Readme File

POS/Ethernet IPv4 Forwarding Application for IXDP28x1

Legal Notices

Revision History

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

Contents

1. Introduction

This document describes an IPv4 Forwarding software application for Ethernet and Packet over SONET (POS) implemented on an Intel® IXP2800 Network Processor. The application is called 4oc12_pos_6gb_ethernet_2801. 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 4oc12_pos_6gb_ethernet_2801 is defined in the following script files:

- `ix_sa_registry_data.xml` – This file contains an XML script. It is located in the `/config` directory in the 4oc12_pos_6gb_ethernet_2801 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.
- `rtm_config_linux_sh` – This is a Linux* shell script. It is located in the `./linux_scripts` directory of the 4oc12_pos_6gb_ethernet_2801 root directory. The purpose of this file is to set up the Route Table and L2 Table. The script adds routes, next hops and L2 entries 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 4oc12_pos_6gb_ethernet_2801 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 IXDP28x1 network configuration. Figure 2-1 shows the default network configuration of the 4oc12_pos_6gb_ethernet_2801 application. The appropriate routes have been defined in the `rtm_config_linux_sh` file.

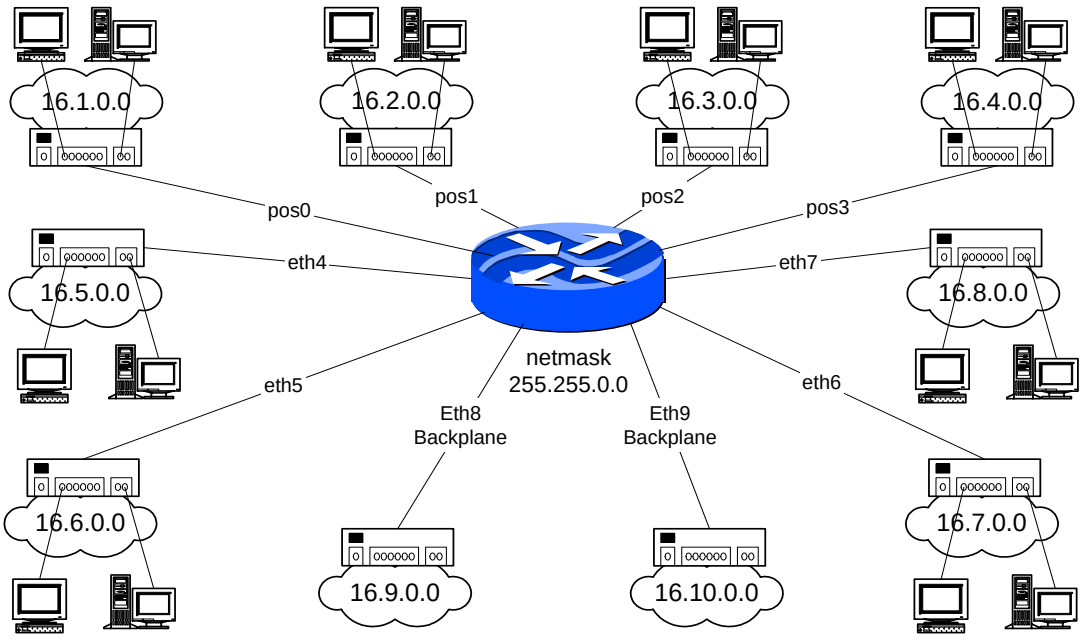

Figure 2-1: Network Configuration

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	MAC Address	Next Hop
POS0	16.1.0.1	N/A	16.1.0.10
POS1	16.2.0.1	N/A	16.2.0.10
POS2	16.3.0.1	N/A	16.3.0.10
POS3	16.4.0.1	N/A	16.4.0.10
ETH4	16.5.0.1	02:01:02:03:00:05	16.5.0.10
ETH5	16.6.0.1	02:01:02:03:00:06	16.6.0.10
ETH6	16.7.0.1	02:01:02:03:00:07	16.7.0.10
ETH7	16.8.0.1	02:01:02:03:00:08	16.8.0.10
ETH8 Backplane	16.9.0.1	02:01:02:03:00:09	16.9.0.10
ETH9 Backplane	16.10.0.1	02:01:02:03:00:0a	16.10.0.10

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
2x1GE*	4xOC-12 POS	4x1GE
2x1GE*	4xOC-12 POS	-
2x1GE*	-	4x1GE
2x1GE*	1xOC-48 POS	4x1GE
2x1GE*	1xOC-48 POS	-
* 2x1Gigabit Ethernet base ATCA interfaces		

If a different hardware configuration is used, the network configuration must be modified appropriately. When considering a setup with multiple IXDP28x1 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 `rtm_config_linux_sh` Linux shell script. Actually the script contains a series of rconfig tool calls. The rconfig is a tool distributed with the 4oc12_pos_6gb_ethernet_2801 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 XML File Description

The ix_sa_registry_data.xml file defines the registry used by the IXA SDK framework to set up several items, including the local ports. The following snippet shows a sample port section of the XML script.

```
<property name="PORT08" type="uint32" value="8">
      <property name="IP_VER" type="string" value="IPV4">
      <property name="IP_ADDR" type="string" value="16.9.0.1"/>
      <property name="IP_MASK" type="string" value="255.255.0.0"/>
      <property name="IP_BCAST" type="string" value="16.9.255.255"/>
      <property name="IP_GATE" type="string" value="16.9.0.254"/>
      </property>
      <property name="MTU" type="uint32" value="1500"/>
      <property name="LINK_SPEED" type="uint32" value="1000"/>
      <property name="MAC_ADDR" type="string" value="020102030009"/>
      <property name="MEDIA_TYPE" type="string" value="1GB_ETHERNET"/>
</property>
```

The above section is for port 8, which is an Ethernet port running at 1 Gbps speed. Table 2-3 presents a brief description of the properties in the port section of the XML script.

