Hindawi Publishing Corporation
International Journal of Digital Multimedia Broadcasting
Volume 2009, Article ID 617203, 9 pages
doi:10.1155/2009/617203

Research Article

A Software and Hardware IPTV Architecture for Scalable DVB Distribution

Georg Acher,[1] Detlef Fliegl,[1] and Thomas Fuhrmann[2]

[1] *BayCom GmbH, Guardinistr 47 81375 München, Germany*
[2] *Lehrstuhl für Netzarchitekturen und Netzdienste, Fakultät für Informatik, Technische Universität München, 85747 Garching bei München, Germany*

Correspondence should be addressed to Georg Acher, acher@baycom.de

Received 1 February 2009; Revised 3 September 2009; Accepted 30 September 2009

Recommended by Maurizio Murroni

Many standards and even more proprietary technologies deal with IP-based television (IPTV). But none of them can transparently map popular public broadcast services such as DVB or ATSC to IPTV with acceptable effort. In this paper we explain why we believe that such a mapping using a light weight framework is an important step towards all-IP multimedia. We then present the NetCeiver architecture: it is based on well-known standards such as IPv6, and it allows zero configuration. The use of multicast streaming makes NetCeiver highly scalable. We also describe a low cost FPGA implementation of the proposed NetCeiver architecture, which can concurrently stream services from up to six full transponders.

1. Introduction

In this paper we present NetCeiver, an IPTV architecture that is fully compatible to existing digital television broadcast systems. This allows an easy convergence of digital television and IPTV. During the transition, consumers can thus benefit from the advantages of both technologies.

Throughout this paper we use the term *convergence* to describe a way of delivering content independently of a specific transport medium. In traditional broadcast systems, this would be called simulcast. IPTV allows convergent deliverance of television content over IP. Convergence does not require restoring all physical capabilities of different transport media even though it is desirable. Therefore, the resulting feature set and quality of the encoded media streams may change.

In the following, we first briefly introduce some typical characteristics of the IPTV systems' basic operation schemes. We then examine traditional television broadcast. The comparison of both approaches allows us to investigate the possible means for convergence. Based on our analysis, we derive a novel architecture for transparently transporting broadcast TV services over IP.

1.1. IPTV. IPTV is a well-known term that describes IP-based television in general. Unlike other techniques there is no common specification nor standard that can be used to implement IPTV-based services for every purpose. In the past years several approaches have been made to achieve different solutions. Each solution has its own set of features and optimizations. Some IPTV systems are proprietary; others rely partly or fully on standards such as RTSP [1], SDP [2] or UPnP-AV [3]. This makes it difficult to easily compare IPTV implementations; more abstract characteristics need to be examined.

Roles. TV in the traditional meaning has one provider (server) and multiple receivers (clients). Typical IPTV applications show a similar setup. They only replace the transport medium by an IP-based technology.

However, some IPTV systems that are based on the Web 2.0 concept have given up the distinction between sender and receiver. YouTube, Vimeo, Make.tv, and the like transmit user generated content. Nevertheless they provide their services in a TV-channel-like fashion, either off-line or via live streaming.

Distribution. *Professional* IPTV systems, are typically based on a centralized infrastructure. They use dedicated hardware and network resources for their content distribution. This allows a reliable service, although at high costs.

A peer-to-peer (P2P) approach can eliminate most of the centralized IPTV infrastructure. P2P systems rely on a large number of participants (aka peers) that provide their service within a P2P overlay network [4]. Each peer is a sender and a receiver at the same time. It relays traffic within the P2P network. This results in better load balancing and fault tolerance. Compared to centralized solutions, P2P systems for IPTV are quite new. They do not yet fully exploit their potential [5, 6].

Interactivity. Often, advertisements list interactivity as important IPTV feature. The back channel over IP allows a wide variety of interaction schemes. In reality, these additional services are seldom really interactive and novel (e.g., rating, feedback, and tagging). Usually, they are only more visually attractive versions of traditional broadcast services such as electronic program guides or teletext. Often, they just combine TV with web browsing.

As an exception, typical video on demand (VOD) services provide interactivity in a different way: the customers can select the particular video they want to watch at a given or chosen time. As zapping in IPTV results in bidirectional communication with a server, this could be understood as interactivity as well.

1.2. Digital Television Broadcast Systems. Despite the fact that IPTV systems have been available on the market for quite some time, legacy distribution systems for digital television are still predominant. Most TV stations employ these systems, and as a result, they are more popular than any other IPTV solution—at least in western countries.

These legacy distribution systems broadcast according to well-established standards over terrestrial antenna, satellite transponders, or broadband cable networks. Inexpensive receivers provide consumers easy access to these services. Typically, their quality exceeds that of most current IPTV implementations.

Digital broadcast distribution systems differ by region. The most popular standards are DVB [7] in Europe, ATSC [8] in North America, ISDB [9] in Japan and Brazil, and DMB [10] in South Korea.

For simplicity we refer to all these systems as DVB, because they all use the MPEG2 transport stream format (MPEG2-TS) for their transmissions.

