Engineering of Embedded Linux ATM for MPC8260 and Derivatives

APOSTOLOS MELIONES
Ellemedia Technologies
223 Syngrou Ave., GR-17121, Athens
meliones@ellemedia.com

Abstract: - Network processors provide enormous potential for differentiated high-end integrated networking and communications systems. The PPC PowerQUICC architecture has made this processor family a basic element for networking and communications systems. Its impressive market share has made this processor family the undisputed leader in the communications processor market. On the other hand, Linux is particularly popular in network embedded systems because of its proven networking capabilities and well suiting for networking applications and services that require high reliability and high availability, and the flexibility it offers to specifically tailor it for a special hardware configuration. Furthermore, ATM is very popular in WAN backbone and access networks, especially because of the large investments made by Telcos and ISPs and the inherent support of scalable bandwidths and guaranteed QoS, which facilitates new classes of multimedia services. This paper discusses for the first time the engineering of Linux ATM on the MPC8260 communications processor and derivative CPUs and presents embedded HTTP and SNMP system management interfaces for ATM configuration and monitoring. This work is important to the embedded networking community for the development of ATM access devices for high-end communication and networking systems.

Keywords: - Linux ATM, ATM device driver, MPC8260, MPC860, ATM access device, integrated access device

1 Introduction

Today, the use of Linux in embedded systems is no laughing matter. Linux has become a preferred operating system for embedded systems, mainly because of its open source, maturity and robustness, the multitude of open source and free software projects around it, the community developing it, and the flexibility it offers to specifically tailor it for a special hardware configuration or for support of a certain type of application [1]. Industry and government agencies are increasingly adopting Linux. It is particularly popular in network embedded systems because of its proven networking capabilities and well suiting for networking applications and services that require high reliability and high availability, and the flexibility it offers to specifically tailor it for customized hardware configurations. ATM dominates WAN backbone and access network technology, especially because of large investments made by Telcos and ISPs and the inherent support of scalable bandwidths and guaranteed QoS in terms of throughput and jitter, which facilitates new classes of multimedia services. However, the real world is dominated by connectionless IP networks. Running IP over ATM is mainly an encapsulation issue, defined in RFC1483/1577. A comprehensive overview of ATM is presented in [2].

According to a report recently released by Gartner Dataquest [3], the PowerQUICC family's market share reached 82.7 percent in 2002, making the processor family the undisputed leader in the communications processor market. To date, 150M units have been shipped. MPC860 and MPC8260 are Motorola's PowerQUICC and PowerQUICC-II architecture respectively single-chip integrated microprocessor and peripheral combination that can be used in a broad range of controller applications, particularly in communications and networking products. MPC8260 is the next generation MPC860, delivering higher levels of system performance, peripheral integration and programmability to the communications market. The integration of a PowerPC core, a RISC-based communications processor module (CPM) and a circuit board's worth of system interface and control functions in a dual-processor architecture provides enormous potential in developing differentiated high-end networking and communications systems, such as SOHO routers, integrated access devices, DSLAMs, central office switches, wireless infrastructure base stations, enterprise and VPN routers, media gateways for packet telephony etc., while significantly reducing time-to-market development stages. The CPM offloads peripheral tasks from the embedded processing core and supports multiple high bit rate communications protocols. The degree of system integration in the PowerQUICC-I/-II architecture could significantly reduce a system's component count,

shorten customers time-to-market and slash component cost.
Linux ATM for the MPC8260 family of network processors is based on a porting of the SourceForge Linux ATM driver for the MPC86x ESAR [4]. Common supported ATM features are UBR and CBR traffic types, AAL0 and AAL5 cell formats, as well as the UTOPIA interface. Besides offering a significant performance increase over MPC86x ESAR, the MPC8260 ATM driver in addition supports VBR real-time and non real-time services. The relevant work sketches the reference system and driver architecture, details items of the segmentation and reassembly mechanism which are additions specific for MPC8260, details UTOPIA mode programming, and resolves important issues related to the Linux kernel, such as the packet forwarding performance and support of new devices in earlier kernels. It also describes embedded HTTP and SNMP management interfaces for ATM configuration and monitoring. We released a pre-alpha release of the device driver including the UTOPIA interface and UBR/CBR services to the Linux community to appease the ever-increasing demand for the release of a Linux ATM driver for the MPC8260 [5]. The importance of this development to the embedded networking community is evident by several thousands visits to the released driver web page by the time of writing.
The Linux ATM driver for MPC8260 conforms to the *Linux ATM device driver interface* [6], which defines the kernel inbuilt interface between the ATM protocol stack and device drivers, and is combined with the *ATM on Linux distribution* [7] compiled for a PPC target. This package contains a kernel patch for ATM protocol stacks, management and traffic tools, and supports raw ATM connections (PVCs and SVCs), classical IP over ATM compliant with RFC1577, LANE, MPOA, etc. For PPPoATM support compliant with RFC2364 a user space PPP daemon is needed in addition to the above [8], which has to support the pppoatm plugin [9].

