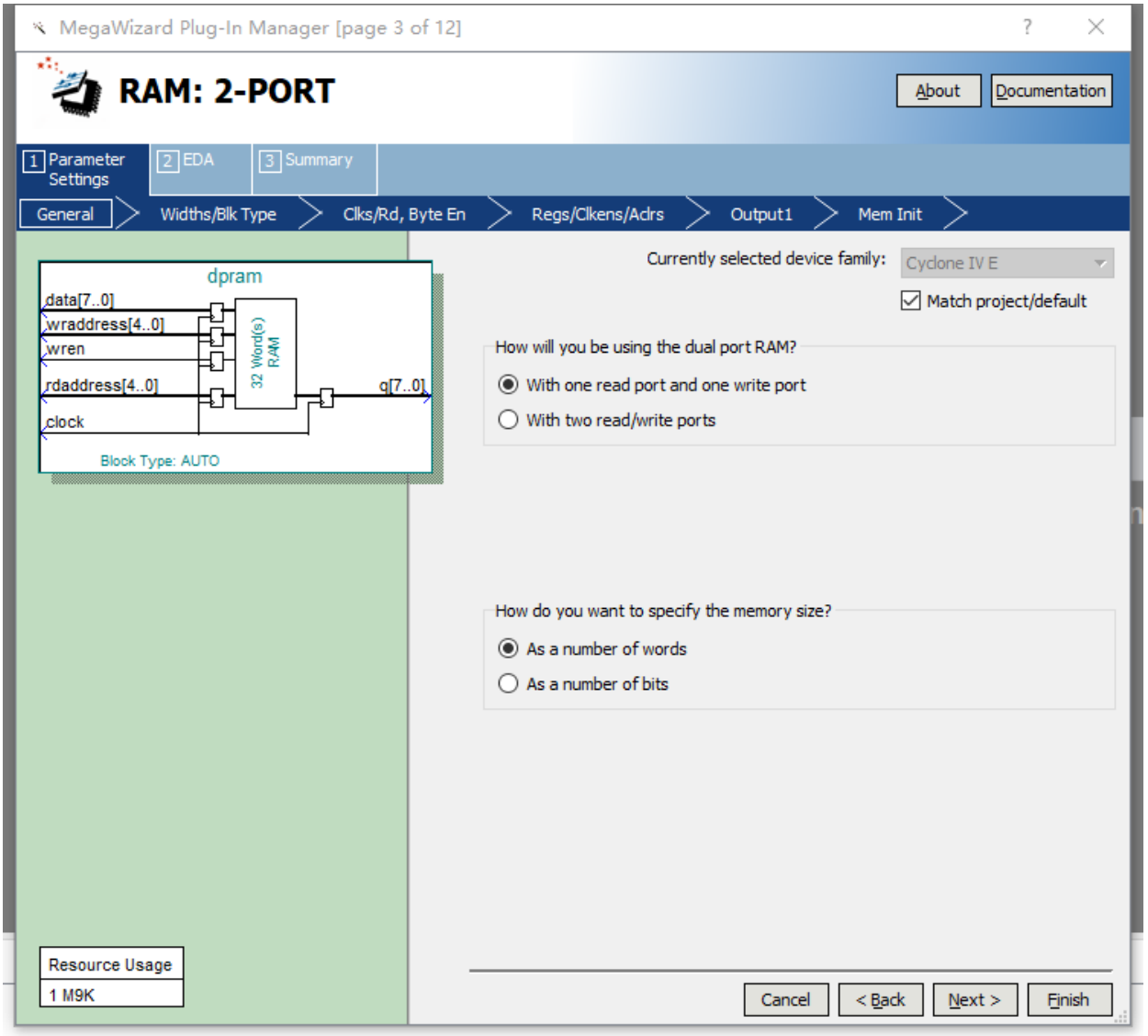
小梅哥实验：通过串口发送数据到FPGA中，FPGA接收到数据后将数据存储在双口RAM的一段连续空间中，通过quartus ii 软件提供的In-System Memory Content Editor 工具查看RAM中接收到的数据，当需要时，按下按键0，则FPGA将RAM中存储的数据通过串口发送出去。



简单双口RAM：有一组读数据和读地址线，一组写数据和写地址线，所以能同时进行读和写操作，但不能同时对同一地址进行读和写操作。

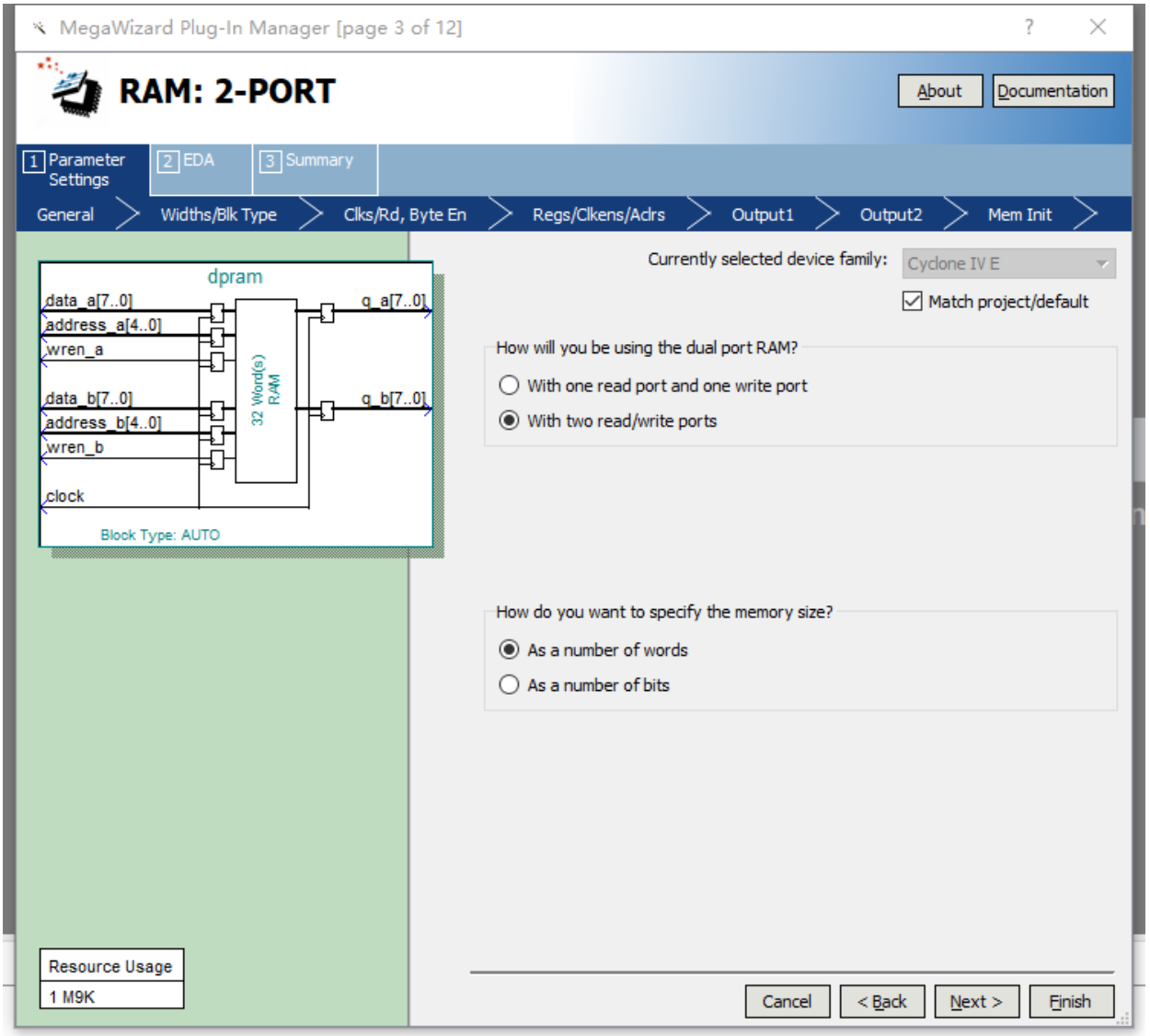
data[7:0]：数据写入端口

wraddress[4:0]：写入地址

wren:写请求信号

rdaddress[4:0]：读出地址

读取数据不需要请求信号



真双口RAM：有两组数据线和地址线；能同时进行两个端口读；能同时进行两个端口写；

能同时一个端口读同时另一个端口写；

真双口RAM其实就是两个单口RAM组合在一起构成的，只是真双口RAM里的两个单口RAM是操作的同一片存储空间。我的理解：真双口RAM中没有读地址，应该是与写地址共用地址。

