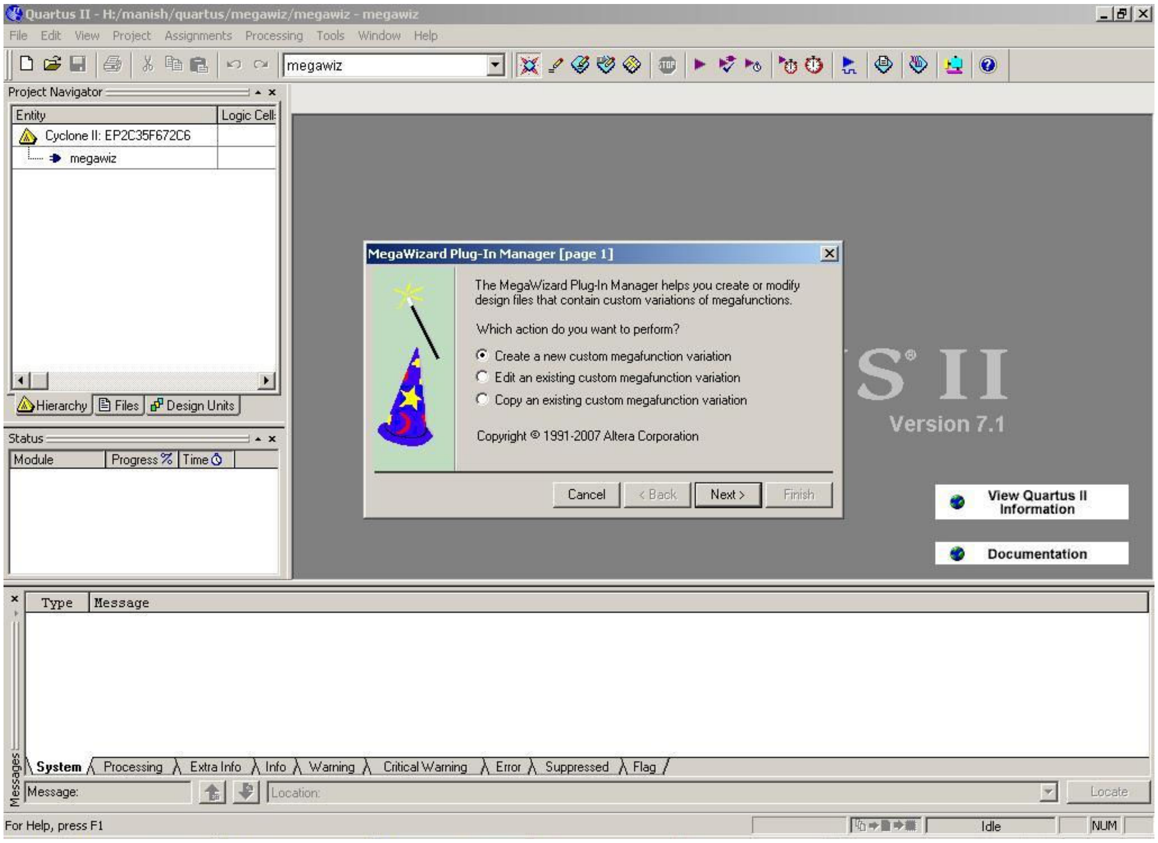
Run-Time Content Editable Memory using MegaWizard Plug-In Manager

Note: Before you start with this tutorial it is important to go through tutorial 'Altera Quartus II and DE2 manual' available on course website.

1. To start creating the memory module, go to Tools -> MegaWizard Plug-In Manager. And click on "Create a new custom megafunction variation" as shown in Fig 1. Then click Next.



MegaWizard Plug-In Manager [page 1]

The MegaWizard Plug-In Manager helps you create or modify design files that contain custom variations of megafunctions.

Which action do you want to perform?

● Create a new custom megafunction variation

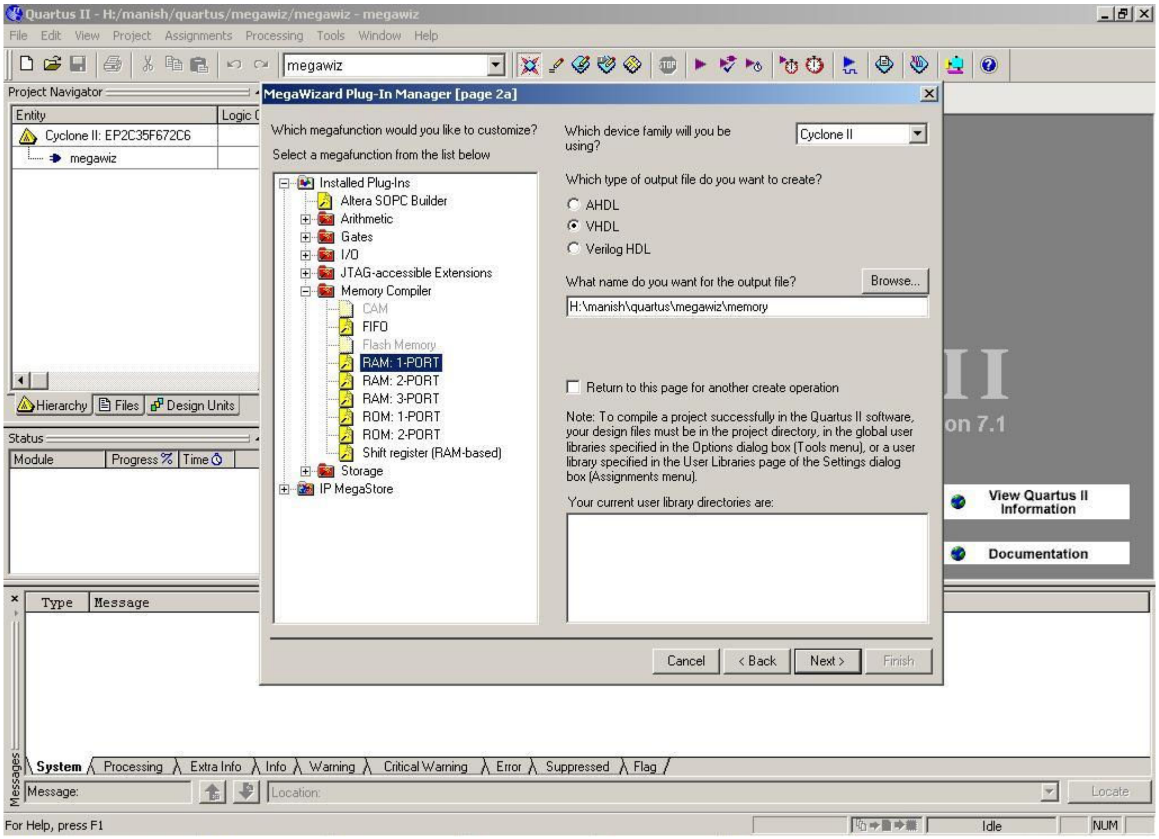
○ Edit an existing custom megafunction variation

