

Block RAM

2013 – 2014 - 2020

Ando Ki, Ph.D.
(adki@future-ds.com)

Agenda

- ❑ On-Chip and On-board Memories
- ❑ Memories for FPGA
- ❑ Distributed RAM: Xilinx FPGA LUT as a memory
- ❑ Distributed RAM using LUT
- ❑ BRAM for Xilinx Spartan-3
- ❑ Block RAM Waveforms
- ❑ Vivado IP manager
- ❑ Bram interface types
- ❑ Generated files
- ❑ Example: memory using BRAM
- ❑ How to fill contents
- ❑ Xilinx BRAM coefficient file
- ❑ Example: memory with initial value
- ❑ Projects
- ❑ References

On-Chip and On-board Memories

On-board memories (off-chip memories)

- ◆ Volatile memory
- ◆ Non-volatile memory
- ◆ Static memory
- ◆ Dynamic memory
- ◆ Synchronous memory
- ◆ Asynchronous memory
- ◆ SDR (single data rate) memory
- ◆ DDR (double data rate) memory
- ◆ DDR2, DDR3

On-chip memories

- ◆ RAM
- ◆ ROM
- ◆ OTP (One-Time Programmable)

Memories for FPGA

FPGA normally has two types of memory elements.

- ◆ Distributed RAM takes the form of flip-flops that are created within the general-purpose logic fabric of the chip.
 - ◆ This is good for storing small amounts of data, making registers, shift registers, etc. Block RAM is dedicated, configurable memory with address, data, and control ports.
- ◆ BRAM (Block RAM)

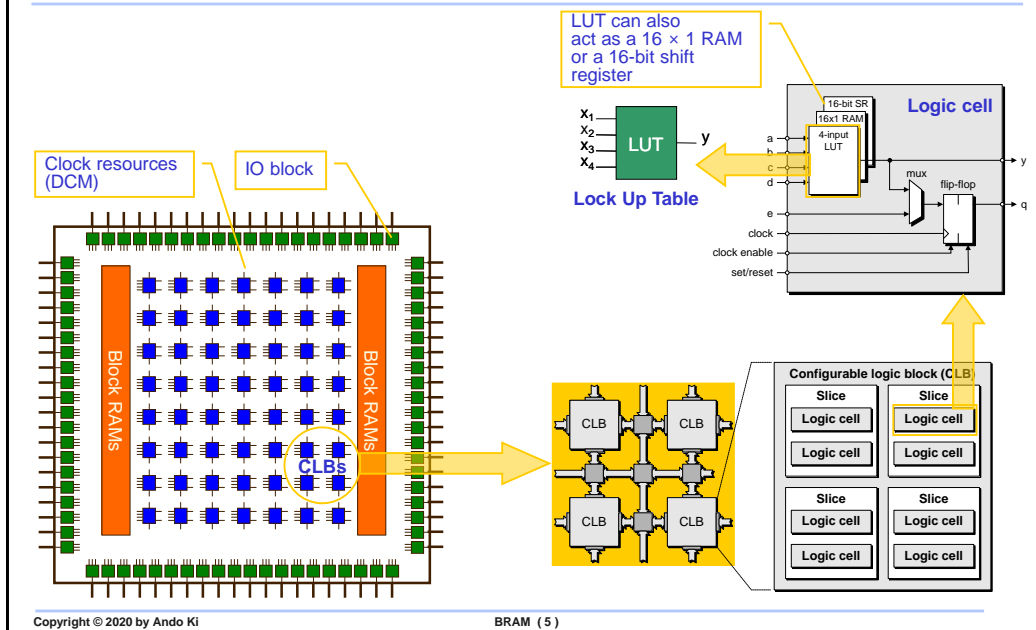
Good things of BRAM

- ◆ Fast,
- ◆ Contents can be initialized at mapping time

Bad things of BRAM

- ◆ Small of size

Distributed RAM: Xilinx FPGA LUT as a memory



Distributed RAM using LUT

CLB LUT configurable as Distributed RAM

- ◆ An LUT equals 16×1 RAM
- ◆ Cascade LUTs to increase RAM size

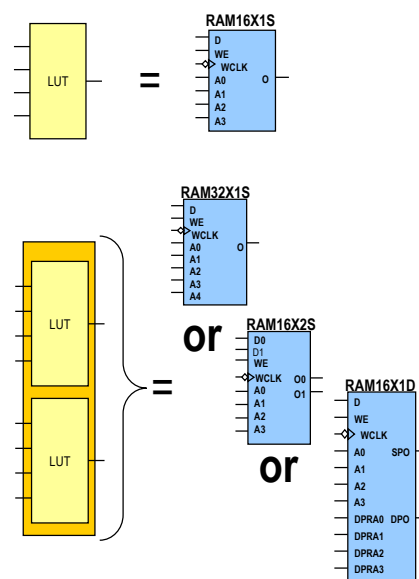
Synchronous write

Asynchronous read

- ◆ Can create a synchronous read by using extra flip-flops
- ◆ Naturally, distributed RAM read is asynchronous

Two LUTs can make

- ◆ 32×1 single-port RAM
- ◆ 16×2 single-port RAM
- ◆ 16×1 dual-port RAM



BRAM for Xilinx Spartan-3

- Xilinx Spartan-3 has multiple block RAM memories, organized in columns.
- Each block RAM contains 18,432 bits (18kbit) of fast static RAM, 16K bits of which is allocated to data storage (i.e., 2K byte) and, in some memory configurations, an additional 2K bits allocated to parity.
 - BRAM can be instantiated using the appropriate "RAMB16" module from the Xilinx design library: RAM16_S9
 - XC3S1000: 24 BRAMs (432Kbits)
 - XC3S2000: 40 BRAMs (720Kbits = 40x18Kbit)
 - XC3S5000: 104 BRAMs (1,827Kbits)

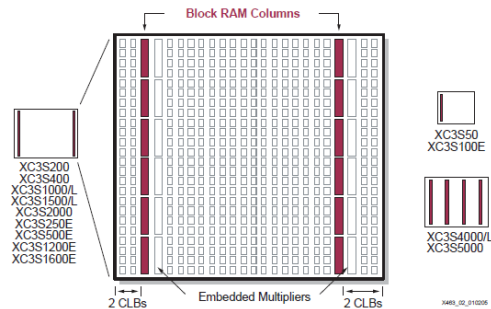


Figure 2: Block RAMs Arranged in Columns with Detailed Floorplan of XC3S200

BRAM for Xilinx Spartan-3

- BRAM is true dual-port memory, but can be configured to single-port memory.

