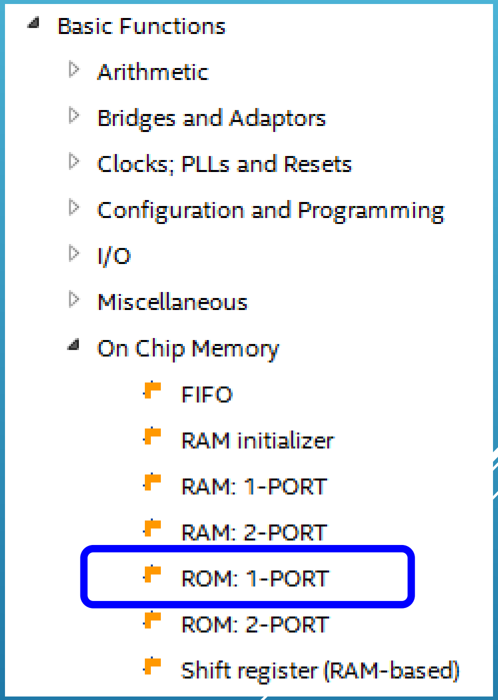
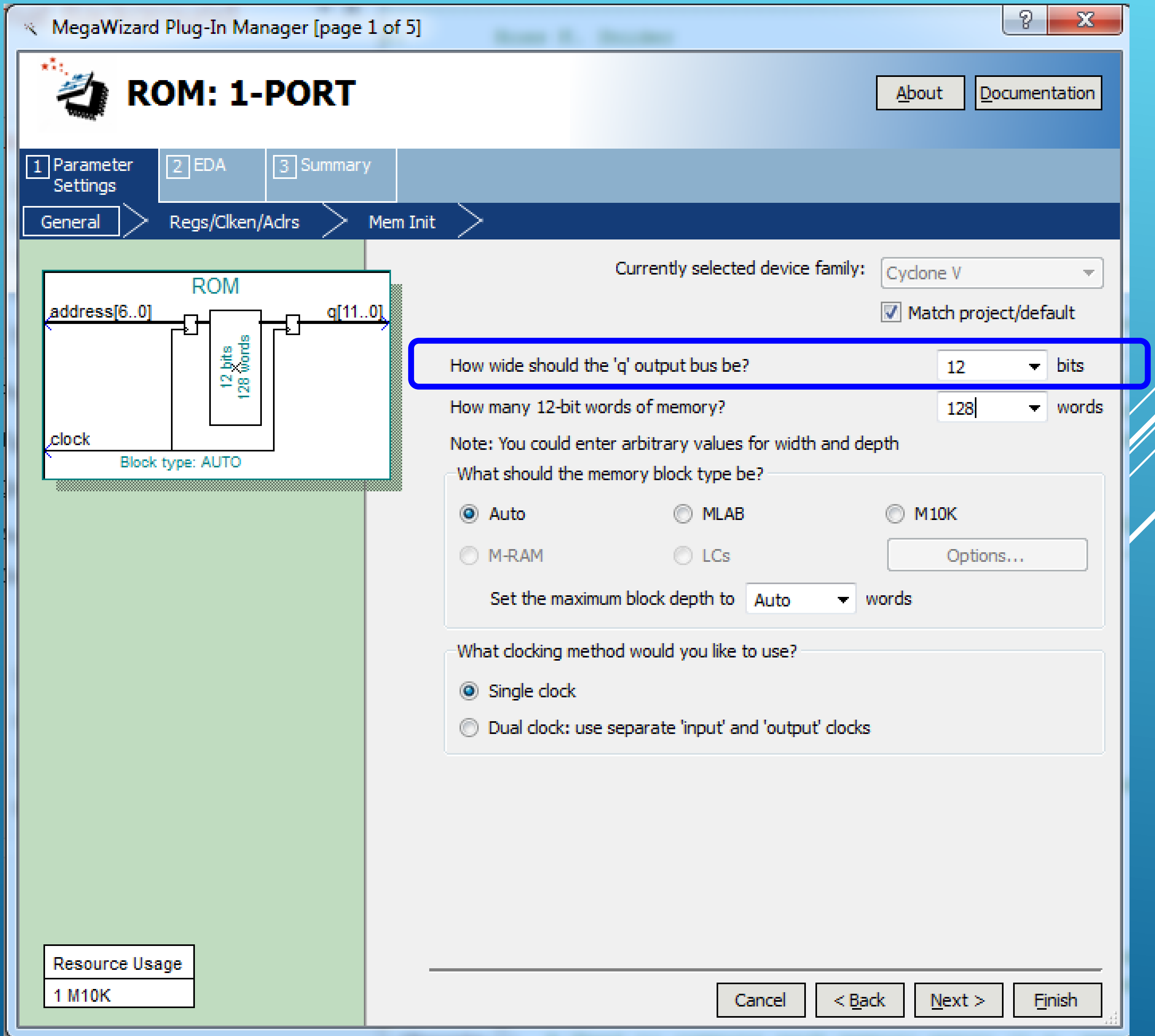
Creating an FPGA ROM

