



**TURUN AMMATTIKORKEAKOULU  
TURKU POLYTECHNIC**

# **INTEL IXP1200 NETWORK PROCESSOR AND DIGITAL VIDEO BROADCASTING**

Bachelor's Thesis

December 2001

**Markku Leiniö**

Turku Polytechnic

School of Telecommunications and e-Business

Degree Program in Telecommunications Engineering

Specialization in Internet Technology

Supervisors: Olli Mertanen, D.Sc. (Tech.) and Olli Ojala, B.Sc.

Instructors: Olli Mertanen, D.Sc. (Tech.) and Olli Ojala, B.Sc.

## ACKNOWLEDGEMENTS

This study is based on the research work I did in Software Production Development Center in the summer 2001. I would like to acknowledge especially *Kristiina Sunell*, MBA, the Research Manager in SPDC, and *Olli Ojala*, B.Sc., in charge of Internet Technology Specialization at Turku Polytechnic, for providing me the possibility to do my Bachelor's Thesis in SPDC on this subject.

Turku, 10 December 2001

Markku Leiniö <[markku@iki.fi](mailto:markku@iki.fi)>

## ABSTRACT

Institute: Turku Polytechnic  
Degree Program: Telecommunications Engineering  
Specialization: Internet Technology  
Author: Markku Leiniö  
Title: Intel IXP1200 Network Processor and Digital Video Broadcasting  
Type of Work: Bachelor's Thesis  
Date: December 2001  
Supervisors: Olli Mertanen, D.Sc. (Tech.) and Olli Ojala, B.Sc.  
Instructors: Olli Mertanen, D.Sc. (Tech.) and Olli Ojala, B.Sc.  
Commissioned by: DVB/IP Pilot Project  
Keywords: network processor, IXP1200, routing, digital video broadcasting

The objective of this Bachelor's Thesis was to explore Intel IXP1200 network processor, which is the best-known network processor at the moment. Its internal design and programming principles were to be examined. Digital Video Broadcasting was the other research subject from the network processor point of view.

Example projects were used with the IXP12EB Ethernet Evaluation Kit, which is an example system with 10/100 Mbps and gigabit Ethernet ports, implemented on top of Intel IXP1200 network processor. The study shows how to use the evaluation system.

In this study, the requirements for transmitting the MPEG-2 transport stream were found to be excessive for IXP1200. The processor was still seen suitable for different implementations that contain dynamic properties, like Quality of Service and firewalling systems.

## TIIVISTELMÄ

Korkeakoulu:	Turun ammattikorkeakoulu
Koulutusala:	Tietoliikennetekniikka
Suuntautumisvaihtoehto:	Internet-tekniikka
Työn tekijä:	Markku Leiniö
Työn nimi:	Intel IXP1200 Network Processor and Digital Video Broadcasting
Työn laji:	Insinööri työ
Valmistumisaika:	Joulukuu 2001
Valvojat:	Tekniikan tohtori Olli Mertanen ja insinööri Olli Ojala
Ohjaajat:	Tekniikan tohtori Olli Mertanen ja insinööri Olli Ojala
Yritys:	DVB/IP Pilot Project
Avainsanat:	verkkoprosessori, IXP1200, reititys, digital video broadcasting

Tämän insinööri työnsä tarkoituksena oli tutkia Intel IXP1200 -verkkoprosessoria, joka on tunnetuin verkkoprosessori tällä hetkellä. Sen rakenne ja ohjelmointiperiaatteet olivat keskeisinä kohteina. Toinen tutkimuksen kohteista oli digitaalisen videokuvan levitys verkkoprosessorin kannalta.

Työssä käytettiin IXP12EB Ethernet Evaluation Kit -koejärjestelmää, joka on esimerkkiteutus IXP1200-prosessorista 10/100 megabitin ja gigabitin Ethernet-porteilla. Työ kertoo koejärjestelmän käytön periaatteista.

MPEG-2-videon siirron vaatimukset todettiin tässä työssä liian suuriksi IXP1200-prosessorille. Prosessoria voidaan silti pitää hyvänä vaihtoehtona erilaisiin dynaamisiin verkon laitetoteutuksiin, kuten palomureihin ja palvelun laadun takaaviin laitteisiin.

## ACRONYMS AND ABBREVIATIONS

AAL	ATM Adaptation Layer
ALU	Arithmetic and Logic Unit
ASIC	Application-Specific Integrated Circuit
ATM	Asynchronous Transfer Mode
ATSC	Advanced Television Systems Committee
BSP	Board Support Package
CAT	Conditional Access Table
DMA	Direct Memory Access
DVB	Digital Video Broadcasting
FIFO	First In, First Out; a queue type
Gbps	Gigabits per second, a measure of data speed
IXA	Internet Exchange Architecture
MAC	Media Access Control
Mbps	Megabits per second
MPEG	Moving Picture Experts Group
MPTS	Multi-Program Transport Stream
NIT	Network Information Table
OSI	Open Systems Interconnection
PAT	Program Association Table
PCI	Peripheral Component Interconnect
PCR	Program Clock Reference
PES	Packetized Elementary Stream
PID	Packet Identifier
PMT	Program Map Table
PSI	Program-Specific Information
RTOS	Real-Time Operating System
SDK	Software Development Kit
SDRAM	Synchronous Dynamic Random Access Memory
SPTS	Single Program Transport Stream
SRAM	Static Random Access Memory

## TABLE OF CONTENTS

ACKNOWLEDGEMENTS

ABSTRACT

TIIVISTELMÄ

ACRONYMS AND ABBREVIATIONS

1	INTRODUCTION.....	1
2	NETWORK PROCESSORS.....	2
2.1	The Definition of a Network Processor.....	2
2.2	Network Processors in General.....	2
2.3	Intel IXP1200 Network Processor.....	3
3	IXP1200 ARCHITECTURE.....	4
3.1	Overview.....	4
3.2	Functional Units.....	5
3.2.1	StrongARM Core Microprocessor.....	5
3.2.2	Microengines.....	6
3.2.2.1	Instructions and Execution.....	6
3.2.2.2	Registers.....	8
3.2.3	IX Bus Unit and IX Bus.....	11
3.2.4	SDRAM Unit.....	13
3.2.5	SRAM Unit.....	14
3.2.6	PCI Unit.....	15
4	PROGRAMMING PRINCIPLES.....	17
4.1	The Packet Processing Paths.....	17
4.2	The Operating System on IXP1200.....	18
4.3	Developing Code for the StrongARM Core.....	19
4.4	Developing Code for the Microengines.....	20
4.4.1	IXP1200 Developer Workbench.....	20
4.4.2	Design Projects.....	22
5	IXP12EB ETHERNET EVALUATION KIT.....	24
5.1	Overview of the Evaluation Kit.....	24
5.2	Hardware.....	25

5.3	Setting Up the System .....	27
5.4	Starting Up .....	27
5.4.1	BootMgr and the Basic VxWorks Loader .....	28
5.4.2	Loading the Image .....	29
5.4.3	Configuring and Launching the Target Server .....	30
5.4.4	Opening the Tornado Shell .....	31
5.4.5	Loading the VxWorks NetApp Image .....	32
5.5	Loading an Example Project .....	32
5.6	Setting up the Routing Table .....	33
5.7	Running the Project .....	33
5.8	Benchmarking the Evaluation System .....	34
6	RELATED WORK .....	36
6.1	Balan and Hengartner .....	36
6.2	Karlin, Peterson and Spalink .....	37
6.3	Virtanen .....	37
7	DIGITAL VIDEO BROADCASTING .....	39
7.1	The Reasons for Digital Broadcasting .....	39
7.2	Standards .....	39
7.3	DVB System Overview .....	40
7.4	MPEG-2 Streams .....	41
7.4.1	Program Stream .....	41
7.4.2	Transport Stream .....	42
7.4.2.1	Transport Stream Header .....	42
7.4.2.2	Program Clock Reference .....	44
7.4.2.3	Error Detection .....	44
7.4.2.4	Program-Specific Information .....	45
7.5	Broadcasting a Transport Stream .....	45
8	DVB AND NETWORK PROCESSORS .....	47
9	RESULTS .....	48
10	CONCLUSIONS .....	50
10.1	Problems with the Evaluation System .....	50
10.2	Product Support .....	51
11	FUTURE WORK .....	52
	REFERENCES .....	53

## 1 INTRODUCTION

With the evolution of high-speed backbone network technologies, the requirements for the switching and routing hardware have become higher. More and more packets should be moved between the ports of the device. Also the requirements of the product development processes have increased. The design and manufacturing processes are getting faster. These challenges are presented against the processors of the network devices. Programmable network processors have been designed to fight against these challenges. With different software these processors can handle different implementations.

Traditionally, there are two major processor implementation types for switches. In inexpensive (low-end) models the whole switching process may be made with a general-purpose processor that is programmed accordingly. In faster and more expensive high-end switches ASIC (Application-Specific Integrated Circuit) chips are used. They are specially designed for that specific purpose. However, designing and manufacturing those chips is very expensive and time-consuming.

Intel IXP1200 is a hybrid data processor with high-performance parallel processing power. Its programmability makes faster development processes possible while offering high performance at the same time.

This Bachelor's Thesis shows the properties of Intel IXP1200. It presents the most important programming aspects and introduces the IXP12EB Ethernet Evaluation Kit. The usage and the installation of the evaluation system are shown with an example project. Other research studies are reviewed as well.

Digital Video Broadcasting (DVB) is one major application for high-speed networks. It presents some requirements for the networks and the active devices. Especially the timing aspects with MPEG-2 transport stream are reviewed in this thesis.

## 2 NETWORK PROCESSORS

### 2.1 The Definition of a Network Processor

By network processors it is usually meant programmable processors that are used in network devices. Because of their programmability, they can be used in many applications, just by changing the software.

Network processors are specialized CPUs optimized to support the implementation of network protocols at the highest possible speed. This approach results in unconventional architectures that create new challenges for the software engineer.

A network processor is used in a network traffic manager, which occupies the space between a network interface and a switch fabric in a switch or router. The traffic manager decides where, when and how incoming and outgoing data will be sent next. It strips, adds and modifies packet headers as well as makes routing and schedule decisions. (1.)

### 2.2 Network Processors in General

Network processors are powerful devices and a challenge for embedded software engineers. There is a variety of architectures, but they share a defining characteristic: The network processors sit in high-speed data paths and they manipulate network data at sustained speeds of gigabits per second in software. (1.)

Designing and manufacturing an integrated circuit chip is very expensive. Therefore the ability to use the chip for various applications is an important factor. By using network processors, the product life span can be enhanced. Same physical device can be used while different software releases offer different functions.

The network processor market share is growing. According to a research carried out by Cahners In-Stat Group, the value of the sales of network processors will raise from 200 million dollars this year to seven billion dollars by the year 2005 (2).

### 2.3 Intel IXP1200 Network Processor

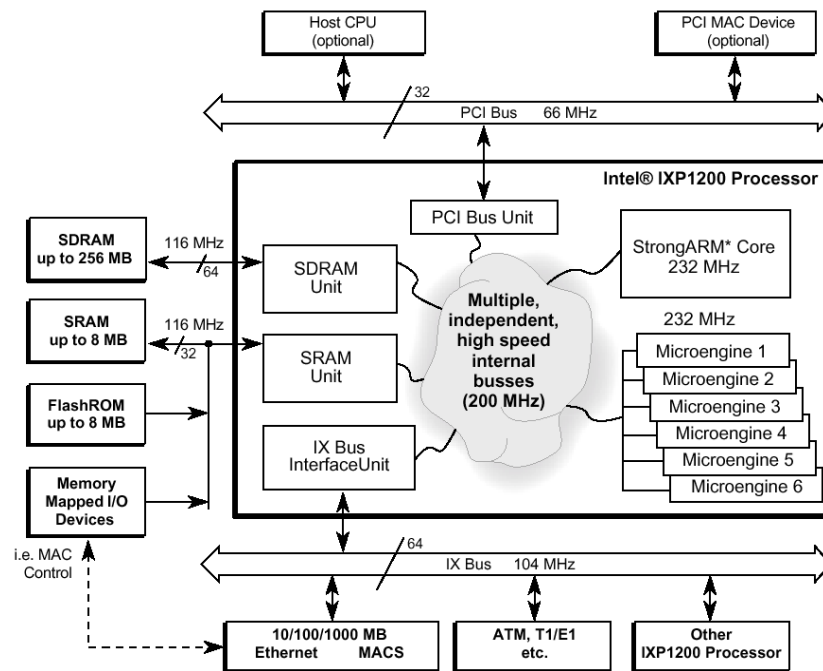
Intel IXP1200 Network Processor family (3) is Intel's alternative for this multi-service network device market. Intel IXP1200 network processors consist of a high-speed processor core and six programmable multithreaded microengines. The processor family was introduced in 1999 and it now has four models: IXP1200 (the basis of the series), IXP1240 (with CRC error checking built in), IXP1250 (with CRC and error-correcting memory support) and IXP1250 for extended temperature range. Functionally and architecturally they are the same with the mentioned exceptions.

