ECE 414 Computer Organization and Design

Tutorial –III – Testbench design for Evaluating CPU

Prepared by Randil Gajasinghe and Dr. Onur Tigli

This tutorial describes how to design the testbench for testing your CPU. The testbench consists of the following sections

i. ROM

ROM contains assembly instructions in binary format. The instructions have to be written into the ROM at core-generation stage. The ROM core, and the programming file, has to be regenerated every time ROM contents are changed.

ii. CPU

This is your design of CPU. The interface of input and output lines should to match those in the '*dummy_unit*' module in the provided Verilog code.

```
module CPU(                              //represent the CPU
    input [15:0] data_from_rom,
    input reset,
    input clk,
    inout [15:1] data _ram,
    output [5:0] address_to_rom,
    output enable_to_rom,
    output write_enable_to_ram,          //enable write
    output [5:0] address_to_ram,
    output read_enable_to_ram,
    output enable_ram_read               //enable signal for ram read and UART module. This signal indicates that all
                                         //operations of the 'CPU' are finished.
);
```

iii. RAM

The CPU will use this memory.

iv. READ RAM

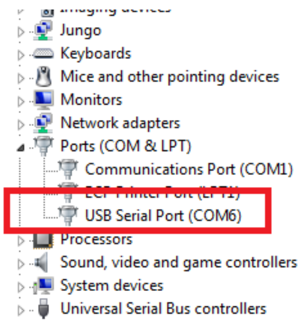
After CPU finish executing the program in the ROM, it has to enter idle state and send a continuous enable signal to this unit. READ RAM module reads the RAM and sends the contents byte by byte to the UART module.

v. UART

This module transmits the data to the computer.

Drivers for *USB to Serial* convertor in the FPGA board must be installed for the computer to receive data. Windows 7 will automatically install drivers when the board is connected to computer by the USB cable. Open device manager and note the COM port used.

