Quartus ROM Creation Instructions

(in Quartus Prime Lite 18.1)

Problem: You have an ASM or CPU that you would like to control/test from a ROM (EEPROM or Flash). How can you simulate the ROM under Quartus?

Solution:
Pick a device that has memory, e.g., Cyclone V. I picked the last device in the Cyclone 5 family.

Note that if you have to add a Cyclone V to **YOUR** version of Quartus **(perhaps v18.1)**:

1. Go to https://www.intel.com/content/www/us/en/programmable/downloads/download-center.html
2. Select by Device
3. Cyclone Series
4. Cyclone V
5. Use **YOUR** version of Quartus Prime Lite Edition (perhaps v18.1)
6. Check
7. Cyclone V device support
8. In Windows, run Intel FPGA … > Device Installer (Quartus Prime …). This will open a program to allow you to install new devices.
9. Follow instructions

We will use the ROM: 1-PORT device available in the "IP Catalog" (see Figure 1), found on the right side of the Quartus screen. Under IP Catalog, select "Installed IP | Library | Basic Function | On Chip Memory."

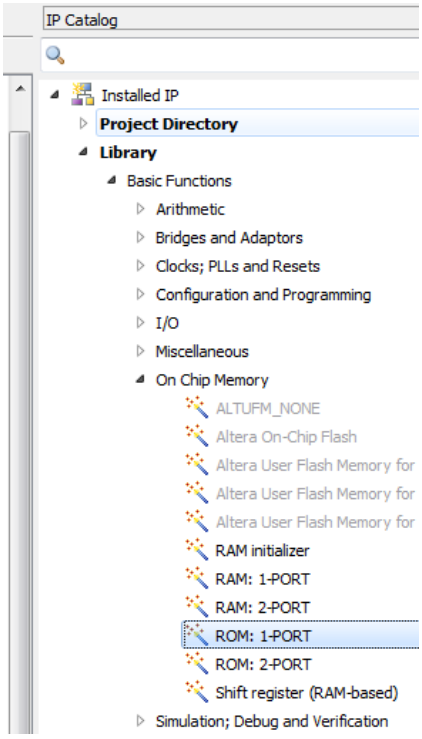

Figure 1: `ROM: 1-PORT` in `IP Catalog`.

When you select ROM: 1-PORT, Figure 2 will appear. I already added a filename (ROM_example) at the end of the path. Now select "OK."

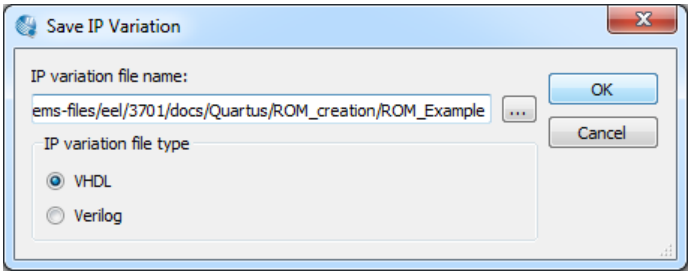

Figure 2: IP Variation filename.

Quartus ROM Creation Instructions
(in Quartus Prime Lite 18.1)

A MegaWizard will appear (as shown in Figure 3). Select the size of the output bus (width of the memory) and the number of addresses (number of words of memory). For example, a 1k x 8 ROM will have the values 8 and 1024, as shown in Figure 4. Then select "Next >."