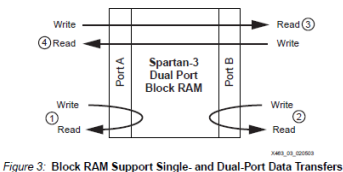
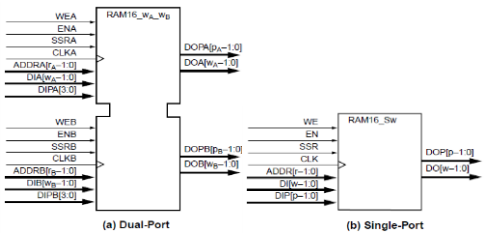


Figure 3: Block RAM Support Single- and Dual-Port Data Transfers



BRAM for Xilinx Spartan-3

- ❑ Cascade multiple BRAMs to create deeper and wider memory organizations
- ❑ A single BRAM can be used different ways as follows.

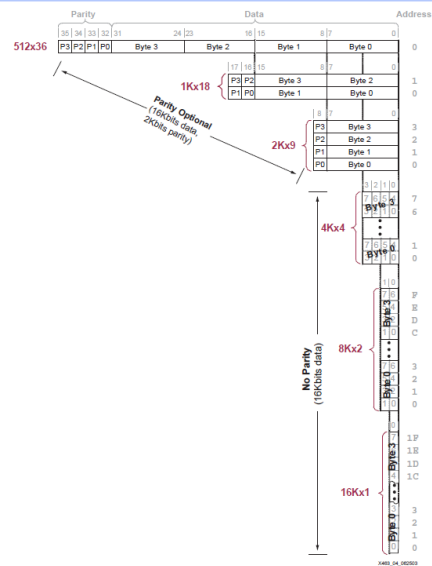


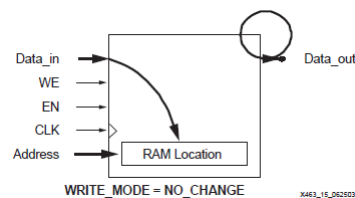
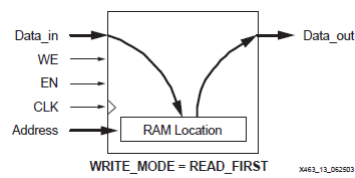
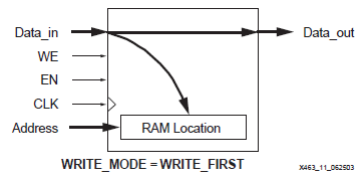
Figure 4: Data Organization and Mapping Between Modes

Copyright © 2020 by Ando Ki

BRAM (9)

BRAM for Xilinx Spartan-3

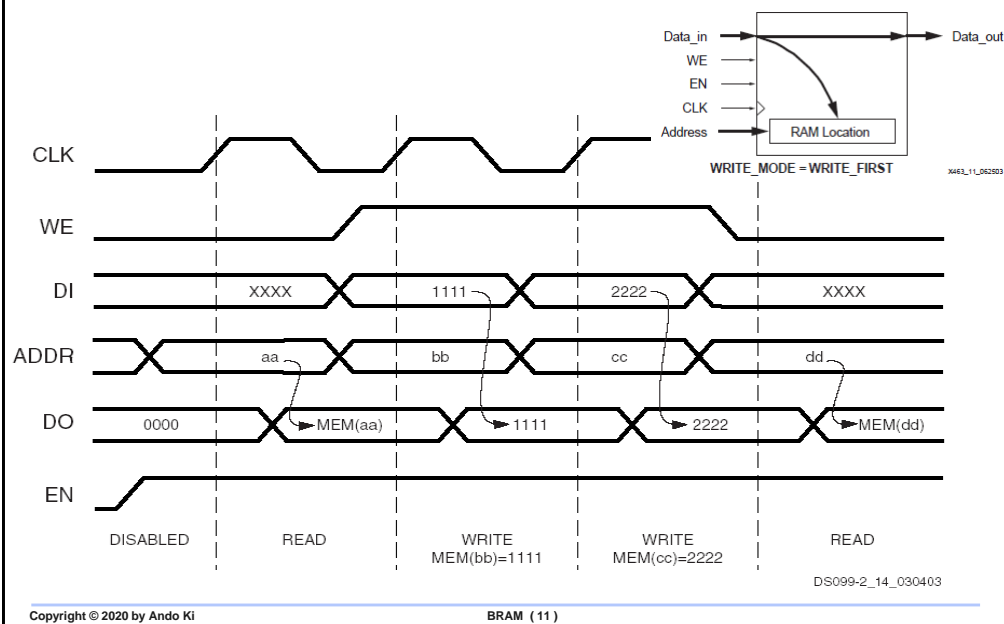
- ❑ Write mode
 - ◆ WRITE_FIRST: Read After Write (Default)
 - ◆ READ_FIRST: Read Before Write
 - ◆ NO_CHANGE: No Read on Write



