XAPP540 (v1.0) September 17, 2004

An Embedded SMTP Client Using VxWorks and the PowerPC

Author: Rob Armstrong

Summary

This application note describes an embedded Simple Mail Transfer Protocol (SMTP) client reference design that demonstrates the capacity of a network-enabled embedded system to report on its status via E-mail. It describes how to set up the Platform Studio design environment for the PowerPC™ 405, configure the 10/100 Ethernet MAC core, and create the Board Support Package (BSP) for VxWorks®. The reference design is targeted to the Xilinx ML300 Virtex-II Pro demonstration platform and was created using EDK 6.2i service pack 2. The reference design can be targeted to other development boards with minor changes.

Introduction

One advantage of Platform Studio is the ease with which a hardware platform can be constructed in an FPGA and integrated with a software operating system (OS). Operating systems simplify many tasks for embedded software designers. One area in which they excel is the ease with which TCP/IP networking connections can be created. With the 10/100 Ethernet MAC core from Xilinx and the provided drivers, a simple network application, such as the one described in this application note, can be completed in as little as one hour.

Constructing the Reference Design

Creating the Basic Hardware System

The Platform Studio Base System Builder is an easy way to begin designing an embedded processor system targeting a Xilinx FPGA. With the Base System Builder, a sample embedded system including all basic bus infrastructure and many peripherals can be created from a simple menu-driven interface.

For the reference design, the Xilinx ML300 Virtex-II Pro™ XC2VP7-FF676 demonstration board was selected. Peripherals were chosen according to the following system specifications in Table 1 and Table 2. Although the OPB UART16550 was chosen for the reference design, the OPB UART Lite can also be used with VxWorks.

Table 1: **Processor Settings**

Processor	System Clock	Processor Core Clock	Bus Clock	Debug Interface
Embedded IBM PowerPC 405	100 MHz	100 MHz	100 MHz	FPGA JTAG

Table 2: **Included Peripheral Configuration**

Peripheral	Bus Interface	Options
UART 16550	OPB	• Configure as UART16550 • Use Interrupt
10/100 Ethernet MAC	PLB	• No DMA • Use Interrupt

Table 2: **Included Peripheral Configuration *(Continued)***

Peripheral	Bus Interface	Options
General-Purpose I/O (GPIO)	OPB	-
DDR SDRAM Controller	PLB	• Include High-Speed Pipeline Stage
Block RAM Interface Controller	OPB	• 16 kb of memory

This system does not meet timing directly following its creation with Base System Builder. To meet timing, it is necessary to place timing ignore (TIG) constraints on the asynchronous reset and IRQ lines. Also, although the Ethernet MAC core most likely works properly as is, additional timing constraints should be placed into the UCF file to guarantee system performance. The following lines should be added to the UCF file:

```
#### Additional Timing Constraints

##Ignore timing on asynchronous signals
NET C405RSTCHIPRESETREQ TIG;
NET C405RSTCORERESETREQ TIG;
NET C405RSTSYSRESETREQ TIG;
NET EICC405EXTINPUTIRQ TIG;

##Ethernet MAC timing constraints
NET "Ethernet_MAC/phy_rx_clk" TNM_NET = "RXCLK_GRP";
NET "Ethernet_MAC/phy_tx_clk" TNM_NET = "TXCLK_GRP";
TIMESPEC "TSTXOUT" = FROM "TXCLK_GRP" TO "PADS" 10 ns;
TIMESPEC "TSRXIN" = FROM "PADS" TO "RXCLK_GRP" 6 ns;
NET "Ethernet_MAC/PLB_rst" TIG;
NET "Ethernet_MAC/phy_tx_clk" MAXSKEW= 2.0 ns;
NET "Ethernet_MAC/phy_rx_clk" MAXSKEW= 2.0 ns;
NET "Ethernet_MAC/phy_rx_clk" PERIOD = 40 ns HIGH 14 ns;
NET "Ethernet_MAC/phy_tx_clk" PERIOD = 40 ns HIGH 14 ns;
NET "Ethernet_MAC/phy_rx_data<3>" NODELAY;

##Include constraints for DDR SDRAM
NET "sys_clk_s" TNM_NET = "sys_clk_s";
TIMESPEC "TS_sys_clk_s" = PERIOD "sys_clk_s" 7 ns HIGH 50%;
NET "clk_90" TNM_NET = "clk_90";
TIMESPEC "TS_CLK90" = PERIOD "sys_clk_s" 7 ns HIGH 50%;
TIMESPEC "TSCLK2CLK90" = FROM "sys_clk_s" TO "clk_90" 2 ns;
NET "ddr_clk_90" TNM_NET = "ddr_clk_90";
TIMESPEC "TS_ddr_clk_90" = PERIOD "ddr_clk_90" 7 ns HIGH 50%;
```

To ensure proper timing for the DDR controller, the location of the DCM and BUFGs should be constrained. For more information, refer to Xilinx Answer Record 19385.

