Readme File

ATM/Ethernet IPv4 Forwarding Application

Contents

Legal Notices

Revision History

Date	Revision	Description
November 2003	IXA SDK 3.5	New this release.

1. Introduction

This document describes the IPv4 Forwarding software application for ATM and Ethernet implemented on an Intel® IXP2400 Network Processor. The application is called oc12_atm_gbeth_2401. A detailed description of the application can be found in the *Intel® Internet Exchange Architecture (IXA) Software Building Blocks Applications Design Guide* distributed on the IXA SDK 3.5 Software Framework CD.

1.1 Getting Started

Detailed information on how to arrange the hardware and set up the operating system can be found in the *Intel® Internet Exchange Architecture Software Development Kit (IXA SDK) 3.5.0 Software Framework Installation Guide*.

2. IP Network Configuration

The IP network configuration of the oc12_atm_gbeth_2401 is defined in the following script files:

- `ix_sa_registry_data.xml` – This file contains an XML script. It is located in the `/xscale_config` subdirectory in the oc12_atm_gbeth_2401 root directory. The purpose of this file is to define the application registry. The registry holds information about all ports and their network configuration. A recompilation of the application is required upon any modification of this file.
- `test_configuration_m3.sh` – This is a Linux* shell script. It is located in the `./init_scripts` directory of the oc12_atm_gbeth_2401 root directory. The purpose of this file is to set up the IP Route Table and the VCCs. The script adds routes, next hops, L2 entries, and VCCs for all defined ports. Changes in this file do not require recompilation of the application. The only required action is restarting the application with the modified script.

2.1 Default IP Network Configuration

The oc12_atm_gbeth_2401 application has a default configuration, which allows it to be run without modifications, provided that the IP environment is properly set up to accommodate the IXDP2401 network configuration. Figure 2-1 shows the default network configuration of the oc12_atm_gbeth_2401 application. The appropriate routes have been defined in the `test_configuration_m3.sh` file for Linux.

Table 2-1 lists all defined ports with corresponding IP and MAC addresses. The definition of the ports can be found in the `ix_sa_registry_data.xml` file.

Table 2-1: Network Addresses Assigned to Interfaces

Interface	IP Address	VPI / VCI	MAC Address	Next Hop
ATM1	16.1.0.1	0 / 100	N/A	16.1.0.10
ATM1	16.2.0.1	0 / 101	N/A	16.2.0.10
ETH1	16.3.0.1	N/A	02:01:02:03:00:03	16.3.0.10

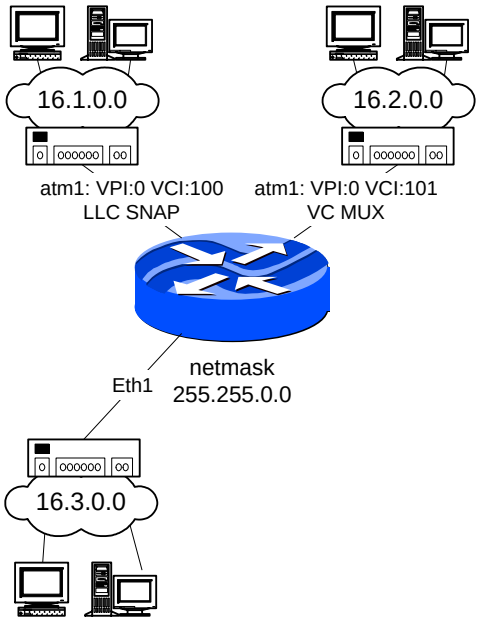

Figure 2-1: Network Configuration

The above default network configuration is sufficient for the supported hardware configurations listed in Table 2-2.

Table 2-2: Hardware Configurations Supported by the Default Network Configuration

Backplane	DB1	DB2
N/A	4xOC-12 ATM	4x1 GbE

If a different hardware configuration is used, the network configuration must be modified appropriately. When considering a setup with multiple boards, one of the boards may use the default network configuration but the remaining boards must have the network configuration customized. The rest of this chapter briefly describes the configuration scripts.