Intel's Internet Exchange Architecture (IXA) (4) is a packet processing architecture that focuses on Intel network processors.

### 3 IXP1200 ARCHITECTURE

#### 3.1 Overview

The IXP1200 chip contains a StrongARM core and six independent 32-bit RISC data engines (microengines) as well as SRAM, SDRAM, PCI and IX bus controllers (Figure 1). The operating frequencies are 166 - 232 MHz. The performance of the processor is said to be 3 million packets per second (3 Mpps), which makes 1.5 Gbps (3000000 \* 64 bytes \* 8 bits). Performance can be increased by connecting several processors in parallel. In that way a total capacity of 24 Mpps can be achieved with eight processors. (5, p. 14.)



\* Other names and brands may be claimed as the property of others.

A8488-01

Figure 1. The IXP1200 block diagram (6, p. 2-2)

## 3.2 Functional Units

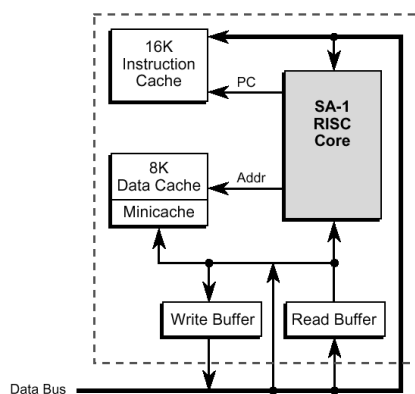
IXP1200 consists of several functional units that communicate with each other in various ways. In general, each unit can work independently and signal other units only when needed. This method helps to eliminate unnecessary waiting for units to complete their work.

The units discussed here are StrongARM core, six microengines, IX bus unit and IX bus, SDRAM unit, SRAM unit and PCI unit.

### 3.2.1 StrongARM Core Microprocessor

The StrongARM core is the same industry standard 32-bit RISC processor as used in the Intel StrongARM SA-1100. It is compatible with the StrongARM processor family currently used in applications such as network computers, Personal Digital Assistants (PDAs), palmtop computers and portable telephones. It provides high performance in a low-power, compact design. (5, p. 14.)

The StrongARM core has two caches: a 16-kilobyte cache for instructions and an 8-kilobyte cache for data (Figure 2). In addition, the core contains a 512-byte minicache, which is intended for data that is read once, operated on, and then discarded. The intention is to reduce flushing of the main cache.



A8516-01

Figure 2. The StrongARM block diagram (6, p. 2-15)

The StrongARM core may be used in different ways, depending on the application. If the system already contains a high-level host processor, the StrongARM core can run a mini-kernel and execute routing protocols, while the microengines do the fast-path packet processing (6, p. 2-15).

In designs where there is no other host processor, the StrongARM is the main processor. As the microengines are used in actual data transmission, the StrongARM core is used for running a Real-Time Operating System (RTOS) and more complex tasks such as address learning, network management and building and maintaining forwarding tables.

The StrongARM core operates at the nominal frequency of the processor (166 - 232 MHz).

### 3.2.2 Microengines

Six 32-bit, multithreaded RISC data engines (microengines) perform data movement and processing without assistance from the StrongARM core. The microengines are especially designed for data transmission and handling with bit, byte, word and longword operations.

The microengines are programmed with symbolic microcode that is developed with Intel IXA Software Development Kit (SDK). Each microengine has four independent program counters. That makes a total of 24 microcode contexts in a single IXP1200 processor. The threading is hardware-based and context switching within each microengine requires no extra instructions.

#### 3.2.2.1 Instructions and Execution

Each microengine contains its own instruction store (Program Control Store), its own set of 256 registers and other Control/Status Registers (CSRs). All the microengines are identical so that the functions may be interchanged between them. There are no fixed

functions for any of the microengines because they are all fully programmable. (6, p. 2-17.)

The microengines operate at the core clock frequency of the IXP1200 like the StrongARM core. All microengine instructions execute in one clock cycle. The microengines are implemented as five-stage pipelined processors. In the first stage, the instruction is fetched from the instruction store; in the second stage, the instruction is decoded; in the third stage the operands are fetched from the registers; in the fourth stage the operands proceed through the ALU (Arithmetic and Logic Unit) and in the fifth stage, the result is written out to the destination register. (6, p. 2-17.)

As each microengine has only one instruction store, each four threads running in the same microengine have basically the same program code. This is not a limitation, however, because each thread can start the execution at different address. The instruction store has 2048 instruction words (earlier steppings of IXP1200 have only 1024 words). Each instruction is a 32-bit longword. The instruction store is loaded by a loader program running in the StrongARM core at the boot time.

### Deferred Instructions

In general, branching inside the program code destroys the idea of pipelining the instruction execution. As told earlier, the processor pipeline is loaded with the instruction a few steps before its actual execution. When a branch instruction is executed, the instructions following the branch instruction may be aborted and their operands have to be discarded. Then the pipeline has to be loaded again from the new point of execution.

To avoid this kind of penalties, the IXP1200 implements deferred execution of instructions. Using this method, up to three instructions that are usually placed before the conditional branch instruction, can be placed after the branch instruction, and the microengine will still load them before branching. While still executing those instructions, the microengine can already fetch the instructions and operands from the branch destination.

For example, the instruction `"br [target#] , defer [2]"` (unconditional branch) changes the execution to label "target" after executing two instructions following the instruction.

The microcode assembler does this optimization automatically although the programmer can use this property himself too.

## Multithreading

Many microengine instructions need to access other units of IXP1200. Usually those external (from the microengine point of view) accesses are slower than the operating speed of the microengines. The hardware-based multithreading enables other threads to run while one thread is waiting for memory or any other external unit access results. There can always be some code running when some other code is stopped.

The `"ctx_swap"` optional token is used to instruct the thread context to be swapped while waiting for data to be complete.

## Branch Prediction

Branch prediction is another method that is used to optimize the performance. The microengine can be instructed to fetch next instructions from the location of the branch destination. That can help keeping the execution pipeline filled with valid instructions if the branch is predicted correctly. The `"guess_branch"` optional token indicates the branch prediction.

### 3.2.2.2 Registers

There are two kinds of registers in the microengines: general purpose registers (GPRs) and transfer registers. The 128 general purpose registers are in turn divided into two banks, an A bank and a B bank. This is done to enable the ALU to fetch two operands

simultaneously. The ALU instructions use one operand from the A bank and one from the B bank (Figure 3).

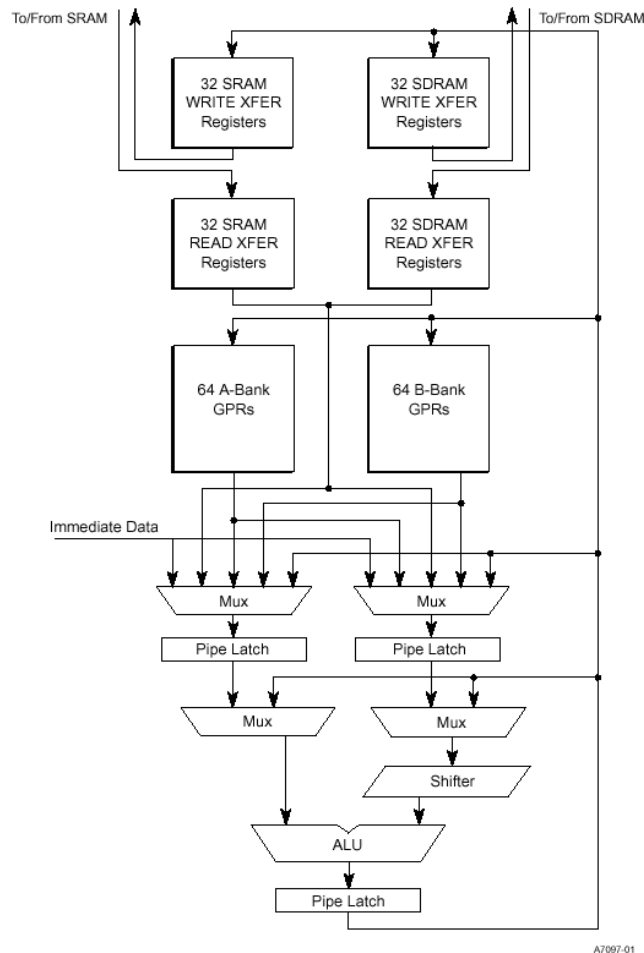


Figure 3. Microengine data path (7, p. 2-2)

The transfer registers are divided into SDRAM and SRAM transfer registers, and they are divided into read and write portions. That makes 32 registers in each portion, and 128 registers in total. Each of these sets of registers can be used at the same time because they have separate data paths in respective functional units.

Registers can be accessed in two addressing mode: context-relative and absolute. In context-relative mode each thread gets its own register. In other words, one register name is actually associated with four different physical registers, depending on the thread that is addressing the register. In absolute addressing mode data can be shared between threads because each thread is addressing the same physical registers.

The registers are named symbolically. The programmer names the registers he uses and the microcode assembler assigns the names to appropriate registers. There are some character prefixes that are used to indicate the type of the register (GPR, SDRAM, SRAM) and the addressing mode (Table 1).

Table 1. Microengine register naming conventions (7, p. 2-4)

<b>Register name form</b>	<b>Meaning</b>
regname	Context-relative GPR
@regname	Absolute GPR
\$regname	Context-relative SRAM transfer register
@\$regname	Absolute SRAM transfer register
\$\$regname	Context-relative SDRAM transfer register
@\$\$regname	Absolute SDRAM transfer register

Since the microengines do all the hard work while transferring data, they have to have certain access to other units of the processor. Those resources are listed in Table 2.

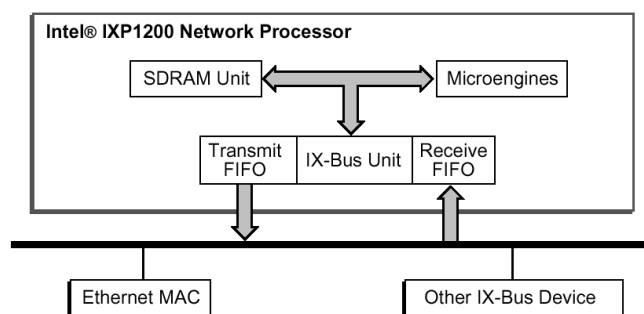
Table 2. Microengine accessible resources (6, p. 2-20)

<b>Unit</b>	<b>Resource</b>
StrongARM core	The microengines may interrupt the StrongARM core. The StrongARM core can read a register to determine which microengine generated the interrupt. No other access to StrongARM core.
Microengines	Each microengine has access to its Programs Control Store so it can execute instructions. The microengines do not have read or write access to the Control Store (they cannot read or write their own Control Store, only the StrongARM core can do that). Inter-thread signaling between microengine threads is provided. Each microengine is self-contained, so one microengine cannot access the Control Store or registers of another microengine.
IX bus unit	Full access to the IX bus unit, including the scratchpad RAM, hardware hashing unit, receive and transmit FIFOs, Ready bus and control/status registers.
SDRAM unit	Full access to SDRAM.
SRAM unit	Full access to SRAM, including flash and other devices hooked up to the SRAM bus.
PCI unit	No access to the PCI bus. Access only to the control/status registers for the two DMA controllers in the PCI unit. By programming these registers the microengines may initiate DMA transfers between a block of SDRAM memory and a PCI device.

Hardware semaphores are used to signal different conditions to ensure that other hardware units can fully utilize the microengines. (5, p. 14.)

### 3.2.3 IX Bus Unit and IX Bus

The IX bus unit (also referred as the FBI unit) controls the IX bus. It transfers data to and from the receive and transmit queues, receive FIFO and transmit FIFO. The IX bus connects the processor to MAC (Media Access Control) devices such as 10/100 Mbps or gigabit Ethernet controllers and also to parallel IXP1200 processors (Figure 4). The operating frequency is 66 - 104 MHz and the maximum capacity is 4 - 6.26 Gbps depending on the operating frequency of the processor. (5, p. 15.)



A8520-01

Figure 4. IX bus data flow (6, p. 2-29)

The IX bus can be configured as either a 64-bit bidirectional bus or as two 32-bit unidirectional buses, which are transferring data independently to different directions.