with two read/write ports:有两个读和两个写端口，共四端口

data_a[7:0]：数据写入端口a

address_a[4:0]：写入地址a

wren_a:写请求信号a

q_a[7:0]:a端口的输出

data_b[7:0]：数据写入端口b

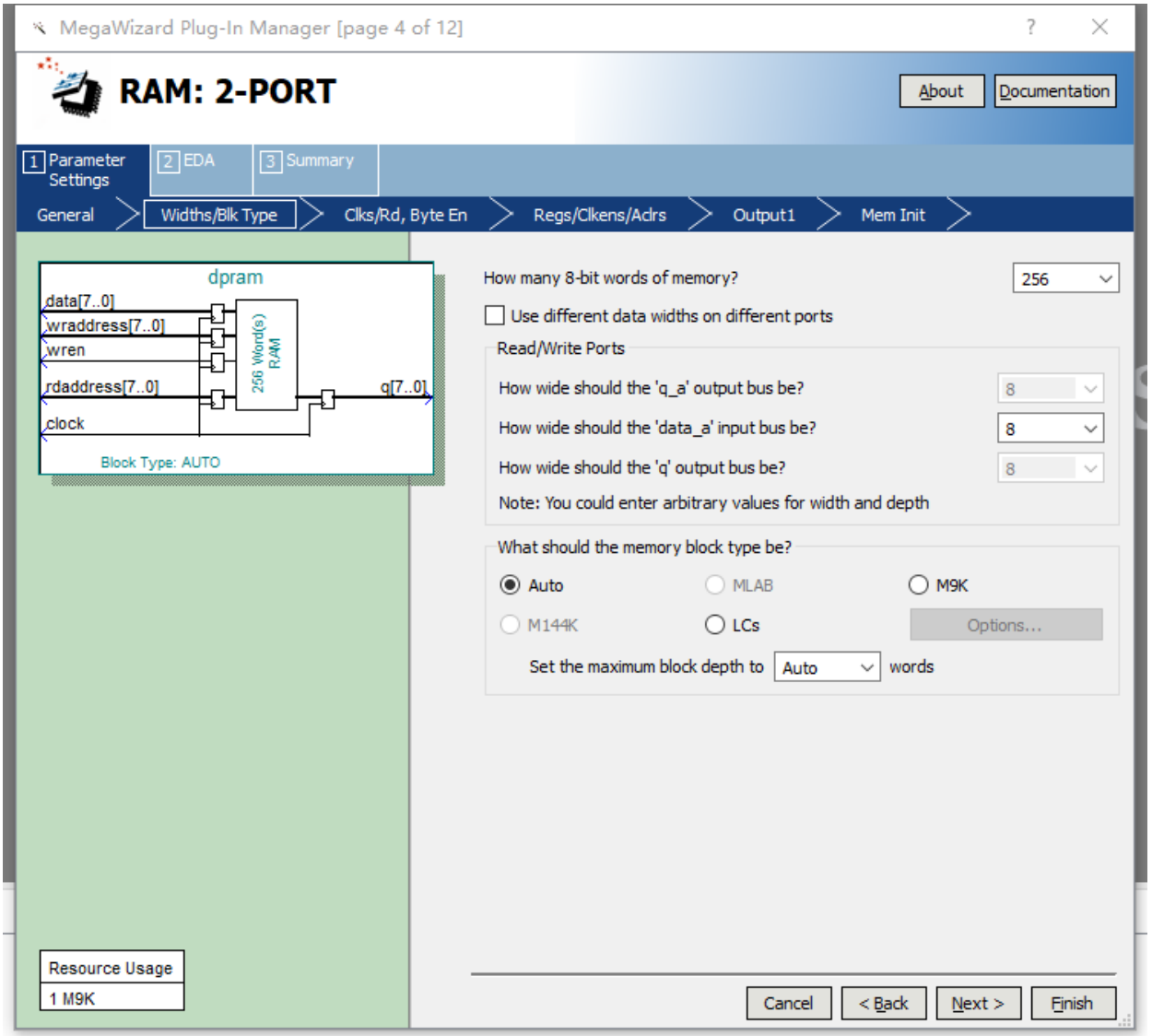
address_b[4:0]：写入地址b

wren_b:写请求信号b

q_b[7:0]:b端口的输出

clock ： 时钟信号

实验中选择的是：with one read port and one write port



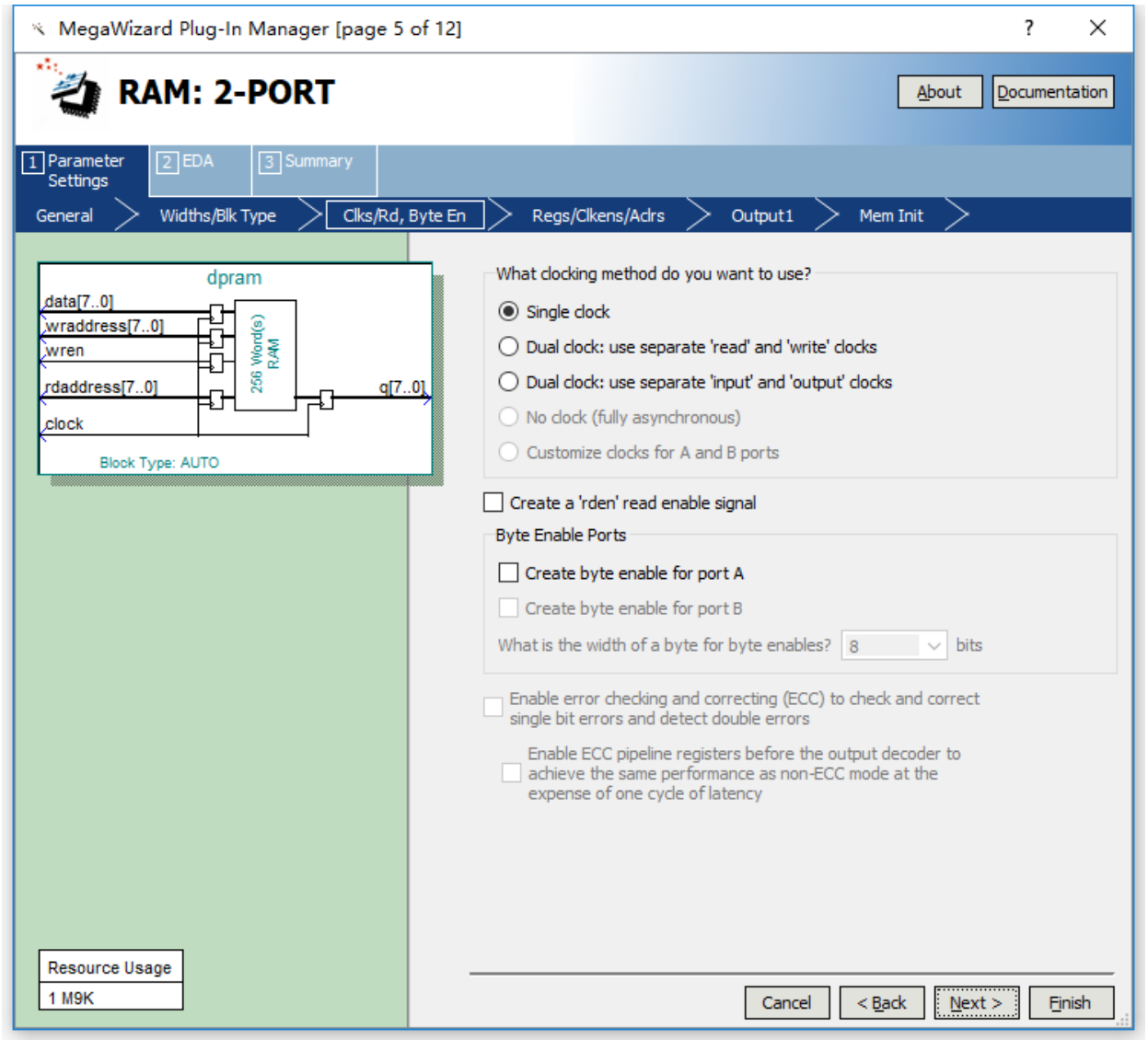
Use different data widths on different ports:可以设置不同端口和不同的数据长度

What should the memory block type be?