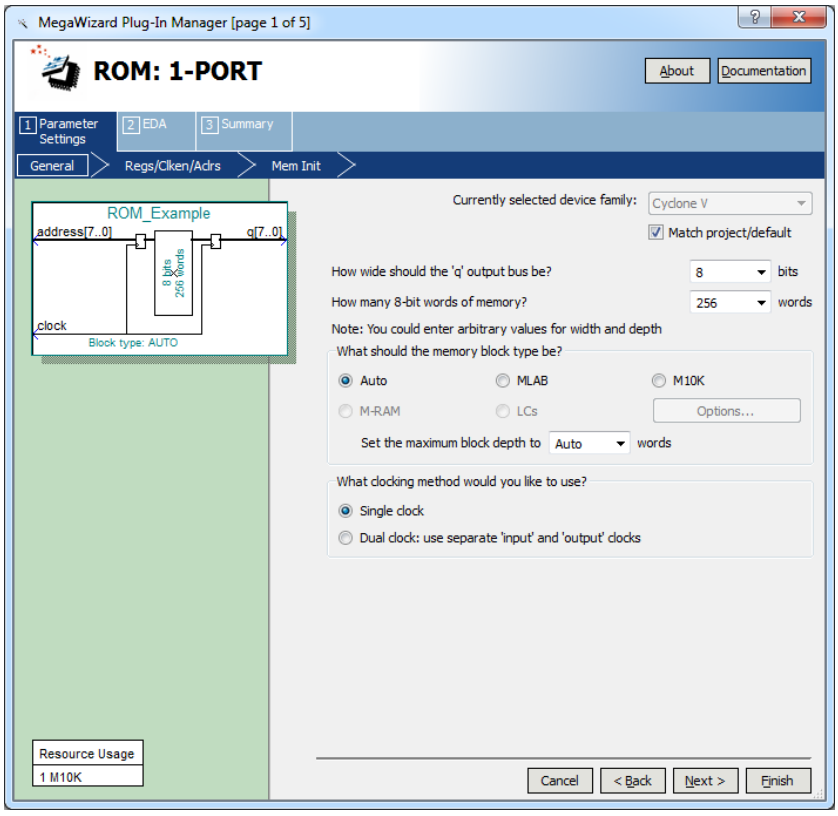

Figure 3: MegaWizard.

Un-check anything that is checked on the page 2 of 5 of the MegaWizard (see Figure 5) that asks "Which ports should be registered?" Then select "Next >."

On page 3 of 5 of the MegaWizard, select "Yes, use this file for the memory content data" should be checked. Browse to the file name of the ROM memory initialization file (mif), e.g., `contents.mif`. (An example mif file is described later in this document and shown in Figure 9.) Note that you will need to change the "Files of type:" to allow mif files to be used. Once you select this file (you should make it first), the select "Open." Then select "Next >." Then select "Next >" again when Simulation Libraries are discussed.

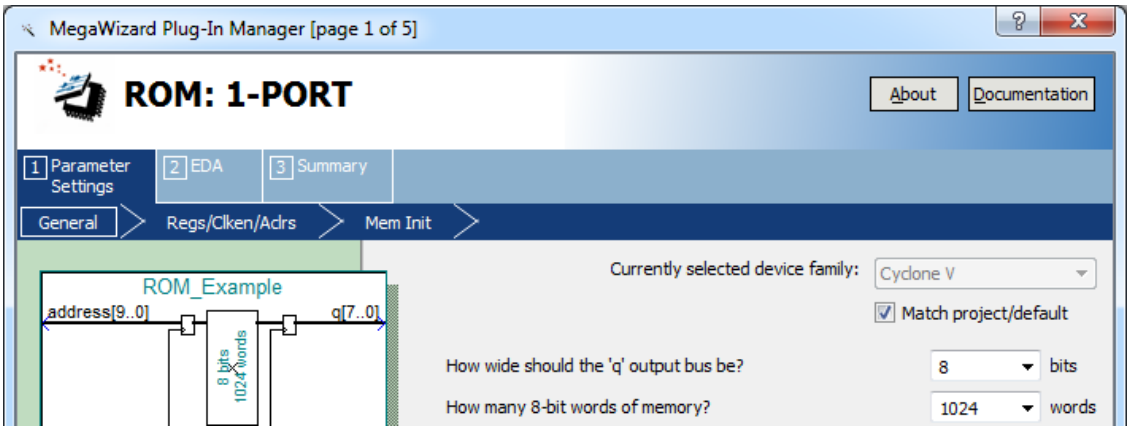

Figure 4: ROM size specifications.

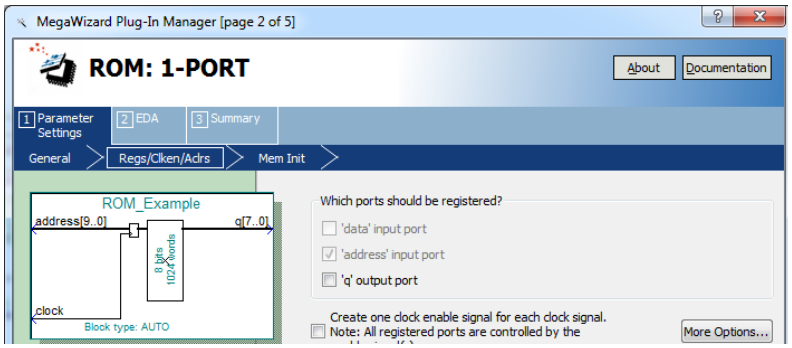

Figure 5: Which ports should be registered?