Once the data is in the receive FIFO, it can be transferred to either the microengines (their transfer registers) or to SDRAM. Data to be transmitted to devices on the IX bus can only come from the transmit FIFO, and then transmitted from there to the devices on the IX bus.

The IX bus unit contains (among the receive and transmit FIFOs) control and status registers, a 4-kilobyte scratchpad RAM and a hash unit for generating 48- and 64-bit hash

keys. There is also a sideband bus operating in parallel to the IX bus. That is called the Ready bus. It consists of eight data pins and five control pins. The Ready bus operates synchronously with the IX bus.

The receive and transmit FIFOs are constructed as tables of 16 \* 64 bytes, and the software can operate on each component as needed. The FIFOs are shared by all of the microengine threads. Software has to ensure that the FIFOs are used in a sensible way by all the threads. (6, p. 2-29.)

The IX bus unit is connected to other units as shown in Table 3.

Table 3. IX bus unit internal connections (6, p. 2-32)

Unit	Access
StrongARM core	Limited access. The StrongARM core can access the scratchpad RAM within the IX bus unit, as well as a number of control/status registers within the IX bus unit. The StrongARM core cannot access the receive and transmit FIFOs within the IX bus unit. The StrongARM core can program the Ready bus controller within the IX bus unit.
Microengines	Full access, including access to the scratchpad RAM, the hashing unit, programming the Ready bus controller, the receive and transmit FIFOs and all control/status registers.
SDRAM unit	Two direct, unshared, independent busses connect the SDRAM unit with the IX bus unit. The SDRAM read bus is connected to the receive FIFO, while the write bus is connected to the transmit FIFO.
SRAM unit	No connection between the SRAM unit and the IX bus unit's FIFOs. Data that needs to be transferred between the IX bus (transmit and receive FIFOs) and SRAM must be transferred by the microengines (by way of their transfer registers).
PCI unit	No connection between the IX bus unit and the PCI unit. Data that needs to be transferred between the IX bus (transfer and receive FIFOs) and the PCI bus must be transferred by the microengines (by way or their transfer registers).

### 3.2.4 SDRAM Unit

The IXP1200 provides an SDRAM unit to access low cost, high bandwidth memory for mass data storage. SDRAM is used for forwarding information and transmit queues. The StrongARM core address space allows up to 256 megabytes of SDRAM to be addressed.

The SDRAM bus is 64 bits wide. It is accessed using quadword (64 bits, 8 bytes) operations. When byte, word or longword operations occur from the StrongARM core or PCI unit, a quadword is read from SDRAM. Only the necessary bytes are modified. Finally the entire quadword is written back to SDRAM. These three steps (read-modify-write) are performed automatically. (5, p. 19.)

One microcode instruction can initiate a transfer of 16 quadwords (128 bytes) at a time. From the microengines only quadword accesses are supported. Less than 8 bytes can be written using the byte mask within an instruction, but it results in read-modify-write cycles (5, p. 19).

SDRAM unit internal connections are summarized in Table 4.

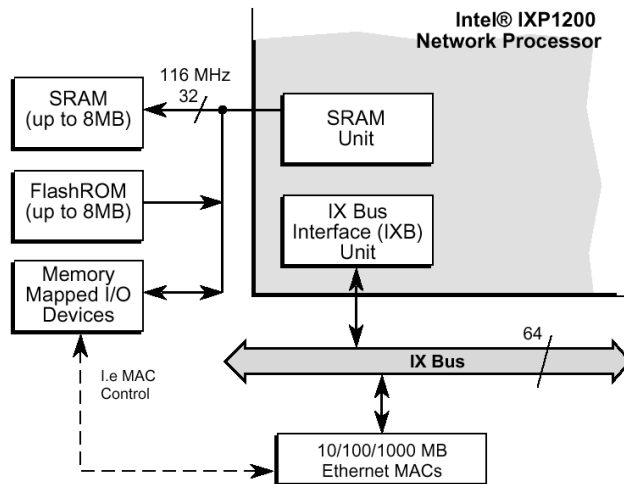
Table 4. SDRAM unit internal connections (6, p. 2-26)

<b>Unit</b>	<b>Resource</b>
StrongARM core	Full access.
Microengines	Full access.
IX bus unit	There is a 32-bit bus connecting the SDRAM unit and the IX bus unit's receive and transmit FIFOs.
SRAM unit	No connection between the SDRAM unit and the SRAM unit.
PCI unit	There is a separate and unshared 32-bit bus connecting the SDRAM unit and the PCI bus unit. This allows devices on the PCI bus better access to data buffers within the SDRAM unit. The two DMA controllers in the PCI unit have access to SDRAM data.

The SDRAM interface operates at half the core frequency, providing a peak bandwidth of 928 megabytes per second at 232 MHz (5, p. 18).

### 3.2.5 SRAM Unit

An SRAM unit is provided for very high bandwidth memory for storing the lookup tables and other data for the packet processing microengines. The SRAM unit controls the SRAM (up to 8 megabytes), FlashROM (up to 8 megabytes) for booting and 2 megabytes of SlowPort address space for peripheral device access (Figure 5). (5, p. 20.)



A8517-01

Figure 5. SRAM unit external interfaces (6, p. 2-22)

In the figure above, the SlowPort address space is used for interfacing to Ethernet MAC access ports for MAC programming and for accessing MAC manageability registers (6, p. 2-21).

The SRAM interface is 32 bits wide. That is half of the SDRAM bus width because SRAM is not intended for bulk data storage but for fast lookups.

Like for the SDRAM interface, the SRAM interface operating frequency is half the core frequency.

SRAM unit connections are presented in Table 5.

Table 5. SRAM unit internal connections (6, p. 2-24)

Unit	Access
StrongARM core	Full access.
Microengines	Full access.
SDRAM unit	No connection between the SRAM unit and the SDRAM unit.
IX bus unit	No connection between the SRAM unit and the IX bus unit.
PCI unit	No connection between the SRAM unit and the PCI unit.

### 3.2.6 PCI Unit

The PCI unit provides an industry standard 32-bit PCI (Peripheral Component Interconnect) bus to interface to PCI peripheral devices such as host processors and MAC devices. The PCI unit supports operating speeds up to 66 MHz and is "PCI Local Bus Specification Revision 2.2" compliant. Above 33 MHz operation, only two PCI devices are supported: the IXP1200 and a second PCI device. To support more devices on higher frequencies, a PCI-to-PCI bridge can be used. (5, p. 23.)

The PCI unit is connected to the SDRAM unit and the StrongARM core. Devices on the PCI bus have full access to SDRAM. The StrongARM core also has full access to the PCI interface and can be a bus master.

Two DMA controllers are integrated into the PCI unit. Both of them can be used by either the StrongARM core or the microengines. The DMA controllers are programmed by means of DMA descriptors that reside in SDRAM. The descriptors can be chained so that non-contiguous blocks of data from SDRAM can be transferred to PCI as one block.

When transferring data between the PCI unit and IX bus unit, the microengines have to be used. Typically, the data would be transferred from the IX bus unit's FIFOs directly to SDRAM. Then, the microengine thread would use PCI DMA controller to transfer data from SDRAM to the PCI interface. (6, p. 2-28.)

Table 6 shows the PCI unit internal connections.

Table 6. PCI unit internal connections (6, p. 2-28)

<b>Unit</b>	<b>Resource</b>
StrongARM core	Full access. The StrongARM core can access any devices on the PCI bus. The StrongARM core can program any of the PCI unit's control/status registers.
Microengines	The microengines can only access the control/status registers for the DMA controllers within the IX bus unit. No other access. The microengines cannot access devices on the PCI bus (and vice versa).
IX bus unit	No connection between the IX bus unit and the PCI unit. Data that needs to be transferred between the IX bus (receive and transmit FIFOs) and the PCI bus must be transferred by the microengines (by way of their transfer registers).
SDRAM unit	Unshared, direct 32-bit bus connecting the SDRAM unit and the PCI unit. Devices on the PCI bus have access to SDRAM. DMA controllers within the PCI unit can transfer data between SDRAM and devices on the PCI bus.
SRAM unit	No connection between the SRAM unit and the PCI unit. Devices on the PCI bus cannot access SRAM. DMA controllers within the PCI unit cannot access SRAM.

## 4 PROGRAMMING PRINCIPLES

### 4.1 The Packet Processing Paths

In network processor architectures there can be found concepts called *fast path* and *slow path*. The fast path is usually the native way of transferring data in certain architecture. It is very efficient and brings the most of the performance in use. The fast path is used when no special operations for the data packets is needed. The fast path is found in the data processor half (data plane) of the network processor. In IXP1200, the fast path is serviced by the microengines (Figure 6).

If the packet entering the system needs some extra servicing, as for example routing protocol packets or filter configuration packets do need, the packet has to be sent to the slow path. The slow path is handled by the control processor part of the network processor (control plane). As its name tells, the slow path is slower than the fast path. In the slow path, however, the packets can be manipulated and processed in more complex ways than in the fast path. The slow path is implemented by the operating system running in the StrongARM core of IXP1200.

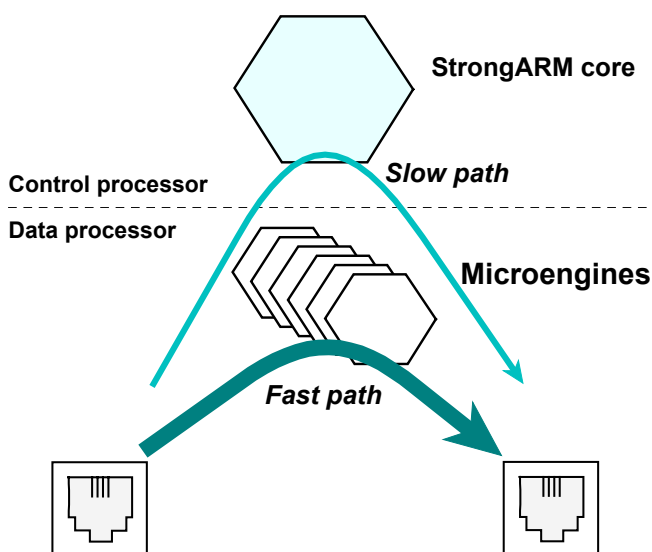


Figure 6. The slow and the fast path in IXP1200 architecture

For incoming data, the decision between the data plane and the control plane has to be chosen carefully. If the performance is critical, as it usually is in gigabit-class devices, the slow path processing has to be kept low.

Because the different natures of the StrongARM core and the microengines, IXP1200 programming is divided into two parts: StrongARM core programming and microengine programming. The code for them is developed separately.

## 4.2 The Operating System on IXP1200

The IXP1200 system needs an operating system to be able to run. In general, an operating system is defined as the low-level software that provides the interface to peripheral hardware, allocates storage, schedules tasks and presents an interface to the user.

In the IXP1200 system, the operating system does the overall process management work. It has all the necessary routines to manipulate address and forwarding tables so that the code running in microengines can decide packet forwarding destinations. The operating system runs software that loads the microengines' instruction stores and starts and stops the microengines when necessary.

There are two operating systems supported by Intel: VxWorks and an embedded Linux.

VxWorks is a real-time operating system developed by Wind River Systems, Inc. (8). It is a very popular operating system in embedded industry. It is the run-time component of Tornado II embedded development platform.

VxWorks is interfaced with the embedded target hardware by the Board Support Packages (BSP). Intel ships IXA SDKs with a BSP for IXP1200 StrongARM core. The BSP includes the core libraries of the VxWorks operating system ported to that specific processor.

Intel also supports the use of an embedded Linux system with IXP1200. The Intel IXA SDK 2.0, which became publicly available in the fall 2001, includes the Embedded Linux

IDE (Integrated Development Environment) to develop code for Linux. Intel has released a how-to document of setting up a Linux system on IXP1200 (9).

This study only reviews the use of VxWorks in the IXP1200 system. Whichever operating system is selected and used, the operating system will be running on the StrongARM core of IXP1200.

### 4.3 Developing Code for the StrongARM Core

The code for the StrongARM core is developed using Tornado II by Wind River Systems, Inc. It is an embedded development platform running on 32-bit Unix and Windows workstations (Figure 7).

