Altera DE2:
DM9000A Ethernet Controller
Application Notes

By: Tarek Kaddoura
Jigar Nahar

Table of Contents

Introduction

The Ethernet controller on the DE2 board is a Davicom DM9000A Fast Ethernet Controller. The controller consists of a 16 KB SRAM, a MAC unit, a 10/100 PHY transceiver, and an interface to communicate with it. This document will explain the steps necessary to add the DM9000A to the hardware configuration, and the drivers needed to get it working.

Hardware Configuration

In order to simplify the hardware configuration process, Terasic has provided the HDL and TCL files necessary to add the component to a system through SOPC builder. These files can be downloaded from
https://www.ualberta.ca/~delliott/local/ece492/appnotes/2013w/Ethernet_DM9000A/DM9000A.tar

SOPC Builder

The Ethernet controller will need to be supplied with a 25 MHz clock. This can be accomplished by creating a new clock, or dividing the system clock by 2. The system clock is assumed to run at 50 MHz. The second approach will be taken here by using an ALTPLL.

Note: The following steps only show what needs to be added to get the DM9000A working. They assume that a NIOSII processor and other components (such as RAM) have already been added to the system.

1. In the downloaded file, there is a folder named hardware. Copy the contents of this folder to the root of your Quartus project.
2. In Quartus, go to Tools → SOPC Builder.
3. Add a Avalon ALTPLL if one doesn't already exist.
4. Use any unused pll clock and give it a clock division factor of 2.
5. In the component list in SOPC Builder, there should be a new category: Ethernet. Under this category is the DM9000A component. Add this component to the system.
6. Generate the system.
7. Add the generated files to your project.

Top Level Modifications

At this point, the system contains the DM9000A component. Now, the component needs to be connected through the top level HDL file. Also, the Ethernet clock needs to be connected to the 25 MHz PLL clock created above.

Note: The following instructions assume that the DM9000A component was named “dm9000a” in SOPC Builder and the 25 MHz PLL clock is called pllclock2. Also, all necessary code is shown in VHDL.

1. Add the pins necessary in your top level entity.

```
ENET_DATA       :       inout       std_logic_vector(15 downto 0);
ENET_CS_N       :       out         std_logic;
ENET_CMD        :       out         std_logic;
```

```
ENET_INT        :   in      std_logic;
ENET_RD_N       :   out     std_logic;
ENET_WR_N       :   out     std_logic;
ENET_RST_N      :   out     std_logic;
ENET_CLK        :   out     std_logic;
```

2. Add the necessary signals into your system component (assuming the DM9000A was called DM9000A_0 in SOPC Builder).

```
-- the_DM9000A_0
ENET_CMD_from_the_DM9000A_0 : OUT STD_LOGIC;
ENET_CS_N_from_the_DM9000A_0 : OUT STD_LOGIC;
ENET_DATA_to_and_from_the_DM9000A_0 : INOUT STD_LOGIC_VECTOR (15 DOWNTO 0);
ENET_INT_to_the_DM9000A_0 : IN STD_LOGIC;
ENET_RD_N_from_the_DM9000A_0 : OUT STD_LOGIC;
ENET_RST_N_from_the_DM9000A_0 : OUT STD_LOGIC;
ENET_WR_N_from_the_DM9000A_0 : OUT STD_LOGIC;
```

3. Add a pll_c2 signal to hold the pll clock created above.
4. Connect the pll_c2 signal to the output of the 25 MHz pll clock.
5. Connect the pins as shown below in the VHDL code.

```
ENET_CMD_from_the_DM9000A_0 => ENET_CMD,
ENET_CS_N_from_the_DM9000A_0 => ENET_CS_N,
ENET_DATA_to_and_from_the_DM9000A_0 => ENET_DATA,
ENET_INT_to_the_DM9000A_0 => ENET_INT,
ENET_RD_N_from_the_DM9000A_0 => ENET_RD_N,
ENET_RST_N_from_the_DM9000A_0 => ENET_RST_N,
ENET_WR_N_from_the_DM9000A_0 => ENET_WR_N
```

6. Connect the pll_c2 signal to the ENET_CLK.
   ```
   ENET_CLK <= pll_c2;
   ```

The DM9000A component should now be connected. The final step is to program the board with the new configuration.