ECE 414 Computer Organization and Design

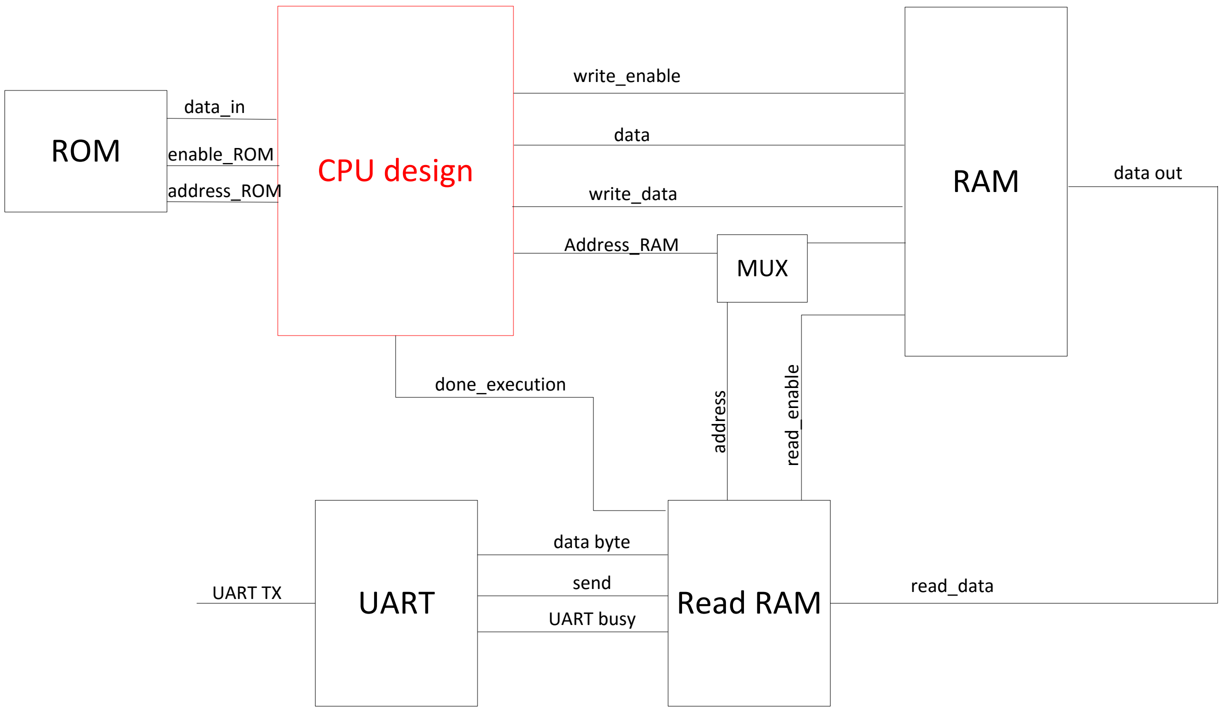


Imaging devices
Jungo
Keyboards
Mice and other pointing devices
Monitors
Network adapters
Ports (COM & LPT)
Communications Port (COM1)
ECP Printer Port (LPT1)
USB Serial Port (COM6)
Processors
Sound, video and game controllers
System devices
Universal Serial Bus controllers

You can use ***PuTTY*** program to receive the data. PuTTy must establish a serial connection with the FPGA board using the following settings

-- *9600 Baud Rate*
-- *8 data bits, LSB first*
-- *1 stop bit*
-- *no parity*
-- *COM port as noted in previous step*

A simplified block diagram of testbench for testing the CPU design is shown below. Verilog code files for testbench circuit will be provided. You will have to add them to your design (CPU design) for testing and demonstration. The interface between the CPU and the testbench has to be correct.



i. Data_in : data in from ROM [16 bit]
ii. Enable_ROM : enable signal to ROM. Should be high for a single cycle when reading from ROM
iii. Address_ROM : read address to the ROM [6 bit]
iv. Address_RAM : read/write address to the memory
v. Write_data : data to be written to the RAM [16 bit]
vi. Write_enable : enable signal for RAM write. Should be high for a single cycle when writing to ROM
vii. Read_data : data read from RAM [16 bit]
viii. Read_enable : enable signal for RAM read. Should be high for a single cycle when reading ROM
ix. Done_execution : CPU has finished executing code. This signal should be high continuously.

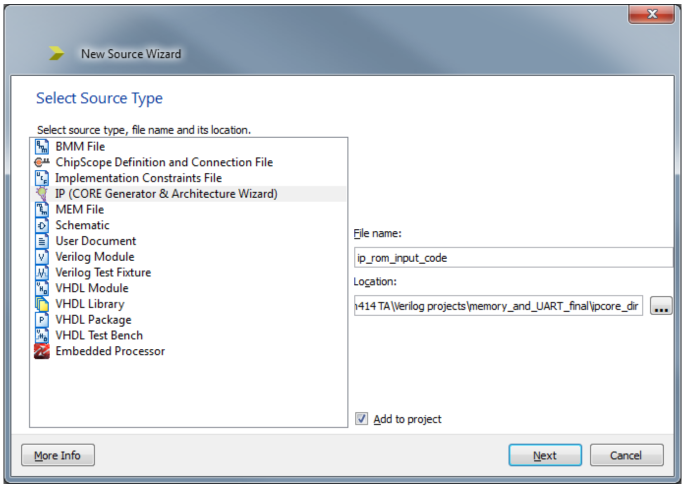
Please note that the names of the input/output lines have to match section '(ii) CPU' above.

The following section explains how to generate memory and the complete testbench.

1. Section one – ROM

The instructions to be executed will be stored in a ROM. We will use an ip-core ROM. Instructions have to be stored in the ROM at core generation stage. The ROM and the *bit* file have to be re-generated when the instructions in the ROM are changed.

i. Add a new IP (CORE generator & Architecture Wizard) source to your design. Name the new core '*ip_rom_input_code*'. Click next.