In Quartus, right panel: under Installed IP
Select **ROM: 1-PORT**



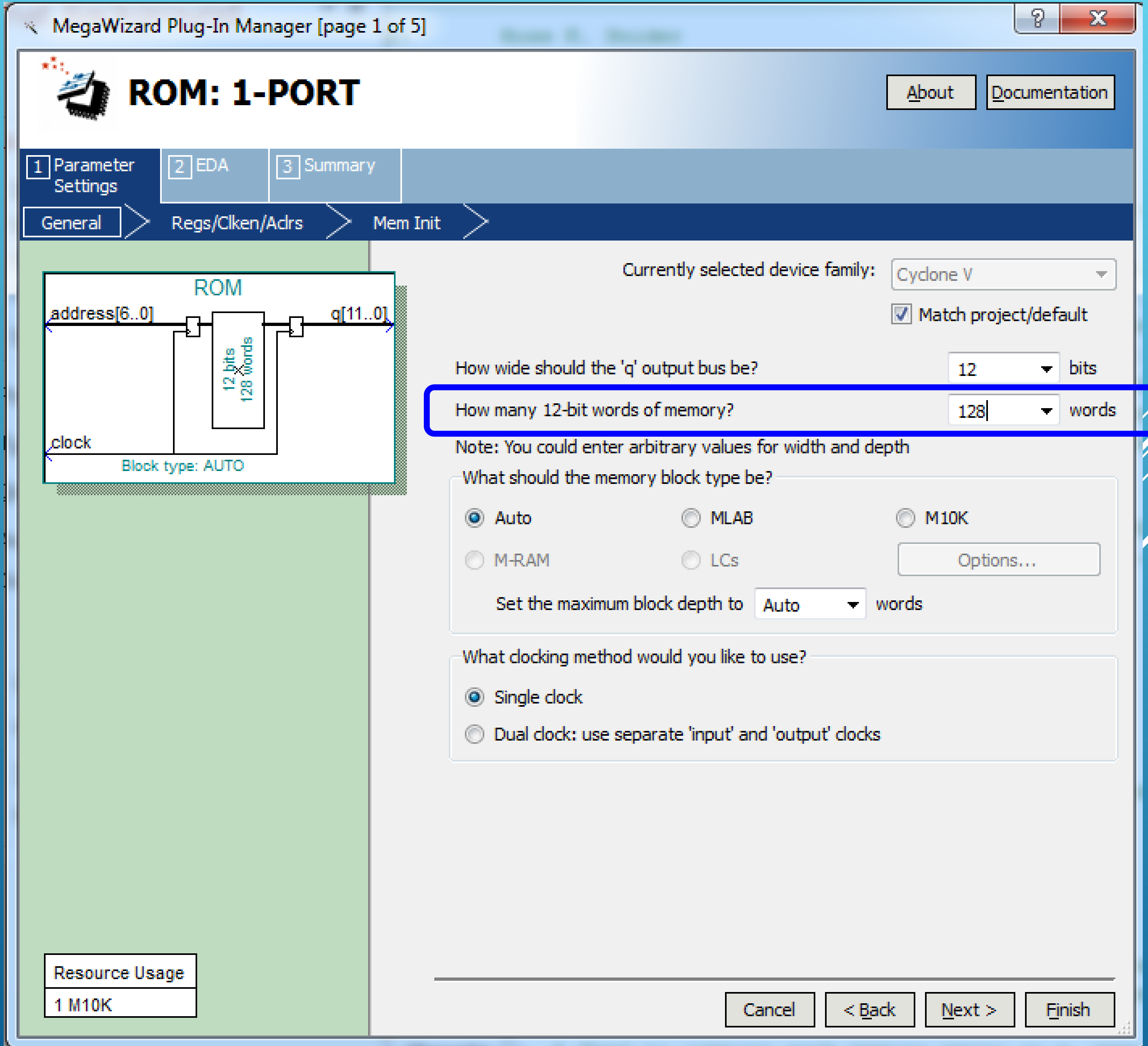
Creating an FPGA ROM



Select how large in bits the words in memory should be.