Digital television is widespread. Consumers accept it thoroughly. DVB provides high-quality audio and video transmissions, fast channel switching, and a vast number of additional services such as electronic program guides or conditional access (Pay-TV). Many of those services are fully standardized. As a result, consumers can trust that any commodity receiver is compatible with their preferred service provider. As stated above, this situation is still totally different in current IPTV systems.

If we had the same features for IPTV, we could rely on well-standardized client applications or set-top boxes that we could use with more than one IPTV service only. Unfortunately, this is unlikely to happen in foreseeable future, because IPTV systems have to meet too many different, mostly commercial interests.

In this paper we present a possible solution to this dilemma. Our NetCeiver approach is based on DVB. It makes all features of DVB broadcast systems available from within IP-based local area networks. Before presenting the NetCeiver architecture, in particular its communication protocols and hardware details, we take a look at other IPTV approaches.

1.3. State-of-Art IPTV Implementations. In the early 90's, a very first implementation of IPTV was based on the MBONE network [11]. This network enabled delivery of IPv4 multicast across nonmulticast capable networks. The original purpose of transmitting audio and video was providing a conferencing facility. Therefore a session announcement based on SAP [12] and SDP [2] is being used to show all multicast senders in a browser application. Based on this information it is possible to join multicast groups containing audio and video streams in RTP [13] format. Soon organizations like NASA started to broadcast shuttle launches and other popular events over the MBONE network.

Today, depending on the application, a vast number of IPTV standards and services exist. Still IPv4 multicast is being used in large-scale deployments [14]. This generally requires an infrastructure capable of multicast routing. Advanced IPTV systems seem to use a mixture of proprietary multicast and unicast protocols. As most of them not yet disclosed internals only speculations on their operation scheme can take place. Only few systems such as Universal Plug and Play [3] or DVB-IPTV [15] rely on open standards. These systems will be explained shortly.

Universal Plug and Play (UPnP) refers to a set of standards published by the UPnP Forum. It is primarily designed for usage in small or home networks. The UPnP system is based on IPv4 or IPv6 addressing and it uses the Simple Service Discovery Protocol [16]. Devices and services are described in XML formatted data structures and device functions are controlled by the Simple Object Access Protocol (SOAP) [17]. A subset of standards called UPnP-AV defines different components that can be used to implement IPTV. A server component called MediaServer and a client component called MediaRenderer have to be implemented. The standard defines different methods for the media access based on an URL scheme. The media streams can be encoded in several ways with different codecs depending on the capabilities of the single components. Today many UPnP-AV capable devices are commercially available. Standards like DLNA [18] integrate the UPnP specification which often results in DLNA/UPnP compliant devices.

The DVB-IPTV standard was originally introduced under the name Digital Video Broadcast Internet Protocol Infrastructure (DVB-IPI) in the year 2001 by the DVB

project, an alliance of about 300 worldwide companies. DVB-IPTV covers a set of open and interoperable technical specifications. Its primary goal is the distribution of DVB services within a large-scale IP network. It is based on MPEG2 transport streams carried over IPv4 networks as unicast or multicast in RTSP format. For audio and video the same codecs as in broadcast systems have been selected. The standard also contains mechanisms for service discovery and service selection. Therefore the normal DVB Service Information (DVB-SI [19]) is being used in a XML coded data structure. Interactive content can be transported according to the DVB Multimedia Home Platform (DVB-MHP [20]) specification. Further DVB-IPTV will release a home network specification based on the DLNA/UPnP guidelines. Today operational DVB-IPTV systems can be found rarely as the standard is not yet fully developed. DVB-IPTV has a great potential as it could fully replace traditional television broadcast.

A general drawback of many IPTV architectures has been a high channel switching time. This is caused by different factors such as multicast request forwarding, buffering and decoding delay. Various approaches have been developed to reduce or hide these delays,

(i) Zapping mostly takes place between adjacent channels of a known list. When switching to a channel the client application joins the multicast group of the next channel in the list [21]. If zapping continues the stream of the next channel is already present. As soon as a channel gets watched for a while the multicast group of the next channel is being dropped again.

(ii) Advanced mechanisms use a so-called "GOP server" to enable an instant channel change (ICC) [21]: After selecting a new channel a multicast group gets joined, but also a "GOP server" is contacted via unicast. The server delivers a data burst containing a full group of pictures that enable the client to instantly display the new programe. After the multicast stream gets switched through to the client, the unicast connection is being closed.

(iii) The requirement of a specialized GOP server might reduce availability and cause load peaks within the network. Therefore another solution was found: so-called "secondary channel change stream" is provided for each regular channel [22]. As soon as the client switches to a new channel two corresponding multicast groups are being joined. The first one contains the regular high bandwidth stream, the second is a lower bitrate version carrying out only I-frame and associated audio. The latter enables the client to instantly display the new channel.

