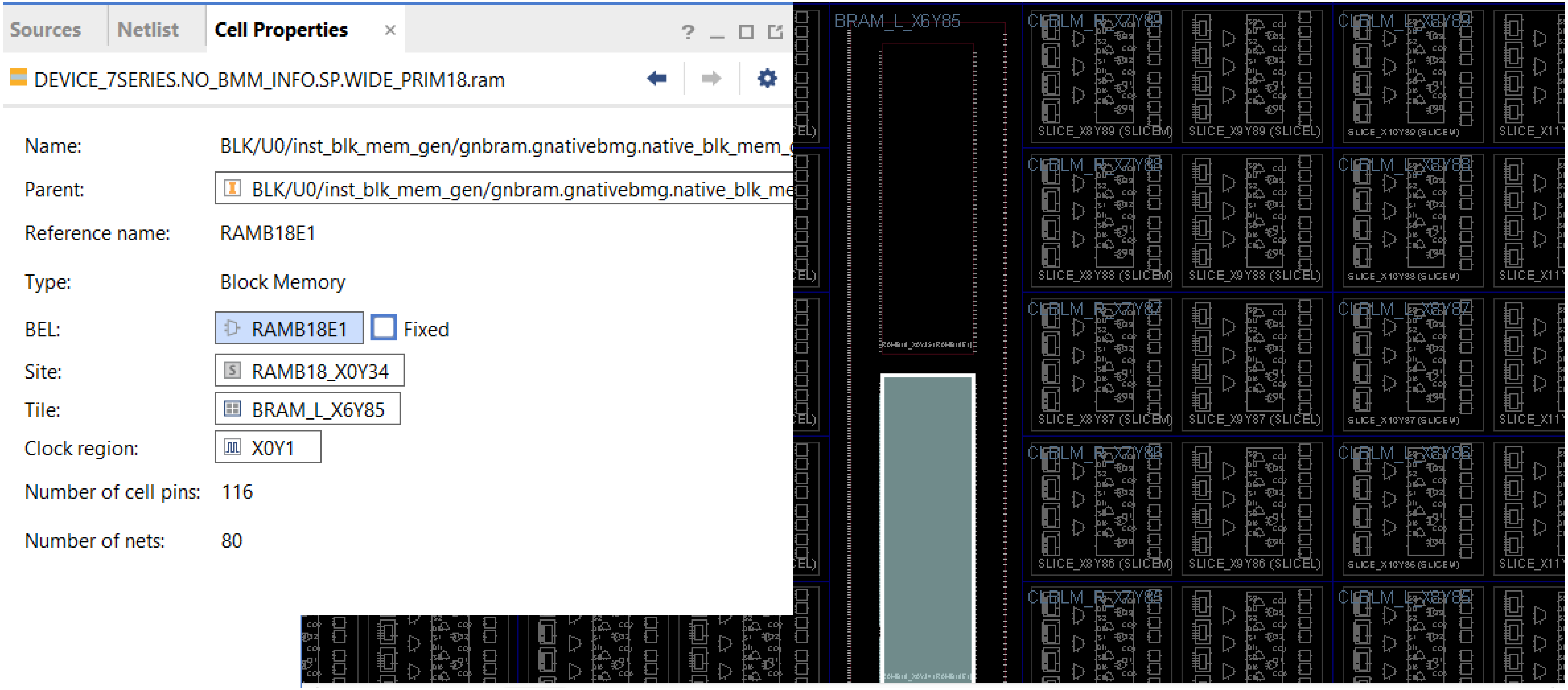
Block Memory Generator

- The Block Memory Generator IP core uses embedded Block Memory primitives in Xilinx FPGAs.
- It extends the functionality and capability of a single **primitive** to memories of arbitrary widths and depths.

BRAM Cells




X4Y89
(SLICEL)
SLICE_X5Y89 (SLICEL)
CLBLM_R_X5Y89
SLICE_X6Y89 (SLICEM)
SLICE_X7Y89 (SLICEL)
BRAM_L_X6Y85
CLBLM_R_X7Y89
SLICE_X8Y89 (SLICEM)
SLICE_X9Y89 (SLICEL)
CLBLM_L_X8Y89
SLICE_X10Y89 (SLICEM)
SLICE_X11
X4Y88
(SLICEL)
SLICE_X5Y88 (SLICEL)
CLBLM_R_X5Y88
SLICE_X6Y88 (SLICEM)
SLICE_X7Y88 (SLICEL)
CLBLM_R_X7Y88
SLICE_X8Y88 (SLICEM)
SLICE_X9Y88 (SLICEL)
CLBLM_L_X8Y88
SLICE_X10Y88 (SLICEM)
SLICE_X11
X4Y87
(SLICEL)
SLICE_X5Y87 (SLICEL)
CLBLM_R_X5Y87
SLICE_X6Y87 (SLICEM)
SLICE_X7Y87 (SLICEL)
CLBLM_R_X7Y87
SLICE_X8Y87 (SLICEM)
SLICE_X9Y87 (SLICEL)
CLBLM_L_X8Y87
SLICE_X10Y87 (SLICEM)
SLICE_X11
X4Y86
(SLICEL)
SLICE_X5Y86 (SLICEL)
CLBLM_R_X5Y86
SLICE_X6Y86 (SLICEM)
SLICE_X7Y86 (SLICEL)
CLBLM_R_X7Y86
SLICE_X8Y86 (SLICEM)
SLICE_X9Y86 (SLICEL)
CLBLM_L_X8Y86
SLICE_X10Y86 (SLICEM)
SLICE_X11
X4Y85
CLBLM_R_X5Y85
CLBLM_R_X7Y85
CLBLM_L_X8Y85