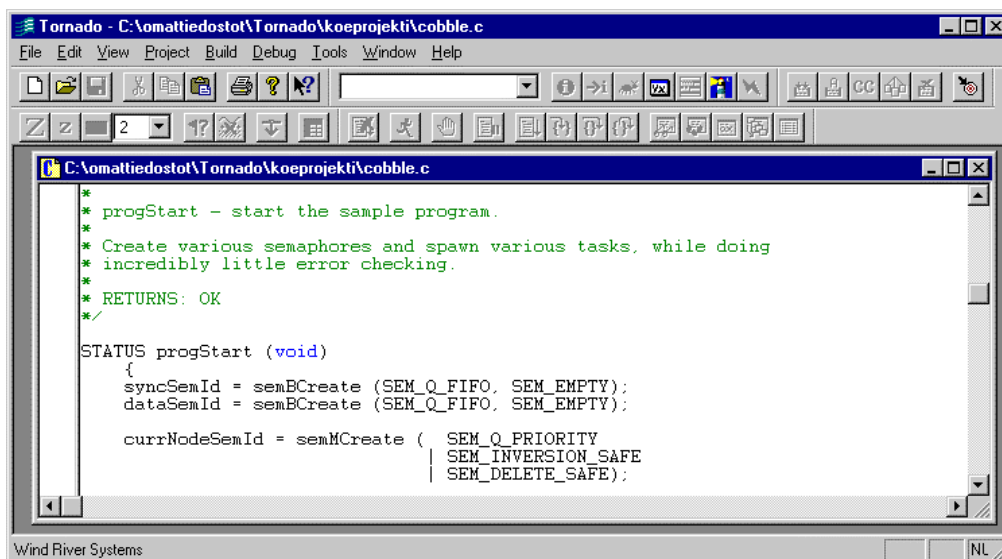


Figure 7. Tornado II embedded development environment on Windows workstation

Tornado comprises the following elements:

- VxWorks, a high-performance real-time operating system
- Application-building tools (compilers and associated programs)
- An integrated development environment (IDE) that facilitates managing and building projects, establishing and managing host-target communication, and running, debugging and monitoring VxWorks applications. (10, p. 1.)

Tornado environment is used to compile the C/C++ language source files into binary executables that can be downloaded into the IXP1200 system. In particular, a bootable image of the operating system can be stored in the flash memory of IXP1200 system so that the operating system is loaded automatically during the power-up of the system. Other binary images can be loaded afterwards.

Chapter 3.2, Functional Units, told about the resources that the StrongARM core has access to in other functional units. Those resources can be used from the code developed with the Tornado environment.

#### 4.4 Developing Code for the Microengines

##### 4.4.1 IXP1200 Developer Workbench

The code for the IXP1200 microengines is developed using IXP1200 Developer Workbench that is included in the Intel IXA SDK. The code is symbolic microcode that is assembled with an assembler before linking. Figure 8 shows an overview of the whole development process for the IXP1200.

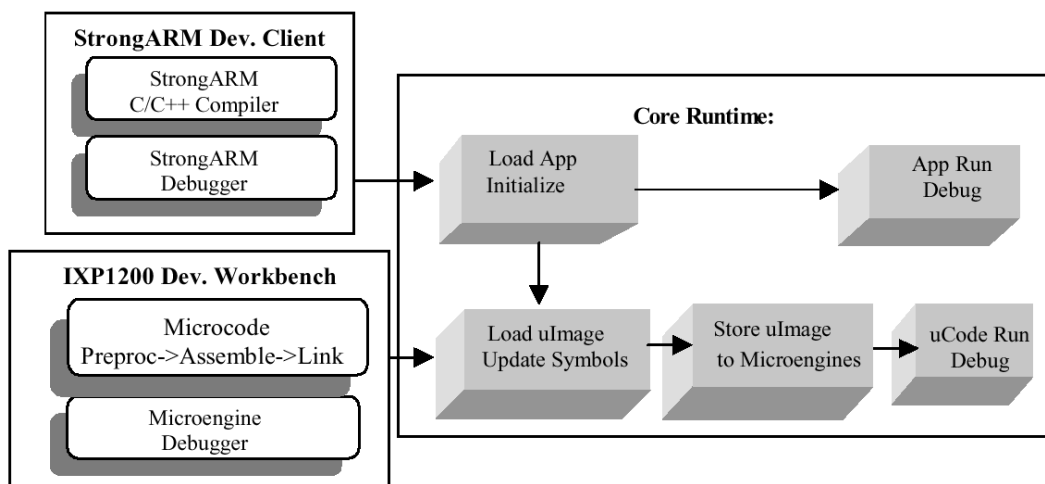


Figure 8. IXP1200 development environment block diagram (11, p. 1-3)

The most important features of Developer Workbench include:

- source level debugging
- execution history and statistics
- IX bus device and network traffic simulation.

(12, p. 2-1.)

Developer Workbench comprises a preprocessor, assembler and linker to link the microcode with the StrongARM code. Also it includes a comprehensive debugger. The preprocessor processes directives, loops, conditional code and expressions and performs macro inline expansion. In addition, it does a high-level syntax check. The microassembler performs low-level syntax checking and branch optimization and assigns symbolic variables to registers while checking that the microcode is legal (the amount of usable registers is limited). (11, p. 1-3.)

Developer Workbench can run the code in a simulation mode or in the hardware if the evaluation kit is present (see chapter 5 about the evaluation kit). In the simulation mode, a foreign model can be used. By a foreign model it is meant software modules that can emulate different external hardware devices. A foreign model is an extension to the capabilities of Transactor, which is a cycle and data accurate software model of IXP1200. With a foreign model and the Transactor it is possible to simulate an IXP1200 system very accurately.

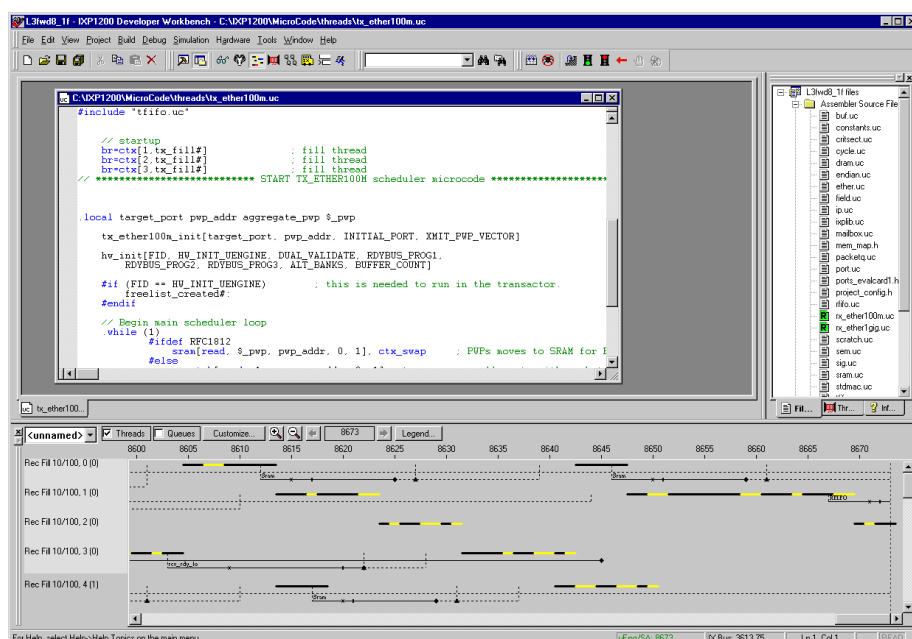


Figure 9. IXP1200 Developer Workbench with the thread history window

Figure 9 shows an example of a simulation session in Developer Workbench. There are typically several windows open while simulating. In the example, the thread history window shows a timeline presentation of the execution of all the threads running in the microengines. It indicates different stages of execution in different colors. From that kind of information the programmer can easily see if any piece of code is using plenty of time in waiting for other units to complete, for example.

Other information windows contain watches for memory locations and registers, microcode browsers, subroutines or macro definitions and packet buffer status controllers. In conclusion, IXP1200 Developer Workbench offer very sophisticated tools for microcode development.

#### 4.4.2 Design Projects

IXP1200 Developer Workbench includes several example microcode design projects. The projects have different port configurations and functioning models. There are projects that work only with 100 Mbps Ethernet ports, projects with only gigabit-port support, and projects that support both types. By function, they are divided in layer 2 bridging and layer 3 routing example designs. Some of the example projects are for simulation only, but most of them can also be used with the evaluation system.

Workbench projects are based on macro building blocks. Small pieces of code are combined into reusable macros. The projects then use the macros to form the programs.

Most of the files in the projects are microcode source files (extension `.uc`). There are also other files, like header include files (`.h`) and stream simulator files (`.strm`).

The code is divided for the microengines depending on the type of ports that are used. A generic method is to divide the threads as receive and transmit threads. Among the transmit threads there are usually some scheduler threads that share the outgoing packets between the actual transmit threads.

There is a major difference between gigabit and 10/100 Mbps ports. The processing power of a thread is adequate for 10/100 Mbps ports so that one thread can receive and process the whole incoming data packet with wire speed, and continue with next packet.

With gigabit ports, however, data is coming in with such a speed that the same thread cannot receive and process all the packets. Therefore the load is distributed to several threads. The threads read packets and they are assigned identifiers so that the packets can be rearranged when sending them further.

## 5 IXP12EB ETHERNET EVALUATION KIT

### 5.1 Overview of the Evaluation Kit

Intel has developed an evaluation kit for IXP1200, the IXP12EB Ethernet Evaluation Kit (Figure 10). It is a complete hardware implementation of a network device product developed around the IXP1200 processor. It demonstrates the use of multiple 10/100 Mbps and gigabit Ethernet ports. Also it provides an example of the electrical, mechanical and component layout used in a typical IXP1200 design (13, p. 1-1).

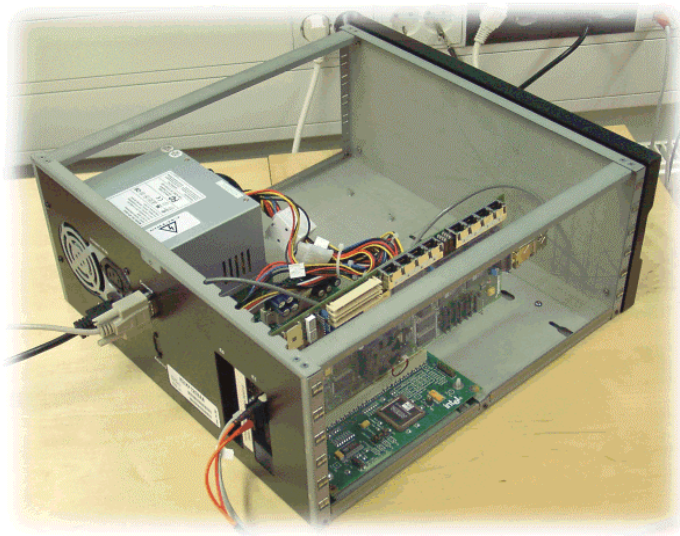


Figure 10. The IXP12EB evaluation system. The processor card (see Figure 12) is shown in the middle.

The evaluation kit is interfaced with the host system by an Ethernet card and a serial cable (Figure 11). The host system is typically a networked PC.

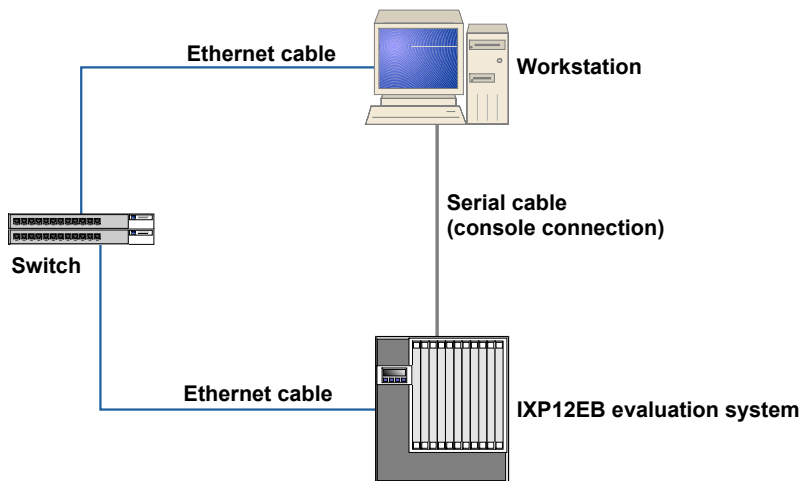
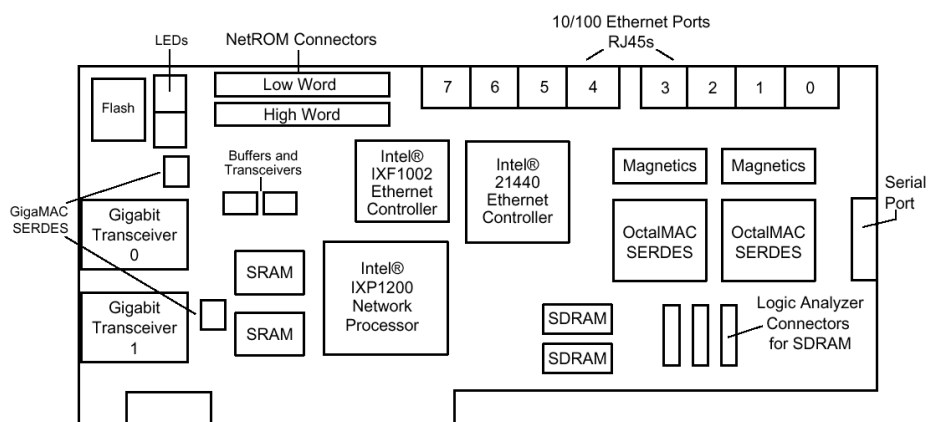


Figure 11. IXP12EB evaluation system connected to the host computer

Version 1.2 of Intel IXA SDK was supplied with the evaluation kit. Version 1.3 can be ordered from the IXP1200 homepage (3). The SDK includes all the documentation on the CD-ROM in Adobe Acrobat PDF format.