2. DVB to IP Network Bridging

The idea of delivering digital television broadcast to IP networks is not new at all. There already exist several implementations that receive, transcode, and distribute TV programes via IP. All these solutions provide random access

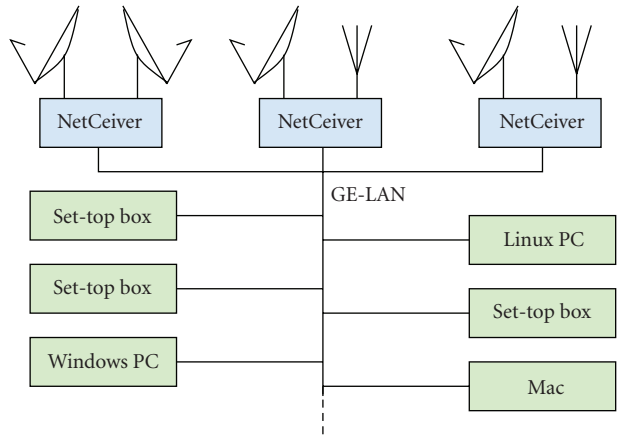

Figure 1: Usage scenario.

to audio and video streams, which enables the user to watch TV on LAN-attached PCs or set-top boxes. But a closer look at these implementations reveals that they do not transparently bridge all digital broadcast features. Therefore only a fixed set of additional services such as the electronic program guide or teletext is converted for IP usage. Furthermore, these systems do not keep the broadcast character of the received streams which means they use unicast transport connections. DVB-IPTV is an exception to this. It reflects the broadcast character of DVB, but it is not yet widely accepted for production use and requires an expensive infrastructure.

2.1. The NetCeiver Architecture. The primary goal of the NetCeiver architecture is to map full DVB services into IPTV without changing their quality, feature set, or broadcast character. It should be possible to access digital television services without prior (network) configuration, proprietary client software, or proprietary set-top boxes. Multiple DVB services, potentially received on different frequencies, should be available for multiple clients in the local IP network. The latter has been achieved in a scalable manner. As depicted in Figure 1, the NetCeiver can be seen as a head end for DVB distribution within IP-based networks.

Regarding our characterization in Section 1, NetCeiver is an infrastructure-based client/server system with a clear role assignment. Interactivity on one hand depends on the features of DVB distribution. NetCeiver does not support them as thoroughly as typical IPTV applications. On the other hand, NetCeiver benefits from all the capabilities of Internet access. The NetCeiver system provides a fully convergent solution in terms of our definition above. There are no restrictions when using DVB services from an IP network.

The NetCeiver architecture is based on highly specialized, cost efficient hardware with low power requirements. It provides slots for DVB tuners, and CAMs and has a Gigabit Ethernet interface (see below). It runs a Linux-based operating system, which allows resource efficient, simple and flexible software development.

A portable library provides IPTV client support for the NetCeiver architecture. It can be used on (embedded) Linux systems, Microsoft Windows, and MacOS. The library is designed to support full DVB access without any prior network configuration or other restrictions. Kernel drivers and daemons for Linux-and-Windows based systems emulate DVB capable hardware. This enables any DVB client software to transparently use the NetCeiver system.

2.2. Software Architecture. As stated above, we designed the NetCeiver system as an IPTV system for DVB services. For this purpose we had to first select a suitable transport protocol. Although many multimedia streaming frameworks already exist, we decided to find a new, more efficient way that meets the following requirements:

(1) transparent DVB access: support for all quality modes, the entire DVB feature set, and low zapping times (below 1-2 seconds),

(2) zero configuration: a self-organizing system that runs without any prior setup,

(3) scalability: multiple servers can serve multiple clients in an efficient way,

(4) low cost solution: server and client are designed to meet low cost requirements.

The rationale for these requirements is based on the way how digital television broadcast systems operate: DVB services are organized as a multiplex called *transport stream* (TS). Each TS contains a number of logical channels, which are identified by a unique 13-bit value, the *Packet Identifier* (PID). The so-called *Program Map Table* (PMT) lists and describes all PIDs that belong to a DVB service. Therefore it is necessary to selectively access single PIDs of a TS. A DVB receiver can bootstrap all available services (PMTs) by accessing the *Program Allocation Table* (PAT). The PAT is always located on PID 0. Studying various IPTV and streaming frameworks we learned that they do not well support this DVB operation scheme because of the following:

(1) none of the established protocols or streaming framework efficiently satisfies the requirement of transparently addressing single PIDs of DVB transport streams,

(2) some existing protocols, like UPnP-AV [3] or Multicast DNS [23], provide zero configuration support, but they are focused on already preassembled services. Implementing them for the NetCeiver system might be possible to some extent, but it would not be fully compatible with the predefined DVB schemes.

(3) the system has to be bandwidth efficient to be scalable: when different clients request the same PID, the corresponding data still should be sent over the network just once. This requirement can only be met by multicast capable streaming protocols,

(4) solutions like DVB-IPTV require a costly server infrastructure, where many different services and protocols have to be supported. On the client side, traditional DVB only set-top boxes need significant software changes; it is not easily possible to just replace the integrated tuner by a network data source.

