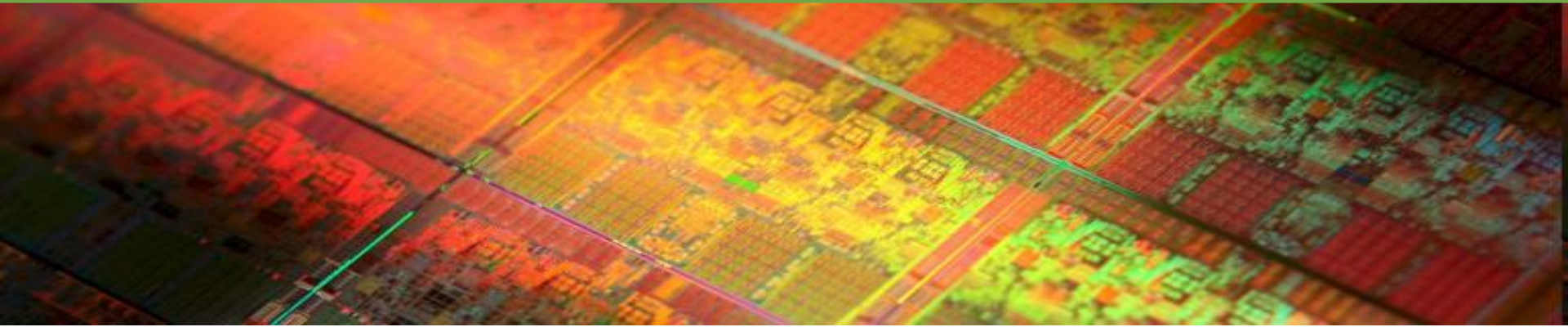


EE 2000

Logic Circuit Design (Fall 2018)



Tutorial 3

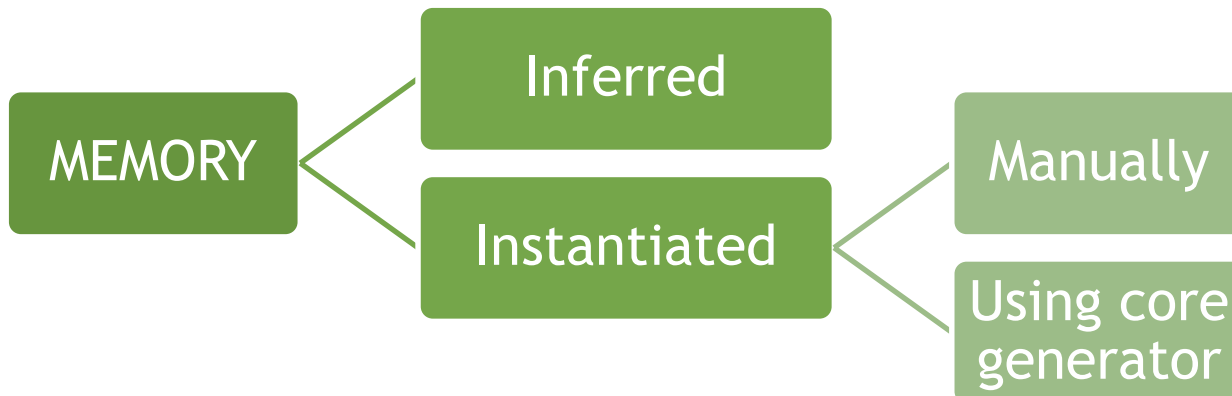
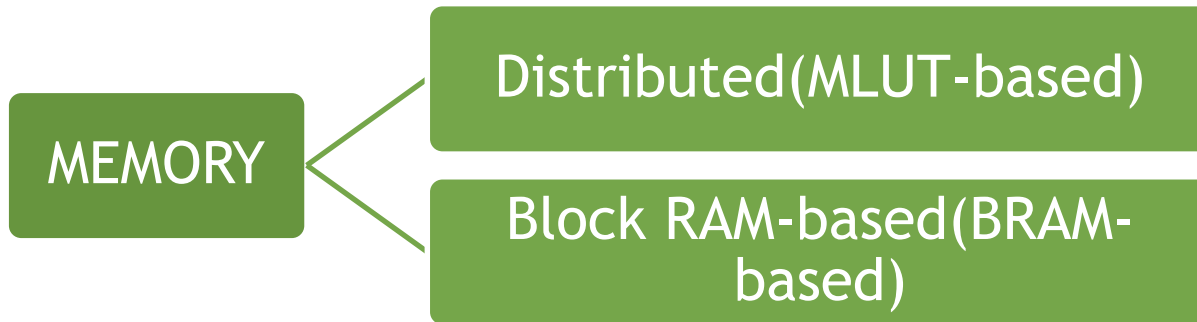
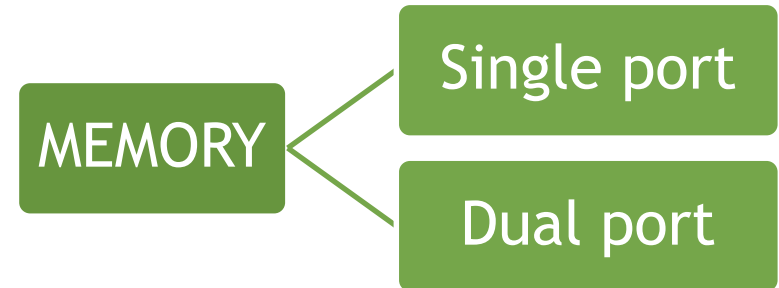
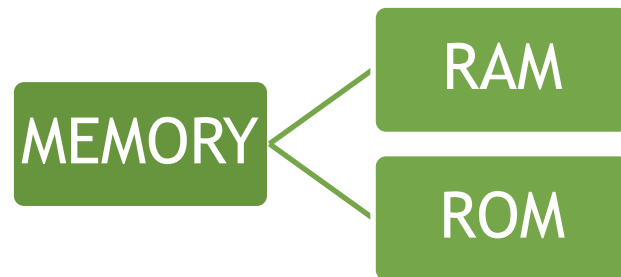
RAM Initiation and Utilization

Instructor: Dr. Ray C.C. Cheung

Objectives

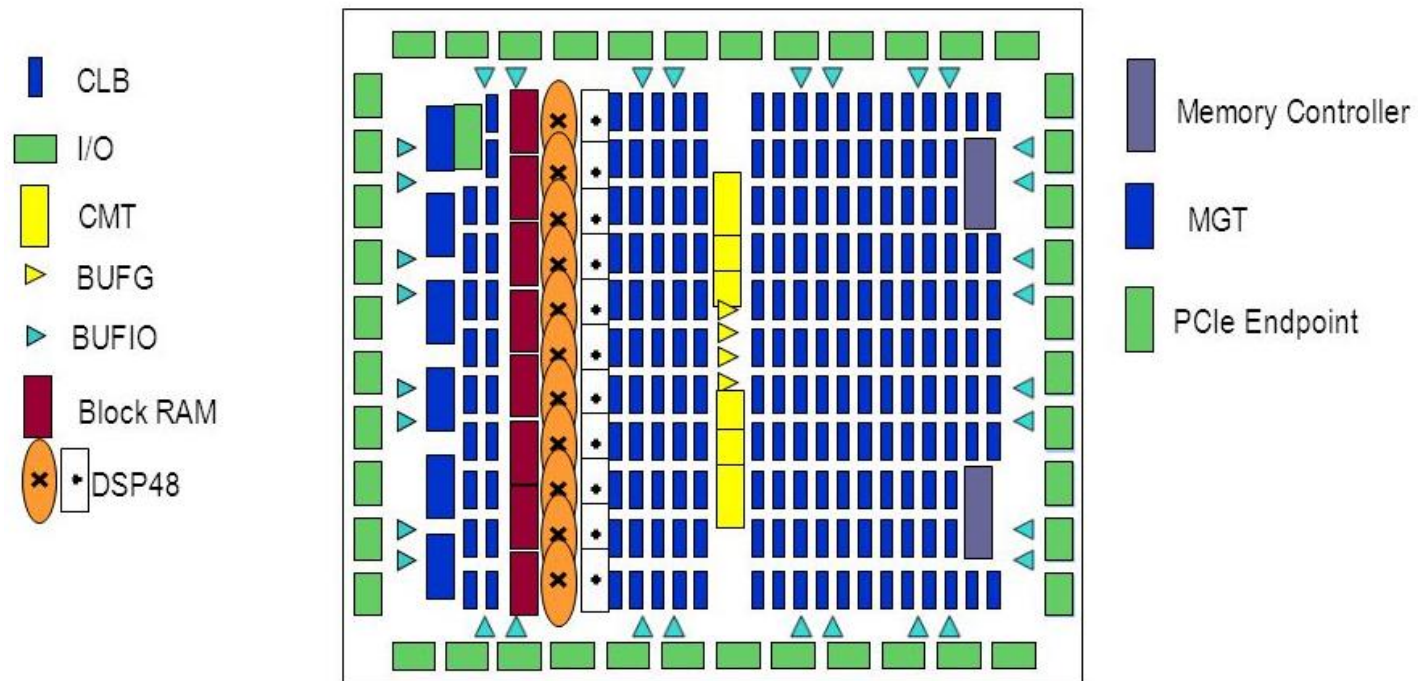
- Learn how to initialize a Block-RAM (BRAM) and a distributed RAM
- Learn how to use the RAM IP
- Using Vivado simulator to observe how RAM works