New Source Wizard

Select Source Type

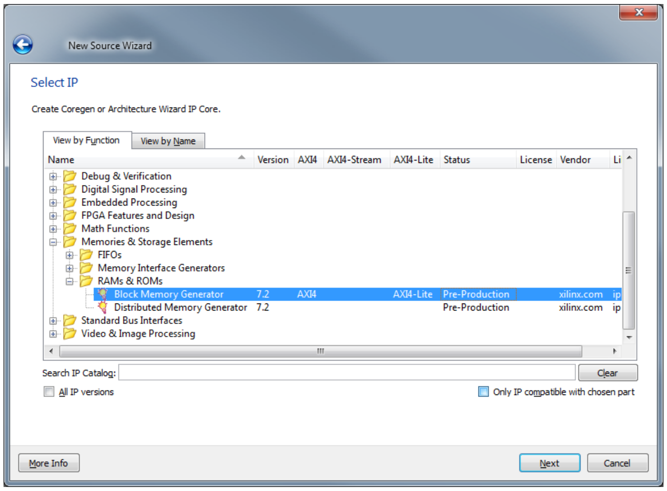
Select source type, file name and its location.

File name: ip_rom_input_code

Location: \414 TA\Verilog projects\memory_and_UART_final\ipcore_dir

☑ Add to project

ii. Select 'Block Memory Generator' and click next and finish.



Select IP

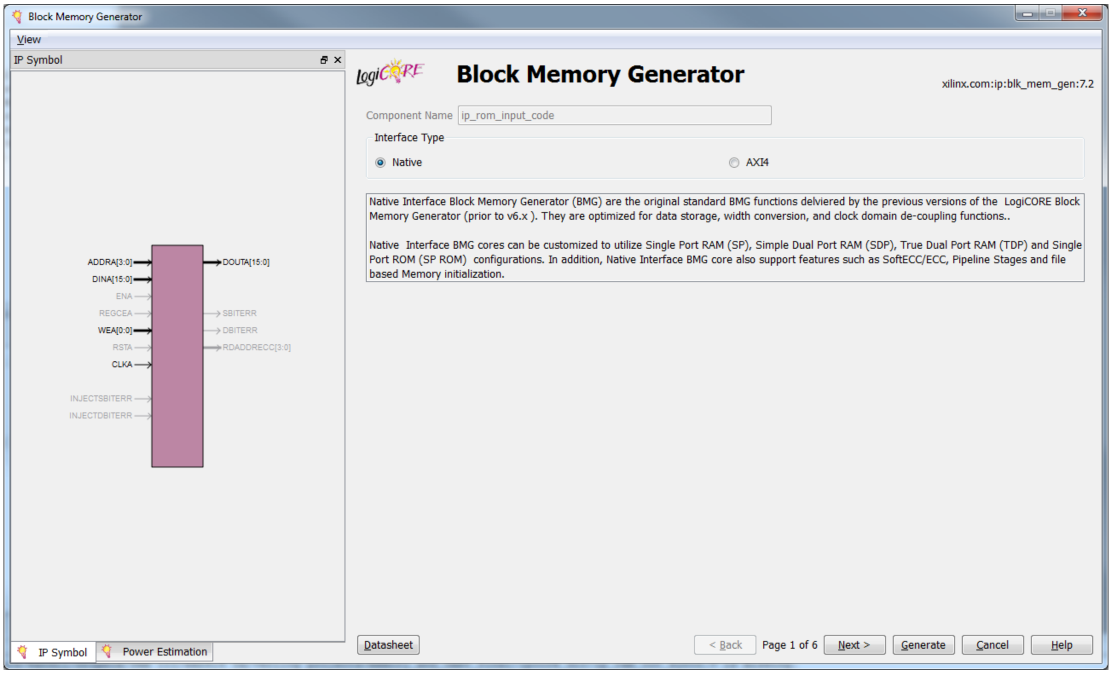
Create Coregen or Architecture Wizard IP Core.