Auto:自动分配

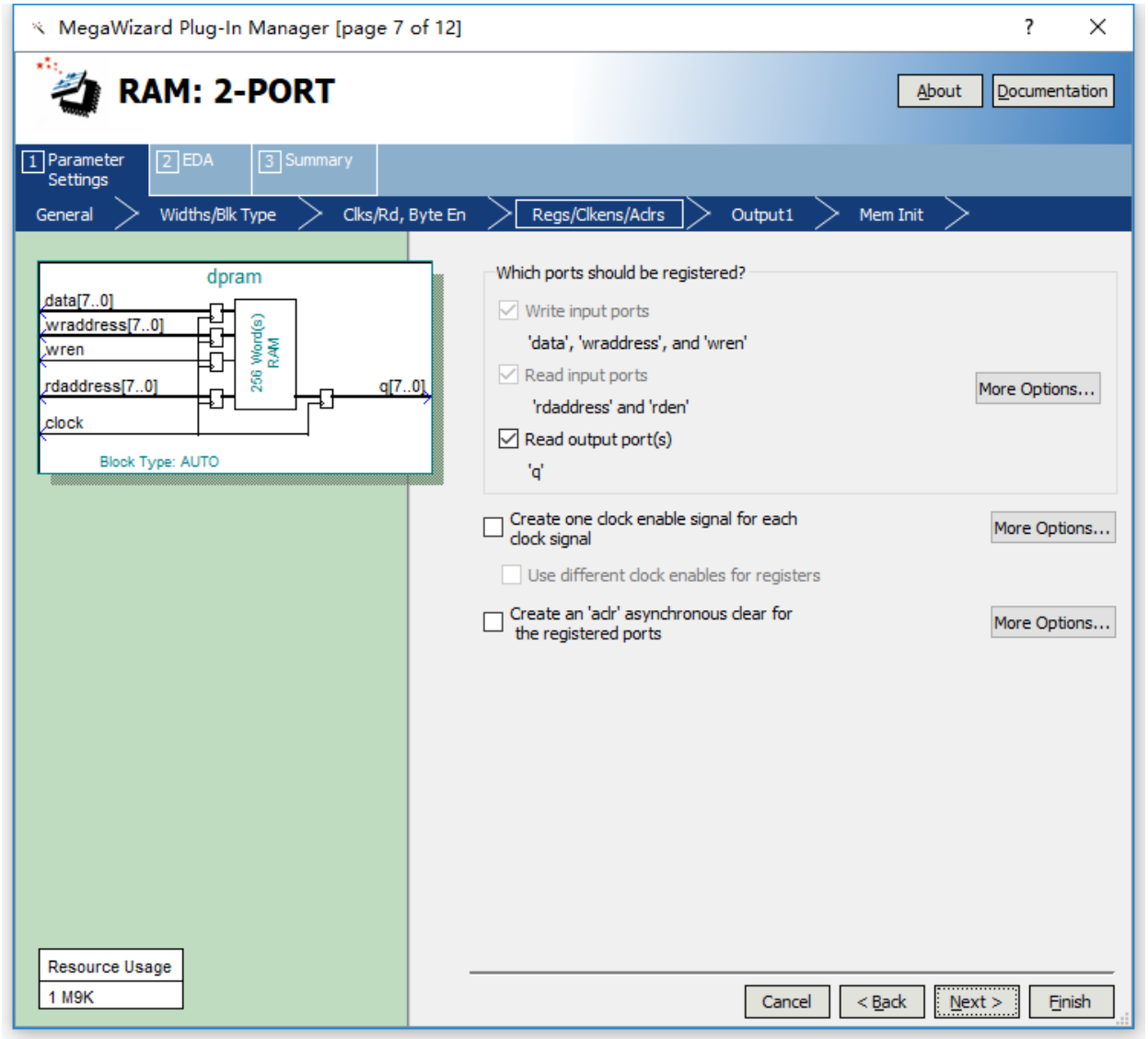
M9K:内部块RAM

LCs:内部寄存器



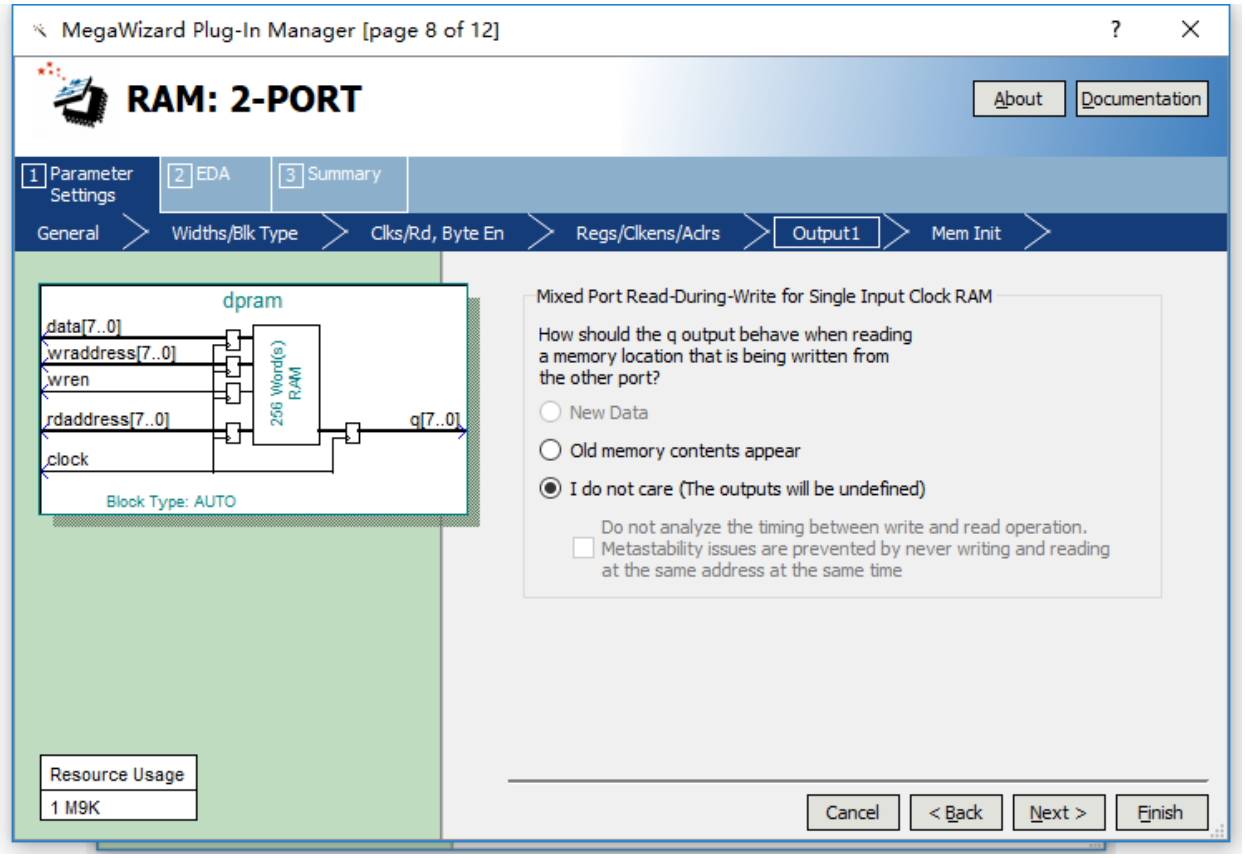
single clock 写入和读出使用同一个时钟

Create a 'rden' read enable signal：可以添加读使能信号

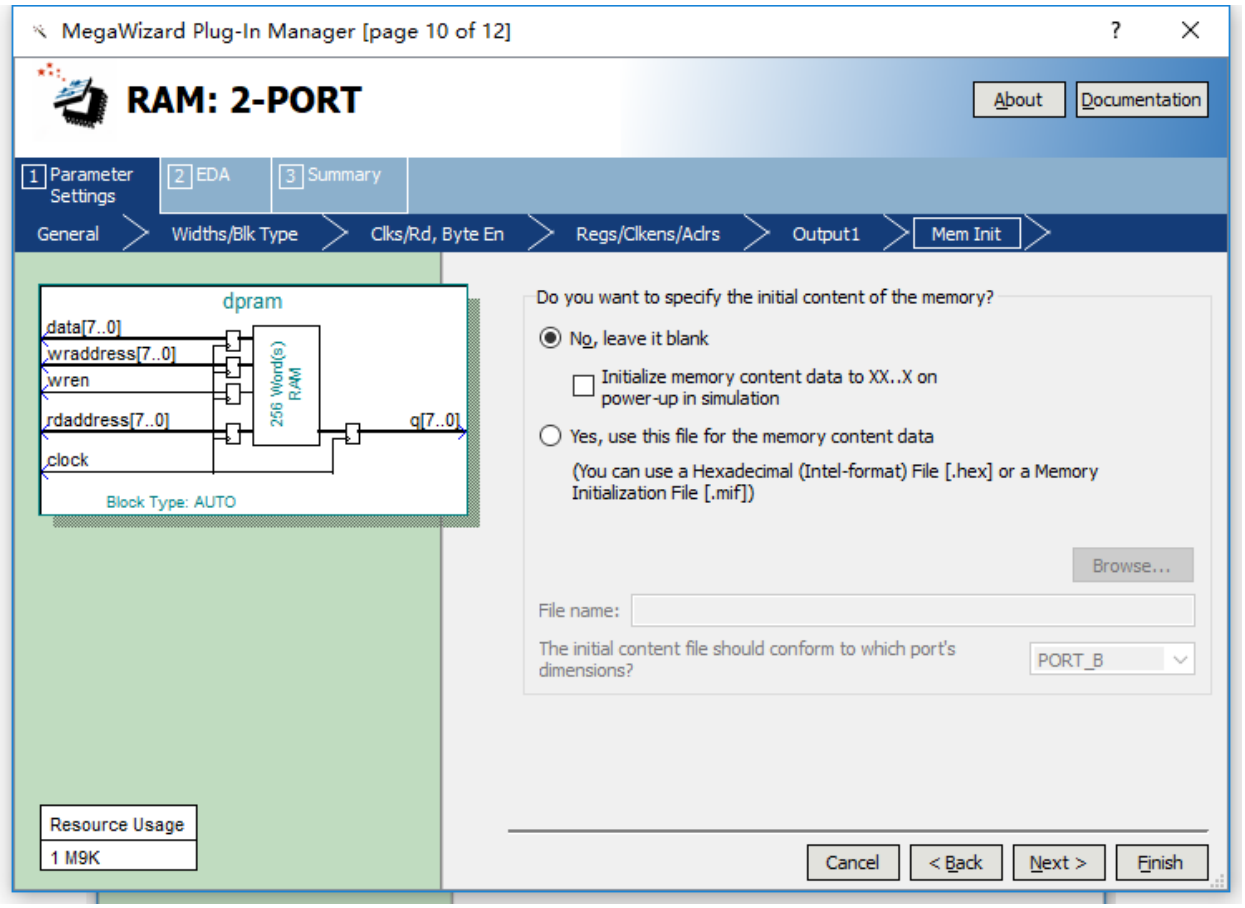


输入输出都是先通过寄存器后在输入输出的，输入寄存器是默认存在的，输出寄存器可以去掉

Create an 'aclr' asynchronous clear for the registerd ports：创建可以清除的寄存器的信号，只是清零寄存器，不会清零ram



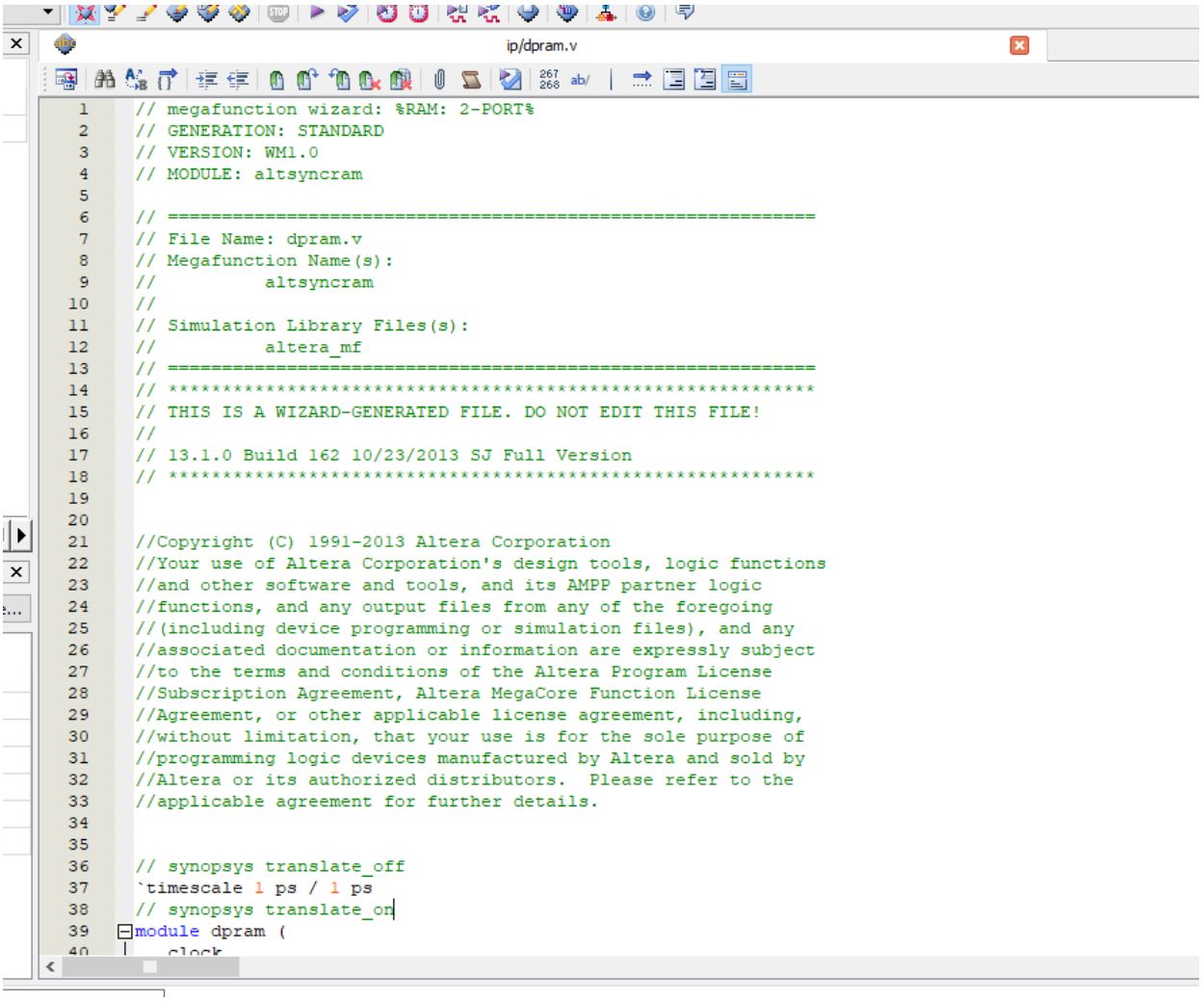
当写入的时候如果此时读取，因为只有一个时钟，所以读取的数据不确定，此处有两种选择：1读出的是老数据；2不关心读出的是啥