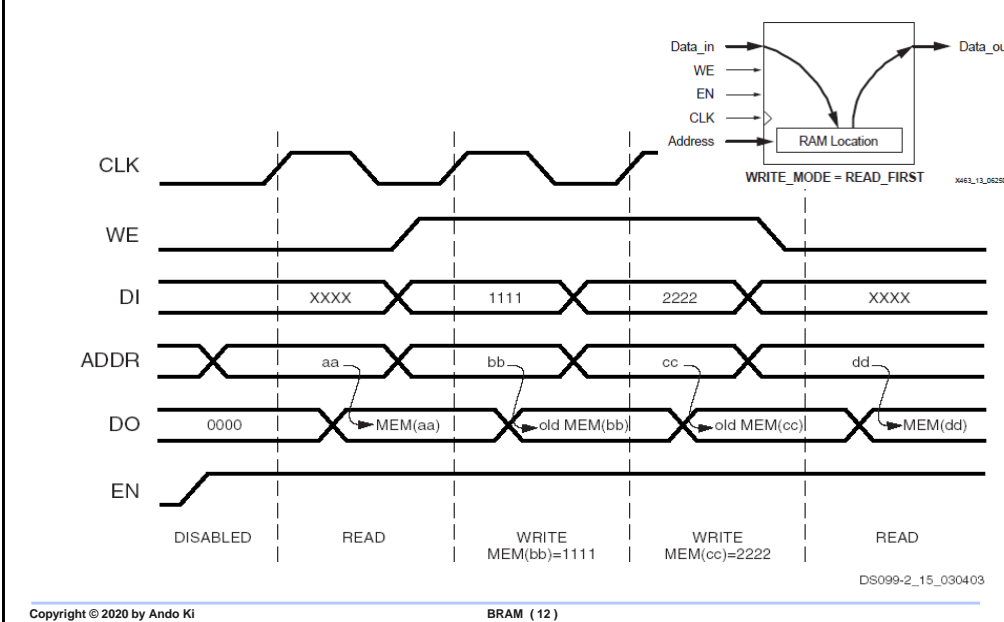
Copyright © 2020 by Ando Ki

BRAM (10)

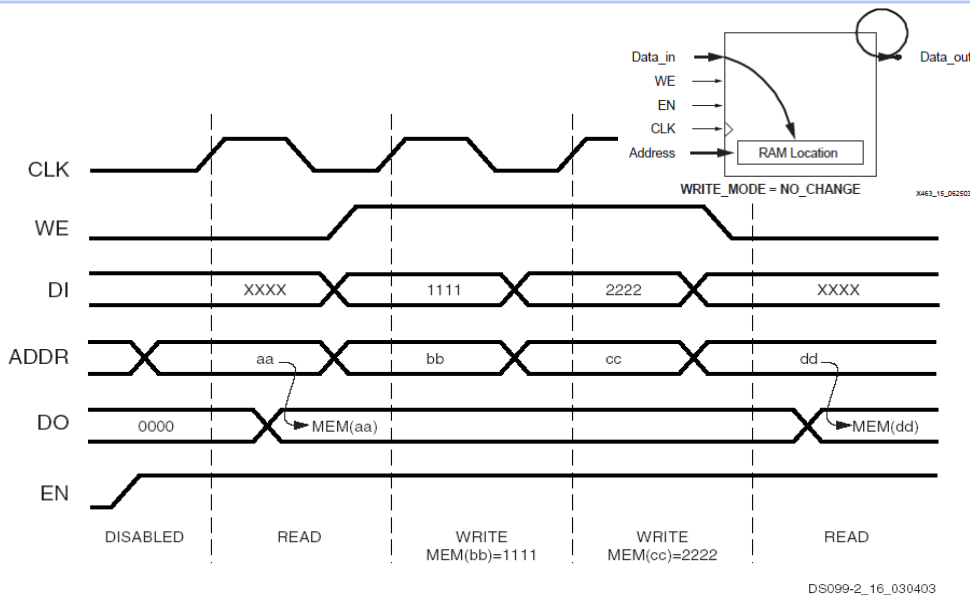
Block RAM Waveforms – WRITE_FIRST mode



Block RAM Waveforms – READ_FIRST mode



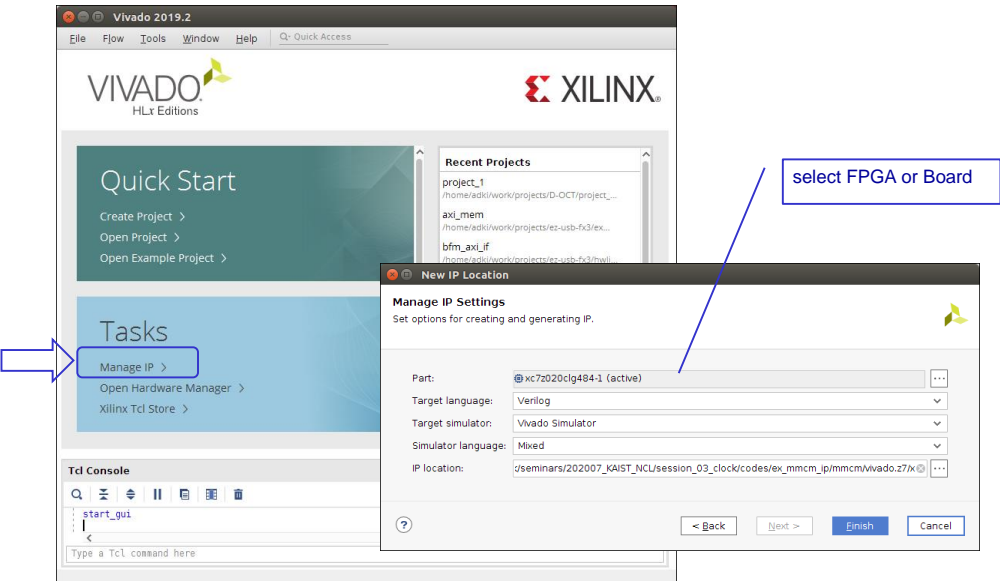
Block RAM Waveforms – NO_CHANGE mode



Copyright © 2020 by Ando Ki

BRAM (13)