Name	Version	AXI4	AXI4-Stream	AXI4-Lite	Status	License	Vendor	Li
Debug & Verification								
Digital Signal Processing								
Embedded Processing								
FPGA Features and Design								
Math Functions								
Memories & Storage Elements								
FIFOs								
Memory Interface Generators								
RAMs & ROMs								
Block Memory Generator	7.2	AXI4		AXI4-Lite	Pre-Production		xilinx.com	ip
Distributed Memory Generator	7.2				Pre-Production		xilinx.com	ip
Standard Bus Interfaces								
Video & Image Processing								

Search IP Catalog:

☐ All IP versions ☐ Only IP compatible with chosen part

iii. In the '*Block memory generator*' window, make changes as shown below.



Block Memory Generator

xilinx.com:ip:blk_mem_gen:7.2

Component Name: ip_rom_input_code

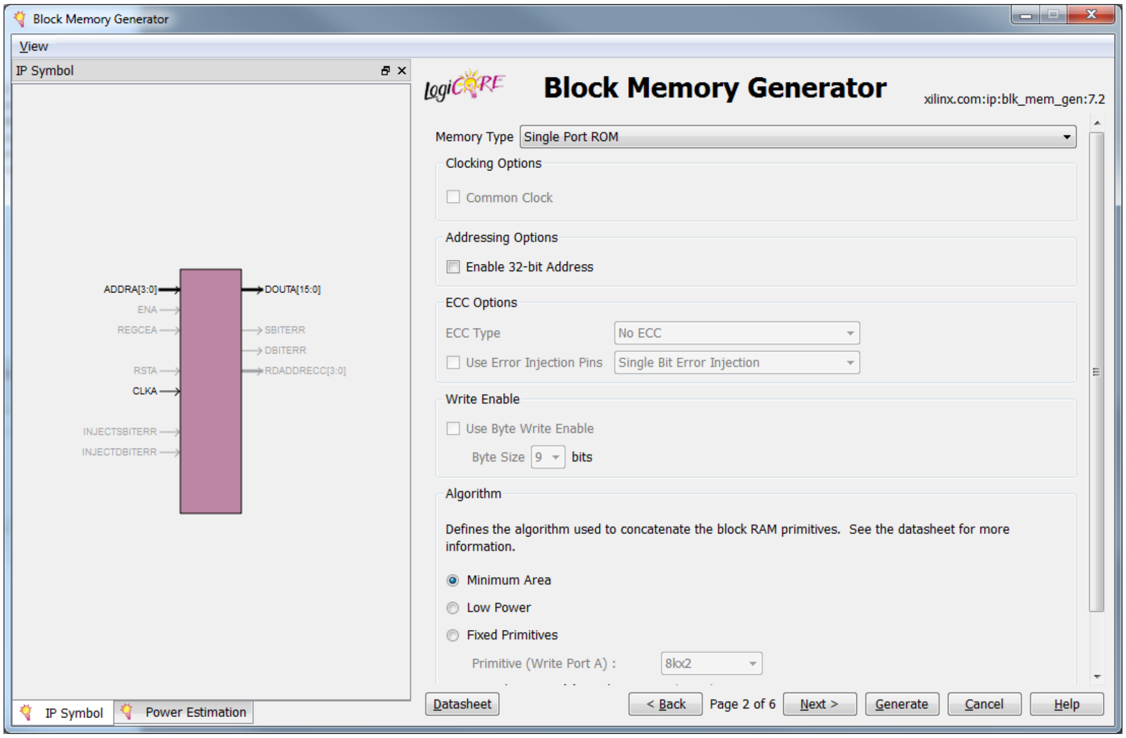
Interface Type

● Native ○ AXI4

Native Interface Block Memory Generator (BMG) are the original standard BMG functions delviered by the previous versions of the LogiCORE Block Memory Generator (prior to v6.x). They are optimized for data storage, width conversion, and clock domain de-coupling functions..

Native Interface BMG cores can be customized to utilize Single Port RAM (SP), Simple Dual Port RAM (SDP), True Dual Port RAM (TDP) and Single Port ROM (SP ROM) configurations. In addition, Native Interface BMG core also support features such as SoftECC/ECC, Pipeline Stages and file based Memory initialization.