需不需要对这个ram进行初始化，即在编译时就写入这个ram数据

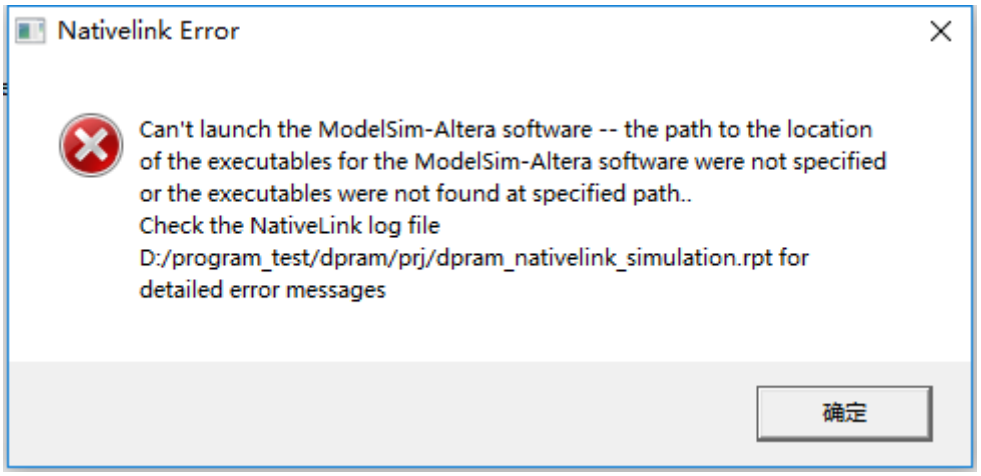
rom必须使用mif文件初始化

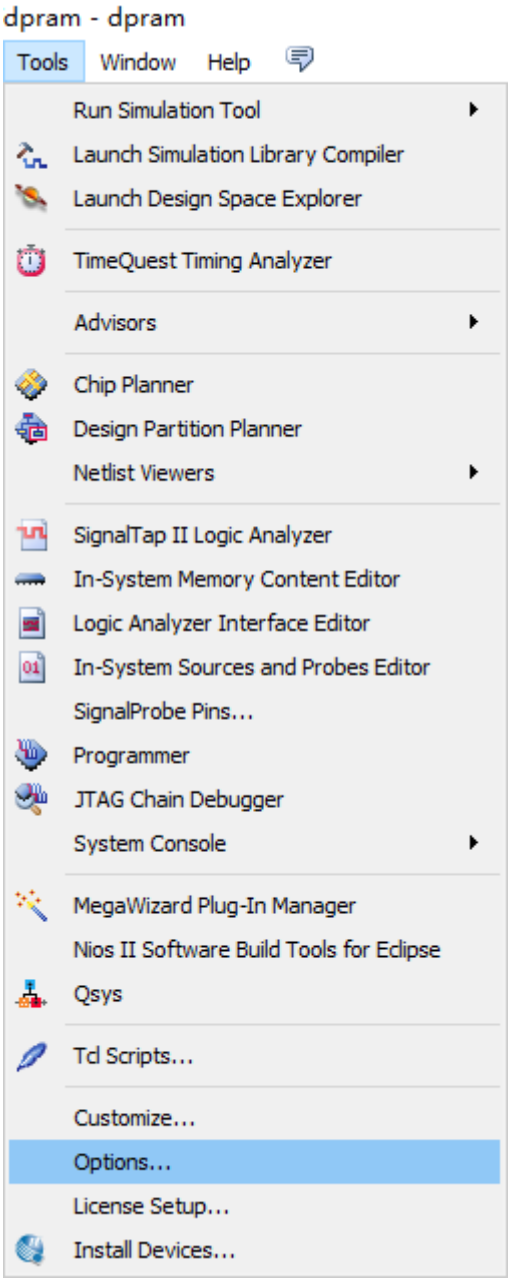
ctrl + O，打开dpram.v文件，这个就是刚才生成的IP核的源代码

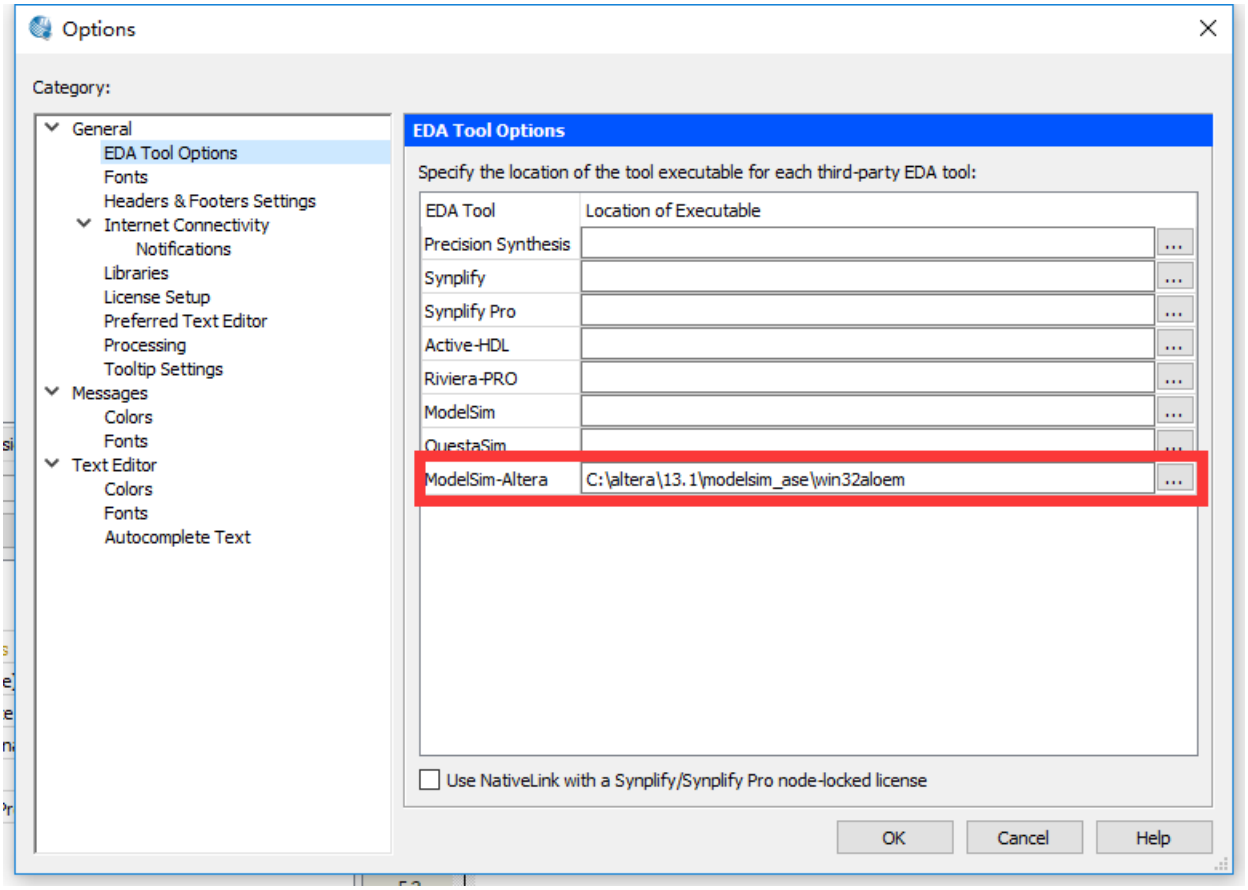


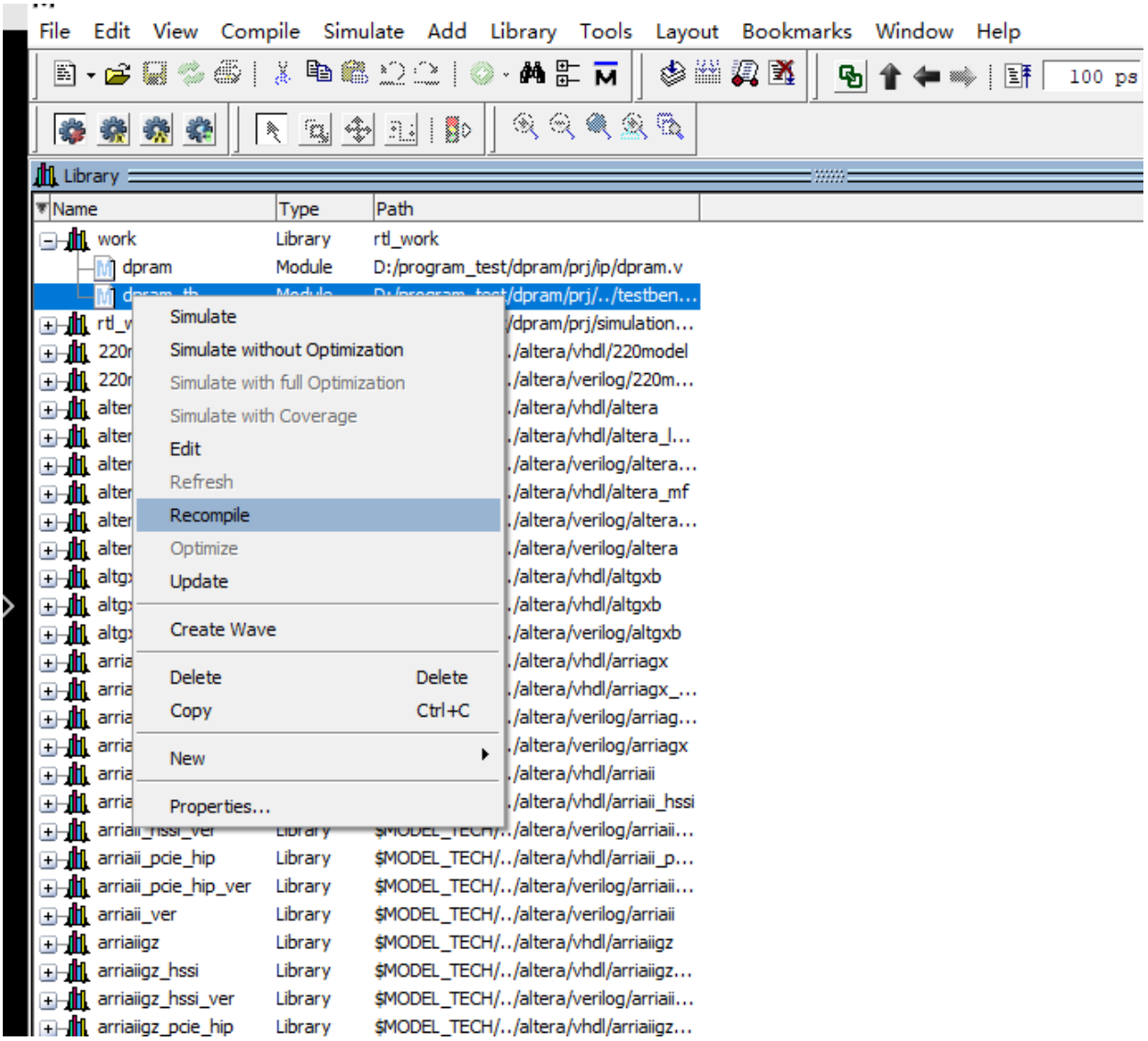
接下来是建立testbench文件对此IP核进行仿真。

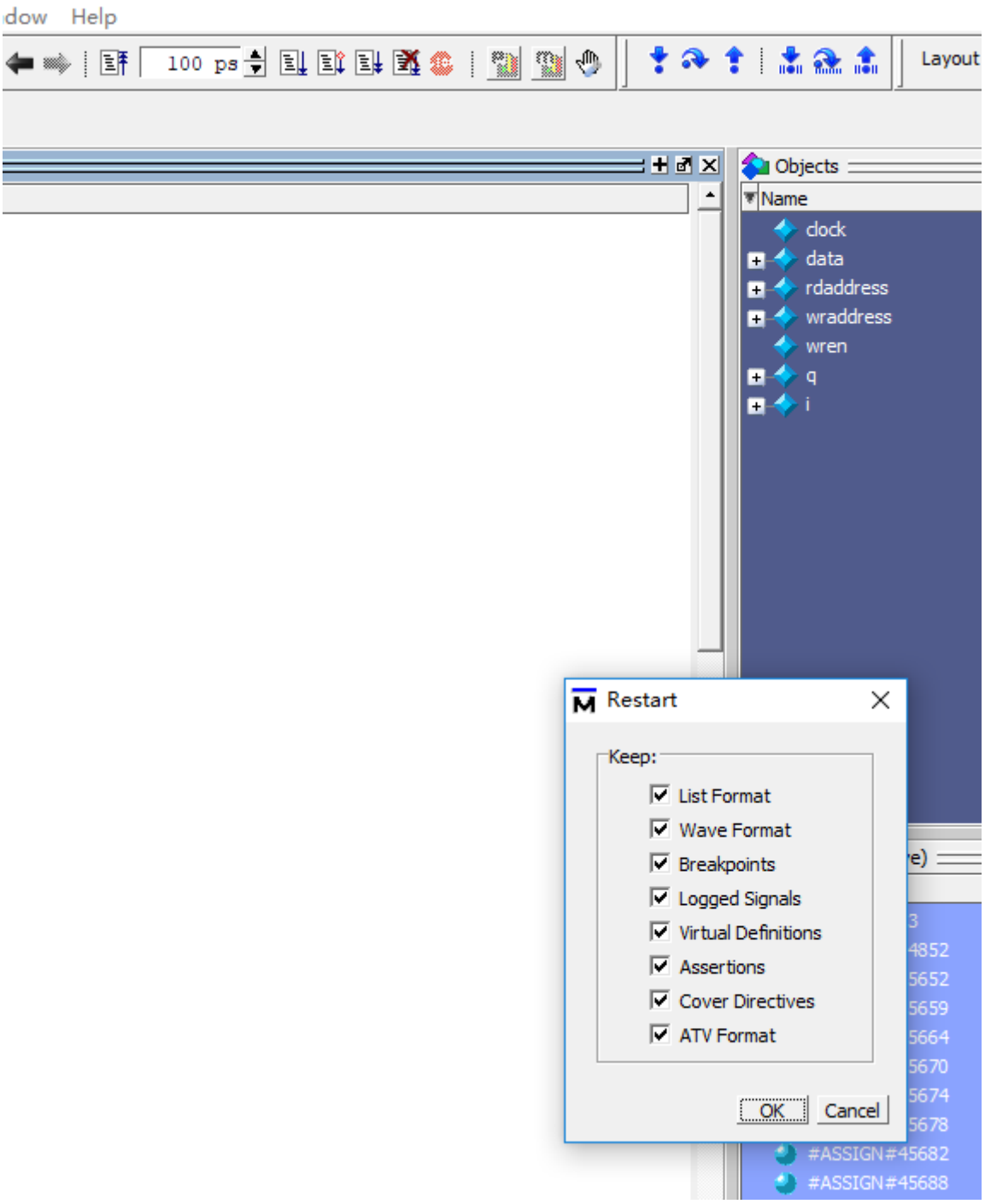
路径问题




dpram - dpram
Tools
Window
Help
Run Simulation Tool
Launch Simulation Library Compiler
Launch Design Space Explorer
TimeQuest Timing Analyzer
Advisors
Chip Planner
Design Partition Planner
Netlist Viewers
SignalTap II Logic Analyzer
In-System Memory Content Editor
Logic Analyzer Interface Editor
In-System Sources and Probes Editor
SignalProbe Pins...
Programmer
JTAG Chain Debugger
System Console
MegaWizard Plug-In Manager
Nios II Software Build Tools for Eclipse
Qsys
Tcl Scripts...
Customize...
Options...
License Setup...
Install Devices...