2 MPC8260 Linux ATM Architecture

Fig.1 depicts the MPC8260 Linux ATM reference architecture. The reference system is powered by MPC8264, and includes a UTOPIA ATM interface, an ATM T1/E1 WAN interface through a transmission convergence sub-layer and a multi-port fast Ethernet LAN, connected to CPM peripheral ports FCC1, FCC2 and FCC3 respectively. The implementation of the serial ATM interface and the T1/E1 application are incremental tasks that require the existence of the ATM segmentation and reassembly and UTOPIA subsystems and is not examined in this paper. With minor additions, the same architecture can also support xDSL applications and inverse multiplexing for ATM over a multi-port T1/E1 WAN interface. The engineering of Linux serial ATM on the MPC8260 communications processor for TDM access applications is analytically discussed in [10].
The *Linux ATM device interface* is basically a set of common data structures and conventions for ATM-related system services under Linux. The *Linux ATM user space* denotes whatever uses the ATM

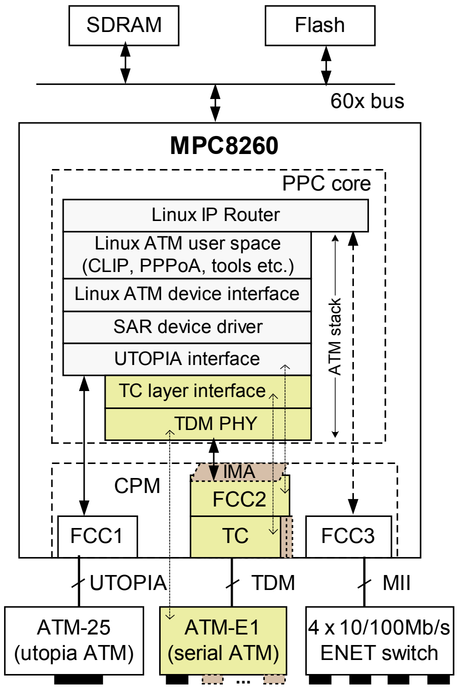
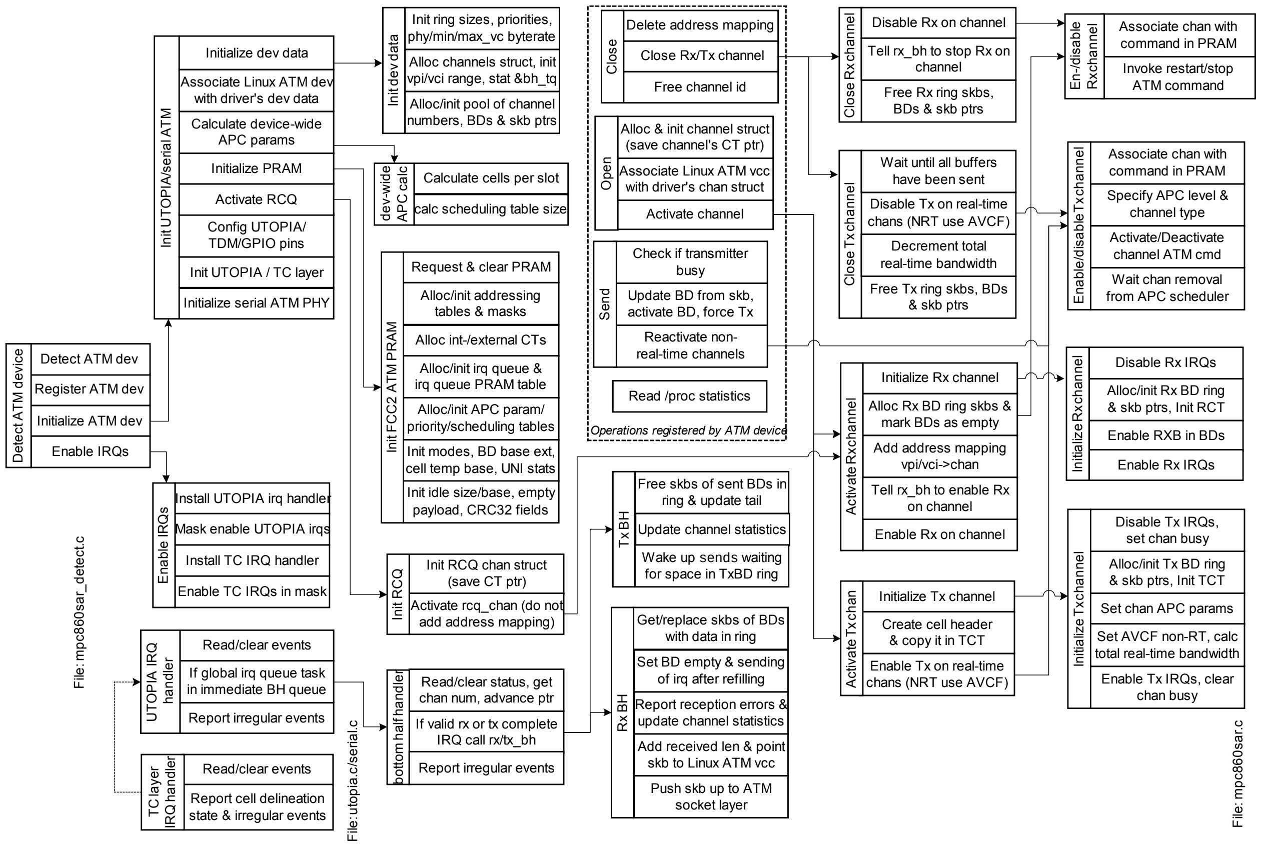


Fig.1: Reference system architecture.

device in a given context, such as a raw transport or IP over ATM. The *SAR device driver* is responsible for controlling the segmentation and reassembly and traffic shaping processes performed directly in hardware, for resource allocation and for coordination with the protocol, the PHY driver and the hardware. *PHY* controls physical layer operation and gathers statistics. The reason for splitting the SAR and the PHY component is because several different PHY chips can be used with the same SAR. The kernel is capable of routing IP traffic between the supported network interfaces through the IP router subsystem, which is a process running completely in CPU mode. Fig.2 sketches the architecture of the MPC8260 Linux ATM driver, which is implemented through patching the MPC86xESAR driver with the mechanisms described in this paper.