Memory types



Block RAM(BRAM) vs Distributed RAM

- Block RAM: dedicated memory block
- Distributed RAM: configure LUT in CLB as RAM, 1LUT=16*1 RAM

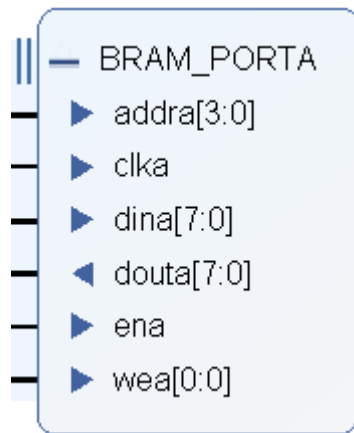


Block RAM(BRAM) vs Distributed RAM (cont'd)

- BRAM: Many blocks, most efficient, not fast as distributed RAM.
- Distributed RAM: fast, but large consumption of LUT will limit the resources to implement your logic design

PART1: Block RAM (BRAM)

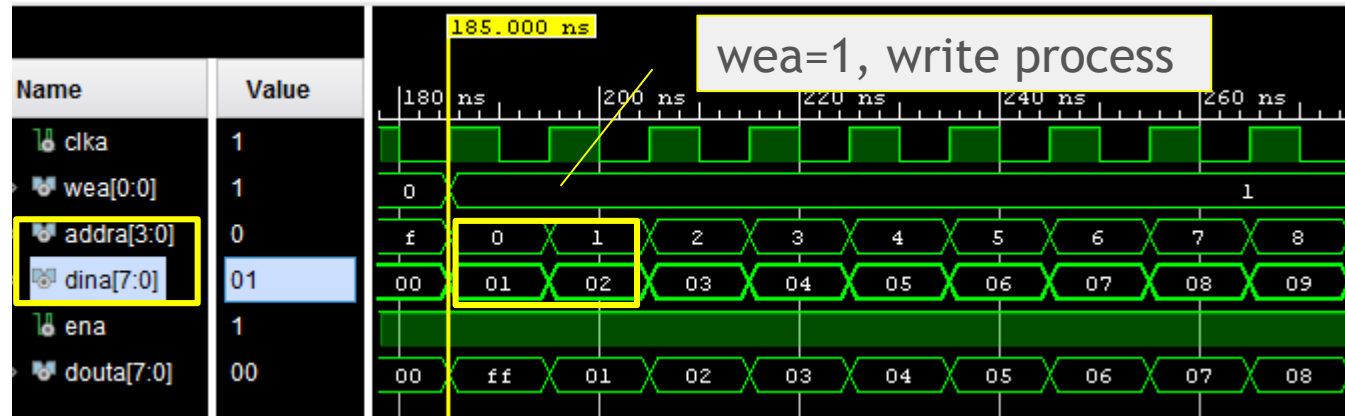
- The diagram of a single port block RAM is as follows.



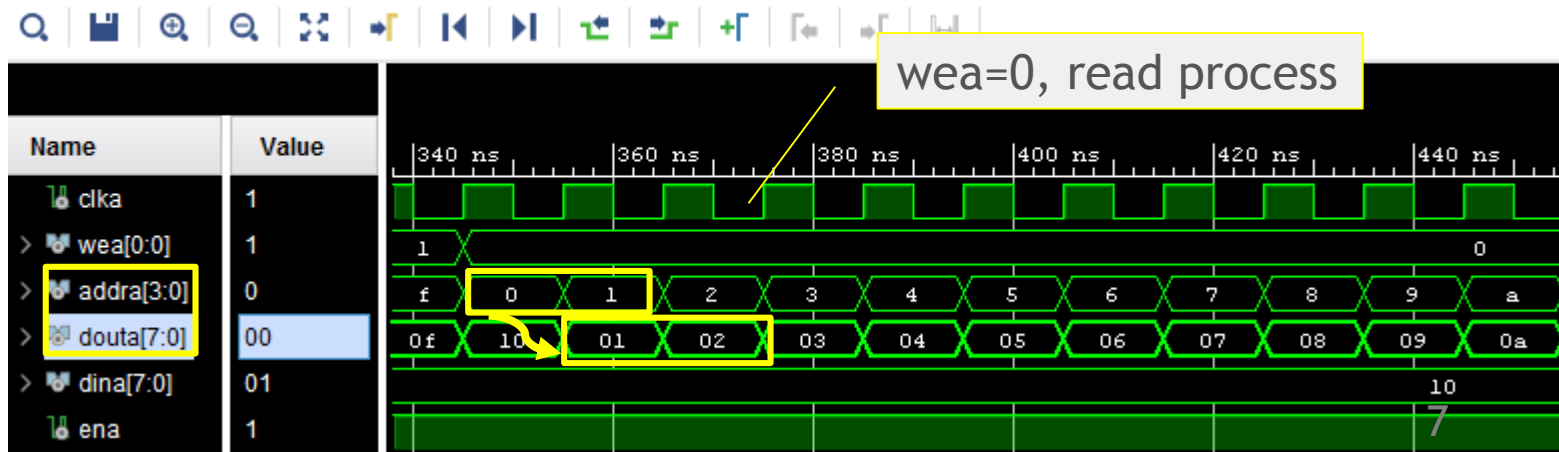
Signal name	Direction	Functionality	Remark
addr[3:0]	input	address	[3:0]: the address bitwidth is 4, i.e. depth of RAM is 16
clka	input	clock	1 bit
dina[7:0]	input	Write data	[7:0]: Data bitwidth is 8
douta[7:0]	output	Read data	[7:0]: Data bitwidth is 8
ena	input	RAM enable	1 bit
wea	input	Write enable	1 bit, wea=1, write; wea=0, read;

Block RAM (BRAM)