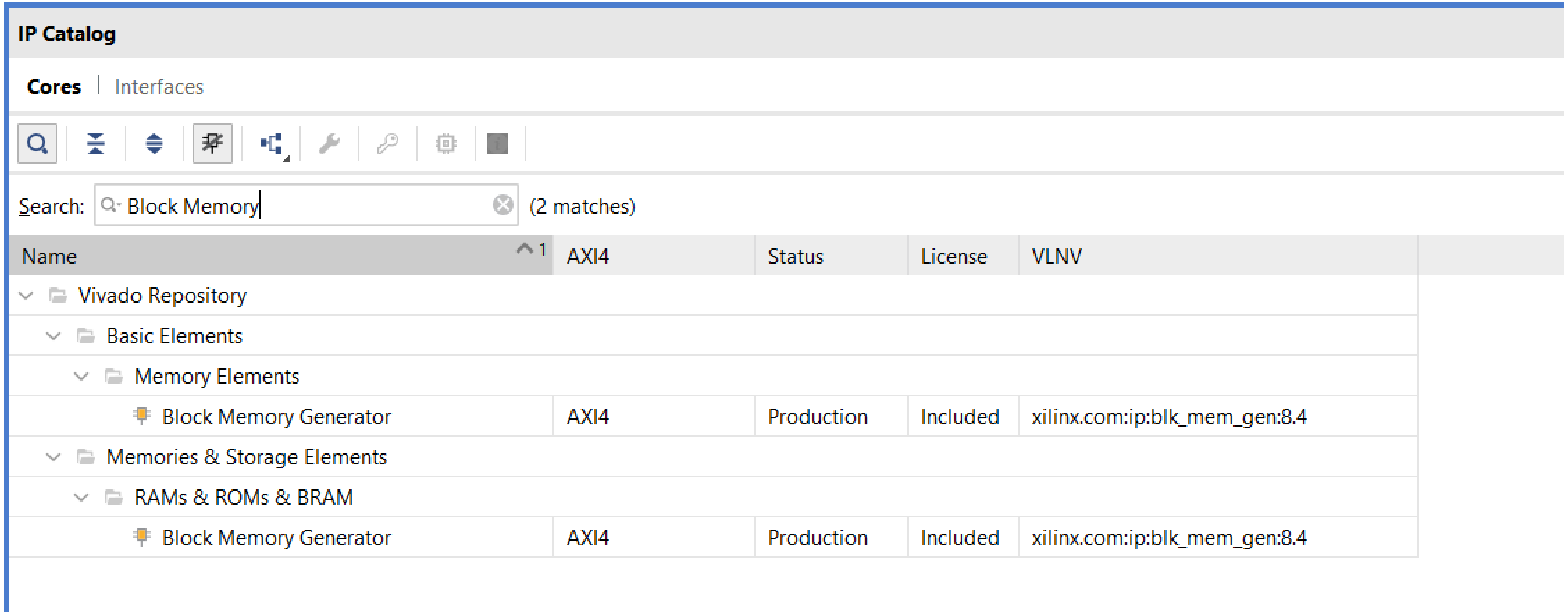


Sources | Netlist | **Cell Properties** ×

DEVICE_7SERIES.NO_BMM_INFO.SP.WIDE_PRIM18.ram

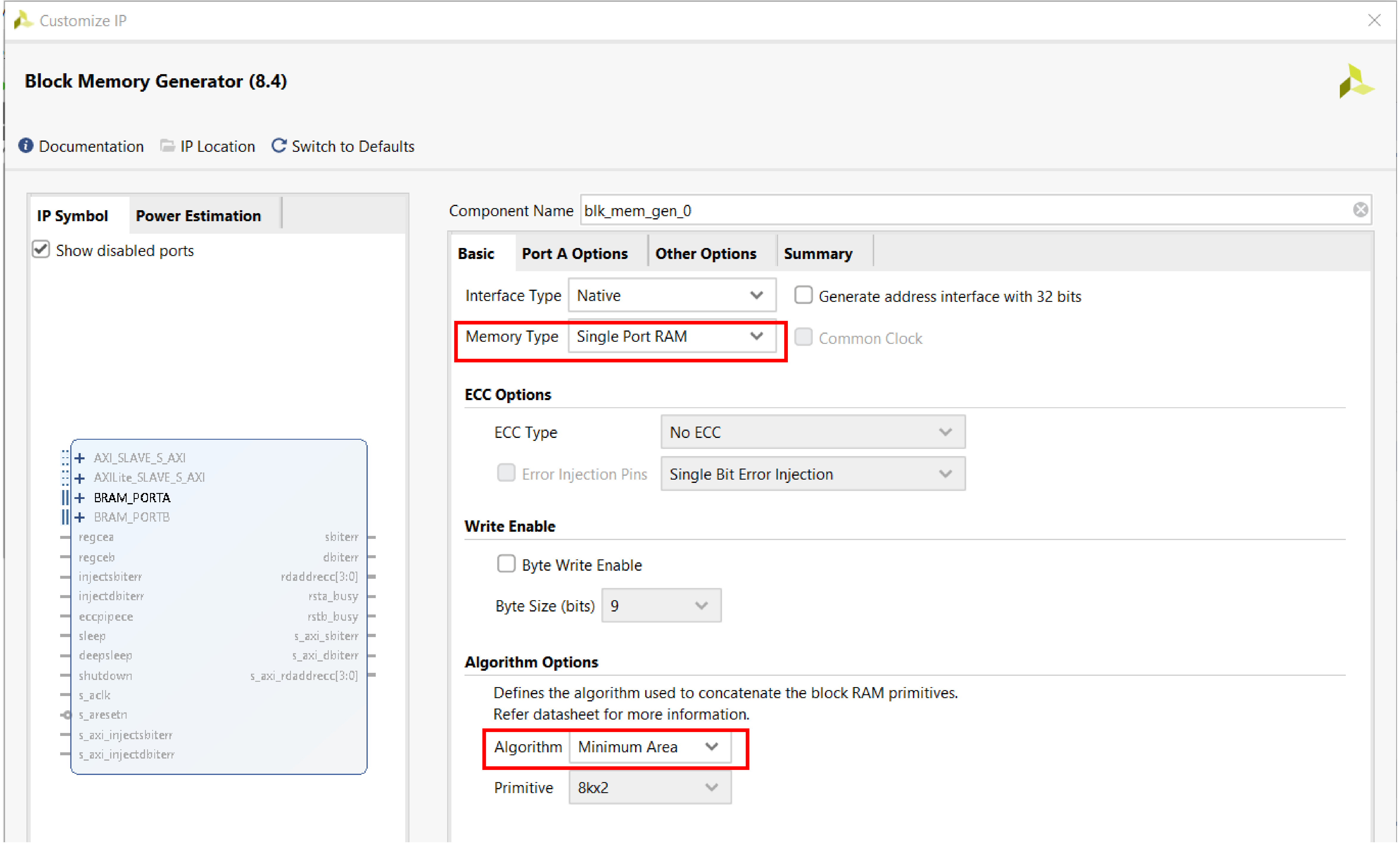
Name:	BLK/U0/inst_blk_mem_gen/gnbram.gnativebmg.native_blk_mem_
Parent:	BLK/U0/inst_blk_mem_gen/gnbram.gnativebmg.native_blk_me
Reference name:	RAMB18E1
Type:	Block Memory
BEL:	RAMB18E1 □ Fixed
Site:	RAMB18_X0Y34
Tile:	BRAM_L_X6Y85
Clock region:	X0Y1
Number of cell pins:	116
Number of nets:	80

- From the IP Catalog:



Search: Block Memory (2 matches)

Name	AXI4	Status	License	VLNV
Vivado Repository				
Basic Elements				
Memory Elements				
Block Memory Generator	AXI4	Production	Included	xilinx.com:ip:blk_mem_gen:8.4
Memories & Storage Elements				
RAMs & ROMs & BRAM				
Block Memory Generator	AXI4	Production	Included	xilinx.com:ip:blk_mem_gen:8.4