Detect ATM device
Detect ATM dev
Register ATM dev
Initialize ATM dev
Enable IRQs
Init UTOPIA/serial ATM
Initialize dev data
Associate Linux ATM dev with driver's dev data
Calculate device-wide APC params
Initialize PRAM
Activate RCQ
Config UTOPIA/ TDM/GPIO pins
Init UTOPIA / TC layer
Initialize serial ATM PHY
Enable IRQs
Install UTOPIA irq handler
Mask enable UTOPIA irqs
Install TC IRQ handler
Enable TC IRQs in mask
File: mpc860sar_detect.c
UTOPIA IRQ handler
Read/clear events
If global irq queue task in immediate BH queue
Report irregular events
TC layer IRQ handler
Read/clear events
Report cell delineation state & irregular events
File: utopia.c/serial.c
Init dev data
Init ring sizes, priorities, phy/min/max_vc byterate
Alloc channels struct, init vpi/vci range, stat &bh_tq
Alloc/init pool of channel numbers, BDs & skb ptrs
dev-wide APC calc
Calculate cells per slot
calc scheduling table size
Init FCC2 ATM PRAM
Request & clear PRAM
Alloc/init addressing tables & masks
Alloc int-/external CTs
Alloc/init irq queue & irq queue PRAM table
Alloc/init APC param/ priority/scheduling tables
Init modes, BD base ext, cell temp base, UNI stats
Init idle size/base, empty payload, CRC32 fields
Init RCQ
Init RCQ chan struct (save CT ptr)
Activate rcq_chan (do not add address mapping)
bottom half handler
Read/clear status, get chan num, advance ptr
If valid rx or tx complete IRQ call rx/tx_bh
Report irregular events
Close
Delete address mapping
Close Rx/Tx channel
Free channel id
Open
Alloc & init channel struct (save channel's CT ptr)
Associate Linux ATM vcc with driver's chan struct
Activate channel
Send
Check if transmitter busy
Update BD from skb, activate BD, force Tx
Reactivate non-real-time channels
Read /proc statistics
Operations registered by ATM device
TxBH
Free skbs of sent BDs in ring & update tail
Update channel statistics
Wake up sends waiting for space in TxBD ring
RxBH
Get/replace skbs of BDs with data in ring
Set BD empty & sending of irq after refilling
Report reception errors & update channel statistics
Add received len & point skb to Linux ATM vcc
Push skb up to ATM socket layer
Close Rxchannel
Disable Rx on channel
Tell rx_bh to stop Rx on channel
Free Rx ring skbs, BDs & skb ptrs
Close Txchannel
Wait until all buffers have been sent
Disable Tx on real-time chans (NRT use AVCF)
Decrement total real-time bandwidth
Free Tx ring skbs, BDs & skb ptrs
Activate Rxchannel
Initialize Rx channel
Alloc Rx BD ring skbs & mark BDs as empty
Add address mapping vpi/vci->chan
Tell rx_bh to enable Rx on channel
Enable Rx on channel
Activate Txchan
Initialize Tx channel
Create cell header & copy it in TCT
Enable Tx on real-time chans (NRT use AVCF)
En-/disable Rxchannel
Associate chan with command in PRAM
Invoke restart/stop ATM command
Enable/disableTxchannel
Associate chan with command in PRAM
Specify APC level & channel type
Activate/Deactivate channel ATM cmd
Wait chan removal from APC scheduler
Initialize Rxchannel
Disable Rx IRQs
Alloc/init Rx BD ring & skb ptrs, Init RCT
Enable RXB in BDs
Enable Rx IRQs
Initialize Txchannel
Disable Tx IRQs, set chan busy
Alloc/init Tx BD ring & skb ptrs, Init TCT
Set chan APC params
Set AVCF non-RT, calc total real-time bandwidth
Enable Tx IRQs, clear chan busy
File: mpc860sar.c

Fig.2: Linux MPC8260 ATM device driver architecture.

3 ATM SAR Controller

An important aspect of programming the ATM controller is the understanding of the operational procedures for transmission and reception and the various data structures used for configuration and operation. The programming of the ATM controller is based on a porting of the SourceForge Linux ATM driver for the MPC86xESAR [4]. Detailed descriptions refer only to items that are additions specific for MPC8260. Furthermore, we discuss all problems in the course of design and implementation of the driver including their workarounds. Elements of the ATM device driver interface necessary for MPC86x and MPC82xx ATM device driver programming across Linux kernels 2.4 and 2.6 have been presented in [11].

3.1 Address Mapping

The address compression mechanism to look up channel numbers for incoming cells differs from MPC860 in the entry size of the VC tables and the index creation into the VP table. A new element is the Match Status bit. This should indicate *no match* for all VC table entries during initialisation and following the removal of an entry from the VC table. It should indicate a successful match when updating the corresponding VC table entry with the channel number to use for incoming cells with a specific VPI/VCI value. Incorrect handling of the MS bit will result in the receiving of signalling traffic on the raw cell queue before the synchronisation of the user-space ATM signalling demon with its counterpart on the network side and may as well result to the discarding of cells.

3.2 Connection Tables

Since MPC8260 stores TCTs and RCTs in separate arrays the single MPC86x function accessing the coupled connection table mapped to by a channel is split into separate transmit and receive functions. Channel codes of 255 or less indicate internal channels. External channels are assigned numbers 256 and above. Channel 1 is reserved as the raw cell queue and channel 0 is reserved. No additional CT entry is reserved for CPM internal use.