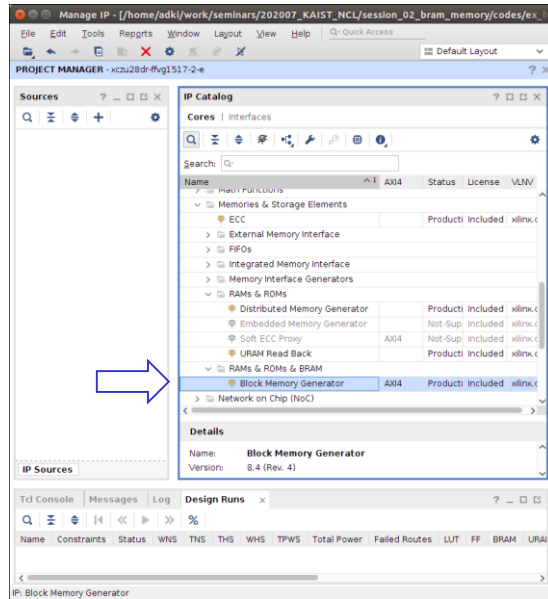
Vivado IP manager



Copyright © 2020 by Ando Ki

BRAM (14)

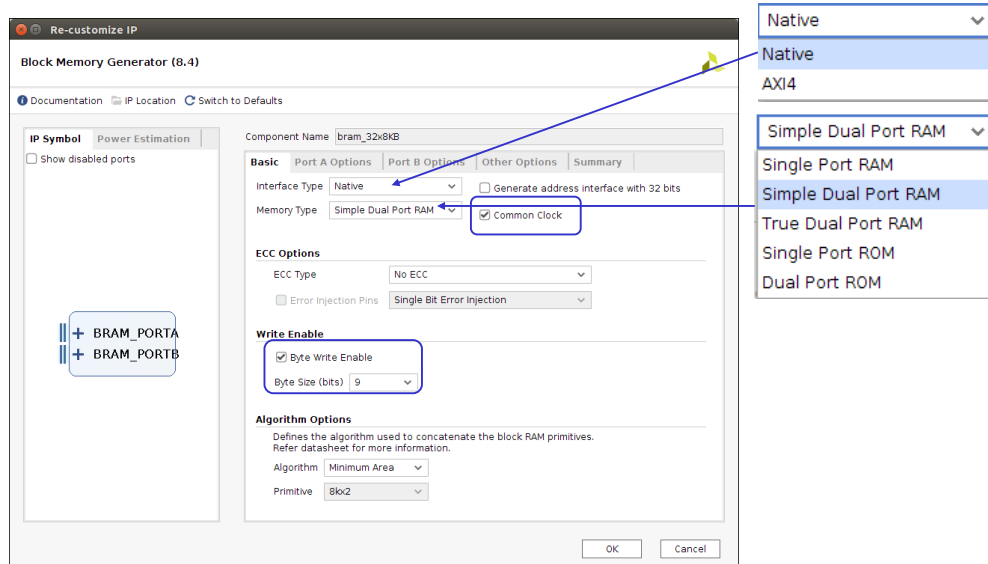
Vivado IP manager



Copyright © 2020 by Ando Ki

BRAM (15)

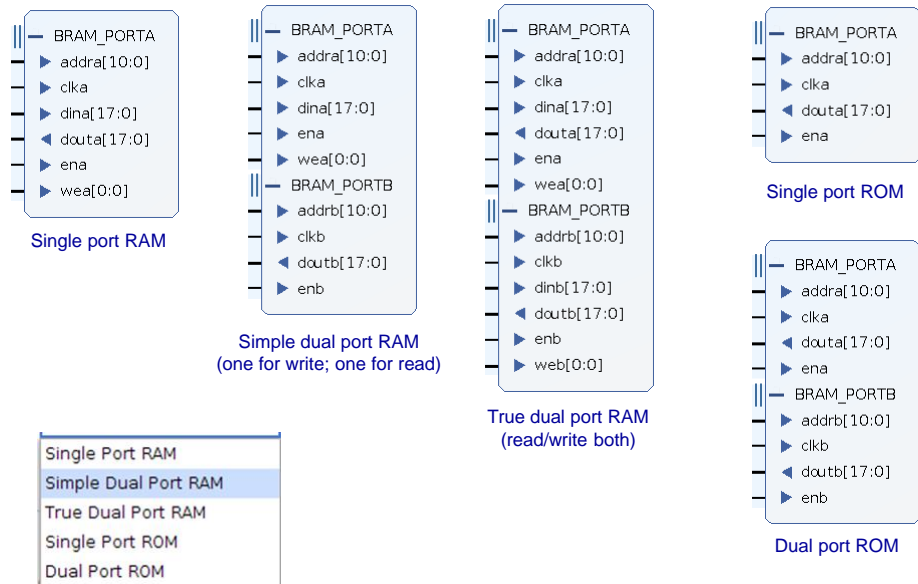
Vivado IP manager



Copyright © 2020 by Ando Ki

BRAM (16)

Bram interface types



Copyright © 2020 by Ando Ki

BRAM (17)

Generated files

Name
doc
hdl
misc
sim
simulation
synth
bram_32x8KB.dcp
bram_32x8KB.veo
bram_32x8KB.vho
bram_32x8KB.xci
bram_32x8KB.xml
bram_32x8KB_ooc.xdc
bram_32x8KB_sim_netlist.v
bram_32x8KB_sim_netlist.vhdl
bram_32x8KB_stub.v
bram_32x8KB_stub.vhdl
summary.log

- ❏ DCP
 - ◆ Design check point
 - ◆ a sort of archive file
- ❏ VEO, VHO
 - ◆ Verilog/VHDL template file
- ❏ XCI
 - ◆ Xilinx Core Instance
 - ◆ a kind of project file
- ❏ XDC
 - ◆ constraint
- ❏ sim_netlist.v, sim_netlist.vhdl
 - ◆ Gate-level simulation
- ❏ stub.v, stub.vhdl
 - ◆ E.g., black box

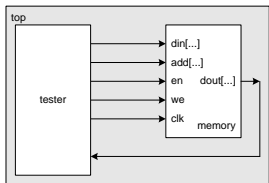
Copyright © 2020 by Ando Ki

BRAM (18)