## 5.2 Hardware

The evaluation kit consists of the following parts: a system box (including the power supply), a passive PCI backplane, a 10/100 Mbps Ethernet card on the PCI bus (for interfacing with the host computer) and the processor card (Figure 12).



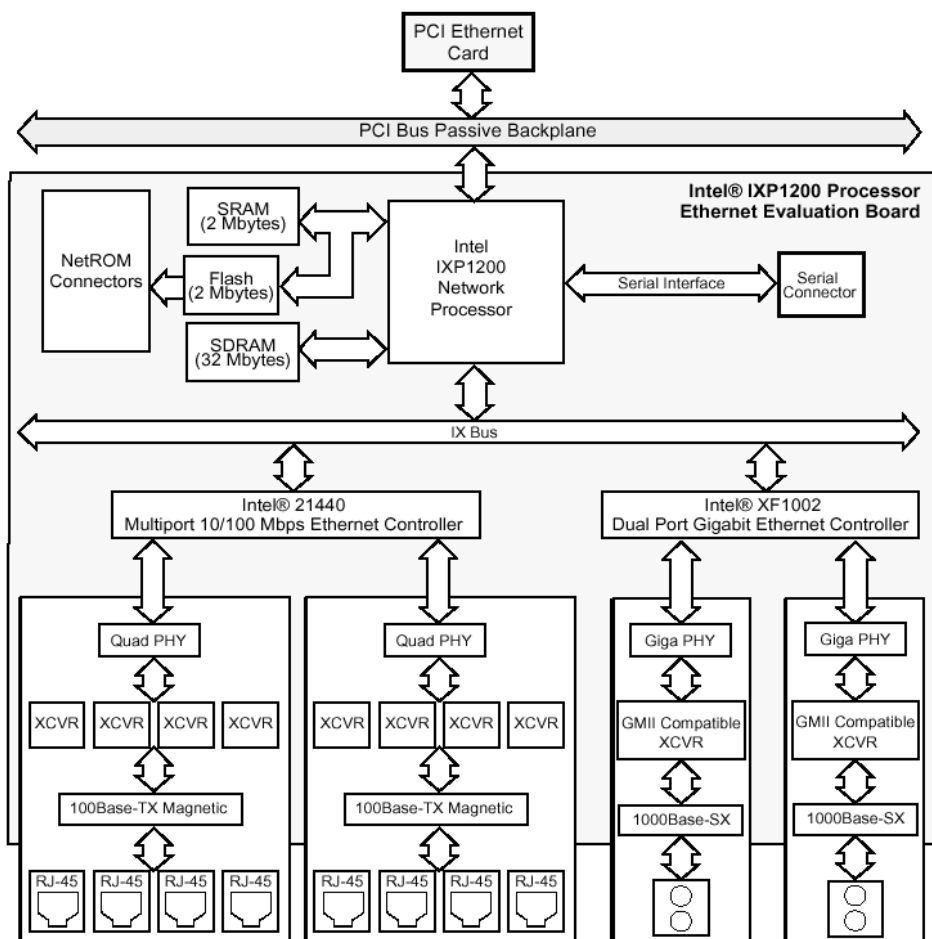
A8056-01

Figure 12. The processor card layout of the IXP12EB Ethernet Evaluation Kit (13, p. 1-5)

The processor card includes all the functionality of the evaluation kit. Only an external Ethernet card is used for interfacing with the host computer. The processor card plugs into the PCI backplane like the Ethernet card. The most important components on the card are:

- a multi-port 10/100 Mbps Ethernet controller (and 8 RJ-45 connectors)
- a dual-port gigabit Ethernet controller (and two fiber connectors)
- 32 megabytes of SDRAM
- 2 megabytes of SRAM
- 2 megabytes of Flash memory
- a serial port connector (for console interface to the host computer)
- the IXP1200 processor.

A functional block diagram of the Ethernet evaluation system is shown in Figure 13.



A8057-01

Figure 13. A functional block diagram of the Ethernet evaluation system (13, p. 1-4)

### 5.3 Setting Up the System

Windows NT 4.0 Workstation (with Service Pack 6) is the supported environment for running the development software on the host computer. The other host requirements are Intel Pentium II 266 MHz processor (or faster), 64 megabytes of RAM, 2 gigabytes of hard disk, a network interface card and a CD-ROM drive (13, p. 2-1). For connecting the IXP12EB console cable, the host computer should have one serial port available.

Windows' HyperTerminal can be used as a terminal emulator to access the console port.

The IXP12EB processor card should be set up according to chapter 2.2 ("Hardware Setup") in the IXP1200 Ethernet Evaluation System User's Guide (13). In general, the default settings can be accepted.

There are some requirements that must be fulfilled to load the VxWorks operating system. The Wind River Tornado II development environment and the IXP1200 Developer Workbench must be installed to the host computer.

The Tornado installation includes an FTP (File Transfer Protocol) server that is used to load the operating system image to the evaluation system. Any other FTP server software can also be used. The FTP server has to be configured to have a user account so that the evaluation system boot software can download the necessary images from the server. In this case, a user called "ixp" was created with a password "ixpuser". The user's home directory was set to `c:\ixp1200\boardsupport\bin\vb\` which is the directory that contains the images when IXP1200 Developer Workbench is installed (the drive specification may differ).

### 5.4 Starting Up

Connect the console port of the evaluation system to an available serial port in the host computer, and start the HyperTerminal for that serial port. Correct communication settings are 9600-8-N-1 (9600 bits per second, 8 data bits, no parity, 1 stop bit) with the flow control set to None. Tornado installation includes correct shortcuts for COM1 and COM2 in *Start - Programs - Tornado2* menu.

### 5.4.1 BootMgr and the Basic VxWorks Loader

By default, the system loads BootMgr at the startup. It is a boot manager that can be used to start the basic VxWorks operating system from the flash memory. When the evaluation system is powered up, the auto-boot sequence is shown in the console window. Pressing the space bar will stop the countdown. In the BootMgr prompt, VxWorks can be started with "b 3" command. The following information appears with a second auto-boot countdown:

```
BSP IXP Version: 1.3.141, built on Jun  5 2001, 19:18:39
CPU ID: 6901C121
CPU Speed: 200 MHz
Early serial debug initialized
muxLoad failed!
0x
                                VxWorks System Boot

Copyright 1984-1998  Wind River Systems, Inc.

CPU: Intel ixp1200eb - ARM IXP1200
Version: 5.4
BSP version: 1.2/2
Creation date: Jun  5 2001, 19:18:39

Press any key to stop auto-boot...
```

After pressing a key the [VxWorks Boot] prompt is displayed. Enter command "c" to change the settings:

```
[VxWorks Boot]: c

'. ' = clear field;  '-' = go to previous field;  ^D = quit

boot device           : eeE0
processor number      : 0
host name             : giga5
file name            : vxWorks
inet on ethernet (e) : 192.168.0.109
inet on backplane (b):
host inet (h)        : 192.168.0.105
gateway inet (g)     :
user (u)             : ixp
ftp password (pw) (blank = use rsh): ixpuser
flags (f)            : 0x0
target name (tn)     : giga9
startup script (s)   :
other (o)            :

[VxWorks Boot]:
```

The IP addresses and host names are assigned by the network administrator of the laboratory network. They can be private network addresses, if applicable. Enter the FTP settings as set up in the FTP server configuration. The full description of the settings can be checked from the Ethernet Evaluation System User's Manual (Chapter 2.5.2).

#### 5.4.2 Loading the Image

The operating system image can now be loaded with "@" command in the [VxWorks Boot] prompt:

```
[VxWorks Boot]: @

boot device          : eeE
unit number         : 0
processor number    : 0
host name           : giga5
file name           : VxWorks
inet on ethernet (e) : 192.168.0.109
host inet (h)       : 192.168.0.105
user (u)            : ixpuser
ftp password (pw)   : ixpuser
flags (f)           : 0x0
target name (tn)    : giga9

Attached TCP/IP interface to eeE0.
Attaching network interface lo0... done.
Loading... 712620 + 14184 + 67968
Starting at 0x1000...

BSP IXP Version: 1.3.141, built on Jun  5 2001, 19:18:11
CPU ID: 6901C121
CPU Speed: 200 MHz
Early serial debug initialized
muxDevLoad failed for device entry 0!
Attached TCP/IP interface to eeE unit 0
Attaching network interface lo0... done.
0x7ffe64 (tRootTask): dc0 - Failed to read ethernet address
NFS client support not included.

VxWorks

Copyright 1984-1998 Wind River Systems, Inc.

CPU: Intel ixp1200eb - ARM IXP1200
VxWorks: 5.4
BSP version: 1.2/2
Creation date: Jun  5 2001
WDB: Ready.
```

The last line indicates that the operating system is now loaded and that the target server can be started from the Tornado system.

### 5.4.3 Configuring and Launching the Target Server

The target server is a part of the Wind River Tornado II development environment. It is used to interface the host system with the target system. The target system is the embedded hardware for which the software is being developed.

The next task is to configure the target server. Tornado environment is started (*Start - Programs - Tornado2 - Tornado*), and the configuration is done by selecting *Target Server - Configure...* from the *Tools* menu. From *Target Server Properties* drop-down list select *Back End* according to Figure 14.

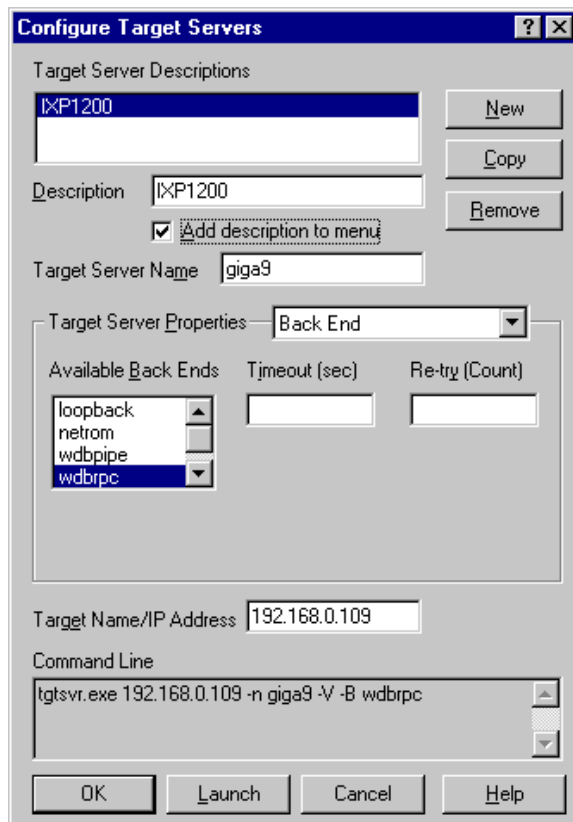


Figure 14. Target server configuration dialog

Important settings here are *Target Server Name* and *IP Address*.

Select *Core File and Symbols* from *Target Server Properties* list. Check *File Path From Target (If Available)*.

Select *Memory Cache Size* from *Target Server Properties* list. Check *Specify* and input the value *8192*.

Select *Console and Redirection* from *Target Server Properties* list. Check *Redirect Target IO*.

Now the command line in the window should be like "`tgtsvr.exe 192.168.0.109 -n giga9 -V -m 8388608 -B wdbRPC -redirectIO`". Click on *Launch*, and the target server is started.

#### 5.4.4 Opening the Tornado Shell

The Tornado Shell is a command line interface to the target VxWorks operating system. It is started by selecting the proper target and clicking on Launch Shell (Figure 15 and Figure 16).