Software Configuration

After adding the component to the system, the software drivers are not added automatically in the NIOS II IDE. The following additions and initializations need to be done in software.

1. Add the files from the “software” folder of the downloaded tar file to the root of your software project.

2. If you named your component something other than “DM9000A” in SOPC Builder, then open the dm9000a.h file and add the line:

   ```
   #define DM9000A_BASE SOPCNAME_BASE
   ```

3. In web_server.c, define the green leds base and lcd display name if they're different than the default ones in the file.

   ```
   #define LED_PIO_BASE GREEN_LEDS_BASE
   ```

#define LCD_DISPLAY_NAME LCD_0_NAME

4. Define the flash name and base in network_utilities.c. Add the following line.

#define EXT_FLASH_NAME CFI_FLASH_0_NAME

#define EXT_FLASH_BASE CFI_FLASH_0_BASE

5. Initialize the controller by adding the following lines in the main() function in web_server.c. Add the dm9000a.h include file first on top in web_server.c.

DM9000A_INSTANCE(DM9000A_0, dm9000a_0);
DM9000A_INIT(DM9000A_0, dm9000a_0);

Application Example: Web Server

Altera has provided an example web server that is really easy to set up. Running the web server will confirm that the DM9000A is indeed working correctly.

In order for the web server to work, the system needs to contain a CFI Flash, and an LCD. The web server's files will be stored in this flash memory. If a CFI Flash is not available, the Ethernet can still be tested by creating a "Simple Socket Server" project instead of the Web Server project. The LCD will be used to display the IP address once connected successfully. Finally, the board needs to be connected to a router through an Ethernet cable, or to a computer using a crossover Ethernet cable.

First, open the NIOS II IDE and follow the instructions below:

1. Go to File → New Project → Web Server.
2. Apply the software modifications mentioned in the previous chapter.
3. Build the web server project.
4. Make sure the board is connected through Ethernet and run the project on the board.
5. After a while, the LCD should display the IP address obtained by the board.

If the IP address does not show up on the LCD, then double check that the software modifications have been correctly applied.

Now that the board is connected and the IP displayed, the board's web site can be accessed by accessing the IP through a browser. A simpler test can be performed by simply pinging the IP of the board from a computer.

Issues and Notes

Running Off Flash

Running the web server off SDRAM works very nicely: the DM9000A is detected, the MAC Address is read from the CFI Flash, and the IP Address shows up on the LCD.

On the other hand, trying to run the software off of flash starts to produce complications. The DM9000A is still detected, but the MAC address isn't read correctly from the flash. This stalls the program and never displays the IP.

Further Reading

The web server and simple socket server provided by Altera are extremely good examples on how to use the Ethernet controller as well as how to set up a whole web server. The altera directory provided on the lab computers has a DE2 demonstration named DE2_NIOS_HOST_MOUSE_VGA that utilizes the Ethernet.

Note: The demonstration projects provided on the University labs are quite old, and will likely lead to hardware or other compilation errors. A newer version of the demonstrations updated to support Quatus II 10.0 are the DE2_70 demonstrations. These demonstrations are not for the Cylone II, but they contain excellent examples of updated code and drivers for the newer Quartus. These demonstrations can be downloaded from here: http://www.terasic.com/downloads/cd-rom/de2_70/DE2_70_demonstrations_V10.rar

The above rar file contains three projects that utilize the Ethernet: DE2_70_NET, DE2_70_NIOS_HOST_MOUSE_VGA and DE2_70_SD_Card_Audio_Player.

References

Previous application notes from winter 2012 for running a web server:
http://www.ece.ualberta.ca/~elliott/ece492/appnotes/2012w/Webserver/

Datasheet for the DM9000A:
http://www.cs.columbia.edu/~%20sedwards/classes/2013/4840/Davicom-DM9000A-Ethernet-controller.pdf