Here we have selected the word size to be 12 bits.

Creating an FPGA ROM



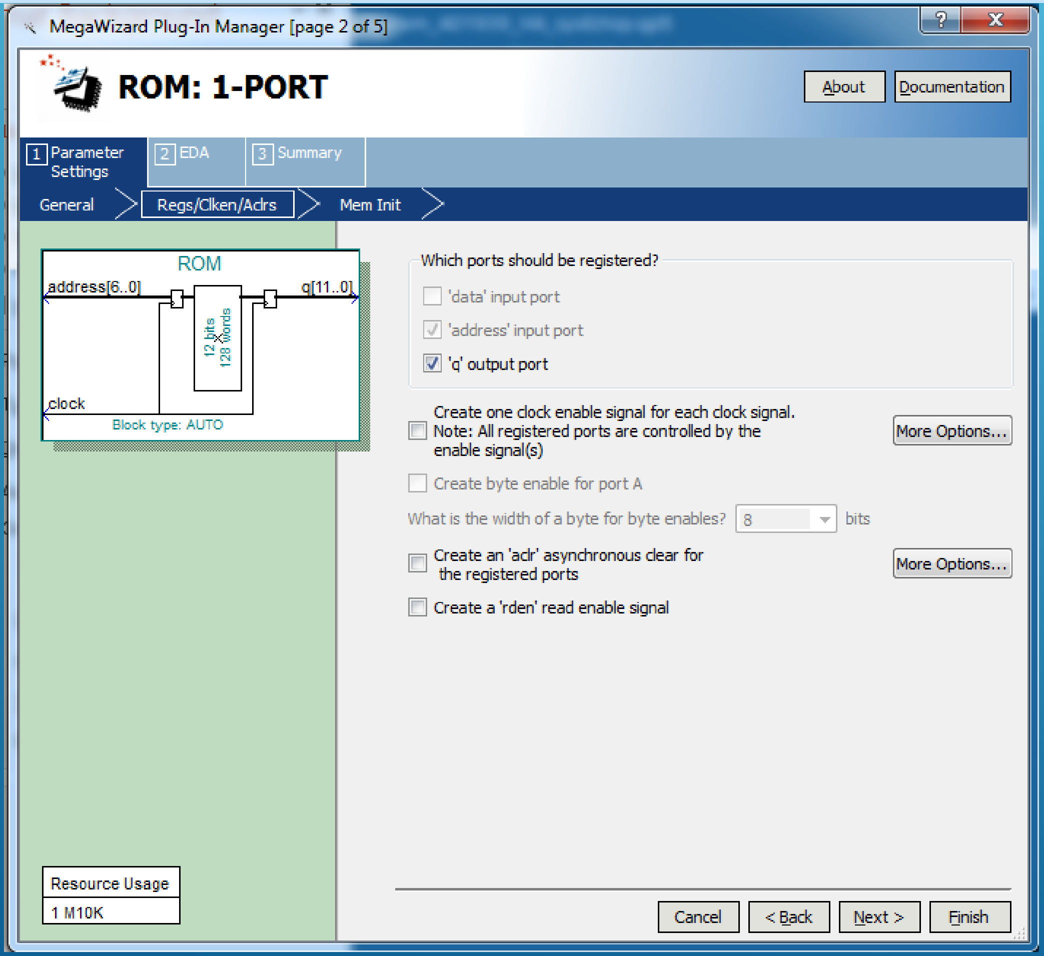
Select how many words in memory.