3.3 ATM Parameter RAM

The ATM parameter RAM initialisation follows the corresponding procedure in MPC86x specifically tailored to the MPC8260 mapping. The ATM PRAM is stored in the dual-port RAM at offset 0x8400 or 0x8500 to the DPR base depending on the peripheral used. The ATM controller data structures are packed and any padding will result in improper mapping of data structure elements to PRAM. Improper alignment of parameters or initialisation will result in kernel crash during system boot or improper operation. DPR stores connection tables for fast channels. The amount of DPR memory needed for a single FCC ATM device with 128 internal channels is 9272 bytes. A 128-byte explicit allocation, or 256 bytes in case of dual-FCC driver operation, is fixed at the end of the DPR data area for CPM use. Therefore, we modified the kernel DPR allocator limit. In case a microcode package is downloaded in the DPR to recover from a certain device error or to enhance the CPM functionality the DPR allocator base has to change accordingly, reducing the maximum number of internal channels. Related to the parameter RAM initialisation is the definition of one priority interrupt queue and a unit global interrupt threshold, of priority and scheduling tables for real-time and non-real-time service classes and max iteration allowed in the APC, as well as the selection of 60x bus, VP table in DPR, and checking of unallocated bits during address compression.

3.4 External Memory Allocation

The external memory allocations used with the CPM require physically contiguous, uncached pages, because the CPM accesses memory without the use of the cache. In MPC8260 a number of physically contiguous pages of host memory is allocated for the CPM early in kernel initialisation. We increased the number of CPM host memory pages in the kernel and used the CPM host memory allocator to acquire space in external memory for the ATM specific data structures. Alternatively, the MPC86x function can be used, in which the page table entries must be modified so that the pages are designated as write-through in addition to caching-inhibited.

3.5 Buffer Descriptors and Socket Buffers

The BD allocation mechanism matches the static buffer allocation mode. The limitation that all BDs must lie within a 256KByte region does not exist and thus it is possible to use kmalloc() for this purpose. The fact that this mechanism allows client code to allocate blocks of varying size means we can support different ring sizes per traffic class. Large rings could support fast channels while keeping ring sizes small for slow channels to save memory. Handling of socket buffers (here for storing ATM frames) is unique throughout all the Linux network subsystems.

3.6 Interrupt Mechanism

Besides tailoring the interrupt handling mechanism to the MPC8260-specific interrupt queue and BD structures, the bottom half handler clears the interrupt queue entry in the interrupt queue parameter table each time an overrun occurs in the circular queue. The transmit BH handler ignores CRC error indications for AAL0 traffic since a CRC field is not included in the cell payload. In case the default AAL5 MTU size, which defines the length of AAL5 buffers, is smaller than the default IP MTU value of 9180 for use over ATM AAL5 defined in RFC1626 [12] and a corresponding IP fragmentation is not defined with the atmarp CLIP daemon the receive BH handler will drop large received frames. This will happen because several protocols in wide use throughout the Internet, such as the Network File System, use large frame sizes (e.g. 8KB).

3.7 ATM Tx/Rx and Auto-VC-off/VC-on

MPC86x supports ATM transmit activate/deactivate channel commands by writing to the CPM command register. In MPC8260 the CPM command set includes the ATM TX command. This turns a passive channel into an active one by inserting it into the APC scheduling table. Before issuing the command we set the flag VCON in the channel's TCT and initialise the COMM_INFO fields (channel code, APC priority) in the parameter RAM. The command operates on the used peripheral. The deactivate command is implemented by setting the Stop TX (STPT) flag in the TCT of the specified channel. STPT is not set for auto-deactivated UBR channels. After issuing the command the host has to wait for the TCT VCON flag to clear, indicating that the channel has been removed from the scheduling table. The host can issue another TX command only after the CPM has cleared VCON. The TX command should be issued only after the channel's TCT is completely initialised and the channel has BDs ready to transmit.
MPC8260 does not include any ATM receive command, unlike MPC86xESAR which includes stop/restart receive commands. MPC8260 is stopped from receiving cells by disabling the receive buffer and frame events in the RCT of the specified channel. Enabling those events restarts the receiver to allow receiving cells for the specified channel. Note that the receive CLIP signalling daemon is exiting when stopping the ATM receiver through disabling the FCC receiver and you will have to reinitiate the protocol stack on both sides of the ATM connection. A typical error is to try to mimic the ATM stop/restart receive commands with the FCC receiver full or shortcut disable/enable sequence.

3.8 Connection Table Programming Model

The connection table programming model in MPC8260 resembles to that of MPC86x. Byte ordering is additionally specified as big endian and the GBL pin is asserted to force snooping of bus/memory transactions by the data cache in order to maintain data coherency on the 60x bus. Keeping a running total allocated bandwidth is extended to real-time channels. Note that you will have to replace *min_pcr* with *max_pcr* in *net/atm/proc.c* in order to get non-zero peak cell rate information in */proc/net/atm/pvc* for use by the management interface, because the transmit rate is passed to the driver via the *max_pcr* QoS field of the Linux ATM device interface VC descriptor structure. UBR channels are assigned the maximum transmission rate. Real-time channels will still take priority, as non-real-time channels will be scheduled in the low priority table. The APC unit equation (1) determines the time-slot scheduling rate of a channel, modified to prevent numerical overflow as well as the calculation of incorrect values for PCR and PCRF under integer arithmetic in the C programming language.

$$R = \left\lfloor \frac{\text{line byte rate} \times 256 + \text{VC byte rate} \times \text{cells per slot} - 1}{\text{VC byte rate} \times \text{cells per slot}} \right\rfloor \quad (1)$$

$$PCR = R / 256 \qquad PCRF = R \% 256$$

TCT[AVCF] is set only for UBR channels. A UBR channel is deactivated whenever the CPM detects there is no further data to send from that channel. To continue transmission after the host adds buffers, a new transmit command is needed. An inactive UBR channel is activated by send() (indicated by VCON bit not set). If AVCF is not set, the APC unit does not remove the channel from the scheduling table when there is no more data to send. Cell headers are copied in a channel's TCT entry in big endian format.