Example: memory using BRAM

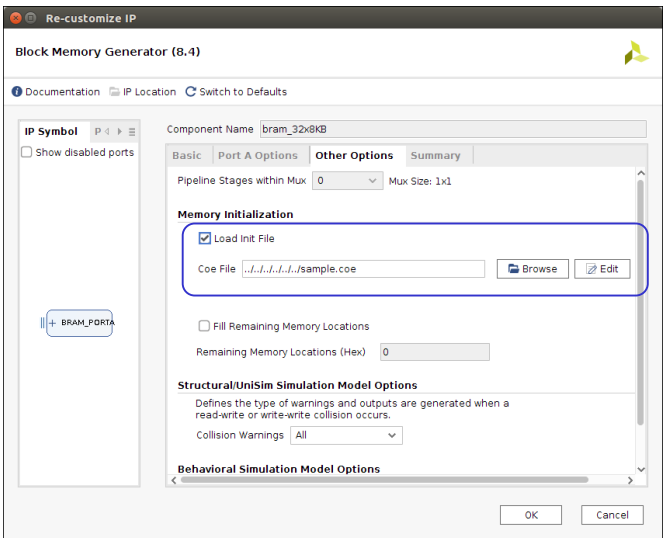
Run an example code as follows

- Prepare a single port BRAM of 8KByte with 32-bit data width
- Run simulation and check VCD
 - do not forget to add '-testplusarg VCD=1' option for 'xsim', which enables to generate VCD file



```
[user@host] cd $(PROJECT)/codes/bram/ex_bram/sim/xsim
[user@host] make
[user@host] gtkwave wave.vcd &
```

How to fill contents



Xilinx BRAM coefficient file

***** Example of Dual Port Block Memory .COE file *****

```
; Sample memory initialization file for Dual Port Block Memory,
; v3.0 or later.
;
; This .COE file specifies the contents for a block memory
; of depth=16, and width=4. In this case, values are specified
; in hexadecimal format.
memory_initialization_radix=2;
memory_initialization_vector=
1111,
1111,
1111,
1111,
1111,
1111,
0000,
0101,
0011,
0000,
1111,
1111,
1111,
1111,
1111,
1111;
```

***** Example of Single Port Block Memory .COE file *****

```
; Sample memory initialization file for Single Port Block Memory,
; v3.0 or later.
;
; This .COE file specifies initialization values for a block
; memory of depth=16, and width=8. In this case, values are
; specified in hexadecimal format.
memory_initialization_radix=16;
memory_initialization_vector=
ff,
ab,
f0,
11,
11,
00,
01,
aa,
bb,
cc,
dd,
ef,
ee,
ff,
00,
ff;
```

Copyright © 2020 by Ando Ki

BRAM (21)

Example: memory with initial value

Run an example code as follows



```
File Edit Tools Syntax Buffers
memory_initialization_radix=16;
memory_initialization_vector=
04030201,
08070605,
0C0B0A09,
000F0E0D;
```

```
[user@host] cd $(PROJECT)/codes/bram/ex_bram_coe/sim/xsim
[user@host] make
[user@host] gtkwave wave.vcd &
```

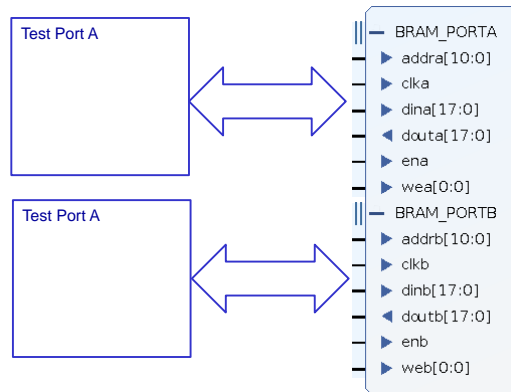
Copyright © 2020 by Ando Ki

BRAM (22)

Projects

❏ Make a BRAM of asynchronous 'True Dual Port RAM'

- ◆ Make sure 'clka' and 'clkb' are not the same



Copyright © 2020 by Ando Ki

BRAM (23)

References

- ❏ Using Block RAM in Spartan-3 Generation FPGAs, XAPP463, Xilinx 2005.
- ❏ Block Memory Generator LogiCORE IP Product Guide Vivado Design Suite PG058

Copyright © 2020 by Ando Ki

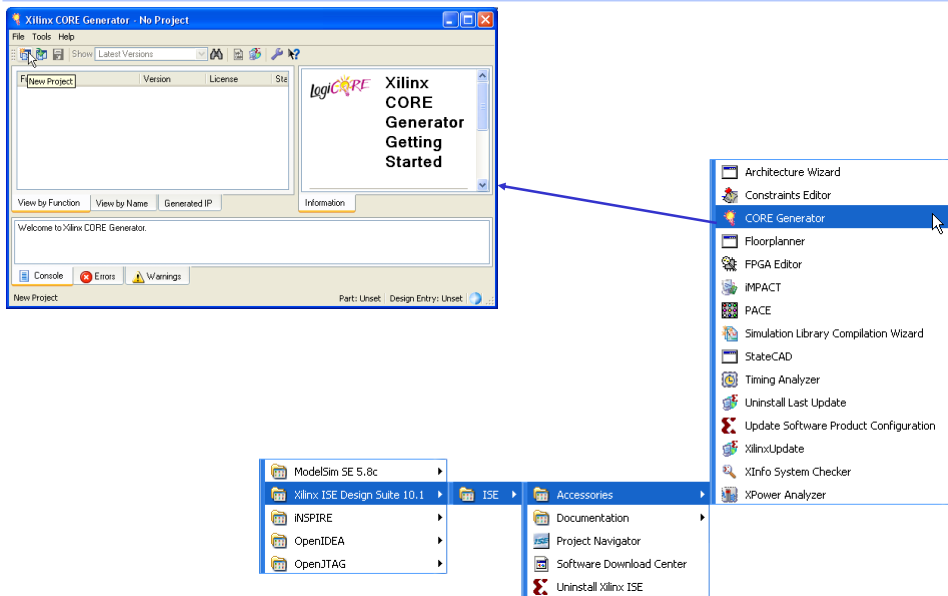
BRAM (24)