Here we select 128 12-bit words, which gives us a 7-bit address space.

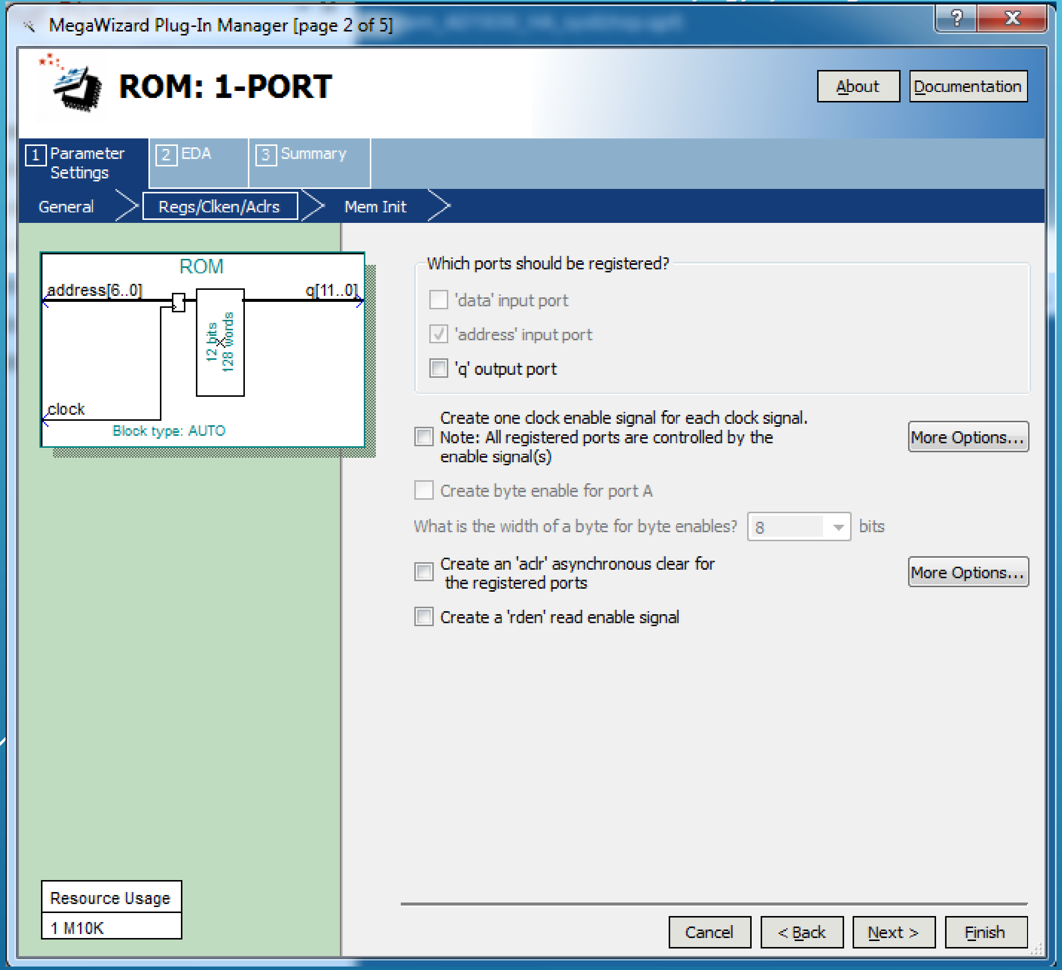
Creating an FPGA ROM

Determine if the output q should be registered.