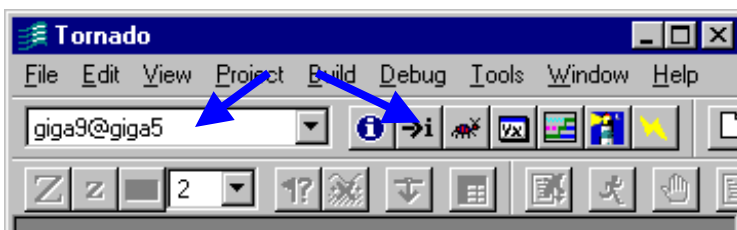


Figure 15. Opening the Tornado Shell

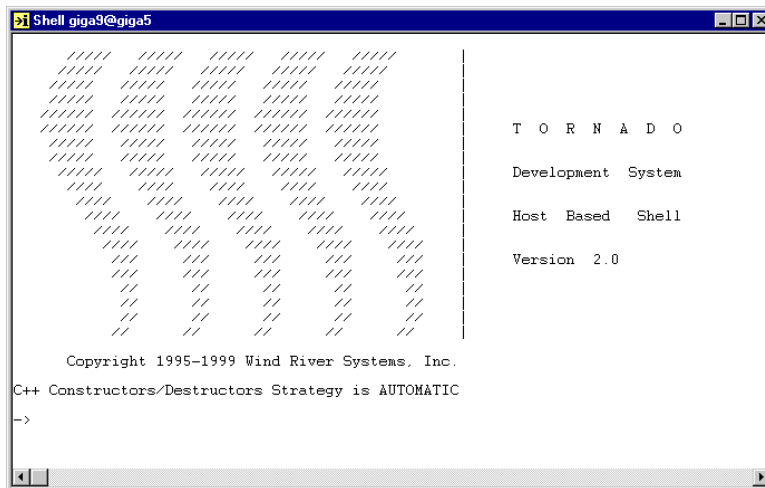


Figure 16. The Tornado Shell window

#### 5.4.5 Loading the VxWorks NetApp Image

After launching the target server and Tornado Shell, the VxWorks NetApp image must be loaded and initialized. The NetApp image provides the ability to test the Developer Workbench connection and to run projects (13, p. 2-18).

Click on the shell window and enter command

"ld < c:\ixp1200\vxworks\_lib\vxworks\_gig.o". This command loads the image from the specified directory. The image is then initialized with command "NetApp\_GigInit" (note case sensitivity).

#### 5.5 Loading an Example Project

An example project is loaded with *File - Open Project...* command in IXP1200 Developer Workbench. Ready-made projects are found under

c:\ixp1200\microcode\workbench\_projects directory. In this example, L3fwd8\_1f project is selected from corresponding directory. Project files have names ending .dwp.

L3fwd8\_1f is a project based on macro building blocks, supporting Layer 3 routing with eight 10/100 Mbps ports and one gigabit port.

When running the project on hardware, it has to be selected from Debug menu.

Hardware - Options... menu item is checked so that in Connections page, Connect via Ethernet is selected and the node name is specified as the same as configured in Chapter 5.4.3 (here: giga9). In Ports page, all ports are unchecked.

## 5.6 Setting up the Routing Table

While the project is now ready to run, it has no information about the routes it should use.

They are set in Tornado Shell window in StrongARM code. One command to set the routes is `route_add()`. It is used to set up a configuration, where two computers are connected to 10/100 Mbps ports of the evaluation system, in ports 0 and 7:

```
route_add("192.168.3.1", "255.255.255.255", "", 0, 0x0000863c,
0xb408)
route_add("192.168.3.2", "255.255.255.255", "", 7, 0x0004762f,
0xb113)
```

The last parameters are the MAC addresses of the network cards.

A quick-guide of the routing table manipulation commands is shown with `rt_help()` command.

## 5.7 Running the Project

The debugging session is started in IXP1200 Developer Workbench with Debug - Start Debugging command (or by pressing F12). The project is run by pressing F5 (Debug - Run Control - Go). If everything went fine, the forwarding process is now running.

Traditional ping testing with the computers connected to the evaluation system is not working yet, however. This is due to example project's inability to forward Ethernet

broadcasts, which in turn results in ARP (Address Resolution Protocol) request failures. This means that the computers cannot see each other.

The problem is solved by using "arp -s" command at both computers to specify each other's MAC addresses. The syntax is "arp -s ipaddress macaddress". The command is a standard TCP/IP tool in most operating systems.

After setting up the ARP tables, all traffic is forwarded between the two hosts.

## 5.8 Benchmarking the Evaluation System

The efficiency of the IXB12EB evaluation system was measured with a simple NetPIPE (A Network Protocol Independent Performance Evaluator) tool (14). NetPIPE uses TCP (Transmission Control Protocol) packets of different size to measure the speed of the network. The purpose of this test was to get a picture of the packet forwarding speed of the evaluation system compared to a regular 10/100 Mbps switch. A cross-over cable was used as well to connect the computers. Figure 17 shows the results of the benchmark.

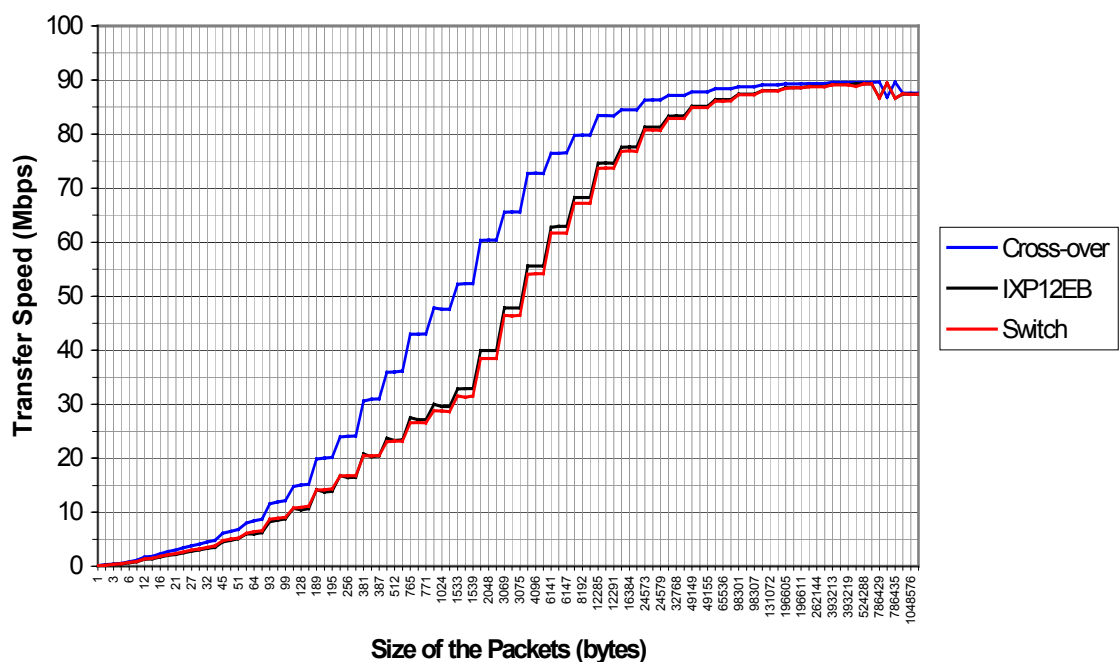


Figure 17. Transfer speeds with packets of different size (NetPIPE test)

From the figure it can be seen that the speed increases when the packet size increases. The cross-over connection is fastest all the time until the 100 Mbps Ethernet becomes congested. When comparing the evaluation system and the switch, the evaluation system is slightly faster but the difference is almost unnoticeable.

This simple test was used to realize the functionality of the example project. The same test was also run with a total of four hosts, connected in pairs, with the same kind of results. It has to be noticed, however, that this was a very simple method to compare the devices with different capabilities, as the evaluation system was not supposed to do anything else than plain packet forwarding.

## 6 RELATED WORK

In this chapter, other network or protocol processor related studies are reviewed.

### 6.1 Balan and Hengartner

In the fall 2000, Rajesh Krishna Balan and Urs Hengartner from Carnegie Mellon University did a study titled "Performance Analysis of the Intel IXP1200 Network Processor" (15). They concentrated on the microengines and their ability to run code efficiently. The metrics they used were the average number of instructions in flight, instructions per cycle and branch mispredictions. They also analyzed memory bus bandwidth and memory access latencies.

Balan and Hengartner used the IXP1200 Developer Workbench to run their evaluation because the hardware evaluation system doesn't provide any detailed performance information. They ran different projects in the simulator with differently sized data packets.

They noticed, among other facts, that the microengines were functioning most efficiently with small data packets. This is because large data packets need more time to transfer the data between different units and memory locations. The overhead, caused by the header processing and forwarding information lookup, remains constant.

Separate Workbench projects are designed in different ways. The scheduling and transferring tasks are divided differently. This leads to variations in average-instructions-in-flight values. Also separate microengines had different instructions-per-cycle values because of their different dependencies on external units.

According to their report, some example projects could be optimized by using the branch prediction capability of the microassembler. In measurements, over 50 percent of branches in a certain block of code were systematically assuming the improbable branch destination. Those cases would profit from branch prediction.

## 6.2 Karlin, Peterson and Spalink

In November 2000, Scott Karlin, Larry Peterson and Tammo Spalink published a paper "Evaluating Network Processors in IP Forwarding" (16) in Princeton University. Their study dealt with the challenges involved in network processor programming. They had Intel IXP1200 as their example case.

The first challenge is dividing the processing work for multiple contexts of the processor. This is important due to hardware latencies that occur when memory and other external resource references are made. With multiple contexts and hardware multithreading, it can be ensured that there is always productive processing going on in the processor. The task that referenced the memory can give its processing time to other threads.

The authors emulated infinitely fast network ports and noticed that the forwarding-rate bottleneck is the SDRAM. They managed to calculate a 26-percent capacity improvement if there were faster memory.

The other major challenge with network processors is dynamic programming. This means adding new functionality to the processor so that the tight timing constraints are taken into account. There is limited space for adding processing to incoming packets. The space depends on the type of processing that is required. Both CPU cycles and memory access cycles are valuable at gigabit speeds. The authors demonstrated in their study that it is possible to support eight 100 Mbps ports with enough headroom to access up to 224 bytes of state information for each minimum-sized IP packet (64 bytes).

## 6.3 Virtanen

Seppo A. Virtanen, a Ph.D. student from Turku Centre for Computer Science (TUUS), has written several publications on protocol processors. In TACO (Tools for Application-specific hardware/software CO-design) research project (17), led by professor Johan Lilius, he explores new system design technologies like System-on-Chip (SoC) and

Application-Specific Instruction-set Processor (ASIP). Those designs aim at fulfilling the increased performance requirements and at achieving shorter development times for protocol processors. Intel IXP1200 is not used in the project at the implementation level, but they concentrate on the concepts.

In SoC design the project team tries to reduce the number of microchips needed to build a certain system. An SoC is designed from reusable intellectual property (IP) blocks. The hardware architecture and the instruction set of ASIP are designed to perform certain specific tasks as efficiently as possible.

They have found that in control-oriented protocol processing there are some common operations that, in practice, occur similarly in all communications protocols. By finding the operations, these can be made into IP blocks, regardless of the protocol. The IP blocks can then be used in designing an ASIP specification and design environment. The TACO research project objective is to design and implement such an environment that is optimized for the specification and synthesis of a communications protocol processor.

(17.)

## 7 DIGITAL VIDEO BROADCASTING

There has been a digital technology evolution for television broadcasting for over thirty years. Until these days, the only analogue part used in television broadcasting has been the final transmission to the consumer.

### 7.1 The Reasons for Digital Broadcasting

While the input and output signals of television systems at the camera and at the receiver are analogue by their nature, the digital choice has still established a firm foothold. Analogue signals are easily distorted and the receiver cannot distinguish the original signal from the disturbances. Digital signals, however, can be regenerated so that they represent exactly the original signal at any stage.

Digital bit streams can be interleaved within a single channel. This interleaving process allows ancillary signals to exist along with the video and audio streams. Furthermore, sophisticated compression techniques can be used to minimize the amount of data to be transmitted. (18, p. 5.)

### 7.2 Standards

There are two main standards for digital television broadcasting: DVB (Digital Video Broadcasting) and ATSC (Advanced Television Systems Committee). ATSC is an American-originated digital television broadcasting standard. The DVB standards are published by the European Telecommunications Standards Institute (ETSI).

The DVB Project (19) was originally a European project but produces standards and guides accepted in many areas of the world. These standards and guides encompass all transmission media, including satellite (DVB-S), cable (DVB-C) and terrestrial broadcasting (DVB-T) (20, p. 39). This study concentrates on the DVB implementation.

### 7.3 DVB System Overview

International Telecommunication Union's (ITU) digital terrestrial television broadcasting model is shown in Figure 18.