Options
Category:
General
EDA Tool Options
Fonts
Headers & Footers Settings
Internet Connectivity
Notifications
Libraries
License Setup
Preferred Text Editor
Processing
Tooltip Settings
Messages
Colors
Fonts
Text Editor
Colors
Fonts
Autocomplete Text
EDA Tool Options
Specify the location of the tool executable for each third-party EDA tool:
EDA Tool
Location of Executable
Precision Synthesis
Synplify
Synplify Pro
Active-HDL
Riviera-PRO
ModelSim
QuestaSim
ModelSim-Altera
C:\altera\13.1\modelsim_ase\win32aloem
Use NativeLink with a Synplify/Synplify Pro node-locked license
OK
Cancel
Help


File Edit View Compile Simulate Add Library Tools Layout Bookmarks Window Help
100 ps
Library
Name
Type
Path
work
Library
rtl_work
dpram
Module
D:/program_test/dpram/prj/ip/dpram.v
Simulate
Simulate without Optimization
Simulate with full Optimization
Simulate with Coverage
Edit
Refresh
Recompile
Optimize
Update
Create Wave
Delete
Delete
Copy
Ctrl+C
New
Properties...
arriaii_pcie_hip
Library
arriaii_pcie_hip_ver
Library
arriaii_ver
Library
arriaiigz
Library
arriaiigz_hssi
Library
arriaiigz_hssi_ver
Library
arriaiigz_pcie_hip
Library



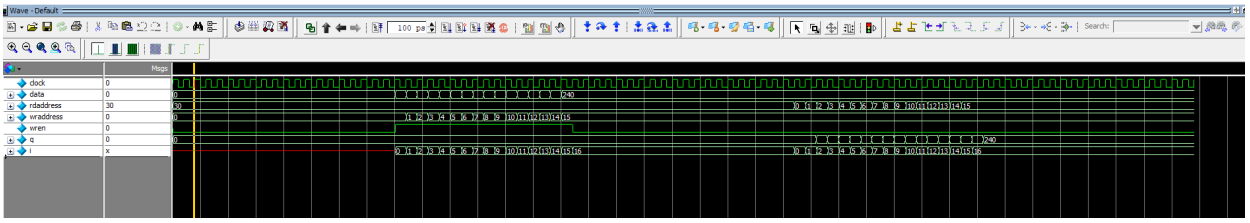
测试脚本：

```
`timescale 1ns/1ns
`define clk_period 20

module dpram_tb;

//---输入信号
reg clock;
reg [7:0]data;//写入端口
```

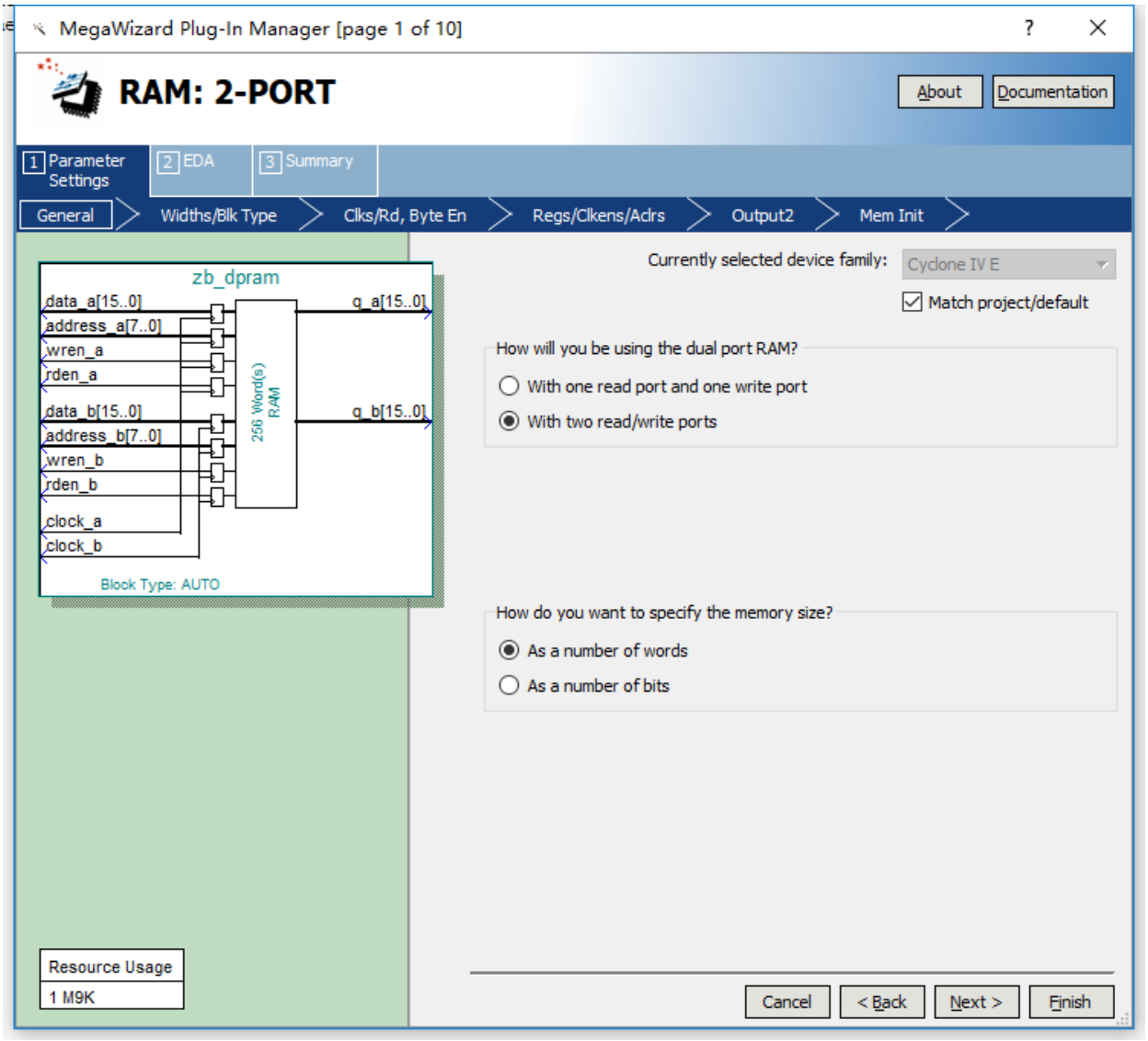
```
reg [7:0]rdaddress;//读出地址
reg [7:0]wraddress;//写入地址
reg wren;//写使能信号

wire [7:0]q;//输出信号
integer i; //注意for循环的写法

dpram dpram0(
	.clock(clock),
	.data(data),
	.rdaddress(rdaddress),
	.wraddress(wraddress),
	.wren(wren),
	.q(q)
);

initial clock = 1;
always#(`clk_period/2)clock = ~clock;

initial begin
	data = 0;
	rdaddress = 30;
	wraddress = 0;
	wren = 0;
	#(`clk_period*20 + 1);//初始化完成后延时一段时间
	/*进行数据写入仿真*/
	for(i=0;i<=15;i=i+1)//不能写i++，只能写i=i+1
	begin
		wren = 1;//数据开始写入
		data =255 - i;//数据为255-i
		wraddress = i;//写入的地址为i
		#`clk_period;
	end
	wren = 0;//数据写入完成
	#(`clk_period*20);
```

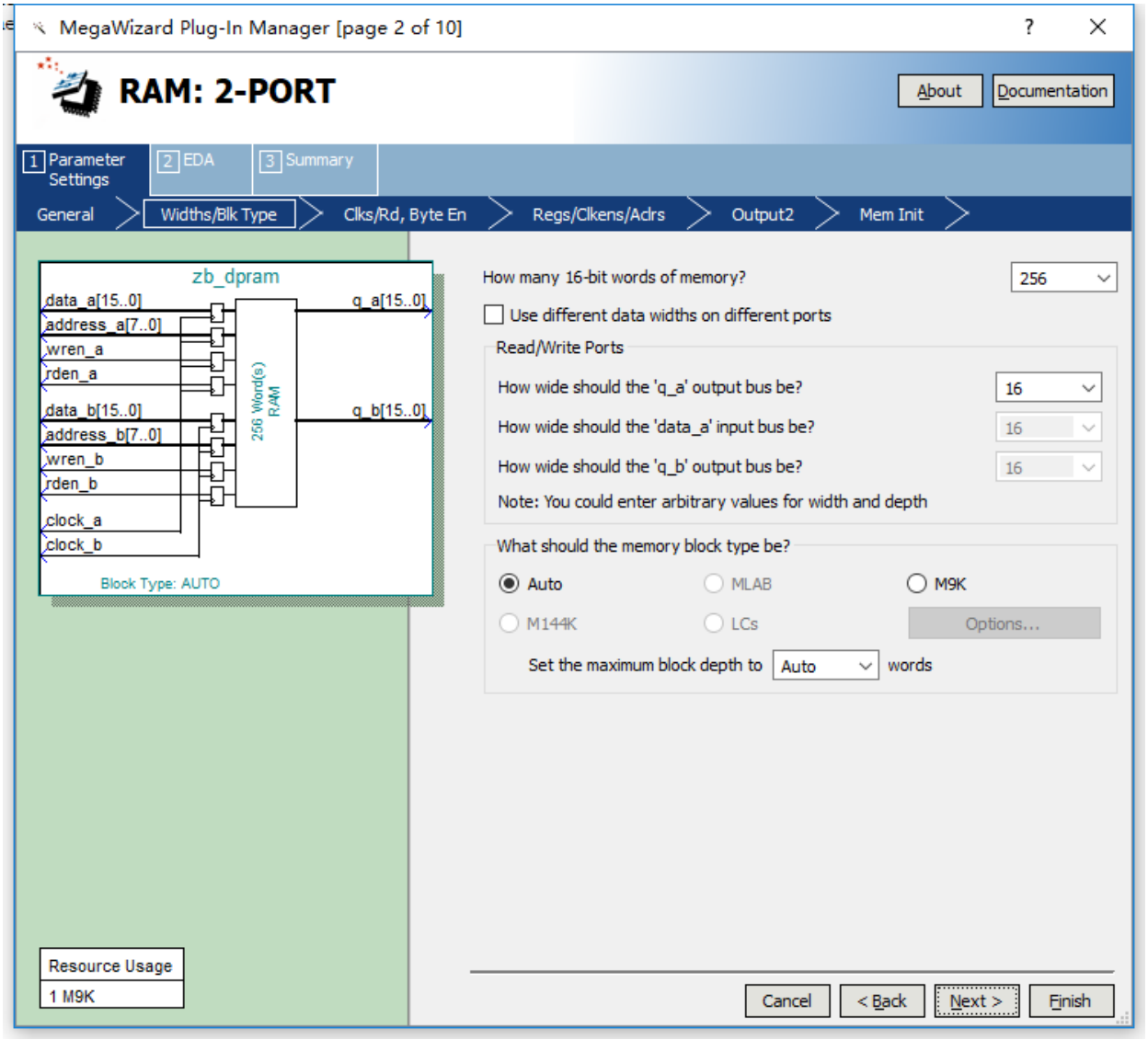
```
        //测试数据读出
        for(i=0;i<=15;i=i+1)
        begin
                rdaddress = i;//写入读地址
                #`clk_period;
        end
        #(`clk_period*20);
        $stop;
end

endmodule
```