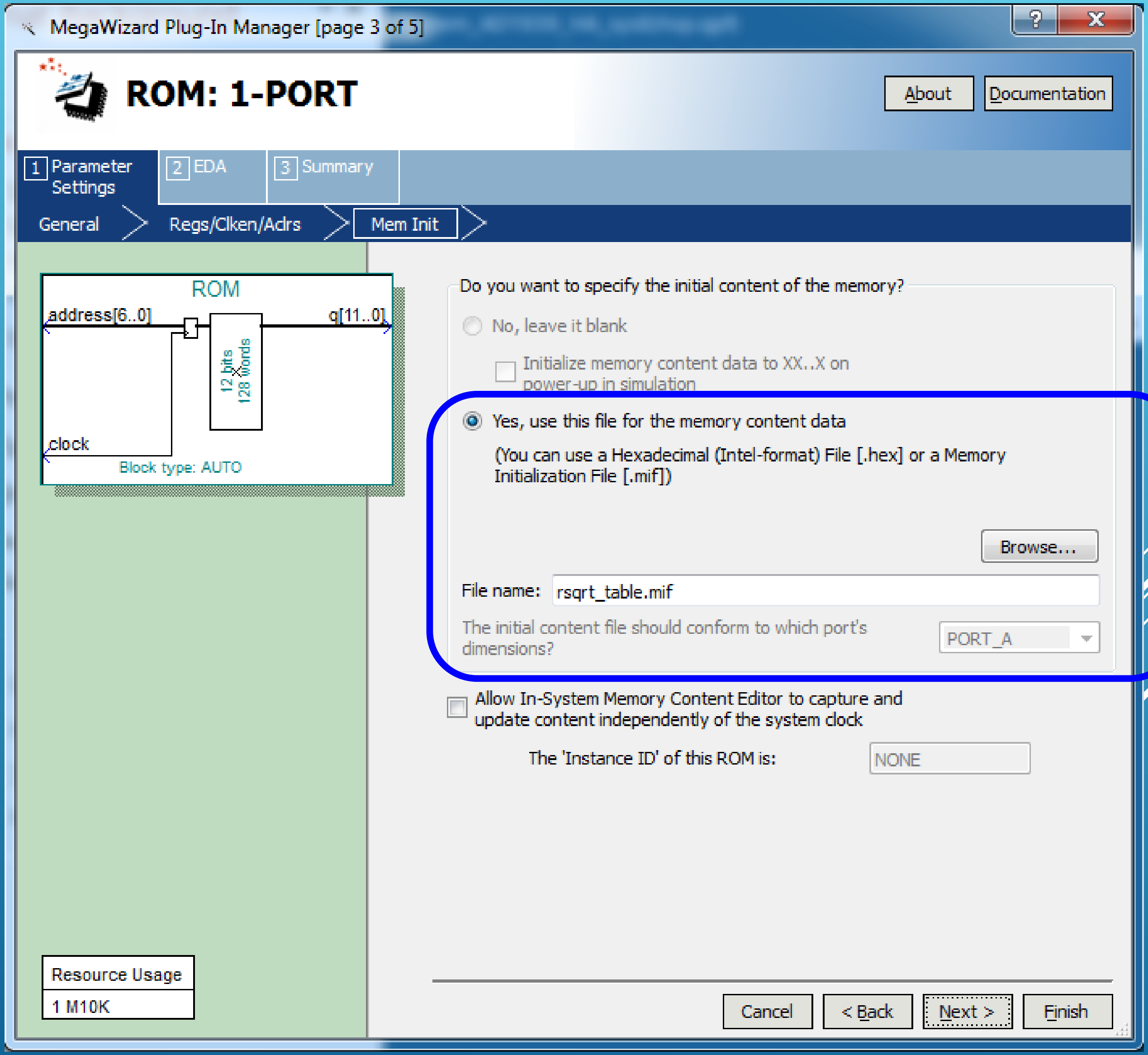
The output is registered.
The design will be able to run with a faster clock.
The result will take an extra clock cycle to show up