3.9 Enabling VBR Traffic

We have extended the device driver to support VBR traffic classes. We maintained the two-priority levels scheme for servicing real-time channels before non-real-time channels. VBR-rt and CBR channels share the high-priority scheduling table, while VBR-nrt channels join the UBR channels in the low priority table. A new function is added for accessing the transmit connection table extension mapped to by a VBR channel. Internal TCTEs are stored in DPR for fast channels. The COMM_INFO fields in the ATM Transmit command should be initialized for the VBR service. Furthermore, we set in the TCT entry for the channel the traffic type and the PCR APC parameters, and calculate PCR of VBR-rt channels in the running total bandwidth allocated for real-time channels. The SCR APC parameters stored in the TCTE entry are determined using (1). The SCR and MBS VBR traffic parameters are passed to the device driver through the modified user-space atmarp CLIP daemon via the modified Linux ATM device interface VC descriptor structure. Equation (2) determines the BT parameter of the APC leaky bucket algorithm.

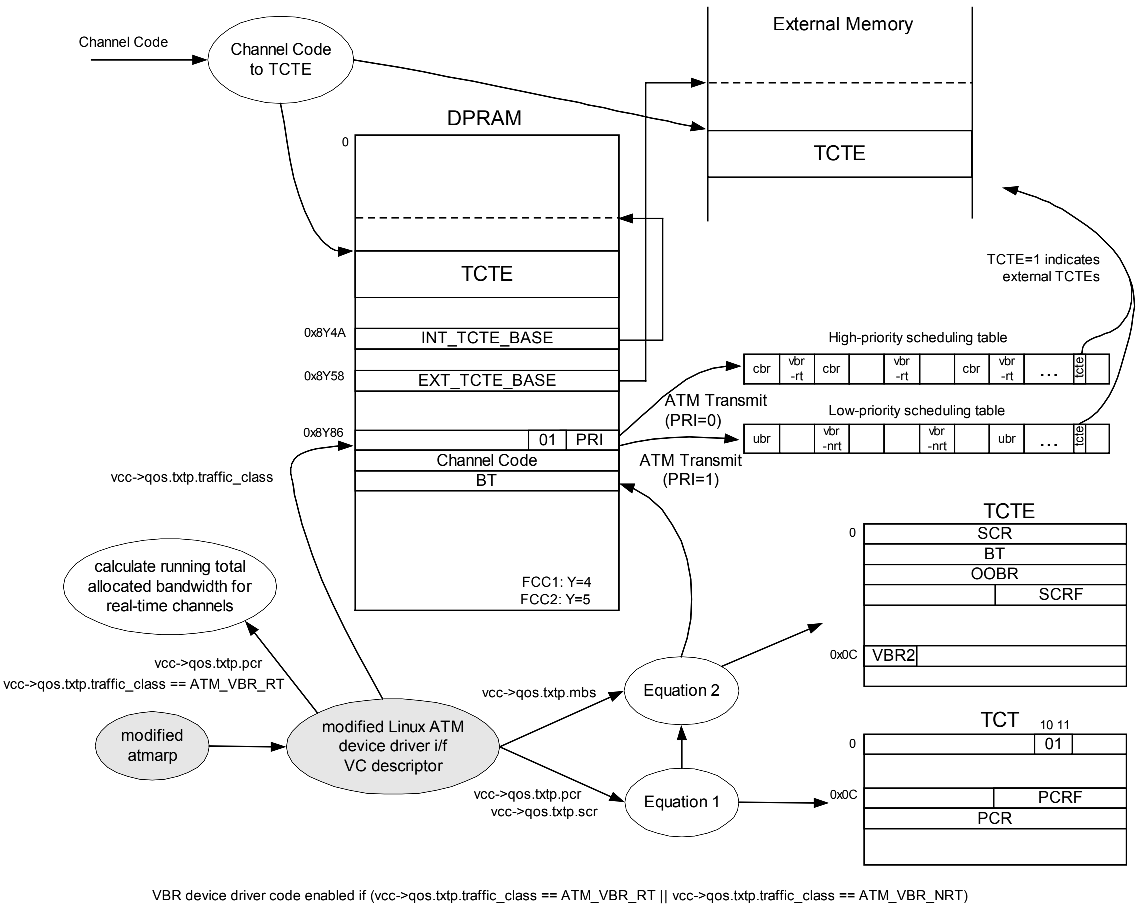
$$BT = (MBS - 2) \times (SCR - PCR) + SCR + \left\lfloor \frac{(MBS - 2) \times (SCRF - PCRF) + SCRF}{256} \right\rfloor$$

In the TCTE entry we also set OOBR and the way scheduling is affected by CLP. External VBR channels should be indicated in their scheduling tables. Fig.3 depicts the schematic VBR code. The shaded processes refer to the modified kernel ATM device interface and user-space CLIP protocol. The former is modified to define the VBR traffic classes and include VBR parameters in the ATM traffic parameters structure. The latter is modified to support VBR QoS parameters when converting the binary encoding of QoS parameters to textual representation and vice versa. This should be reflected on the maximum ATM QoS length operated by the user-space driver. A VBR service is requested through atmarp.