Address	Content
0	01
1	02



Untitled 2



Step by step

Initialize new BRAM IP



Instantiate a BRAM



Simulate BRAM IP



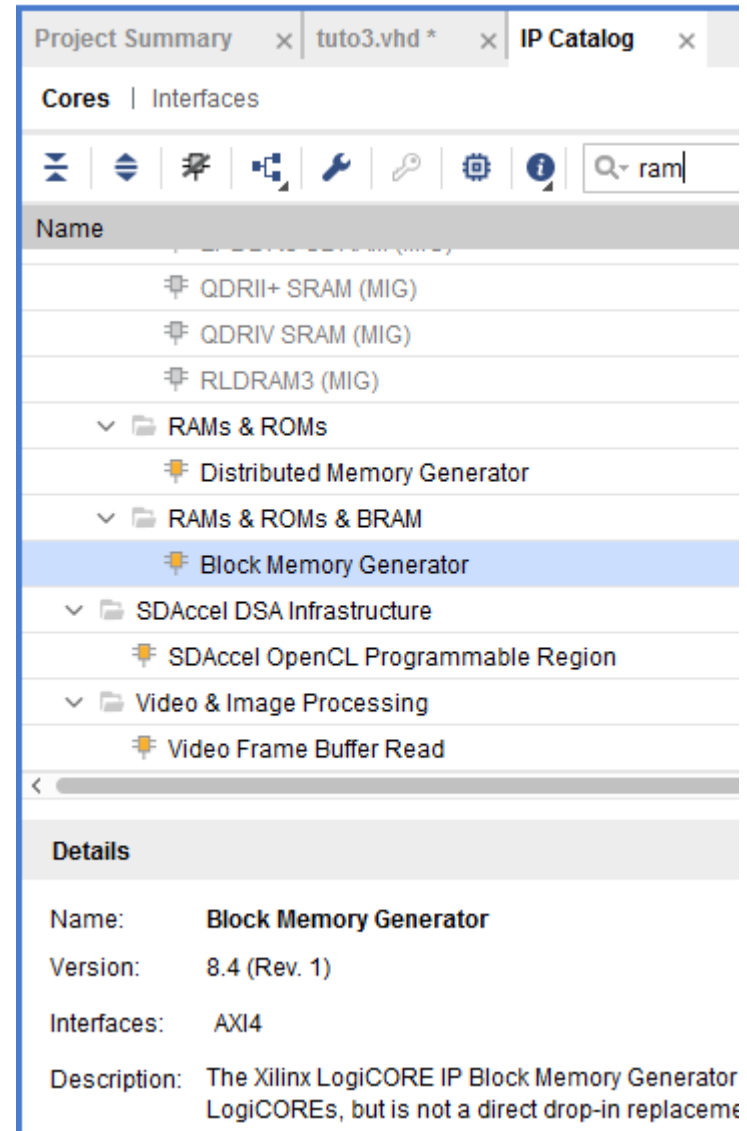
Synthesis and Implementation

Create project and sources

- Create your own project
- In the project manager panel, click **add sources** and create a blank design source as previous tutorial

Initialize a BRAM IP

- In the project manager panel, click **IP catalog**
- In the workspace, search **ram**:
 - choose **Block Memory Generator** to generate **BRAM**
 - Choose **Distributed Memory Generator** to generate distributed RAM



Initialize a BRAM IP(cont'd)

- Click Port A options panel, modify write width to 8 bits.

The screenshot shows the 'Block Memory Generator (8.4)' window with the 'Port A Options' tab selected. The 'Component Name' is 'blk_mem_gen_0'. The 'Memory Size' section shows 'Write Width' set to 8, 'Read Width' set to 8, 'Write Depth' set to 16, and 'Read Depth' set to 16. The 'Operating Mode' is 'Write First' and 'Enable Port Type' is 'Use ENA Pin'. The 'Port A Optional Output Registers' section has 'Primitives Output Register' checked. The 'Port A Output Reset Options' section has 'RSTA Pin (set/reset pin)' and 'Reset Memory Latch' unchecked, with 'Output Reset Value (Hex)' set to 0 and 'Reset Priority' set to 'CE (Latch or Register Enable)'. The 'READ Address Change A' section has 'Read Address Change A' unchecked. A button labeled '+ BRAM_PORTA' is visible on the left side of the window.

Customize IP

Block Memory Generator (8.4)

Documentation IP Location Switch to Defaults

IP Symbol Power Estimation

Show disabled ports

Component Name blk_mem_gen_0

Basic Port A Options Other Options Summary

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 Use ENA Pin

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

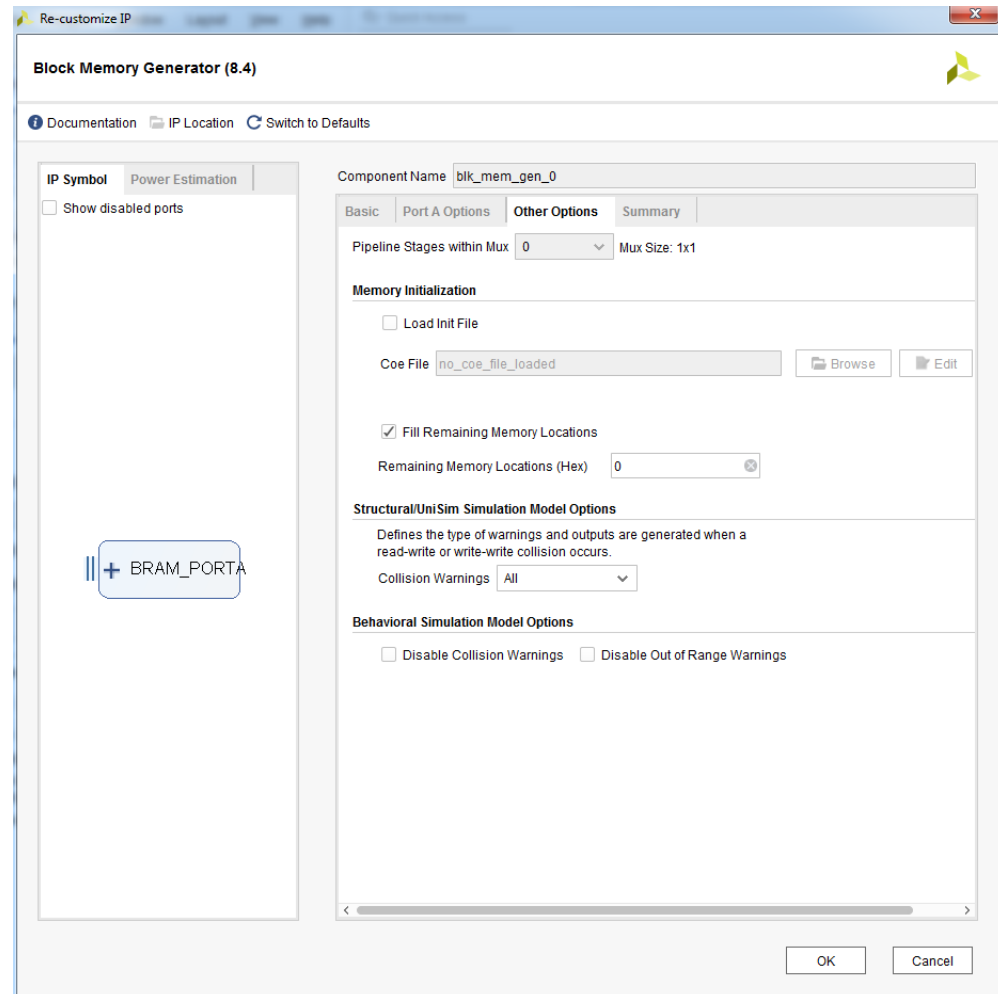
☐ Read Address Change A

+ BRAM_PORTA

OK Cancel

Initialize a BRAM IP (cont'd)

- In other options panel,
 - Choose **Load Init File**, and use a coe file to specify the initial value of the BRAM.
 - And/or choose **Fill remaining memory locations** and set the default initial value.