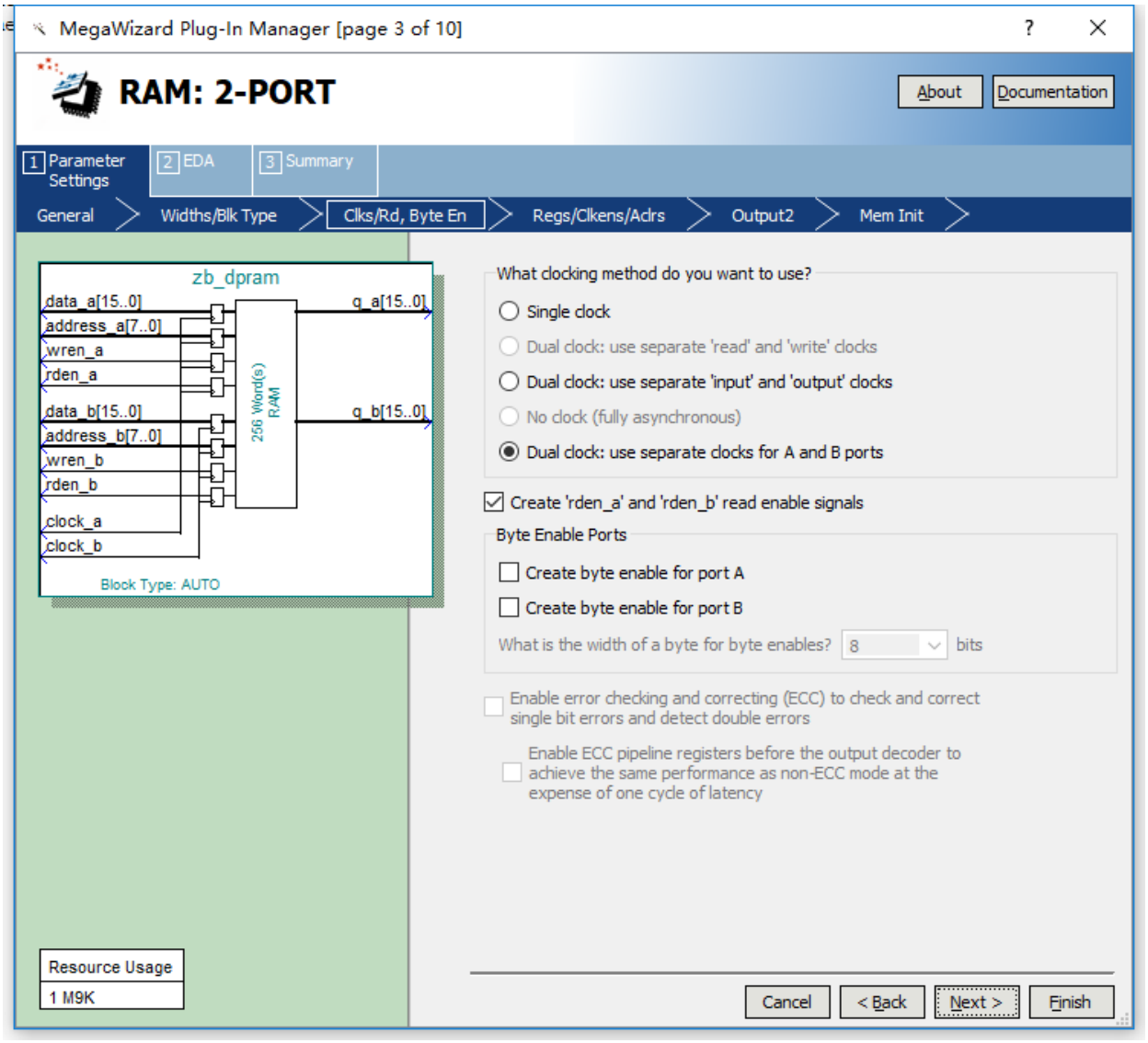
测试结果：

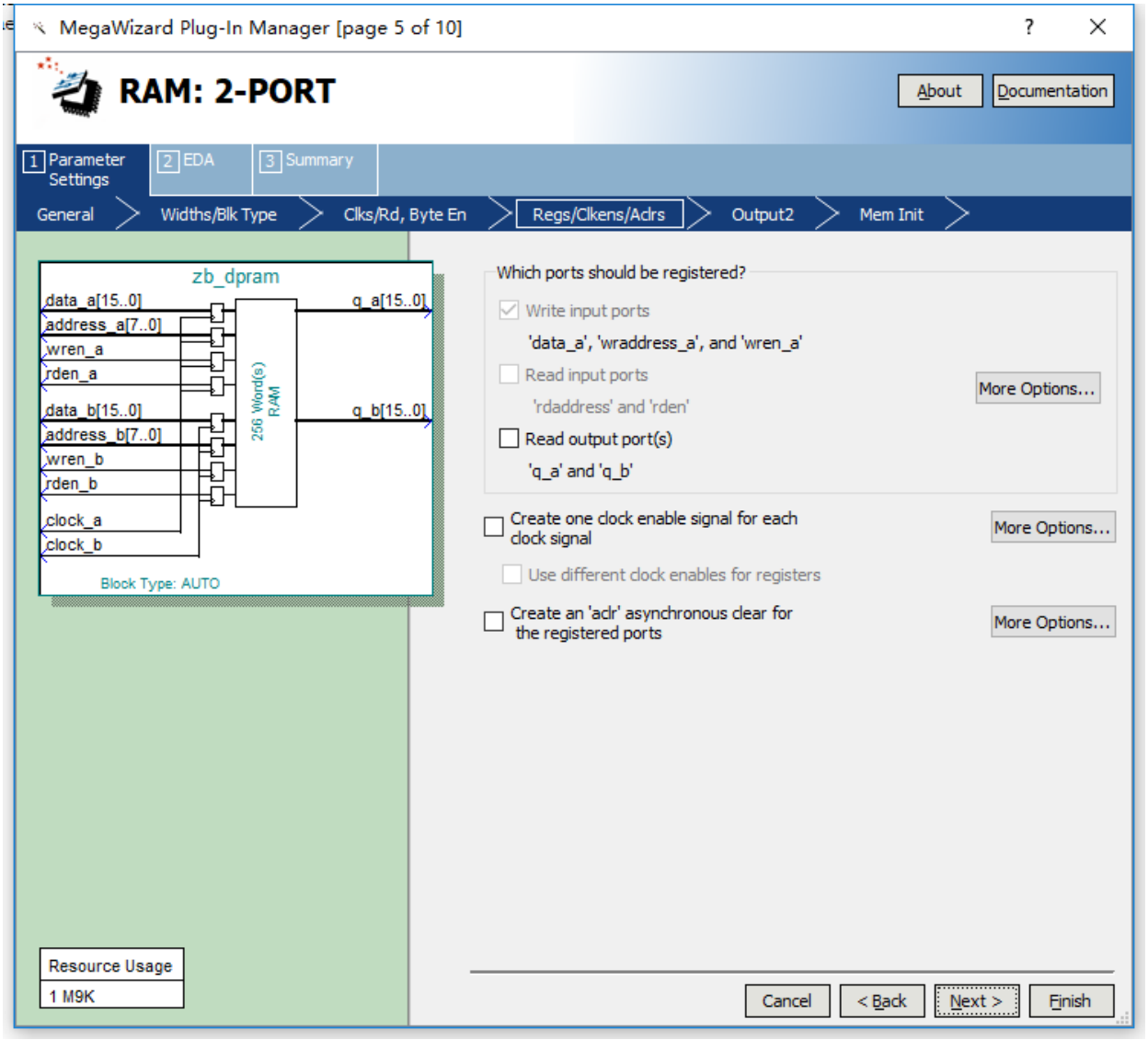


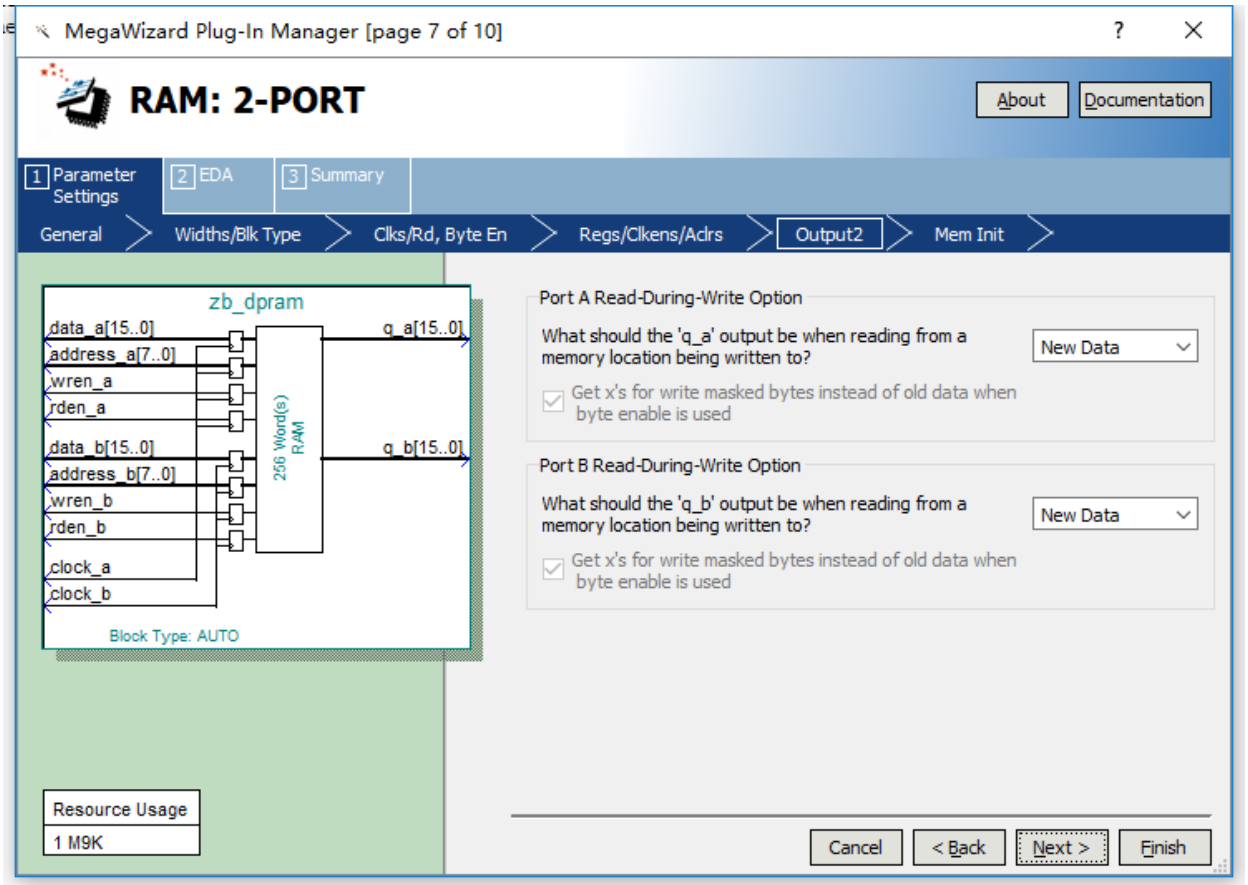
真双口RAM

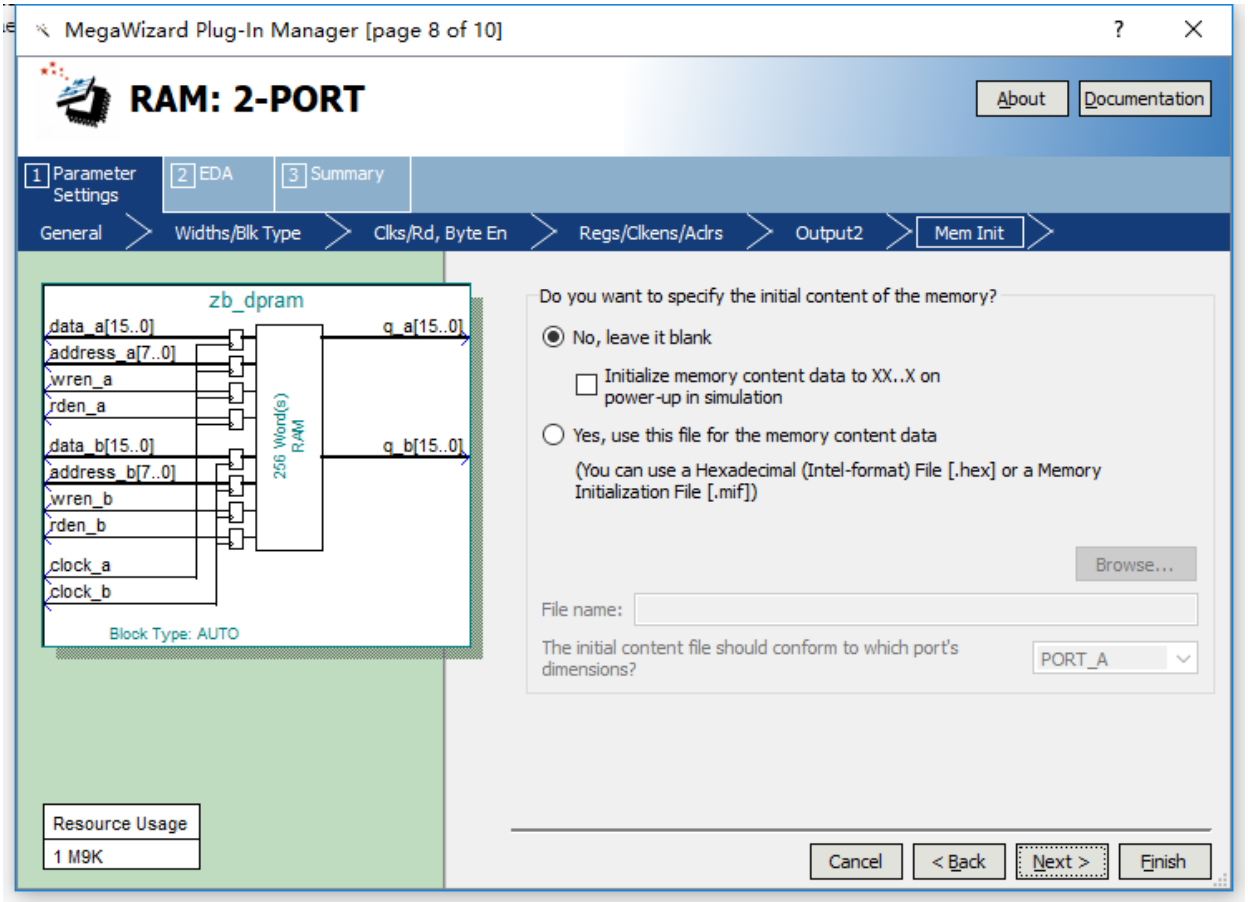

MegaWizard Plug-In Manager [page 1 of 10]
RAM: 2-PORT
About
Documentation
1 Parameter Settings
2 EDA
3 Summary
General
Widths/Blk Type
Clks/Rd, Byte En
Regs/Clkens/Adrs
Output2
Mem Init
zb_dpram
data_a[15..0]
address_a[7..0]
wren_a
rden_a
data_b[15..0]
address_b[7..0]
wren_b
rden_b
clock_a
clock_b
q_a[15..0]
q_b[15..0]
256 Word(s) RAM
Block Type: AUTO
Currently selected device family:
Cyclone IV E
Match project/default
How will you be using the dual port RAM?
With one read port and one write port
With two read/write ports
How do you want to specify the memory size?
As a number of words
As a number of bits
Resource Usage
1 M9K
Cancel
< Back
Next >
Finish