Configuring the Software Boot Loop

The software image for an operating system most likely will not fit within the memory space available within the FPGA block RAM. Therefore, it is necessary to constrain the software to place the majority of the code and data segments within external memory—in this case, the on-board SDRAM. However, without defined software at the reset vector of the PowerPC the processor boots up into an unknown state.

In most stand-alone applications, either the external memory is initialized at system configuration using a SystemACE™ CF controller or other external loader, or the executable is stored in an external non-volatile Flash memory. Due to the high latency of Flash memories, it

is typically advantageous to copy the code to a fast external memory using a boot loader utility, which is loaded into the FPGA block RAM to execute after the processor comes out of reset.

During the initial process of creating a system, however, it is often advantageous to quickly download new program sources to the processor on the fly, allowing any system changes to be quickly tested and verified without having to rewrite the external non-volatile storage. For these situations, Xilinx has provided a boot loop that continually loops the processor at the reset vector. This ensures that the processor is in a known state from which the executable can be downloaded via a JTAG utility, such as the Xilinx Microprocessor Debugger (XMD).

This reference design uses the boot loop to keep the processor in a known state, so that the system code can be compiled using the Wind River Systems Tornado 2.2 IDE and the executable can be downloaded to the SDRAM via XMD.

To ensure that the boot loop is incorporated into the on-chip block RAM from the Applications tab, right-click on “Default: ppc405_0_bootloop” and select “Mark to Initialize BRAMs”. At this point, Tools -> Update Bitstream can be run to generate a fully finalized and downloadable bitstream, including boot loop. The resulting bitstream is located in $PROJECT_DIR/implementation/download.bit, where $PROJECT_DIR is the top-level directory in which the project was created.

Generating the VxWorks Board Support Package

To easily integrate various hardware setups with third party operating systems, the Platform Studio has the capability to generate BSPs for both VxWorks and Linux. A BSP consists not only of information regarding the memory map of a system, but also a series of drivers to integrate between the included Xilinx device drivers and the standardized system calls of the operating system being targeted.

To generate a BSP for VxWorks, first the MSS file must be configured to target the operating system with the required parameters. To facilitate this, Platform Studio provides a graphical interface from which all of the necessary parameters can be easily set. To configure the MSS file, follow the following sequence of events:

From the System tab, right click on PPC405_0 and select S/W settings.

1. Select the Software Platform tab.
2. In the Kernel and Operating Systems dialog, from the PPC405_0 OS drop-down menu, select VxWorks 5.5.
3. Select the Library/OS Parameters tab
4. Using the drop-down menus, select the STDIN and STDOUT peripherals (in this example, use the UART16550 named RS232).
5. Click the “…” button in the Connected Peripherals row.
6. Click the Add button to add another peripheral to the list, bringing the total to 2.
7. Select the Ethernet MAC and the UART (in this example, use Ethernet_MAC and RS232).
8. Click OK to save these settings.

Specifying the connected peripherals determines which interrupt-driven peripherals are connected to the operating system. All interrupt-driven peripherals should be listed here.

After the proper parameters are set in the MSS file, it is possible to generate a BSP. To generate the BSP, run Tools -> Generate Libraries and BSPs. This creates the BSP in the $PROJECT_DIR/ppc405_0/bsp_ppc405_0 directory.