The output is **NOT** registered.
Easier to design into system



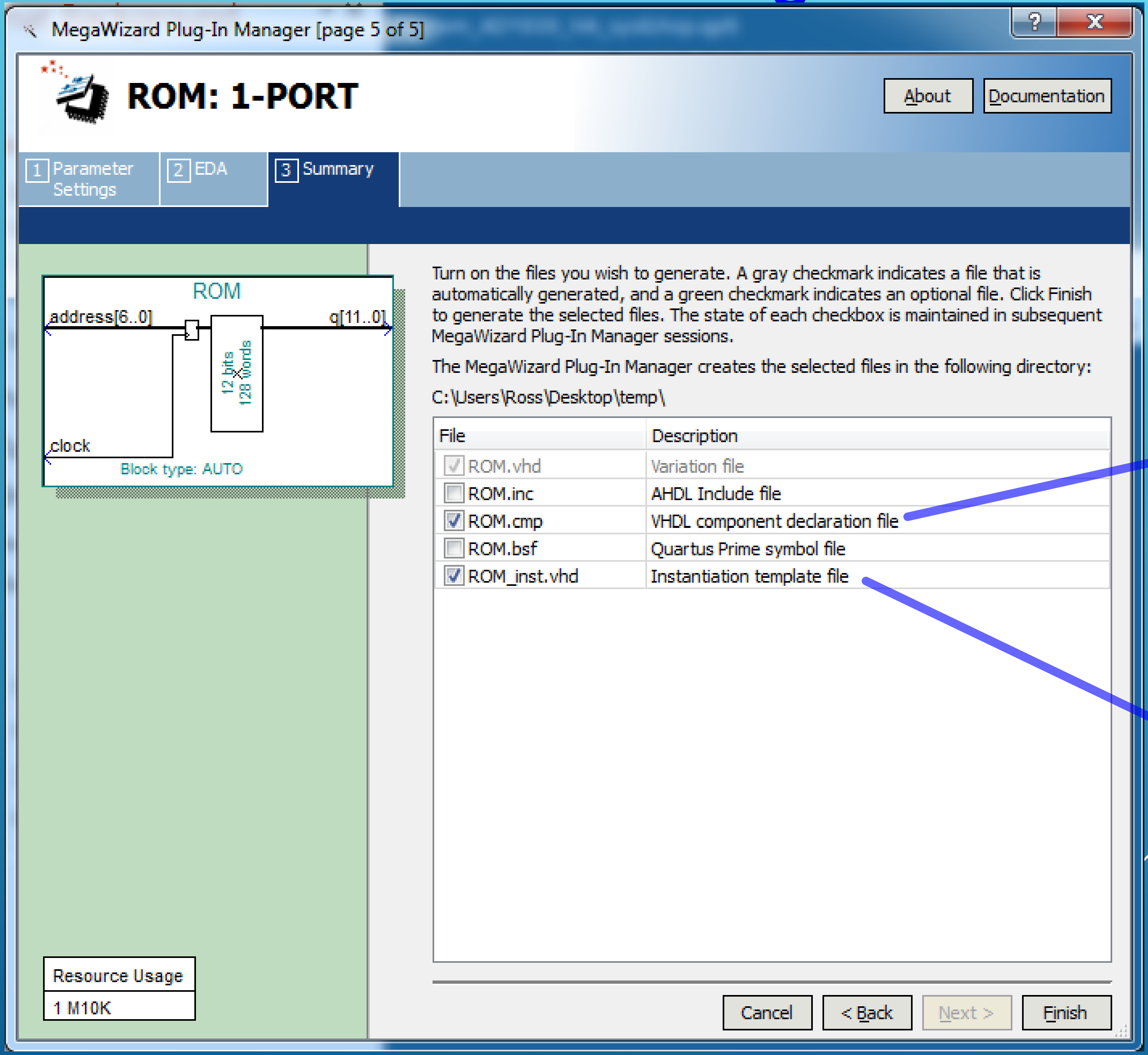
Creating an FPGA ROM



Specify .mif file that initializes memory with custom values.