In the second page, choose "*single port ROM'*



Block Memory Generator

xilinx.com:ip:blk_mem_gen:7.2

Memory Type: Single Port ROM

Clocking Options

□ Common Clock

Addressing Options

□ Enable 32-bit Address

ECC Options

ECC Type: No ECC

□ Use Error Injection Pins: Single Bit Error Injection

Write Enable

□ Use Byte Write Enable

Byte Size 9 bits

Algorithm

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

● Minimum Area

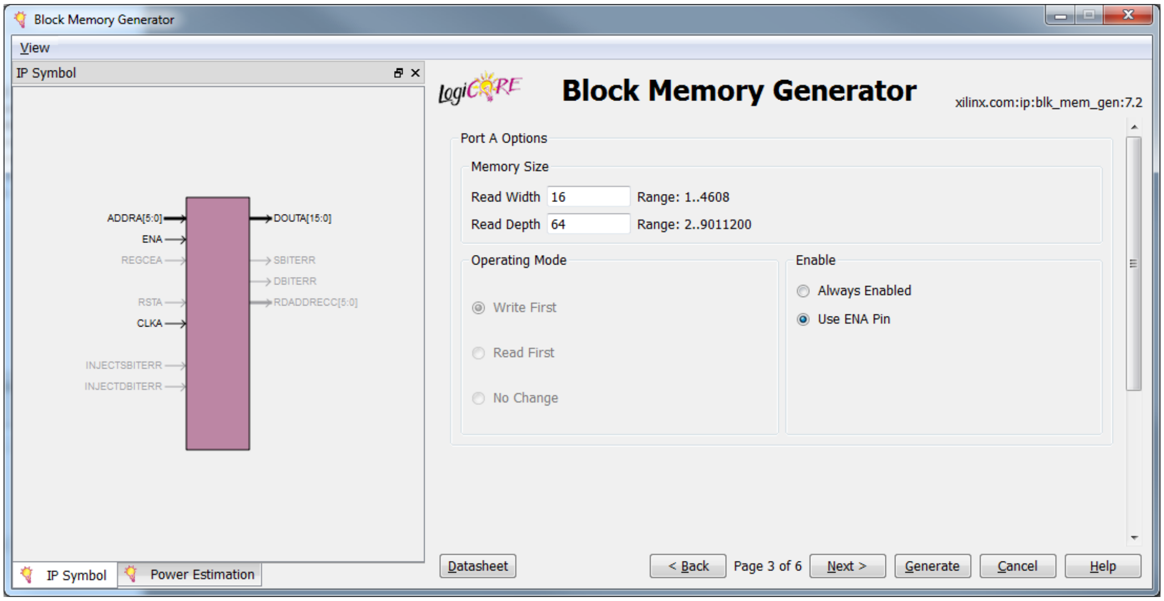
○ Low Power

○ Fixed Primitives

Primitive (Write Port A) : 8kx2

Page 2 of 6

Choose 16 as the read width and 64 as the read depth. Select the '*Use ENA Pin'* radio button.