2.2 Route Table

The Route Table configuration is set by the `test_configuration_m3.sh` Linux shell script. Actually the script contains a series of rconfig tool calls. The rconfig is a tool distributed with the oc12_atm_gbeth_2401 application. It adds IP routes, next hops, and L2 entries to the Route Table Manager. The rconfig takes the following parameters:

```
rconfig < addRoute| addNextHop| addV4EthEntry> <parameter string>
```

- `addRoute` – adds a route to the RTM.
- `addNextHop` – adds a Next Hop to the RTM.
- `addV4EthEntry` – adds an L2 entry to the L2 table.
- `parameter string` – the parameter string contains parameters for the action preceding the string. The parameter string holds a series of parameters divided with spaces. The parameters are dependent on the action executed by the rconfig tool.

2.2.1 addRoute

This action adds an IPv4 route to the RTM. The parameter string takes the following parameters:

```
rconfig addRoute "<IP address> <IP mask> <NextHop Index>"
```

- `IP address` – the IP address of the subnet
- `IP mask` – the mask of the subnet
- `NextHop Index` – the index of the corresponding next hop

Note: The next hop entry must be added before the corresponding route, otherwise the addRoute operation will fail.

2.2.2 addNextHop

This action adds an IPv4 next hop to the RTM. The parameter string takes the following parameters:

```
rconfig addRoute "<NextHop Index> <Blade ID> <L2 index> <Port Index>
<mtu> <Flags> <Host IP> <L2 Index type>"
```

- `NextHop Index` – the index of the next hop entry
- `Blade ID` – the identifier of the board
- `L2 index` - the index of the corresponding L2 entry
- `Port Index` – the port index as defined in the ix_sa_registry_data.xml file
- `mtu` – Max Transfer Unit
- `Flags` – various next hop flags. A detailed description on available flags can be found in the *Intel® Internet Exchange Architecture (IXA) Software Building Blocks Developer's Manual* in the "IPv4 Forwarding Microblock" chapter.
- `Host IP` - IP address of the host attached to the port
- `L2 Index type` – indicates the next hop type. For this application the value must be 0 (IPv4).

2.2.3 addV4EthEntry

This action adds a complete IPv4/Ethernet address to the L2 Table. The parameter string takes the following parameters:

```
rconfig addV4EthEntry "<L2 Index> <IP Address> <Dest MAC> <Source
MAC> [name]"
```

- `L2 Index` - Index into the L2 Table to be added, in Hex
- `IP Address` - IP address in dotted-decimal form
- `Dest MAC` - Destination MAC address (hex byte array). The Destination MAC address may be set to all zeros because the application is capable of learning the address from an ARP request.
- `Source MAC` - Source MAC address (hex byte array)
- `name` - Optional named table (or "DEFAULT" if left blank)

2.3 ATM Configuration

The ATM configuration is set by the test_configuration_m3.sh Linux shell script. Actually the script contains a series of atmsar_config tool calls. The atmsar_config is a tool distributed with the oc12_atm_gbeth_2401 application. It adds and removes VCCs and collects statistics per port and per VCC. The atmsar_config takes the following parameters:

```
atmsar_config <addVc | delVc | updateVc | statsVc | statsPort >
<parameter string>
```

- `addVc` – creates a VCC.
- `delVc` – removes a VCC.
- `updateVc` – updates a VCC, the header type is selected, a VCC goes to the active state.
- `statsVc` – shows statistics for a VCC.
- `statsPort` – shows statistics for a port.
- `parameter string` – the parameter string contains parameters for the action preceding the string. The parameter string holds a series of parameters divided with spaces. The parameters are dependent on the action executed by the atmsar_config tool.

2.3.1 addVc

This action adds a VCC. The created VC is not active and cannot receive or transmit cells. The parameter string takes the following parameters:

```
atmsar_config addVc "direction vcid port vpi vci profile"
```

- `direction` – the ATMSAR instance
- `vcid` – the VCC identifier
- `port` – the port number

- `vpi` – the VPI of the created VCC
- `vci` – the VCI of the created VCC
- `profile` – the predefined traffic profile identifier

2.3.2 delVc

This action removes a created VCC. The parameter string takes the following parameters:

```
atmsar_config delVc "direction vcid"
```

- `direction` – the ATMSAR instance
- `vcid` – the VCC identifier

2.3.3 updateVc

This action updates a previously created VCC. The updated VC can receive/transmit cells. The parameter string takes the following parameters:

```
atmsar_config updateVc "direction vcid l2port hdrtype"
```

- `direction` – the ATMSAR instance
- `vcid` – the VCC identifier
- `l2port` – not used by this pipeline
- `hdrtype` – defines the VCC encapsulation