For the case of ISE

Create BRAM

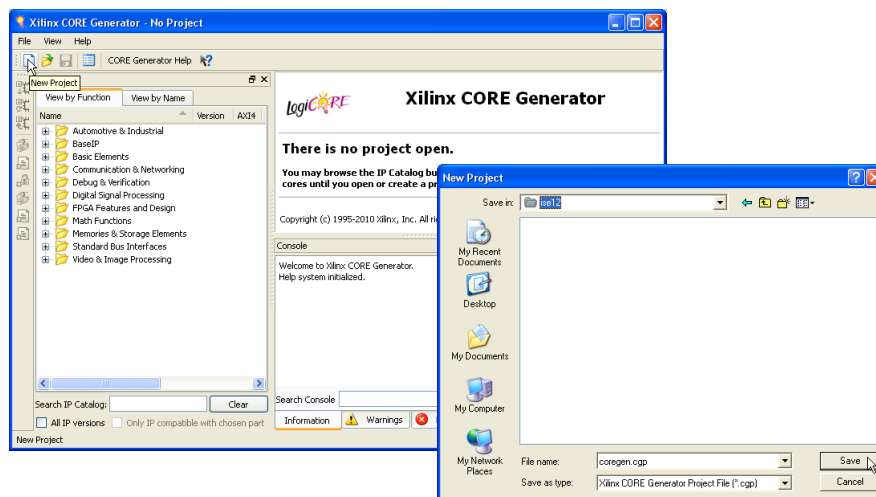
 Use Xilinx CORE Generator

Invoke Xilinx ISE CORE Generator

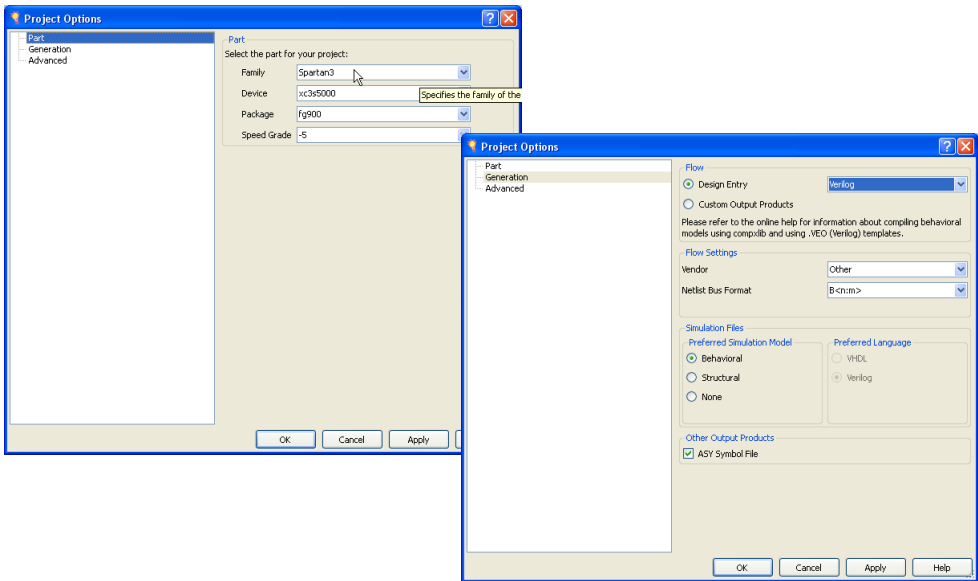


Create BRAM: CORE Generator

- Create a new project at
 $\diamond \$(PROJECT)/codes/bram/ex_bram/coregen/ise12$



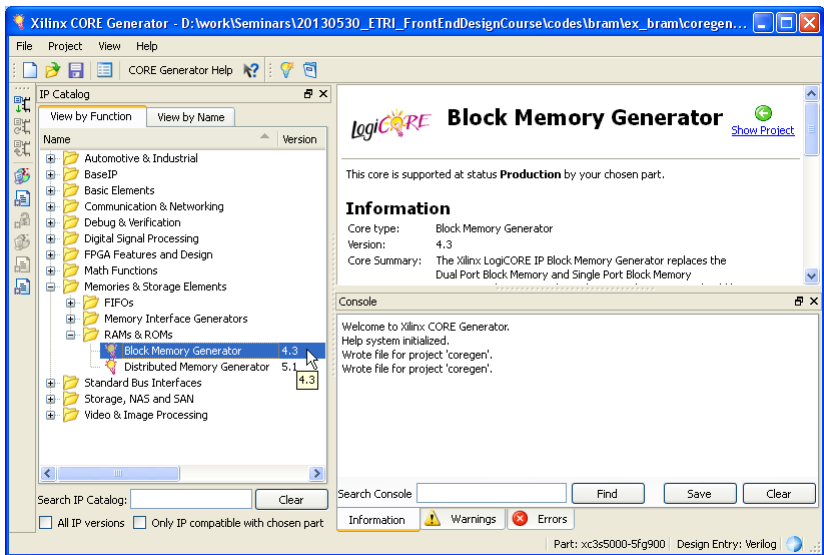
Create BRAM: CORE Generator



Copyright © 2020 by Ando Ki

BRAM (29)

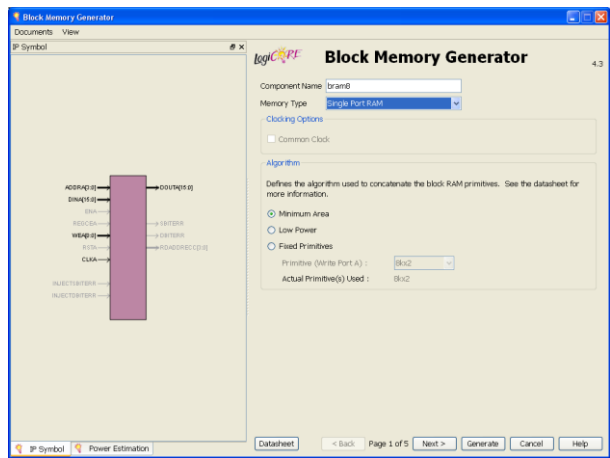
Create BRAM: CORE Generator



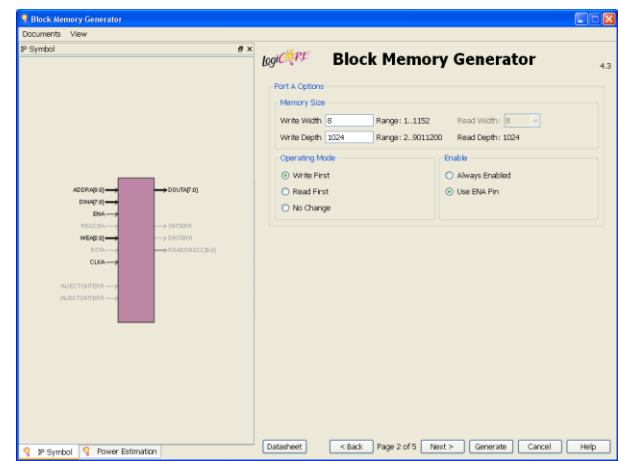
Copyright © 2020 by Ando Ki

BRAM (30)