3.10 Maintaining Memory Coherency

Unlike DPR internal channels, external channels store their connection tables in external memory. When an ATM transmission on an external channel is requested the transmit command to the CPM is issued before the TCT initialisation actually completes in external memory. Consequently, transmission will never start or Tx-buffer-not-ready events will be noticed in case the CPM tries to open mismatched TxBDs due to incorrect TBD_BASE in the TCT. In order to maintain a coherent memory for use with the CPM we force writing the modified data cache block out to memory following the TCT initialisation. Flushing the data cache is not necessary following the stop transmit command because the host is waiting for the VCON flag to clear, which follows the completion of TCT[STPS] set in memory. Similarly, we flush the data cache following the RCT initialisation.

A similar error appears infrequently with native ATM traffic while the system is using NFS. It takes a while before the device driver command writing the channel code in the VC table during receive channel initialisation actually completes in memory. If data is received on this channel during this time cells will be discarded due to address look-up failure and counted as misinserted cells. Reception will start abruptly as soon as the channel code is eventu-



Channel Code
Channel Code to TCTE
External Memory
DPRAM
TCTE
0
0x8Y4A
INT_TCTE_BASE
0x8Y58
EXT_TCTE_BASE
0x8Y86
01 PRI
Channel Code
BT
FCC1: Y=4
FCC2: Y=5
TCTE=1 indicates external TCTEs
High-priority scheduling table
cbr vbr-rt cbr vbr-rt cbr vbr-rt ... tcte
ATM Transmit (PRI=0)
Low-priority scheduling table
ubr vbr-nrt vbr-nrt ubr ... tcte
ATM Transmit (PRI=1)
vcc->qos.txtp.traffic_class
calculate running total allocated bandwidth for real-time channels
vcc->qos.txtp.pcr
vcc->qos.txtp.traffic_class == ATM_VBR_RT
modified atmarp
modified Linux ATM device driver i/f VC descriptor
vcc->qos.txtp.mbs
Equation 2
Equation 1
vcc->qos.txtp.pcr
vcc->qos.txtp.scr
TCTE
SCR
BT
OOBR
SCRF
0x0C
VBR2
TCT
10 11
01
PCRF
PCR
VBR device driver code enabled if (vcc->qos.txtp.traffic_class == ATM_VBR_RT || vcc->qos.txtp.traffic_class == ATM_VBR_NRT)

Fig.3: Schematic VBR device driver code.

ally written in memory and a CRC/LNE error will be reported. The problem disappears using a flash-disk or with CLIP traffic, in which case channels are opened at early time during connection setup. To solve the problem we flush the modified data cache block following the VC table entry write command.

4 UTOPIA Mode Programming

MPC8260 offers full 155-Mbps ATM SAR through UTOPIA II interface on FCC1 and FCC2 with support for up to 31 PHYs per interface, which make it ideal for high-density DSLAM line cards. ATM-25 and OC-3 applications are implemented along the same lines. In our system application we used the IDT 77V106 transceiver with default initialization. We further tested the interface on a MPC8260ADS development platform using the PMC-SIERA PM5350 ATM-155 UNI. This section describes the CPM configuration for the UTOPIA mode, which corresponds to the *Init UTOPIA PHY* architectural block in Fig.2. This involves UTOPIA specific FCC1 programming and setting up of I/O ports and clocks. Using two or more ATM ports requires operation in MPHY mode. Pin configuration involves the selection of dedicated UTOPIA interface signals among the peripheral functions multiplexed onto the parallel I/O ports and of general purpose I/O pins, as illustrated in Fig.4, through proper programming of the port registers. Pin configuration corresponds to the *Config UTOPIA pins* architectural block in Fig.2. The interrupt handler for the UTOPIA application corresponds to the FCC1 interrupt vector.

FCC1 is operated in ATM internal rate mode as a UTOPIA master in a single PHY environment. In case the total transmission rate is less than the PHY rate this mode saves CPM performance, because it is the PHY that fills the unused bandwidth with idle cells. Using the CPM multiplexing logic BRG5 is assigned to the FCC1 internal rate generator to clock the transmitter at the PHY rate. BRG5 was programmed to generate transmit cell requests equivalent to the PHY rate every 2238 CPM clocks. BRGCLK is half the CPM clock and is divided by an odd number in the BRG configuration to ensure a 50% duty cycle. Furthermore, BRG6 was assigned as the FCC1 receive and transmit clock and programmed to generate a 33MHz UTOPIA clock. In case of ATM-155, BRG5 should generate a transmit cell request every 360 CPM clocks. Passing of

FCC1 interrupt requests to the core is enabled in the SIU interrupt mask register. A command is issued to the CPM to initialise the transmit and receive parameters in the FCC1 parameter RAM involving the FCC1 (not the ATM) sub-block code. BRG5 and the FCC1 receiver and transmitter are enabled as the last step in the initialization process.

Table 1: UTOPIA mode programming.