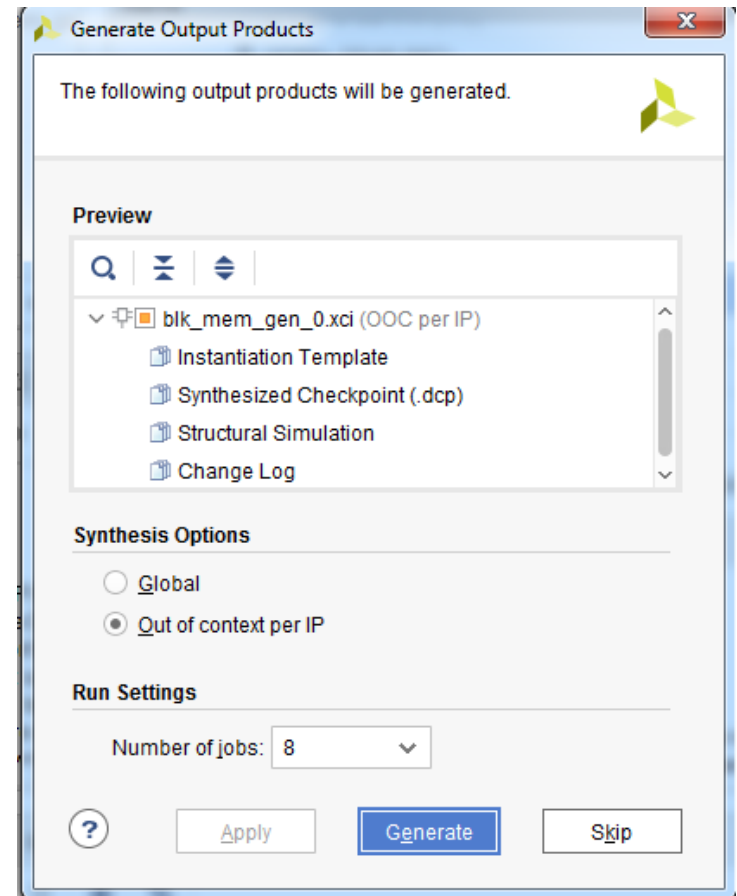
Initialize a BRAM IP (cont'd)

- Example of a coe file initiate a memory of depth=16, and width=8 using hexadecimal

```
memory_initialization_radix=16;  
memory_initialization_vector=  
ff,  
ab,  
f0,  
11,  
11,  
00,  
01,  
aa,  
bb,  
cc,  
dd,  
ef,  
ee,  
ff,  
00,|
```

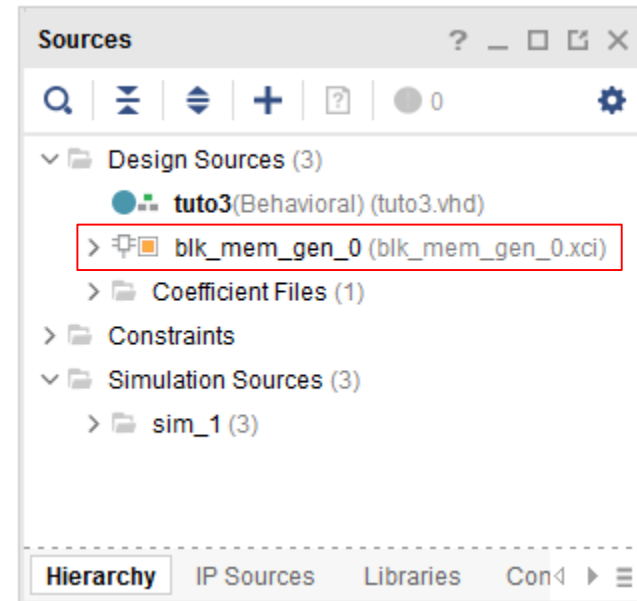
Initialize a BRAM IP (cont'd)

- Click **OK** and **Generate**
- Synthesis option:
 - **out of context per IP:** the IP synthesis is run in out of context mode during IP generation itself. You will find synthesis DCP in IP location.
 - **Global:** IP synthesis will not be run during IP generation. The IP will be synthesized during top level synthesis along with other RTL files



Initialize a RAM IP (cont'd)

- In sources panel, you can find the new generated block RAM as circled.
- blk_mem_gen_0 is the reference name of the BRAM we generate before



Instantiate a BRAM IP

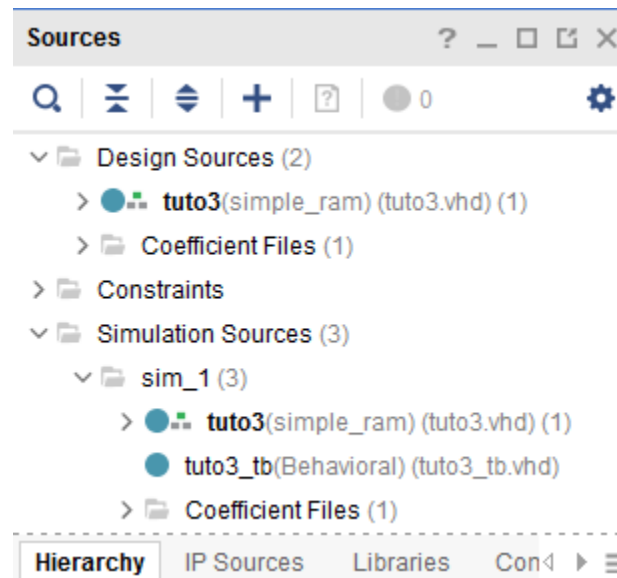
- Go back to the design source file, use the circled statements to declare a block ram as a component and instantiate a block ram.

```
entity tuto3 is
port (
  clka : IN STD_LOGIC;
  ena : IN STD_LOGIC;
  wea : IN STD_LOGIC_VECTOR(0 DOWNTO 0);
  addra : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
  dina : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
  douta : OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
end entity tuto3;

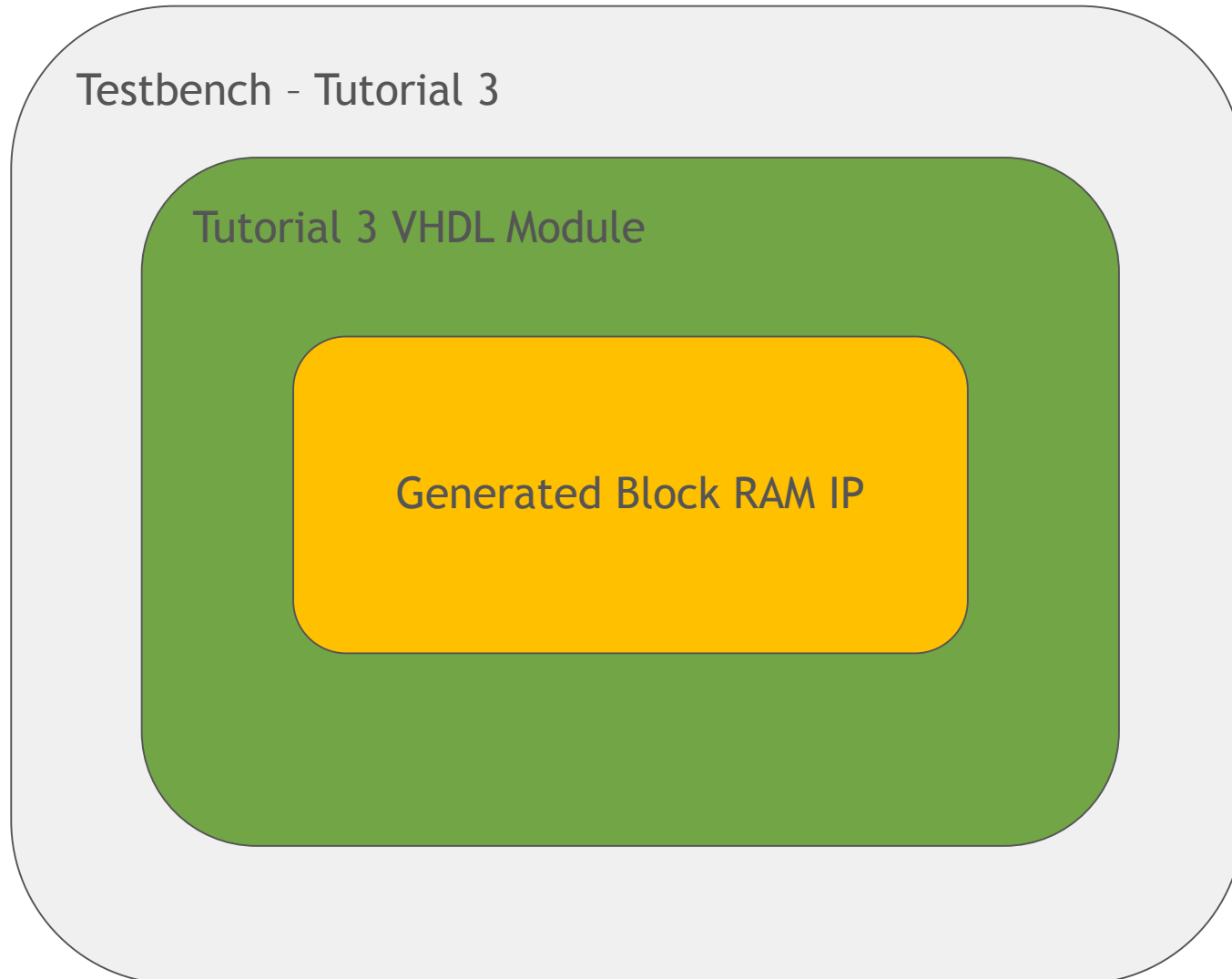
architecture simple_ram of tuto3 is
  component blk_mem_gen_0
    PORT (
      clka : IN STD_LOGIC;
      ena : IN STD_LOGIC;
      wea : IN STD_LOGIC_VECTOR(0 DOWNTO 0);
      addra : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
      dina : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
      douta : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
    );
  end component;
begin
  U1: blk_mem_gen_0 port map (clka,ena,wea,addra,dina,douta);
end simple_ram;
```