2.2.1. IPv6 Multicast Based DVB Access. In order to meet these requirements, we designed the NetCeiver to selectively stream single PIDs of a DVB transport stream into a LAN using IPv6 multicast. Each stream is controled by the standard IPv6 multicast listener discovery protocol (MLDv2 [24]). This standard describes how to report multicast groups within a network segment: an IPv6 multicast capable router periodically sends out a multicast listener query to discover the presence of multicast listeners. Every multicast capable network node that has joined or recently left at least one multicast group reports its list of groups to the querier. Furthermore, a multicast node reports joining or leaving groups immediately without a prior listener discovery.

The NetCeiver exploits the MLDv2 protocol by sending out multicast listener queries periodically and receiving all multicast listener reports. This results in an up-to-date list of all joined multicast groups.

Up to this point, MLDv2 only handles the multicast management like IGMP for IPv4. But additionally the NetCeiver itself (ab)uses the very same join message from a client to determine all tuning parameters and sets up the tuner and streaming hardware. So by just blindly joining a not yet existing multicast group, and without other side-band communication, the full stream mechanism comes to life on demand.

To achieve easy scalability and transparency, we chose to broadcast each PID on a different multicast group instead of combining multiple PIDs for one service in one stream. Thus there is no need for managed "channel lists" on the NetCeiver. Also a client can arbitrarily request individual PID streams as with typical PID filtering in set-top box hardware. This eliminates any software changes above the low-level DVB drivers.

For a tune request, the NetCeiver needs to know which PID from which transponder or frequency has to be streamed on a given multicast group. It achieves this by applying an algorithm that transforms specific receiver settings into an IPv6 multicast group and vice versa.

An IPv6 address consists of 128 bits, which are written with hexadecimal characters in eight groups of 16 bits each [25]. Depending on the application there exist different address schemes. This means that not the entire 128 bits address space can be used freely. For example, multicast applications have to set the first 8 bits to 0xff, followed by a 8 bit scope identifier. Only the remaining 112 bits differentiate between the multicast groups.

When a specific PID of a DVB transponder is selected, our algorithm calculates a corresponding 112 bit multicast group address. The so-calculated multicast group uniquely identifies the tuning parameters. Hence, we can also calculate the inverse mapping. Moreover, 128 bits contain enough space to identify all relevant parameters: a delivery system (satellite, terrestrial, etc.), orbital position (for satellites),

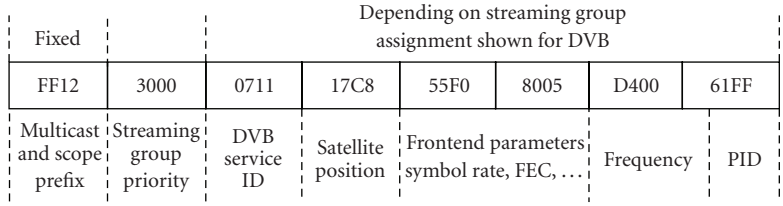

Figure 2: IPv6 multicast group decoding (example for DVB).

frequency, polarization, symbol rate, FEC modes, and PID. Figure 2 gives an example of this scheme. Thus a client tune request is issued just by joining a matching multicast group which contains all relevant information.

2.2.2. Zero Configuration Protocol. So far, we have just presented a transparent multicast-based mapping of DVB parameters. The next requirement deals with the zero configuration feature of the NetCeiver system. To this end, we defined a fixed set of multicast groups. All clients and servers have to join these groups, independent of any DVB streams (cf. Figure 3). All NetCeivers distribute their hardware capabilities and their current tuner allocations on these groups periodically. We use XML coded data structures for this purpose. They comply with the W3C RDF standard CC/PP (Composite Capability/Preference Profiles [26]).

Based on this information, a client can find available NetCeivers and configure itself automatically accordingly. When a client joins a multicast group for receiving a specific PID, it also joins a corresponding group that distributes the signal information from the tuner in charge.

When there are multiple NetCeivers on a network segment, it is also necessary to synchronize all NetCeivers so that they have consistent information about their state. This ensures a proper handling of requests if more than one NetCeiver could serve a multicast group. It also enables resilience in case one NetCeiver fails.

All of these features work in IPv6 without any prior network setup because every IPv6 capable network interface automatically receives a link local address. With such an address it is possible to communicate within a network segment. Unlike DVB-IPTV/UPnP, no DHCP or Auto IP mechanisms for IPv4 are required.

2.2.3. Scalability. The last requirement remaining is scalability. As said above, we wanted to have a resource and bandwidth efficient system that can handle multiple servers and clients. The unique multicast group calculation algorithm prevents the system from duplicating streams for different clients. The self-organization features allow a potentially large number of clients to concurrently feed from one or multiple NetCeivers.

Given the bandwidth limitation of a gigabit network, it is possible to stream a large number of DVB services: Assuming a bandwidth of 8 Mb in H.264 for HDTV and 5 Mb MPEG2 video in SDTV, About 100 HDTV or 160 SDTV programs could be streamed simultaneously. A small number of NetCeivers suffice to satisfy such a high demand: each NetCeiver can stream six full DVB-S2/DVB-C transponders, leading to a maximum bandwidth of about 300 Mb. Given a typical mixture of requests, the number of NetCeivers should not exceed five, unless the switching hardware can snoop MLDv2 and thus limit the traffic to the respective clients.