Table 2-3: XML Script Properties for Local Ports

Property	Value	Description
PORTXX	X	Port index
IP_VER	IPV4	IP version 4

Property	Value	Description
IP_VER/IP_ADDR	XXX.XXX.XXX.XXX	The host IP address of the port
IP_VER/IP_MASK	XXX.XXX.XXX.XXX	The IP mask of the ports subnet
IP_VER/IP_BCAST	XXX.XXX.XXX.XXX	The broadcast address of the port
IP_VER/IP_GATE	XXX.XXX.XXX.XXX	The default gateway in the ports subnet
MTU	X	Max Transfer Unit as available for the given media type
LINK_SPEED	155 or 622	The link speed property is used to distinguish between OC3 and OC12 on POS/ATM interfaces. For all other interfaces, the link speed is read from the baseboard driver automatically.
MAC_ADDR	XXXXXXXXXXXX	The MAC address of the interface if the media type is Ethernet. For all other media types this property is ignored.
MEDIA_TYPE	POS, ATM or 1GB_ETHERNET	The media type of the port

3. Compilation

The 4oc12_pos_6gb_ethernet_2801 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 IXDP28X1 platform must have the IX_PLATFORM_2801 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

The microblocks portion of the application is compiled using the IXA_SDK tools in the Workbench environment. It is programmed in the IXP2800 microcode. To compile microcode that will run on IXDP28X1 hardware platform for 4oc12_pos_6gb_ethernet_2801 project, you must do the following:

1. Open the project file named 4oc12_pos_6gb_ethernet_2801.dwp in the Workbench application (use File > Open project). The project file is located in this directory:
   `<IXA_SDK directory>/src/applications/ipv4_forwarder/4oc12_pos_6gb_ethernet_2801/wbench_project/`
2. Set appropriate preprocessor flags. For microcode that will be used with core components, you must set the flag USE_IMPORT_VAR instead of SIMULATION. To change flag settings, select Workbench Build > Settings and make changes in Preprocessor definitions.
3. Run the rebuild operation in Workbench by selecting Build > Rebuild

The compiled code will be placed in the same directory as the project file and will have a uof extension.

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

3.2 Compiling Core Components

Framework and application compilation is done under Linux on a host machine. You must define the following three 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`
- IXP2800_KERNEL_SOURCE_ROOT
  This variable must point to Linux sources. For example,

```
IXP2800_KERNEL_SOURCE_ROOT=/user/programmer/vobs/components/
linux_kernel/ixp2000_kernel/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 `ixa_sdk/ixa_sdk_3.5/src/applications/ipv4_forwarder/4oc12_pos_6gb_ethernet_2801`
2. Issue this command: `make -f Makefile.linux_kernel`
3. All executables and loadable modules appear in the IXA_SDK_DEV_TARGET directory.

4. Running the Application

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 subdirectory in the ixa directory (for instance pos_eth/).
3. Copy 4oc12_pos_6gb_ethernet_2801.uof file to the pos_eth/ directory.
4. Copy the files listed below to the pos_eth/ directory:
   a. Copy l2config, rconfig, sa from IXA_SDK_DEV_TARGET directory
   b. Copy ethernet_media_driver.o from IXA_SDK_DEV_TARGET/linux_kernel/xscale_be/ixp2400
   c. Copy everything from directory IXA_SDK_DEV_TARGET/linux_kernel/xscale_be/ixp2400/debug
   d. Copy halMeDrv.o, halMev2_lib.o ossl_lib.o from IXA_SDK_DEV/me_tools/bin_linux_kernel_be
   e. Copy WBSrvr from IXA_SDK_DEV/me_tools/bin_linux_be
   f. Copy make_linux_kernel_devices.sh, run_4oc12_pos_6gb_ethernet_2801.sh, rtm_config_linux.sh from directory IXA_SDK_DEV/src/applications/ipv4_forwarder/4oc12_pos_6gb_ethernet_2801/linux_scripts
5. Boot the board. Note you must have the correct Linux kernel image that is appropriate for your version of the IXA SDK.
6. Login as root and change to the /ixa/pos_eth directory.
7. Change mode for all files: `chmod 777 *.sh l2config rconfig sa`
8. Run `run_4oc12_pos_6gb_ethernet_2801.sh <spin>`
   where parameter `<spin>` determines the hardware version of pos-atm/oc3oc12 mezzanine
9. Run `rtm_config_linux.sh`

Now your system is prepared to forward traffic between Ethernet ports on the front and backplane.