Simulate BRAM IP

- In project manager panel, click **add sources** -> choose **add or create simulation sources** -> **next** -> choose **create file** -> specify file name as “tuto3_tb” -> click **ok** and finish
- In **sources** panel, expand **sim_1**, double click **tuto3_tb**



VHDL Module Hierarchy



Simulate BRAM IP (cont'd)

- In tuto3_tb, edit your testbench as follows

```
library IEEE;  
use IEEE.STD_LOGIC_1164.ALL;  
use ieee.std_logic_unsigned.all;  
entity tuto3_tb is  
end tuto3_tb;
```

Include this package to make std_logic_vector support add operation

```
architecture Behavioral of tuto3_tb is  
  component tuto3 is  
    PORT (  
      clka : IN STD_LOGIC;  
      ena : IN STD_LOGIC;  
      wea : IN STD_LOGIC_VECTOR(0 DOWNTO 0);  
      addra : IN STD_LOGIC_VECTOR(3 DOWNTO 0);  
      dina : IN STD_LOGIC_VECTOR(7 DOWNTO 0);  
      douta : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)  
    );  
  end component;
```

Declare our design module tuto3 as a component

```
signal clka : std_logic := '0';  
signal dina : std_logic_vector(7 downto 0) := (others => '0');  
signal addra : std_logic_vector(3 downto 0) := (others => '0');  
signal wea : STD_LOGIC_VECTOR(0 DOWNTO 0) := (others => '0');  
signal ena : std_logic := '0';  
signal douta : std_logic_vector(7 downto 0) := (others => '0');
```

Declare all the signal that drive the input ports of tuto3 and the signal connect to the output port of tuto3

Simulate BRAM IP (cont'd)

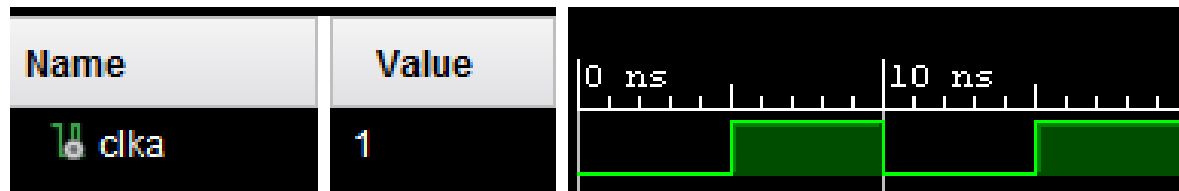
- (cont'd)

```
tuto : tuto3 port map (clka,ena,wea,addra,dina,douta);
```

Instantiate tuto3
design module

```
clkPro : process --100MHz  
begin  
    clka <= '0';  
    wait for 5ns;  
    clka <= '1';  
    wait for 5ns;  
end process;
```

Clock generate process

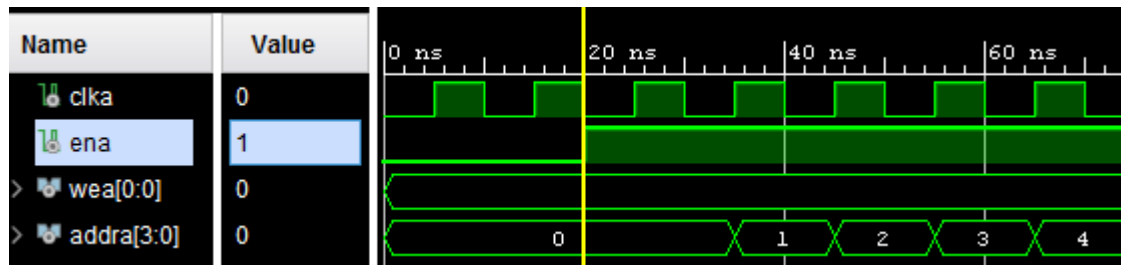


Simulate BRAM IP (cont'd)

```
simuliPro : process
begin
--initialization
  dina<=(others => '0');
  addra<=(others => '0');
  wea<=(others => '0');
  wait for 20ns;
--enable block ram
  ena<='1';
--read process
  wait until rising_edge(clka);
  wea<=(others => '0'); --read
  addra<=(others => '0');--read address 0

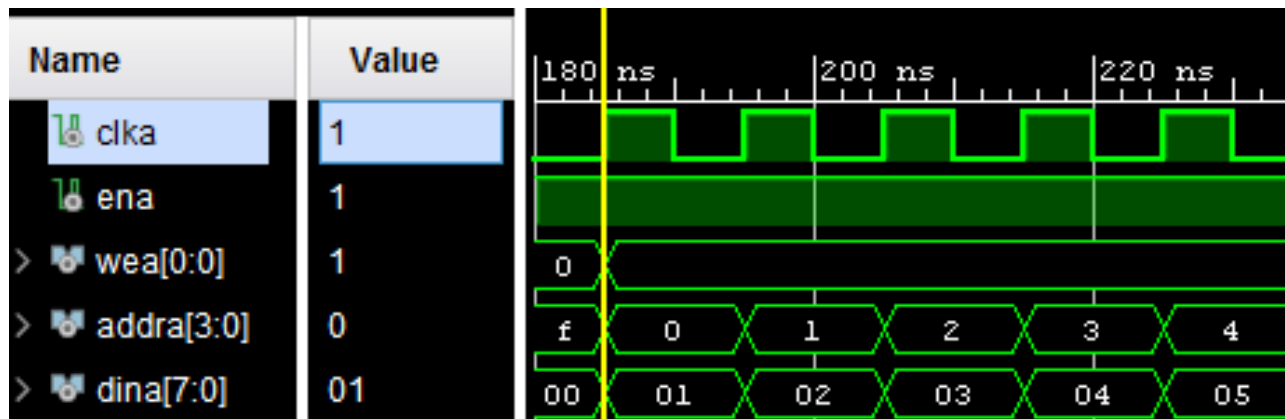
  for I in 1 to 15 loop
    wait until rising_edge(clka);
    addra <= addra + '1';--loop other address
  end loop;
```