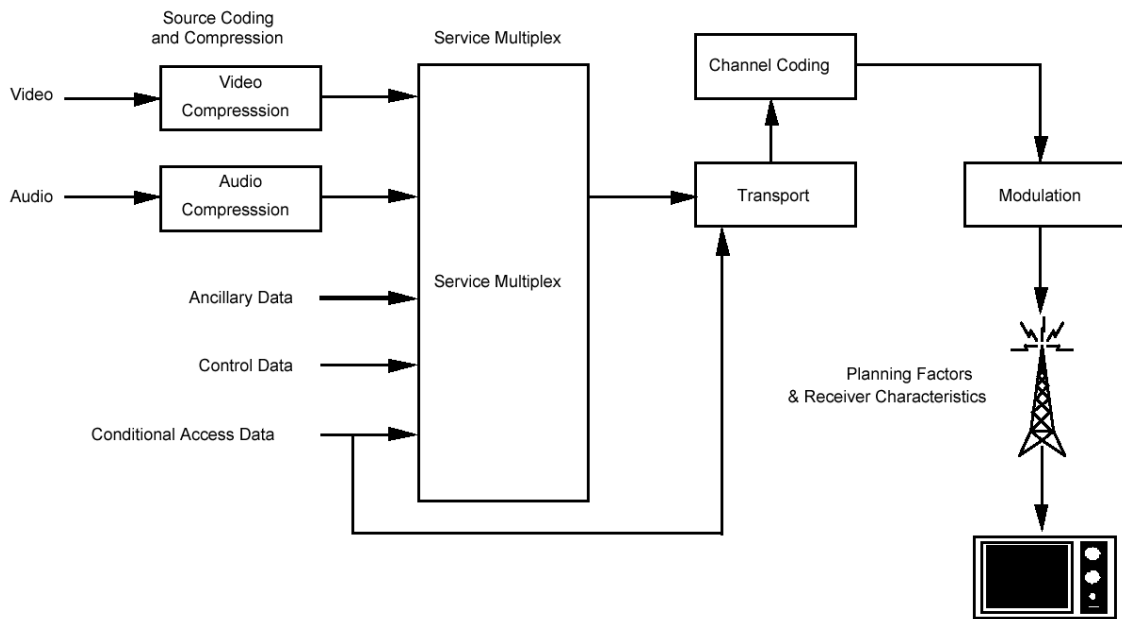


Figure 18. ITU digital terrestrial television broadcasting model (18, p. 8)

The DVB systems use MPEG-2 (21) compression for audio and video. The compressed audio and video streams are combined together along with the ancillary and control data. This process is called multiplexing. In pay-per-view cases also conditional access data is multiplexed. The multiplexer output is MPEG-2 transport stream. Channel coding is applied to transport stream before modulation and broadcasting.

The MPEG internals are not in the scope of this study for other parts than the MPEG-2 transport stream.

## 7.4 MPEG-2 Streams

MPEG-2 defines two types of streams for the transmission: Program Stream (PS) and Transport Stream (TS). The process in the previous chapter describes a Single Program Transport Stream (SPTS). Transport stream is usually a Multi-Program Transport Stream (MPTS) when there are several programs multiplexed together (Figure 19).

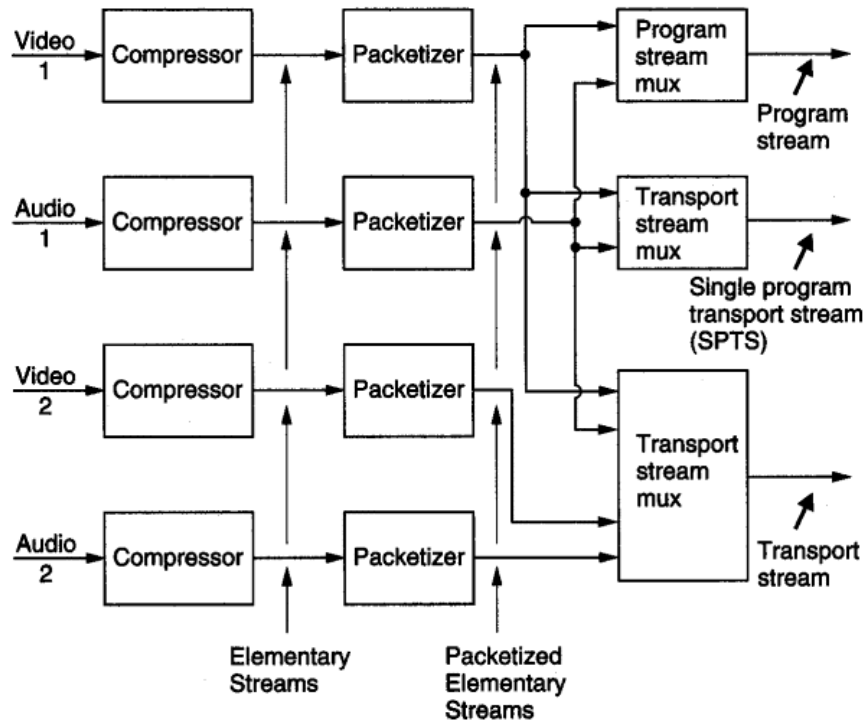


Figure 19. The bitstream types of MPEG-2 (22, p. 27)

For both types of streams the audio and video elementary streams are packetized to Packetized Elementary Streams (PES) prior to multiplexing.

### 7.4.1 Program Stream

One type of MPEG-2 stream is program stream. Program streams are used in single program transmissions over relatively error-free channels and for example DVD (Digital Versatile Disc) distribution.

Program stream consists of variable-sized packets of audio and video from the same source. There is no error-correction or detection data included. For those reasons, it is not a suitable format for DVB transmission.

#### 7.4.2 Transport Stream

MPEG-2 transport stream is suitable for transmissions when there is potential corruption or loss of packets. Transport stream can also be used to transfer several programs at a time by multiplexing them into same stream.

Transport stream consists of 188-byte fixed-length packets. Each audio and video PES packet is broken into transport stream packets and then multiplexed into a transport stream (Figure 20). This multiplex can contain one or more streams carrying audio, video and data. Besides programs, transport stream also contains different kinds of control and information packets about the programs being transferred.

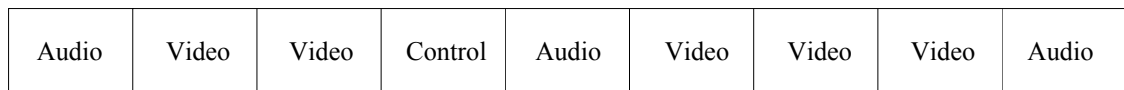


Figure 20. Transport stream: fixed-length packets with control information

##### 7.4.2.1 Transport Stream Header

One transport stream packet has a four-byte header and 184-byte payload. Header structure is shown in Figure 21.

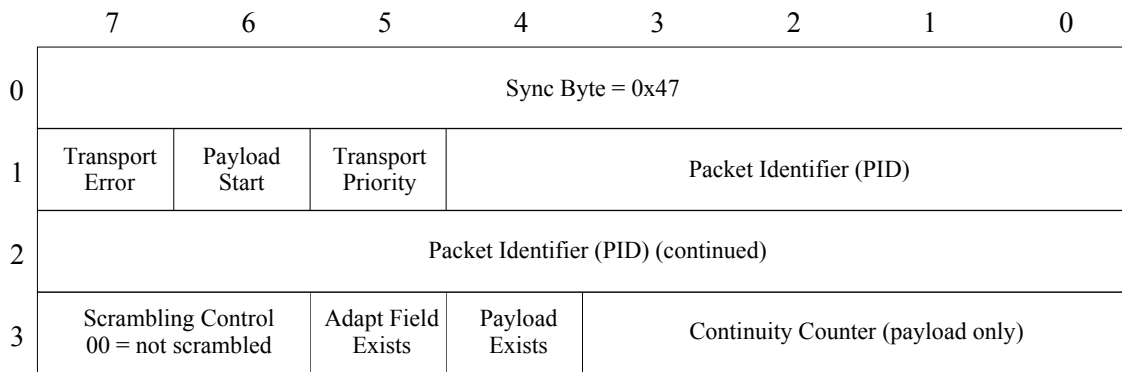


Figure 21. Transport stream header structure (4 bytes)

Because of the relatively small packet size, receiver resynchronization is easy after transmission losses and errors. The synchronization is done by checking for the Sync Byte that appears in the first byte of each packet.

The size of the packet, 188 bytes, is chosen so that it can be broken into four 48-byte ATM cell payloads, where each cell has one byte of AAL-1 (ATM Adaptation Layer) information. An ATM cell is actually 53 bytes with 5 bytes of header information.

As one transport stream may contain many different elementary streams, each elementary stream is identified by a unique 13-bit Packet Identifier (PID) that is included in the header. The receiving demultiplexer seeking a particular elementary stream can simply check the PID of every packet and accept those that match while rejecting the rest. (22, p. 333.)

While MPEG-2 program stream is a variable bit rate stream, transport stream has a constant bit rate. This can be accomplished by adding null packets into the stream when necessary. Null packets have a PID of 0x1FFF (8191 in decimal). If the transport stream is remultiplexed, the remultiplexer can remove null packets if it needs to insert new streams into the transport stream.

The actual header can be extended with an adaptation field. This is signaled with the Adaptation Field Exists bit in the transport stream header. If an adaptation field exists, the payload is respectively smaller so that the transport stream packet length is always 188 bytes. The adaptation field can contain information for synchronizing the stream and splicing information to mark positions where the program can be cut. The synchronizing information is called program clock reference.

#### 7.4.2.2 Program Clock Reference

A transport stream is a constant-bit-rate multiplex of several television programs. These may have originated from widely different locations, which causes the programs to have different synchronizations. The decoder running from a transport stream has to lock to the encoder. The program clock reference (PCR) provides this mechanism. (22, p. 333.)

The video encoder has a 27 MHz clock and counter. The counter is sampled to get individual PCR values that are inserted periodically in the transport stream adaptation field.

The demultiplexer in the receiving end takes the interesting packets based on the PID and extracts the PCR values. Those values are compared to the arrival time of the packets. The difference is then used to drive a 27 MHz voltage controlled oscillator. In practice, the transport stream packets will suffer from transmission jitter. Therefore, a loop filter is used and the oscillator averages a large number of time values to lock to the encoder and reject the jitter. As this may take some time when changing the program, the loop filter can modify its time constants to speed up this process during the lock-up. (22, p. 334.)

#### 7.4.2.3 Error Detection

MPEG-2 transport stream does not handle error checking by itself. It is used as input to the Transport Layer in ISO Open System Interconnection (OSI) seven-layer network reference model (23). Transport stream requires the underlying layer to indicate possible transfer errors in the Transport Error field in the header (24). Packet losses can be detected by a discontinuity of the Continuity Counter. For each PID, the counter is normally incremented with each packet.

#### 7.4.2.4 Program-Specific Information

Usually a transport stream carries several programs in the same stream. Therefore there has to be information sent with the stream for the receiver to be able to distinguish different programs and data related to them. This information is called Program-Specific Information (PSI). PSI consists of several tables that are sent with their own PIDs in the transport stream.

Each program has its own Program Map Table (PMT). The program map table has all the information to combine single audio and video elementary streams into a program. PMT lists all the streams associated with that program. That can include audio and video streams as well as subtitles for different languages.

The program map table, in turn, is found based on the information contained in the Program Association Table (PAT). The program association table has one entry for every program in the transport stream multiplex.

The program association table is always carried by packets that have a PID of 0. The PIDs of the program map tables are told in the PAT.

The first program in the PAT refers to Network Information Table (NIT). The NIT may contain information about other transport streams that are available to the same decoder. That can include different frequencies or dish positions (with DVB-S broadcasting).

The Conditional Access Table (CAT) contains information regarding any encrypted programs with limited access. CAT packets have a fixed PID of 1.

### 7.5 Broadcasting a Transport Stream

A single 6-8 MHz terrestrial television transmission channel can handle a digital stream of approximately 20 Mbps. Since each television program takes about 5 megabits per second, one MPEG-2 transport stream multiplex can contain from three to five television channels, depending on the quality and type of the channels.

Transport stream multiplex can also contain data that is not directly associated with any television program in the same multiplex. This property can be used to provide Internet connections to television spectators. This method needs other means for sending data back from the receiving end (see Figure 22). Dial-up connections can be used for that purpose. However, because of the broadcast nature of this method, the bandwidth for sending data to a particular receiver is very low.