Creating the Basic VxWorks System

The Wind River Tornado environment simplifies creating the basic operating system image and customizing it to include networking components, shells, etc. To generate the basic operating system used in this reference design, follow these steps:

1. Open up the Tornado environment.
2. From the initial dialog box, or from File -> New Project, under the New tab select “Create a bootable VxWorks image (custom configured)” and click OK.
3. Select a name for the project – this can be anything.
4. Select a location for the project files.
5. Optionally, enter a project description.
6. Click Next to proceed.
7. When presented with the choice to use an existing project or a BSP, select the BSP radio button. This activates the selection dialog for the BSP.
8. Click the “…” button next to the path.
9. Browse to the $PROJECT_DIR/ppc405_0/bsp_ppc405_0 directory, where the BSP was initially created, and click OK.
10. Notice that the default compiler for this BSP has been changed to Wind River’s Diab compiler tool. This is the default compiler for the Xilinx Wind River BSP.
11. Click Next, then Finish.

At this point, the BSP has been successfully imported and the basic operating system is configured. At this point, after compiling the project, it is possible to download the operating system to the development board. It operates successfully, but without shell or user code, it is not of much use.

Activating the VxWorks Target Shell

VxWorks includes a target shell that can be used to monitor and debug programs, as well as load new objects into memory. Since this shell is a useful debugging tool, many users prefer it to be enabled during the debugging phase of their design. Many users also leave the shell enabled indefinitely to allow for remote debugging of their application (although it is not a secure interface). To enable the user shell:

1. From the Workspace browser, select the VxWorks tab.
2. Click the + button next to the project name to expand the project components.
3. Click the + next to “development tool components” to expand it.
4. Right click on “target shell components” and select “Include ‘target shell components.’”
5. A dialog box that contains the different components of the target shell appears. By default, “shell startup script” is not enabled. Click OK to include the target shell.
6. A dialog box appears showing the included components and the components upon which they are dependent. This window also shows the size of the included components. Click OK to continue.
7. Click the Files tab in the Workspace browser to return to the project files.

At this point, the target shell components have been enabled. If the OS is generated and downloaded at this point, the target shell interface appears on STDOUT. Recall that for this reference design, STDOUT is declared as the UART16550 module on the development board.

Disabling the PowerPC 405 Caches

In order to use the PowerPC with certain peripherals, especially those that use DMA, such as the 10/100 Ethernet MAC, steps are necessary to ensure cache coherency. For simplicity, this reference design does not take those steps. Therefore, for this reference design it is necessary to disable cache support in the operating system. To disable cache support:

1. From the Workspace browser, select the VxWorks tab.
2. Click the + button next to the project name to expand the project components.
3. Right click on Hardware and select Properties.
4. Scroll the list box to USER_D_CACHE_ENABLE.
5. Select USER_D_CACHE_ENABLE from the list. In the text box, change TRUE to FALSE.
6. Select USER_I_CACHE_ENABLE from the list and in the text box, change TRUE to FALSE.
7. Click OK to save these settings.

Now that the operating system is configured, it is time to add the user application, which in this case is a simple SMTP client implementation. Before adding the software, it is important to understand what the code is intended to accomplish.

An Overview of SMTP

The SMTP protocol is used in nearly all E-mail applications in use today. Therefore, it provides an ideal interface for an embedded system. In addition, due to the simplicity of the protocol interface, an application can be quickly created using a TCP/IP sockets interface that can interface with nearly every SMTP server in existence. In a system already containing an operating system and an Ethernet MAC, the additional code space necessary to implement an SMTP client is minimal.

The heart of an SMTP transaction is a series of messages passed between the server and the client. Each of these message transactions is a short ASCII text transaction, and each message from the server is preceded by a three-digit code. By parsing this code to determine the server's response, the client can ignore any other text in the server's message. A typical SMTP transaction appears as follows (messages from the server are in italics).

```
220 mailserver.domain.com - Server ESMTP (iPlanet Messaging Server 5.2
HotFix 1.21 (built Sep 8 2003)
HELO embedded_system.com
250 mailserver.domain.com Hello IDENT:root@[192.168.1.1], pleased to meet
you
MAIL FROM:<system@embedded_system.com>
250 <system@embedded_system.com>… Sender ok
RCPT TO:<user@destination.com>
250 <user@destination.com>… Recipient ok
DATA
354 Enter mail, end with "." on a line by itself

From: system@embedded_system.com
To: user@destination.com
Subject: An example SMTP transaction
Content-type: text/html

This is the text of an example e-mail
.
250 Message accepted for delivery
QUIT
221 mailserver.domain.com closing connection - Goodbye
```

Rather than parse the entire text string of the response from the server, which varies depending on the particular implementation of the SMTP protocol user in the server software, a small embedded SMTP application can parse the three-digit response code from the server to determine the intended message. These messages are defined as part of the SMTP specification (RFC 821).