Stimuli generate process,
Assign proper value to all
the drive signal in different
time point



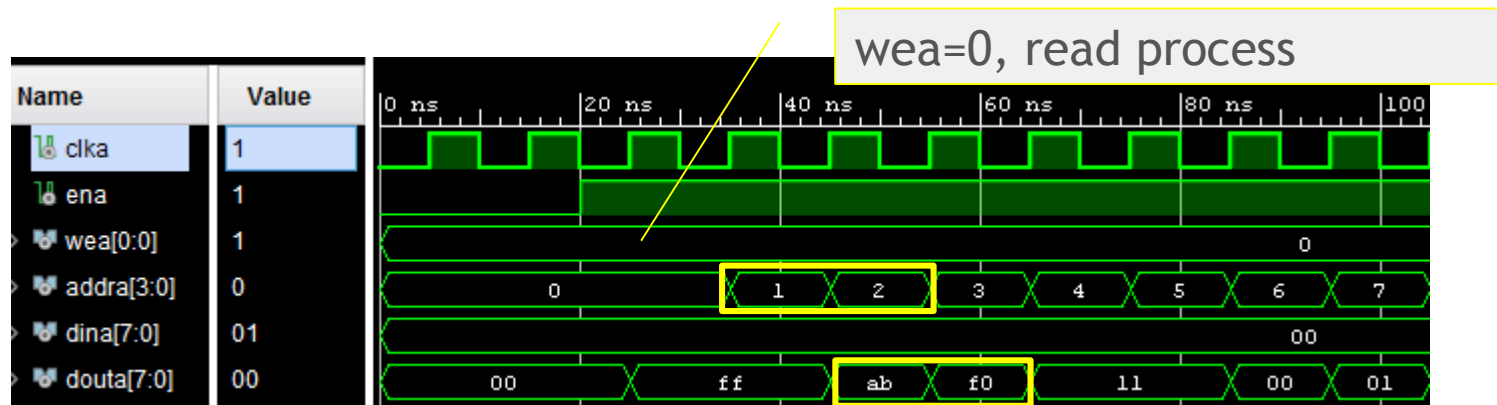
Simulate BRAM IP (cont'd)

```
--write process
  wait until rising_edge(clka);
  dina<=(0=>'1', others => '0');--data input '00000001'
  addra<=(others => '0');--write to address 0
  wea<=(others => '1');--write
  for I in 1 to 15 loop
    wait until rising_edge(clka);
    wea <= (others => '1');
    addra <= addra + '1';--write to other address,1,2,3...
    dina <= dina + '1';--assign different input data,2,3,4...
  end loop;
```



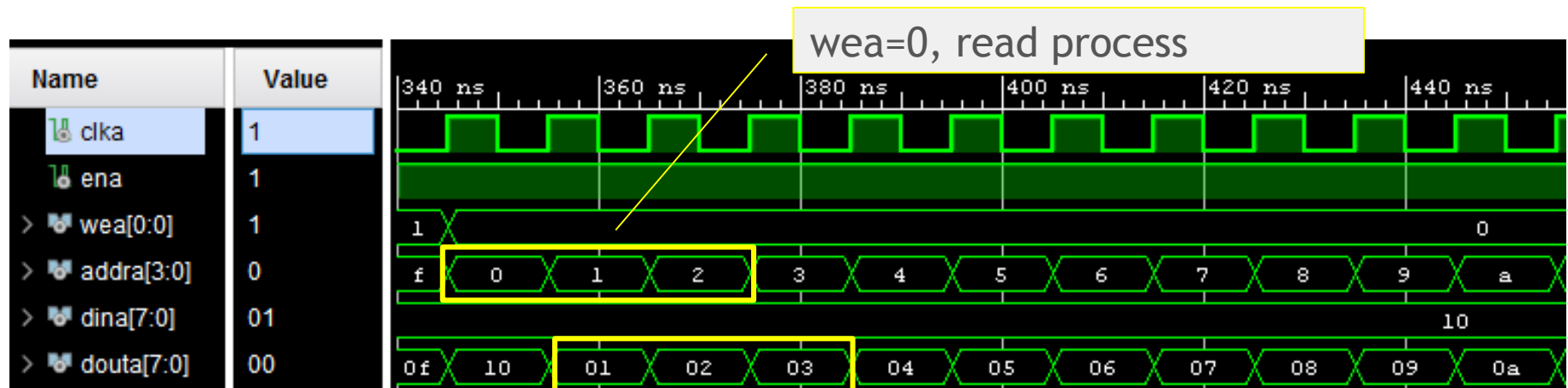
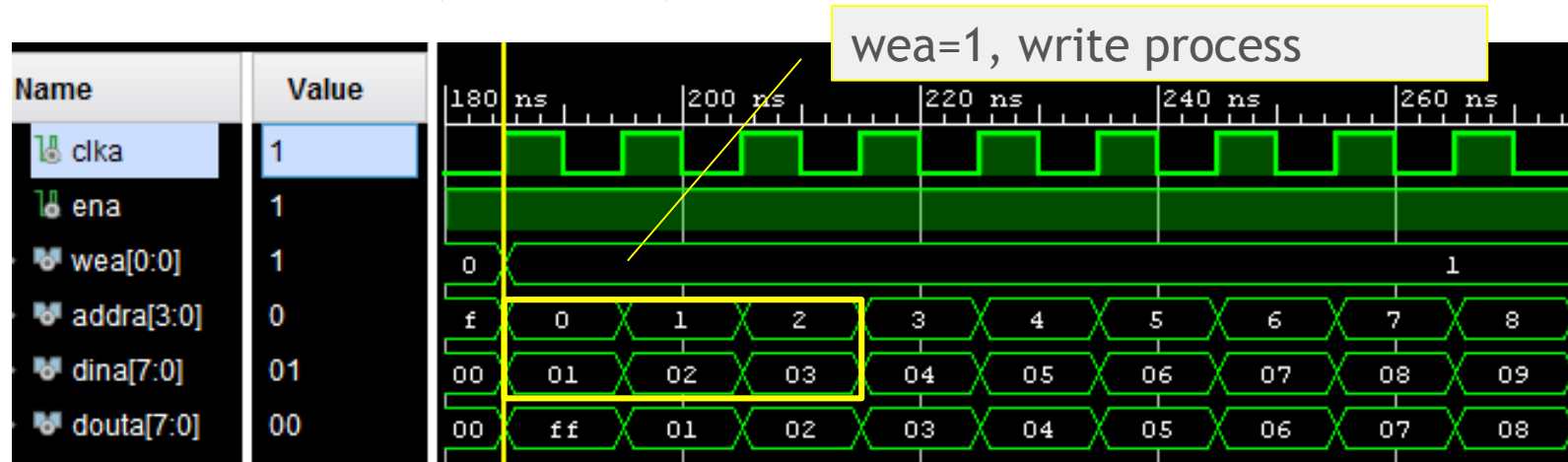
Simulate BRAM IP (cont'd)

- In project manager panel, click **Run Simulation** -> choose **Run Behavioral Simulation**
- After simulation completed, in the **waveform window**, **zoom the waveform to appropriate level**.