Currently, there exists a restriction when accessing a NetCeiver beyond router boundaries. Due to the operation scheme of multicast routing [27], it is impossible to propagate join operations for groups under certain conditions: traffic on multicast groups is being generated by the NetCeiver upon an incoming join request of a client. A router cannot determine where to forward this request for a yet inactive group of a specific NetCeiver.

3. Hardware

3.1. Overview. Figure 4 shows the NetCeiver hardware components. The central part is a Xilinx XC3S1600E FPGA [28] with nominally about 1.6 million gates. It acts as a dedicated System-on-Chip (SoC). We decided to use an FPGA because no standard SoC provides the required performance and data handling capabilities. Although, in general, an FPGA is more expensive than a standard SoC, it is nevertheless the key for the NetCeiver's low component cost (less than €50 excluding tuners). Moreover, the dedicated streaming logic also has a very good power efficiency, approximately 4–6 W for the base system and additionally 2–4 W for each S2-tuner. The FPGA does not require a heat sink so that the system does not need a forced cooling. This further increases its overall reliability.

The FPGA interfaces to three exchangeable tuner cards. Each card can deliver the transport stream (TS) of two tuners. Thus a NetCeiver board can use up to 6 DVB tuners in parallel. The overall input bandwidth peaks at about 35 to 40 MByte/s, depending on the tuned transponders.

Two slots for Conditional Access Modules (CAMs) can optionally decrypt the received streams. A flexible TS handling matrix in the FPGA can insert the CAMs into any tuner stream.

The NetCeiver requires only a few active components besides the FPGA: an AVR microcontroller with a MMC/SD-Card interface initializes the SoC with the FPGA bitstream and the uCLinux image. After bootup, the AVR allows simple block device IO to the flash card. It also acts as an IO expander for the GPIOs and the RS232 interfaces, and it helps in monitoring the analog voltages on the board.

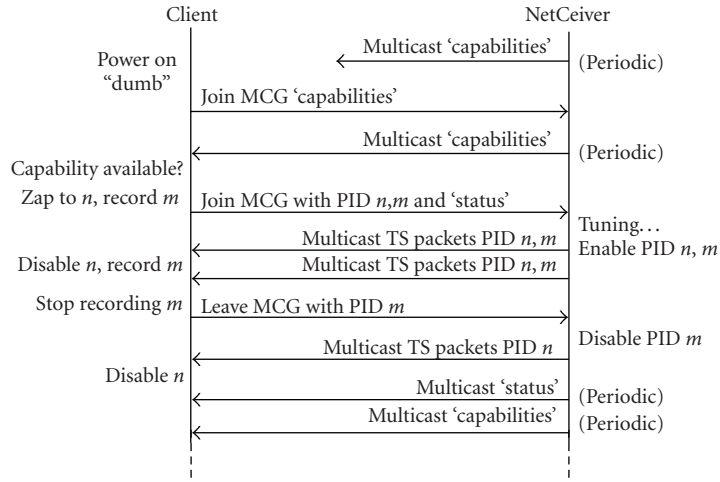

Figure 3: Zero configuration and implicit controlling via join and leave.

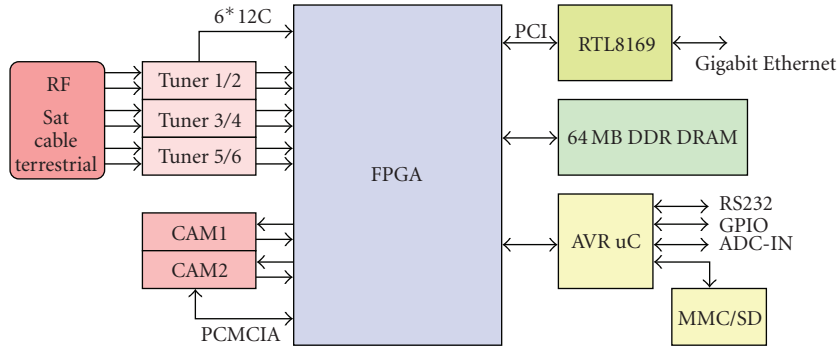

Figure 4: Hardware overview.

The FPGA has access to a 64-MByte DDR-DRAM. One half of the memory is used by the streaming hardware to buffer the TS packets; the other half is dedicated to the uCLinux-OS.

A PCI-connected Realtek 8169 Gigabit Ethernet interface handles the network connection. Even though network chips from Realtek used to have a bad reputation, this no longer true for the newer variants. The RTL8169 provides a high performance (above 100 MBytes/s) for very low cost. This makes it an ideal interface that did not require much development work. Additionally, the chip has two separate transmit queues. This made it easy for us to implement an independent hardware controlled streaming in parallel to the operating system. We explain this in detail later.

A summary of the technical specifications is given in Table 1; the PCB is shown in Figure 5.

3.2. FPGA Data Flow

3.3. Overview. Figure 6 depicts the SoC that is implemented inside the FPGA. It consists of the CPU core, the memory controller, a PCI subsystem, and the TS and network stream processor. Apart from the CPU, which uses IP cores provided by Xilinx in the Embedded Development Kit (EDK), we developed all other blocks specifically for the NetCeiver.