Block Memory Generator (8.4)

☑ Show disabled ports

Component Name: blk_mem_gen_0

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

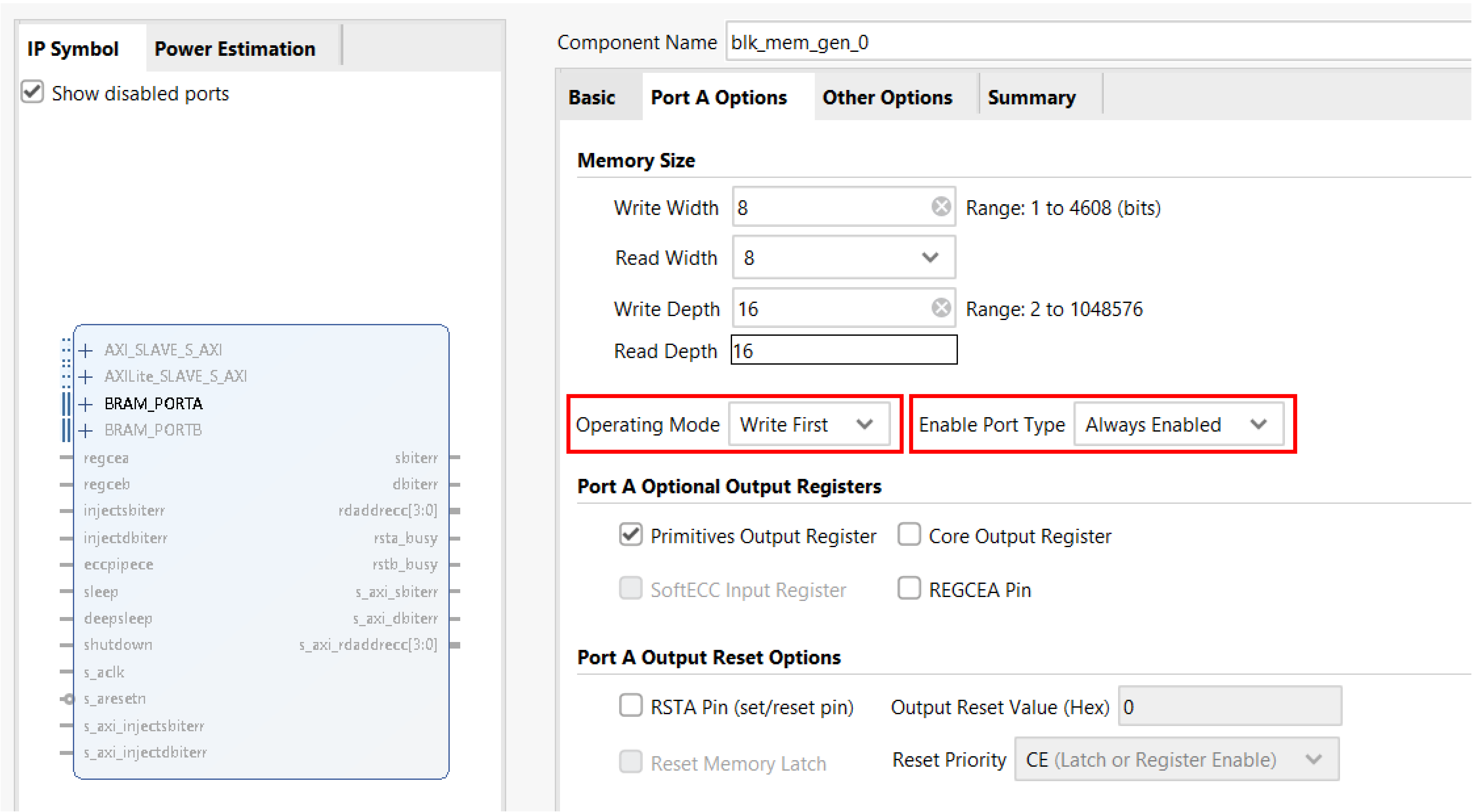
Byte Size (bits): 9

Algorithm Options

Defines the algorithm used to concatenate the block RAM primitives.
Refer datasheet for more information.

Algorithm: Minimum Area

Primitive: 8kx2



☑ Show disabled ports

Component Name: blk_mem_gen_0

Memory Size

Write Width: 8 — Range: 1 to 4608 (bits)

Read Width: 8

Write Depth: 16 — Range: 2 to 1048576

Read Depth: 16

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)

Memory Types

- Single-port RAM
- Simple Dual-port RAM
- True Dual-port RAM

- Single-port ROM
- Dual-port ROM

Single-port RAM

- It allows **Read** and **Write** access to the memory through a single port (port A).

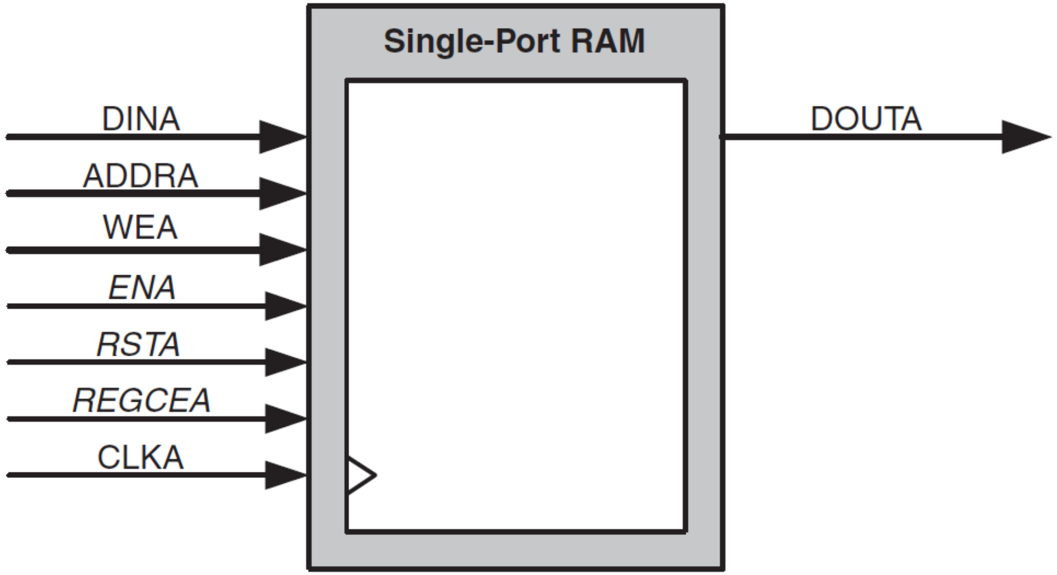


Figure 3-3: **Single-port RAM**

Simple Dual-port RAM

- Dual-port RAM provides two ports, A and B.
- **Write** access to the memory is allowed through port A, and **Read** access is allowed through port B.