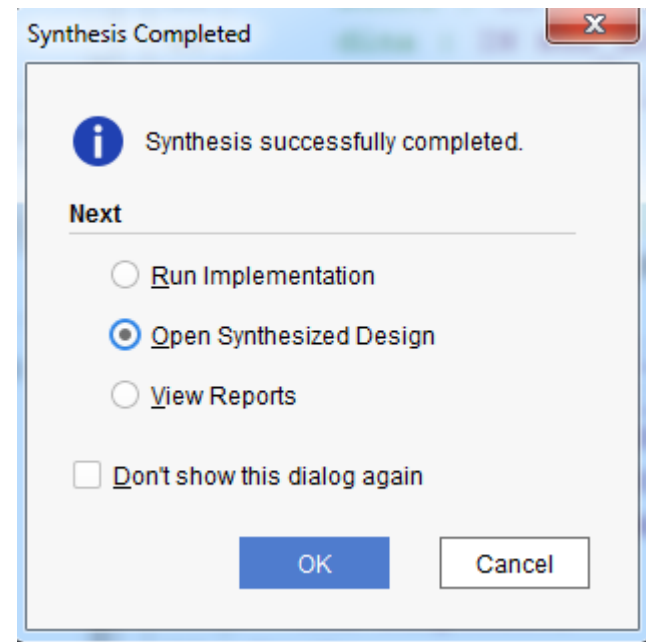
Simulate BRAM IP (cont'd)

- Waveform (cont'd)



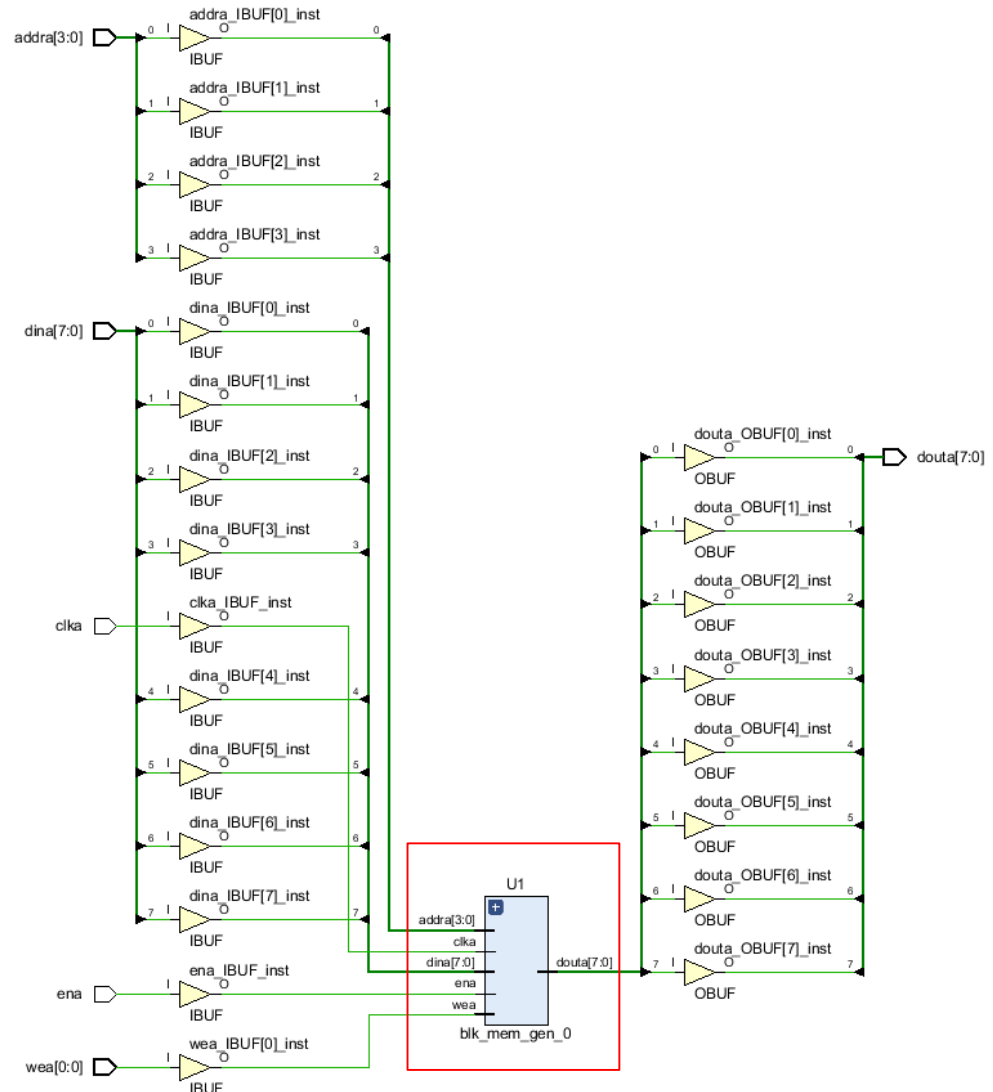
Run synthesis

- In project manager panel, click run synthesis.
- After the process completed, choose open synthesized design.



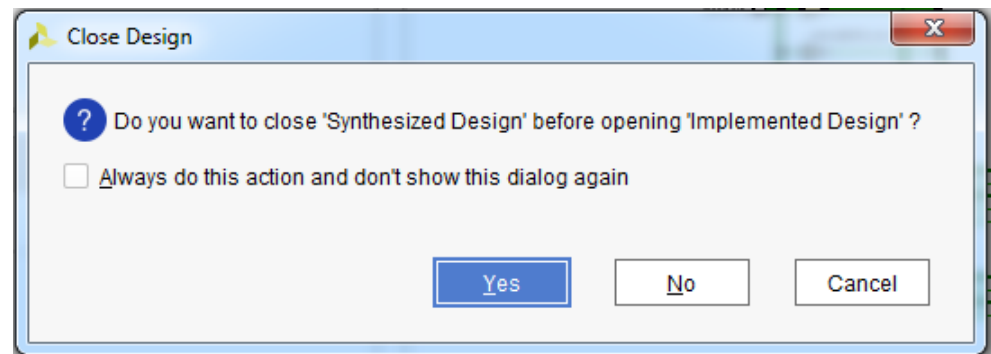
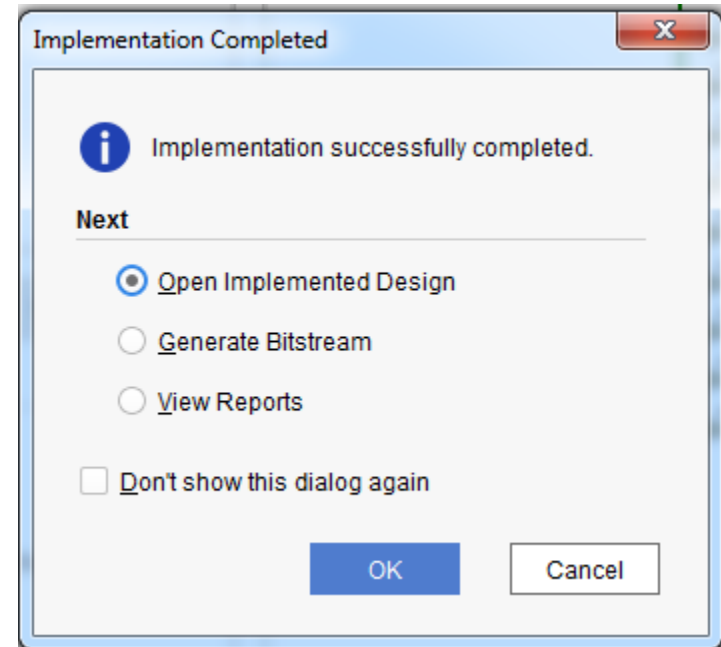
Run synthesis(cont'd)

- In the workspace, we can find the schematic.