Block Memory Generator

xilinx.com:ip:blk_mem_gen:7.2

Port A Options

Memory Size

Read Width 16 Range: 1..4608

Read Depth 64 Range: 2..9011200

Operating Mode

● Write First

○ Read First

○ No Change

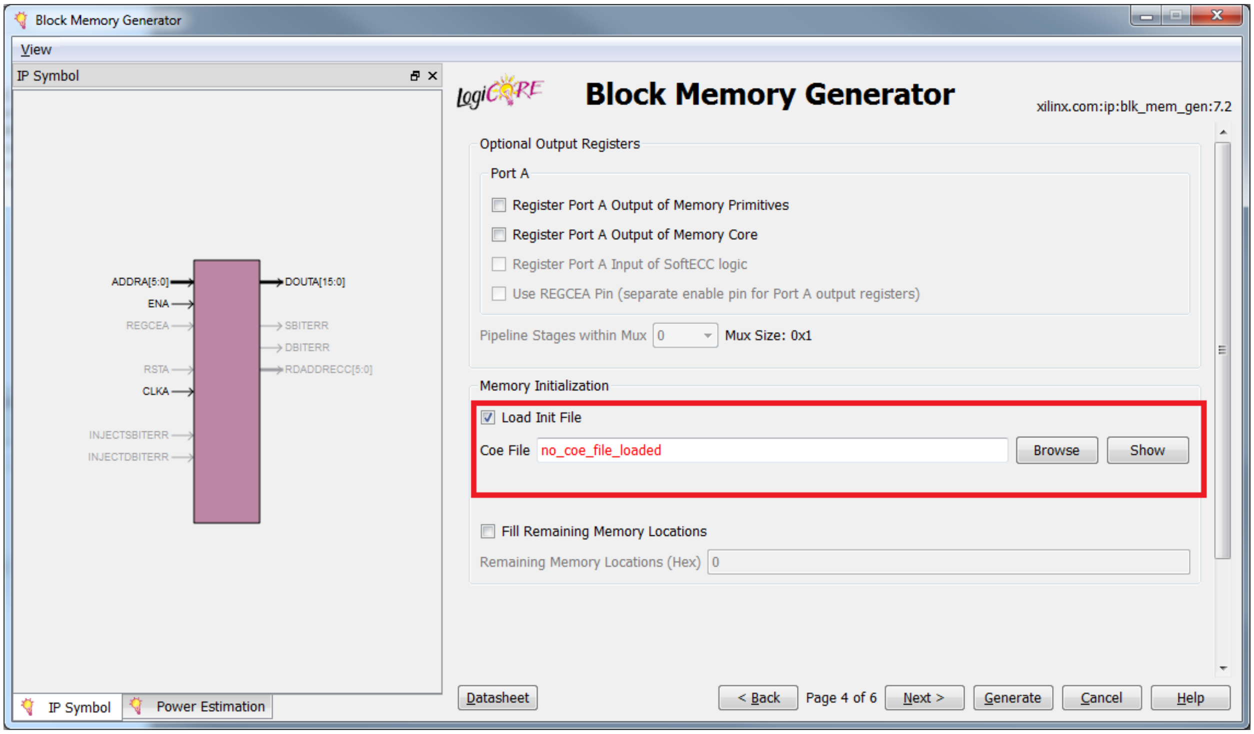
Enable

○ Always Enabled

● Use ENA Pin

Page 3 of 6

Select '*Load Init File*' and browse and select the ***.coe*** file containing the binary (or HEX) code of the instructions.



Optional Output Registers

Port A

☐ Register Port A Output of Memory Primitives

☐ Register Port A Output of Memory Core

☐ Register Port A Input of SoftECC logic

☐ Use REGCEA Pin (separate enable pin for Port A output registers)

