Quartus ROM Creation Instructions

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

Solution:
Use the ROM model found in the "megafunctions | storage" library called "lpm_rom." Unfortunately, Quartus no longer supports asynchronous memory devices with Quartus or their new devices. Therefore, we will have to add a clock that is not needed in the actual EEPROM.

Design Procedure:

1. Create a new project and BDF file (File->New…). I called mine rom_1kx8 and my rom_1kx8.bdf, respectively.
2. For the device family, pick something that has RAM. The Cyclone II family has internal RAM, but the EP2C8T1448 that we use in EEL 4712: Digital Design has only about 2kB. The Cyclone II family, EP2C70F896I8 device (the last on the list in Quartus) has 1,152,000 bit of RAM (= 144,000 bytes), which is more than 128kB. For our example we will use the Cyclone II family, EP2C70F896I8 device, so that we can use this part for both this example and in the future.
3. In your schematic (BDF file), add an "lpm_rom" component found in the "megafunctions" library under "storage." Select "Launch MegaWizard Plug-In." The MegaWizard Manager should now help you with the rest of your design.
4. It does not really matter what "type of output file" you create, but I usually choose VHDL. The MagaWizard will give a default name. I suggest you use it. Select "Next."
5. We will make a 1k x 8 bit device (1kB), so we'll need 8 data bits and 10 address bits (210=1024 words). Answer "8" for "How wide should the 'q' output bus be?" Answer "1024" for "How many 8-bit words of memory?" (Make sure that the next two items are set to auto and that "Single clock" is selected.) Select "Next".

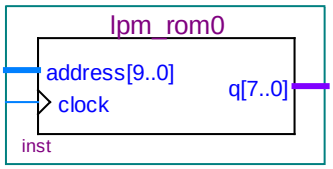

Figure 1: An lpm_rom component.

6. Verify that 'q' output port is NOT selected. Because we deselected the output port, it is not registered. (We would also like to de-select 'address' input port, but it is not possible for this device. Because the address input port is selected, the address is registered.) Select "Next".
7. Enter a new file name for your memory initialization file (MIF) or use a previously made file like "rom_1kx8_data.mif" provided on our web site. Select "Next". Select "Next" again. Then select "Finish".
8. Check the box on the next screen once and select "Yes" and you will not see this window again.
9. Place your ROM device somewhere in your ".bdf" window. You should see something like Figure 1 when you try to place the component onto your schematic:
10. Add a bus to the address inputs, address[9..0], and a bus to the data outputs, q[9..0]. Label the address or data bus by drawing it then typing the name. In Figure 2, we have used A[9..0] and D[7..0] 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 (as shown in Figure 2).

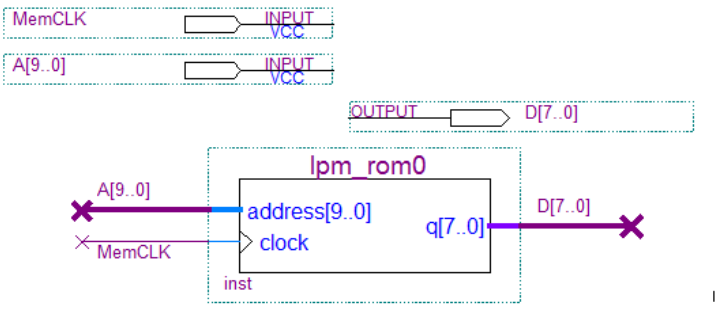

Figure 2: ROM with signals named and defined.