Register	Value (hex)	Description
GFMR1	A (4000000A)	ATM mode, disable FCC1 Rx/Tx (CPM43)
FPSMR1	400 (200400)	Single PHY master mode, disable Rx parity (CPM43)
CMXUAR	&=FF0F	FCC1 internal rate clk=BRG5
BRGC5	8BC	Cell rate programming, disable count
FTIRR1 PHY0	80	Internal rate mode
CMXFCR	09000000	Connect FCC1 to NMSI pins, FCC1 Rx/Tx clk=BRG6
BRGC6	10002	33MHz
SIMRL	80000000	Allow IRQs from FCC1
CPCR	12010280	Init Rx/Tx parameters in FCC1 PRAM
BRGC5	|=10000	Enable scheduling of cells
GFMR1	|=30	Enable FCC1 Rx/Tx

A software workaround is provided against the CPM43 device error on HiP3 Rev. A.1 devices. When the system bus clock is 66MHz the FCC1 transmitter will ignore the negation of the TxCLAV signal in single PHY mode. Consequently the UTOPIA slave will be unable to control the flow of cells. The workaround is to configure FCC1 in MPHY master mode using a single PHY.

5 Linux Kernel Considerations

5.1 Routing Performance

The packet forwarding mechanism is inefficient in kernels up to 2.4.7. The IPv4 packet handler and routing mechanisms are adequately efficient, as the journey of a packet inside the kernel networking subsystem in either the local delivery - in which case the layer 4 protocol is processed and the packet is passed to a userspace process - or the “forward to another network” direction follows the determination of the packet’s destination by the routing function. The kernel assigns the whole Linux networking subsystem to a *soft interrupt* mechanism. The CLIP protocol hands the packet over to the networking subsystem. Then a softirq is requested to handle the packet. In order to require quicker attention from the packet forwarding mechanism to avoid latency issues in a high-interrupt-rate device, we force handling of pending softirqs following the handling of hardware interrupts in the generic interrupt handler (*kernel/softirq.c:do_softirq()* is called from within *arch/ppc/kernel/irq.c:do_IRQ()*).

5.2 HiP4 and later derivatives of MPC8260

Kernels up to 2.4.10 do not support HiP4 or later derivatives of MPC8260. This has been fixed as of kernel 2.4.11 in *arch/ppc/kernel/cputable.c*. The most significant bit in the PVR mask for MPC82xx was cleared so as the kernel could recognize the 603e core for HiP4 devices. Kernel 2.4.10 introduced a big reorganization in the PPC architecture, so the described bug fix could not be patched to older kernels. For those we added an assembly patch in *arch/ppc/kernel/misc.S* in the function that reads the PVR register, which clears bit 0 of the value read. This way we deceived the kernel to take HiP4-MPC8260 for a HiP3 device.

6 Management Interface

The reference system features an embedded HTTP server, which can be accessed via a web browser for system configuration and monitoring. Fig.5 depicts the CLIP HTTP management interface. The selected configuration is translated into a user-space script for CLIP setup [7]. The depicted configuration defines a CBR service over PVC 0.0.32 with ILMI assigned address, default UNI signaling, 20Mbps constant bitrate, and specific source and destination IP addresses belonging to the same LIS. The selection of a VBR-rt service translates further into a QoS parameter to atmarp (e.g. qos rt-vbr:pcr=6mbps,scr=2mbps,mbs=1000). A similar configuration page exists for PPPoATM including username, password, bandwidth, VPI/VCI, and CHAP/PAP authentication protocols. Another page monitors ATM traffic statistics and receiver errors per device.

In addition to the ATM command line and HTTP management interfaces, SNMP support is also integrated in the reference system. The proprietary ATM MIB provides for the remote management of the ATM devices. Wherever applicable the construction of the MIB is in compliance with RFC2515 [13]. The MIB contains tables for the ATM interfaces present, for PPPoATM, for CLIP VCs and for their traffic parameters description. The last table checks that all necessary values are inserted before row activation. This table is an exact replicate of the corresponding table in RFC2515, though the parameters interpretation is different.