○ Copy an existing custom megafunction variation

Copyright © 1991-2007 Altera Corporation

Fig. 1

2. On the 2nd page select the device family “Cyclone II”. On the left hand side under the name memory compilers choose RAM: 1-PORT module as shown in the Fig 2. Type of output file that needs to be selected is VHDL. And finally name your output file. Then click Next.



MegaWizard Plug-In Manager [page 2a]

Which megafunction would you like to customize?

Select a megafunction from the list below

Which device family will you be using? Cyclone II

Which type of output file do you want to create?

○ AHDL

● VHDL

○ Verilog HDL

What name do you want for the output file? H:\manish\quartus\megawiz\memory

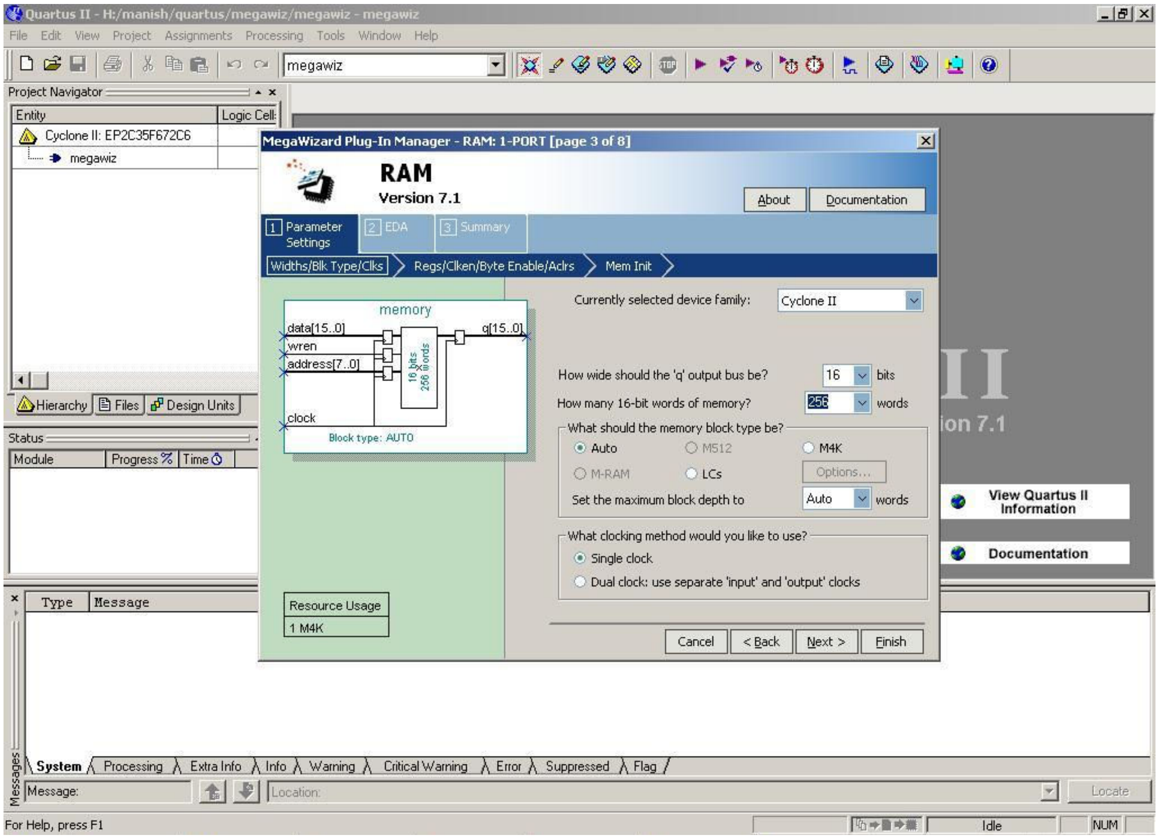
□ Return to this page for another create operation

Note: To compile a project successfully in the Quartus II software, your design files must be in the project directory, in the global user libraries specified in the Options dialog box (Tools menu), or a user library specified in the User Libraries page of the Settings dialog box (Assignments menu).

Your current user library directories are:

Fig. 2

3. On page 3, ‘q’ output bus should be made 16 bits wide and the number of words can be selected as 8192 as shown in the Fig. 3. The number of words you are going to require depend on the length of your program. Though a 16 bit address line can access 2^{16} = 64K locations, we don’t need this much memory. Then click “Next”.



MegaWizard Plug-In Manager - RAM: 1-PORT [page 3 of 8]

RAM
Version 7.1

Currently selected device family: Cyclone II

How wide should the 'q' output bus be? 16 bits

How many 16-bit words of memory? 256 words

What should the memory block type be?

● Auto ○ M512 ○ M4K

○ M-RAM ○ LCs

Options...