  1 – LLC encapsulation

  2 – VC MUX IPv4

  3 – VC MUX IPv6

2.3.4 statsVc

This action displays the VCC statistics. The parameter string takes the following parameters:

```
atmsar_config statsVc "direction vcid"
```

- `direction` – the ATMSAR instance
- `vcid` – the VCC identifier

2.3.5 statsPort

This action displays the port statistics. The parameter string takes the following parameters:

```
atmsar_config statsPort "direction port"
```

- `direction` – the ATMSAR instance
- `port` – port number

3. Compilation

The oc12_atm_gbeth_2401 application consists of two components:

- Microblocks
- Core Components

Each portion of the application is compiled separately in a different environment.

Note: All applications developed for the IXDP2401 platform must have the IX_PLATFORM_2401 flag defined in the project makefiles for both the Core Components and the Microblocks. An example of required flag definitions may be found in the makefiles of this application.

3.1 Compiling Microblocks

3.1.1 Hardware Project

The microblocks component for hardware is compiled using the IXA_SDK tools in the Workbench environment. To compile microcode that will run on the IXDP2401 development platform for the oc12_atm_gbeth_2401 project, you must do the following:

1. Open the project file named oc12_atm_eth_ai2401.dwp using the Workbench application (use File > Open project). The project file is located in this directory: `<IXA_SDK directory> /src/applications/ipv4_forwarder/oc12_atm_gbeth_2401/`
2. Run the rebuild operation in Workbench by selecting Build > Rebuild

The compiled code will be placed in this directory: `<IXA_SDK directory>/src/ applications/ipv4_forwarder/oc12_atm_gbeth_2401/bin/ai2401`

The image name is oc12_atm_ipv4_eth.uof.

Note: `<IXA_SDK directory>` indicates the directory in which the IXA SDK software was installed.

3.1.2 Simulation Project

The microblocks component for the transactor is compiled using the IXA_SDK tools in the Workbench environment. To compile microcode that will run on the transactor, the following actions must be performed:

1. Open project file named oc12_atm_ipv4_eth.dwp using the Workbench application (use File > Open project). The project file is located in this directory: `<IXA_SDK directory>/ src/applications/ipv4_forwarder/oc12_atm_gbeth_2401/`
2. Run the rebuild operation in Workbench by selecting Build > Rebuild

The compiled code will be placed in the directory `<IXA_SDK directory>/src/ applications/ipv4_forwarder/oc12_atm_gbeth_2401/bin/simulator`.

The image name is oc12_atm_ipv4_eth.uof.

3.2 Compiling Core Components

Framework and application compilation is done under Linux on a host machine. You must define the following variables in the Linux environment:

- IXA_SDK_DEV
  This variable must point to the src directory of IXA SDK, for instance `IXA_SDK_DEV=/user/programmer/vobs/ixa_sdk/ixa_sdk_3.5`
- IXP2XXX_KERNEL_SOURCE_ROOT
  This variable must point to Linux sources. For example, `IXP2XXX_KERNEL_SOURCE_ROOT=/user/programmer/vobs/ixa_sdk/ixa_sdk_3.5/ linux`
- IXP2400_KERNEL_SOURCE_ROOT
  This variable must point to Linux sources. For example, `IXP2400_KERNEL_SOURCE_ROOT=/user/programmer/vobs/ixa_sdk/ixa_sdk_3.5/ linux`
- IXA_SDK_DEV_TARGET
  This variable must point to the directory where all executables and loadable modules will be stored after build, for instance `IXA_SDK_DEV_TARGET=/user/programmer/ixa_cc_targets`

In addition, the host environment must be configured to have access to the cross-compiler for IXP platform, namely xscale_be-gcc.

To compile, perform the following:

1. Change directory to `src/applications/ipv4_forwarder/oc12_atm_gbeth_2401/`
2. Issue this command: `make -f Makefile.linux_kernel`
3. All executables and loadable modules appear in the IXA_SDK_DEV_TARGET directory.

3.3 Compiling Without Core Components

The simulation (transactor) project can be also tested on hardware. It allows testing just the microcode without any support offered by the Core Components. The static configuration described in Section 2.1 is well implemented in Workbench configuration files (*.ind). These files are used to prepare a simple configuration application that may be used instead of the Core Components

application described earlier. That application performs tasks such as Ethernet/ATM driver initialization and setting up the entire configuration with IP routes, VCCs, free lists, etc.