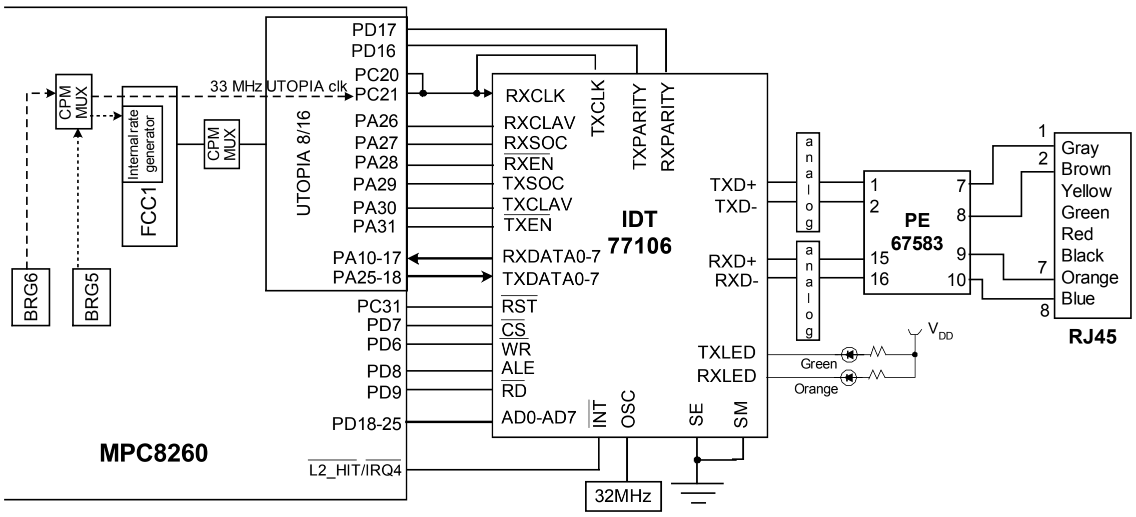
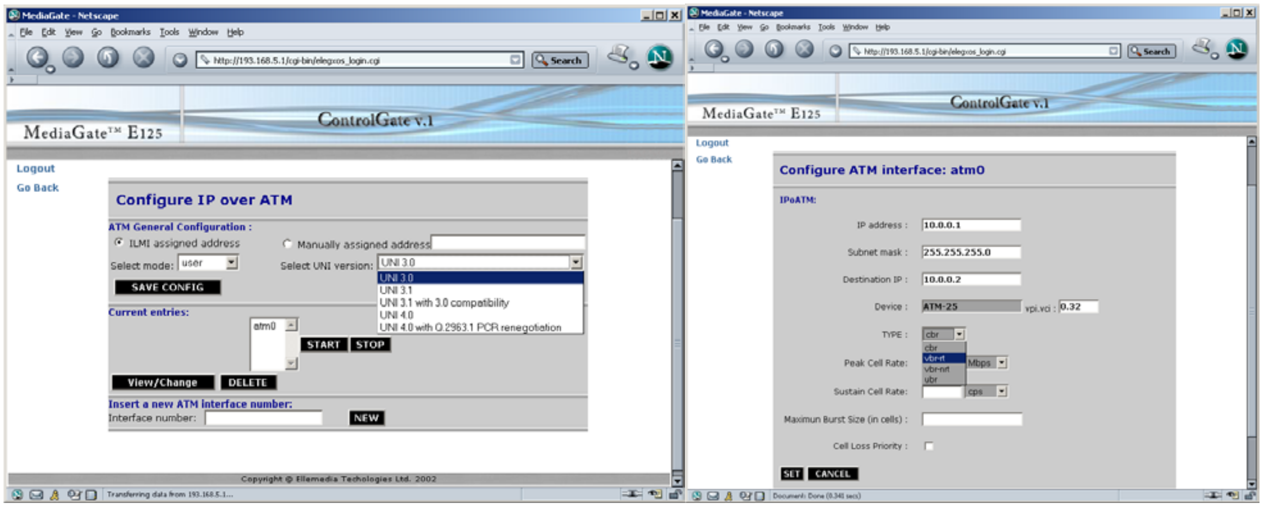


Fig.4: Block/schematic diagram of UTOPIA ATM-25 application.



MediaGate™ E125 ControlGate v.1

Logout
Go Back

Configure IP over ATM

ATM General Configuration :
ILMI assigned address [✓] Manually assigned address
Select mode: user
Select UNI version: UNI 3.0
UNI 3.0
UNI 3.1
UNI 3.1 with 3.0 compatibility
UNI 4.0
UNI 4.0 with Q.2963.1 PCR renegotiation

Current entries:
atm0

Insert a new ATM interface number:
Interface number:

Copyright © Ellemedia Techologies Ltd. 2002

Transferring data from 193.168.5.1...

MediaGate™ E125 ControlGate v.1

Logout
Go Back

Configure ATM interface: atm0

IPoATM:

IP address :	10.0.0.1
Subnet mask :	255.255.255.0
Destination IP :	10.0.0.2
Device :	ATM-25 vpi.vci : 0.32
TYPE :	cbr
Peak Cell Rate:	Mbps
Sustain Cell Rate:	cps
Maximun Burst Size (in cells) :	
Cell Loss Priority :	

Document: Done (0.341 secs)

Fig.5: CLIP HTTP Management Interface.

7 Conclusions

This paper discusses for the first time the engineering of Linux ATM on the PowerQUICC-II communications processor architecture. This work is important to the embedded networking community for the development of ATM access applications for high-end communication and networking systems.

References

[1] K. Yaghmour, *Building Embedded Linux Systems*, O'Reilly, 2003, ISBN 0-596-00222-X.

[2] A. Alles, Internetworking with ATM, White Paper, *Cisco Systems*, May 1995, http://cio.cisco.com/warp/public/614/12.html

[3] Communications Semiconductor and Optical Component Market Share, 2002 © 2003 Gartner Inc.

[4] D. Pegler, D. Nevil, and A. Zeffertt, Linux ATM driver for MPC86xESAR, http://sourceforge.net/projects/mpc860sar

[5] SourceForge Linux ATM driver for MPC8260, http://atm8260.sourceforge.net

[6] W. Almesberger, Linux ATM device driver interface, Technical Report No. DI 96/178, Feb. 1996

[7] W. Almesberger, ATM on Linux - The 3rd Year, *4th Int. Linux Kongress 1997*, March 1997, Würzburg, Germany (http://linux-atm.sourceforge.net).

[8] P. Mackerras, PPP for Linux, release ppp-2.4.2, http://samba.org/ftp/unpacked/ppp

[9] M. Blank, D. Monks, PPPoATM for Linux, http://www.sfgoth.com/~mitch/linux/atm/ pppoatm/

[10] A. Meliones, Engineering of ATM TDM Access Applications for Linux MPC8260 Integrated Access Devices, 9th WSEAS CSCC Int. Conf.

[11] A. Meliones, S. Spanos, and N. Zervos, Linux ATM Interface for PowerPC-based Network Embedded Systems, WSEAS Transactions on Communications, Vol. 3, No. 2, April 2004, ISSN 1109-2742.

[12] R. Atkinson, Default IP MTU for use over ATM AAL5, RFC-1626, 1994.

[13] K. Tesink, Definitions of Managed Objects for ATM Management, RFC-2515, 1999.