Set the maximum block depth to Auto words

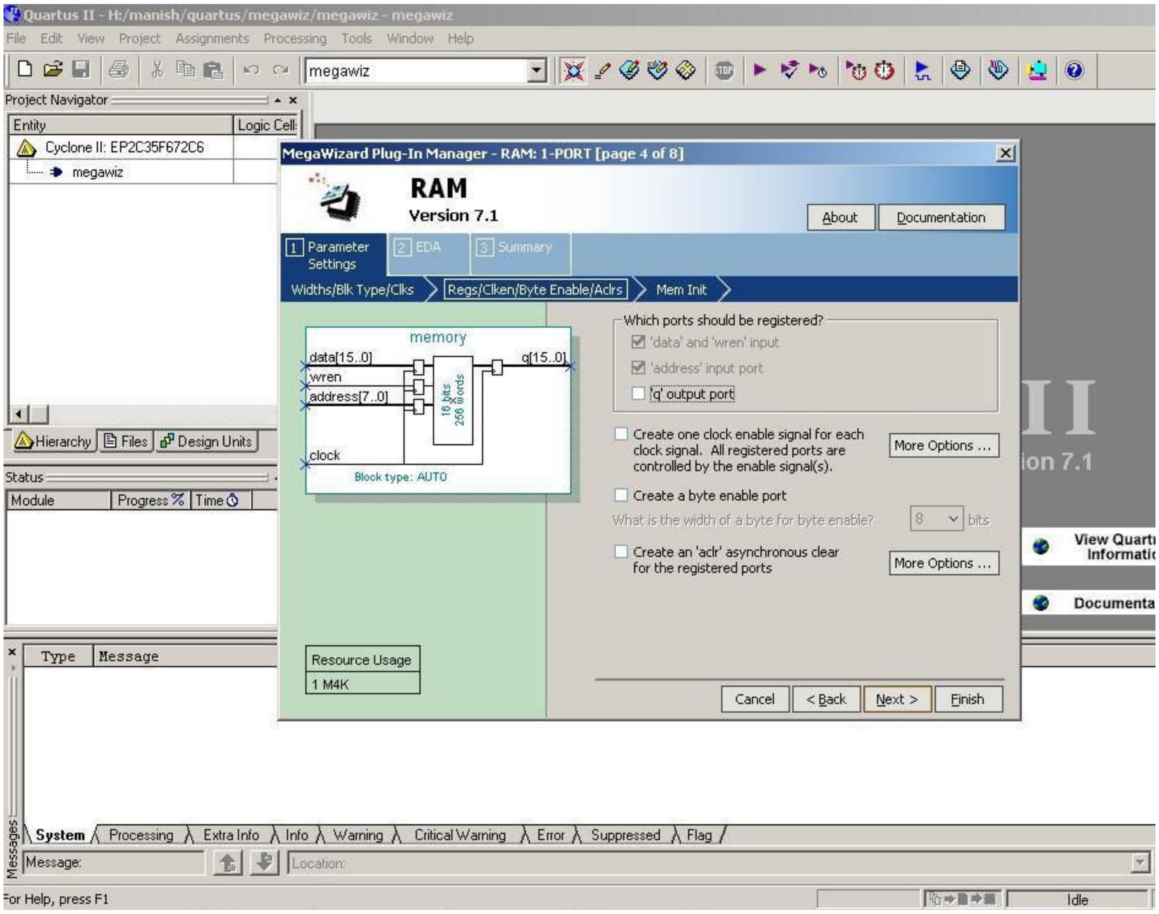
What clocking method would you like to use?

● Single clock

○ Dual clock: use separate 'input' and 'output' clocks

Fig. 3

4. Page 4 will appear as shown in Fig 4 . Uncheck ‘q’ output port . Click Next.



Which ports should be registered?

☑ 'data' and 'wren' input

☑ 'address' input port

□ 'q' output port

□ Create one clock enable signal for each clock signal. All registered ports are controlled by the enable signal(s).

□ Create a byte enable port

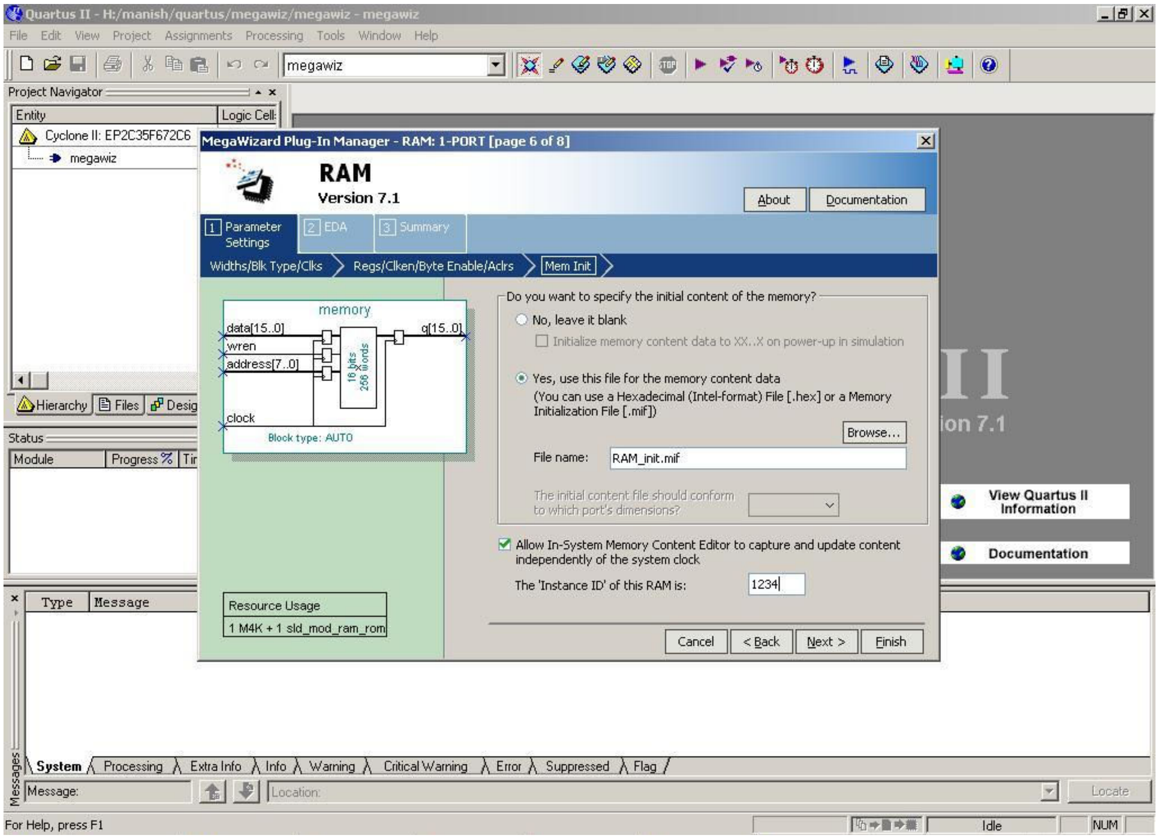
What is the width of a byte for byte enable? 8 bits