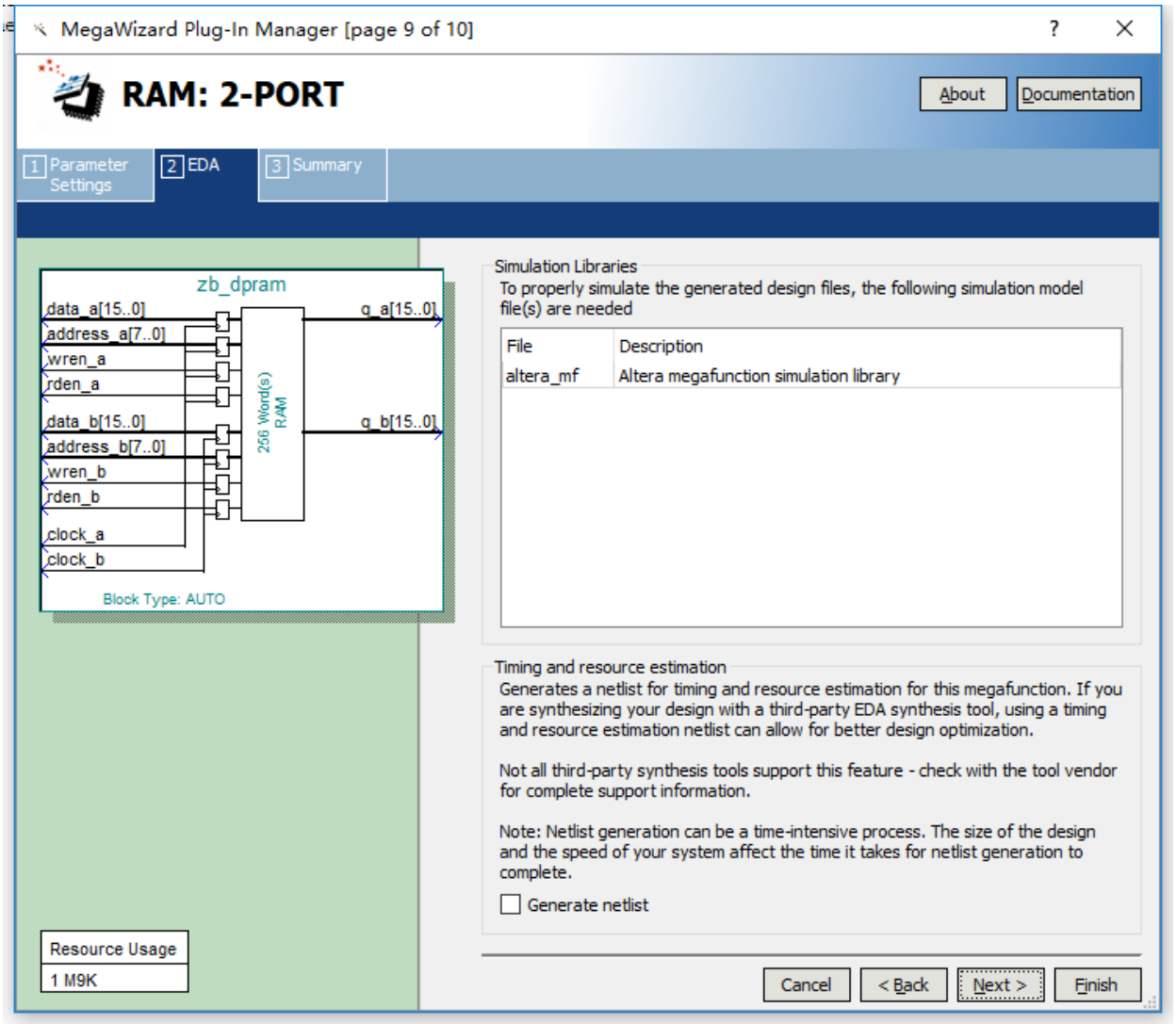

MegaWizard Plug-In Manager [page 2 of 10]
RAM: 2-PORT
About
Documentation
1 Parameter Settings
2 EDA
3 Summary
General
Widths/Blk Type
Clks/Rd, Byte En
Regs/Clkens/Adrs
Output2
Mem Init
zb_dpram
data_a[15..0]
address_a[7..0]
wren_a
rden_a
data_b[15..0]
address_b[7..0]
wren_b
rden_b
clock_a
clock_b
q_a[15..0]
q_b[15..0]
256 Word(s) RAM
Block Type: AUTO
How many 16-bit words of memory?
256
Use different data widths on different ports
Read/Write Ports
How wide should the 'q_a' output bus be?
16
How wide should the 'data_a' input bus be?
16
How wide should the 'q_b' output bus be?
16
Note: You could enter arbitrary values for width and depth
What should the memory block type be?
Auto
MLAB
M9K
M144K
LCs
Options...
Set the maximum block depth to
Auto
words
Resource Usage
1 M9K
Cancel
< Back
Next >
Finish