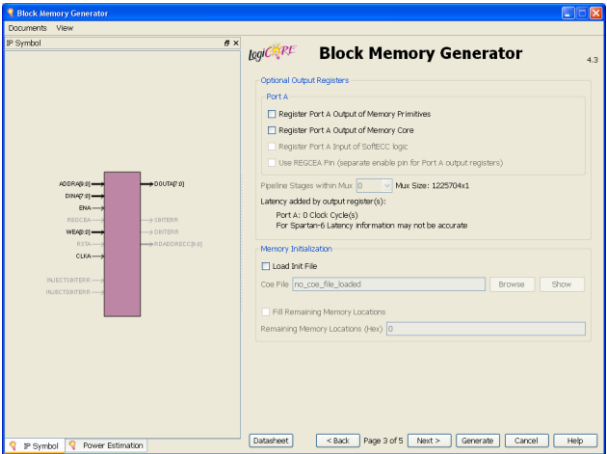
Create BRAM: CORE Generator



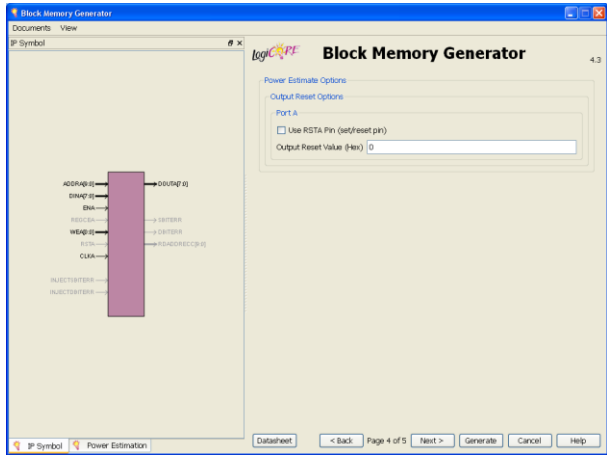
Create BRAM: CORE Generator



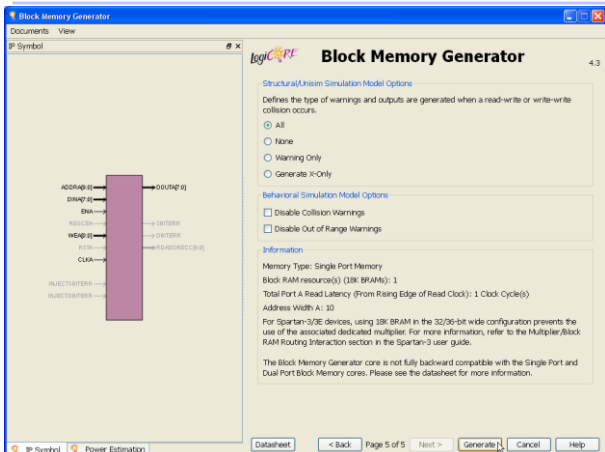
Create BRAM: CORE Generator



Create BRAM: CORE Generator



Create BRAM: CORE Generator



Block Memory Generator

Structural/Unknown Simulation Model Options

Defines the type of warnings and outputs are generated when a read-write or write-write collision occurs.

☒ All

☐ None

☐ Warning Only

☐ Generate X-Only

Behavioral Simulation Model Options

☒ Disable Collision Warnings

☒ Disable Out of Range Warnings

Information

Memory Type: Single Port Memory

Block RAM Resource(s): 1 (1K BRAMs)

Total Port A Read Latency (From Rising Edge of Read Clock): 1 Clock Cycle(s)

Address Width A: 32

For Spartan-3/3E devices, using 18K BRAM in the 32/36-bit wide configuration prevents the use of the associated dedicated multiplier. For more information, refer to the Multiplier/Block RAM Routing Interaction section in the Spartan-3 user guide.

The Block Memory Generator core is not fully backward compatible with the Single Port and Dual Port Block Memory cores. Please see the datasheet for more information.

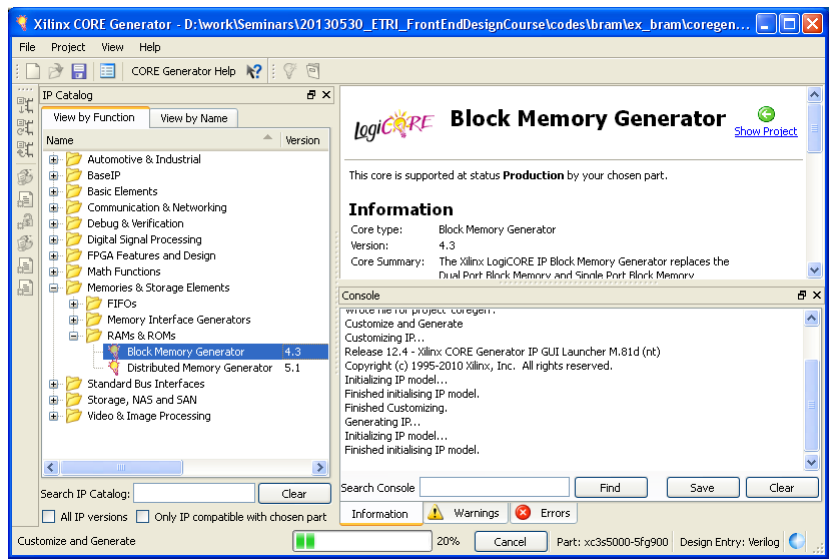
IP Symbol Power Estimation

Generate

Copyright © 2020 by Ando Ki

BRAM (35)

Create BRAM: CORE Generator



Xilinx CORE Generator - D:\work\Seminars\20130530_ETRI_FrontEndDesignCourse\codes\bram\ex_bram\coregen...