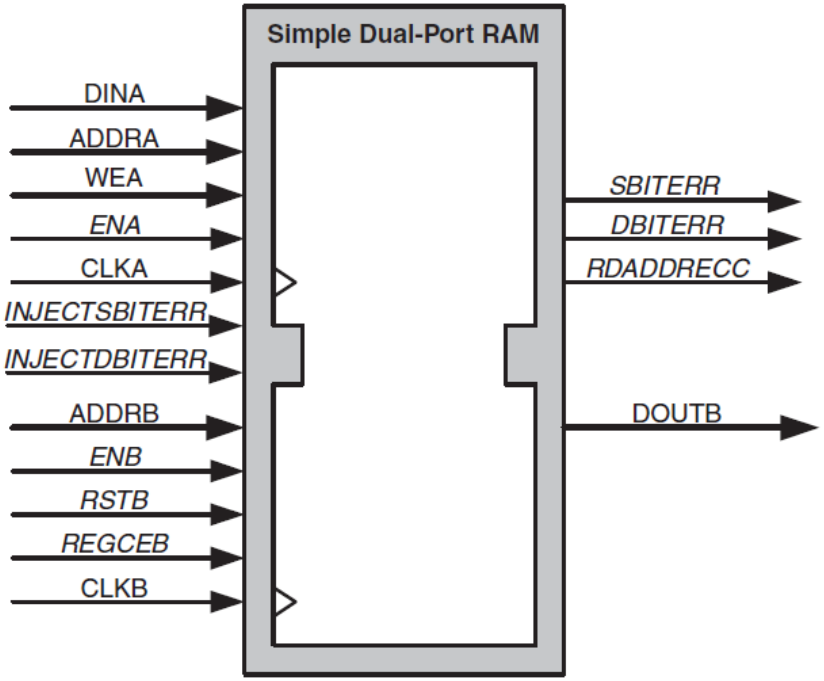


Figure 3-4: **Simple Dual-port RAM**

True Dual-port RAM

- The True Dual-port RAM provides two ports, A and B,
- **Read** and **Write** accesses to the memory are allowed on either port.