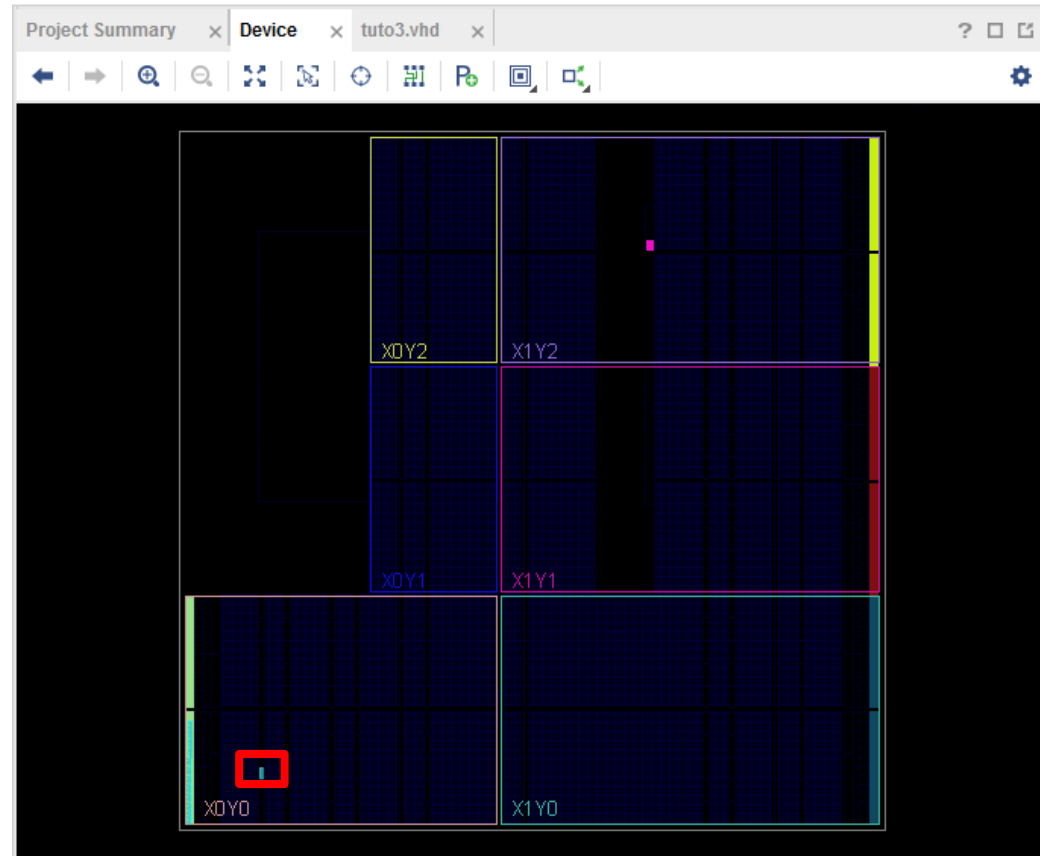
Run implementation

- In project manager panel, click run implementation.
- After it completes, choose open implementation design.
- Choose yes to close the previous opened synthesis design.



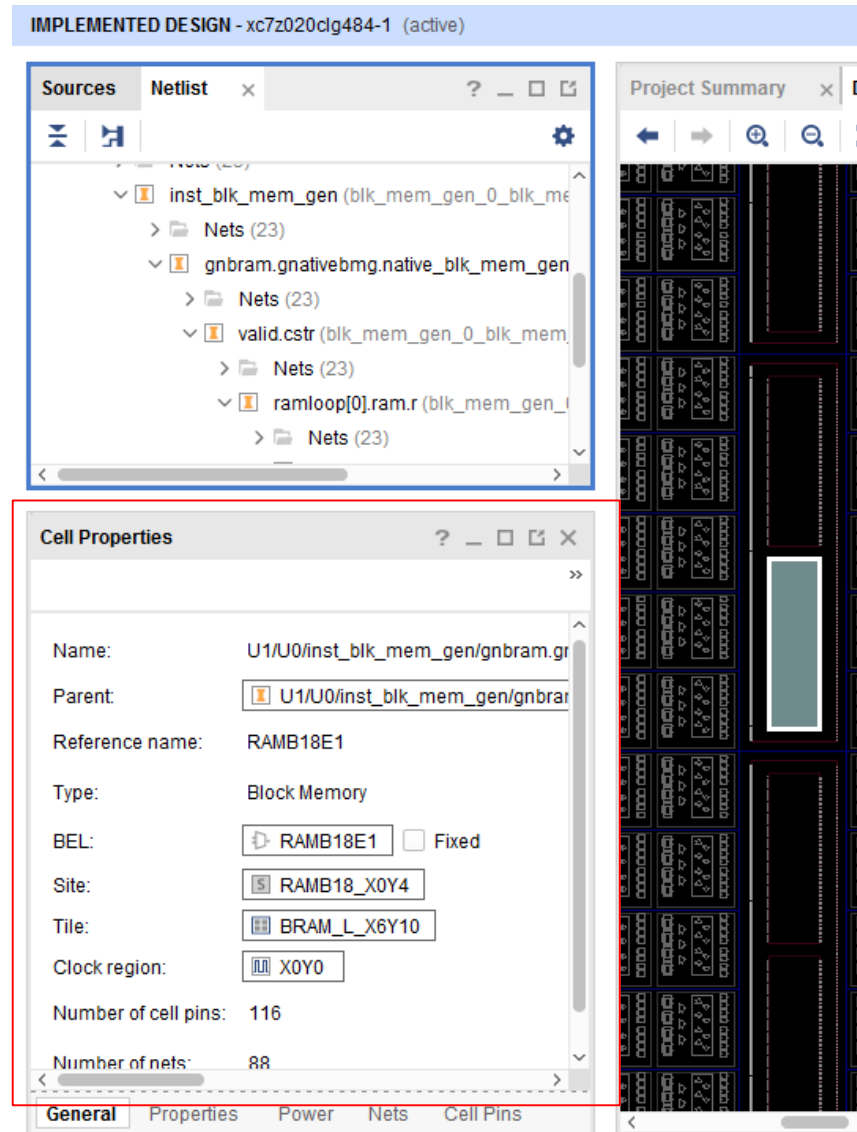
Run implementation (cont'd)

- In the device window, we can find a blue spot



Run implementation (cont'd)

- Zoom in on the blue spot, and click it. We can find **Cell properties** panel on the left side of the schematic.
- We can find that we use half a BRAM



Run implementation (cont'd)

- In the workspace, choose Project Summary option pane -> roll down and choose Table as circled
- In the utilization report, we can find we use 0.5 BRAM and no LUT.

IMPLEMENTED DESIGN - xc7z020clg484-1 (active)

Sources Netl x ? _ □ □

inst_blk_mem_c
Nets (23)
gnbram.gna
Nets (23)
valid.cst
Nets
ram

Cell Pro ? _ □ □ x

Name: U1/U0/i
Parent: U1/
Reference name: RAMB1
Type: Block M
BEL: RA
Site: RA
Tile: BR
Clock region: XOY
Number of cell pins: 116
Number of nets: 88

General Properties

Project Summary x Device x tuto3.vhd x

Status: Complete
Messages: 189 warnings
Active run: synth_1
Part: xc7z020clg484-1
Strategy: Vivado Synthesis Defaults
Report Strategy: Vivado Synthesis Default Reports

DRC Violations

Summary: 2 critical warnings
1 warning
Implemented DRC Report

Utilization Post-Synthesis | Post-Implementation

Graph | Table

Resource	Utilization	Available	Utilization ...
BRAM	0.50	140	0.36
IO	23	200	11.50

Reference

- Design a Block RAM Memory in IP Integrator in Vivado
 - <https://www.youtube.com/watch?v=gfpE81yMBwQ>
- Block Memory Generator, Xilinx
 - https://www.xilinx.com/products/intellectual-property/block_memory_generator.html#documentation

Tutorial - Remarks

- Please raise your hand if you have questions!
 - 9 course tutors + course Instructors
 - If you finish the tutorial quickly, please try different settings in the Vivado tool.
- You can also refer to the course Canvas homepage.
- Get familiar with the Design tool.
- Explore the options in the waveform window.
- Explore how Dual-port RAM works.