Quartus ROM Creation Instructions
(in Quartus Prime Lite 18.1)

On the summary page (Figure 6), make sure the .cmp file (VHD component declaration file) is checked and **also check** the .bsf (Quartus II symbol file), as shown. Then select "Finish."

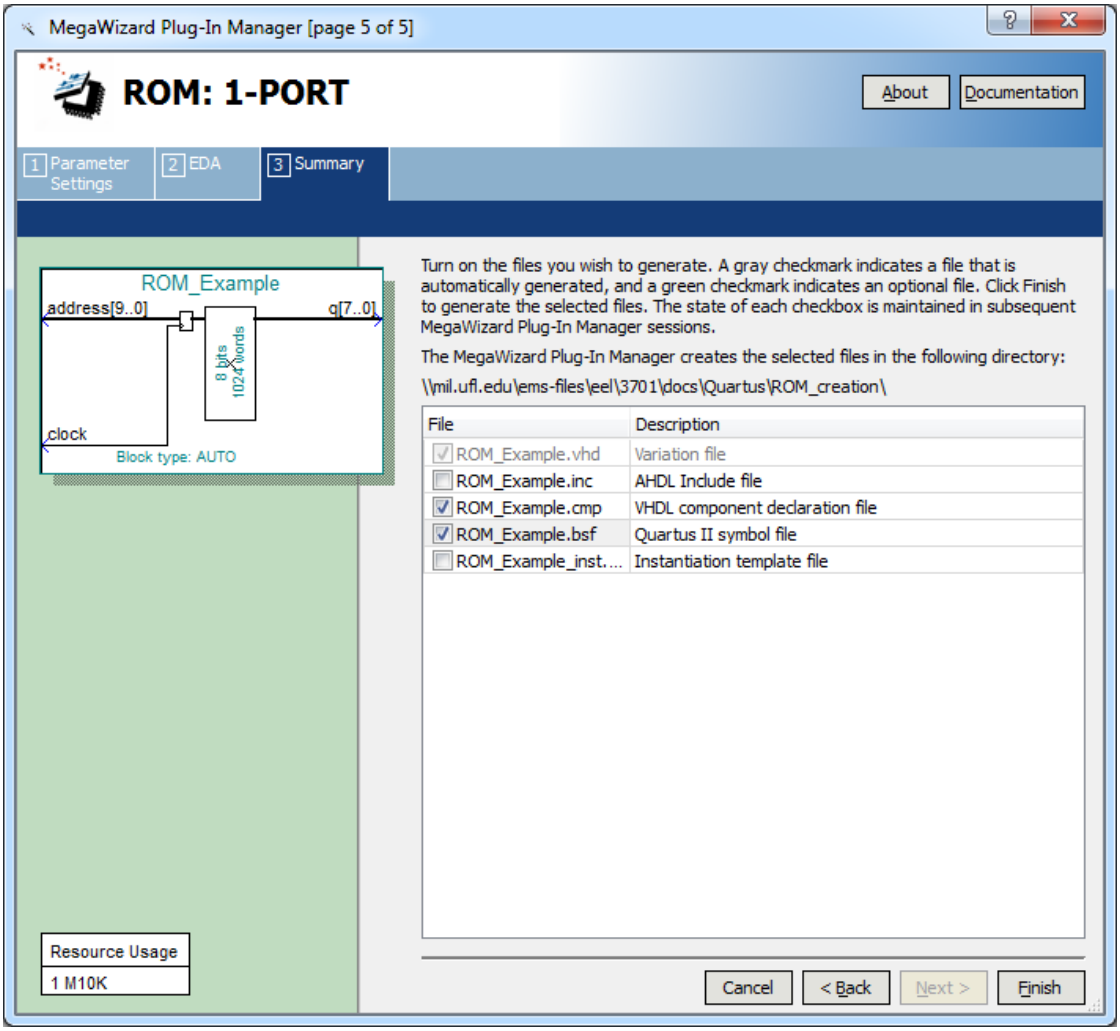

Figure 6: MegaWizard summary.