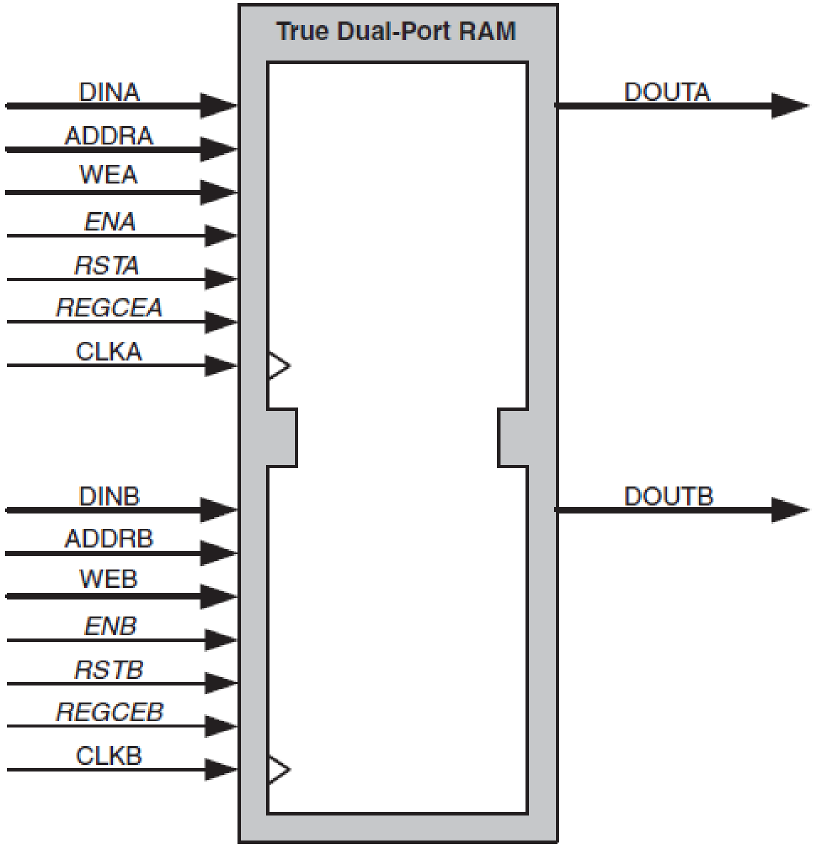


Figure 3-5: **True Dual-port RAM**

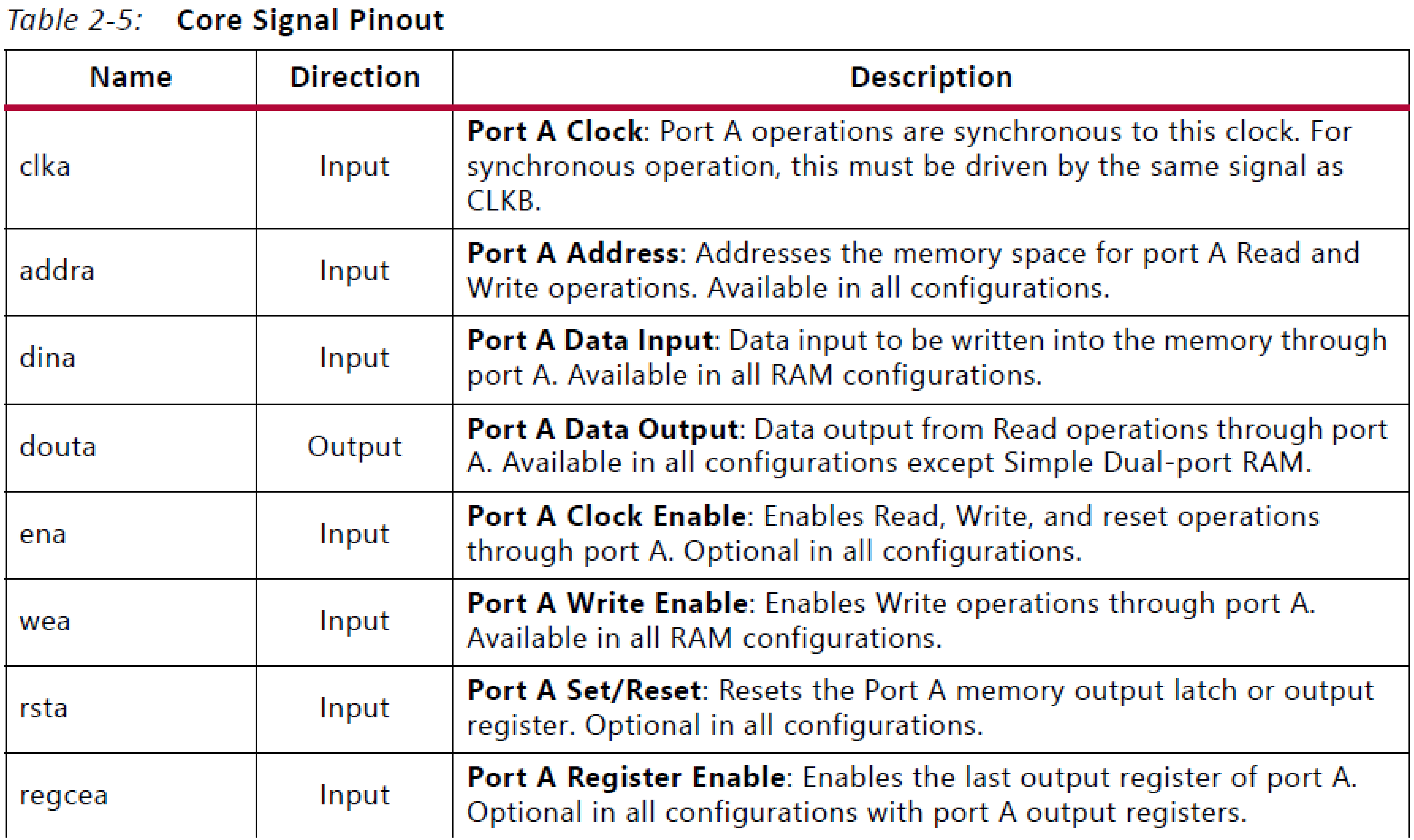


Table 2-5: **Core Signal Pinout**

Name	Direction	Description
clka	Input	**Port A Clock**: Port A operations are synchronous to this clock. For synchronous operation, this must be driven by the same signal as CLKB.
addra	Input	**Port A Address**: Addresses the memory space for port A Read and Write operations. Available in all configurations.
dina	Input	**Port A Data Input**: Data input to be written into the memory through port A. Available in all RAM configurations.
douta	Output	**Port A Data Output**: Data output from Read operations through port A. Available in all configurations except Simple Dual-port RAM.
ena	Input	**Port A Clock Enable**: Enables Read, Write, and reset operations through port A. Optional in all configurations.
wea	Input	**Port A Write Enable**: Enables Write operations through port A. Available in all RAM configurations.
rsta	Input	**Port A Set/Reset**: Resets the Port A memory output latch or output register. Optional in all configurations.
regcea	Input	**Port A Register Enable**: Enables the last output register of port A. Optional in all configurations with port A output registers.

Operating Mode

- The operating mode for each port determines the relationship between the Write and Read interfaces for that port.
    1) Write First Mode
    2) Read First Mode
    3) No Change Mode

Operating Mode

- **Write First Mode:** the input data is simultaneously written into memory and driven on the data output.
- **Read First Mode:** data previously stored appears on the data output, while the input data is being stored in memory.

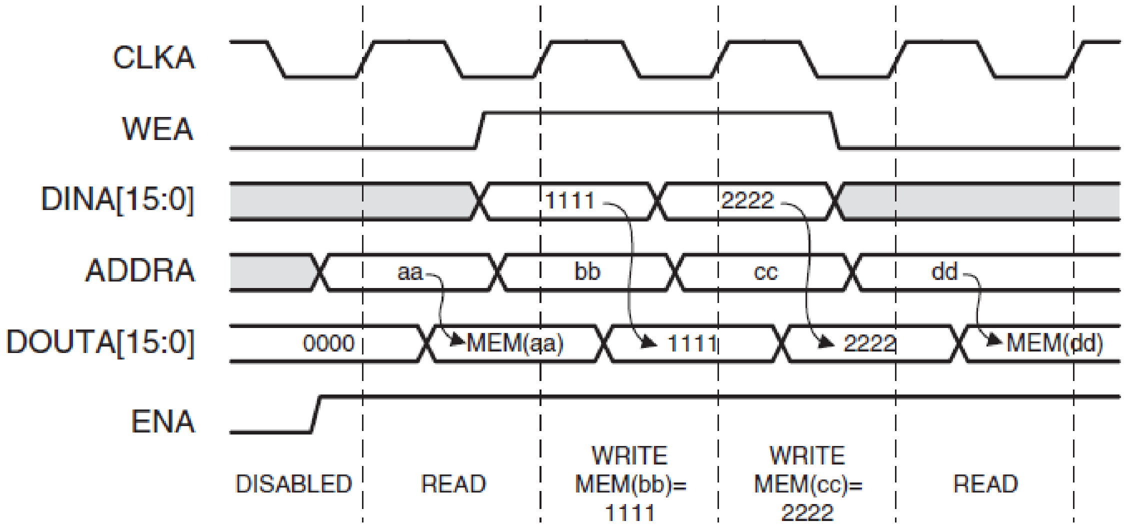


Figure 3-9: **Write First Mode Example**

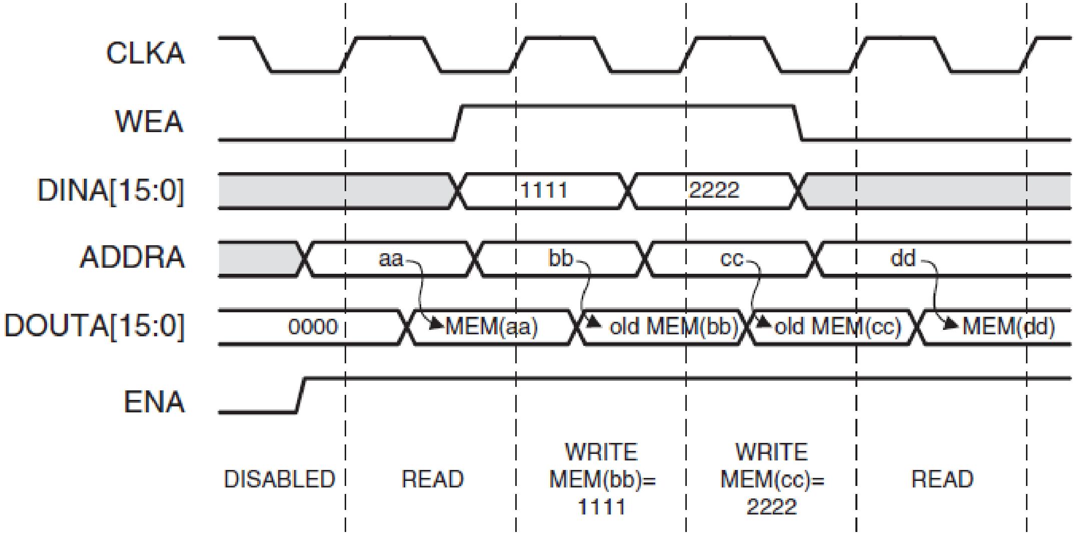


Figure 3-10: **Read First Mode Example**

Operating Mode

- **No Change Mode**: the output latches remain unchanged during a Write operation.