Note: The .mif file must match the memory size in terms of word size and number of memory words.

Creating an FPGA ROM



Select the component declaration file and the Instantiation template file.

```
component ROM
    PORT
    (
            address     : IN STD_LOGIC_VECTOR (6 DOWNTO 0);
            clock       : IN STD_LOGIC   := '1';
            q           : OUT STD_LOGIC_VECTOR (11 DOWNTO 0)
    );
end component;
```

```
ROM_inst : ROM PORT MAP (
    address  => address_sig,
    clock    => clock_sig,
    q        => q_sig
);
```

Creating the Memory Initialization File

Open Files

rsqrt_table.mif ×

```
DEPTH = 256;
WIDTH = 16;
ADDRESS_RADIX = BIN;
DATA_RADIX = BIN;
CONTENT
BEGIN
00000000 : 1000000000000000;   -- 1 : (1)^(-3/2)=1
00000001 : 0111111101000001;   -- 2 : (1.0039)^(-3/2)=0.99417
00000010 : 0111111010000100;   -- 3 : (1.0078)^(-3/2)=0.9884
00000011 : 0111110111001000;   -- 4 : (1.0117)^(-3/2)=0.98267
00000100 : 0111110100001111;   -- 5 : (1.0156)^(-3/2)=0.97702
00000101 : 0111110001010111;   -- 6 : (1.0195)^(-3/2)=0.97141
00000110 : 0111101110100001;   -- 7 : (1.0234)^(-3/2)=0.96585
00000111 : 0111101011101101;   -- 8 : (1.0273)^(-3/2)=0.96036
00001000 : 0111101000111010;   -- 9 : (1.0313)^(-3/2)=0.9549
00001001 : 0111100110001001;   -- 10 : (1.0352)^(-3/2)=0.94949
00001010 : 0111100011011010;   -- 11 : (1.0391)^(-3/2)=0.94415
00001011 : 0111100000101100;   -- 12 : (1.043)^(-3/2)=0.93884
00001100 : 0111011110000000;   -- 13 : (1.0469)^(-3/2)=0.93359
00001101 : 0111011011010110;   -- 14 : (1.0508)^(-3/2)=0.92841
00001110 : 0111011000101101;   -- 15 : (1.0547)^(-3/2)=0.92325
00001111 : 0111010110000101;   -- 16 : (1.0586)^(-3/2)=0.91812
00010000 : 0111010011100000;   -- 17 : (1.0625)^(-3/2)=0.91309
00010001 : 0111010000111011;   -- 18 : (1.0664)^(-3/2)=0.90805
00010010 : 0111001110011001;   -- 19 : (1.0703)^(-3/2)=0.90311
00010011 : 0111001011110111;   -- 20 : (1.0742)^(-3/2)=0.89816
00010100 : 0111001001011000;   -- 21 : (1.0781)^(-3/2)=0.89331
00010101 : 0111000110111001;   -- 22 : (1.082)^(-3/2)=0.88846
00010110 : 0111000100011100;   -- 23 : (1.0859)^(-3/2)=0.88367
00010111 : 0111000010000001;   -- 24 : (1.0898)^(-3/2)=0.87894
00011000 : 0110111111100111;   -- 25 : (1.0938)^(-3/2)=0.87424
00011001 : 0110111101001110;   -- 26 : (1.0977)^(-3/2)=0.86957
00011010 : 0110111010110110;   -- 27 : (1.1016)^(-3/2)=0.86493
00011011 : 0110111000100000;   -- 28 : (1.1055)^(-3/2)=0.86035
00011100 : 0110110110001100;   -- 29 : (1.1094)^(-3/2)=0.85583
00011101 : 0110110011111000;   -- 30 : (1.1133)^(-3/2)=0.85132
00011110 : 0110110001100110;   -- 31 : (1.1172)^(-3/2)=0.84686
00011111 : 0110101111010101;   -- 32 : (1.1211)^(-3/2)=0.84244
00100000 : 0110101101000101;   -- 33 : (1.125)^(-3/2)=0.83804
00100001 : 0110101010110111;   -- 34 : (1.1289)^(-3/2)=0.83371
00100010 : 0110101000101010;   -- 35 : (1.1328)^(-3/2)=0.82941
00100011 : 0110100110011110;   -- 36 : (1.1367)^(-3/2)=0.82513
00100100 : 0110100100010011;   -- 37 : (1.1406)^(-3/2)=0.82089
00100101 : 0110100010001001;   -- 38 : (1.1445)^(-3/2)=0.81668
00100110 : 0110100000000001;   -- 39 : (1.1484)^(-3/2)=0.81253
00100111 : 0110011101111010;   -- 40 : (1.1523)^(-3/2)=0.80841
```