Select "Yes" on the next screen titled "Quartus II IP Files. (Do not select "Automatically add …")

Create a new bdf file, just as you have been doing all semester. Add a new component, just as you have been doing all semester, i.e., double-click in the bdf window. In the Libraries window select the arrow next to "Project" and then select your ROM, e.g., ROM_Example. Now select "OK." Figure 7 shows what will now appear in your bdf file. Now add address inputs, a clock, and data outputs.

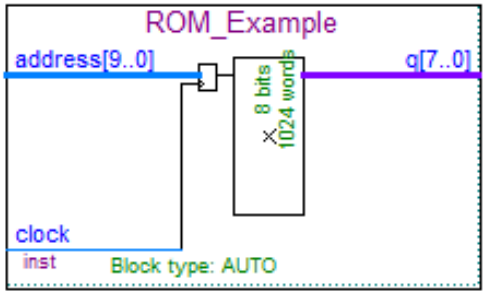

Figure 7: Example ROM.

Quartus ROM Creation Instructions
(in Quartus Prime Lite 18.1)

Add a bus to the address inputs, address[9..0], and a bus to the data outputs, q[7..0], as shown in Figure 8. Also add a clock signal. In Figure 7, I have used A[9..0], D[7..0], and CLK as the example names. They must be in the form of "Name[msb..0]" where msb is the most significant bit's position, starting from zero on the right. You can now use these signals anywhere else in your circuit or as inputs & outputs. Save the file and then do a functional compilation of this design as you normally do.

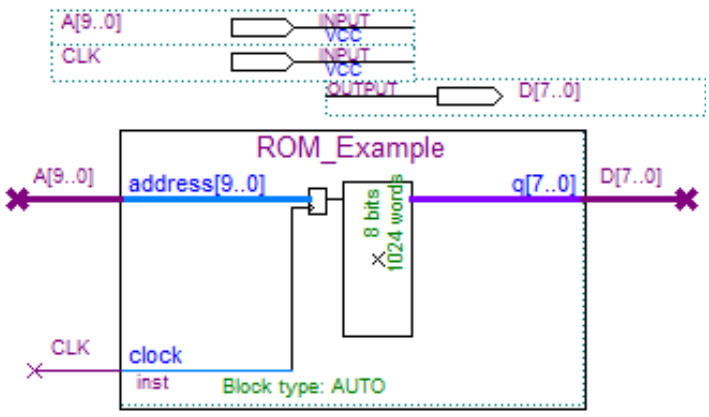

Figure 8: ROM circuit.

Now create a vwf simulation file that includes the addresses of your mif file. Add all of the signals and change the address and data bus signals to hexidecimal (right click on each signal and select radix). Create addresses in your mif file to verify the data inside your ROM. Figure 9 shows a sample mif file.

```
DEPTH = 1024; % Memory depth and width are required   %
WIDTH = 8;    %   Enter decimal numbers for each      %

ADDRESS_RADIX = HEX;       % Address and value radixes are optional       %
DATA_RADIX = HEX;    % Enter BIN, DEC, HEX, or OCT; unless    %
                     % otherwise specified, radixes = HEX           %

-- Specify values for addresses, which can be single address or range

CONTENT
BEGIN

[0..F]    :    0;    % First 16 values are zero %
10        :    33;   % Single address data %
11        :    5C;   % Addr[11] = 5C %
12        :    99;
13        :    A1;   % Addr[13] = A1 %
14        :    B2;
15        :    C3;
16        :    D4;   % Addr[16] = D4 %
[17..3FF] :    FF;   % remaining locations are FF %
END ;                % You must have END statement! %
```

Figure 9: Example MIF file.

Note that the memory clock **SHOULD BE FASTER** than the state machine clock so that memory reads will be completed by the next state. A memory clock that is twice as fast as the state clock is generally sufficient. Figure 10 shows how a T-FF can be used to create the state machine clock (CLK) from the memory clock (MemCLK). **Note that the CLK in the previous figures should therefore use MemCLK and the state machine should use CLK.**

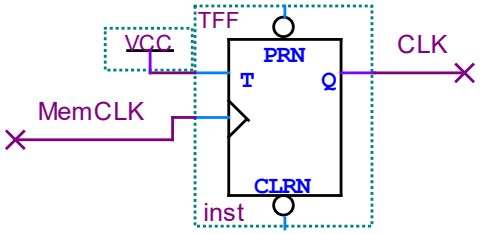

Figure 10: Relation of memory clock (MemCLK) to state machine clock (CLK).