□ Create an 'aclr' asynchronous clear for the registered ports

Fig. 4

5. To place processor instructions into the memory, you need to specify *initial values* that should be stored in the memory once your circuit has been programmed into the FPGA chip. This can be done by telling the wizard to initialize the memory using the contents of a *memory initialization file (MIF)*. I have specified a file named *RAM_init.mif*, which then has to be saved in the directory that contains the Quartus II project. MIF format is available on course website. Write your MIF File as shown in that format. And click on "Yes, use this file for memory content data" and add your RAM_init.mif file as shown in Fig. 5. Then click Next.

   To make the memory contents run-time editable, you need to check box against "Allow in system memory content editor to capture and update content independently of the system clock". The Instance ID is any 4 digit number like 1234.



MegaWizard Plug-In Manager - RAM: 1-PORT [page 6 of 8]

RAM
Version 7.1

Do you want to specify the initial content of the memory?

○ No, leave it blank

□ Initialize memory content data to XX..X on power-up in simulation

● Yes, use this file for the memory content data
(You can use a Hexadecimal (Intel-format) File [.hex] or a Memory Initialization File [.mif])

File name: RAM_init.mif

The initial content file should conform to which port's dimensions?

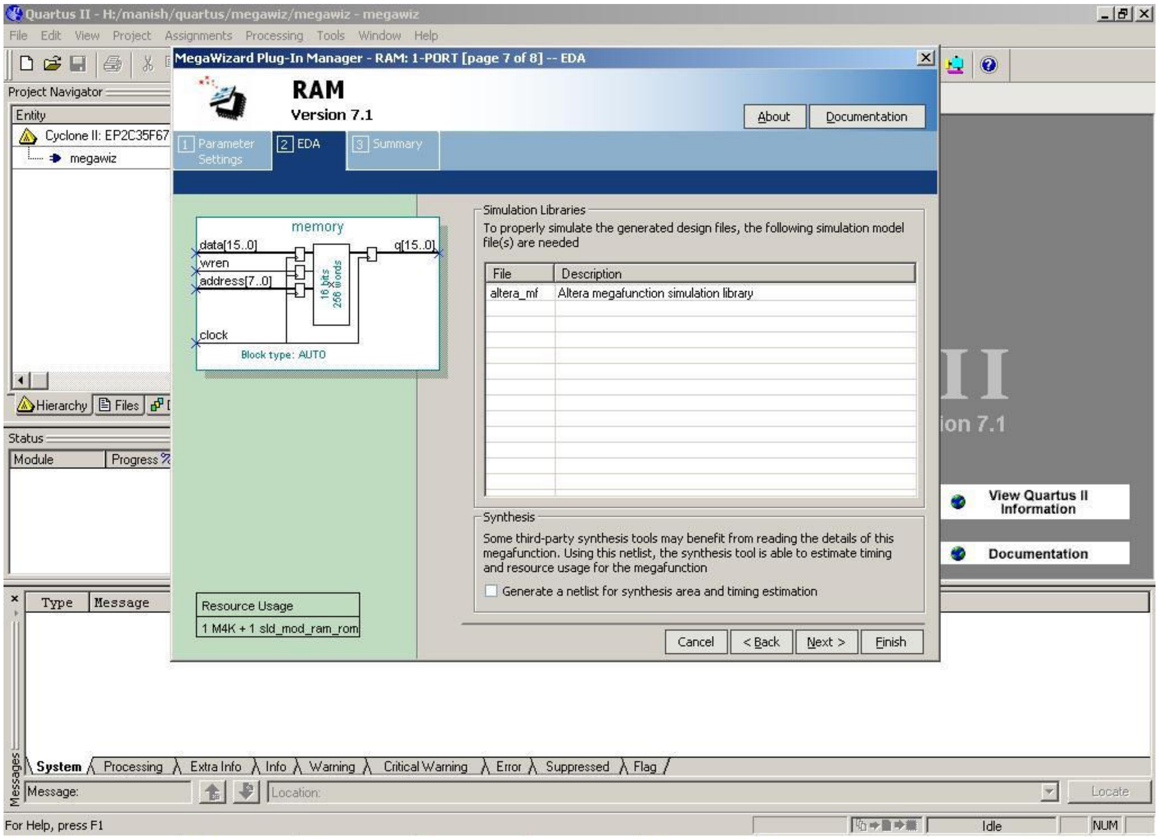
☑ Allow In-System Memory Content Editor to capture and update content independently of the system clock

The 'Instance ID' of this RAM is: 1234

Resource Usage
1 M4K + 1 sld_mod_ram_rom

Fig. 5

6. Finally your .vhd file for RAM module will be created in your working directory. Select Next on page 7. (Fig. 6)



Quartus II - H:/manish/quartus/megawiz/megawiz - megawiz

MegaWizard Plug-In Manager - RAM: 1-PORT [page 7 of 8] -- EDA

RAM

Version 7.1

Simulation Libraries

To properly simulate the generated design files, the following simulation model file(s) are needed

File	Description
altera_mf	Altera megafunction simulation library

Synthesis

Some third-party synthesis tools may benefit from reading the details of this megafunction. Using this netlist, the synthesis tool is able to estimate timing and resource usage for the megafunction