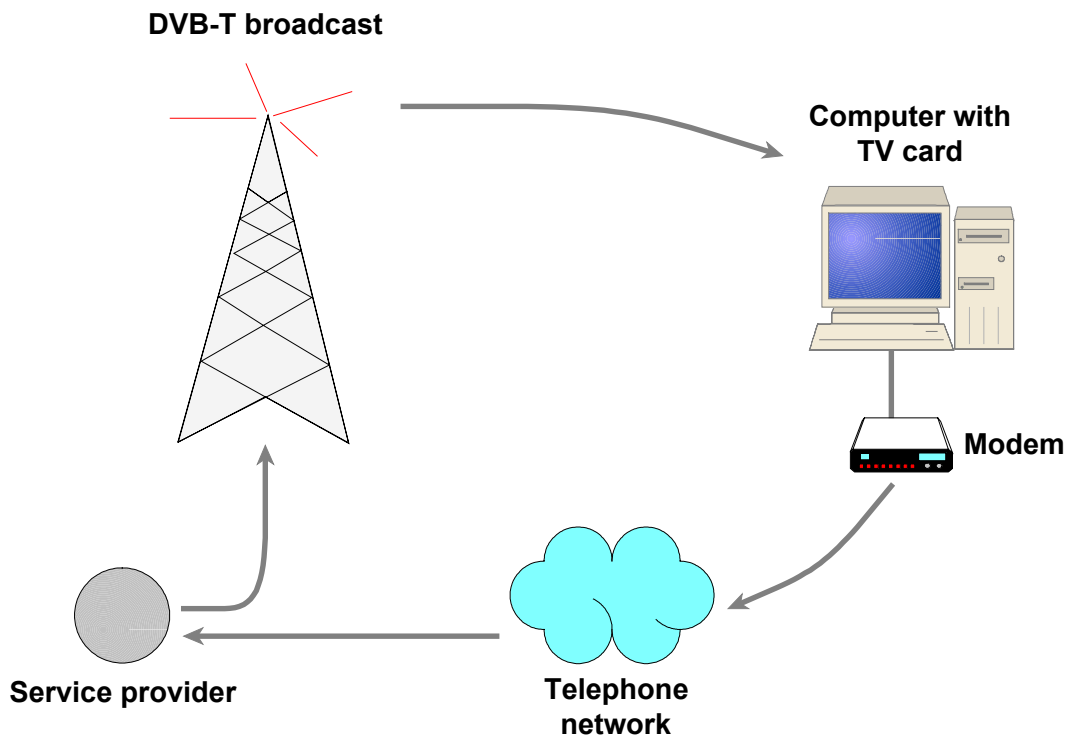


Figure 22. Receiving data from the DVB-T broadcast with a modem uplink

## 8 DVB AND NETWORK PROCESSORS

One of the most demanding types of network data is digitized video. For transmitting several simultaneous real-time video streams, very high bandwidth is required for the network. High bandwidth means huge amount of data transmitted every second. Packets have to be forwarded rapidly between ports of the network devices to avoid congestion and the decisions of the destinations have to be done fast.

In DVB the transport stream is the data that has to be transmitted in the network. As told in Chapter 7.4.2, one major property of the transport stream is its constant bit rate. The included timing information, Program Clock Reference (PCR), is used to restore any jitter caused by intermediate network devices. This task requires intensive timing capabilities from the device that parses the transport stream.

Typical network processor uses include handling high-bandwidth data streams so that the packets are forwarded between the ports based on some decision. This leads to an idea to use a network processor to demultiplex a transport stream.

One scenario would be a gigabit-class backbone network that carries hundreds of single-program transport streams over Internet Protocol (IP). This network would replace the conventional cable television network. The streams would include all the television programs that the spectators could have in their homes. The same network could be used for other IP traffic also.

Network processors could be used in the distribution network where the streams would be directed to smaller-capacity networks that consist of a few houses. The network processor would have some addressing table based on, for example, multicasting, which enables several network hosts to receive the same streams.

The network processors would have the responsibility to deliver the streams based on the subscription information and also to remove any jitter caused by the transmission. The jitter-removal needs high processing power with timing capabilities; just forwarding the packets is not enough.

## 9 RESULTS

The purpose of this study was to introduce the Intel IXP1200 network processor and its properties. Also its suitability for Digital Video Broadcasting transport streams was explored.

Digital Video Broadcasting has very strict requirements for the data streams. To guarantee a smooth audio and video decoding process, the network must be able to forward the transport streams so that there will be minimal jitter in the receiving end.

The fast path-slow path division sets some limitations for using IXP1200 network processor for DVB stream handling. The fast path is able to deal with packets arriving at gigabit speed, but taking care of the timing information (Program Clock Reference) of the transport stream would require using the StrongARM core of IXP1200 because the microengines do not have any timing capabilities.

The StrongARM core, however, is not designed to handle the packets alone. It should be used only in exceptional cases like when routing and forwarding information is changed or when some other administrative control is needed.

For the mentioned reasons, IXP1200 was not seen to be suitable for controlling real-time video streams when strict timing is necessary.

Intel IXP1200 can be used for other applications. Quality of Service (QoS) is one major trend in network computing. QoS includes defining the rules based on which the transiting data packets are classified. The rules can use protocols, ports, and source and destination addresses to determine the priority. Traffic is then forwarded according to those priorities. Microengines can be used to sort the packets to separate queues, and the transmit scheduler can transmit packets from different priority queues.

Other target application for IXP1200 could be firewalling. Certain rules are made according to which the microengines either filter or pass through the packets.

The most suitable applications are dynamic by their nature. That way the programmability of the network processor can be brought out. Plain packet forwarding can be done with hardware or generic processors. The main point in network processors is their adaptability to different purposes with different software.

## 10 CONCLUSIONS

Intel IXP1200 was found to be an exciting choice for developing flexible network devices. The network processor concept is relatively new in the world of ASICs and generic embedded processors. The benefits of programmable network processors are worth inspecting. The development of the software is faster and more cost-effective than designing new ASICs.

Intel IXP1200 is the best-known network processor at the moment. While other network processor manufacturers keep a low profile with their products, Intel is openly spreading information about IXP1200 series.

### 10.1 Problems with the Evaluation System

There were some problems with the evaluation system during the project. Intel supplied a whole bunch of documentation in PDF form on CD-ROM, but it consisted of very many individual documents, and they even gave different information about the same subjects. In principle, the setup of the evaluation system was documented, but the meaning of the separate phases of configuration was not clarified. As Chapter 5 told about the configuration, the setup of the system consists of many operations.

The evaluation system was shipped with several example projects. They were not documented, however. One problem was also the fact that some documents referred to example projects that were not anymore shipped with the evaluation system.

The main problem was with the gigabit fiber ports. They were not functioning the way they were documented. The 10/100 Mbps twisted pair Ethernet ports worked like charm. The gigabit ports had to be used in a very special way compared to 10/100 Mbps ports, which caused difficulties when solving the problems.

## 10.2 Product Support

During the research project, the support for Intel network processors was found to be very poor in Finland. The vendor from whom the IXP12EB evaluation system was purchased was not able to help with any problems that arose while trying to get the system running. The vendor asked to contact Intel Finland, but they were not willing to help an end-user directly and asked to contact the vendor instead.

After some trying, the system was managed to get running. However, the gigabit ports were still causing problems and help was needed for them.

According to one rumor, Intel had IXP1200 specialists in the United Kingdom and they were contacted. They pointed out a person in Sweden that knew about IXP1200 and could even speak Finnish. However, this person was not very helpful even though he promised to return to the matter. It was another dead-end.

In the mean time, Intel Finland's people were spoken to. It seemed like all the Intel employees, or at least they who knew the Intel IXA architecture, were working for Nokia. This fact may have affected the ability to get any help for a project that was not directly connected to Nokia.

Later in the summer a message was received from Intel UK. They admitted that there was a fault in gigabit port handling routines. When asking for more information, they cancelled their error report. This caused a long exchange of email messages and led to sending the processor card to tests to Swindon, UK. The card was tested, but no defects were found.

After all, the support from Intel for using their evaluation system was not very impressive. Even though it was asked many times, they did not supply any information about different software components and their compatibility. This resulted in unnecessary uncertainty when problems arose.

## 11 FUTURE WORK

With IXP12EB Ethernet Evaluation Kit it is possible to implement various network devices. The ready-made example projects allow modifying and enhancing the code to develop other applications. These include Quality of Service applications, firewalls, Voice over IP gateways and IPSec (IP Security) implementations, for example.

Intel IXA SDK version 2.0 includes a microengine C compiler for IXP1200. The compiler makes it possible to use a high-level programming language to write code for the microengines. The compiler also includes libraries of routines to be used. The C compiler allows faster code-development process with a language familiar for almost every programmer. Exploring the possibilities of the new compiler can be advantageous over writing plain microcode.

Extensive simulations and measurements were not used in this research project. When modifying existing applications, it would be useful to carefully analyze the microcode execution. IXP1200 Developer Workbench offers extensive tools for analyzing the programs.

## REFERENCES

- 1 "Network Processor Programming", Embedded Systems Programming, <http://www.embedded.com/story/OEG20010730S0053>, 15 October 2001
- 2 Hämäläinen, Pertti, "Verkkosuorittimet vahvassa myötäisessä", Tietokone 10 (2001), p. 93 - 94
- 3 "Intel IXP1200 Network Processor Family", Intel Corporation, <http://developer.intel.com/design/network/products/npfamily/ixp1200.htm>, 10 September 2001
- 4 "Intel Internet Exchange Architecture", Intel Corporation, <http://developer.intel.com/design/ixa/>, 10 September 2001
- 5 "IXP1200 Datasheet", part# 278298-008, Intel Corporation, May 2001
- 6 "IXP1200 Hardware Reference Manual", part# 278303-007, Intel Corporation, June 2001
- 7 "IXP1200 Microcode Programmer's Reference Manual", part# 278304-008, Intel Corporation, June 2001
- 8 The homepage of Wind River Systems Inc., <http://www.windriver.com/>, 28 October 2001
- 9 "IXP1200 Linux How To", Intel Corporation, [http://www.netwinder.org/~urnaik/ixp1200\\_howto.html](http://www.netwinder.org/~urnaik/ixp1200_howto.html), 28 October 2001
- 10 "Tornado User's Guide, 2.0 (Windows Version)", Edition 1, 9 Apr 1999, part# DOC-12612-8D-01, Wind River Systems Inc.
- 11 "IXP1200 Microcode Software Reference Manual", part# 278306-007, Intel Corporation, June 2001
- 12 "IXP1200 Development Tools User's Guide", part# 278302-007, Intel Corporation, June 2001
- 13 "IXP1200 Ethernet Evaluation System User's Manual", part# A14987-007, Intel Corporation, June 2001
- 14 "NetPIPE - A Network Protocol Independent Performance Evaluator", <http://www.scl.ameslab.gov/netpipe/>, 27 November 2001
- 15 Balan, Rajesh Krishna - Hengartner, Urs, "Performance Analysis of the Intel IXP1200 Network Processor", <http://www.cs.cmu.edu/~uhengart/15-740/>, 15 May 2001

- 16 Karlin, Scott - Peterson, Larry - Spalink, Tammo, "Evaluating Network Processors in IP Forwarding", <http://www.cs.princeton.edu/nsg/papers/ixp.html>, 15 May 2001
- 17 Virtanen, Seppo A., "TACO Research Project", <http://users.utu.fi/seaavi/res1.html>, 24 November 2001
- 18 "A Guide to Digital Terrestrial Television Broadcasting in the VHF/UHF Bands", International Telecommunication Union (ITU), Document 11-3/3-E, 15 January 1996
- 19 The homepage of DVB Project, <http://www.dvb.org/>, 3 November 2001
- 20 " A Guide to MPEG Fundamentals and Protocol Analysis (Including DVB and ATSC)", Tektronix, Inc.,  
[http://www.tek.com/Measurement/App\\_Notes/mpegfund/25W\\_11418\\_3.pdf](http://www.tek.com/Measurement/App_Notes/mpegfund/25W_11418_3.pdf), 3 November 2001
- 21 The homepage of Moving Picture Experts Group (MPEG),  
<http://mpeg.telecomitalia.com/>, 3 November 2001
- 22 Watkinson, John, "The MPEG Handbook - MPEG-1, MPEG-2, MPEG-4", Focal Press, Great Britain, 2001
- 23 "Introduction to Internet", Cisco Systems Inc.,  
[http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito\\_doc/introint.htm](http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/introint.htm), 1 August 2001
- 24 Fairhurst, Godred, "MPEG-2 Transmission",  
[http://www.erg.abdn.ac.uk/public\\_html/research/future-net/digital-video/mpeg2-trans.html](http://www.erg.abdn.ac.uk/public_html/research/future-net/digital-video/mpeg2-trans.html), 3 November 2001