This is the method used by the included reference design. By pre-storing destination addresses and parsing the three-digit code at the beginning of each message from the server, the complete SMTP dialog can be implemented in a relatively small code space.

Adding the Client Application to VxWorks

VxWorks provides a software call during startup that launches any user applications. This function call is contained within the `usrAppInit.c file` located in the main VxWorks project directory. For this application, the usrAppInit(void) function is modified to spawn a task at priority 80. This task monitors the buttons on the demonstration board and, if one is pressed, it opens a connection to the SMTP server and sends an E-mail stating which button was pressed.

The source code for this application exists in the /Software directory of the reference design and consists of the following five files:

- `smtpClient.c`
- `smtpClient.h`
- `userApp.c`
- `userApp.h`
- `usrAppInit.c`

This modified `usrAppInit.c` file provides the task spawn call for the SMTP client and includes the necessary task library. To include these files into the VxWorks project, first copy them to the VxWorks project directory, overwriting the original `usrAppInit.c` file with the new one that spawns the client task. Then, from Tornado, select Project -> Add/Include -> File, then select `userApp.c` and `smtpClient.c` to add them to the project.

After the software application has been added to the VxWorks project, it must be configured to work in your local network. To configure the application, open the `smtpClient.c` file. Change the defined SMTP_SERVER_IP to point to the SMTP server on your local network. Also, change the global variable char *head_to to point to the proper recipient for your tests.

Finally, the VxWorks drivers for the hardware peripherals have some parameters hard-coded into them that might need to be changed, based on the end application. In this case, to use the Ethernet MAC at full speed, the parameter XEEA_DFT_SPEED in `xemac_end_adapter.c` needs to be changed from 10 Mb to 100 Mb. From the Workspace browser, click the + next to External Dependencies to expand the dependencies list. Open the `xemac_end_adapter.c` file, and change the line:

```
#define XEEA_DFT_SPEED 10000000

To

#define XEEA_DFT_SPEED 100000000
```

At this point, the software setup for VxWorks is complete. From Tornado, run Build -> Build to build the software application.

Running the Reference Design

Now that the hardware and software for the reference design have been implemented successfully, it is time to test the application in hardware. The first step is to download the BIT file to the FPGA containing the software boot loop. To do this, either use iMPACT to download the `$PROJECT_DIR/implementation/download.bit` file or click the Download button from XPS. After the bit file has downloaded to the FPGA, the VxWorks image can be downloaded. If iMPACT was used to download the bitstream, close it at this point.

From XPS, click the XMD button in the toolbar or run Tools -> XMD. This opens the XMD command line. Also, open a serial terminal program such as HyperTerminal or Tera Term Pro. By default the UART16550 drivers in VxWorks are configured to use 19200 bps, 8 data bits, no parity, and 1 stop bit, so configure your terminal program accordingly. After XMD and the terminal program are open, download the software using the following commands:

1. Connect to the PowerPC 405 processor with “ppcconnect.”
2. Change to the VxWorks image directory using the “cd” command. For example, “`cd c:\SMTP\VxWorks\default`”
3. Download the VxWorks image using “dow”. By default, the image is named VxWorks, so the command is “dow VxWorks”
4. After the program has successfully downloaded, run the application with “con”.

At this point, the VxWorks splash screen should have appeared in the terminal program, and there should be a VxWorks command prompt visible as well.

To operate the reference design, press one of the buttons located on top of the ML300 board surrounding the TFT display. The client application opens a connection to the SMTP server and initiates the E-mail transaction. Status information is displayed in the terminal program as well. The E-mail contains information about which button was pressed.

Reference Design Files

The reference design files are located on the Xilinx website at: xapp540.zip.

Conclusion

An embedded system with a network interface such as the 10/100 Ethernet MAC or a Gigabit Ethernet MAC can greatly benefit from the addition of an operating system to simplify the interface between the user’s software application and the system hardware. After an operating system is added to the design, additional functionality can easily be added to the system to create capabilities such as the one described in this application note and reference design. The reference design can also be expanded to include configuration of the recipient, SMTP server, etc. And, with the capabilities of Platform Studio and the ease of use of most modern embedded operating systems, time-to-market and development headaches can both be greatly reduced.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
09/17/04	1.0	Initial Xilinx release.