11. Create a MIF file that will contain the memory contents you want programmed in the ROM. This example is called "rom_1kx8_data.mif" and can be created with a text editor in Quartus or other text editiors (i.e., Notepad, emacs, etc.). This file is shown in Figure 3. Note that you can use any base you want (BIN, DEC, HEX or OCT) in the MIF file. The DEPTH is the number of addresses (memory words) in the ROM and the WIDTH is the size of the data bus (the number of data bits per word). This example has 16 locations with zeros, 7 different data values and then $FF for the remaining locations.

```
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 3: Example MIF file, rom_creation.mif.

12. You can also create a MIF file by selecting "File | New | Other Files | Memory Initialization File.
    a. Then you must supply the "Number of Words" and the "Word Size". These are 1024 and 8, respectively, in our example.
    b. A table will now appear. Enter your data directly in this table. All data in this table must used base 10 (unsigned decimal).
    c. Figure 4 shows the results of using or creating the rom_1kx8_data.mif file with a text editor and then opening it in Quartus.
    d. I suggest that you open MIF files in the format shown in Figure 3, not Figure 4. To do this, in the select "Text" in the "Open as" box.
13. To change the MIF file associated with the ROM, double-click on the ROM and go through the MegaWizard Plug-In Manager until you find the screen that associates the ROM with a MIF file.

Addr	+0	+1	+2	+3	+4	+5	+6	+7
000	00	00	00	00	00	00	00	00
008	00	00	00	00	00	00	00	00
010	33	5C	99	A1	B2	C3	D4	FF
018	FF	FF	FF	FF	FF	FF	FF	FF
020	FF	FF	FF	FF	FF	FF	FF	FF
028	FF	FF	FF	FF	FF	FF	FF	FF
030	FF	FF	FF	FF	FF	FF	FF	FF
038	FF	FF	FF	FF	FF	FF	FF	FF
040	FF	FF	FF	FF	FF	FF	FF	FF
048	FF	FF	FF	FF	FF	FF	FF	FF
050	FF	FF	FF	FF	FF	FF	FF	FF

Figure 4: Example MIF file, rom_creation.mif,

14. Save the file, then compile it, and you are now ready to simulate the design.

Simulation:

The simulation is very similar to any other simulation you have performed. I suggest that you select only the already grouped A and D, as well as the MemClk signal. I also suggest that you display the A and D in hexadecimal. The nodes used in simulation were MemCLK, A (A9, A8, A7, … A0) and D (D7, D6, D5, … D0). The address lines were then set to use a "Count Value" for easy simulation. An example is given in Figure 5.

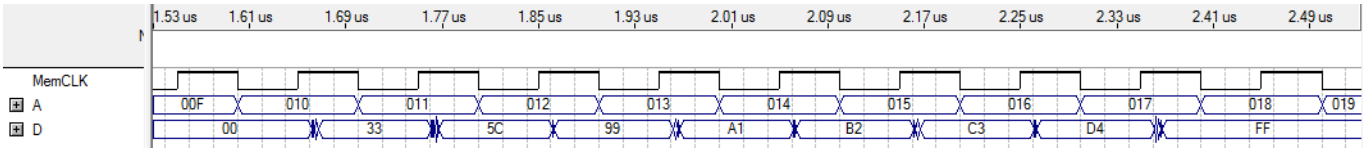

Figure 5: Simulation output.

Very Important Note: The ROMs are very slow. Therefore when you simulate you should change the smallest increment of time to about 100ns and the end time to at least 20µs (where µs=micro seconds in Quartus is selected with "us").

You can increase the simulation end time by selecting "Edit | End Time ..." and entering the desired end time. This window is shown in Figure 6.

To set the count time, select the address bus (A) with the right mouse key, then select "Value", the "Count Value." Now go to the "Timing" menu and change the "Count every" entry to 100ns. This window is shown in Figure 7.

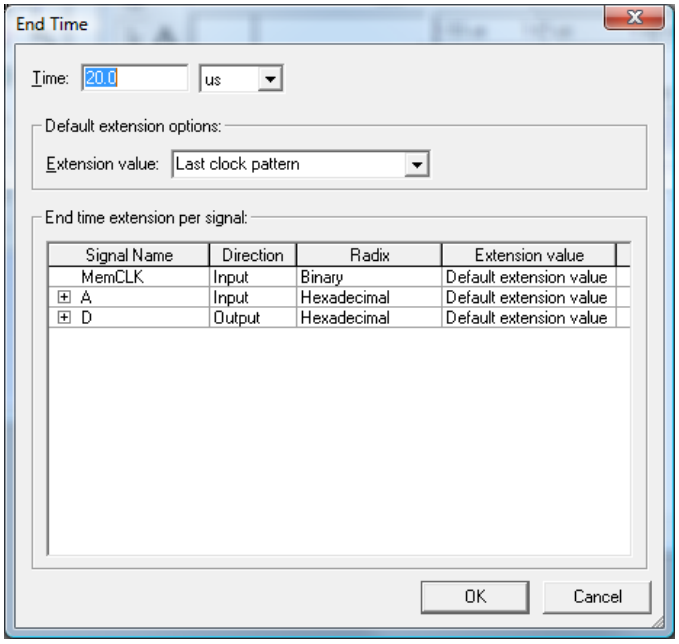

Figure 6: Count value end time.

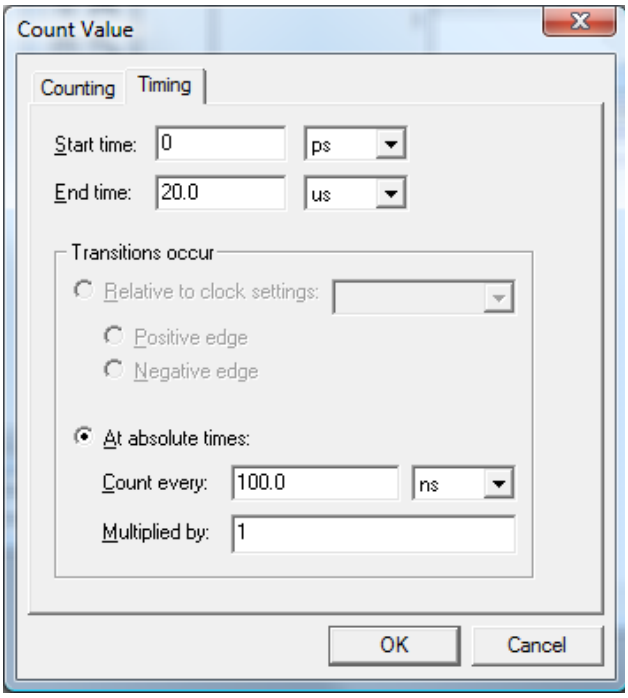

Figure 7: Count value timing increment.

Note also that the memory clock **SHOULD BE FASTER** than the state machine clock so that memory reads will be completed by the next state. Generally, a memory clock that is twice as fast as the state clock is sufficient. Figure 8 shows how a T-FF can be used to create the state machine clock (CLK) from the memory clock (MemCLK).

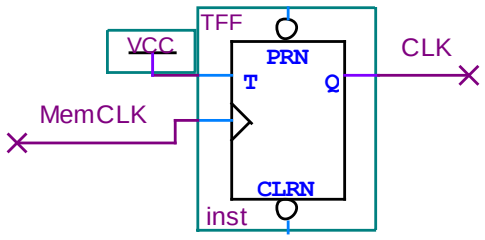

Figure 8: Relation of memory clock (MemCLK) to state machine clock (CLK).