The main CPU block is realized by a Microblaze IP core [29] running at 66 MHz. The CPU core has instruction and data caches with 8 KB each. Although the Microblaze is a full-featured 32-bit RISC CPU and can run the MMU-less uCLinux, it requires only about 10% of the FPGA's resources.

The memory controller interfaces to a single-chip, 16 bit data bus DDR-DRAM clocked at 132 MHz. Internal provisions allow a high performance with a sustained memory bandwidth of about 400 MByte/s for longer burst accesses.

The PCI subsystem implements a 32-bit/33 MHz host bridge (initiator/target) to connect to the RTL8169. This allows us to use the 8169 Linux driver without modifications. The relatively slow CPU allows only about 2-3 MByte/s IP network bandwidth. This is sufficient for control connections, but obviously not for streaming multiple DVB channels. They require a much higher bandwidth.

For this purpose, the PCI core contains a bypass data path. It is enabled when accessing a specific memory area. This area does not map to system memory but to special registers in the network streaming controller. The second transmit queue of the RTL8169 reads on-demand hardware-generated DMA descriptors and packet data over this bypass.

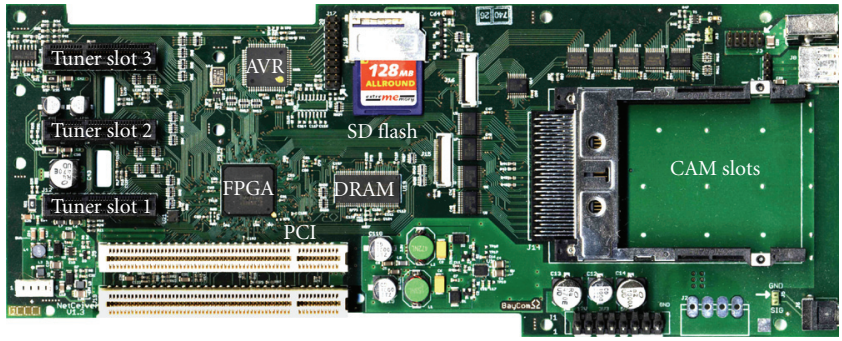

Figure 5: PCB of the NetCeiver.

Table 1: Technical specifications for the first NetCeiver HW implementation.

Tuners	6 TS inputs (each up to 9 MByte/s)
CAMs	2 (2 TS in, 2 TS out)
Network	RTL8169 Gigabit Ethernet
FPGA	XC3S1600
Internal clocks	66 MHz, except 33 MHz PCI, 132 MHz DDR & TS matrix
Memory	64 MB DDR-DRAM, 128 MB Flash
Power consumption	4–6 W (excl. tuners and CAMs)
OS	uCLinux
Boot time	Approx. 25s (depending on tuners)
Dimensions	Approx. 28 cm∗12 cm (16 cm∗12 cm w/o CAMs)
Height	8 cm (incl. network and tuner cards)
PCB	4 layer
Component cost	< €50 (at mid volume quantities, excl. tuners)

This fully offloads the CPU from the packet generation process for streaming. A benchmark using synthetic data packets showed that the RTL8169 and the PCI-bypass achieved a transmit rate of about 100 MB/s, which is about 80% of a Gigabit link capacity. This is fully sufficient for our purpose, even when the clients request six full DVB transponders.

On the input side, the TS data coming from the six tuners is routed through a matrix, which also allows to flexibly insert the decrypting CAMs into the stream. Afterwards, a table lookup that is based on the packet ID (PID) determines whether at least one client requires this PID at all. This *PID-info table* then also determines which way the packet should take. For the main data path network streaming FIFO areas (1 MB per tuner) in system memory buffer the packets. In total, there are 31 FIFO destinations possible for the PID-info. They are also used internally for monitoring and service information parsing.

One key element of the NetCeiver multicast protocol is that each PID is sent over a separate multicast address. Unfortunately, DVB delivers the PIDs in a randomly multiplexed order. They are not sorted to form bursts of one PID that could fill up the maximum transfer unit of a network packet. Thus we need to store and aggregate the packets. For this purpose, we scan 32 successive packets in the FIFO and sort them according to the PID. This allows us—at least in the case of the high bandwidth video PIDs—to fully utilize the MTU. This ensures the desired high throughput. Typically, a packet contains 1378 bytes, 62 bytes in the headers (Ethernet, IPv6 and UDP) and 1316 bytes in seven TS packets.

The network data generation unit then assembles the full network packet from such aggregated TS packets. The basic IPv6 multicast streaming header is based on templates that the software in the CPU core fills in advance. When the network chip accesses the header area over the bypass path, the data is finalized on the fly while being transferred. That means that the transfer unit inserts header fields such as the payload length or the destination IPv6 address into the template placeholders. After the header data it reads the packet data as is from the respective packet buffer.