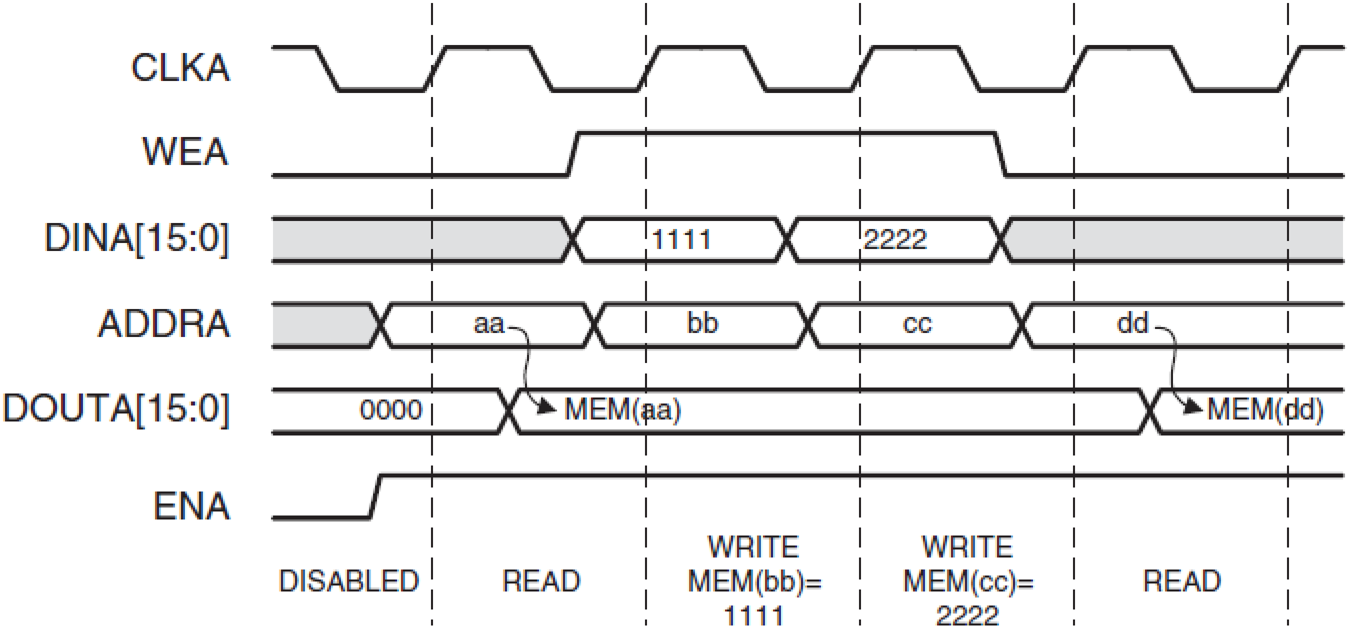
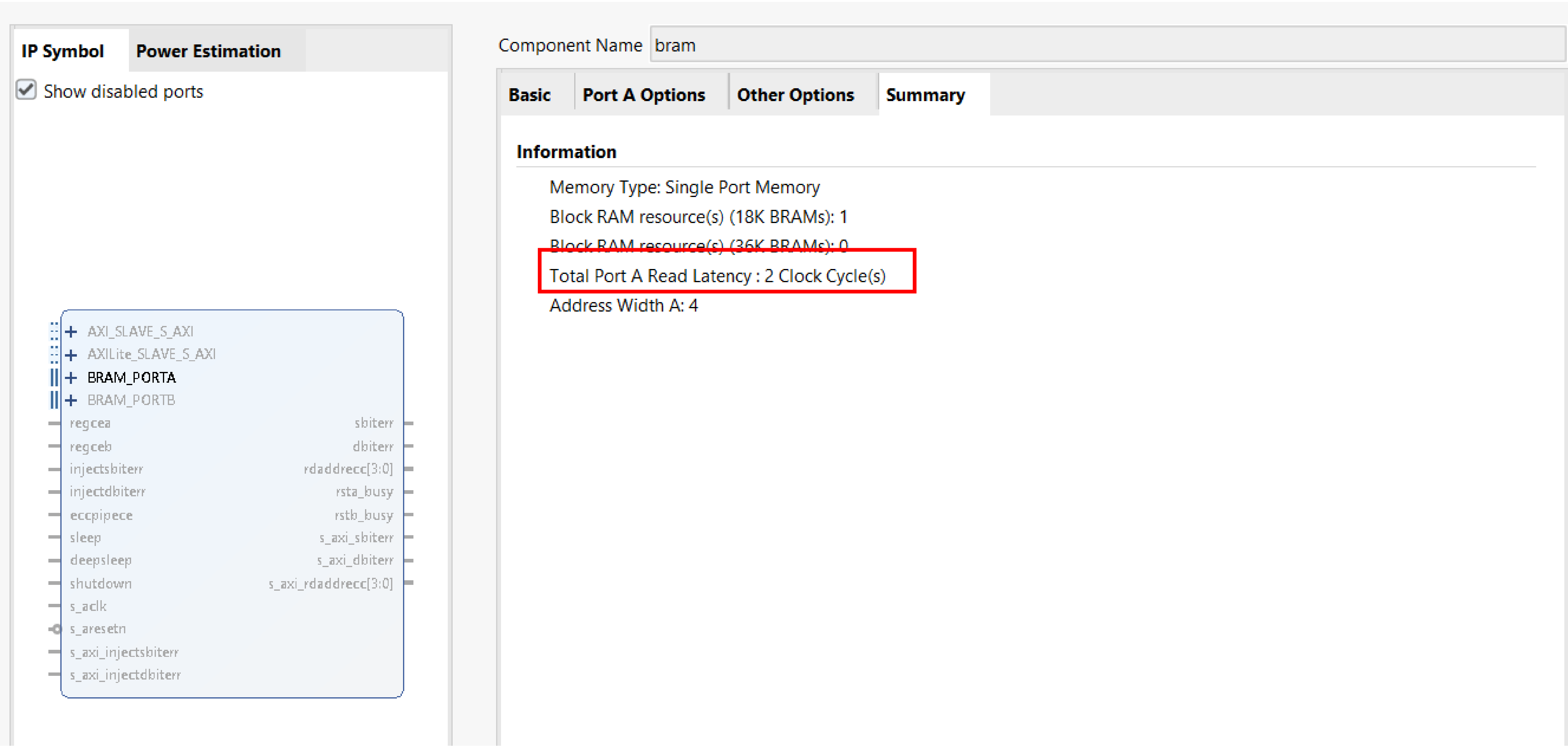