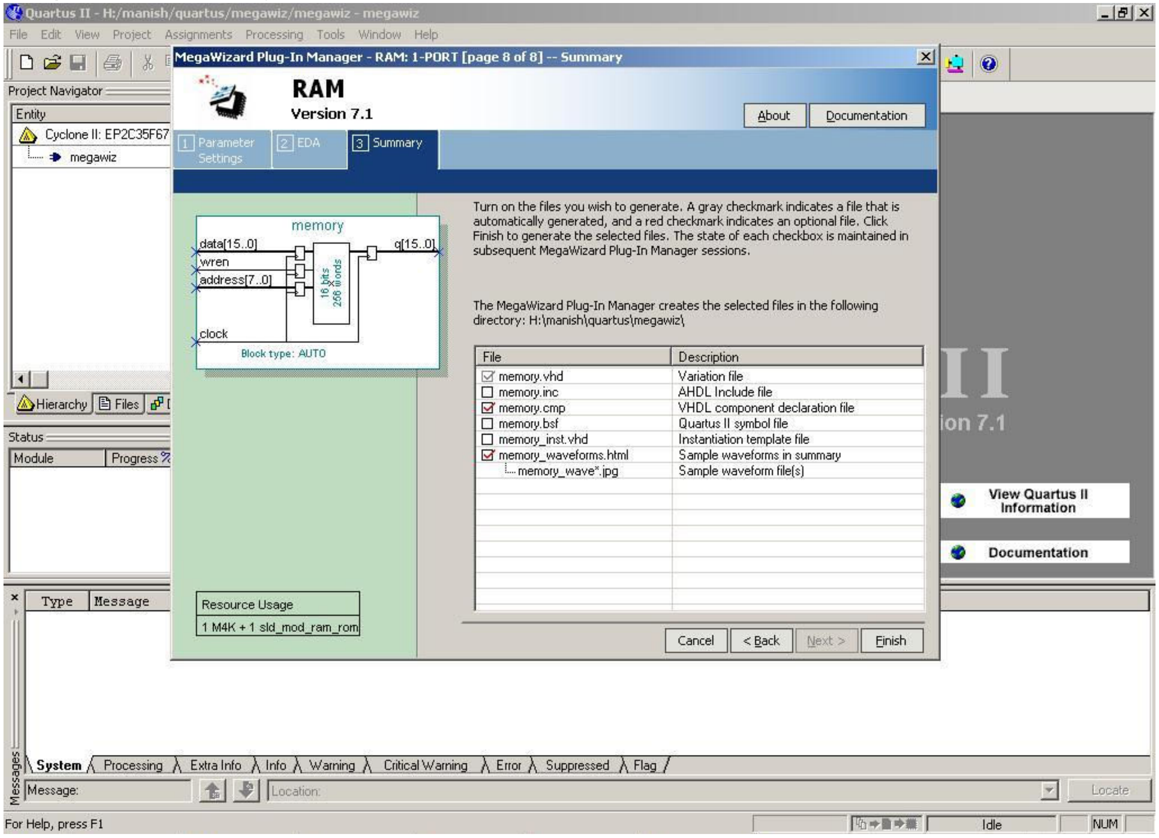
□ Generate a netlist for synthesis area and timing estimation

Resource Usage

1 M4K + 1 sld_mod_ram_rom

Fig. 6

7. Fig. 7 is a summery of all the files that will be created. Click Next.



Turn on the files you wish to generate. A gray checkmark indicates a file that is automatically generated, and a red checkmark indicates an optional file. Click Finish to generate the selected files. The state of each checkbox is maintained in subsequent MegaWizard Plug-In Manager sessions.

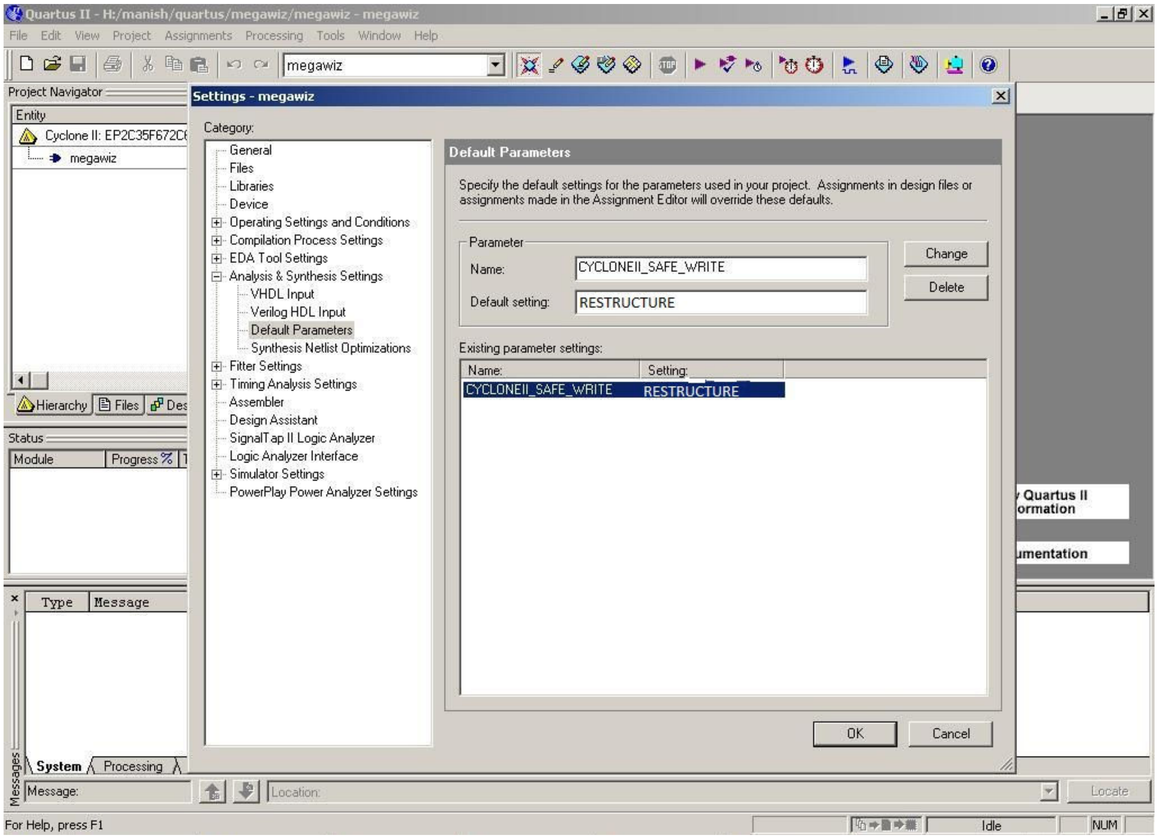
The MegaWizard Plug-In Manager creates the selected files in the following directory: H:\manish\quartus\megawiz\