Pipeline Stages within Mux 0 Mux Size: 0x1

Memory Initialization

☑ Load Init File

Coe File no_coe_file_loaded

☐ Fill Remaining Memory Locations

Remaining Memory Locations (Hex) 0

Click *Generate.*

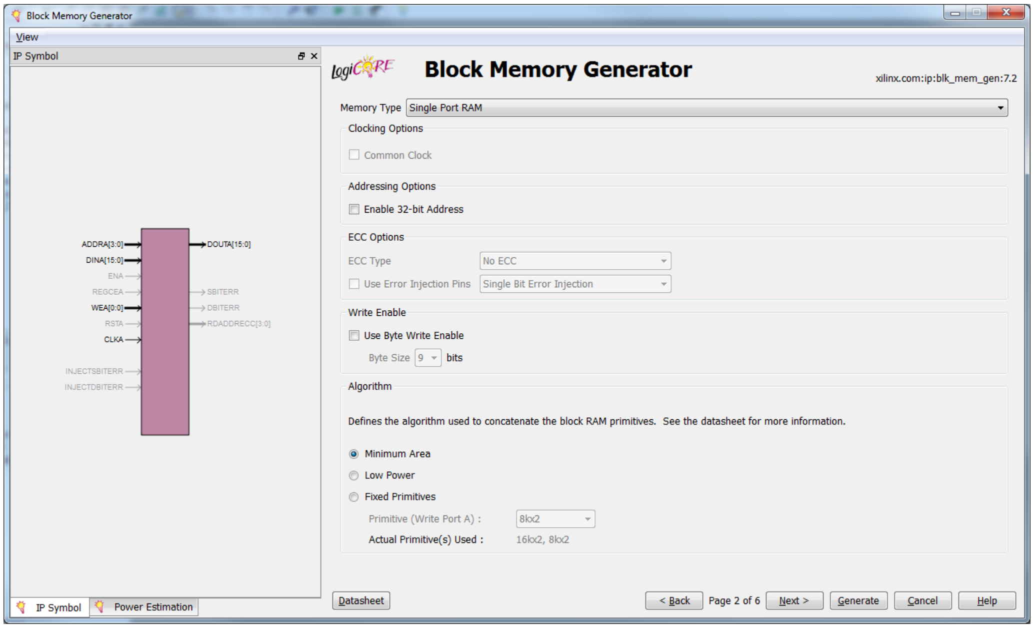
2. Section two – RAM

i. Add a new IP (CORE generator & Architecture Wizard) source to your design. Name the new core '*ip_ram_output_code*'. Click next.

ii. Select 'Block Memory Generator' and click next and finish.

iii. In the '*Block memory generator'* window, make changes as shown below.