Figure 3-11: **No Change Mode Example**

Example

- From the IP catalog, find the Block Memory Generator.
- Customize and add it with the following specifications:
    - Memory Type: **Single Port RAM**,
    - Algorithm: **Minimum Area**,
    - WriteWidth: **8**,
    - WriteDepth: **8,**
    - OperatingMode: **Write First**,
    - Enable: **Always Enabled**,



Component Name: bram

Information

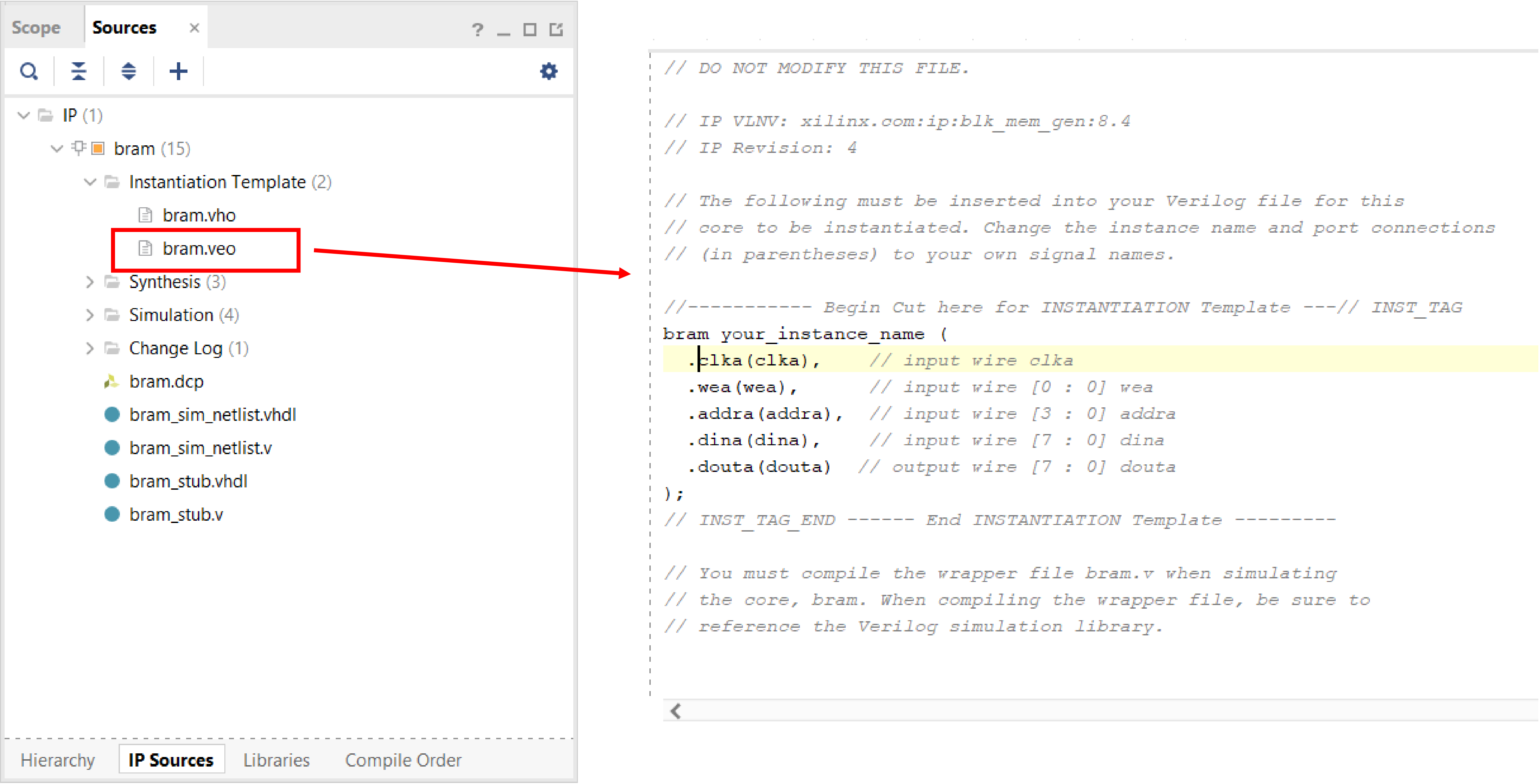
Memory Type: Single Port Memory

Block RAM resource(s) (18K BRAMs): 1

Block RAM resource(s) (36K BRAMs): 0

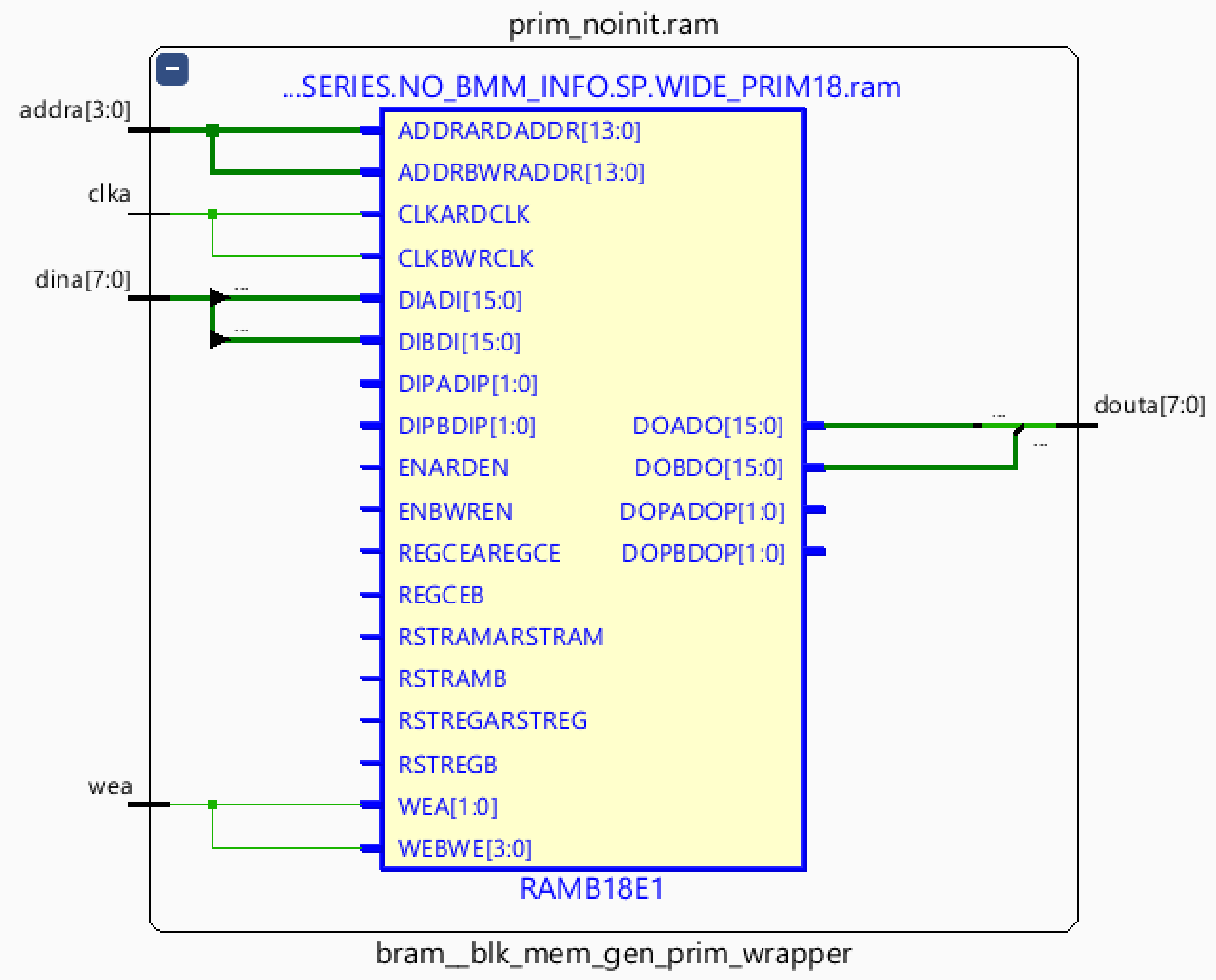
Total Port A Read Latency : 2 Clock Cycle(s)