Besides supporting IPv6 multicast, we made provisions in the PID-info table to enable IPv4 multicast and unicast, too. A unique ID, which can be propagated to the data generation engine, selects a linked list of individual connection data such as a destination MAC, an IPv4 destination address, RTP sequence counters, etc. This allows us to implicitly multiplex different PIDs to one destination address. As a result, NetCeiver also supports traditional streaming protocols and clients in hardware.

All the SoC components, the Microblaze CPU, the memory controller, the PCI module, the transport stream handler, and the network interface occupy only about 50%

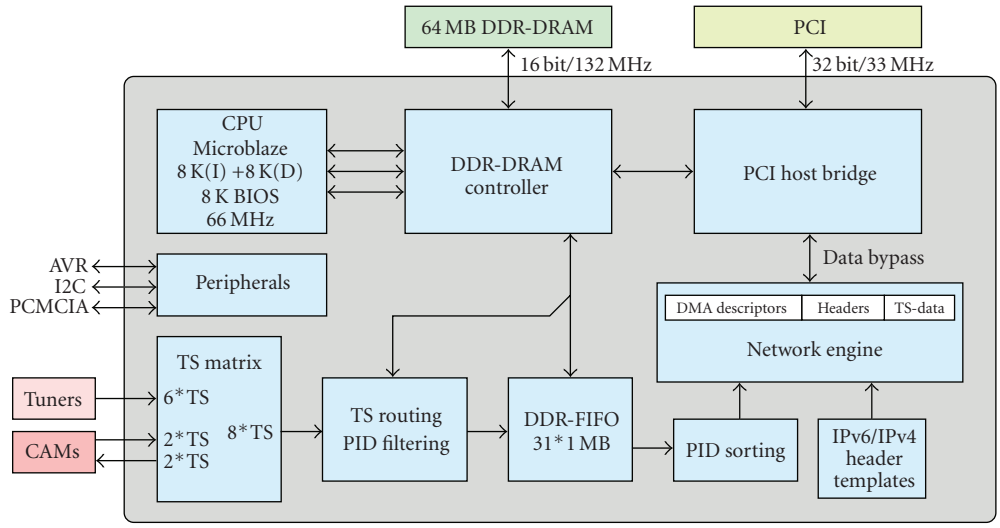

Figure 6: Internal components of the FPGA.

of the FPGA so far. This leaves plenty resources for future extensions.

4. Performance

The described TS handling and network packet generation runs in parallel to the CPU without any CPU intervention. The CPU only configures the IP header templates and updates the PID-info tables for each started or stopped streaming request. Our tests confirmed that the hardware is actually capable of streaming the full content of six tuner transport streams (the aforementioned 35–40 MB/s) without lost packets, although this scenario with all PIDs enabled is unlikely to occur in real life conditions. It merely demonstrates that at least the hardware data processing poses no performance bottlenecks.

For an interactive service, the zapping latency is important. Since we wanted no additional infrastructure like the mentioned GOP-server, our goal was not to differ too much from a locally connected tuner. This aspect is in our opinion very well handled by the dual use of MLDv2 for multicast management *and* tuning requests, as the binary protocol requires only little CPU resources for decoding and generation.

We have measured the time difference between issuing a tune request on a client and the arrival of the first transport stream packets. With 3 clients continuously tuning to random services with multiple PIDs on random DVB-S/S2 transponders, the latency varies between 70 ms and about 2 s with an average of about 0.5 s. The lower zapping times occur when a tuner already has a lock and only the PID filtering needs to be modified. The average latency is mainly determined by the tuning itself and includes the time to acquire a lock (50–200 ms) and the DiseqC-communication to the LNB (about 100 ms). Zapping times over 1s happen only with a probability of less than 10% and are mainly caused by missing a multicast listener report at the server which is sent by the client every second.

Since the NetCeiver uses XML to broadcast its capabilities and internal status, an exemplary experience for XML-handling on slow CPUs was also obtained. The internal status including all tuner parameters and satellite lists can be up to 20 KB in uncompressed XML. This size is caused by the CC/PP syntax with long descriptive tags. This report is generated every 10 s by using the popular libxml2-library facilities. The generation of this document needs about 7 s on the Microblaze. Further analysis showed that about 90% of the time is consumed by the malloc/free memory management of uCLib for the various XML result nodes. Based on this surprising result, the generation frequency was drastically reduced. Although solvable, this example shows that uncritical XML handling with standard frameworks can have a large performance impact on slow CPUs.

5. Conclusion and Outlook

In this paper we have presented the NetCeiver architecture for distributing DVB-like broadcast over IP networks. NetCeiver uses standard protocols such as IPv6 multicast with MLDv2. Our main design goals were zero configuration, scalable streaming to an arbitrary number of clients, and full service transparency for typical DVB receivers. Especially the zero configuration property was considered to be important for home and other non-administered environments.

We have developed a cost-efficient FPGA based hardware that implements our NetCeiver architecture. It allows easily deployment in the consumer market. Moreover, our design is power efficient and allows a very high network performance. We achieve this with a specialized stream handling inside the FPGA.