To compile, perform the following:

1. Change directory to: `src/applications/ipv4_forwarder/oc12_atm_gbeth_2401/linux`
2. Issue this command: `make –f Makefile-linux`
3. All executables are located in this directory: `src/applications/ipv4_forwarder/oc12_atm_gbeth_2401/linux/target`

4. Running the Application

4.1 Running on Core Components

This description applies to a configuration where the board boots from TFTP server.

1. Prepare an /ixa directory in root directory in TFTP directory tree.
2. Prepare a /build subdirectory in /ixa directory.
3. Prepare a /me_tools subdirectory in /ixa directory.
4. Copy the oc12_atm_ipv4_eth.uof file to the /build directory.
5. Copy all executables from IXA_SDK_DEV_TARGET directory to the /build directory.
6. Copy the files listed below to the /me_tools directory:
    a. Copy halMeDrv.o, halMev2_lib.o, ossl_lib.o from IXA_SDK_DEV/me_tools/bin_linux_kernel_be
    b. Copy WBSrvr from IXA_SDK_DEV/me_tools/bin_linux_be
    c. Copy make_linux_kernel_devices.sh, oc12_atm_gbeth_2401_run.sh, test_configuration_m3.sh from directory IXA_SDK_DEV/src/applications/ipv4_forwarder/oc12_atm_gbeth_2401/linux_scripts
7. Boot the board. Note you must have the correct Linux kernel image that is appropriate for your version of the IXA SDK.
8. Login as root and change to the /ixa/ directory.
9. Change mode for all files: `chmod 777 *.sh`
10. Change mode for all files: `chmod 777 ./build/`
11. Change mode for all files: `chmod 777 ./me_tools/`
12. Run `oc12_atm_gbeth_2401_run.sh <spin>` where <spin> parameter determines the hardware version of the mezzanine card.
13. Run the configuration script `test_configuration_m3.sh`

Now your system is prepared to forward traffic.

4.2 Running without Core Components

This description applies to a configuration where the board boots from TFTP server.

To run the oc12_atm_gbeth_2401 application without core components, perform the following:

1. Prepare an /ixa directory in the root directory of the TFTP directory tree.
2. Prepare a /build subdirectory in the /ixa directory.
3. Prepare a /target subdirectory in the /ixa directory.
4. Prepare a /me_tools subdirectory in the /ixa directory.

5. Copy all executable files from the IXA_SDK_DEV_TARGET directory to the /build directory.
6. Copy all executable files from the src/applications/ipv4_forwarder/ oc12_atm_gbeth_2401/linux/target to the /target directory
7. Copy halMeDrv.o, halMev2_lib.o, ossl_lib.o from IXA_SDK_DEV/me_tools/bin_linux_kernel_be to the /me_tools directory
8. Copy WBSrvr from IXA_SDK_DEV/me_tools/bin_linux_be to the /me_tools directory
9. Copy init.sh from directory IXA_SDK_DEV/src/applications/ ipv4_forwarder/oc12_atm_gbeth_2401/init_scripts to the /target directory
10. Boot the board. Note you must have the correct Linux kernel image that is appropriate for your version of the IXA SDK.
11. Login as root and change to the /ixa/ directory.
12. Change mode for all files: `chmod 777 ./target/`
13. Change mode for all files: `chmod 777 ./build/`
14. Change mode for all files: `chmod 777 ./me_tools/`
15. Change directory to /ixa/target
16. Run `init.sh <spin>` where <spin> parameter determines the hardware version of the mezzanine card.
17. Open the Workbench project `oc12_atm_ipv4_eth.dwp` located in directory: `ixa_sdk/ixa_sdk_3.0/src/applications/ipv4_forwarder/ oc12_atm_gbeth_2401`
18. Choose debugging on hardware in the Workbench (Debug > Hardware)
19. Modify hardware options:
    a. Hardware -> Options -> Connections -> Connect via Ethernet: enter the blade IP address
    b. Hardware -> Options -> Startup: select “Reset all microengines and load the microcode”
20. Rebuild the project by selecting Build > Rebuild
21. Start debugging by selecting Debug > Start Debugging
22. Launch the microcode by selecting Debug > Run Control > Go