File Project View Help

CORE Generator Help

IP Catalog

View by Function View by Name

Name Version

- Automotive & Industrial
- BaseIP
- Basic Elements
- Communication & Networking
- Debug & Verification
- Digital Signal Processing
- FPGA Features and Design
- Math Functions
- Memories & Storage Elements
 - FIFOs
 - Memory Interface Generators
 - RAMs & ROMs
 - Block Memory Generator 4.3**
 - Distributed Memory Generator 5.1
- Standard Bus Interfaces
- Storage, NAS and SAN
- Video & Image Processing

Search IP Catalog: Clear

☐ All IP versions ☐ Only IP compatible with chosen part

Customize and Generate

Block Memory Generator

This core is supported at status **Production** by your chosen part.

Information

Core type: Block Memory Generator

Version: 4.3

Core Summary: The Xilinx LogiCORE IP Block Memory Generator replaces the Dual Port Block Memory and Single Port Block Memory.

Console

Initialize for project: coregen...

Customize and Generate

Customizing IP...

Release 12.4 - Xilinx CORE Generator IP GUI Launcher M.81d (nt)

Copyright (c) 1995-2010 Xilinx, Inc. All rights reserved.

Initializing IP model...

Finished initialising IP model.

Finished Customizing.

Generating IP...

Initializing IP model...

Finished initialising IP model.

Search Console Find Save Clear

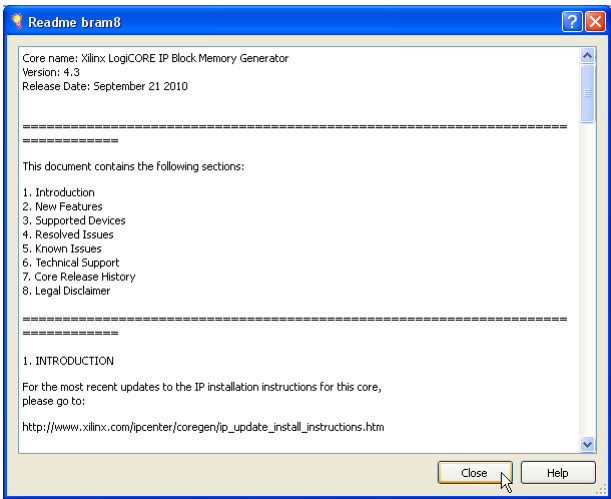
Information Warnings Errors

Cancel Part: xc3s5000-5fg900 Design Entry: Verilog

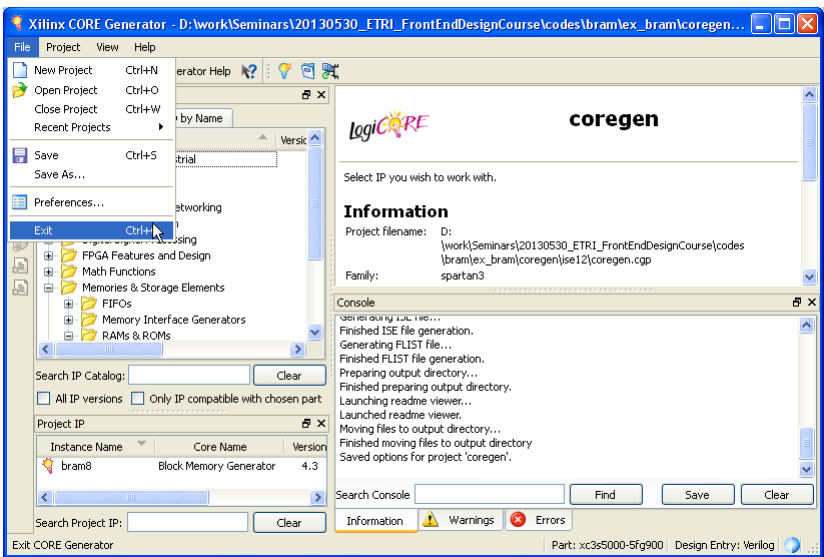
Copyright © 2020 by Ando Ki

BRAM (36)

Create BRAM: CORE Generator



Create BRAM: CORE Generator



Create BRAM: CORE Generator

Name	Size	Type
.._msgs		File Folder
tmp		File Folder
xilinx_auto_0_xdb		File Folder
blk_mem_gen_ds512.pdf	3,632 KB	Adobe Acrobat 문서
blk_mem_gen_readme.txt	7 KB	Text Document
bram8.asy	1 KB	ASY File
bram8.gise	2 KB	GISE File
bram8.ngc	14 KB	NGC File
bram8.v	7 KB	..v
bram8.vco	5 KB	VEO File
bram8.xco	3 KB	XCO File
bram8.xise	37 KB	Xilinx ISE Project
bram8_list.txt	1 KB	Text Document
bram8_xmdf.tcl	3 KB	ActiveTcl Script
coregen.cgc	18 KB	CGC File
coregen.cgp	1 KB	CGP File

```
`timescale 1ns/1ps

module bram8(clka, ena, wea, addra, dina, douta);
input clka;
input ena;
input [0 : 0] wea;
input [9 : 0] addra;
input [7 : 0] dina;
output [7 : 0] douta;

// synthesis translate_off

    BLK_MEM_GEN_V4_3 #(
    ... ..
    .C_XDEVICEFAMILY("spartan3"))
    inst (.CLKA(clka),
        .ENA(ena),
        .WEA(wea),
        .ADDRA(addra),
        .DINA(dina),
        .DOUTA(douta),
        ... .. );

// synthesis translate_on

endmodule
```