In the second page, choose “*single port RAM’*



Block Memory Generator

xilinx.com:ip:blk_mem_gen:7.2

Memory Type: Single Port RAM

Clocking Options

□ Common Clock

Addressing Options

□ Enable 32-bit Address

ECC Options

ECC Type: No ECC

□ Use Error Injection Pins: Single Bit Error Injection

Write Enable

□ Use Byte Write Enable

Byte Size 9 bits

Algorithm

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

● Minimum Area

○ Low Power

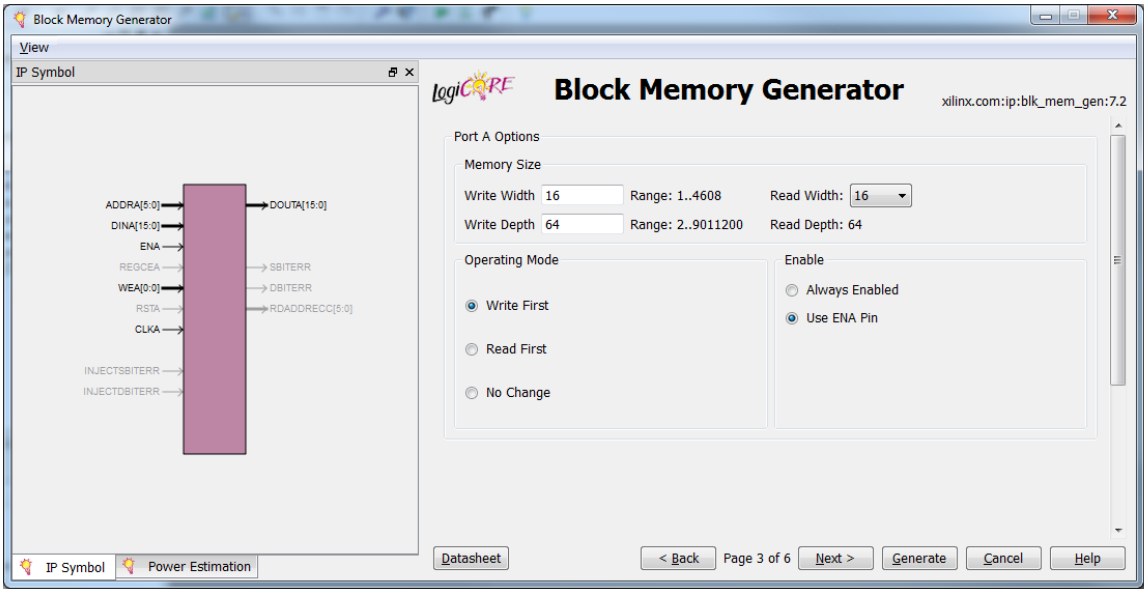
○ Fixed Primitives

Primitive (Write Port A) : 8kx2

Actual Primitive(s) Used : 16kx2, 8kx2

Page 2 of 6

In the third page, choose 16 bits for width and 64 bits for depth. Select ‘*use ENA pin’*



Block Memory Generator

xilinx.com:ip:blk_mem_gen:7.2

Port A Options

Memory Size

Write Width 16 Range: 1..4608 Read Width: 16

Write Depth 64 Range: 2..9011200 Read Depth: 64

Operating Mode

● Write First

○ Read First

○ No Change

Enable

○ Always Enabled

● Use ENA Pin

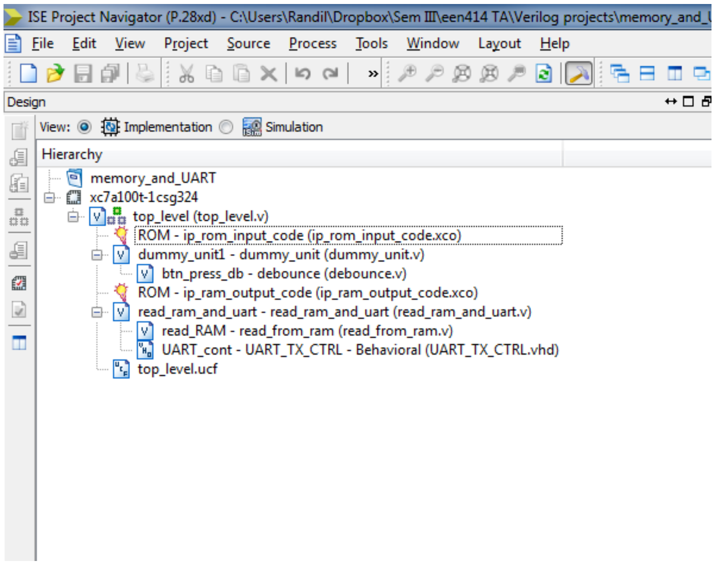
Page 3 of 6

In the fourth page, you can initiate the RAM if necessary.

3. Section three – add provided files and your CPU design to the design.

- Read_ram_and_uart.v
- Read_from_ram.v
- UART_TX_CTRL.vhd
- Top_level.v
- Top_level.ucf
- Your CPU design file

Your design should look like the figure below. (dummy unit will be your CPU)


ISE Project Navigator (P.28xd) - C:\Users\Randil\Dropbox\Sem III\een414 TA\Verilog projects\memory_and_
File
Edit
View
Project
Source
Process
Tools
Window
Layout
Help
Design
View:
Implementation
Simulation
Hierarchy

```
memory_and_UART
xc7a100t-1csg324
  top_level (top_level.v)
    ROM - ip_rom_input_code (ip_rom_input_code.xco)
    dummy_unit1 - dummy_unit (dummy_unit.v)
      btn_press_db - debounce (debounce.v)
    ROM - ip_ram_output_code (ip_ram_output_code.xco)
    read_ram_and_uart - read_ram_and_uart (read_ram_and_uart.v)
      read_RAM - read_from_ram (read_from_ram.v)
      UART_cont - UART_TX_CTRL - Behavioral (UART_TX_CTRL.vhd)
    top_level.ucf
```

Delete the following file from the ***\ipcore_dir*** directory. Otherwise you will get an error.

- ip_ram_output_code.v
- ip_rom_input_code.v

Generate the programming file. Load the file to FPGA and establish a serial connection with the computer (PuTTY program). Press the push button to enable CPU. (take a look at provide UCF file) the output of the assembly program should be displayed in PuTTY.