Address Width A: 4

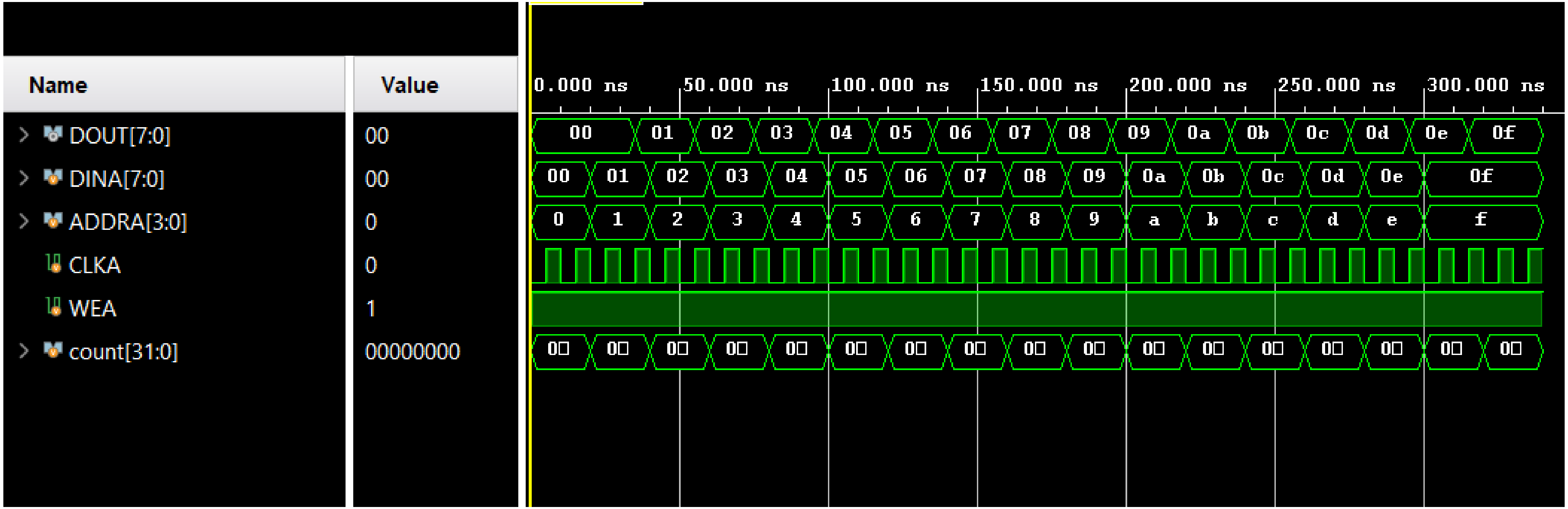


```
// DO NOT MODIFY THIS FILE.

// IP VLNV: xilinx.com:ip:blk_mem_gen:8.4
// IP Revision: 4

// The following must be inserted into your Verilog file for this
// core to be instantiated. Change the instance name and port connections
// (in parentheses) to your own signal names.

//----------- Begin Cut here for INSTANTIATION Template ---// INST_TAG
bram your_instance_name (
  .clka(clka),      // input wire clka
  .wea(wea),        // input wire [0 : 0] wea
  .addra(addra),    // input wire [3 : 0] addra
  .dina(dina),      // input wire [7 : 0] dina
  .douta(douta)    // output wire [7 : 0] douta
);
// INST_TAG_END ------ End INSTANTIATION Template ---------

// You must compile the wrapper file bram.v when simulating
// the core, bram. When compiling the wrapper file, be sure to
// reference the Verilog simulation library.
```

```verilog
module top( dataout, datain, address,
clock, enable );

output [7:0] dataout;
input [7:0] datain;
input [3:0] address;
input clock; input enable;

// instance
bram BLK( .clka(clock), .wea(enable),
          .addra(address), .dina(datain),
          .douta(dataout) );
endmodule
```



```verilog
module stimulus;

wire [7:0] DOUT;
reg [7:0] DINA = 0;
reg [3:0] ADDRA = 0;
reg CLKA; reg WEA = 1; // write enable
integer count = 0;

// instance
bram BLK( .clka(CLKA), .wea(WEA), .addra(ADDRA), .dina(DINA), .douta(DOUT) );

// clock
initial
begin
   CLKA = 0;
   forever #5 CLKA = ~CLKA;
end

initial
   $monitor( $time ,"ADDRA=%b | DOUT=%b", ADDRA, DOUT);

initial
begin
   while( count<16 )
   begin
     DINA = count;
     ADDRA = count;
     #20 count = count + 1;
   end
   #20 $finish;
end
endmodule
```



Name	Value
DOUT[7:0]	00
DINA[7:0]	00
ADDRA[3:0]	0
CLKA	0
WEA	1
count[31:0]	00000000

0.000 ns
50.000 ns
100.000 ns
150.000 ns
200.000 ns
250.000 ns
300.000 ns
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
0 1 2 3 4 5 6 7 8 9 a b c d e f
0□ 0□ 0□ 0□ 0□ 0□ 0□ 0□ 0□ 0□ 0□ 0□ 0□ 0□ 0□ 0□ 0□

References and Further Information

- ➢LogiCORE IP BlockMemory Generator v7.3 Product Guide
- ➢Vivado Design Suite Designing with IP Tutorial