File	Description
☑ memory.vhd	Variation file
☐ memory.inc	AHDL Include file
☑ memory.cmp	VHDL component declaration file
☐ memory.bsf	Quartus II symbol file
☐ memory_inst.vhd	Instantiation template file
☑ memory_waveforms.html	Sample waveforms in summary
memory_wave*.jpg	Sample waveform file(s)

Resource Usage
1 M4K + 1 sld_mod_ram_rom

Fig. 7

8. Before you can use the In-System Memory Content Editor tool, one additional setting has to be made. In the Quartus II software select Assignments > Settings to open the window in Fig. 8, and then open the item called Default Parameters under Analysis and Synthesis Settings. As shown in the figure, type the parameter name CYCLONEII_SAFE_WRITE and assign the value RESTRUCTURE. This parameter allows the Quartus II synthesis tools to modify the single-port RAM as needed to allow reading and writing of the memory by the In-System Memory Content Editor tool. Click OK to exit from the Settings window.



Quartus II - H:/manish/quartus/megawiz/megawiz - megawiz

Settings - megawiz

Default Parameters

Specify the default settings for the parameters used in your project. Assignments in design files or assignments made in the Assignment Editor will override these defaults.

Parameter

Name: CYCLONEII_SAFE_WRITE

Default setting: RESTRUCTURE

Existing parameter settings:

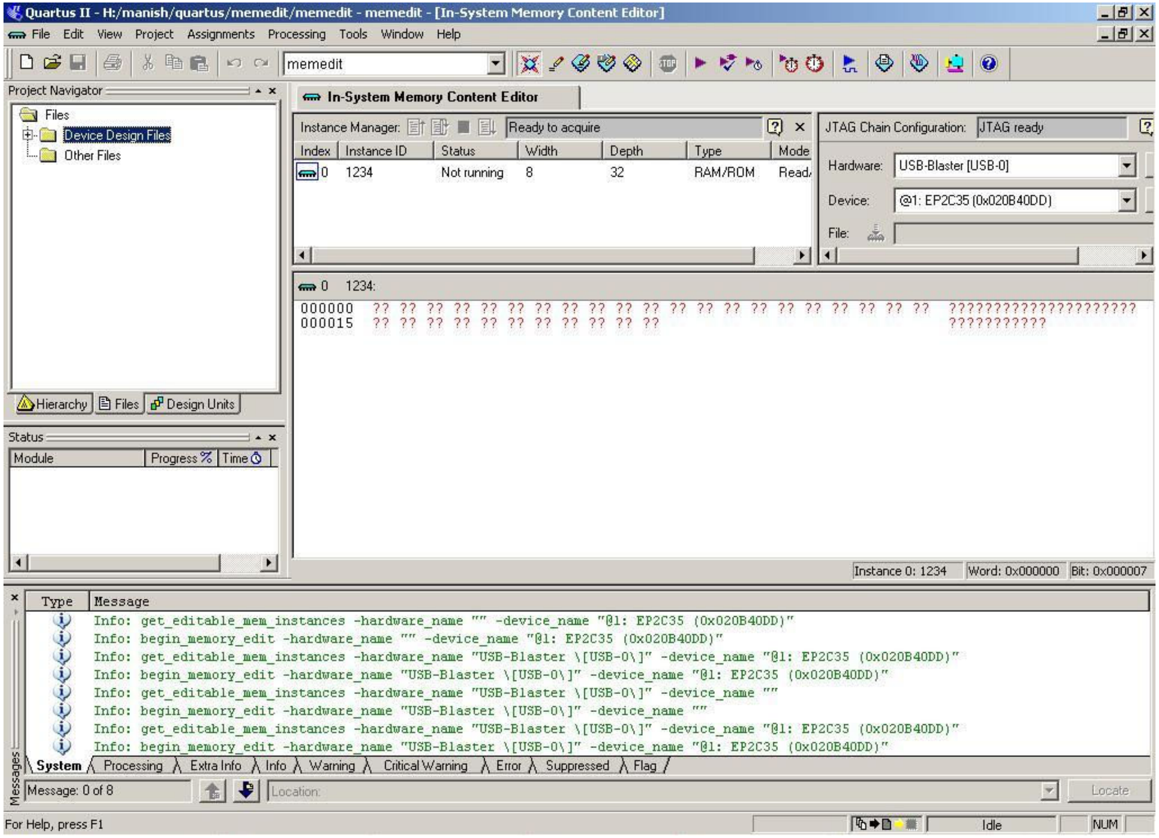
Name:	Setting:
CYCLONEII_SAFE_WRITE	RESTRUCTURE

Fig. 8

9. Now use this memory in your design by port mapping. Compile your code and download the circuit onto the DE2 board as instructed in the tutorial 'Altera Quartus II and DE2 manual' available on course website.

10. After downloading the design in FPGA, Select Tools > In-System Memory Content Editor, which opens the window in Fig. 9. To specify the connection to your DE2 board click on the Setup button on the right side of the screen. In the window in Figure 9 select the USB-Blaster hardware, and then close the Hardware Setup dialog.
    <u>Note</u>: While doing steps 10 and 11 the FPGA board should be powered up and should be in RUN mode.
    a. You can observe that the device ID specified appears in the instance manager.
    b. The status is ready to acquire and we can see the '??' indicating that the memory has not been read yet.

Select the memory in Instance manager and click the read data from system memory button (one with a red box) as shown in Fig. 10. Now you can see the contents in the memory.