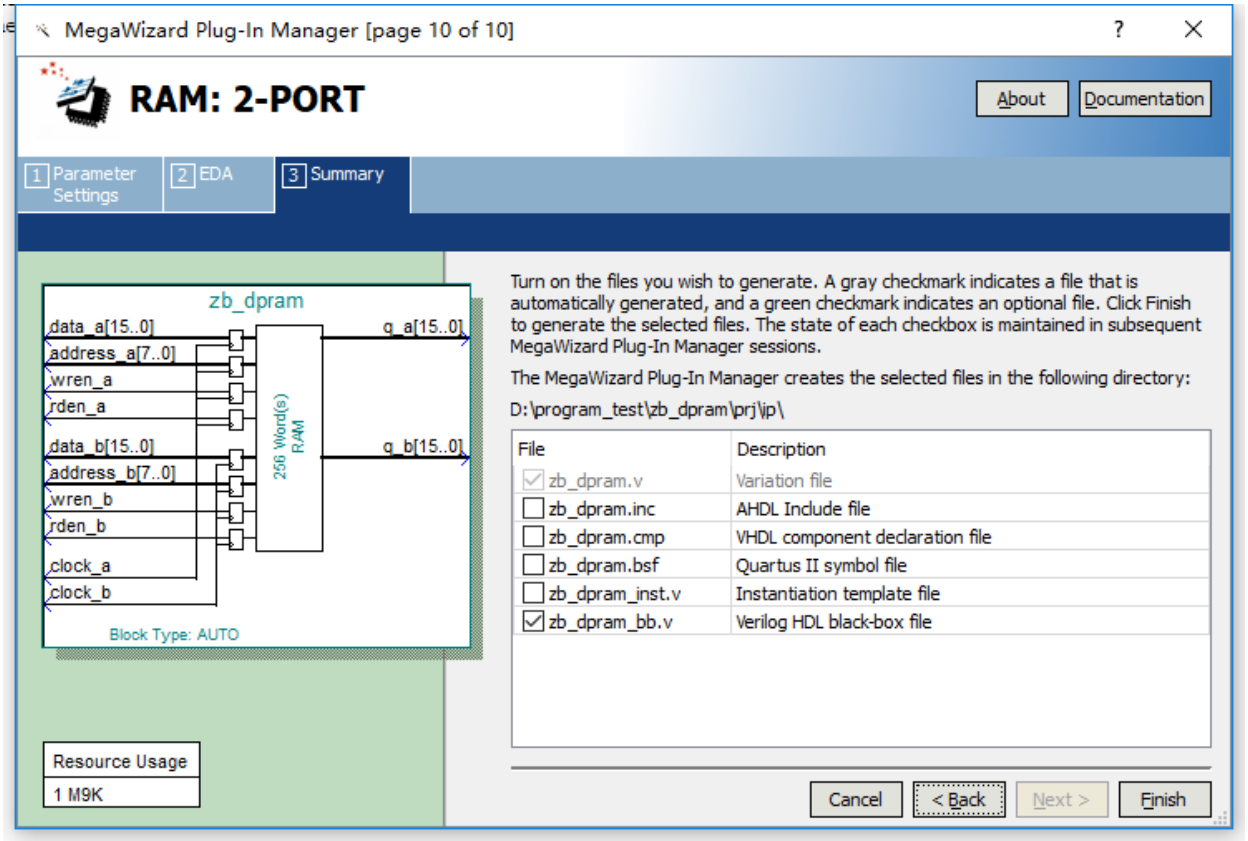

MegaWizard Plug-In Manager [page 3 of 10]
RAM: 2-PORT
About
Documentation
1 Parameter Settings
2 EDA
3 Summary
General
Widths/Blk Type
Clks/Rd, Byte En
Regs/Clkens/Adrs
Output2
Mem Init
zb_dpram
data_a[15..0]
address_a[7..0]
wren_a
rden_a
data_b[15..0]
address_b[7..0]
wren_b
rden_b
clock_a
clock_b
q_a[15..0]
q_b[15..0]
256 Word(s) RAM
Block Type: AUTO
What clocking method do you want to use?
Single clock
Dual clock: use separate 'read' and 'write' clocks
Dual clock: use separate 'input' and 'output' clocks
No clock (fully asynchronous)
Dual clock: use separate clocks for A and B ports
Create 'rden_a' and 'rden_b' read enable signals
Byte Enable Ports
Create byte enable for port A
Create byte enable for port B
What is the width of a byte for byte enables? 8 bits
Enable error checking and correcting (ECC) to check and correct single bit errors and detect double errors
Enable ECC pipeline registers before the output decoder to achieve the same performance as non-ECC mode at the expense of one cycle of latency
Resource Usage
1 M9K
Cancel
< Back
Next >
Finish


MegaWizard Plug-In Manager [page 5 of 10]
RAM: 2-PORT
About
Documentation
1 Parameter Settings
2 EDA
3 Summary
General
Widths/Blk Type
Clks/Rd, Byte En
Regs/Clkens/Aclrs
Output2
Mem Init
zb_dpram
data_a[15..0]
address_a[7..0]
wren_a
rden_a
data_b[15..0]
address_b[7..0]
wren_b
rden_b
clock_a
clock_b
q_a[15..0]
q_b[15..0]
256 Word(s) RAM
Block Type: AUTO
Which ports should be registered?
Write input ports
'data_a', 'wraddress_a', and 'wren_a'
Read input ports
'rdaddress' and 'rden'
More Options...
Read output port(s)
'q_a' and 'q_b'
Create one clock enable signal for each clock signal
More Options...
Use different clock enables for registers
Create an 'aclr' asynchronous clear for the registered ports
More Options...
Resource Usage
1 M9K
Cancel
< Back
Next >
Finish


MegaWizard Plug-In Manager [page 7 of 10]
RAM: 2-PORT
About
Documentation
1 Parameter Settings
2 EDA
3 Summary
General
Widths/Blk Type
Clks/Rd, Byte En
Regs/Clkens/Adrs
Output2
Mem Init
zb_dpram
data_a[15..0]
address_a[7..0]
wren_a
rden_a
data_b[15..0]
address_b[7..0]
wren_b
rden_b
clock_a
clock_b
q_a[15..0]
q_b[15..0]
256 Word(s) RAM
Block Type: AUTO
Port A Read-During-Write Option
What should the 'q_a' output be when reading from a memory location being written to?
New Data
Get x's for write masked bytes instead of old data when byte enable is used
Port B Read-During-Write Option
What should the 'q_b' output be when reading from a memory location being written to?
New Data
Get x's for write masked bytes instead of old data when byte enable is used
Resource Usage
1 M9K
Cancel
< Back
Next >
Finish


MegaWizard Plug-In Manager [page 8 of 10]
RAM: 2-PORT
About
Documentation
1 Parameter Settings
2 EDA
3 Summary
General
Widths/Blk Type
Clks/Rd, Byte En
Regs/Clkens/Adrs
Output2
Mem Init
zb_dpram
data_a[15..0]
address_a[7..0]
wren_a
rden_a
data_b[15..0]
address_b[7..0]
wren_b
rden_b
clock_a
clock_b
q_a[15..0]
q_b[15..0]
256 Word(s) RAM
Block Type: AUTO
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])
Browse...
File name:
The initial content file should conform to which port's dimensions?
PORT_A
Resource Usage
1 M9K
Cancel
< Back
Next >
Finish


MegaWizard Plug-In Manager [page 9 of 10]
RAM: 2-PORT
About
Documentation
1 Parameter Settings
2 EDA
3 Summary
zb_dpram
data_a[15..0]
address_a[7..0]
wren_a
rden_a
data_b[15..0]
address_b[7..0]
wren_b
rden_b
clock_a
clock_b
q_a[15..0]
q_b[15..0]
256 Word(s) RAM
Block Type: AUTO
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
Timing and resource estimation
Generates a netlist for timing and resource estimation for this megafunction. If you are synthesizing your design with a third-party EDA synthesis tool, using a timing and resource estimation netlist can allow for better design optimization.
Not all third-party synthesis tools support this feature - check with the tool vendor for complete support information.
Note: Netlist generation can be a time-intensive process. The size of the design and the speed of your system affect the time it takes for netlist generation to complete.
Generate netlist
Resource Usage
1 M9K
Cancel
< Back
Next >
Finish


MegaWizard Plug-In Manager [page 10 of 10]
RAM: 2-PORT
About
Documentation
1 Parameter Settings
2 EDA
3 Summary
zb_dpram
data_a[15..0]
address_a[7..0]
wren_a
rden_a
data_b[15..0]
address_b[7..0]
wren_b
rden_b
clock_a
clock_b
q_a[15..0]
q_b[15..0]
256 Word(s) RAM
Block Type: AUTO
Turn on the files you wish to generate. A gray checkmark indicates a file that is automatically generated, and a green 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:
D:\program_test\zb_dpram\prj\ip\
File
Description
zb_dpram.v
Variation file
zb_dpram.inc
AHDL Include file
zb_dpram.cmp
VHDL component declaration file
zb_dpram.bsf
Quartus II symbol file
zb_dpram_inst.v
Instantiation template file
zb_dpram_bb.v
Verilog HDL black-box file
Resource Usage
1 M9K
Cancel
< Back
Next >
Finish

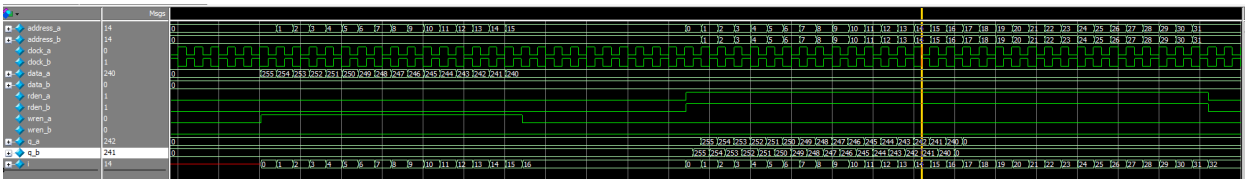
测试脚本：

设计思路：1.先用a端口向RAM中写入数据，数据为：255~240，地址为：0~15；2.两个端口同时读出地址为0~31中数据；3.用b端口向RAM写入数据，数据为：84~69，地址为16~31；4.两个端口同时读出地址为0~31中数据；

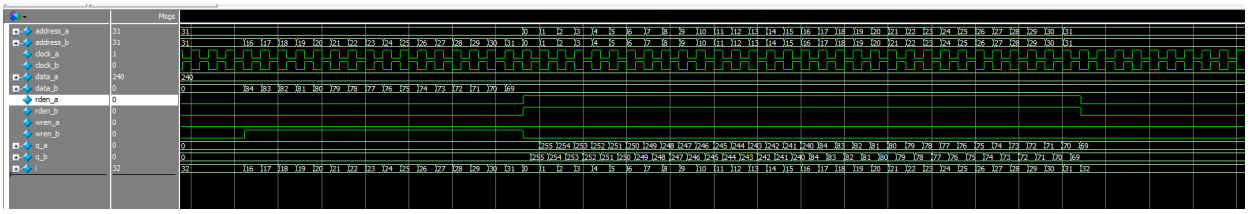
第1—2步现象；

测试结果说明：

a端口和b端口同时操作的是一段地址



第3—4步现象



```
`timescale 1ns/1ns
`define clk_period 20

module zb_dpram_tb;

//输入信号
reg [7:0]address_a;
reg [7:0]address_b;
reg clock_a;
reg clock_b;
reg [15:0]data_a;
reg [15:0]data_b;
reg rden_a;
reg rden_b;
reg wren_a;
reg wren_b;
```

```
//输出信号
wire [15:0]q_a;
wire [15:0]q_b;

integer i;//用于for循环

zb_dpram zb_dpram0(
        .address_a(address_a),
        .address_b(address_b),
        .clock_a(clock_a),
        .clock_b(clock_b),
        .data_a(data_a),
        .data_b(data_b),
        .rden_a(rden_a),
        .rden_b(rden_b),
        .wren_a(wren_a),
        .wren_b(wren_b),
        .q_a(q_a),
        .q_b(q_b)
);

//产生时钟信号
initial
begin
        clock_a = 1;
        clock_b = 0;
end
always#(`clk_period/2)clock_a = ~clock_a;
always#(`clk_period/2)clock_b = ~clock_b;

initial begin
        data_a = 0;
        data_b = 0;
        address_a = 0;
```

```
        address_b = 0;
        wren_a = 0;
        wren_b = 0;
        rden_a = 0;
        rden_b = 0;
        #(`clk_period*10 + 1);//初始化完成后延时一段时间

/*-------------------------------------*/
        /*进行数据写入仿真——向a端口中写入数据*/
        for(i=0;i<=15;i=i+1)
        begin
                wren_a = 1;//数据开始写入
                data_a = 255 - i;//数据为255-i
                address_a = i;//写入的地址为i
                #`clk_period;
        end
        wren_a = 0;//数据写入完成
        #(`clk_period*10);

/*-------------------------------------*/
        /*测试数据读出——两个端口同时读出数据*/
        for(i=0;i<=31;i=i+1)
        begin
                rden_a = 1;//数据开始写入
                rden_b = 1;//数据开始写入
                address_a = i;//写入地址
                address_b = i;//写入地址
                #`clk_period;
        end
        rden_a = 0;
        rden_b = 0;
        #(`clk_period*10);

/*-------------------------------------*/
        /*进行数据写入仿真——向b端口中写入数据*/
```

```
        for(i=16;i<=31;i=i+1)
        begin
                wren_b = 1;//数据开始写入
                data_b = 100 - i;//数据为84~69
                address_b = i;//写入的地址为i
                #`clk_period;
        end
        wren_b = 0;//数据写入完成

/*--------------------------------------*/
        /*测试数据读出——两个端口同时读出数据*/
        for(i=0;i<=31;i=i+1)
        begin
                rden_a = 1;//数据开始写入
                rden_b = 1;//数据开始写入
                address_a = i;//写入地址
                address_b = i;//写入地址
                #`clk_period;
        end
        rden_a = 0;
        rden_b = 0;
        #(`clk_period*10);

        $stop;

end

endmodule
```