Creating the Memory Initialization File

rsqrt_lookup_table_mif_gen.m

```
%-----------------------------------------
% Create lookup table for (x_beta)^(-3/2)
%-----------------------------------------
Nbits_address = 8;  % How many fraction bits will be used as the address?
Nbits_output_fraction  = 7;  % The number of fractional bits in result.  The output word size will be Nbits_output + 1, since we need a bit
Nwords = 2^Nbits_address
for i=0:(Nwords-1)  % Need to compute each memory entry (i.e. memory size)
    x_beta_table{i+1}.address = i;  % Memory Address
    fa = fi(i,0,Nbits_address,0);
    fa_bits = fa.bin;               % Memory Address in binary
    fb = fi(0, 0, Nbits_address+1, Nbits_address);  % Set number of bits for result
    fb.bin = ['1' fa_bits]; % set the value using the binary representation. The address is our input value 1 <= x_beta < 2  where the leadi
    x_beta_table{i+1}.input_value = double(fb);  % convert this binary input to it's double representation
    x_beta_table{i+1}.input_bits = fa.bin;  % keep track of the input bits
    fy = fi(double(fb)^(-3/2),0,Nbits_output_fraction+1,Nbits_output_fraction);  % compute (x_beta)^(-3/2) and convert to fixed-point
    x_beta_table{i+1}.output_value = double(fy); % store the result as a double
    x_beta_table{i+1}.output_bits = fy.bin;   % store the resulting binary representation
end
```

Creating the Memory Initialization File

rsqrt_lookup_table_mif_gen.m

```
%-----------------------------------------------------
% Create the Altera .mif file for the lookup table
%-----------------------------------------------------
fid = fopen('rsqrt_table.mif','w');
%-------------------------------------
% Write File Header
%-------------------------------------
line = ['DEPTH = ' num2str(2^Nbits_address) ';'];  % The size of memory in words
fprintf(fid,'%s\n',line);
line = ['WIDTH = ' num2str(Nbits_output_fraction+1) ';']; % The size of data in bits
fprintf(fid,'%s\n',line);
line = ['ADDRESS_RADIX = BIN;'];  % The radix for address values
fprintf(fid,'%s\n',line);
line = ['DATA_RADIX = BIN;'];  % The radix for data values
fprintf(fid,'%s\n',line);
line = ['CONTENT'];
fprintf(fid,'%s\n',line);
line = ['BEGIN'];
fprintf(fid,'%s\n',line);  % start of (address : data pairs)
```

Creating the
Memory Initialization File
rsqrt_lookup_table_mif_gen.m

```
%-------------------------------------
% Write Memory Data
%-------------------------------------
for data_index = 1:Nwords
    line = [x_beta_table{data_index}.input_bits ' : ' x_beta_table{data_index}.output_bits ';    -- '
          num2str(data_index) ' : (' num2str(x_beta_table{data_index}.input_value) ')^(-3/2)='
          num2str(x_beta_table{data_index}.output_value) ];
    fprintf(fid,'%s\n',line);
end
%-------------------------------------
% Write File End
%-------------------------------------
line = ['END;'];
fprintf(fid,'%s\n',line);
%-------------------------------------
% Close File
%-------------------------------------
fclose(fid);
```