Quartus II - H:/manish/quartus/memedit/memedit - memedit - [In-System Memory Content Editor]

Instance Manager: Ready to acquire

Index	Instance ID	Status	Width	Depth	Type	Mode
0	1234	Not running	8	32	RAM/ROM	Read/

JTAG Chain Configuration: JTAG ready

Hardware: USB-Blaster [USB-0]

Device: @1: EP2C35 (0x020B40DD)

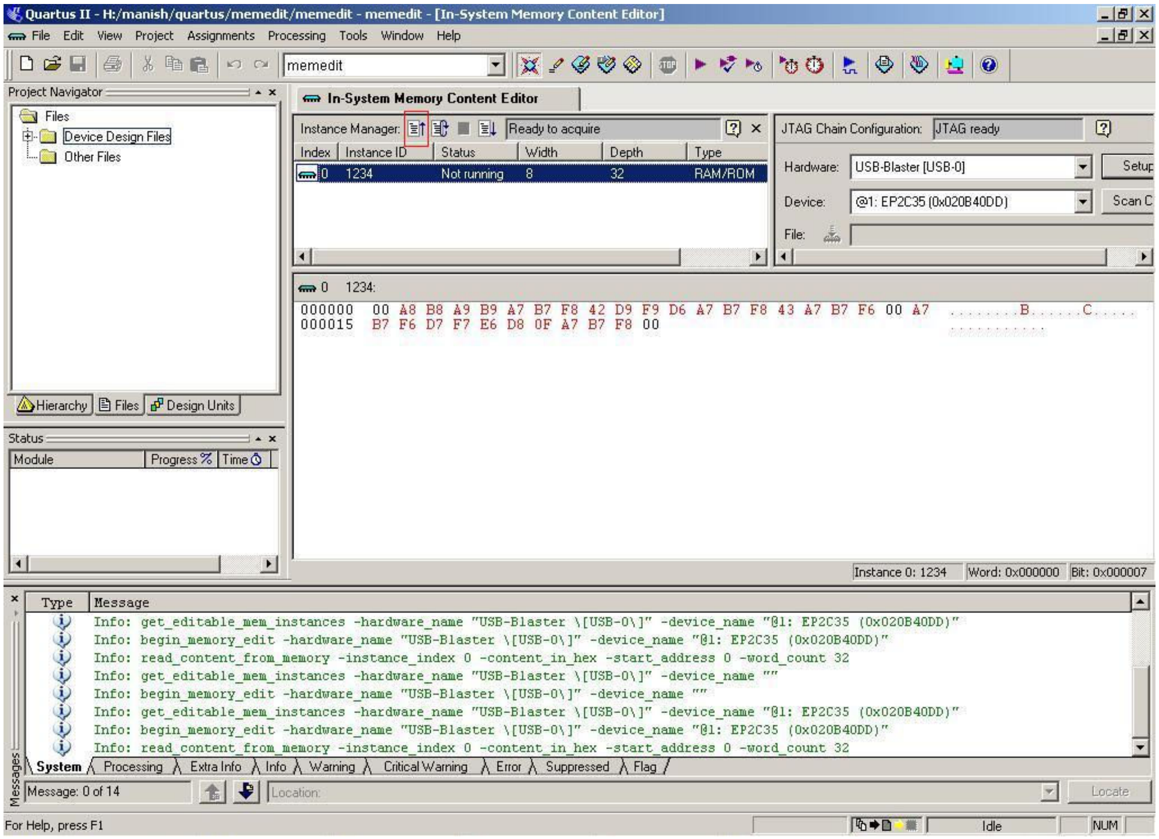
File:

```
0   1234:
000000  ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ??   ?????????????????????
000015  ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ??                                  ???????????
```

Instance 0: 1234 Word: 0x000000 Bit: 0x000007

```
Type  Message
      Info: get_editable_mem_instances -hardware_name "" -device_name "@1: EP2C35 (0x020B40DD)"
      Info: begin_memory_edit -hardware_name "" -device_name "@1: EP2C35 (0x020B40DD)"
      Info: get_editable_mem_instances -hardware_name "USB-Blaster \[USB-0\]" -device_name "@1: EP2C35 (0x020B40DD)"
      Info: begin_memory_edit -hardware_name "USB-Blaster \[USB-0\]" -device_name "@1: EP2C35 (0x020B40DD)"
      Info: get_editable_mem_instances -hardware_name "USB-Blaster \[USB-0\]" -device_name ""
      Info: begin_memory_edit -hardware_name "USB-Blaster \[USB-0\]" -device_name ""
      Info: get_editable_mem_instances -hardware_name "USB-Blaster \[USB-0\]" -device_name "@1: EP2C35 (0x020B40DD)"
      Info: begin_memory_edit -hardware_name "USB-Blaster \[USB-0\]" -device_name "@1: EP2C35 (0x020B40DD)"
```

Fig. 9



Quartus II - H:/manish/quartus/memedit/memedit - memedit - [In-System Memory Content Editor]

Instance Manager: Ready to acquire

Index	Instance ID	Status	Width	Depth	Type
0	1234	Not running	8	32	RAM/ROM

JTAG Chain Configuration: JTAG ready

Hardware: USB-Blaster [USB-0]

Device: @1: EP2C35 (0x020B40DD)

File:

0 1234:

```
000000  00 A8 B8 A9 B9 A7 B7 F8 42 D9 F9 D6 A7 B7 F8 43 A7 B7 F6 00 A7   ........B......C.....
000015  B7 F6 D7 F7 E6 D8 0F A7 B7 F8 00                                 ...........
```

Instance 0: 1234 Word: 0x000000 Bit: 0x000007

```
Type  Message
      Info: get_editable_mem_instances -hardware_name "USB-Blaster \[USB-0\]" -device_name "@1: EP2C35 (0x020B40DD)"
      Info: begin_memory_edit -hardware_name "USB-Blaster \[USB-0\]" -device_name "@1: EP2C35 (0x020B40DD)"
      Info: read_content_from_memory -instance_index 0 -content_in_hex -start_address 0 -word_count 32
      Info: get_editable_mem_instances -hardware_name "USB-Blaster \[USB-0\]" -device_name ""
      Info: begin_memory_edit -hardware_name "USB-Blaster \[USB-0\]" -device_name ""
      Info: get_editable_mem_instances -hardware_name "USB-Blaster \[USB-0\]" -device_name "@1: EP2C35 (0x020B40DD)"
      Info: begin_memory_edit -hardware_name "USB-Blaster \[USB-0\]" -device_name "@1: EP2C35 (0x020B40DD)"
      Info: read_content_from_memory -instance_index 0 -content_in_hex -start_address 0 -word_count 32
```