The NetCeiver architecture is available through a commercial vendor since late 2007. The customer feedback shows that it meets its design goals. The NetCeiver product works as a detachable DVB tuner component of a high-end set-top box. It is also available as a stand-alone head end for inexpensive and lightweight STB clients.

Yet, the development is not finished. In the future, we will extend the NetCeiver core functionality by VOD-services and add multicast security features. Although not as lightweight as the NetCeiver multicast protocol, industry standards like UPnP-AV and the related DVB-IPTV exist and should not be neglected. We plan to integrate the NetCeiver hardware into these frameworks.

Although DVB-IPTV is currently getting more attention, we think that our combined HW/SW design with a non-conventional interpretation of a low-level network protocol can contribute to the still heterogenic IPTV market, as it allows a low- and mid-scale deployment at a very low cost and almost no software changes in a typical DVB set-top box client.

References

[1] IETF, "Real Time Streaming Protocol (RTSP), RFC2326," April 1998, http://tools.ietf.org/html/rfc2326.

[2] IETF, "SDP: Session Description Protocol, RFC2327," April 1998, http://tools.ietf.org/html/rfc2327.

[3] UPnP Forum, "UPnP AV Architecture," December 2008, http://www.upnp.org/standardizeddcps/default.asp.

[4] Wikimedia Foundation, "Wikipedia, P2PTV," January 2009, http://en.wikipedia.org/wiki/P2PTV.

[5] T. Silverston and O. Fourmaux, "P2P IPTV measurement: a comparison study," 2006, http://arxiv.org/pdf/cs/0610133.pdf.

[6] Y.-F. Chen, Y. Huang, R. Jana, et al., "When is P2P technology beneficial for IPTV services," 2007, http://www.nossdav.org/2007/files/file-22-session4-paper1-chen.pdf.

[7] ETSI, "DVB standards," January 2009, http://www.etsi.org/WebSite/Technologies/DVB.aspx.

[8] The advanced television systems committee, "ATSC Standards," January 2009, http://www.atsc.org/standards.

[9] Digital broadcasting experts group, "ISDB-T System," 2008, http://www.dibeg.org.

[10] ETSI, "DAB standards," January 2009, http://www.etsi.org/WebSite/Technologies/DAB.aspx.

[11] Wikipedia, "Mbone," August 2009, http://en.wikipedia.org/wiki/Mbone.

[12] RFC Draft, "Session Announcement Protocol, RFC2974," October 2000, http://tools.ietf.org/html/rfc2974.

[13] RFC Draft, "RTP: a transport protocol for Real-Time Applications, RFC3550," November 1995, http://tools.ietf.org/html/rfc3550.

[14] D. Minoli, *IP Multicast with Applications to IPTV and Mobile DVB-H*, John Wiley & Sons, New York, NY, USA, 2008.

[15] DVB project, "DVB-IPTV 1.4: Transport of MPEG 2 TS Based DVB Services over IP Based Networks (and associated XML) (dTS 102 034 V1.4.1)," January 2009, http://dvb.org/technology/standards/index.xml#internet.

[16] Internet Draft, "Simple Service Discovery Protocol," October 1999, http://tools.ietf.org/html/draft-cai-ssdp-v1-03.

[17] W3C, "SOAP Version 1.2," April 2007, http://www.w3.org/TR/soap12-part1.

[18] Digital Living Network Alliance, "DLNA," September 2009, http://www.dlna.org.

[19] DVB Project, "Specification for service information (SI) in DVB systems (ETSI EN 300 468)," July 2009, http://dvb.org/technology/standards/index.xml#multiplexing.

[20] DVB Project, "MHP 1.2," July 2009, http://www.mhp.org/mhpgem12.htm.

[21] T. Janevski and Z. Vanevski, "Statistical analysis of multicast versus instant channel changing unicast IPTV provisioning," in *Proceedings of the 16th Telecommunications Forum (TELFOR '08)*, Belgrade, Serbia, November 2008.

[22] D. Banodkar, K. K. Ramakrishnan, S. Kalyanaraman, A. Gerber, and O. Spatscheck, "Multicast instant channel change in IPTV systems," in *Proceedings of the 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08)*, pp. 6–10, Bangalore, India, January 2008.

[23] RFC draft, "Multicast DNS," September 2008, http://files.multicastdns.org/draft-cheshire-dnsext-multicastdns.txt.

[24] IETF, "Multicast Listener Discovery Version 2 (MLDv2) for IPv6, RFC3810," June 2004, http://tools.ietf.org/html/rfc3810.

[25] IETF, "IP Version 6 Addressing Architecture, RFC4291," February 2006, http://tools.ietf.org/html/rfc4291.

[26] W3C, "Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0," January 2004, http://www.w3.org/TR/CCPP-struct-vocab.

[27] IETF, "Protocol Independent Multicast—Sparse Mode (PIM-SM), RFC2362," June 1998, http://tools.ietf.org/html/rfc2362.

[28] Xilinx Inc., "Spartan-3E FPGA family data sheet," http://www.xilinx.com/support/documentation/index.htm.

[29] Xilinx Inc., "MicroBlaze processor," http://www.xilinx.com/tools/microblaze.htm.