Message: 0 of 14

Location:

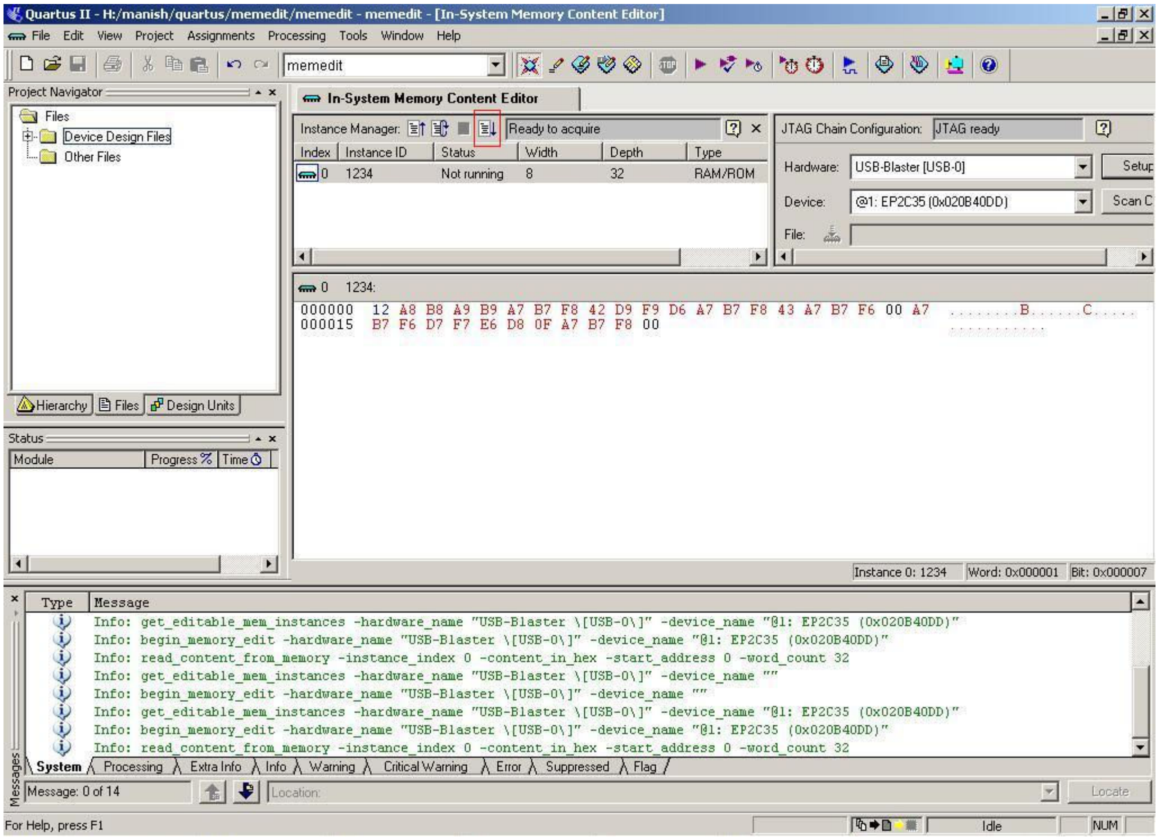
For Help, press F1

Idle

NUM

Fig. 10

11. The contents can be modified by directly overwriting them as shown in Fig. 11 at location 000000 we have changed the contents from 00 to 12. This is just a change made in the editor. To actually write the new value in the memory you can click the 'Write' button as highlighted in Fig. 11 with a red box.



Quartus II - H:/manish/quartus/memedit/memedit - memedit - [In-System Memory Content Editor]

memedit

Instance Manager: Ready to acquire

Index	Instance ID	Status	Width	Depth	Type
0	1234	Not running	8	32	RAM/ROM

JTAG Chain Configuration: JTAG ready

Hardware: USB-Blaster [USB-0]

Device: @1: EP2C35 (0x020B40DD)

File:

```
0   1234:
000000  12 A8 B8 A9 B9 A7 B7 F8 42 D9 F9 D6 A7 B7 F8 43 A7 B7 F6 00 A7   ........B......C.....
000015  B7 F6 D7 F7 E6 D8 0F A7 B7 F8 00                                 ...........
```

Instance 0: 1234 | Word: 0x000001 | Bit: 0x000007

```
Type  Message
      Info: get_editable_mem_instances -hardware_name "USB-Blaster \[USB-0\]" -device_name "@1: EP2C35 (0x020B40DD)"
      Info: begin_memory_edit -hardware_name "USB-Blaster \[USB-0\]" -device_name "@1: EP2C35 (0x020B40DD)"
      Info: read_content_from_memory -instance_index 0 -content_in_hex -start_address 0 -word_count 32
      Info: get_editable_mem_instances -hardware_name "USB-Blaster \[USB-0\]" -device_name ""
      Info: begin_memory_edit -hardware_name "USB-Blaster \[USB-0\]" -device_name ""
      Info: get_editable_mem_instances -hardware_name "USB-Blaster \[USB-0\]" -device_name "@1: EP2C35 (0x020B40DD)"
      Info: begin_memory_edit -hardware_name "USB-Blaster \[USB-0\]" -device_name "@1: EP2C35 (0x020B40DD)"
      Info: read_content_from_memory -instance_index 0 -content_in_hex -start_address 0 -word_count 32
```

Message: 0 of 14

Fig. 11

12. After changing the contents you don't have to recompile your entire design in the FPGA. Just reset your circuit and now the design will operate on the changed memory contents. This run time memory content editor can be used to change one or two operands in your sample program or you can change the entire program as well.